JURI DISPAN

CONCEPT AND VALIDATION OF AN ANALYTICS DRIVEN SECURITY INFORMATION AND EVENT MANAGEMENT (SIEM) FOR HIGH PERFORMANCE COMPUTING SYSTEMS

universität ulm

CONCEPT AND VALIDATION OF AN ANALYTICS DRIVEN SECURITY INFORMATION AND EVENT MANAGEMENT (SIEM) FOR HIGH PERFORMANCE COMPUTING SYSTEMS

JURI DISPAN

Master's Thesis

Institute of Information Resource Management
Faculty of Engineering, Computer Science and Psychology
Ulm University

July 2022

Prof. Dr.-Ing. Dr. h.c. Stefan Wesner
Prof. Dr. Hans Kestler

Juri Dispan: *Concept and Validation of an Analytics Driven Security Information and Event Management (SIEM) for High Performance Computing Systems,* Master's Thesis, © July 2022

ABSTRACT

High-Performance computing (HPC) systems are a high-value target for cyber-attacks. However, detecting security breaches before attackers can do catastrophic damage remains a largely unsolved issue. In this thesis, we propose a *Security Information and Event Management* (SIEM) capable of identifying malicious user behaviour as it occurs. This SIEM uses unsupervised machine learning to process log data and alerts administrators when indicators for malicious behaviour are discovered. We perform an experimental evaluation of parts of the system in which we select the most suitable algorithms for anomaly detection and report the performance in terms of accuracy, recall and F1-score. Our results suggest that different sources of log data available in HPC systems can successfully be combined to detect malicious behaviour. Identifying anomalous logins and anomalous job submissions can be performed with high reliability (reaching an F1-score of 0.944 and 0.800 respectively). Because of a lack in real-world test data, the viability of detecting anomalous behaviour through programme execution paths cannot be determined in a meaningful way. We further explore the possibility of detecting anomalous user movement inside HPC clusters conceptually and leave experimental evaluation thereof for future work.

ACKNOWLEDGMENTS

The author acknowledges support by the state of Baden-Württemberg through bwHPC and the German Research Foundation (DFG) through grant No INST 40/575-1 FUGG (JUSTUS 2 cluster).

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

1 INTRODUCTION

1.1 MOTIVATION

This thesis is motivated by a security incident lasting from December 2019 until May 2020. Unknown attackers gained access to numerous high-performance computing (HPC) centres in Europe. The goal of these attacks is still not clear, speculations include theft of research results or crypto-currency mining. These attacks are interesting, because they went undetected for several months. It was only in May 2020 that the University of Edinburgh gained knowledge of the breach. As a result, other HPC centres started investigating their systems for similar attacks and indeed many were found. The affected clusters were immediately taken offline for investigation of the security breach, impeding academic research for weeks [8, 32].

This attack is not surprising, because HPC systems are high-value targets for hackers. They are designed to offer tremendous computing resources to researchers for performing computationally demanding tasks such as weather forecasting, the search for a vaccine against COVID-19 or climate research. Their power makes them a valuable tool for the scientific community but also for cybercriminals, who might want to use their computational capabilities to mine crypto-currencies, steal research results or even weaponise the HPC system itself to perform large-scale denial of service attacks. Because such abusive behaviour hinders research by wasting CPU-cycles, HPC system operators try to take measures to prevent or detect attacks early on.

Preventing attacks turns out to be quite difficult, because security has to be weighted against other objectives of HPC operations [8]. HPC systems are expensive, both in acquisition and in operation. A high degree of utilisation is required to make the system worth its costs. In order to achieve high utilisation, the system must be available to eligible researchers, which in turn often means exposing it to the internet and potentially malicious actors. Furthermore, HPC clusters typically do not receive security updates immediately. Each update bears the risk of introducing incompatibilities between different software modules into the system. Patches need to be tested and verified to interact flawlessly with existing software before they can be applied. In order to perform software upgrades, it is often necessary to cease normal HPC operations. In order to keep downtime time low, patches are often delayed such that multiple updates can be bundled and performed together. The attack-surface of HPC systems is thus far

from zero. Combined with the amount of criminal energy focused on gaining access to such systems, it is only a matter of time until the next attack on a HPC centre is successful.

As perfect security cannot be guaranteed in HPC systems, it is desirable to at least detect attacks early on or even in real-time. If operators gain knowledge of a breach, they can initiate countermeasures and stop possible misuse of the system, minimising the attack's impact on research.

1.2 AIM OF THIS WORK & RESEARCH QUESTIONS

In this work, we investigate the feasibility of analysing system logs for detecting malicious behaviour in HPC systems. Logs are commonly used for debugging purposes, therefore many systems already collect and store logs during operation. As we describe in Section 2.1, certain kinds of system misuse produce anomalies in logs. Thus, anomalies found in log files can be an indicator of ongoing attacks on the system. Traditionally, operators of large-scale computer systems have resorted to manual inspection of log files in order to detect anomalous system behaviour. This approach is often performed by searching for certain keywords (e.g. *ERROR* or *WARNING*) or regular expressions, or by using domain-specific knowledge to implement rule-based anomaly detection. However, as the size of computer systems grows, the amount of logs reaches levels that cannot be dealt with manually. Further, user behaviour can change over time. The introduction of new programmes to computing infrastructure can introduce previously unseen log messages, requiring constant maintenance of regular expressions and rules used for anomaly detection. This places a considerable load on system operators [26].

In order to reduce manual labour, it is desirable to have logs of a running system inspected automatically by a computer programme, alerting the human operators when anomalies are detected. This thesis' aim is to develop a concept for such a *Security and Information Management* (SIEM) system. We formulate the following requirements that we aim to meet with the SIEM we propose:

- The system should be able to combine different sources of logs in order to detect malicious behaviour.
- The system should work in a streaming manner, i.e. consume logs as they are produced.
- The system should be efficient enough to cope with the large amounts of logs produced by modern HPC systems. It should be able to classify events as either normal or anomalous in a timely manner, ideally in real-time.

- The system should be able to adapt to changing user behaviour, i.e. operators should be able to tag false positives as such, the system should consequently no longer classify similar behaviours as malicious.

We implement parts of this system and evaluate its effectiveness on the JUSTUS2 HPC cluster located at Ulm University.

The thesis poses the following research questions:

- Can we reliably classify user behaviour as (non-)malicious?
- What algorithms are suitable for this classification?
- What are the computational requirements for classification?
- What information is required to improve classification reliability?

1.3 THESIS STRUCTURE

In this Section, we outline the structure of this thesis. Chapter 2 provides information about cyberattacks on HPC infrastructure and the sources of log data available in a typical HPC system. In Chapter 3, we discuss existing work related to log-based anomaly detection. We propose a concept for a SIEM suited for HPC systems in Chapter 4. For that, we first introduce the general architecture in Section 4.1. Afterwards, we discuss the individual elements of this architecture. The rather technical steps of preprocessing and session grouping are discussed in Section 4.2 and Section 4.3. We propose four different methods to detect anomalous behaviour in Section 4.4 and a method to combine the results of these methods in Section 4.5. Section 4.6 proposes a mechanism for the system to adapt to changing user behaviour.

Chapter 5 describes our methodology for evaluating the performance of various elements of the SIEM we propose. In Chapter 6, we give details on our implementation of the proposed concept. Chapter 7 contains results of our experiments. We discuss these results in Chapter 8 and conclude the thesis in Chapter 9.

2 FUNDAMENTALS

In this Section, we cover fundamental knowledge easing the understanding of this thesis.

2.1 CYBERATTACKS ON HPC INFRASTRUCTURE

In order to gain an understanding of typical attacks on HPC infrastructure, we had conversations with operators of HPC systems affected by the security incident described in Section 1.1. The information acquired through these conversations combined with descriptions of security incidents in literature helps us identify a typical sequence of events that is observed during attacks on HPC systems [8].

An attack on a HPC system can typically be separated into four phases:

1. Establishing a foothold: In this phase, attackers gain access to the targeted HPC system. This is often achieved through stolen user credentials, which allow the attackers to login on the HPC system as a valid user without special permissions.

2. Lateral movement: A HPC system typically consists of many interconnected nodes. During the lateral movement phase, attackers try to move around between these nodes, exploring the structure of the system and looking for valuable information (credentials, research results) or vulnerable software that can be used for gaining privileged access to the system.

3. Privilege escalation: Depending on their goals, attackers may attempt to gain privileged access to the system. If credentials of privileged users were found during Phase 2, attackers can use them to gain elevated permissions. If available, attackers can exploit flaws in installed software to gain additional privileges.

4. Achieving the objective: When the attackers have acquired sufficient privileges, they are free to achieve their end goal, e.g. misusing the HPC system's capacity to mine crypto-currency, steal research results, steal user credentials for subsequent attacks or install ransomware.

Note that, depending on the attackers' background and goal, some phases may be skipped. For example, a legitimate researcher on a HPC system attempting to misuse the system's computing power to

mine crypto-currency can skip Phase 1, Phase 2 and Phase 3 because as an insider, they already possess the privileges required to submit computing jobs.

All of the phases of an attack leave traces in log files. Login attempts (Phase 1) and movement inside the cluster (Phase 2) are logged to `/var/log/secure`. Attempts to exploit vulnerable software (Phase 3) often leave anomalous log messages in `/var/log/messages`. Compute jobs submitted to misuse the cluster's capacity (Phase 4) are logged in the system's workload manager (WM). In this work, we leverage the traces left by attackers in order to detect anomalous behaviour as it is happening. The sources of log data available for achieving this goal are described in further detail in Section 2.2.

2.2 LOG DATA

In this Section, we describe the sources of log data we use to detect malicious behaviour in HPC systems. This includes the system's WM (Section 2.2.1) and the two files `/var/log/messages` and `/var/log/secure` (Section 2.2.2).

2.2.1 Workload Manager

HPC facilities typically feature a WM, which is used for submitting compute jobs. This WM stores diverse information about submitted jobs, e.g. the resources requested by the user, the resources actually used by the job, the exit code of the job etc. The exact information stored depends on the WM used. Popular choices include Slurm[1], Grid Engine[2] and HTCondor[3].

2.2.2 /var/log/messages and /var/log/secure

Many HPC systems use *syslog*[4] for aggregating system log messages. These system logs are stored in two locations: `/var/log/messages` contains all log messages except the ones related to user authentication, which are instead saved to `/var/log/secure`.

System log messages are comprised of several fields. Consider the log message below, which is taken from `/var/log/messages` on a HPC system running a Linux-based operating system:

```
Jan 1 23:33:37 n0111 slurmd[2847]:
    Launching batch job 4656494 for UID 939381
```

[1] https://slurm.schedmd.com/overview.html
[2] https://www.altair.com/grid-engine/
[3] https://htcondor.org/
[4] https://datatracker.ietf.org/wg/syslog/documents/

This log message includes the fields *timestamp*, *machine*, *process name*, *process id* and *raw message content*. The first four fields contain metadata about the log message and are in a fixed format. *Raw message content* on the other hand contains free text emitted by running programmes, providing rich information about their internal state. The challenge of parsing these free text messages is tackled in Section 3.1.1.

`/var/log/secure` follows the same format as `/var/log/messages`. It contains log messages related to authentication, i.e. successful and unsuccessful authentication attempts, the opening and closing of user sessions and the creation of new users or groups.

3 RELATED WORK

In Chapter 4, we propose an architecture for a SIEM aimed at detecting malicious behaviour in HPC systems. This architecture integrates algorithms developed in previous works, which we introduce in this Chapter. Section 3.1 gives an overview over existing approaches for detecting anomalous behaviour through the analysis of system logs. A prerequisite for such an analysis is *log parsing*, which is covered in Section 3.1.1. Four techniques for performing anomaly detection though system logs are laid out in Section 3.1.2.

The SIEM we propose makes heavy use of algorithms for *novelty detection*. In Section 3.2, we introduce four such algorithms, namely Isolation Forest, One-class Support Vector Machine, One-class Support Vector Machine with Stochastic Gradient Descent, and Local Outlier Factor.

3.1 ANOMALY DETECTION USING SYSTEM LOGS

Anomaly detection in HPC systems using system logs is a well-studied field [1, 3–5, 9, 11, 14, 15, 26, 35, 37, 39, 40, 44, 59, 69, 73]. However, many works that investigate anomaly detection through system logs do not focus on the discovery of security incidents, but rather on the detection of anomalous system behaviour in general (e.g. hardware faults or software crashes). Nevertheless, some of these approaches can also be used to detect anomalies related to security incidents and are, among others, discussed in this Section. Going forward, we use the term *anomalous behaviour* in the sense of *behaviour potentially indicating malicious intent*.

Usually, approaches for anomaly detection in HPC systems using system logs consist of four steps: log collection, log parsing, feature extraction and anomaly detection [9, 26, 39]. This pipeline is visualised in Figure 3.1.

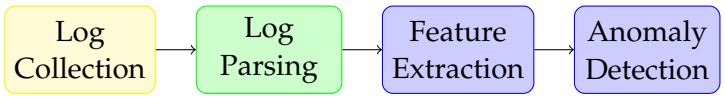


Figure 3.1: Anomaly detection in HPC systems is usually done by performing four steps: log collection, log parsing, feature extraction and anomaly detection.

Log collection is a prerequisite for log analysis. Systems can perform logging at different levels, e.g. hardware, kernel or application level. The exact amount of logging performed in a HPC system must be carefully balanced by the system's operators. More logging means more potentially helpful information for anomaly detection or system debugging, but also increases runtime overhead and storage requirements. We consider the problem of log collection to be out of scope for this work. The SIEM we propose only requires logs from three sources (`/var/log/secure`, `/var/log/messages` and the workload manager), which are available on most HPC systems by default.

Log parsing describes the process of transforming unstructured system logs into a structured format. Many attempts have been made to automate this transformation. We review some of these attempts in Section 3.1.1. This step is only necessary if the log data at hand are in an unstructured format. If it is structured (as in the case of logs from a HPC workload manager), it can be skipped.

The steps of feature extraction and anomaly detection are often performed together. We review approaches for these steps in Section 3.1.2.

3.1.1 Log Parsing

A *log* is a sequence of log messages. A *log message* is a line of text emitted by a system during run-time that describes the system's internal state at a given point in time. We describe the structure of system log messages at our disposal in Section 2.2.2. These log messages are only partially structured, because their field *raw message content* contains unstructured data and all other fields contain structured data. The text found in *raw message content* is produced by logging statements (e.g. `printf(·)`, `logger.debug(·)`) inside running programmes. Application developers face no restrictions writing these logging statements; any valid string is also a valid log message and can appear in logs. This flexibility means that we often find rich information regarding the internal state of running programmes in this field. On the other hand, it means that *raw message content*'s format can differ widely across applications or even different parts of the same application. Because most automated approaches for detecting anomalies from logs require structured input, it is necessary to transform these unstructured logs into a sequence of structured events [24].

The *raw message content* field of a log message can usually be split into two parts: a *constant* part and a *variable* part. The constant part remains constant across different calls to a logging statement, while the variable part can vary, as it is created dynamically. Parsing a *raw message content* means separating the constant part from the variable part. The exemplary *raw message content* given in Section 2.2.2 has a constant part of `Launching batch job * for UID *` and a variable

part of `4656494`, `939381`. We call the constant part a *log event* or *log key* and the variable parts the *parameters* of this log event.

The traditional approach for parsing log messages is using regular expressions. However, this is rather cumbersome, because it requires writing one regular expression for each possible log event. Since new log events can be introduced by software updates or users running previously unseen programmes, it also requires constant maintenance as well as deep domain knowledge. For large multi-user systems, a manual approach is clearly not feasible [14, 24, 26].

Lots of efforts have been made to automate log parsing. A recent study by Zhu et al. [76] compares 13 methods for automatic log parsing in terms of efficiency, mode (online vs. offline), and accuracy on 16 different datasets. These methods are *SLCT* [66], *AEL* [29, 30], *IPLoM* [42, 43], *LKE* [17], *LFA* [49], *LogSig* [64], *SHISO* [47], *LogCluster* [67], *LenMa* [60], *LogMine* [21], *Spell* [13], *Drain* [25] and *MoLFI* [45]. Additionally, the authors provide *logparser*[1], which is a software bundle containing open-source implementations of the evaluated algorithms, the datasets used for evaluation and a framework for benchmarking automatic log parsers. We perform an evaluation of the parsers studied by Zhu et al. in Section 7.1.

3.1.2 Anomaly Detection

In this section, we review methods for anomaly detection in system logs. We only consider unsupervised detection methods because labelled log data are hard to acquire.

The first method we review is DeepLog by Du et al. [14], an approach for detecting anomalous system logs using neural networks.

We also review LogCluster, Principal Component Analysis (PCA) and Invariants Mining, which are three approaches for anomaly detection in system logs using classical machine learning algorithms.

3.1.2.1 DeepLog

DeepLog [14] is a deep neural network architecture. It operates on logs and is designed to detect anomalous behaviour. It attempts to learn the normal behaviour of the system under observation during its training phase. In the detection phase, the system is monitored for deviations from the patterns that constitute normal behaviour.

DeepLog enjoys three properties that make it very attractive to us:

- It supports online training: if session is flagged as anomalous but manually marked as normal afterwards, the model can be adjusted on-the-fly.
- Computational cost in the detection phase is low.

[1] https://github.com/logpai/logparser

- It can operate in a streaming manner.

DeepLog detects anomalous system states in two different ways. *Execution path anomaly detection* purely focuses on the log keys encountered in the log file, discarding the corresponding parameters. The idea is to build a language model that, given a sequence of h log keys, predicts the most probable subsequent log keys. More formally, the model calculates $P(m_t = k_i | (m_{t-h}, m_{t-h+1}, \ldots, m_{t-1}))$, i.e. the probability that log key k_i follows a sequence $(m_{t-h}, m_{t-h+1}, \ldots, m_{t-1})$ of log keys. If the observed next log key m_t is not one of the g most probable next log keys, it is considered anomalous. Du et al. use a neural network architecture called *long short-term memory* (LSTM, [27]) for this task.

Parameter value and performance anomaly detection on the other hand disregards the sequence of log keys and focuses on detecting unusual parameters of log events or suspicious performance degradations. A log message's parameter values can be represented as a vector. The parameter value vectors for a given log key form a parameter value vector sequence. Combined with the parameter value vectors' timestamps, the vector sequence forms a multivariate time-series. DeepLog uses a LSTM network to predict the next value vector in this time-series. If the observed next parameter value vector differs from the predicted one by more than a specified threshold, the corresponding user session is marked as anomalous.

DeepLog can be tuned by adjusting four main hyper-parameters. h is the window size to be used for execution path anomaly detection. Given a log sequence of size h, the model makes a prediction on the g most likely subsequent log keys. If the observed log key is not one of the predicted keys, the model considers it anomalous. The layout of the LSTM networks to be used is dictated by the number of hidden layers L and the number of memory units per layer α.

DeepLog supports online updates of its anomaly detection models. If the system marks a normal sample as anomalous, users can flag this sample as a false positive. DeepLog is then able to adjust the weights in its LSTM networks. This adjustment can be performed on-the-fly without retraining the system, allowing DeepLog to adapt to changing operating conditions during runtime.

Du et al. [14] report DeepLog's performance using the metrics of precision, recall and F1-score. For evaluation, they use two datasets: one dataset consisting of Hadoop Distributed File System (HDFS) logs and one containing OpenStack logs. These datasets are labelled by domain experts and contain 28 (HDFS) and 40 (OpenStack) different log keys. On the HDFS dataset, DeepLog achieves a precision of 0.95, a recall of 0.96 and an F1-score of 0.96. The results on the OpenStack dataset are similarly promising: precision is 0.96, recall is 1.00 and F1-score is 0.98.

3.1.2.2 LogCluster

LogCluster (Lin et al., [35]) performs anomaly detection by clustering. Logs are assigned into log sequences, which contain all logs produced in one session. Logs sequences are vectorised and the resulting vectors are clustered using Agglomerative Hierarchical Clustering [19] under the constraint that the maximum cosine distance within a cluster is at most θ. The authors propose to determine a good value for θ empirically depending on the specific use case.

In order to minimise the memory footprint, the algorithm does not store the vectorised logs after clustering has taken place. Instead, each cluster is represented by its centroid.

New log sequences are classified as normal or anomalous by computing the minimum cosine distance of the vectorised sequence to the cluster centroids. If it is larger than a threshold δ, the sequence is classified as anomalous.

He et al. [26] evaluate the method's performance using a HDFS logs and system logs from BlueGene/L (BGL), which is a type of supercomputer. On HDFS logs, LogCluster achieves (Precision/Recall/F1-score) of (0.87/0.74/0.80), whereas it achieves (0.42/0.87/0.57) on BGL logs.

3.1.2.3 PCA

Xu et al. [73] propose to construct an event count vector of dimension n, summarising each log sequence. PCA is then used to compute a normal space S_n (using the first k principal components) and an anomalous space S_a (using the remaining $n - k$ principal components). Sequences are classified by projecting their event count vector to S_a and measuring its length. If the length exceeds a threshold, the sequence is considered anomalous. This threshold can be calculated automatically such that predictions are made with a confidence level of $(1 - \beta)$ [26].

He et al. [26] report the systems performance (Precision/Recall/F1-score) as (0.98/0.67/0.79) on HDFS log data and as (0.50/0.61/0.55) on BGL log data.

3.1.2.4 Invariants Mining

Invariants are linear relationships that always hold during normal operations in a system. For example, one invariant in computer systems is that a session which has opened a file should always close this file at some point, i.e. the number of logged *open file x* events should equal the number of *close file x* events. If this invariant is violated, i.e. the number of open and close events differs, a session can be considered anomalous.

Lou et al. [37] propose a method to extract such relationships automatically from log files. Essentially, they use a brute-force search

to uncover these invariants. New log sequences are then checked on whether or not they violate one of these invariants. If they do, they are tagged as anomalous. The algorithm by Lou et al. [37] can be tuned with two parameters. The support ratio s describes how many log sequences must adhere to an invariant for the it to be considered valid. The parameter ϵ is a threshold which describes the degree to which a log sequence can violate an invariant and still be considered to adhere to it.

He et al. [26] report the method's (Precision/Recall/F1-score) metrics as (0.88/0.95/0.91) on a HDFS dataset and as (0.83/0.99/0.91) on a BGL dataset.

3.2 NOVELTY DETECTION

Novelty Detection is the process of identifying data points that differ from a given dataset in some way. It requires the availability of a dataset containing many normal data points and few abnormal data points. This training dataset is used to build a model, which can then be used to classify test data as normal or abnormal [53].

In the scope of this thesis, we deal with data collected during day-to-day operations of HPC systems. We previously defined anomalous behaviour as malicious acts, e.g. use by unauthorised users or attempts to misuse computing resources. As such attacks on HPC systems are relatively rare, the vast majority of these data describe normal operating conditions. This leaves us with a dataset with many normal examples, but few anomalous ones. Further, some anomalous modes of operation are so rare that they might not be present in the available dataset. Therefore, we cannot assume that all types of anomaly follow a known pattern. That requires us to perform anomaly detection in a way that leaves room for recognising previously unseen anomalies. For this reason, we make heavy use of methods for novelty detection in this work. Each utilised method is summarised below.

3.2.1 Isolation Forest

Isolation Forests [36] are a method for novelty detection that is based on measuring the number of steps necessary to isolate a data point from all other data points [74].

An Isolation Forest is a set of t Isolation Trees. An Isolation Tree is basically a decision tree. Its inner nodes contain a test, comparing one attribute of a data point to a threshold.

An Isolation Tree is constructed by recursively splitting an input dataset X. If the tree has reached a height limit, $|X| = 1$ or all elements in X are equal, the recursion is ended and a leaf node is constructed. Otherwise, a test node is constructed by choosing at random an at-

tribute q and a threshold p. Elements of X where attribute q is less than p are assigned to the left child node, whereas all others are assigned to the right one. An Isolation Tree assesses the anomaly of a given data point x by measuring the depth of the leaf node corresponding to x. Accordingly, an Isolation Forest measures the anomaly of a given data point by averaging over the anomaly scores given by its Isolation Trees.

3.2.2 One-Class Support Vector Machine

Support Vector Machines (SVMs) [2, 16, 56] are a machine learning technique. They estimate a hyperplane which separates different classes of data points. In order to deal with data that are not linearly separable, they use a kernel function to map their input into a higher-dimensional space, in which the individual classes are linearly separable. Common types of functions used as kernels are *polynomial*, *radial-basis* and *sigmoid* functions. One-class SVMs (OCSVMs) [16] are a version of SVM that can be used for novelty detection. OCSVMs find a hyperplane that separates regions of high density from regions of low density in the training dataset. New data points are then classified as normal or anomalous by calculating their signed distance to the separating hyperplane. If the signed distance is positive, the new data point is in a region of high density and thus considered normal. If it is negative, it is considered anomalous. Additionally, the distance of a data point to the separating hyperplane can be interpreted as level of confidence that the classification is correct [55].

Using kernel functions is time- and memory intensive. Calculating a kernel matrix is at least quadratic in terms of computational complexity. One method for speeding up OCSVMs is the usage of *linear OCSVMs* (which do not use a kernel for mapping its input into a higher-dimensional space) combined with a kernel approximation technique. A popular kernel approximation technique is called the *Nyström method* [12, 71, 75]. The method replaces the kernel matrix by a different matrix of lower rank, approximating the kernel matrix. This approach reduces computational overhead at the cost of accuracy. Because this combination of a linear OCSVM and the Nyström method uses stochastic gradient descent (SGD) for optimising, we refer to it as OCSVM-SGD.

OCSVM and OCSVM-SGD can be tuned via three hyper-parameters. The first parameter that can be varied is the kernel function to be used or approximated. Hyper-parameter γ is a scaling factor for the distances OCSVM and OCSVM-SGD calculate between vectors. Intuitively, increasing the value of γ means fitting the hyperplane more tightly around the regions of normal points. Too low values of γ lead to underfitting, too large values lead to overfitting. The hyper-parameter ν controls the maximum permitted training error, i.e. the

maximum share of training data that we accept to be misclassified. In the case of ν, low values can lead to overfitting and large values can lead to underfitting.

3.2.3 Local Outlier Factor

Breuning et al. introduce a *Local Outlier Factor* (LOF) [7]. This factor is calculated for each point in a dataset and describes the degree to which a data point can be considered an outlier. The authors use LOF for outlier detection, i.e. splitting a given dataset into normal and abnormal data points. However, our use case is to perform novelty detection, i.e. classifying new data points as normal/anomalous with respect to an existing training dataset that is assumed to contain normal points only. As we show below, is is possible to use LOF for novelty detection. Prior to that, we give a brief explanation of LOF to provide the reader with an intuition of its meaning. We refer to Breunig et al. [7] for a detailed explanation of the calculation of LOF.

LOF is a density-based metric. Given a dataset X and a data point $x \in X$, we calculate $LOF(x)$ through the following steps. First, we assign each point $p \in X$ a *local reachability density*, which is influenced by the distance to the point's closest neighbours. $LOF(x)$ is then calculated by comparing x's local reachability density to that of its neighbours. The LOF of points inside or at the edge of a cluster is close to 1, because their local reachability density is approximately the same as their nearest neighbours'. In contrast to that, outliers have a smaller density than their nearest neighbours and thus a LOF significantly greater than 1. Thus, all data points with a LOF greater than some threshold can be considered abnormal.

When using this method for novelty detection, we are given a training dataset $X_{training}$ and a point x which may or may not be in $X_{training}$. In order to calculate $LOF(x)$, we perform the steps above with $X = X_{training} \cup \{x\}$.

Notably, Breunig et al. do not specify a distance metric to be used for the calculation of distances between data points. The optimal distance metric to be used depends on the task at hand and can be determined empirically.

4 APPROACH

4.1 GENERAL ARCHITECTURE

We propose a system for detecting security-related anomalies in HPC systems. The architecture of this system is shown in Figure 4.1, with the parts we implemented and evaluated experimentally being filled with colour. Elements of this architecture can be grouped into different categories. The available sources of log data are displayed in yellow (`/var/log/secure`, `/var/log/messages` and the workload manager) and described in detail in Section 2.2.1 and Section 2.2.2. Data from these sources go through a preprocessing step. In this step, they are converted into a format suitable for performing anomaly detection. This means that the unstructured log data are transformed into structured data: login events are extracted, the beginning and end of user sessions are reconstructed, and system log events are assigned to the user sessions in which they occurred. The preprocessing steps are displayed in green and described further in Section 4.2.

The structured data are then grouped by user session (purple, see Section 4.3). A user session is defined as a time-span during which a user is logged in on at least one node in the HPC system.

The most interesting elements of the architecture are the anomaly detectors (shown in blue, see Section 4.4). Anomaly detectors look for specific types of anomaly that could indicate malicious behaviour inside the HPC system. For each session s, they calculate an *anomaly score* ($p_1(s) - p_4(s)$), which indicates the degree of certainty that an anomaly has been found. Anomaly scores fulfil the following properties:

- $\forall s \forall i \in \{1 \dots 4\} : p_i(s) \in [0, 1]$
- Larger anomaly scores indicate a higher certainty that an anomaly has been found.

Despite representing certainties, $p_1(s) - p_4(s)$ cannot be interpreted as probabilities (i.e. an anomaly score of 0.5 does not necessarily mean that an anomaly has been found with 50% probability). They are only useful for comparing the degree to which sessions are anomalous.

The anomaly scores are then combined through a thresholding scheme (red, see Section 4.5). The result is a binary value (yes/no) that classifies the session in question as either normal or anomalous. If a session is classified as anomalous, a system operator is alerted. They are shown the logs produced during the session in question as well as the anomaly scores $p_1(s) - p_4(s)$, which they can use to investigate the

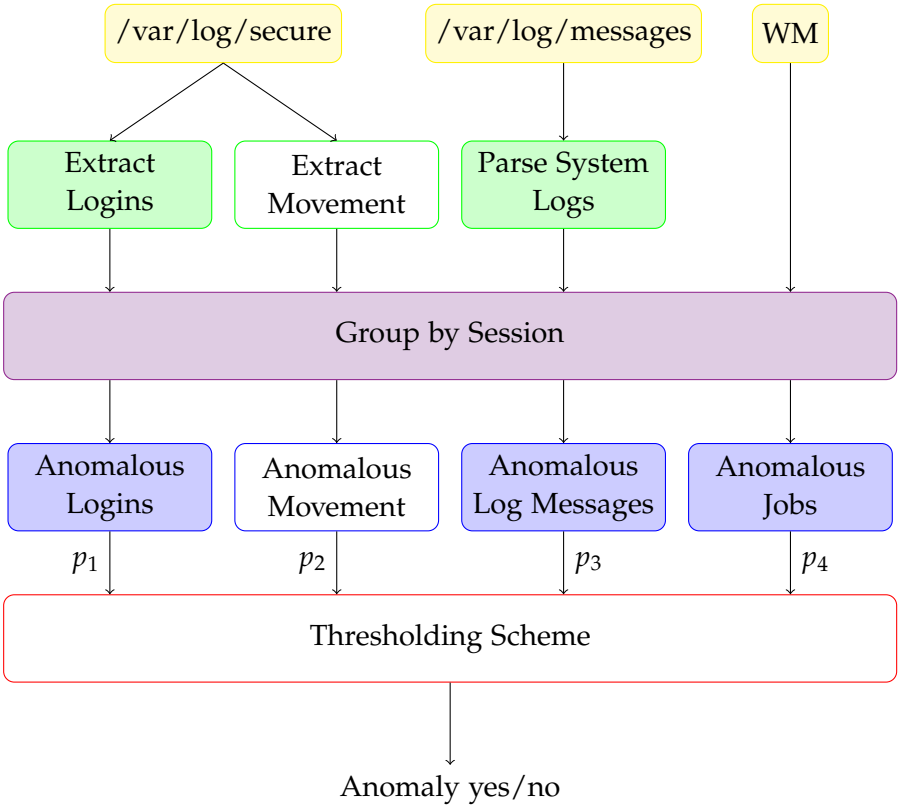


Figure 4.1: Architecture of a framework for anomaly detection in HPC environments.

suspected anomaly. If the suspected anomaly turns out to be normal behaviour (false positive), the operator can feed this information back into the system. The system then readjusts the anomaly detectors, which consequently no longer classify this behaviour as abnormal, lowering their false positive rate.

In the following Sections we describe the individual components of our architecture in detail. Section 4.2 focuses on the preprocessing steps necessary to transform the data into a suitable format. In Section 4.3, we explain the idea of grouping data into user sessions. Section 4.4 explains each anomaly detector as well as the algorithms used to implement them. We elaborate on the thresholding scheme used to combine the individual detectors' anomaly scores in Section 4.5. An explanation of the aforementioned inclusion of operator feedback is given in Section 4.6.

4.2 PREPROCESSING

4.2.1 Extracting Logins

`/var/log/secure` contains logs related to authentication. We are primarily interested in successful *Login* events. A login event can be described by the attributes *user*, *source ip*, *authentication method*, *node* and *time*. We extract login events from `/var/log/secure` by filtering it for logs in which the log message starts with the word `Accepted`. These lines correspond to a successful authentication and contain all information needed to describe the login process. We provide an example of such a log line in Listing 4.1. Further, we enrich the extracted events with one additional field in order to ease analysis. This field (which we name *location*) contains the name of the location to which the source IP address of the respective login event corresponds.

4.2.2 Extracting Movement

Movement of users inside the cluster can be traced via `/var/log/secure`. Opening and closing a session for a user via *ssh*, *sudo*, *su* or similar programmes leaves log entries in this file.

As we did not implement the extraction of movement and the anomalous movement detector, we cannot comment on convenient representations of movement data. One possibility is to represent the sequence of visited nodes as an univariate time series. We do not store the exact names of nodes visited; instead, we only save the types of node visited. We do this because nodes of one type are usually identical, their individual identity does not matter us.

For example, the movement of a user starting a session on node `login123` at time t_1, jumping to node `compute456` at t_2 and lastly logging into node `visualise789` at t_3 can be represented through the list

$$((\texttt{login}, t_1), (\texttt{compute}, t_2), (\texttt{visualise}, t_3))$$

4.2.3 Parsing System Logs

Log parsing, i.e. bringing the unstructured log messages into a structured format, can be achieved using an algorithm for automated log

Listing 4.1: A log line from `/var/log/secure` showing a successful login event. The line shows all relevant information (time of login, node, authentication method, user and source IP.)

```
Mar 11 14:22:38 login03 sshd[357112]: Accepted publickey for xyzw
     from 2001:...:9a port 63594 ssh2 ED25519 SHA256:eRtT...
```

parsing. We evaluate numerous such algorithms in Section 7.1 and determine that Drain is suited best for usage in the proposed architecture, because it is very efficient and reasonably accurate.

4.3 SESSION GROUPING

4.3.1 Method

We define a user session as a time-span during which a user is logged in on at least one node in the HPC system. Each session has a numeric identifier (Session ID), a start time and an end time. We group the collected data by session, i.e. each date is enriched with the ID of the session it belongs to.

4.3.1.1 Logins & Movement

Assigning logins and user movement to user sessions is straightforward, because `/var/log/secure` contains records of login and logout events. If a login for a user is registered in `/var/log/secure`, we check whether this user is already logged into the system (i.e. if there exist more login than logout events for this user). If they are not, the new login is assigned to a new user session, using the login's timestamp as its start time. If they are, the login is assigned to the running user session. If a logout event is encountered, we check whether this logout terminates the current user session (i.e. whether the number of logins registered for this user equals the number of logouts for this user). In this case, the logout's timestamp is used as the session's end time. By keeping track of the nodes that a user logs into during a session, movement in the cluster can be assigned to sessions as well.

4.3.1.2 Job Information

Job information acquired from the workload manager typically contains a timestamp indicating the submission time. We assign each job to the user session active during its submission. As user sessions are non-overlapping, this assignment is always unambiguous.

4.3.1.3 System Logs

Assigning log messages from `/var/log/messages` to user sessions is more complicated. These entries do not carry information directly linking them to the user who triggered them. Instead, we use knowledge about the time-spans during which specific users are logged into specific nodes to infer the user responsible for log entries. In other words: if a node is used by one user exclusively, all log entries are assigned to this user. For this reason we make the following assumption:

during the time-span in which a job submitted by user u is running on a node, no user except u is permitted to login onto this node. Indeed, this assumption holds for many WMs in productive use (e.g. Slurm). For this reason, we can use the job information from the WM (i.e. start and end times of all jobs as well as the names of allocated nodes) to determine which user ran jobs on which nodes at what times. All logs produced on a node during the execution time of a job from user u are therefore assigned to u, or more precisely to the user session during which u submitted the job in question.

4.3.2 Limitations

Unfortunately it is not possible to assign all available data to user sessions. The method for assigning system log entries to user sessions is only possible on nodes executing compute jobs, i.e. compute nodes. Further, system logs produced during idle time (while no job is executed) cannot be traced back to a particular user or session. Login nodes do not run compute jobs, therefore system logs produced on login nodes can not be assigned to specific user sessions. Theoretically, it is possible to detect time-spans in which only one user is logged into a login node and to assign logs produced during this time-span to this user session. However, we doubt that this procedure would succeed in assigning a large number of log entries to user sessions and do not consider it further.

4.4 ANOMALY DETECTORS

4.4.1 Anomalous Logins

Prior to performing any kind of action on a HPC system, users have to login onto the system. According to operators of existing HPC systems, these logins exhibit some regularity. For example, users usually log in from one or two locations exclusively. This makes sense, since most users of HPC systems are either researchers or professionals, who use the provided services either from their workplace or from their home. Further, many users only log in at specific times of the day or using one fixed authentication method. We suspect that deviations from these patterns could indicate an attack.

For detecting anomalous logins, we use authentication logs found in `/var/log/secure`. Logins can be classified as either normal or abnormal. Finding a classification function

$$f : A \to \{0,1\},\ x \to \begin{cases} 0 & x \text{ is not anomalous} \\ 1 & x \text{ is anomalous} \end{cases}$$

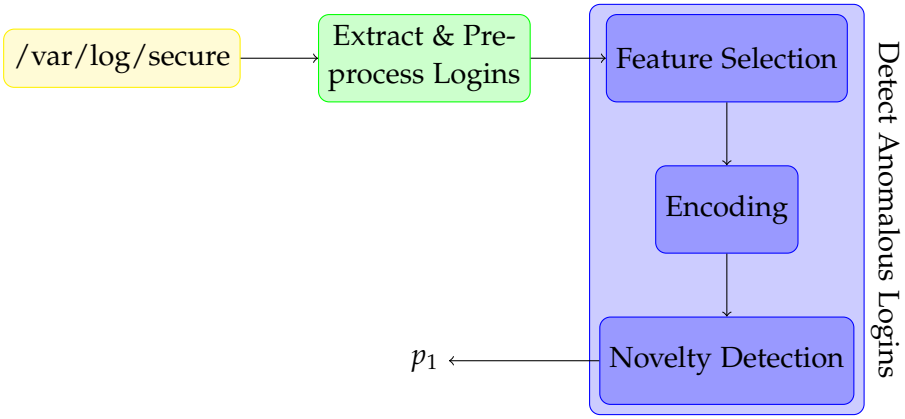


Figure 4.2: Data flow necessary for the process detecting anomalous logins.

with A denoting the set of possible logins is not trivial, because is not clear what exactly constitutes an abnormal login. It is hard for a domain expert to come up with general rules for classifying an login as either normal or abnormal. Because of that, we utilise methods for novelty detection in order to detect anomalous logins automatically. The methods we use are described in Section 3.2.

We work under the assumption that users have a login behaviour that is specific to them and may differ from the login behaviour of other users. Therefore, we analyse the logins of each user separately.

The flow of data for detecting anomalous logins is given in Figure 4.2. We describe the structure of the authentication logs in Section 2.2.2 and the login extraction/preprocessing of these data in Section 4.2.1. After preprocessing, we select appropriate features, which are then encoded into a format suitable for input into novelty detection models (see Section 4.4.1.1). The encoded features are fed into a model for novelty detection. We test four different models for novelty detection, namely Isolation Forest, OCSVM, OCSVM-SGD and LOF. The evaluation process and its results are described in detail in Section 7.2.

4.4.1.1 Feature Selection and Encoding

After preprocessing, a login has several features, namely *user*, *source ip*, *authentication method*, *node*, *time* and *location*. We decide to only consider the fields *authentication method*, *time* and *location*. The field *user* is not necessary, because we perform the analysis on a per-user basis anyways and including the field would result in it having the same value across all data points. The field *node* is not considered, because users generally do not control which node they log in to. *Success* is not considered because we only consider successful logins.

We need to encode the selected features in a way that is suitable as input for a novelty detection model. *Authentication method* and *location* are categorical data, and therefore encoded as one-hot vectors. In order to perform one-hot encoding, it is necessary to know all

possible values for the fields beforehand. For *authentication method*, this is clearly the case, as systems usually only support a limited number of authentication methods (e.g. via a public key, via a password etc.) For *location*, we assume that login is permitted only from a small number of locations (as is the case in our test system).

Time is more complicated to encode. We assume that patterns in login behaviour are periodical. For example, a pattern might be that a user typically logs into their account between 9:00 and 11:00 on weekdays. In order to recognise such a pattern, the only relevant information encoded in *time* is the time of day and the day of the week. Therefore, we transform the field *time* so that we are left with two new fields, *time-of-day* and *day-of-week*. As *day-of-week* is a categorical value, we transform it into a one-hot encoded vector. *Time-of-day* is not to be viewed as a categorical value. We argue that some time-of-days are closer to each other than others. This is a property that we would like to preserve in the encoding of this feature, as it might help recognising usage patterns or anomalous usage. In particular, we want to make the correct difference of time-of-days obvious in our model for novelty detection. Suppose we chose an encoding which encodes each time-of-day as its hour-of-day (i.e. 0:00 is encoded to 0, 1:00 is encoded to 1 etc.) In this encoding, it would seem like the difference between 23:00 and 1:00 was larger than the difference between 23:00 and 2:00 (because $|1 - 23| > |2 - 23|$).

To avoid this, we transform time-of-day into a Grey encoded bit vector. Grey codes were introduced in 1953 by Grey [20]. A n-bit Grey code can encode 2^n distinct values, each of which is encoded as a n-bit codeword. An example of a 3-bit Grey code is given in Table 4.1. Grey codes have the property that successive values differ by exactly one bit. This *adjacency property* holds even in the case of an overflow, i.e. the largest and the smallest representable values in a n-bit Grey code differ by exactly one bit [18, pp. 100–101].

INTEGER	BINARY	GREY ENCODING
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

Table 4.1: Comparison of the first 8 integers' binary notations and 3-bit Grey encodings.

In order to profit from the adjacency property, we must make use of all possible codewords in a given n-bit Grey encoding. If we truncated a given time-of-day to its hour-of-day before encoding, we would have 24 distinct values to encode. Hence, we would need to set $n \geq 3$ in order to have $2^n \geq 24$ available codewords. Because 24 is not a power of 2, we cannot find an n such that the 2^n available codewords are all used. In order to achieve full utilisation, we transform the time-of-day before encoding it by using Equation 4.1, where t is the time-of-day in seconds and ℓ is the total number of seconds in a day, i.e. $\ell = 24 \cdot 60 \cdot 60 = 86400$.

$$t_{trans} = \left\lfloor \frac{t \cdot 2^n}{\ell} \right\rfloor \tag{4.1}$$

Conceptually, this divides the day into 2^n parts of equal length. Each time-of-day is then mapped to the part which it falls into. Because $t_{trans} \in \{0 \dots 2^n - 1\}$ and $|\{0 \dots 2^n - 1\}| = 2^n$, we utilise all possible codewords of an n-bit Grey code.

Through this approach, the selected features (*authentication method*, *location*, *day-of-week* and *time-of-day*) are encoded as bit-vectors. *Authentication method*, *location* and *day-of-week* are one-hot encoded, whereas *time-of-day* is transformed as described above and Grey encoded. The input into the employed novelty detection model (of which we test several ones), is thus a rather large bit-vector. In order to calculate the size of this input vector, we add up the sizes of its components. Let $F_{location}$ equal the set of possible locations from where a login can be performed, $F_{authmethods}$ the set of possible authentication methods and F_{dow} the set of days of the week. Using an n-bit Grey code to encode *day-of-week*, the length of the input vector ℓ_{login} is thus given as $\ell_{login} = |F_{location}| + |F_{authmethods}| + |F_{dow}| + n$.

4.4.1.2 Novelty Detection Model

The input vectors are fed into a novelty detection model. Such a model can be in one of two modes: training mode and prediction mode. In training mode, the model is fed with training data, which the model learns to recognise as normal behaviour. In prediction mode, the model is fed with data points for which it is unknown whether they are normal or not. The model makes an estimation on the normality of each of these points. This estimation can be binary (yes/no) or continuous. If it is continuous, it indicates how anomalous the data point in question is.

In Section 7.2, we test four different novelty detection methods Isolation Forest, OCSVM, OCSVM-SGD and LOF. LOF is tested with several commonly used distance metrics. We find that OCSVM-SGD without online adjustments performs best in terms of both precision and recall. Going forward, we therefore use OCSVM-SGD to determine

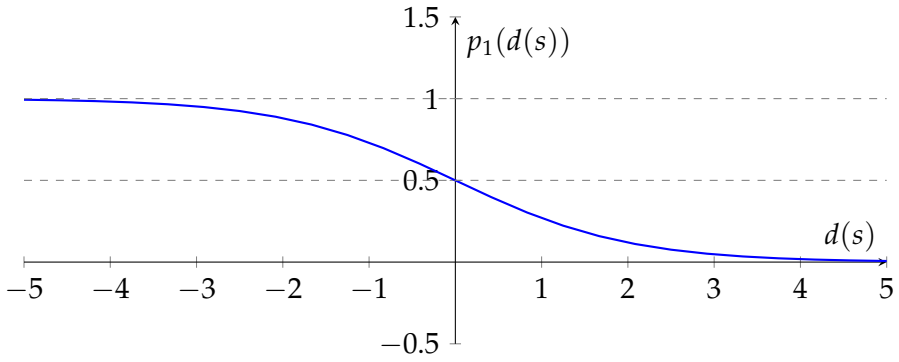


Figure 4.3: Plot of $p_1(d(s)) = \frac{1}{1+e^{d(s)}}$.

the abnormality of logins. Let $d(s)$ be the signed distance of a session s to the separating hyperplane as calculated by the OCSVM-SGD model. If $d(s) \geq 0$, s is considered normal and if $d(s) < 0$, it is considered anomalous. We calculate the anomaly score p_1 by projecting $d(s)$ into the interval $[0, 1]$. For this, we use the sigmoid function $f : \mathbb{R} \to [0, 1], \text{sig}(x) = \frac{1}{1+e^{-x}}$ [58, p. 148]. We define the anomaly score $p_1(s) = \frac{1}{1+e^{d(s)}}$, i.e. the sigmoid function applied to the negative signed distance, which is shown in Figure 4.3. As we see, $p_1(s)$ grows towards 1 as $d(s)$ approaches $-\infty$ and goes towards 0 as $d(s)$ approaches ∞. Anomalous logins are thus assigned an anomaly score > 0.5, whereas the anomaly score of normal logins is ≤ 0.5.

The relationship between $d(s)$ and $p_1(s)$ is illustrated on various normal and anomalous logins in Appendix B.

4.4.2 Anomalous Movement

A HPC system usually consists of many nodes. These nodes typically belong to different classes, i.e. login nodes, compute nodes, visualisation nodes, administrative nodes or IO nodes. A session typically starts with the user logging into a login node. From this login node, the user can move inside the cluster, depending on the goal they want to accomplish.

In conversations with domain experts we discovered that compromised user accounts misused for an attack tend to show conspicuous movement behaviour. For example, it has been observed that during an attack, malicious actors often try to move to administration nodes. This of course is denied as they lack the permissions to do so. In one instance, attackers exploited a kernel vulnerability on one node in order to gain root privileges. This lead to a situation in which root was present on a node, producing log entries, without ever logging into this node.

Even less obvious instances of anomalous movement behaviour can be an indication for malicious user behaviour. For example, a user

randomly jumping from one node to another could be an attacker mapping out the system topology.

We believe that movement information can be a valuable source of information for detecting malicious user behaviour. Because we did not implement a detector capable of this, we reflect on how the challenge of detecting anomalous movement can be solved in Section 4.4.2.1.

4.4.2.1 Possible Approaches

In order to detect anomalies in movement data as laid out in Section 4.2.2, we need to perform anomaly detection on a univariate time series of categorical values.

Taha and Hadi [63] suggest two approaches for detecting anomalies in categorical data, which they call *signature-based* and *anomaly-based*. A signature-based anomaly detector compares the time series at hand to signatures of known anomalous behaviour. If a signature is matches, the time series is flagged as anomalous. On the other hand, anomaly-based methods usually build a model of normal time series and flag deviating data as anomalous.

Signature-based anomaly detectors require a database of signatures describing anomalous behaviour. They can only detect anomalies for which signatures are present. In order to detect novel anomalies, an update of the database is required; signature-based anomaly detectors therefore need constant maintenance. However, in the context of detecting anomalous movement between nodes in a HPC cluster as part of our proposed SIEM, a signature-based approach might be sufficient. The number of suspicious movement behaviour possible might be limited. If this is the case, the effort to compile a database of anomalous signatures might be worthwhile.

Anomaly-based anomaly detection does not require constant maintenance. However, to our knowledge, anomaly-based anomaly detection on categorical time series has not been widely studied [63]. A recent work by Horak, Chandrasekaran, and Tobar [28] achieves promising results on multivariate time series of categorical data, but it is not clear whether this performance can be matched on univariate time series.

We believe that the execution path anomaly detection in DeepLog by Du et al. [14] (see Section 3.1.2.1) might be well-suited for the task of detecting anomalous movement. It was originally designed to detect anomalous log messages and achieves this task by treating log sequences of univariate categorical data. For the performance of the model, it should make no difference if the categorical data fed into it represents log keys or movement across a HPC cluster. We thus suspect that DeepLog might solve the task of detecting anomalous movement.

Unfortunately, because of time constraints, we cannot evaluate DeepLog's performance as a detector for anomalous movement. This is left for future work.

4.4.3 Anomalous System Logs

As discussed in Section 2.2.2, system logs (*/var/log/messages*) record rich information about the system's state over time. Malicious behaviour in a computer may be reflected by anomalies in system logs, e.g. through unusual log key patterns, unusual log parameters or novel log keys.

HPC systems are usually dedicated to a narrow scientific field. Access is often limited to a small group of researchers who run a small number of known programmes [8, 65]. We suspect that HPC system logs reflect this regular computing behaviour and that anomalies in these logs are indeed an indicator of malicious behaviour. This makes the analysis of system logs an interesting tool to us.

As discussed in Section 3.1, this process consists of three steps: log collection, log parsing, feature extraction and anomaly detection. For this work, log collection is out of scope and log parsing is covered in Section 4.2.3. We propose to realise the remaining steps of feature extraction and anomaly detection using existing solutions, i.e. one of the anomaly detection methods reviewed in Section 3.1.2.

4.4.3.1 Feature extraction

The traditional and neural network based machine learning approaches we consider differ in the way they extract features from the input log sequences. DeepLog (as a neural network based machine learning approach) consumes the log event sequences themselves, leaving the recognition and extraction of important features to the neural network architecture. The traditional machine learning approaches we review (LogCluster, Invariants Mining and PCA) on the other hand transform their input to an event count matrix before processing. In order to construct an event count matrix, each input log sequence is transformed into an event count vector. An event count vector describes how often each event occurred in the log sequence at hand. For example, the event count vector $(4, 0, 2, 3, 0, 0)$ would indicate that event 1 occurred four times, event 3 occurred two times, event 4 occurred three times and events 2, 5 and 6 did not occur at all. The event count vectors of the input log sequences form an event count matrix. LogCluster, Invariants Mining and PCA take this event count matrix as input, both for training and for prediction [9, 26].

4.4.3.2 Anomaly Detection

The methods for anomaly detection introduced in Section 3.1.2 (DeepLog, PCA, LogCluster and Invariants Mining) have some similarities: Each of them consists of an unsupervised training phase and a prediction phase. In the training phase, a model describing normal log sequences is learned. During prediction phase, new log sequences are

classified as normal or anomalous by checking whether or not they fit into this model.

The prediction phase of DeepLog differs from the prediction phases of the other methods: While the other methods work on a *per log sequence* basis, DeepLog operates on a *per log key* basis. In other words, DeepLog works in a streaming manner: it continually consumes log keys of a sequence as they are produced, judging the normality of each log key separately. This means that it is able to process sequences in real-time: anomalous log keys can be detected immediately after they are produced, even if the responsible user session is not finished yet. In contrast, the methods operating on a *per log sequence* basis cannot predict the normality of an incomplete sequence; prediction is only possible after the corresponding user session has finished. Using an anomaly detector that is not able to work in a streaming manner may reduce the effectiveness of the overall SIEM, because it prevents the system from detecting anomalous log messages as they are produced. For example, a long-running malicious user session could emit anomalous log messages for hours or days without triggering an alert. It is only after the session finished that the operators are alerted, but at this point significant damage may already have been done.

In Section 7.3 we perform experiments in order to find the most suitable method for anomaly detection and the best-performing hyperparameters for this method. We find that LogCluster performs best, both in terms of precision and in terms of recall (and thus also in F1-score). Its efficiency is sufficient to enable rapid classification of log sequences and frequent model retraining. The only candidate working in a streaming manner, DeepLog, cannot provide satisfactory performance. Despite the disadvantages of anomaly detectors working on a *per log sequence* basis, we decide to use LogCluster for detecting anomalous log sequences, i.e. calculating the anomaly score p_3. Thus, p_3 is a binary value: 0 for logs that LogCluster classifies as normal and 1 for logs classified as anomalous. In principle, it would be possible to have LogCluster output a continuous value describing the degree to which a log sequence is anomalous: the minimum cosine distance of the vectorised log sequence to the cluster centroids. Because the implementation of LogCluster that we use does not support this, we do not investigate the impact of this idea on the overall performance of the SIEM further, but leave it for future work.

4.4.4 Anomalous Jobs

HPC systems typically make use of workload managers (WMs). In order to submit a compute job, users have to interact with the WM. This includes specifying the required computing resources, the commands to be run or dependencies between submitted jobs. The WM usually keep a record of jobs submitted by users, including rich information

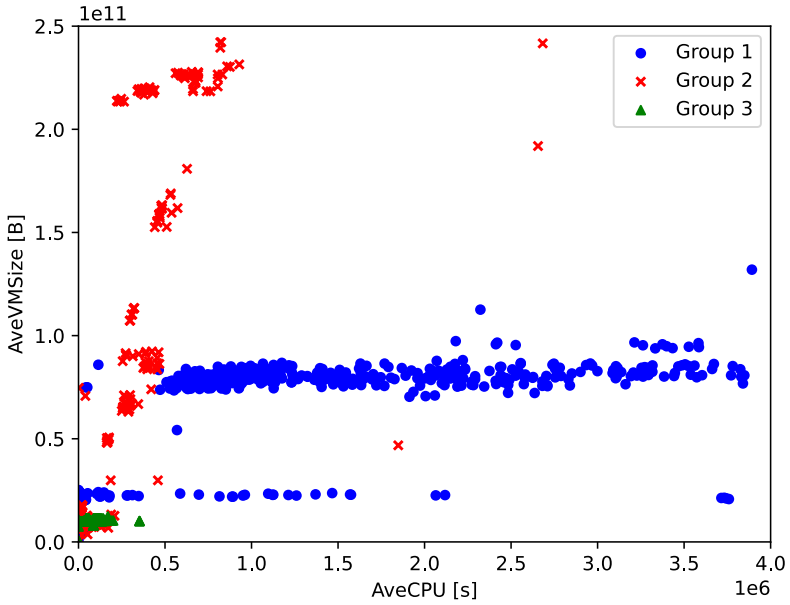


Figure 4.4: The computing behaviours of different research groups (which are expressed in terms of the submitted jobs' average virtual memory size and average CPU time across all tasks of a job).

about the requested resources, the resources actually used as well as other metadata. We use this record to find anomalous job submissions that could indicate malicious behaviour in the HPC system.

Users on HPC systems are typically assigned to research groups. These groups represent teams of researchers whose members work on solving a particular problem or answering one specific question. We assume that these groups have individual usage patterns: they differ in the types of jobs they submit. Some groups might use methods requiring short but memory-intensive jobs, while other groups might submit long-running, CPU-heavy workloads running on a large number of nodes. A user suddenly submitting jobs that differ significantly from their group's usual behaviour can be an indicator that the user's account has been compromised and is now being used for illicit purposes, e.g. crypto-currency mining.

Figure 4.4 demonstrates that groups indeed have observable typical behaviour. It shows jobs submitted by three research groups on a real-world system (JUSTUS2, see Section 5.1). The x- and y-axis represent the CPU time and memory that nodes executing the jobs consumed on average. We immediately see that the three research groups exhibit different computing behaviour. Group 1's jobs almost never consume more than 10^{11} B (= 100 GB) of memory, but have a high spread in terms of consumed CPU time. The jobs form two horizontal lines in the plot, suggesting that Group 1 uses two programmes with different resource requirements for their research. Group 2's usage pattern is

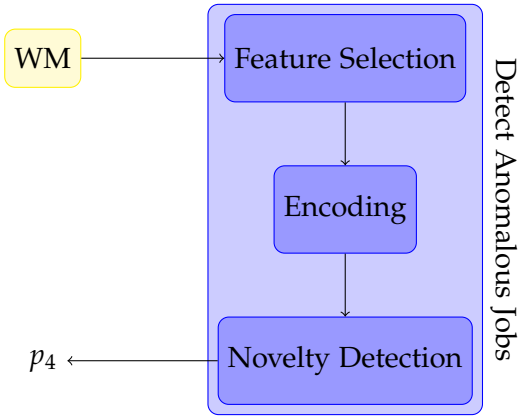


Figure 4.5: Data flow during the process of detecting anomalous jobs.

different: the average CPU time does typically not exceed 10^6 s ($\approx$ 11 days). However, memory requirements of Group 2's jobs are very heterogeneous. Lastly, Group 3's jobs require much less memory and CPU time than the other groups'.

Evidently, the jobs of a given research group form clusters. Normal jobs lie inside one of these clusters, while anomalous jobs do not. Deciding whether a jobs is anomalous or not is therefore equivalent to deciding whether or not it lies inside the clusters of normal behaviour.

We approach the problem of calculating whether or not jobs lie inside a cluster using a similar approach as in Section 4.4.1, i.e. utilising methods for novelty detection. This approach consists of two phases: a training phase and a prediction phase. During the training phase, we use job data collected during normal HPC operations to build a model of normal behaviour. We build one model per research group, which then captures characteristics of typical compute jobs submitted by it. During the prediction phase, newly submitted compute jobs are classified as either normal or anomalous.

In order to build a model of normal behaviour, we first have to perform feature selection and feature encoding. Afterwards, we can utilise methods for novelty detection in order to detect anomalous jobs. This process is shown in Figure 4.5.

4.4.4.1 *Feature Selection and Encoding*

WMs provide rich information about the jobs submitted by users. Slurm, which is a popular WM, provides 108 data fields for each job. Many of these features are not useful for anomaly detection, e.g. non-numeric fields such as *JobName* or fields over which the user has no control such as *DBIndex*, which is used internally by Slurm. Some fields (such as the *Req** family of fields, used for denoting the resources requested by users for a job) are left blank by Slurm for reasons unbeknownst to us. We consequently do not consider these fields further.

After excluding non-useful fields, we are left with 22 features for detecting anomalous jobs: *AveCPU*, *AveCPUFreq*, *AveDiskRead*, *AveDiskWrite*, *AvePages*, *AveRSS*, *AveVMSize*, *ConsumedEnergyRaw*, *CPUTimeRAW*, *ElapsedRaw*, *MaxDiskRead*, *MaxDiskWrite*, *MaxPages*, *MaxRSS*, *MaxVMSize*, *MinCPU*, *NCPUS*, *NNodes*, *NTasks*, *TimelimitRaw*, *GPUs*, *Interactive*.

Ave-/MinCPU, *AveCPUFreq*, *Ave-/MaxDiskRead*, *Ave-/MaxDiskWrite*, *Ave-/MaxPages*, *Ave-/MaxRSS* and *Ave-/MaxVMSize* denote the average/maximum/minimum CPU time, CPU frequency, number of bytes read and written, number of page faults, resident set size and virtual memory size of all tasks of a job. *ConsumedEnergyRaw* denotes the total energy consumed by the job. *ElapsedRaw* indicates the job's elapsed wall-time, *CPUTimeRAW* indicates the CPU time used by the job, i.e. $ElapsedRaw \cdot NCPUS$. *NCPUS* and *NNodes* are the number of CPUs/Nodes allocated to the job, *NTASKS* is the number of tasks a job consists of. *TimelimitRaw* is the maximum time-span a job is allowed to run. *GPUs* and *Interactive* are not directly included in the output of Slurm, but can be deduced by combining information from the fields *AllocTRES* (Allocated Trackable Resources) and *JobID*. *GPUs* is the number of GPUs allocated for a given job; *Interactive* describes the mode of the job and is equal to 1 for interactive jobs and 0 for batch jobs.

Other WMs than Slurm may report information about jobs in a different format. If this is the case, our approach is only suited if fields equivalent to the ones we selected above are available. Detection of anomalous jobs may perform significantly worse than reported in Section 7.4 when using a WM providing less information about jobs than Slurm.

At this point, input vectors are 22-dimensional. Generally speaking, high-dimensional data exhibit undesired properties such as the *curse of dimensionality*. The more dimensions, the harder it becomes to build accurate models of the data using machine-learning techniques [31]. We try to mitigate this issue by performing dimensionality reduction. Dimensionality reduction aims to reduce the number of dimensions in the data at hand without losing substantial information [41]. We use PCA for this task. In order to select the number of components to keep we use MLE, a technique for automatic dimensionality selection introduced by Minka [46].

After performing dimensionality reduction, each of the fields we selected is converted to a floating point value and normalised. In order to normalise a field, we remove its mean and scale it to a variance of 1.

4.4.4.2 Anomaly Detection Model

We use methods for novelty detection for classifying jobs as normal or anomalous. We test the performance of the four algorithms for novelty detection outlined in Section 3.2: Isolation Forest, OCSVM,

OCSVM-SGD and LOF. From our tests (see Section 7.4) we conclude that LOF using the Bray-Curtis dissimilarity as a distance metric is suited best for the detection of anomalous jobs, because it achieves a high recall while keeping precision at an acceptable level.

As a user session s can contain multiple jobs, we only use the most anomalous job submitted during a session, i.e. the job having the largest LOF, to calculate $p_4(s)$. Further, for reasons of comparability, p_4 must lay in the interval $[0, 1]$. In a similar approach as described in Section 4.4.1.2, we use a sigmoid function to project the maximum calculated LOF into this interval. Let $j_i, i \in \{1 \dots m\}$ be the jobs submitted during a user session s of a user belonging to research group g. We define $p_4(s) = \text{sig}(\max\{\text{LOF}_g(j_i) \mid i \in \{1 \dots m\}\})$, where $\text{sig}(x) = \frac{1}{1+e^{-x}}$ [58, p. 148].

4.5 THRESHOLDING SCHEME

In order to classify a user session s as normal or anomalous, it is fed into each anomaly detector. This results in four anomaly scores, $p_1(s), p_2(s), p_3(s), p_4(s)$. Generally, the greater $p_i(s)$, the higher the certainty that an anomaly has been found. The thresholding scheme element of our SIEM uses these four values to decide whether or not the user session in question is anomalous.

In conversations with operators of HPC systems we discovered that real-world anomalous user sessions typically exhibit multiple types of anomalous behaviour. We leverage this fact in order to keep the number of false positives low and thus reduce load on HPC staff by alerting operators only if a certain number of anomaly detectors report anomalous behaviour.

Per user session s, our SIEM produces four anomaly scores $p_i(s)$, $i \in \{1 \dots 4\}$. An alert is triggered if the weighted sum of these anomaly scores meets a set threshold, i.e. if Equation 4.2 holds.

$$\sum_{i=1}^{4} \frac{p_i(s)}{\psi_i} > \omega \tag{4.2}$$

The values $\frac{1}{\psi_i}$ determine the weighting of the individual anomaly scores. ψ_i can be interpreted as a threshold: if $p_i(s) > \psi_i$, then $p_i(s)$ indicates an anomaly. We use this interpretation to select values for ψ_i. As discussed in Section 4.4.1.2, $p_1(s) > 0.5$ for anomalous logins and $p_1(s) \leq 0.5$ for normal logins. We thus set $\psi_1 = 0.5$.

We conjecture that DeepLog [14] is well-suited for calculating $p_2(s)$. DeepLog returns a boolean value (0 or 1); we set $\psi_2 = 1 - \epsilon$ (with ϵ being an arbitrarily small positive value less than 0.5). The reason for introducing ϵ is that we would like to achieve the property that $0 < \psi_2 < 1$.

Through our experiments we discovered that the algorithm suited best for calculating $p_3(s)$ is LogCluster. It too returns a binary value. We thus set $\psi_3 = 1 - \epsilon$.

$p_4(s)$ is produced by the detector for anomalous jobs, which uses LOF as its detection algorithm. We follow Pedregosa et al. [52] in selecting the threshold above which LOFs are considered anomalous as 1.5. We thus set $\psi_4 = \text{sig}(1.5) \approx 0.818$.

If the weighted sum of the anomaly score exceeds ω, the session in question is considered anomalous and an alert is triggered. A good value for this parameter can most likely only be determined empirically; we propose to use $\omega = 4$ as a start and adjust as necessary. With $\omega = 4$, no single anomaly detector can trigger an alert, reducing the number of false positives. In order to classify a session as anomalous, more than one anomaly detector must return an anomaly score that is considered too large to be normal. Real-life anomalies caused by malicious actors typically exhibit more than one type of anomaly, resulting in them being detected by our SIEM.

4.6 OPERATOR FEEDBACK

Upon investigating sessions categorised as anomalous by our SIEM, system operators may encounter false positives. This can be due to the fact that the anomaly detectors in use only use a small percentage of the available log data as training data. The training data may not reflect every possible normal behaviour, and previously unseen behaviour is likely to be classified as anomalous. Using a larger percentage of the available data for training is not a good mitigation strategy, because it substantially increases training- and prediction time. Further, user behaviour might change over time. For example, a new researcher joining a research group might change the group's computing behaviour substantially. Instead, we update the models for anomaly detection if false positives are noticed.

DeepLog, which we use for detecting anomalous movement, supports online updates. Log sequences falsely classified as anomalous can be used to adjust the model in such a way that it no longer recognises this behaviour as anomalous. This adjustment is performed without retraining the model from scratch, taking only marginally longer than a normal prediction. If a false positive is found, we simply adjust the model using the procedure outlined by Du et al. [14].

OCSVM-SGD, LogCluster and LOF, which we use for detecting anomalous system log messages and anomalous jobs, do not support online updates of their models. Fortunately, these algorithms have very little training overhead, which makes frequent retraining of the models a viable alternative. If a false positive is produced, we simply train the

model from scratch with the wrongly classified session included in the training dataset.

For all models, we recommend to prune old data points (older than $\approx$ 3 months) from the training datasets and retrain the models from scratch. This prevents old behaviour patterns, which are no longer exhibited by users, from being considered normal if they appear again.

5 METHODOLOGY

In this Chapter, we describe the methodology of our experiments. We conduct the experiments using logs collected on the HPC cluster JUSTUS2, which is described in Section 5.1. In Section 5.2, we introduce the metrics utilised to quantify the performance of methods for log parsing and anomaly detection. In Section 5.3, we describe the datasets we use for conducting the experiments. The procedure for performing the experiments is given in Section 5.4.

5.1 TEST SYSTEM

JUSTUS2[1] is a HPC cluster located at Ulm University. It is dedicated to computational chemistry and quantum sciences and commenced operations on 6 March 2020 with a total of 702 nodes. JUSTUS2 runs *Rocky Linux*[2] in version 8 as its operating system. *SLURM*[3] is used as the system's workload manager. The nodes in JUSTUS2 serve different purposes (see Figure 5.1). There are 692 compute nodes, which are subdivided into *standard nodes*, *fast I/O nodes*, *large fast I/O nodes* and *special nodes*. Additionally, four nodes serve as login nodes, four nodes are service nodes and two nodes are visualisation nodes. The login and visualisation nodes are connected to the internet via *BelWue*[4] (AS553), which is the data network for universities in the state of Baden-Württemberg, Germany. The system is protected by geofencing: only clients from within BelWue are permitted to login. This restricts the source location of logins to the 61 institutions participating in BelWue, which are mainly universities or other research facilities. Login is also permitted from within JUSTUS2 or from selected research institutions that are not participating in BelWue. Authentication can be performed either via password or public key.

Further, we perform all necessary computations for anomaly detection on JUSTUS2.

[1] https://www.uni-ulm.de/einrichtungen/kiz/service-katalog/high-performance-computing/justus2/

[2] https://rockylinux.org/

[3] https://slurm.schedmd.com/overview.html

[4] https://www.belwue.de/

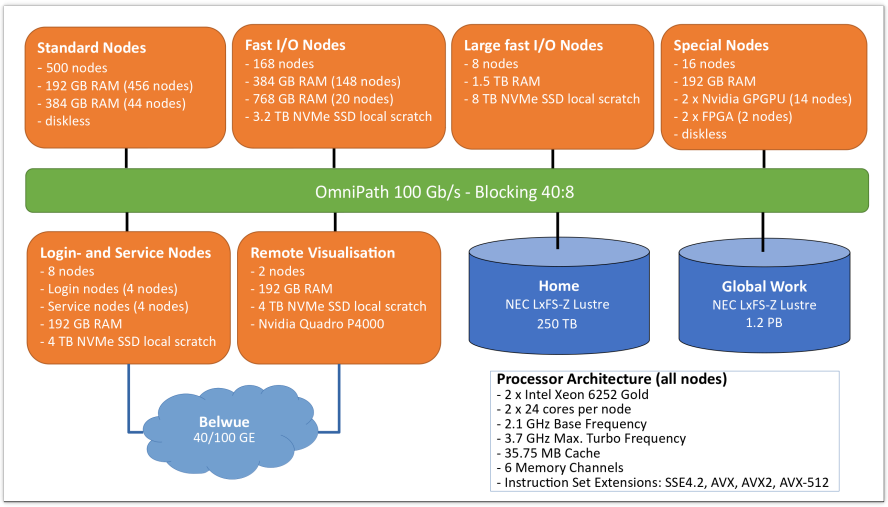


Figure 5.1: Architecture of JUSTUS2 [22]

5.2 EVALUATION METRICS

In this Section, we introduce the metrics we use for evaluating the individual parts of our proposed architecture. In Section 5.2.1, we define metrics used for evaluating log parsers; in Section 5.2.2 we introduce the metrics we use for evaluating anomaly detectors.

5.2.1 Log Parsers

For evaluating the performance of log parsers, we use the metrics of *accuracy* and *efficiency* [76]. The efficiency of log parsers is determined by measuring the time taken to parse the test dataset and dividing it by the number of log lines parsed. This gives us the average time taken to parse one log line.

Accuracy is defined as the ratio of correctly parsed log messages to the total number of log messages. During parsing, each log message's log key is extracted. For calculating the accuracy, the messages with sharing the same log key are then grouped together. If such a group contains the same log messages as in the ground truth, all of its members are considered to be parsed correctly. Otherwise, none are.

5.2.2 Anomaly Detectors

In order to evaluate methods for detecting anomalous logins, jobs and system logs, we provide numerous metrics that reveal each method's performance. We call normal logins that are predicted to be normal *true negatives* (TN) and the ones that are predicted to be abnormal *false positives* (FP). Similarly, abnormal logins that are predicted to

be abnormal are *true positives* (TP), whereas the ones predicted to be normal are *false negatives* (FN).

In addition we calculate the derived metrics of precision, recall and F1-score [14]. These are defined as

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$
$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Intuitively, the precision measures how many of the samples classified as positives are true positives. Recall describes how many of the positive samples are detected as such.

5.3 DATA

The operators of JUSTUS2 provided the authors of this work with real-world log data, which was collected over a period of 81 days. We use these data for the evaluation of the proposed concept for detecting malicious anomalies. An overview of the volume of data available to us is given in Table 5.1.

NAME	SIZE (GZIPPED)	SIZE (RAW)	#LINES
WM (Slurm)	880.0 MB	9.9 GB	8,863,641
/var/log/messages	2.9 GB	37.0 GB	289,053,199
/var/log/secure	173.0 MB	3.0 GB	28,690,929

Table 5.1: Volume of operational records of a real-world HPC system we use in evaluating the proposed SIEM.

5.3.1 Log Parsers

For evaluating the time-efficiency of various methods for log parsing, we use the first 100,000 lines found in `/var/log/messages`.

5.3.2 Anomaly Detectors

We conduct experiments to determine the performance and efficiency of the anomaly detectors introduced in Section 4.4. For that, we use the combined data from all three data sources described in Section 5.3. The following Sections outline preprocessing steps we performed in order to make the data suitable as input for the algorithms under test.

5.3.2.1 Anomalous Logins

During preprocessing, each login date is enriched with its source location. For IP-addresses belonging to an institution participating in AS553, the field *location* is set to the name of this institution (e.g. *Universitaet Ulm*, *Universitaet Stuttgart* etc.). Source IP-addresses in IP-block `10.0.0.0/8` indicate a login from within JUSTUS2, which we mark with source location *Internal*. Other source IP-addresses belong to institutions which are not part of AS553, but permitted login access to JUSTUS2 nevertheless. As these institutions and their respective IP-blocks are not public; we mark logins from such IPs with source location *Other*. This gives us a total of 63 possible values for *location*.

We encode the preprocessed data as described in Section 4.4.1.1. For encoding *time-of-day*, we chose a 5-bit Grey code. *Time-of-day* thus has 32 possible values. *Authentication method* has a total of three possible values: *keyboard-interactive/pam*, *publickey* and *hostbased*. The encoded input vector has dimension

$$\begin{aligned}\ell_{login} &= |F_{location}| + |F_{authmethods}| + |F_{dow}| + n \\ &= 63 + 3 + 7 + 5 \\ &= 78\end{aligned}$$

We find that the recorded contents of `/var/log/secure` contain 26,739 instances of successful logins from a total of 344 users. The number of logins per user ranges from 1 to 2,756. In our analysis we only consider users having more than 100 registered logins in the recorded time period. This is true for 76 users.

The available data are unlabelled, i.e. the logins are not known to be either normal or anomalous. Because, to our knowledge, JUSTUS2 suffered no security incident during the time of data collection, we assume the available data to represent normal behaviour. In order to test the detectors' capabilities in terms of detecting anomalous behaviour, we synthesise a number of irregular logins. Irregular logins differ from all recorded logins in every feature.

5.3.3 Anomalous Log Messages

For detecting anomalous Linux system logs, we use logs from the standard Linux log file `/var/log/messages`. These logs are grouped by user session as described in Section 4.3. Due to the limitations outlined in Section 4.3.2, this is not possible for all log lines: from the total of 289,053,199 lines collected in `/var/log/messages`, only 43,937,809 (15.206%) can be assigned to user sessions. The other log lines are discarded. In the following, we refer to all log messages assigned to the same user session as a *log sequence*.

The collected logs include a large amount of routine log messages. These are messages from periodic maintenance jobs, produced by monitoring tools, the workload manager, stress-test tools etc. As these messages appear regardless of user behaviour on the system, we consider them to be noise and discard them. After discarding routine log messages, we are left with a total of 9,026,210 important lines, which belong to 138,714 different log sequences.

Given that these data was collected over a time-span of 81 days, we observe that JUSTUS2 produces 1,332 log sequences per day, which is about one log sequence per minute. In order to cope with this volume of logs, an anomaly detector must be able to classify a log sequence in less than approximately one minute.

We assume the available log data to contain only normal behaviour with the same reasoning as in Section 5.3.2.1. In order to conduct a performance evaluation we generate anomalous log sequences randomly. The anomaly detection methods under test consider log keys only, disregarding the log parameters. Therefore, we only generate sequences of log keys.

We generate an anomalous sequence by choosing a random integer r. r is uniformly distributed between r_{min} and r_{max}, where r_{min} and r_{max} are the minimum/maximum observed sequence lengths in the collected log data. The r log keys in the generated sequence are chosen uniformly from the log keys observed in the collected log data.

We are aware that conducting our experiments using randomly generated log sequences as anomalous data may yield results that do not reflect real-world performance of the detectors under test. Anomalous log sequences occurring in a productive system as a result of malicious behaviour are not completely random. It is almost certain that methods for anomaly detection perform worse in differentiating them from normal log sequences as they do when differentiating random from normal log sequences. However, as we do not have access to anomalous log sequences observed in a real-world setting, using randomly generated data as anomalous data is the only way for us to compare the different anomaly detectors' performances. This approach is not unheard of in the field of machine learning (e.g. see Liang, Li, and Srikant [34]).

5.3.3.1 Anomalous Jobs

In HPC systems, data about submitted jobs are provided by the WM. In the case of our test system, JUSTUS2, the WM is Slurm. Through the operators of JUSTUS2, we acquired Slurm log files containing information about a large number of jobs that have been submitted over a time period of 81 days. The files have a total of 8,863,641 lines which hold the details of 7,832,175 jobs.

Because memory constraints prevent us from loading all data into memory at once, we decide to perform our experiments on a subset of

the log files, i.e. a period of 18 days during which 871,877 jobs were submitted. Within this period, 58 different research groups were active on the cluster, each of which submitted between 3 and 534,945 jobs.

We perform the preprocessing and feature selection steps discussed in Section 4.4.4.1. Performing PCA with automatic dimensionality selection reduces the number of features from 22 to 21. In order to have at least five examples per dimension for training the machine learning models, we decide to only consider research groups having submitted a sufficient number of jobs for the training set to be larger than $21 \cdot 5 = 105$. This amounts to 14 groups.

For the evaluation of models detecting anomalous logins and logs in Section 7.2 and Section 7.3, we faced the problem that data representing normal behaviour were abundantly available, but data representing anomalous behaviour were not. Because of that, we resorted to generating anomalous data points randomly. This, however, means that the model performances we measures might not be transferable into real-world settings, because real anomalous data might look significantly different from randomly generated data. Using random data is therefore a last-resort that we use to be able to compare the models under test, but not to make any predictions on the real-world performance of these models. In the case of detecting anomalous jobs however, we do not need to generate anomalous data randomly. Instead, we test the performance of the models by defining one group's jobs as normal and the jobs of all other groups as anomalous.

5.4 PROCEDURE

5.4.1 Log Parsers

We compare several methods for log parsing. In order to be a candidate for usage in the SIEM we propose, a parser must meet the following criteria:

- It must be able to operate in online mode, i.e. process incoming log messages immediately without the need to await further input. It must be able to operate on an (potentially infinite) stream of log lines.
- It must achieve a high accuracy while parsing Linux system logs.
- It must be efficient enough to cope with the amount of logs produced during operation of a typical HPC system.

From our test data (see Section 5.1) we calculate that JUSTUS2 produces 41.303 log messages per second on average. In order to be able to keep up with this stream of logs, a log parser must process logs in less than 24.211 ms per line.

We compare the following methods for automatic log parsing: *SLCT* [66], *AEL* [29, 30], *IPLoM* [42, 43], *LKE* [17], *LFA* [49], *LogSig* [64], *SHISO* [47], *LogCluster* [67], *LenMa* [60], *Spell* [13], *Drain* [25], *LogMine* [21] and *MoLFI* [45]. Zhu et al. [76] compare these log parsers in terms of efficiency, mode and accuracy. We rely on their results for comparing log parsers in terms of mode and accuracy. The reason for this is that in order to calculate a parser's accuracy on a particular dataset, one needs to possess a ground truth, i.e. a structured version of the dataset. Unfortunately, we do not possess a structured version of Linux logs from our test system and lack the resources to produce a structured version of a log file of sufficient size by hand.

For determining the efficiency of the parsers, we perform our own measurements.

Zhu et al. [76] report that most of the parsers work in offline mode. Only SHISO, LenMa, Spell and Drain are online parsers and therefore suitable for the scope of this thesis. In order to select a suitable parser for parsing large amounts of Linux log files, we compare the four online parsers by accuracy and efficiency. For the accuracy, we rely on the results of Zhu et al. [76]. For the efficiency, we perform our own measurements. Zhu et al. determine the accuracy of each parser by running it on 16 different log datasets. These log datasets stem from a wide range of applications, e.g. distributed systems, mobile systems, operating systems and supercomputers. Fortunately, a dataset consisting of Linux log files is included. As these are the only type of log we are interested in parsing, we only consider the accuracy achieved when parsing Linux logs.

In order to determine the efficiency of the parsers under test, we parse a 100,000 line excerpt of `/var/log/messages` from our test system and measure the time taken.

5.4.2 Anomaly Detectors

In this Section, we lay out the general procedure for evaluating anomaly detectors. The anomaly detectors proposed in Section 4.4 can make use of different algorithms internally. The goal of this evaluation is to find out which algorithms are suited best for each detector.

Each evaluation consists of the following steps:

1. Define three datasets $X_{training}$, X_{test} and $X_{validation}$.
2. Define a hyper-parameter search space for each algorithm under test.
3. Perform hyper-parameter optimisation for each algorithm under test using $X_{training}$, X_{test} and the search space from the previous step.

4. Perform a performance evaluation for each algorithm under test using $X_{training}$, $X_{validation}$ and the hyper-parameters found to be optimal in the previous step.

The hyper-parameter optimisation for a particular algorithm is always done in a similar way: perform an exhaustive search over the entire search space. We train the algorithm with every possible combination of hyper-parameters in the search space using $X_{training}$ and calculate the respective F1-scores using X_{test}. The combination of hyper-parameters yielding the highest F1-score is called optimal and selected for the performance evaluation.

The performance evaluation for a particular algorithm consists of training the algorithm with $X_{training}$ and calculating the aforementioned performance metrics using $X_{validation}$.

In the following Sections, we explain the process of defining $X_{training}$, X_{test} and $X_{validation}$ and calculating the performance metrics for the evaluation of each anomaly detector. Further, we name the algorithms under test.

5.4.2.1 Anomalous Logins

We define the training, test and validation datasets as follows: Let $X_{regular}$ be the set of collected logins. We generate $X_{irregular}$, a set of irregular login data, such that $|X_{irregular}| = 0.25 \cdot 0.15 \cdot |X_{regular}|$.

$X_{training}$ is build by randomly selecting $|X_{regular}| \cdot 0.75$ samples from $X_{regular}$. X_{test} and $X_{validation}$ are then generated by randomly splitting $(X_{regular} \setminus X_{training}) \cup X_{irregular}$ into equally-sized parts.

We work under the assumption that each user has a stable login behaviour. However, the login behaviour of different users may differ significantly, and so does the notion of an anomalous login. For this reason, we perform the following steps separately for all users. We collect the respective number of true/false positives/negatives for each novelty detection method m on the collected logins of user u: $(TP_{m,u}, FP_{m,u}, TN_{m,u}, FN_{m,u})$. For evaluation, we calculate the number of true/false positives/negatives per novelty detection method m:

$$
\begin{aligned}
TP_m &= \sum_u TP_{m,u} \\
FP_m &= \sum_u FP_{m,u} \\
TN_m &= \sum_u TN_{m,u} \\
FN_m &= \sum_u FN_{m,u}
\end{aligned}
$$

Using (TP_m, FP_m, TN_m, FN_m) we calculate the precision, recall and F1-score of each novelty detection method m.

We evaluate the viability of four different novelty detection methods for detecting anomalous logins. The methods under test are Isola-

tion Forest, OCSVM, OCSVM-SGD and LOF. LOF is tested with six commonly used distance metrics, namely

1. Bray-Curtis dissimilarity [6]
2. Chebyshev distance [48]
3. Correlation [51]
4. Cosine distance [62]
5. Hamming distance [68, p. 206]
6. Minkowski distance [33]

OCSVM-SGD supports online updates of its underlying model. If normal data points are mistakenly classified as anomalous, the model can be adapted on-the-fly, such that these points are recognised as normal afterwards. Crucially, this adaptation does not require retraining the model from scratch and is therefore relatively fast. We include a version performing this adaptation under the name *OCSVM-SGD-adapt* in our experiments. False positives produced by this algorithm are still counted as such in our evaluation. However, after misclassifying a login as anomalous, the model is adjusted as to include this point into its model of normal behaviour. After this adjustment, similar logins are no longer classified as anomalous, leading to fewer false positives overall.

We perform hyper-parameter optimisation using the following candidate values:

- Isolation Forest: t: $\{10, 20, 30, 50, 100, 200, 500, 1000\}$
- OCSVM / OCSVM-SGD / OCSVM-SGD-adapt:
  - Kernel function: $\{\text{linear}, \text{polynomial}, \text{radial basis}, \text{sigmoid}\}$
  - γ: $\{\text{scale}, \text{auto}, 0.1, 0.2, 0.3, 0.4, 0.6, 0.8\}$
  - ν: $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$

The options *auto* and *scale* for hyper-parameter γ mean that γ's value is dependent on the training data. When using *auto*, $\gamma = \frac{1}{\ell_{login}} = \frac{1}{78}$. When using *scale*, $\gamma = \frac{1}{\ell_{login} \cdot Var(X_{training})}$.

5.4.2.2 Anomalous Log Messages

During our experiments we noticed that the anomaly detection models under test require a larger number of training samples than the novelty detection methods investigated in Section 7.2. Therefore, we use a larger proportion of the collected data for training.

The set X_{normal} contains 138,714 log sequences, $X_{anomalous}$ consists of 10,000 randomly generated log sequences. 20% of X_{normal} are randomly assigned to $X_{training}$. $(X_{normal} \setminus X_{training}) \cup X_{anomalous}$ is then randomly split into X_{test} and $X_{validation}$, such that $|X_{test}| = |X_{validation}|$.

The algorithms under test are DeepLog, LogCluster, PCA and Invariants Mining, which we introduce in Section 3.1.2. We train DeepLog for 300 epochs. This value is adopted from [72]. For hyper-parameter optimisation we consider the following candidate values:

- DeepLog:
  - h: $\{3 \dots 12\}$
  - g: $\{3 \dots 39\}$
  - L: $\{1 \dots 5\}$
  - α: $\{32, 64, 128, 256\}$
- LogCluster:
  - θ: $\{0.1, 0.2, 0.3, 0.4\}$
  - δ: $\{0.1, 0.2, 0.3, 0.4\}$
- PCA:
  - k: $\{1 \dots 18\}$
  - β: $\{0.00001, 0.0001, 0.0005, 0.001, 0.003, 0.005, 0.01, 0.05, 0.08\}$
- Invariants Mining:
  - s: $\{0.96, 0.96, 0.97, 0.98, 0.99, 0.995, 0.999, 1.0\}$
  - ϵ: $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$

5.4.2.3 *Anomalous Jobs*

In order to evaluate the methods for novelty detection for the task at hand, we perform the following steps for each research group: We define X_{normal} as the jobs observed for this group and $X_{anomalous}$ as the jobs observed for all other groups. We select at random $0.1 \cdot |X_{normal}|$ data points from X_{normal} and define the set of these points as $X_{training}$. We further select at random $0.15 \cdot |X_{normal} \setminus X_{training}|$ from $X_{anomalous}$ and define the set of these points as $X'_{anomalous}$. When then define X_{test} and $X_{validation}$ as a random split of $X'_{anomalous} \cup (X_{normal} \setminus X_{training})$ such that $2 \cdot |X_{test}| = |X_{validation}|$.

Both for hyper-parameter search and for the computation of the performance, we gather the respective number of true/false positives/negatives for each novelty detection method m on the collected

jobs of research group g: $(TP_{m,g}, FP_{m,g}, TN_{m,g}, FN_{m,g})$. Then, we calculate the number of true/false positives/negatives for every m:

$$TP_m = \sum_g TP_{m,g}$$
$$FP_m = \sum_g FP_{m,g}$$
$$TN_m = \sum_g TN_{m,g}$$
$$FN_m = \sum_g FN_{m,g}$$

Using (TP_m, FP_m, TN_m, FN_m) we calculate the precision, recall and F1-score of each novelty detection method m.

The novelty detection methods under test are Isolation Forest, OCSVM, OCSVM-SGD and LOF. LOF is tested with five different distance metrics, which are commonly used in novelty detection:

1. Bray-Curtis dissimilarity [6]
2. Chebyshev [48]
3. Correlation [51]
4. Cosine [62]
5. Minkowski [33]

We perform hyper-parameter optimisation using the following candidate values:

- Isolation Forest: t: $\{10, 20, 30, 50, 100, 200, 500, 1000\}$
- OCSVM / OCSVM-SGD:
  - Kernel function: {linear, polynomial, radial basis, sigmoid}
  - γ: $\{\text{scale}, \text{auto}, 0.1, 0.2, 0.3, 0.4, 0.6, 0.8\}$
  - ν: $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$

We perform two experiments, which differ in the data used for training the models for novelty detection: In the first experiment, we train using the entirety of $X_{training}$. In the second experiment, we randomly select a maximum of $\kappa = 1,000$ data points from $X_{training}$, which we use to train the model. This is done to explore the relationship between the size of the training dataset and the performance and efficiency of the resulting models.

6 IMPLEMENTATION

In this Section, we describe the technologies used for implementing the individual elements of the proposed SIEM. All code is available on GitHub[1,2]

6.1 PREPROCESSING & SESSION GROUPING

6.1.1 Extracting Logins

In order to extract logins from `/var/log/secure`, we developed a programme in the programming language Go[3]. Apart from parsing `/var/log/secure`, it also determines the source location of each login. As we perform our experiments on the JUSTUS2 cluster which is only accessible from within BelWue, we use the (publicly available) list of BelWue's IP-Blocks[4] to find out the location of a given source IP. This does not reduce the applicability of our approach to other clusters, because determining the source location of a login can be achieved using any method for geolocation.

6.1.2 Parsing System Logs

As discussed in Section 8.1.1, we chose Drain for the task of parsing Linux log files. This decision is partly based on the evaluation of automatic log parsers of Zhu et al. [76], and partly on our own performance measurements of these parsers. We perform these measurements using the *logparser* toolkit provided by Zhu et al. [76].

For further research, we recommend IBM's implementation of Drain, which is called Drain3 [5]. This implementation provides many useful features, e.g. a comfortable installation process, persistence and overall user friendliness.

6.1.3 Session Grouping

We implemented an algorithm for assigning log messages to user sessions (see Section 4.3.1.1) in the programming language Go.

[1] https://github.com/gutjuri/anomaly-detection/tree/v1.0.1
[2] https://doi.org/10.5281/zenodo.6845610
[3] https://go.dev/
[4] https://ipinfo.io/AS553
[5] https://github.com/IBM/Drain3

6.2 ANOMALY DETECTORS

In this Section, we provide information on the implementation of the anomaly detectors. The detectors for anomalous logins and anomalous jobs are implemented as separate programmes, but using the same technologies; we discuss those in Section 6.2.1. The implementation of the anomalous log messages detector is described in Section 6.2.1. Because of time constraints, we were not able to implement the anomalous movement detector.

6.2.1 Anomalous Logins & Anomalous Jobs

The anomalous logins- and the anomalous jobs detector are implemented using the Python[6] programming language. The algorithms for feature encoding, novelty detection and the implementation of performance metrics are provided by the library scikit-learn[7] [52].

6.2.2 Anomalous Log Messages

We use existing implementations of the algorithms for anomaly detection from system logs described in Section 3.1.2. He et al. [26] provide *loglizer*[8], a toolkit that implements various anomaly detection algorithms and is intended for use in research. Among others, it contains implementations of LogCluster, PCA and InvariantsMiner, which we use for our experiments. Unfortunately, it does not (yet) implement DeepLog. Indeed, no freely accessible, complete implementation of DeepLog is, to our knowledge, currently available for research. Therefore, we resort to the implementation of Wu [72], which only realises execution path anomaly detection. Parameter value and performance anomaly detection as well as online updating of models is not supported. Despite that, this implementation's performance is on par with the performance stated by Du et al. [14].

[6] https://www.python.org/
[7] https://scikit-learn.org/
[8] https://github.com/logpai/loglizer

7 RESULTS

In this Chapter, we present the results of our experiments. A discussion of these results can be found in Chapter 8.

7.1 LOG PARSING

In this Section, we present the results of evaluating various methods for automatic log parsing in terms of time-efficiency, mode and accuracy.

The evaluation of mode and accuracy was performed by Zhu et al. [76]. They determine that only SHISO [47], LenMa [60], Drain [25] and Spell [13] support online parsing; we thus omit the remaining parsers from further evaluation. The results in terms of accuracy and efficiency are given in Table 7.1.

Spell has the lowest accuracy of all online parsers, while SHISO, LenMa and Drain achieve a roughly similar one.

7.2 DETECT ANOMALOUS LOGINS

In this Section, we present the results of evaluating numerous novelty detection algorithms for detecting anomalous logins. Section 7.2.1 covers the results of the hyper-parameter search, Section 7.2.2 covers performance and efficiency.

7.2.1 Hyper-Parameter Search

We perform the search for optimal hyper-parameters as laid out in Section 5.4.2.

The best-performing hyper-parameters are:

PARSER	ACCURACY [76]	MS/LINE
SHISO	0.701	5.13
LenMa	0.701	6.52
Spell	0.605	0.21
Drain	0.690	0.22

Table 7.1: Accuracy and efficiency achieved by online parsers while parsing Linux logs.

METHOD	TP	FP	TN	FN
LOF-chebyshev	79	190	2093	347
iForest	412	1468	815	14
OCSVM-SGD-adapt	167	36	2247	259
LOF-minkowski	230	57	2226	196
OCSVM	426	381	1902	0
LOF-braycurtis	393	93	2190	33
LOF-cosine	391	86	2197	35
LOF-correlation	391	86	2197	35
LOF-hamming	381	65	2218	45
OCSVM-SGD	425	49	2234	1

Table 7.2: Number of true/false positive/negative predictions of various novelty detection methods when detecting anomalous logins. The dataset used for this evaluation contains 426 positive and 2822 negative examples.

- Isolation Forest: $t = 50$
- OCSVM: sigmoid kernel function, $\gamma = 0.4$, $\nu = 0.1$
- OCSVM-SGD: radial-basis kernel function, $\gamma = 0.1$, $\nu = 0.1$
- OCSVM-SGD-adapt: radial-basis kernel function, $\gamma = 0.2$, $\nu = 0.2$

Tables containing the results of all combinations of hyper-parameters tested can be found in Section A.1.

7.2.2 Performance & Efficiency

The performance and efficiency of the different novelty detection methods under test can be found in Table 7.2 and Table 7.3. Table 7.2 shows the absolute numbers of true/false positives/negatives for each detection method. Table 7.3 shows the precision, recall and F1-score achieved by each detection method as well as their runtimes. The runtimes are given *per data point*.

Figure 7.1 shows confusion matrices for the novelty detection methods of Isolation Forest, LOF (with the Hamming distance as the distance metric), OCSVM and OCSVM-SGD, which have been normalised over the predicted values.

The results presented in this Section are further discussed in Section 8.1.2.1.

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
LOF-chebyshev	0.294	0.185	0.227	0.039 ms	0.024 ms
iForest	0.219	0.967	0.357	0.256 ms	0.187 ms
OCSVM-SGD-adapt	0.823	0.392	0.531	0.040 ms	0.663 ms
LOF-minkowski	0.801	0.540	0.645	0.006 ms	0.008 ms
OCSVM	0.528	1.000	0.691	0.005 ms	0.002 ms
LOF-braycurtis	0.809	0.923	0.862	0.049 ms	0.028 ms
LOF-cosine	0.820	0.918	0.866	0.011 ms	0.012 ms
LOF-correlation	0.820	0.918	0.866	0.017 ms	0.018 ms
LOF-hamming	0.854	0.894	0.874	0.019 ms	0.019 ms
OCSVM-SGD	0.897	0.998	0.944	0.014 ms	0.006 ms

Table 7.3: Performance of different novelty detection methods for detecting anomalous logins. t_{train} and $t_{predict}$ are *per data point*.

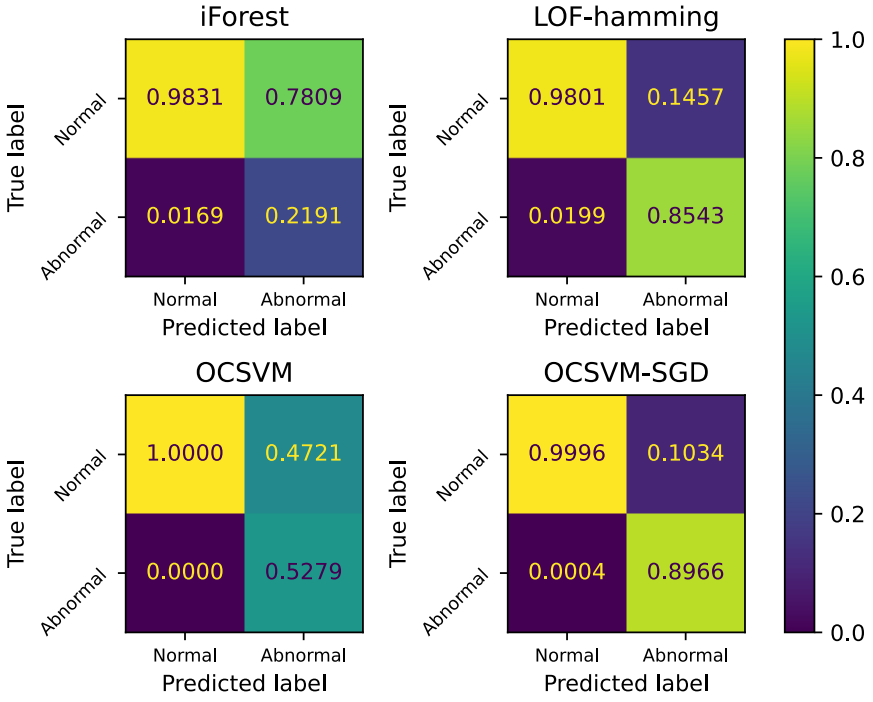


Figure 7.1: Confusion matrices comparing different novelty detection methods when distinguishing normal and abnormal logins.

7.3 DETECT ANOMALOUS SYSTEM LOGS

In this Section, we present the results of our experiments comparing various methods for detecting anomalous Linux system log sequences. In Section 7.3.1, we present the results of the hyper-parameter search. Section 7.3.2 gives the performances and efficiencies of the tested algorithms.

7.3.1 Hyper-Parameter Search

We determine that the hyper-parameters leading to be highest F1-scores are:

- DeepLog: $h = 7$, $g = 39$, $L = 1$, $\alpha = 64$
- LogCluster: $\theta = 0.1$, $\delta = 0.4$
- PCA: $k = 3$, $\beta = 0.0001$
- Invariants Mining: $s = 1.0$, ϵ has no impact.

We note that the choice of hyper-parameters for DeepLog is not only influenced by the performance of the resulting model, but also by resource constraints. The memory requirements for the training phase are immense and increase with L and α. We thus cannot train models with $L \cdot \alpha \gtrsim 300$, as this would require more memory than is available in our test system.

The performance of all tested hyper-parameter combinations can be found in Section A.2.

7.3.2 Performance & Efficiency

In this Section, we present the performances of DeepLog, LogCluster, PCA and Invariants Mining using the best-performing hyper-parameters on the dataset described in Section 5.3.3. The absolute number of true/false positives/negatives can be found in Table 7.4; Table 7.5 shows precision, recall, F1-score and runtimes per data point of the individual detection methods. Figure 7.2 shows confusion matrices of the four models under test. The results are further discussed in Section 8.1.2.2.

7.4 DETECT ANOMALOUS JOBS

In this Section, we present the results of testing several algorithms for detecting anomalous jobs. In Section 7.4.1 we present the most suitable hyper-parameters, which we determined empirically. In Section 7.4.2, we present the performance and efficiency of the algorithms under test.

METHOD	TP	FP	TN	FN
DeepLog	5,024	23,125	18,857	0
PCA	4,976	110	55,352	48
Invariants Mining	5,024	61	55,401	0
LogCluster	5,024	52	55,410	0

Table 7.4: Number of true/false positive/negative predictions of various methods for detecting anomalous log sequences. The dataset used for this evaluation contains 5,024 positive and 55,462 negative examples.

METHOD	PREC.	REC.	F1	t_{train}	$t_{predict}$
DeepLog	0.178	1.000	0.303	122.940 ms	4.657 ms
PCA	0.978	0.990	0.984	0.006 ms	0.050 ms
Invariants Mining	0.988	1.000	0.994	14.477 ms	0.001 ms
LogCluster	0.990	1.000	0.995	2.569 ms	3.380 ms

Table 7.5: Performance of different anomaly detection methods for detecting anomalous log sequences. t_{train} and $t_{predict}$ are *per log sequence*.

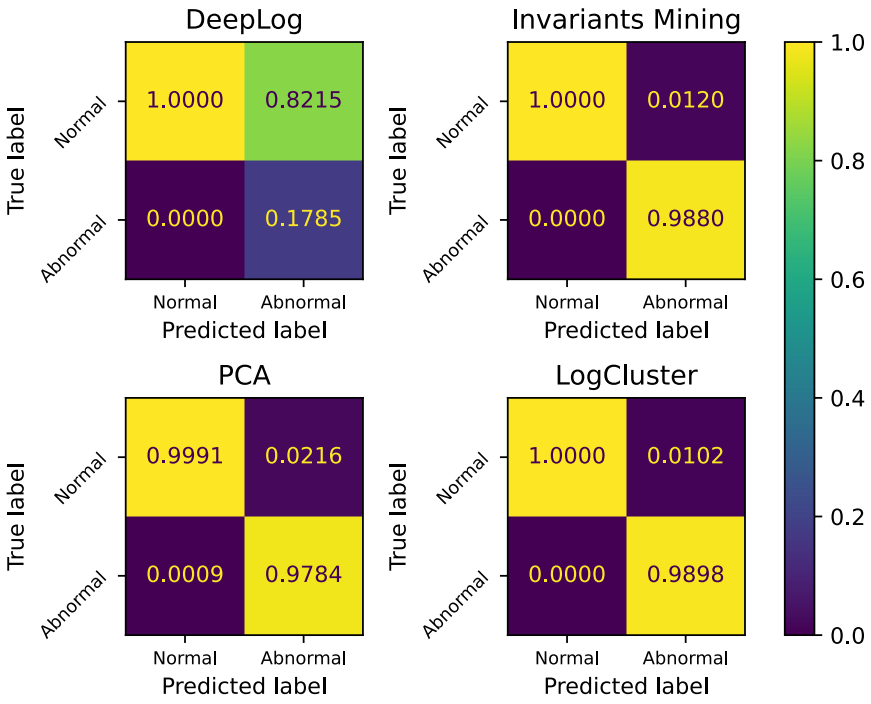


Figure 7.2: Confusion matrices comparing different anomaly detection methods when distinguishing normal and abnormal log sequences.

METHOD	TP	FP	TN	FN
LOF-cosine	76,595	68,412	454,805	2,076
LOF-correlation	76,603	68,073	455,144	2,068
iForest	74,969	64,091	459,126	3,702
OCSVM	77,751	54,761	468,456	9,20
LOF-minkowski	70,910	20,818	502,399	7,761
LOF-chebyshev	71,260	20,274	502,943	7,411
LOF-braycurtis	76,457	19,958	503,259	2,214
OCSVM-SGD	75,239	17,115	506,102	3,432

Table 7.6: Number of true/false positive/negative predictions of various methods for novelty detection under the task of detecting anomalous jobs. The dataset used for this evaluation contains 86,500 positive and 575,537 negative examples.

7.4.1 Hyper-Parameter Search

The following hyper-parameters lead to the highest F1-scores:

- Isolation Forest: $t = 30$
- OCSVM: radial basis kernel function, $\gamma = 0.2$, $\nu = 0.1$
- OCSVM-SGD: radial basis kernel function, $\gamma = 0.2$, $\nu = 0.1$

We give the performances of all tested hyper-parameter combinations in Section A.3.

7.4.2 Performance & Efficiency

For evaluating algorithms for detecting anomalous jobs, we perform two experiments comparing Isolation Forest, OCSVM, OCSVM-SGD and LOF. In the first experiment, we train the models for anomaly detection using the entirety of $X_{training}$. The results of this experiment can be found in Table 7.6, Table 7.7 and Figure 7.3.

In the second experiment, we train the models for anomaly detecting with at maximum $\kappa = 1,000$ examples. The results of this experimental run can be found in Table 7.8, Table 7.9 and Figure 7.4.

METHOD	PREC.	REC.	F1	t_{train}	$t_{predict}$
LOF-cosine	0.528	0.974	0.685	0.819 ms	0.548 ms
LOF-correlation	0.529	0.974	0.686	0.975 ms	0.652 ms
iForest	0.539	0.953	0.689	0.007 ms	0.005 ms
OCSVM	0.587	0.988	0.736	0.306 ms	0.111 ms
LOF-minkowski	0.773	0.901	0.832	0.122 ms	0.081 ms
LOF-chebyshev	0.779	0.906	0.837	0.733 ms	0.488 ms
LOF-braycurtis	0.793	0.972	0.873	0.999 ms	0.665 ms
OCSVM-SGD	0.815	0.956	0.880	0.002 ms	0.001 ms

Table 7.7: Performance of different novelty detection methods for detecting anomalous jobs with. t_{train} and $t_{predict}$ are *per job*.

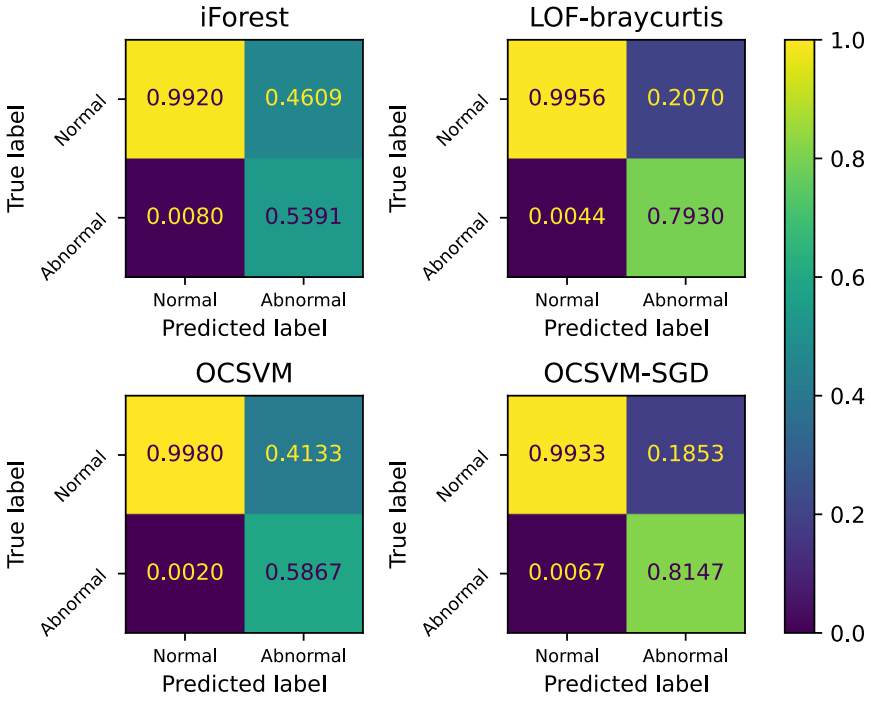


Figure 7.3: Confusion matrices comparing different novelty detection methods for detecting anomalous jobs.

METHOD	TP	FP	TN	FN
LOF-cosine	85,916	105,926	469,611	584
LOF-correlation	85,926	105,729	469,808	574
OCSVM	85,559	62,888	512,649	941
iForest	82,757	57,616	517,921	3,743
LOF-minkowski	83,842	42,996	532,541	2,658
LOF-chebyshev	84,825	43,044	532,493	1,675
LOF-braycurtis	85,766	42,123	533,414	734
OCSVM-SGD	82,983	13,668	561,869	3,517

Table 7.8: Number of true/false positive/negative predictions of various novelty detection methods at the task of detecting anomalous jobs. For this experiment, the size of $X_{training}$ has been capped to $\kappa = 1,000$. The dataset used for this evaluation contains 86,500 positive and 575,537 negative examples.

METHOD	PREC.	REC.	F1	t_{train}	$t_{predict}$
LOF-cosine	0.448	0.993	0.617	0.029 ms	0.017 ms
LOF-correlation	0.448	0.993	0.618	0.021 ms	0.019 ms
OCSVM	0.576	0.989	0.728	0.008 ms	0.003 ms
iForest	0.590	0.957	0.730	0.047 ms	0.005 ms
LOF-minkowski	0.661	0.969	0.786	0.008 ms	0.004 ms
LOF-chebyshev	0.663	0.981	0.791	0.022 ms	0.015 ms
LOF-braycurtis	0.671	0.992	0.800	0.032 ms	0.020 ms
OCSVM-SGD	0.859	0.959	0.906	0.041 ms	0.001 ms

Table 7.9: Performance of different novelty detection methods for detecting anomalous jobs with $\kappa = 1,000$. t_{train} and $t_{predict}$ are *per job*.

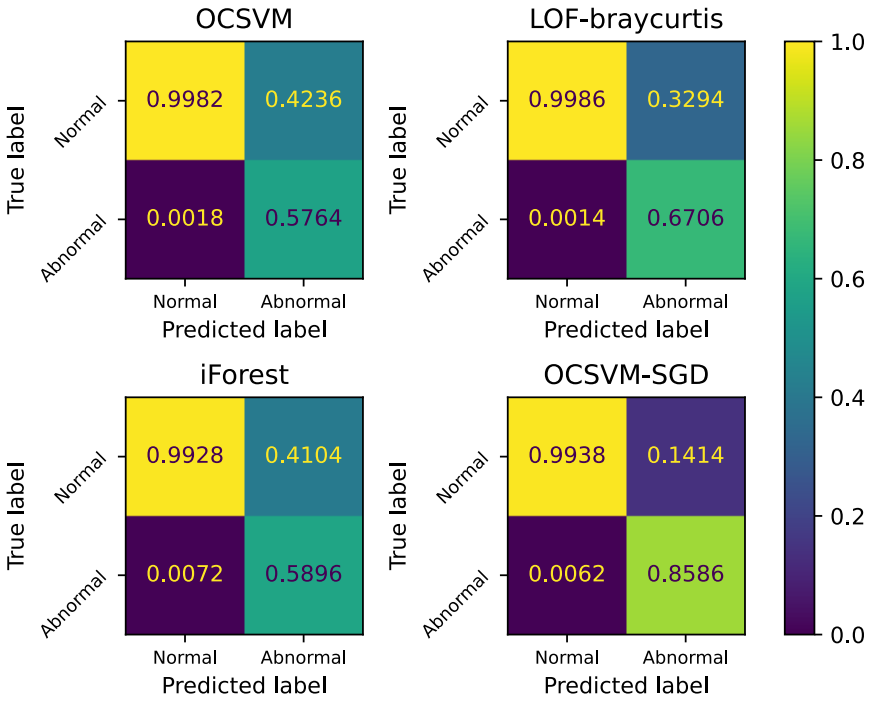


Figure 7.4: Confusion matrices comparing different novelty detection methods for detecting anomalous jobs ($\kappa = 1,000$).

8 DISCUSSION

In this Section, we discuss the results and limitations of our work. Section 8.1 contains discussion of the results we acquired, Section 8.2 points out limitations of the approach as well as threats to the validity of our results. Section 8.3 discusses possible directions of future research.

8.1 FINDINGS

In this Section, we discuss the results of the experiments conducted for this work. Section 8.1.1 – Section 8.1.2 deals with the results of the experimental evaluation of the elements of the proposed SIEM. In Section 8.1.3 we provide answers to the research questions posed in Section 1.2.

8.1.1 Log Parsers

In this Section, we discuss the results of experimenting with numerous methods for automatic log parsing in terms of mode, time-efficiency and accuracy. The results are given in Section 7.1.

The only parsers supporting online parsing are SHISO [47], LenMa [60], Drain [25] and Spell [13], which are thus the only candidates suited for usage in the SIEM we propose.

The performances of these parsers are given in Table 7.1. In terms of accuracy, automatic parsers generally perform relatively moderate on Linux logs: the highest accuracy reported is 0.701, which is achieved by both SHISO and LenMa. This is the highest value among online and offline parsers, meaning that we cannot expect accuracy gains by dropping the requirement for a parser to operate in an online manner. An explanation for the low performance is that Linux logs contain log entries from a diverse set of programmes, leading to a large number of possible log events. A large, diverse set of possible log message templates, some of which might be very similar to each other, makes it hard for parsers to assign a log message correctly to its template. While this fact reduces the accuracy of automatic log parsers, it also makes their usage more attractive, because the alternative (writing a parser by hand and maintaining it) becomes infeasible as the number of possible log events grows large.

In terms of efficiency, all tested online parsers are able to parse log lines in under 24.211 ms on average. However, we still observe a gap in efficiency between Spell and Drain, and SHISO and LenMa. SHISO and LenMa are parsing log lines more than one order of magnitude slower than Spell and Drain.

All in all, we believe that Drain is the most suitable choice for parsing Linux logs. Its accuracy is lower than SHISO's and LenMa's, but only to a negligible extend. It is, however, significantly more efficient than its more accurate counterparts and can thus be deployed in systems that produce up to two orders of magnitude more logs than our test system.

8.1.2 Anomaly Detectors

In this Section, we discuss the results of evaluating algorithms for usage in the anomaly detectors we propose in Section 4.4. We find the following algorithms to be suited best:

- Anomalous Logins Detector: OCSVM-SGD
- Anomalous Log Messages Detector: LogCluster
- Anomalous Jobs Detector: LOF using Bray-Curtis dissimilarity as a distance metric

8.1.2.1 Anomalous Logins

In our experiments, we determined the precision, recall, F1-score and time-efficiency of several algorithms for novelty detection in the task of detecting anomalous logins. The raw results are given in Table 7.2 and Table 7.3.

We observe that OCSVM-SGD has the highest precision, recall and F1-score while keeping processing time acceptably low. Thus, we declare OCSVM-SGD using the hyper-parameters given above the most appropriate novelty detection method for detecting anomalous logins.

Isolation Forest has the lowest F1-score. This is potentially due to the high dimension of the input vector. It also requires the most time for training and prediction. This is surprising, as the number of trees in the Isolation Forest we use is rather low ($t = 50$). However, increasing t does not improve the precision or recall.

The performance of LOF depends heavily on the distance metric used. As we are operating on binary vectors, Chebychev distance performs worst. Using the Hamming distance between two vectors as a distance metric gives the best results.

OCSVM performs worse in terms of F1-score and better in terms of computational requirements than OCSVM-SGD, even though the latter is an approximation of the former. This is due to the fact that

the tested instances of OCSVM and OCSVM-SGD do not use the same parameter γ or even the same type of kernel function. Still, the best version of OCSVM has a lower F1-score than the best version of OCSVM-SGD.

All methods under test have a high true negative and a low false negative rate, backing up the assumption that each user's login behaviour is relatively homogenous. We observe that both Isolation Forest and OCSVM achieve low precision, i.e. classify a large number of negative examples as positives. In a real-world setting, this is undesirable, since each sample predicted to be positive can lead to a request for manual action of an operator. A large number of false positives can lead to *alert fatigue*, which describes a condition in which true positives can no longer be dealt with effectively because they get lost in noise [23]. OCSVM-SGD performs well. It achieves a near perfect true negative rate and an acceptable true positive rate. Its false negative rate is near zero. This is important, because false negatives could result in undetected anomalous logins, potentially leading to an attack remaining unnoticed. Interestingly, the adaptive version of OCSVM-SGD performs significantly worse than the non-adaptive version. It unexpectedly does not have superior precision. Further, its recall is significantly lower than that of the non-adaptive version.

All in all, we are satisfied with the performance of OCSVM-SGD and choose it for detecting anomalous logins.

8.1.2.2 Anomalous Log Messages

We observe that all methods except DeepLog [14] (LogCluster [35], Invariants Mining [37] and PCA [73]) achieve very high F1-scores (see Table 7.4 and Table 7.5). They show almost perfect precision and recall. LogCluster and Invariants Mining have no false negatives, i.e. all anomalous log sequences are reported as such. This is very desirable, because every missed anomaly could result in an attack that is not noticed by HPC operators. Further, LogCluster and Invariants Mining have a low number of false positives. Though not as critical as a low number of false negatives, a low number of false positives is desirable as well. Each log sequence predicted to be positive should be investigated by HPC staff. False positives thus produce an unnecessary workload for HPC operators and can lead to alert fatigue.

DeepLog however does not perform as good. On the one hand, all anomalous log sequences are classified as such, leading to a high recall score. Precision, on the other hand, is very low, because most normal log sequences are classified as anomalous as well. As seen in Figure 7.2, only 17.85% of the log sequences predicted as anomalous are actually anomalous. In productive use, such a high number of false positives would lead to a tremendous workload placed on operators. This low precision is unexpected, as Du et al. [14] report a very promising performance. We explain this discrepancy between the

performance we observe and the performance reported by Du et al. with the different datasets used in the evaluation. Du et al. perform experiments using HDFS and OpenStack logs, which contain 28 and 40 distinct log keys respectively. We, however, use a dataset consisting of Linux logs, which contains 415 distinct log keys. This large number of log keys makes it difficult for DeepLog's neural network to learn complex relationships between observed log sequences and the most likely subsequent log keys. It might be possible to increase DeepLog's performance by selecting large values L and α, but this increases memory requirements to a point where training the model becomes impossible on current hardware. As Du et al. [14] use a much smaller dataset for training than us, yet achieve better results, we doubt that using a larger fraction of our collected logs as training data would increase performance.

All methods for anomaly detection under test are able to classify log entries very efficiently. As discussed in Section 5.3.3, an anomaly detector can only keep up with the logs produced on JUSTUS2 if classification of a log sequence takes less than one minute. We observe that all methods for anomaly detection are well below this threshold.

Training the models requires significantly more time than prediction. We train the anomaly detection models under test with $X_{training}$ (recall that $|X_{training}| = 27,742$). DeepLog finishes training after 3.410 s or approximately 57 minutes, which is quite moderate. In principle, this efficiency allows frequent retraining in order to fit the anomaly detection model to changing user behaviour. However, this is not necessary, because DeepLog models support online updates, meaning that the computationally expensive training phase needs to be performed only once. If operators notice changing user behaviour, i.e. previously unseen log sequences that are erroneously classified as anomalies, the model can be adapted on-the-fly with minimal performance penalties. This means that an even an initial training phase longer that the one observed would be acceptable. The detection performance of DeepLog using the set of hyper-parameters discussed above is not satisfying. Additionally, the durations of DeepLog's training and prediction phases scale with the number of layers L and the number of memory units per layer α. A model with larger values for L and α might therefore provide better detection performance, but it is unclear whether it would be able to do so in an acceptable time-frame.

LogCluster and Invariants Mining are very efficient: training the anomaly detection models took 71.27 s and 401.63 s respectively. These models do not support online updates. In order to keep up with changing user behaviour, they instead need to be retrained periodically using recent system logs. Given that training LogCluster and Invariants Mining takes a few minutes at most, we conclude that this poses no problem and that these two anomaly detection methods are

well-suited for the task of detecting anomalous system logs in HPC systems.

PCA is the most efficient method and completes its training phase in 0.158 s. This makes retraining very cheap. Unlike LogCluster, Invariants Mining and DeepLog, it does not achieve perfect recall.

We conclude that LogCluster is the best-performing method for detecting anomalous log sequences in Linux system logs. It achieves a higher F1-score than all other tested methods and is sufficiently efficient to cope with high volumes of logs. Further, the training phase of LogCluster only takes a few minutes, allowing retraining when user behaviour changes. We thus use LogCluster for classifying logs as normal or anomalous.

8.1.2.3 Anomalous Jobs

We first discuss the results achieved without a limit on the training dataset's size. Afterwards, we discuss the results achieved with $\kappa = 1,000$. We finally compare the two approaches and conclude that anomaly detectors using a limited set of training data do not necessarily perform worse than detectors train on large datasets.

UNLIMITED TRAINING SET SIZE As shown in Table 7.6 and Table 7.7, any of the candidates perform well; the highest performance in terms of F1-score is achieved by OCSVM-SGD. LOF using the Bray-Curtis dissimilarity as a distance metric achieves a lower F1-score and precision but a higher recall. As we value recall over precision, we conclude that LOF with Bray-Curtis dissimilarity is suited best for the task of detecting anomalous jobs.

We observe that every algorithm under test achieves a much higher recall than precision, i.e. significantly more jobs are erroneously flagged as anomalous than normal. This is probably due to our dataset: for each research group, we define this group's jobs as normal and every other group's jobs as anomalous. However, in a real-life setting, groups occasionally show similar computing behaviour. Even though we count jobs of such groups as anomalous, novelty detection models (correctly) classify them as normal. We therefore assume that once deployed, detectors for anomalous jobs achieve a much higher precision than reflected by our experiments.

OCSVM-SGD achieves the highest F1-score. The highest recall is achieved by LOF using either correlation or cosine distance, however, this is at the cost of significantly lower precision.

Figure 7.3 shows confusion matrices for Isolation Forest, OCSVM, OCSVM-SGD and LOF using the Bray-Curtis dissimilarity measure as a distance function. For each of them we observe, that a sample predicted to be normal is indeed normal with a very high probability. However, samples classified as anomalous are often normal as well.

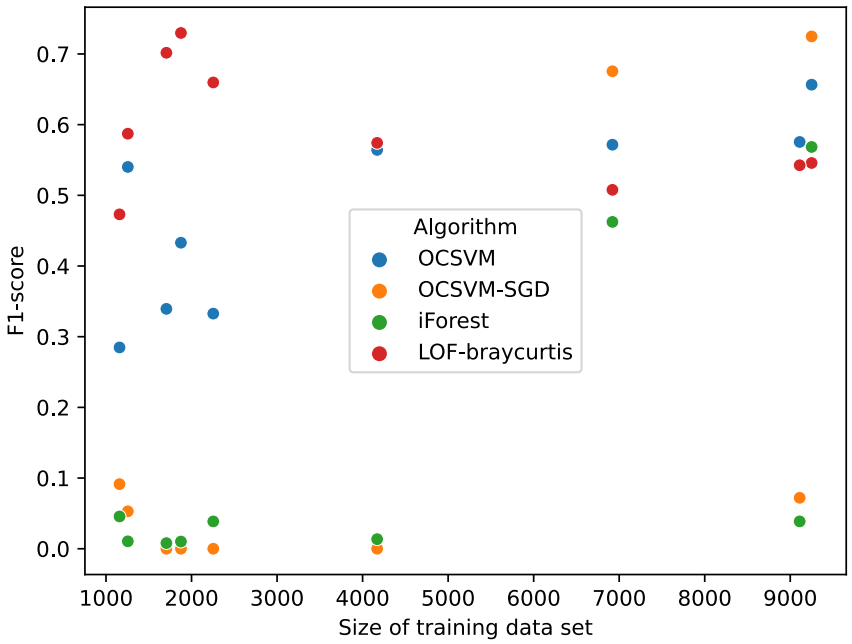


Figure 8.1: Various novelty detection models that have been trained on jobs submitted by different research groups plotted in terms of training set size and achieved F1-score.

All models are reasonably efficient, we observe that LOF models (except the one using Minkowski distance) take the largest amount of time to train and predict. The time to train models with $|X_{training}| < 5,000$ is acceptably low, usually not taking longer than about one minute. However, some research groups submit a large amount of jobs, sometimes up to 25,000 jobs per day. After a few days of collecting data, training a novelty detection model on this amount of jobs will no longer be feasible.

LIMITING THE TRAINING SET SIZE In order to circumvent this issue, we limit the size of the dataset used to train the models. Over the course of our experiments we discovered that the accuracy of models is not strongly correlated with the size of the training set. In Figure 8.1 we plot the achieved F1-score over the number of training samples. We immediately see that a larger training set does not imply a higher model performance. Indeed, the correlation between the F1-score achieved and the number of training samples used can be calculated as 0.362, implying a rather weak correlation.

We use this fact to improve the models' runtimes while retaining good performance. For this, we limit the size of the training datasets to κ. If the $X_{training}$ as defined in Section 5.3.3.1 has a size greater than

κ, we perform training with κ randomly drawn elements from $X_{training}$. We use the hyper-parameters found in Section 7.4.1.

We empirically determine that $\kappa = 1,000$ performs well. The performances of the algorithms under these conditions are shown in Table 7.8 and Table 7.9. All models (except OCSVM-SGD) now require significantly less time to train. Prediction on the LOF models experiences a significant speed-up.

LOF using cosine distance or correlation as its distance metric achieves the highest recall, but very poor precision. If the Bray-Curtis dissimilarity is used as a distance metric, the recall is negligibly lower, but precision is improved considerably. OCSVM-SGD achieves the highest F1-score, but lower recall than LOF. As we value recall over precision and F1-score, we consider LOF using the Bray-Curtis dissimilarity the best-suited novelty detection model for detecting anomalous jobs.

COMPARISON The performance of most algorithms does not suffer from limiting the size of the training dataset. LOF generally achieves a higher recall when $\kappa = 1,000$ but a lower precision, leading to an overall lower F1-score. However, we value recall over precision and regard this as a performance improvement. We thus propose to use LOF using the Bray-Curtis dissimilarity as a distance metric and to cap the size of training sets at $\kappa = 1,000$. This setup achieves excellent time-efficiency and good performance.

8.1.3 Answers to Research Questions

In this Section, we strive to answer the research questions posed in Section 1.2.

CAN WE RELIABLY CLASSIFY USER BEHAVIOUR AS (NON-)MALICIOUS? This research question must be addressed on two levels. On the one hand, we are concerned with the performance the SIEM we propose. On the other hand, we are interested in the question whether or not classification of user behaviour into the categories of malicious and benign is even possible based on the log data we have at hand. If the SIEM we propose performed well, this would obviously be the case.

We first discuss the performance of our proposed SIEM. We propose four anomaly detectors (see Section 4.4) that aim to detect four different types of anomaly found in HPC logs: anomalous logins, anomalous movement, anomalous log messages and anomalous job submissions. We implement three of these detectors; because of time constraints we omit implementation of the anomalous movement detector, which we thus can not evaluate.

In absence of real-world anomalous system logs, we evaluate the detectors for anomalous logins and log messages with randomly

generated anomalously-looking data. OCSVM-SGD and LogCluster perform exceptionally good on this data, but we are aware that this does not necessarily imply good performance detecting real-world anomalies.

The detector for anomalous job submissions uses LOF, an algorithm for novelty detection. Using this algorithm, it is able to learn job submission behaviour for individual research groups. Afterwards, it can tell apart jobs submitted by this group from jobs submitted by other groups, achieving high precision and recall. Because this evaluation was conducted solely on real-world data, we are confident that the detector's performance in a live system would yield satisfying results.

Based on the promising results of the anomaly detectors we believe that detecting malicious behaviour using log files is possible to a satisfying extend. Because of time constraints, we could not conduct experiments evaluating the *end-to-end* performance of the proposed SIEM. Further research attempting such experiments must focus on a) collection or synthesis of realistic malicious sessions b) the impact of the thresholding scheme's parameters $\psi_i, i \in 1 \ldots 4$ and ω on the overall performance of the system.

WHAT ALGORITHMS ARE SUITABLE FOR THIS CLASSIFICATION? The algorithm best employed in each individual anomaly detector depends on the nature of anomalies to be detected. We determined experimentally, that:

- OCSVM-SGD is suited best for detecting anomalous logins.
- LogCluster is suited best for detecting anomalous system log messages.
- LOF using the Bray-Curtis dissimilarity as a distance metric is suited best for detecting anomalous jobs.

We did not determine the best-suited algorithm for detecting anomalous movement experimentally, but suspect that DeepLog could fill this gap with satisfying performance.

We could not evaluate the suitability of the proposed algorithm for use in the thresholding scheme. Further work must focus on this issue and, if the proposed algorithm is found to be a bad fit, explore alternatives.

The algorithms for preprocessing log data are, with the exception of log parsing, straight-forward to formulate and implement. Further, they are sufficiently efficient to cope with the volumes of data to be processed. The issue of log parsing is not yet fully addressed. Drain seems to be the best algorithm currently available for automatic parsing of Linux logs, but still only achieves an accuracy of 0.690 [76]. As the field or automatic log parsing evolves, better algorithms for this task may be found.

WHAT ARE THE COMPUTATIONAL REQUIREMENTS FOR CLASSIFICATION? The computational requirements of the selected algorithms are not particularly high. The steps for preprocessing (extracting logins, extracting movement and parsing system logs) can operate in a streaming manner and run in parallel. Preprocessing and grouping the test dataset (which was collected over a time-span of 81 days) can be performed in under one hour.

The algorithms for anomaly detection we deem well-suited for usage in the proposed SIEM have very moderate resource requirements, which a HPC cluster can easily meet. DeepLog is an exception to this: we were not able to train models with certain hyper-parameter combinations because they required too much memory. However, these experiments investigated the suitability of DeepLog for detecting anomalous Linux system logs. Further experimentation must determine whether or not DeepLog's resource requirements are acceptably low when detecting anomalous movement.

WHAT INFORMATION IS REQUIRED TO IMPROVE CLASSIFICATION RELIABILITY? As noted in Section 5.3.3, only 15.206% of log messages found in `/var/log/messages` can be assigned to user sessions. We assign messages to user sessions by correlating the messages' timestamps with entries in the WM's logs. This approach has the limitation that log messages produced on non-compute nodes or on compute nodes running no job cannot be assigned to sessions.

In order to detect anomalous log messages more reliably, a larger share of `/var/log/messages` must be assigned to user sessions. It is currently not clear how this can be accomplished.

8.2 LIMITATIONS

We are aware of several limitations of both the SIEM we propose and the methodology we use to evaluate it. In this Section, we discuss these limitations.

In Section 4.1, we propose an architecture for a SIEM that detects anomalies in log files indicating malicious behaviour. This SIEM can only function reliably if log collection, which we do not cover in this work, functions reliably. If an attacker can manipulate log files as to erase evidence for malicious behaviour, the SIEM becomes ineffective. Therefore, it is imperative to create a computing environment capable of reliable logging (e.g. by having all nodes in the cluster log to a remote append-only database) before deploying the system we propose.

Another limitation of the proposed SIEM is the lack of explainability. Given a user session s, its output is a binary value (normal/anomalous) and the four anomaly scores $p_1(s)$ - $p_4(s)$. System operators investi-

gating a session flagged as anomalous receive no information what exactly lead to this classification, which forces them to perform the action we aimed to avoid with this work: manual search for anomalies in logs. Utilising explainable machine-learning models or making the models we employ explainable [38] might reduce this manual effort further, which in turn might increase acceptance of the SIEM among system operators.

The experiments we conducted show that all anomaly detectors under test achieve high F1-scores. However, in the case of the anomalous login detector and the anomalous log messages detector it is not clear whether this is due to the fitness of the employed classifier or limitations of our methodology. We evaluated these two detectors by having them distinguish normal log data collected on our test system from randomly generated log data. However, it is not clear that randomly generated log data exhibits the same characteristics as anomalous log data occurring in a real-world setting.

For anomalous logins, this problem is not as severe as for randomly generated system log sequences. All logins in the space of possible logins $A \times T \times L$ (A denoting the set of possible authentication methods, T being the set of timestamps of the form $(\textit{day-of-week}, \textit{time-of-day})$ and L being the set of possible login locations) are plausible and can, in principle, appear during operation of the system. Therefore, we think that our results regarding the detection of anomalous logins indeed reflect real-world performance.

Randomly generated system log sequences however have the potential to be completely absurd. Recall that a sequence of log system log messages reflects the execution path of one or more concurrently running processes. Randomly generated sequences tend to describe execution paths that are logically impossible and thus obviously anomalous. Consider the programme shown in Figure 8.2, which can produce four different log keys: `Start`, `Processing`, `Ok` and `Error`. However, only two distinct log sequences can arise from executing the programme. These log sequences are shown in Figure 8.3. However, by generating log sequences randomly using the log keys of the programme is is possible to arrive at sequences that cannot emerge from its execution (see Figure 8.4 for examples). We call such sequences *impossible sequences*.

We suspect that a large part of randomly generated log sequences are impossible. Let $K = \{k_1 \dots k_n\}$ be the set of log keys that can arise during execution of a given programme. Let $K_i = \{x \in K : x \text{ can appear after } k_i\}$ be the set of log keys that can appear after key k_i during execution of the programme. It is easy to understand that typically, $|K_i| < n$. The mean of $|K_i|, i \in \{1 \dots n\}$, which we denote with ξ, is thus also typically less than n. Generating a random log sequence of length q means choosing q log keys at random. Each of these keys (except the first one) has a chance of (on average) $\frac{\xi}{n}$ to be compatible with the key before it. In total, each generated log sequence

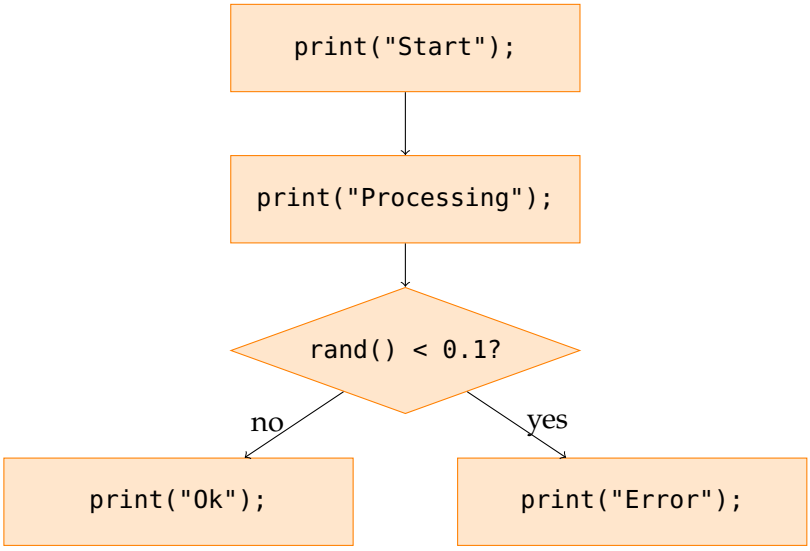


Figure 8.2: An example programme emitting log messages.

`Start`	`Start`
`Processing`	`Processing`
`Ok`	`Error`

Figure 8.3: All log sequences that can possibly be emitted by the programme in Figure 8.2.

of length q thus has a chance of $\left(\frac{\xi}{n}\right)^{q-1}$ not to be an impossible sequence.

In the case of our test system, $n = 415$ and the average log sequence length is ≈ 102. Suppose that $\xi = 400$ (in reality, this value is likely much smaller). Under these conditions, a randomly generated sequence of $q = 102$ log keys has a chance of 0.023 to be not impossible.

This leads us to believe that the vast majority of generates sequence are impossible sequences. A consequence of this is that our experiments do not measure the detectors' performances at the task of telling apart normal from anomalous log sequences, but at the task of distinguishing impossible from not impossible log messages. Hence, we cannot make claims about the performance of the anomalous log messages detector on the task it was designed for.

In order to enable an evaluation yielding valid results, it is necessary to either a) acquire a large corpus of real-world anomalous system log data, or b) develop a method for synthesis of anomalously-looking system log sequences. For this work, Option a is impossible as no such data of our test system exists and Option b is out of scope as developing such a method would likely take months or years of dedicated research.

`Start`	`Error`
`Error`	`Processing`
`Ok`	

Figure 8.4: Log sequences which use the log keys emitted by the programme in Figure 8.2 but cannot arise from executing it.

8.3 FUTURE WORK

In this Section, we discuss potential directions for future work. Section 8.3.1 outlines the need for realistic anomalous log data, which can be used for evaluating the anomalous log messages detector's performance in a meaningful way. Section 8.3.2 discusses the need for explainable machine learning models in the SIEM, which could improve the usability and acceptance of the system. In Section 8.3.3, we argue that research improving the state-of-the-art in automatic log parsing could improve the performance of the SIEM we propose. Section 8.3.4 discusses the potential performance impact of different groupings of job data, which future research must investigate experimentally. Finally, we discuss the importance of implementing the remaining elements of the proposed architecture and an evaluation of the *end-to-end* performance of the SIEM in Section 8.3.5.

8.3.1 Acquisition of Anomalous System Logs

Future work must focus on overcoming the limitations of the methodology of this work. Currently, the most pressing issue is the need for realistic anomalous log sequences. In this work, we use randomly generated log sequences as positive examples for evaluating the performance of the anomalous log messages detector. However, as we discuss in Section 8.2, randomly generated log sequences are not well-suited for this task. The results of our experiments would be more meaningful if realistic anomalous log sequences, i.e. log sequences that are not impossible, were available. To solve this issue, we currently see two possibilities. On the one hand, the log messages produced during real-world attacks on HPC infrastructure could be collected and made available for research. However, it is unclear whether this approach could deliver sufficient amounts of anomalous log data, because unsuccessful attacks are often not recognised as such and successful attack often have the effect that relevant log data are destroyed by the attacker. On the other hand it might be possible to synthesise realistic anomalous system log sequences. We are not aware of any research in this direction.

If realistic anomalous system log sequences were available, an evaluation of the anomalous log sequences detector would produce much more meaningful results. Additionally, the availability of realistic ex-

amples of anomalies would enable us to use supervised machine learning algorithms for anomaly detection. Supervised methods generally achieve better performance in the task of anomalous log sequence detection than unsupervised methods He et al. [26]. The SIEM we propose could thus profit from such methods.

8.3.2 Explainable Models

If a user session s get classified as anomalous, our SIEM presents the system operators with the following information: a note that a session has been classified as anomalous, the values $p_1(s) - p_4(s)$ and the session s itself (including logins, movement, system log messages and submitted jobs). This information can be used by operators to confirm the abnormality of the session in question. If it is a false positive, operators can provide feedback to the SIEM, improving its accuracy. If it is indeed anomalous, the operators must investigate s to determine the cause and the severity of the anomaly. The values $p_1(s) - p_4(s)$ may point investigators into the right direction, but apart from that, this process is completely manual. Manual investigation of logs must be kept to a minimum, because it is a repetitive and time-consuming task. Frequent false positives and a time-consuming investigation process may lead to alert fatigue [23].

In order to aid operators investigating alerts produced by the SIEM we propose, future work may investigate the applicability of explainable machine-learning algorithms for usage in the anomaly detectors.

In our detectors, we deploy one neural network-based and three classical machine learning models. For the classical machine learning models, one could investigate the suitability of approaches like Shap (Lundberg and Lee [38]) or LIME (Ribeiro, Singh, and Guestrin [54]). These methods can be utilised on any classifier and calculate the importances of the different features of the input data.

DeepLog is a neural network architecture based on LSTMs, for which specialised approaches for explainable models exist [50, 61, 70]. Future work is needed for determining whether these methods are fit for making DeepLog explainable.

8.3.3 Automatic Log Parsing

Log parsing is a prerequisite for detecting anomalous system log sequences. To our knowledge, Drain [25] is the log parser achieving state-of-the-art performance in parsing Linux system logs. It provides very good time-efficiency and an accuracy of 0.690 [76]. We believe that a log parser able to parse Linux logs more accurately could improve downstream performance of the anomalous log messages detector.

Recent research has delivered novel algorithms for automated log parsing. *Logram* (Dai et al. [10]) and *Unified Log Parsing tool* (ULP, Sedki

et al. [57]) have been published during the writing of this work and show promising results. Logram is a very time-efficient algorithm capable of working in an online manner. Its authors report that it can parse Linux logs with a higher accuracy than Drain, Spell and LenMa, however, the authors' use a different definition of accuracy than we and Zhu et al. [76]. ULP's authors claim high accuracy and time-efficiency. However, the an evaluation of the algorithm on Linux logs as well as an online parsing mode are (currently) lacking. A further investigation of the suitability of these novel algorithms for online parsing of Linux logs is needed, including a comparison of performance to established log parsers using a standardised accuracy measure.

8.3.4 Impact of Job Grouping

We currently aim to detect anomalous jobs on a *per-research-group* basis (see Section 4.4.4). This relies on the assumption that each research group works with a distinct set of tools appropriate for the group's research focus. We have shown that good anomaly detection performance can be achieved by defining jobs deviating from the groups' typical computing behaviour as anomalous. However, it might be possible that even better performance can be achieved by doing *per-user* anomaly detection. Assuming that each user (as opposed to each research group) has distinct computing behaviour could provide us with a more fine-grained view on the system, which could result in higher recall values (but, as we suspect, also in lower precision values). Future work must determine which grouping (or which combination thereof) is suited best for anomaly detection and yields the highest performance.

8.3.5 Experimental Evaluation of the Remaining Architecture Elements

Because of time constraints, we were not able to implement and evaluate the performance of all elements of the architecture proposed in Section 4.1. In Figure 4.1 shows the architectural elements that have not yet been implemented filled in with white. We leave the experimental evaluation of these elements for future work.

In this Section, we describe how future work could implement or improve element of the proposed SIEM. Section 8.3.5.1 concerns the anomalous movement detector and the necessary preprocessing step. In Section 8.3.5.2, we describe an algorithm to find a suitable parameter ω for the thresholding scheme and to evaluate the overall performance of the SIEM. Section 8.3.5.3 discusses the possibility of improving the anomalous log sequences detector by exploiting a feature of LogCluster.

8.3.5.1 Extracting Movement & Anomalous Movement Detector

In order to evaluate the viability of detecting anomalous user behaviour through movement in the cluster, future work must implement two elements of our architecture: the preprocessing step of extracting movement from `/var/log/secure` and the anomaly detector for anomalous movement, for which we suggest DeepLog [14]. Evaluating the performance of the detector will likely face the problem that none or very few examples of anomalous behaviour are available. In order to obtain meaningful evaluation results, we suggest tackling the issue of acquiring anomalous system logs (see Section 8.3.1) first.

8.3.5.2 Thresholding Scheme

Evaluating the *end-to-end* performance of the SIEM, i.e. performance the SIEM achieves when classifying user sessions, requires an implementation of the thresholding scheme we propose. This thresholding scheme has a parameter ω. If the weighted sum of the anomaly scores exceeds ω for a given user session, this session is considered anomalous. In Section 4.5, we propose to use $\omega = 4$, because this ensures that no single anomaly detector can have the effect of marking a session anomalous, which reduces the number of false positives. However, in order to yield the best performance possible, future work must conduct experiments that determine a better value for ω. In order to perform these experiments, the following conditions have to be met:

- Implementations for all elements of the architecture proposed in Section 4.1 must be available.
- A set of labelled session data has to be available. These data can either be generated or collected during HPC operations and should include examples of both normal and anomalous behaviour.

The process of finding a good value for ω and determining the SIEM's performance is described below:

- Split the labelled session data into three sets: $X_{training}$, X_{test} and $X_{validation}$.
- Select a number of candidate values for ω. We refer to the set of these candidate values as Ω.
- Train the anomaly detectors of the SIEM using $X_{training}$.
- Let the SIEM predict the labels of the user sessions in X_{test}, calculating the SIEM's F1-score. Do this using each candidate value in Ω for ω once.

- Select ω_{best} as the candidate value in Ω that achieved the highest F1-score in the previous step.
- Determine the SIEM's performance in terms of precision, recall and F1-score using $\omega := \omega_{best}$ by predicting the labels of the user sessions in $X_{validation}$.

The implementation of this process is left for future work.

8.3.5.3 Exploring Continuous Return Values for LogCluster

As discussed in Section 8.1.2.2, we propose to use LogCluster [35] for detecting anomalous log sequences. For conducting our experiments, we use an implementation of LogCluster provided by He et al. [26]. This implementation classifies log sequences as either anomalous or normal. There is no notion of some log sequences being *more anomalous* than others – the decision is binary. However, having a continuous value as output (like the detectors for anomalous logins and anomalous jobs) might increase the overall performance of the SIEM we propose, as this would allow for more fine-grained decision making. Future work may be performed to investigate this possibility.

Making LogCluster's output continuous is straight-forward: LogCluster classifies a given log sequence by measuring the vectorised sequence's minimum cosine distance to the clusters representing normal log sequences. We can use this minimum distance as an indicator for abnormality: The larger it is, the higher the certainty that the log sequence in question is indeed anomalous. In order to make the minimum distance suitable as an anomaly score (which must lie in the range $[0, 1]$), we can apply it to a sigmoid function.

The anomaly score $p_3(s)$ of a user session s can then be calculated as follows: Let $b_1 \dots b_n$ denote the log sequences contained in s. Let $\text{dist}(b_k)$ denote the cosine distance of log sequence b_k to the nearest cluster of normal behaviour as calculated by LogCluster. We define $p_3(s) = \text{sig}(\max\{\text{dist}(b_k) | k \in \{1 \dots n\}\})$, where $\text{sig}(x) = \frac{1}{1+e^{-x}}$ [58, p. 148].

The parameter ψ_3 in the thresholding scheme combining the anomaly scores must be selected as $\psi_3 = \text{sig}(\delta)$ with δ being the hyper-parameter LogCluster uses as a threshold for distinguishing normal and anomalous log sequences.

Using this definition for the anomaly score, $p_3(s)$ becomes a continuous value. The question whether this definition improves the performance of the SIEM we propose must be answered by future work.

9 SUMMARY & CONCLUSION

9.1 SUMMARY

In this Section, we summarise the contents of this work. Chapter 1 motivates our research and explains its goals. Further, it poses research questions which are investigated over the course of this thesis. Chapter 2 provides background knowledge regarding attacks on HPC systems and common log files in HPC environments. In Chapter 3, we introduce related work. Section 3.1 deals with existing methods for anomaly detection using log data and Section 3.2 introduces methods for novelty detection. Chapter 4 contains the main contribution of this thesis, which is an architecture for a SIEM suited for anomaly detection in HPC systems. A broad overview of this architecture is provided in Section 4.1. Section 4.2 and Section 4.3 consider the issues of preprocessing and grouping log data. Four different anomaly detectors are proposed in Section 4.4. Each of these detectors is aimed at discovering one distinct type of anomaly in log data: anomalous logins, anomalous movement, anomalous system log messages and anomalous jobs.

We perform experiments to determine the most suitable algorithm for each of the proposed anomaly detectors (except the anomalous movement detector, for which we limit ourselves to a conceptual analysis). In Chapter 5, we explain the methodology for these experiments, Chapter 6 contains implementation details and Chapter 7 contains the results. In Chapter 8, we discuss the results and the limitations of our approach. We find that the algorithms OCSVM-SGD, LogCluster and LOF with distance metric Bray-Curtis dissimilarity are suited best for detecting anomalous logins, system log messages and jobs respectively and perform very well in terms of the selected performance metrics. We assume that the detectors for anomalous logins and anomalous jobs perform well in a real-life setting. However, because of a lack of real-world examples of anomalous log data, we cannot make claims about the anomalous log messages detector's performance in the field. Additionally, we make suggestions for future work.

9.2 CONCLUSION

In this thesis, we propose an architecture for a SIEM to be deployed in HPC environments. This SIEM is able to detect actions with malicious intent though the analysis of log files that are typically available in

HPC systems. Our experimental results suggest that the log files in question provide sufficient information to perform such a detection, however, the reliability of detection in a real-world setting is subject to future work.

Part I

APPENDIX

A HYPER-PARAMETER SEARCH

This Section includes the results of every tested combination of hyper-parameters during hyper-parameter optimisation.

A.1 ANOMALOUS LOGIN DETECTOR

In this Section, we present the results we obtained during hyper-parameter optimisation of the candidate models for the anomalous login detector. The naming of the methods reflects the hyper-parameter combination tested:

- IsolationForest-{t}
- LOF-{*distance metric*}
- OCSVM-{*kernel function*}-{γ}-{ν}
- OCSVM-SGD-{*kernel function*}-{γ}-{ν}
- OCSVM-SGD-ad.{*kernel function*}-{γ}-{ν}

Table A.1 contains the number of true/false positives/negatives; Table A.2 contains the derived metrics and efficiency.

METHOD	TP	FP	FP	TN
OCSVM-SGD-ad.-sigm.-0.8-0.1	19	1	2498	237
OCSVM-SGD-ad.-sigm.-0.6-0.1	20	0	2499	236
OCSVM-SGD-sigm.-0.8-0.9	235	2429	70	21
OCSVM-SGD-ad.-sigm.-0.8-0.8	245	2490	9	11
OCSVM-SGD-ad.-sigm.-0.8-0.9	246	2496	3	10
OCSVM-SGD-sigm.-0.8-0.8	234	2314	185	22
OCSVM-SGD-ad.-sigm.-0.6-0.8	252	2479	20	4
OCSVM-SGD-ad.-sigm.-0.6-0.9	255	2499	0	1
OCSVM-SGD-sigm.-0.6-0.9	246	2394	105	10
OCSVM-SGD-ad.-sigm.-0.4-0.9	256	2491	8	0
OCSVM-SGD-ad.-sigm.-0.3-0.9	256	2470	29	0
OCSVM-SGD-ad.-rbf-0.8-0.9	256	2458	41	0
OCSVM-SGD-ad.-rbf-0.6-0.9	256	2450	49	0
OCSVM-SGD-rbf-0.8-0.9	256	2443	56	0

METHOD	TP	FP	TN	FN
OCSVM-SGD-ad.-rbf-0.4-0.9	256	2424	75	0
OCSVM-SGD-ad.-sigm.-0.2-0.9	256	2424	75	0
OCSVM-SGD-rbf-0.6-0.9	256	2422	77	0
OCSVM-SGD-ad.-rbf-0.3-0.9	256	2416	83	0
OCSVM-SGD-ad.-sigm.-0.1-0.9	256	2403	96	0
OCSVM-SGD-rbf-0.4-0.9	256	2386	113	0
OCSVM-SGD-ad.-rbf-0.2-0.9	256	2374	125	0
OCSVM-SGD-rbf-0.3-0.9	256	2371	128	0
OCSVM-SGD-sigm.-0.4-0.9	255	2343	156	1
OCSVM-SGD-rbf-0.2-0.9	256	2352	147	0
OCSVM-SGD-ad.-rbf-0.1-0.9	256	2352	147	0
OCSVM-SGD-rbf-0.1-0.9	256	2338	161	0
OCSVM-SGD-ad.-sigm.-0.4-0.8	253	2297	202	3
OCSVM-SGD-ad.-rbf-0.8-0.8	256	2327	172	0
OCSVM-SGD-sigm.-0.3-0.9	256	2310	189	0
OCSVM-SGD-sigm.-0.1-0.9	256	2294	205	0
OCSVM-SGD-sigm.-0.2-0.9	256	2293	206	0
OCSVM-SGD-ad.-rbf-0.6-0.8	256	2287	212	0
OCSVM-poly-0.8-0.9	256	2275	224	0
OCSVM-sigm.-0.6-0.9	256	2274	225	0
OCSVM-sigm.-0.8-0.9	256	2274	225	0
OCSVM-SGD-ad.-rbf-0.4-0.8	256	2250	249	0
OCSVM-sigm.-0.3-0.9	256	2246	253	0
OCSVM-poly-0.4-0.9	256	2245	254	0
OCSVM-sigm.-0.4-0.9	256	2243	256	0
OCSVM-SGD-sigm.-0.6-0.8	245	2133	366	11
OCSVM-sigm.-0.2-0.9	256	2240	259	0
OCSVM-rbf-scale-0.9	256	2235	264	0
OCSVM-rbf-0.4-0.9	256	2234	265	0
OCSVM-sigm.-0.1-0.9	256	2218	281	0
OCSVM-SGD-ad.-rbf-0.3-0.8	256	2214	285	0
OCSVM-poly-0.2-0.9	256	2207	292	0
OCSVM-rbf-0.3-0.9	256	2206	293	0
OCSVM-sigm.-scale-0.9	256	2203	296	0
OCSVM-poly-auto-0.9	256	2196	303	0
OCSVM-SGD-rbf-0.8-0.8	256	2190	309	0
OCSVM-SGD-rbf-0.4-0.8	256	2187	312	0

METHOD	TP	FP	TN	FN
OCSVM-SGD-rbf-0.6-0.8	256	2186	313	0
OCSVM-sigm.-auto-0.9	256	2176	323	0
OCSVM-poly-scale-0.9	256	2171	328	0
OCSVM-SGD-rbf-0.3-0.8	256	2164	335	0
OCSVM-SGD-rbf-0.2-0.8	256	2159	340	0
OCSVM-SGD-ad.-sigm.-0.3-0.8	254	2137	362	2
OCSVM-rbf-0.2-0.9	256	2154	345	0
OCSVM-SGD-ad.-rbf-0.2-0.8	256	2154	345	0
OCSVM-SGD-ad.-sigm.-0.1-0.8	256	2149	350	0
OCSVM-lin.-scale-0.9	256	2143	356	0
OCSVM-lin.-auto-0.9	256	2143	356	0
OCSVM-lin.-0.1-0.9	256	2143	356	0
OCSVM-lin.-0.2-0.9	256	2143	356	0
OCSVM-lin.-0.3-0.9	256	2143	356	0
OCSVM-lin.-0.4-0.9	256	2143	356	0
OCSVM-lin.-0.6-0.9	256	2143	356	0
OCSVM-lin.-0.8-0.9	256	2143	356	0
OCSVM-rbf-auto-0.9	256	2139	360	0
OCSVM-poly-0.3-0.9	256	2138	361	0
OCSVM-rbf-0.8-0.9	256	2132	367	0
OCSVM-poly-0.1-0.9	256	2131	368	0
OCSVM-rbf-0.6-0.9	256	2128	371	0
OCSVM-rbf-0.1-0.9	256	2121	378	0
OCSVM-SGD-rbf-0.1-0.8	256	2101	398	0
OCSVM-SGD-ad.-rbf-0.8-0.7	256	2097	402	0
OCSVM-poly-0.8-0.8	256	2087	412	0
OCSVM-SGD-ad.-rbf-0.1-0.8	256	2081	418	0
OCSVM-poly-0.2-0.8	256	2072	427	0
OCSVM-poly-0.4-0.8	256	2072	427	0
OCSVM-poly-0.6-0.9	256	2068	431	0
OCSVM-poly-0.1-0.8	256	2066	433	0
OCSVM-SGD-sigm.-0.8-0.7	228	1798	701	28
OCSVM-rbf-0.4-0.8	256	2049	450	0
OCSVM-SGD-ad.-rbf-0.6-0.7	256	2046	453	0
OCSVM-rbf-0.3-0.8	256	2044	455	0
OCSVM-sigm.-0.8-0.8	255	2033	466	1
OCSVM-SGD-sigm.-0.2-0.8	255	2025	474	1

METHOD	TP	FP	TN	FN
OCSVM-SGD-sigm.-0.1-0.8	256	2030	469	0
OCSVM-SGD-ad.-sigm.-0.2-0.8	255	2018	481	1
OCSVM-SGD-sigm.-0.4-0.8	255	2009	490	1
OCSVM-rbf-0.6-0.8	256	2014	485	0
OCSVM-sigm.-0.6-0.8	255	1999	500	1
OCSVM-SGD-ad.-sigm.-0.4-0.1	29	0	2499	227
OCSVM-rbf-0.1-0.8	256	2003	496	0
OCSVM-SGD-sigm.-0.3-0.8	255	1992	507	1
OCSVM-sigm.-0.4-0.8	256	1995	504	0
OCSVM-SGD-ad.-rbf-0.4-0.7	256	1990	509	0
OCSVM-rbf-auto-0.8	256	1986	513	0
OCSVM-rbf-0.8-0.8	256	1983	516	0
OCSVM-sigm.-0.1-0.8	256	1981	518	0
OCSVM-poly-auto-0.8	256	1969	530	0
OCSVM-sigm.-0.2-0.8	256	1966	533	0
OCSVM-SGD-rbf-0.8-0.7	256	1959	540	0
OCSVM-rbf-0.2-0.8	256	1952	547	0
OCSVM-sigm.-0.3-0.8	256	1951	548	0
OCSVM-lin.-scale-0.8	256	1949	550	0
OCSVM-lin.-auto-0.8	256	1949	550	0
OCSVM-lin.-0.1-0.8	256	1949	550	0
OCSVM-lin.-0.2-0.8	256	1949	550	0
OCSVM-lin.-0.3-0.8	256	1949	550	0
OCSVM-lin.-0.4-0.8	256	1949	550	0
OCSVM-lin.-0.6-0.8	256	1949	550	0
OCSVM-lin.-0.8-0.8	256	1949	550	0
OCSVM-SGD-rbf-0.6-0.7	256	1946	553	0
OCSVM-SGD-ad.-rbf-0.3-0.7	256	1936	563	0
OCSVM-sigm.-scale-0.8	256	1927	572	0
OCSVM-poly-scale-0.8	256	1924	575	0
OCSVM-sigm.-auto-0.8	256	1924	575	0
OCSVM-rbf-scale-0.8	256	1923	576	0
OCSVM-rbf-0.8-0.7	256	1907	592	0
OCSVM-poly-0.8-0.7	256	1905	594	0
OCSVM-rbf-0.4-0.7	256	1896	603	0
OCSVM-rbf-0.6-0.7	256	1856	643	0
OCSVM-rbf-scale-0.7	256	1846	653	0

METHOD	TP	FP	TN	FN
OCSVM-poly-0.3-0.8	256	1845	654	0
OCSVM-poly-0.6-0.8	256	1836	663	0
OCSVM-SGD-ad.-rbf-0.2-0.7	256	1831	668	0
OCSVM-SGD-rbf-0.3-0.7	256	1816	683	0
OCSVM-SGD-ad.-rbf-0.8-0.6	256	1809	690	0
OCSVM-sigm.-0.1-0.7	256	1808	691	0
OCSVM-poly-0.2-0.7	256	1806	693	0
OCSVM-rbf-0.2-0.7	256	1800	699	0
OCSVM-SGD-rbf-0.4-0.7	256	1797	702	0
OCSVM-SGD-rbf-0.2-0.7	256	1794	705	0
OCSVM-poly-0.4-0.7	256	1793	706	0
OCSVM-rbf-0.6-0.6	256	1789	710	0
OCSVM-sigm.-0.8-0.7	254	1767	732	2
OCSVM-sigm.-0.3-0.7	255	1770	729	1
OCSVM-SGD-sigm.-0.6-0.7	239	1640	859	17
OCSVM-SGD-poly-0.1-0.9	256	1771	728	0
OCSVM-rbf-0.8-0.6	256	1762	737	0
OCSVM-sigm.-auto-0.7	256	1759	740	0
OCSVM-poly-0.1-0.7	256	1752	747	0
OCSVM-sigm.-0.6-0.7	254	1722	777	2
OCSVM-sigm.-0.4-0.7	256	1734	765	0
OCSVM-SGD-ad.-rbf-0.6-0.6	256	1733	766	0
OCSVM-rbf-0.4-0.6	256	1728	771	0
OCSVM-SGD-ad.-poly-0.1-0.9	255	1716	783	1
OCSVM-rbf-auto-0.7	256	1718	781	0
OCSVM-SGD-rbf-0.1-0.7	256	1715	784	0
OCSVM-rbf-0.3-0.7	256	1711	788	0
OCSVM-lin.-scale-0.7	256	1709	790	0
OCSVM-lin.-auto-0.7	256	1709	790	0
OCSVM-lin.-0.1-0.7	256	1709	790	0
OCSVM-lin.-0.2-0.7	256	1709	790	0
OCSVM-lin.-0.3-0.7	256	1709	790	0
OCSVM-lin.-0.4-0.7	256	1709	790	0
OCSVM-lin.-0.6-0.7	256	1709	790	0
OCSVM-lin.-0.8-0.7	256	1709	790	0
OCSVM-poly-0.8-0.6	256	1703	796	0
OCSVM-sigm.-0.2-0.7	255	1695	804	1

METHOD	TP	FP	TN	FN
OCSVM-SGD-sigm.-0.2-0.7	255	1684	815	1
OCSVM-SGD-sigm.-0.3-0.7	254	1676	823	2
OCSVM-sigm.-scale-0.7	255	1678	821	1
OCSVM-poly-scale-0.7	256	1684	815	0
OCSVM-SGD-sigm.-0.4-0.7	253	1650	849	3
OCSVM-rbf-0.1-0.7	256	1664	835	0
OCSVM-SGD-sigm.-0.1-0.7	256	1662	837	0
OCSVM-poly-0.4-0.6	256	1651	848	0
OCSVM-SGD-ad.-rbf-0.1-0.7	256	1649	850	0
OCSVM-poly-0.3-0.7	256	1644	855	0
OCSVM-rbf-0.8-0.5	256	1641	858	0
OCSVM-SGD-ad.-rbf-0.4-0.6	256	1641	858	0
OCSVM-rbf-scale-0.6	256	1629	870	0
OCSVM-poly-auto-0.7	256	1626	873	0
OCSVM-poly-0.6-0.7	256	1617	882	0
OCSVM-rbf-0.3-0.6	256	1606	893	0
OCSVM-rbf-0.6-0.4	256	1599	900	0
OCSVM-rbf-0.2-0.6	256	1592	907	0
OCSVM-poly-scale-0.6	256	1580	919	0
OCSVM-SGD-lin.-0.1-0.9	256	1571	928	0
OCSVM-SGD-lin.-0.2-0.9	256	1571	928	0
OCSVM-SGD-lin.-0.3-0.9	256	1571	928	0
OCSVM-SGD-lin.-0.4-0.9	256	1571	928	0
OCSVM-SGD-lin.-0.6-0.9	256	1571	928	0
OCSVM-SGD-lin.-0.8-0.9	256	1571	928	0
OCSVM-SGD-poly-0.1-0.8	256	1563	936	0
OCSVM-sigm.-0.8-0.6	253	1537	962	3
OCSVM-SGD-rbf-0.6-0.6	256	1548	951	0
OCSVM-rbf-0.1-0.6	256	1546	953	0
OCSVM-lin.-scale-0.6	255	1537	962	1
OCSVM-lin.-auto-0.6	255	1537	962	1
OCSVM-lin.-0.1-0.6	255	1537	962	1
OCSVM-lin.-0.2-0.6	255	1537	962	1
OCSVM-lin.-0.3-0.6	255	1537	962	1
OCSVM-lin.-0.4-0.6	255	1537	962	1
OCSVM-lin.-0.6-0.6	255	1537	962	1
OCSVM-lin.-0.8-0.6	255	1537	962	1

METHOD	TP	FP	TN	FN
OCSVM-SGD-rbf-0.8-0.6	256	1544	955	0
OCSVM-rbf-auto-0.6	256	1543	956	0
OCSVM-rbf-0.6-0.5	256	1530	969	0
OCSVM-rbf-0.4-0.4	256	1529	970	0
OCSVM-poly-0.1-0.6	256	1528	971	0
OCSVM-poly-0.8-0.5	255	1519	980	1
OCSVM-rbf-0.8-0.4	256	1521	978	0
OCSVM-SGD-ad.-sigm.-0.1-0.1	37	0	2499	219
OCSVM-poly-0.3-0.6	256	1515	984	0
OCSVM-SGD-ad.-rbf-0.3-0.6	256	1515	984	0
OCSVM-SGD-ad.-lin.-0.1-0.9	255	1507	992	1
OCSVM-SGD-ad.-lin.-0.2-0.9	255	1507	992	1
OCSVM-SGD-ad.-lin.-0.3-0.9	255	1507	992	1
OCSVM-SGD-ad.-lin.-0.4-0.9	255	1507	992	1
OCSVM-SGD-ad.-lin.-0.6-0.9	255	1507	992	1
OCSVM-SGD-ad.-lin.-0.8-0.9	255	1507	992	1
OCSVM-sigm.-0.1-0.6	255	1504	995	1
OCSVM-rbf-0.4-0.5	256	1505	994	0
iForest-1000-0.5	248	1449	1050	8
OCSVM-sigm.-auto-0.6	255	1497	1002	1
iForest-500-0.5	247	1442	1057	9
OCSVM-sigm.-0.4-0.6	254	1489	1010	2
iForest-200-0.5	247	1439	1060	9
OCSVM-SGD-ad.-poly-0.1-0.8	254	1486	1013	2
iForest-30-0.5	245	1421	1078	11
OCSVM-poly-auto-0.6	254	1481	1018	2
OCSVM-sigm.-0.2-0.6	256	1493	1006	0
OCSVM-rbf-scale-0.5	256	1491	1008	0
iForest-10-0.5	245	1411	1088	11
OCSVM-rbf-0.8-0.3	256	1480	1019	0
OCSVM-SGD-rbf-0.4-0.6	256	1479	1020	0
iForest-50-0.5	248	1421	1078	8
OCSVM-sigm.-0.3-0.6	254	1461	1038	2
iForest-100-0.5	248	1420	1079	8
OCSVM-sigm.-0.6-0.6	254	1460	1039	2
iForest-20-0.5	245	1399	1100	11
OCSVM-poly-0.2-0.6	256	1472	1027	0

METHOD	TP	FP	TN	FN
OCSVM-sigm.-scale-0.6	256	1456	1043	0
OCSVM-poly-0.2-0.5	255	1448	1051	1
OCSVM-rbf-0.1-0.5	256	1452	1047	0
OCSVM-poly-0.4-0.5	255	1443	1056	1
OCSVM-SGD-rbf-0.3-0.6	256	1444	1055	0
OCSVM-SGD-ad.-rbf-0.2-0.6	256	1439	1060	0
OCSVM-poly-0.1-0.5	255	1418	1081	1
OCSVM-poly-0.6-0.6	256	1422	1077	0
OCSVM-rbf-0.6-0.3	256	1422	1077	0
OCSVM-poly-scale-0.5	255	1414	1085	1
OCSVM-SGD-rbf-0.2-0.6	256	1420	1079	0
OCSVM-rbf-0.2-0.5	256	1411	1088	0
OCSVM-rbf-0.8-0.2	256	1406	1093	0
OCSVM-rbf-0.3-0.5	256	1405	1094	0
OCSVM-rbf-0.8-0.1	256	1402	1097	0
OCSVM-poly-auto-0.5	254	1385	1114	2
OCSVM-SGD-sigm.-0.1-0.6	253	1355	1144	3
OCSVM-SGD-lin.-0.1-0.8	255	1364	1135	1
OCSVM-SGD-lin.-0.2-0.8	255	1364	1135	1
OCSVM-SGD-lin.-0.3-0.8	255	1364	1135	1
OCSVM-SGD-lin.-0.4-0.8	255	1364	1135	1
OCSVM-SGD-lin.-0.6-0.8	255	1364	1135	1
OCSVM-SGD-lin.-0.8-0.8	255	1364	1135	1
OCSVM-sigm.-auto-0.5	253	1349	1150	3
OCSVM-SGD-rbf-0.1-0.6	256	1366	1133	0
OCSVM-SGD-ad.-sigm.-0.8-0.2	41	3	2496	215
OCSVM-SGD-sigm.-0.2-0.6	253	1342	1157	3
OCSVM-poly-0.3-0.5	255	1349	1150	1
OCSVM-sigm.-0.1-0.5	253	1333	1166	3
OCSVM-SGD-poly-0.1-0.7	254	1338	1161	2
OCSVM-rbf-auto-0.5	256	1342	1157	0
OCSVM-SGD-ad.-lin.-0.1-0.8	252	1309	1190	4
OCSVM-SGD-ad.-lin.-0.2-0.8	252	1309	1190	4
OCSVM-SGD-ad.-lin.-0.3-0.8	252	1309	1190	4
OCSVM-SGD-ad.-lin.-0.4-0.8	252	1309	1190	4
OCSVM-SGD-ad.-lin.-0.6-0.8	252	1309	1190	4
OCSVM-SGD-ad.-lin.-0.8-0.8	252	1309	1190	4

METHOD	TP	FP	TN	FN
OCSVM-sigm.-0.3-0.5	251	1295	1204	5
OCSVM-SGD-ad.-rbf-0.8-0.5	256	1323	1176	0
OCSVM-SGD-poly-0.2-0.9	256	1306	1193	0
iForest-100-0.4	243	1219	1280	13
OCSVM-sigm.-0.8-0.5	253	1279	1220	3
OCSVM-rbf-0.3-0.4	256	1297	1202	0
iForest-10-0.4	237	1177	1322	19
OCSVM-sigm.-0.2-0.5	252	1265	1234	4
OCSVM-sigm.-0.6-0.5	252	1265	1234	4
OCSVM-SGD-sigm.-0.3-0.6	252	1265	1234	4
OCSVM-SGD-ad.-rbf-0.6-0.5	256	1289	1210	0
OCSVM-sigm.-scale-0.5	253	1270	1229	3
OCSVM-sigm.-0.4-0.5	252	1263	1236	4
iForest-200-0.4	243	1207	1292	13
OCSVM-lin.-scale-0.5	253	1265	1234	3
OCSVM-lin.-auto-0.5	253	1265	1234	3
OCSVM-lin.-0.1-0.5	253	1265	1234	3
OCSVM-lin.-0.2-0.5	253	1265	1234	3
OCSVM-lin.-0.3-0.5	253	1265	1234	3
OCSVM-lin.-0.4-0.5	253	1265	1234	3
OCSVM-lin.-0.6-0.5	253	1265	1234	3
OCSVM-lin.-0.8-0.5	253	1265	1234	3
OCSVM-SGD-sigm.-0.6-0.6	238	1174	1325	18
iForest-50-0.4	240	1186	1313	16
iForest-500-0.4	244	1209	1290	12
OCSVM-rbf-0.4-0.3	256	1280	1219	0
OCSVM-SGD-ad.-sigm.-0.6-0.2	43	2	2497	213
iForest-1000-0.4	244	1205	1294	12
OCSVM-rbf-0.6-0.2	256	1275	1224	0
OCSVM-SGD-ad.-sigm.-0.3-0.1	43	0	2499	213
iForest-30-0.4	240	1172	1327	16
OCSVM-SGD-sigm.-0.4-0.6	252	1243	1256	4
OCSVM-SGD-ad.-sigm.-0.1-0.7	250	1227	1272	6
OCSVM-rbf-0.2-0.4	256	1255	1244	0
iForest-20-0.4	241	1166	1333	15
OCSVM-poly-0.8-0.4	255	1238	1261	1
OCSVM-SGD-ad.-poly-0.1-0.7	252	1218	1281	4

METHOD	TP	FP	TN	FN
OCSVM-poly-auto-0.4	253	1218	1281	3
OCSVM-poly-0.6-0.5	255	1226	1273	1
OCSVM-SGD-ad.-sigm.-0.8-0.7	216	999	1500	40
OCSVM-SGD-ad.-poly-0.2-0.9	252	1202	1297	4
OCSVM-rbf-0.6-0.1	256	1222	1277	0
OCSVM-SGD-poly-0.2-0.8	253	1204	1295	3
OCSVM-rbf-0.3-0.3	256	1221	1278	0
OCSVM-SGD-rbf-0.8-0.5	256	1221	1278	0
OCSVM-poly-0.3-0.4	255	1208	1291	1
OCSVM-SGD-sigm.-0.8-0.6	227	1047	1452	29
OCSVM-SGD-lin.-0.1-0.7	253	1194	1305	3
OCSVM-SGD-lin.-0.2-0.7	253	1194	1305	3
OCSVM-SGD-lin.-0.3-0.7	253	1194	1305	3
OCSVM-SGD-lin.-0.4-0.7	253	1194	1305	3
OCSVM-SGD-lin.-0.6-0.7	253	1194	1305	3
OCSVM-SGD-lin.-0.8-0.7	253	1194	1305	3
OCSVM-rbf-0.1-0.4	256	1208	1291	0
OCSVM-rbf-0.4-0.2	256	1198	1301	0
OCSVM-SGD-rbf-0.6-0.5	256	1194	1305	0
OCSVM-SGD-ad.-sigm.-0.2-0.7	252	1168	1331	4
OCSVM-SGD-rbf-0.4-0.5	256	1178	1321	0
OCSVM-rbf-0.1-0.3	256	1173	1326	0
OCSVM-rbf-auto-0.4	256	1172	1327	0
OCSVM-SGD-poly-0.1-0.6	253	1152	1347	3
OCSVM-poly-0.2-0.4	255	1163	1336	1
OCSVM-SGD-ad.-rbf-0.1-0.6	256	1162	1337	0
OCSVM-SGD-ad.-sigm.-0.3-0.7	250	1125	1374	6
OCSVM-SGD-ad.-rbf-0.4-0.5	256	1158	1341	0
OCSVM-sigm.-scale-0.4	251	1130	1369	5
OCSVM-rbf-scale-0.4	256	1150	1349	0
OCSVM-SGD-rbf-0.3-0.5	256	1150	1349	0
OCSVM-rbf-scale-0.3	256	1133	1366	0
OCSVM-poly-scale-0.4	255	1110	1389	1
OCSVM-poly-0.1-0.4	255	1108	1391	1
OCSVM-rbf-0.2-0.3	256	1112	1387	0
OCSVM-rbf-auto-0.3	256	1105	1394	0
iForest-200-0.3	232	977	1522	24

METHOD	TP	FP	TN	FN
OCSVM-SGD-rbf-0.2-0.5	256	1102	1397	0
OCSVM-lin.-scale-0.4	251	1074	1425	5
OCSVM-lin.-auto-0.4	251	1074	1425	5
OCSVM-lin.-0.1-0.4	251	1074	1425	5
OCSVM-lin.-0.2-0.4	251	1074	1425	5
OCSVM-lin.-0.3-0.4	251	1074	1425	5
OCSVM-lin.-0.4-0.4	251	1074	1425	5
OCSVM-lin.-0.6-0.4	251	1074	1425	5
OCSVM-lin.-0.8-0.4	251	1074	1425	5
OCSVM-poly-0.4-0.4	255	1093	1406	1
OCSVM-SGD-ad.-rbf-0.3-0.5	256	1097	1402	0
iForest-100-0.3	231	962	1537	25
iForest-10-0.3	229	948	1551	27
iForest-50-0.3	233	968	1531	23
OCSVM-SGD-ad.-sigm.-0.4-0.7	247	1041	1458	9
iForest-1000-0.3	235	975	1524	21
OCSVM-poly-auto-0.3	253	1069	1430	3
OCSVM-sigm.-0.2-0.4	251	1056	1443	5
iForest-500-0.3	235	972	1527	21
OCSVM-SGD-ad.-sigm.-0.2-0.1	49	0	2499	207
OCSVM-poly-0.8-0.3	254	1071	1428	2
OCSVM-SGD-poly-0.2-0.7	253	1065	1434	3
OCSVM-sigm.-auto-0.4	251	1054	1445	5
iForest-20-0.3	234	961	1538	22
OCSVM-sigm.-0.1-0.4	251	1049	1450	5
OCSVM-sigm.-0.8-0.4	251	1047	1452	5
OCSVM-SGD-lin.-0.1-0.6	251	1044	1455	5
OCSVM-SGD-lin.-0.2-0.6	251	1044	1455	5
OCSVM-SGD-lin.-0.3-0.6	251	1044	1455	5
OCSVM-SGD-lin.-0.4-0.6	251	1044	1455	5
OCSVM-SGD-lin.-0.6-0.6	251	1044	1455	5
OCSVM-SGD-lin.-0.8-0.6	251	1044	1455	5
OCSVM-SGD-ad.-sigm.-0.6-0.7	230	934	1565	26
iForest-30-0.3	232	941	1558	24
OCSVM-SGD-poly-0.3-0.9	253	1042	1457	3
OCSVM-sigm.-0.4-0.4	251	1025	1474	5
OCSVM-SGD-ad.-lin.-0.1-0.7	252	1030	1469	4

METHOD	TP	FP	TN	FN
OCSVM-SGD-ad.-lin.-0.2-0.7	252	1030	1469	4
OCSVM-SGD-ad.-lin.-0.3-0.7	252	1030	1469	4
OCSVM-SGD-ad.-lin.-0.4-0.7	252	1030	1469	4
OCSVM-SGD-ad.-lin.-0.6-0.7	252	1030	1469	4
OCSVM-SGD-ad.-lin.-0.8-0.7	252	1030	1469	4
OCSVM-SGD-ad.-poly-0.2-0.8	252	1029	1470	4
OCSVM-sigm.-0.6-0.4	251	1018	1481	5
OCSVM-SGD-sigm.-0.1-0.5	251	1013	1486	5
OCSVM-sigm.-0.3-0.4	251	1010	1489	5
OCSVM-poly-0.6-0.4	255	1028	1471	1
OCSVM-SGD-sigm.-0.2-0.5	253	1009	1490	3
OCSVM-rbf-0.4-0.1	256	1021	1478	0
OCSVM-rbf-0.3-0.2	256	1019	1480	0
OCSVM-SGD-rbf-0.8-0.4	256	1016	1483	0
OCSVM-SGD-ad.-rbf-0.2-0.5	256	987	1512	0
OCSVM-SGD-rbf-0.1-0.5	256	986	1513	0
OCSVM-SGD-rbf-0.6-0.4	256	982	1517	0
OCSVM-SGD-sigm.-0.3-0.5	251	949	1550	5
OCSVM-rbf-0.2-0.2	256	966	1533	0
OCSVM-SGD-ad.-poly-0.1-0.6	251	933	1566	5
OCSVM-SGD-poly-0.1-0.5	251	927	1572	5
OCSVM-poly-0.4-0.3	254	939	1560	2
OCSVM-rbf-scale-0.2	256	943	1556	0
OCSVM-SGD-poly-0.3-0.8	252	903	1596	4
OCSVM-rbf-0.3-0.1	256	921	1578	0
OCSVM-lin.-scale-0.3	252	900	1599	4
OCSVM-lin.-auto-0.3	252	900	1599	4
OCSVM-lin.-0.1-0.3	252	900	1599	4
OCSVM-lin.-0.2-0.3	252	900	1599	4
OCSVM-lin.-0.3-0.3	252	900	1599	4
OCSVM-lin.-0.4-0.3	252	900	1599	4
OCSVM-lin.-0.6-0.3	252	900	1599	4
OCSVM-lin.-0.8-0.3	252	900	1599	4
OCSVM-poly-auto-0.1	252	899	1600	4
OCSVM-poly-0.1-0.3	254	905	1594	2
OCSVM-poly-0.2-0.3	254	901	1598	2
OCSVM-SGD-rbf-0.4-0.4	256	910	1589	0

METHOD	TP	FP	TN	FN
OCSVM-SGD-sigm.-0.4-0.5	250	877	1622	6
OCSVM-SGD-poly-0.2-0.6	253	888	1611	3
OCSVM-poly-scale-0.3	254	891	1608	2
OCSVM-SGD-ad.-sigm.-0.1-0.2	68	50	2449	188
iForest-10-0.2	206	666	1833	50
OCSVM-poly-0.3-0.3	254	875	1624	2
iForest-20-0.2	208	669	1830	48
iForest-500-0.2	217	707	1792	39
iForest-1000-0.2	218	706	1793	38
OCSVM-poly-0.6-0.3	254	864	1635	2
OCSVM-SGD-lin.-0.1-0.5	251	848	1651	5
OCSVM-SGD-lin.-0.2-0.5	251	848	1651	5
OCSVM-SGD-lin.-0.3-0.5	251	848	1651	5
OCSVM-SGD-lin.-0.4-0.5	251	848	1651	5
OCSVM-SGD-lin.-0.6-0.5	251	848	1651	5
OCSVM-SGD-lin.-0.8-0.5	251	848	1651	5
iForest-100-0.2	217	698	1801	39
iForest-200-0.2	218	701	1798	38
OCSVM-SGD-rbf-0.3-0.4	256	865	1634	0
OCSVM-SGD-sigm.-0.6-0.5	238	783	1716	18
iForest-50-0.2	216	682	1817	40
OCSVM-SGD-ad.-sigm.-0.2-0.6	245	805	1694	11
OCSVM-SGD-ad.-rbf-0.8-0.1	67	33	2466	189
iForest-30-0.2	214	667	1832	42
OCSVM-rbf-0.1-0.2	256	847	1652	0
OCSVM-SGD-ad.-lin.-0.1-0.6	251	824	1675	5
OCSVM-SGD-ad.-lin.-0.2-0.6	251	824	1675	5
OCSVM-SGD-ad.-lin.-0.3-0.6	251	824	1675	5
OCSVM-SGD-ad.-lin.-0.4-0.6	251	824	1675	5
OCSVM-SGD-ad.-lin.-0.6-0.6	251	824	1675	5
OCSVM-SGD-ad.-lin.-0.8-0.6	251	824	1675	5
OCSVM-SGD-ad.-poly-0.2-0.7	252	824	1675	4
OCSVM-SGD-ad.-poly-0.3-0.9	251	817	1682	5
OCSVM-poly-auto-0.2	253	825	1674	3
OCSVM-SGD-ad.-rbf-0.6-0.1	74	59	2440	182
OCSVM-sigm.-0.8-0.3	247	790	1709	9
OCSVM-SGD-rbf-0.2-0.4	256	825	1674	0

METHOD	TP	FP	TN	FN
OCSVM-SGD-poly-0.3-0.7	251	789	1710	5
OCSVM-sigm.-0.1-0.3	252	792	1707	4
OCSVM-sigm.-scale-0.3	251	785	1714	5
OCSVM-sigm.-0.2-0.3	251	780	1719	5
OCSVM-SGD-ad.-rbf-0.8-0.4	256	798	1701	0
OCSVM-sigm.-auto-0.3	252	774	1725	4
OCSVM-SGD-ad.-rbf-0.6-0.4	256	787	1712	0
OCSVM-SGD-poly-0.4-0.9	252	770	1729	4
OCSVM-sigm.-0.6-0.3	248	753	1746	8
OCSVM-SGD-ad.-sigm.-0.1-0.6	247	744	1755	9
OCSVM-SGD-rbf-0.8-0.3	256	765	1734	0
OCSVM-rbf-0.2-0.1	256	756	1743	0
OCSVM-sigm.-0.3-0.3	250	729	1770	6
OCSVM-SGD-poly-0.2-0.5	251	729	1770	5
OCSVM-rbf-scale-0.1	256	746	1753	0
OCSVM-SGD-rbf-0.6-0.3	256	743	1756	0
OCSVM-sigm.-0.4-0.3	249	713	1786	7
OCSVM-SGD-ad.-poly-0.1-0.5	249	711	1788	7
OCSVM-rbf-auto-0.2	256	733	1766	0
OCSVM-SGD-poly-0.1-0.4	251	710	1789	5
OCSVM-SGD-sigm.-0.1-0.4	251	701	1798	5
OCSVM-SGD-ad.-sigm.-0.4-0.2	70	8	2491	186
OCSVM-SGD-ad.-poly-0.2-0.6	251	686	1813	5
OCSVM-SGD-sigm.-0.2-0.4	251	683	1816	5
OCSVM-SGD-ad.-poly-0.3-0.8	252	682	1817	4
OCSVM-SGD-rbf-0.1-0.4	256	695	1804	0
OCSVM-SGD-poly-0.4-0.8	251	665	1834	5
iForest-200-0.1	178	396	2103	78
iForest-10-0.1	167	352	2147	89
OCSVM-poly-0.8-0.2	253	663	1836	3
OCSVM-SGD-ad.-rbf-0.1-0.5	256	672	1827	0
OCSVM-poly-0.6-0.2	253	659	1840	3
OCSVM-SGD-poly-0.3-0.6	251	651	1848	5
iForest-500-0.1	180	394	2105	76
OCSVM-SGD-rbf-0.4-0.3	256	666	1833	0
iForest-100-0.1	179	387	2112	77
OCSVM-SGD-ad.-rbf-0.4-0.4	256	663	1836	0

METHOD	TP	FP	TN	FN
OCSVM-SGD-sigm.-0.8-0.5	213	504	1995	43
OCSVM-SGD-sigm.-0.3-0.4	247	622	1877	9
OCSVM-SGD-ad.-sigm.-0.8-0.3	73	3	2496	183
iForest-1000-0.1	183	393	2106	73
OCSVM-SGD-lin.-0.1-0.4	251	633	1866	5
OCSVM-SGD-lin.-0.2-0.4	251	633	1866	5
OCSVM-SGD-lin.-0.3-0.4	251	633	1866	5
OCSVM-SGD-lin.-0.4-0.4	251	633	1866	5
OCSVM-SGD-lin.-0.6-0.4	251	633	1866	5
OCSVM-SGD-lin.-0.8-0.4	251	633	1866	5
OCSVM-rbf-auto-0.1	256	649	1850	0
iForest-30-0.1	176	366	2133	80
iForest-20-0.1	173	354	2145	83
OCSVM-SGD-ad.-sigm.-0.2-0.2	84	39	2460	172
OCSVM-SGD-ad.-lin.-0.1-0.5	251	623	1876	5
OCSVM-SGD-ad.-lin.-0.2-0.5	251	623	1876	5
OCSVM-SGD-ad.-lin.-0.3-0.5	251	623	1876	5
OCSVM-SGD-ad.-lin.-0.4-0.5	251	623	1876	5
OCSVM-SGD-ad.-lin.-0.6-0.5	251	623	1876	5
OCSVM-SGD-ad.-lin.-0.8-0.5	251	623	1876	5
OCSVM-SGD-ad.-sigm.-0.3-0.6	241	583	1916	15
OCSVM-poly-scale-0.2	253	622	1877	3
OCSVM-poly-0.4-0.2	253	620	1879	3
OCSVM-rbf-0.1-0.1	256	629	1870	0
OCSVM-SGD-ad.-rbf-0.3-0.4	256	629	1870	0
OCSVM-SGD-rbf-0.3-0.3	256	627	1872	0
OCSVM-poly-0.3-0.2	253	613	1886	3
iForest-50-0.1	182	358	2141	74
OCSVM-poly-0.2-0.2	253	593	1906	3
OCSVM-poly-0.1-0.2	253	587	1912	3
OCSVM-SGD-ad.-sigm.-0.3-0.2	86	30	2469	170
OCSVM-sigm.-scale-0.2	251	576	1923	5
OCSVM-sigm.-auto-0.2	252	577	1922	4
OCSVM-SGD-poly-0.4-0.7	252	576	1923	4
OCSVM-SGD-ad.-poly-0.3-0.7	251	568	1931	5
OCSVM-sigm.-0.2-0.2	251	566	1933	5
OCSVM-sigm.-0.8-0.2	241	526	1973	15

METHOD	TP	FP	TN	FN
OCSVM-SGD-poly-0.2-0.4	252	554	1945	4
OCSVM-lin.-scale-0.2	252	550	1949	4
OCSVM-lin.-auto-0.2	252	550	1949	4
OCSVM-lin.-0.1-0.2	252	550	1949	4
OCSVM-lin.-0.2-0.2	252	550	1949	4
OCSVM-lin.-0.3-0.2	252	550	1949	4
OCSVM-lin.-0.4-0.2	252	550	1949	4
OCSVM-lin.-0.6-0.2	252	550	1949	4
OCSVM-lin.-0.8-0.2	252	550	1949	4
OCSVM-SGD-sigm.-0.4-0.4	244	522	1977	12
OCSVM-SGD-ad.-rbf-0.2-0.4	256	560	1939	0
OCSVM-SGD-ad.-poly-0.4-0.9	251	534	1965	5
OCSVM-SGD-ad.-sigm.-0.6-0.3	83	5	2494	173
OCSVM-SGD-rbf-0.2-0.3	256	546	1953	0
OCSVM-sigm.-0.6-0.2	247	514	1985	9
OCSVM-sigm.-0.1-0.2	252	528	1971	4
OCSVM-SGD-poly-0.3-0.5	252	519	1980	4
OCSVM-sigm.-0.4-0.2	250	508	1991	6
OCSVM-SGD-poly-0.1-0.3	252	510	1989	4
OCSVM-SGD-ad.-poly-0.2-0.5	248	494	2005	8
OCSVM-SGD-ad.-rbf-0.1-0.1	103	53	2446	153
OCSVM-sigm.-0.3-0.2	250	492	2007	6
OCSVM-SGD-poly-0.4-0.6	252	489	2010	4
OCSVM-SGD-ad.-sigm.-0.8-0.4	91	10	2489	165
OCSVM-SGD-lin.-0.1-0.3	252	468	2031	4
OCSVM-SGD-lin.-0.2-0.3	252	468	2031	4
OCSVM-SGD-lin.-0.3-0.3	252	468	2031	4
OCSVM-SGD-lin.-0.4-0.3	252	468	2031	4
OCSVM-SGD-lin.-0.6-0.3	252	468	2031	4
OCSVM-SGD-lin.-0.8-0.3	252	468	2031	4
OCSVM-SGD-ad.-poly-0.1-0.4	247	453	2046	9
OCSVM-SGD-ad.-poly-0.8-0.1	91	5	2494	165
OCSVM-SGD-ad.-sigm.-0.4-0.6	229	397	2102	27
OCSVM-SGD-ad.-poly-0.4-0.8	248	451	2048	8
OCSVM-SGD-rbf-0.8-0.2	256	472	2027	0
OCSVM-SGD-ad.-poly-0.3-0.6	248	449	2050	8
OCSVM-SGD-sigm.-0.6-0.4	222	369	2130	34

METHOD	TP	FP	TN	FN
OCSVM-SGD-ad.-poly-0.6-0.1	102	29	2470	154
OCSVM-SGD-poly-0.6-0.9	252	445	2054	4
OCSVM-SGD-ad.-lin.-0.1-0.4	247	431	2068	9
OCSVM-SGD-ad.-lin.-0.2-0.4	247	431	2068	9
OCSVM-SGD-ad.-lin.-0.3-0.4	247	431	2068	9
OCSVM-SGD-ad.-lin.-0.4-0.4	247	431	2068	9
OCSVM-SGD-ad.-lin.-0.6-0.4	247	431	2068	9
OCSVM-SGD-ad.-lin.-0.8-0.4	247	431	2068	9
OCSVM-SGD-rbf-0.6-0.2	256	438	2061	0
OCSVM-SGD-sigm.-0.1-0.3	249	418	2081	7
OCSVM-SGD-rbf-0.1-0.3	256	436	2063	0
OCSVM-SGD-sigm.-0.8-0.1	97	3	2496	159
OCSVM-SGD-sigm.-0.2-0.3	245	392	2107	11
OCSVM-SGD-ad.-poly-0.4-0.7	245	389	2110	11
OCSVM-SGD-poly-0.3-0.4	252	405	2094	4
OCSVM-SGD-poly-0.4-0.5	252	404	2095	4
OCSVM-SGD-poly-0.6-0.8	252	400	2099	4
OCSVM-SGD-ad.-sigm.-0.4-0.3	113	36	2463	143
OCSVM-SGD-poly-0.2-0.3	252	395	2104	4
OCSVM-SGD-ad.-sigm.-0.1-0.5	232	339	2160	24
OCSVM-SGD-ad.-sigm.-0.2-0.5	231	336	2163	25
OCSVM-SGD-ad.-sigm.-0.1-0.3	128	72	2427	128
OCSVM-SGD-rbf-0.4-0.2	256	385	2114	0
OCSVM-SGD-ad.-poly-0.3-0.5	241	345	2154	15
OCSVM-SGD-ad.-rbf-0.4-0.1	132	73	2426	124
OCSVM-SGD-sigm.-0.3-0.3	243	347	2152	13
OCSVM-poly-0.8-0.1	253	371	2128	3
OCSVM-SGD-ad.-poly-0.2-0.4	245	351	2148	11
OCSVM-SGD-ad.-poly-0.1-0.1	126	54	2445	130
OCSVM-poly-0.1-0.1	253	363	2136	3
OCSVM-poly-scale-0.1	253	362	2137	3
OCSVM-poly-0.4-0.1	253	361	2138	3
OCSVM-poly-0.2-0.1	253	357	2142	3
OCSVM-poly-0.6-0.1	253	357	2142	3
OCSVM-SGD-ad.-sigm.-0.8-0.5	119	32	2467	137
OCSVM-SGD-ad.-sigm.-0.6-0.6	194	213	2286	62
OCSVM-SGD-sigm.-0.8-0.4	202	232	2267	54

METHOD	TP	FP	TN	FN
OCSVM-SGD-ad.-poly-0.4-0.6	240	321	2178	16
OCSVM-SGD-poly-0.6-0.7	252	348	2151	4
OCSVM-SGD-ad.-rbf-0.4-0.3	255	354	2145	1
OCSVM-SGD-ad.-rbf-0.6-0.3	255	353	2146	1
OCSVM-poly-0.3-0.1	253	342	2157	3
OCSVM-SGD-ad.-rbf-0.1-0.4	256	349	2150	0
OCSVM-SGD-ad.-sigm.-0.6-0.4	123	34	2465	133
OCSVM-SGD-sigm.-0.4-0.3	238	299	2200	18
OCSVM-SGD-ad.-sigm.-0.2-0.3	141	72	2427	115
OCSVM-SGD-ad.-sigm.-0.8-0.6	157	108	2391	99
OCSVM-SGD-ad.-poly-0.4-0.1	128	40	2459	128
OCSVM-SGD-rbf-0.3-0.2	256	334	2165	0
OCSVM-SGD-ad.-sigm.-0.3-0.3	134	51	2448	122
OCSVM-SGD-poly-0.4-0.4	252	321	2178	4
OCSVM-SGD-ad.-rbf-0.3-0.3	256	329	2170	0
OCSVM-SGD-ad.-sigm.-0.3-0.5	227	257	2242	29
OCSVM-SGD-ad.-poly-0.6-0.9	240	279	2220	16
OCSVM-SGD-poly-0.3-0.3	252	302	2197	4
OCSVM-SGD-ad.-rbf-0.2-0.1	145	64	2435	111
OCSVM-SGD-ad.-poly-0.2-0.1	141	55	2444	115
OCSVM-SGD-ad.-rbf-0.8-0.3	253	301	2198	3
OCSVM-SGD-ad.-poly-0.1-0.3	237	265	2234	19
OCSVM-SGD-ad.-poly-0.3-0.1	135	40	2459	121
OCSVM-SGD-poly-0.1-0.2	250	292	2207	6
OCSVM-lin.-scale-0.1	251	293	2206	5
OCSVM-lin.-auto-0.1	251	293	2206	5
OCSVM-lin.-0.1-0.1	251	293	2206	5
OCSVM-lin.-0.2-0.1	251	293	2206	5
OCSVM-lin.-0.3-0.1	251	293	2206	5
OCSVM-lin.-0.4-0.1	251	293	2206	5
OCSVM-lin.-0.6-0.1	251	293	2206	5
OCSVM-lin.-0.8-0.1	251	293	2206	5
OCSVM-SGD-poly-0.6-0.6	252	290	2209	4
OCSVM-SGD-poly-0.8-0.9	252	289	2210	4
OCSVM-SGD-ad.-lin.-0.1-0.3	237	256	2243	19
OCSVM-SGD-ad.-lin.-0.2-0.3	237	256	2243	19
OCSVM-SGD-ad.-lin.-0.3-0.3	237	256	2243	19

METHOD	TP	FP	TN	FN
OCSVM-SGD-ad.-lin.-0.4-0.3	237	256	2243	19
OCSVM-SGD-ad.-lin.-0.6-0.3	237	256	2243	19
OCSVM-SGD-ad.-lin.-0.8-0.3	237	256	2243	19
OCSVM-SGD-ad.-sigm.-0.4-0.5	208	192	2307	48
OCSVM-SGD-ad.-sigm.-0.1-0.4	179	128	2371	77
OCSVM-SGD-ad.-poly-0.8-0.2	135	32	2467	121
OCSVM-sigm.-auto-0.1	251	279	2220	5
OCSVM-SGD-rbf-0.2-0.2	256	289	2210	0
OCSVM-SGD-ad.-poly-0.3-0.4	236	245	2254	20
OCSVM-SGD-ad.-poly-0.6-0.8	233	236	2263	23
OCSVM-SGD-lin.-0.1-0.2	251	271	2228	5
OCSVM-SGD-lin.-0.2-0.2	251	271	2228	5
OCSVM-SGD-lin.-0.3-0.2	251	271	2228	5
OCSVM-SGD-lin.-0.4-0.2	251	271	2228	5
OCSVM-SGD-lin.-0.6-0.2	251	271	2228	5
OCSVM-SGD-lin.-0.8-0.2	251	271	2228	5
OCSVM-sigm.-0.1-0.1	251	270	2229	5
OCSVM-SGD-ad.-rbf-0.3-0.1	157	73	2426	99
OCSVM-sigm.-0.8-0.1	241	247	2252	15
OCSVM-sigm.-0.6-0.1	243	251	2248	13
OCSVM-SGD-ad.-sigm.-0.3-0.4	175	109	2390	81
OCSVM-SGD-ad.-poly-0.4-0.5	235	234	2265	21
OCSVM-SGD-ad.-sigm.-0.6-0.5	167	90	2409	89
OCSVM-SGD-ad.-sigm.-0.2-0.4	189	134	2365	67
OCSVM-SGD-ad.-sigm.-0.4-0.4	159	71	2428	97
OCSVM-SGD-ad.-rbf-0.2-0.3	254	264	2235	2
OCSVM-SGD-ad.-poly-0.2-0.3	232	218	2281	24
OCSVM-sigm.-0.2-0.1	249	252	2247	7
OCSVM-sigm.-0.3-0.1	249	252	2247	7
OCSVM-sigm.-scale-0.1	250	253	2246	6
OCSVM-sigm.-0.4-0.1	249	247	2252	7
OCSVM-SGD-ad.-poly-0.6-0.7	230	206	2293	26
OCSVM-SGD-ad.-poly-0.6-0.2	153	48	2451	103
OCSVM-SGD-ad.-rbf-0.8-0.2	185	108	2391	71
OCSVM-SGD-sigm.-0.6-0.3	212	159	2340	44
OCSVM-SGD-sigm.-0.6-0.1	133	4	2495	123
OCSVM-SGD-poly-0.6-0.5	252	236	2263	4

METHOD	TP	FP	TN	FN
OCSVM-SGD-poly-0.8-0.8	252	236	2263	4
OCSVM-SGD-poly-0.2-0.2	251	232	2267	5
OCSVM-SGD-sigm.-0.8-0.3	180	89	2410	76
OCSVM-SGD-ad.-poly-0.8-0.3	159	46	2453	97
OCSVM-SGD-ad.-lin.-0.1-0.1	162	51	2448	94
OCSVM-SGD-ad.-lin.-0.2-0.1	162	51	2448	94
OCSVM-SGD-ad.-lin.-0.3-0.1	162	51	2448	94
OCSVM-SGD-ad.-lin.-0.4-0.1	162	51	2448	94
OCSVM-SGD-ad.-lin.-0.6-0.1	162	51	2448	94
OCSVM-SGD-ad.-lin.-0.8-0.1	162	51	2448	94
OCSVM-SGD-ad.-poly-0.4-0.4	230	179	2320	26
OCSVM-SGD-poly-0.4-0.3	252	220	2279	4
OCSVM-SGD-sigm.-0.8-0.2	148	22	2477	108
OCSVM-SGD-poly-0.8-0.7	252	216	2283	4
OCSVM-SGD-ad.-poly-0.3-0.3	228	160	2339	28
OCSVM-SGD-rbf-0.1-0.2	256	209	2290	0
OCSVM-SGD-ad.-poly-0.8-0.9	229	157	2342	27
OCSVM-SGD-sigm.-0.1-0.2	238	172	2327	18
OCSVM-SGD-ad.-poly-0.6-0.6	228	149	2350	28
OCSVM-SGD-ad.-poly-0.6-0.3	185	71	2428	71
OCSVM-SGD-ad.-rbf-0.1-0.3	251	185	2314	5
OCSVM-SGD-sigm.-0.2-0.2	240	165	2334	16
OCSVM-SGD-ad.-poly-0.8-0.4	184	65	2434	72
OCSVM-SGD-ad.-poly-0.8-0.8	223	133	2366	33
OCSVM-SGD-ad.-poly-0.4-0.2	187	70	2429	69
OCSVM-SGD-poly-0.6-0.4	250	179	2320	6
OCSVM-SGD-poly-0.3-0.2	251	178	2321	5
OCSVM-SGD-ad.-poly-0.1-0.2	213	110	2389	43
OCSVM-SGD-ad.-poly-0.3-0.2	200	87	2412	56
OCSVM-SGD-poly-0.8-0.6	250	171	2328	6
OCSVM-SGD-ad.-poly-0.6-0.5	225	128	2371	31
OCSVM-SGD-ad.-poly-0.6-0.4	206	94	2405	50
OCSVM-SGD-ad.-poly-0.8-0.7	217	110	2389	39
OCSVM-SGD-ad.-poly-0.8-0.6	209	96	2403	47
OCSVM-SGD-ad.-poly-0.8-0.5	200	80	2419	56
OCSVM-SGD-sigm.-0.3-0.2	233	135	2364	23
OCSVM-SGD-ad.-lin.-0.1-0.2	227	122	2377	29

METHOD	TP	FP	TN	FN
OCSVM-SGD-ad.-lin.-0.2-0.2	227	122	2377	29
OCSVM-SGD-ad.-lin.-0.3-0.2	227	122	2377	29
OCSVM-SGD-ad.-lin.-0.4-0.2	227	122	2377	29
OCSVM-SGD-ad.-lin.-0.6-0.2	227	122	2377	29
OCSVM-SGD-ad.-lin.-0.8-0.2	227	122	2377	29
OCSVM-SGD-ad.-poly-0.4-0.3	219	108	2391	37
OCSVM-SGD-ad.-rbf-0.6-0.2	236	135	2364	20
OCSVM-SGD-sigm.-0.4-0.2	219	106	2393	37
OCSVM-SGD-ad.-poly-0.2-0.2	218	103	2396	38
OCSVM-SGD-sigm.-0.6-0.2	190	55	2444	66
OCSVM-SGD-ad.-rbf-0.1-0.2	214	92	2407	42
OCSVM-SGD-ad.-rbf-0.3-0.2	247	145	2354	9
OCSVM-SGD-ad.-rbf-0.4-0.2	251	143	2356	5
OCSVM-SGD-poly-0.8-0.5	248	134	2365	8
OCSVM-SGD-sigm.-0.4-0.1	176	18	2481	80
OCSVM-SGD-ad.-rbf-0.2-0.2	247	125	2374	9
OCSVM-SGD-poly-0.4-0.2	248	122	2377	8
OCSVM-SGD-sigm.-0.1-0.1	191	32	2467	65
OCSVM-SGD-rbf-0.6-0.1	256	130	2369	0
OCSVM-SGD-rbf-0.8-0.1	253	120	2379	3
OCSVM-SGD-rbf-0.3-0.1	256	124	2375	0
OCSVM-SGD-rbf-0.4-0.1	256	122	2377	0
OCSVM-SGD-sigm.-0.3-0.1	193	28	2471	63
OCSVM-SGD-poly-0.6-0.3	250	108	2391	6
OCSVM-SGD-poly-0.8-0.4	245	91	2408	11
OCSVM-SGD-sigm.-0.2-0.1	208	34	2465	48
OCSVM-SGD-poly-0.1-0.1	242	75	2424	14
OCSVM-SGD-lin.-0.1-0.1	245	79	2420	11
OCSVM-SGD-lin.-0.2-0.1	245	79	2420	11
OCSVM-SGD-lin.-0.3-0.1	245	79	2420	11
OCSVM-SGD-lin.-0.4-0.1	245	79	2420	11
OCSVM-SGD-lin.-0.6-0.1	245	79	2420	11
OCSVM-SGD-lin.-0.8-0.1	245	79	2420	11
OCSVM-SGD-poly-0.8-0.3	241	70	2429	15
OCSVM-SGD-poly-0.8-0.1	200	14	2485	56
OCSVM-SGD-rbf-0.2-0.1	256	85	2414	0
OCSVM-SGD-poly-0.2-0.1	246	70	2429	10

METHOD	TP	FP	TN	FN
OCSVM-SGD-poly-0.3-0.1	241	62	2437	15
OCSVM-SGD-poly-0.6-0.2	241	61	2438	15
OCSVM-SGD-poly-0.4-0.1	236	47	2452	20
OCSVM-SGD-poly-0.8-0.2	235	45	2454	21
OCSVM-SGD-poly-0.6-0.1	227	29	2470	29
OCSVM-SGD-rbf-0.1-0.1	256	46	2453	0

Table A.1: Number of true/false positive/negative predictions on logins of the novelty detection methods and hyper-parameter combinations under test. The dataset used for this evaluation contains 256 positive and 2499 negative examples.

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
OCSVM-SGD-ad.-sigm.-0.8-0.1	0.950	0.074	0.138	0.007 ms	0.089 ms
OCSVM-SGD-ad.-sigm.-0.6-0.1	1.000	0.078	0.145	0.007 ms	0.089 ms
OCSVM-SGD-sigm.-0.8-0.9	0.088	0.918	0.161	0.012 ms	0.004 ms
OCSVM-SGD-ad.-sigm.-0.8-0.8	0.090	0.957	0.164	0.007 ms	0.724 ms
OCSVM-SGD-ad.-sigm.-0.8-0.9	0.090	0.961	0.164	0.007 ms	0.724 ms
OCSVM-SGD-sigm.-0.8-0.8	0.092	0.914	0.167	0.012 ms	0.004 ms
OCSVM-SGD-ad.-sigm.-0.6-0.8	0.092	0.984	0.169	0.007 ms	0.724 ms
OCSVM-SGD-ad.-sigm.-0.6-0.9	0.093	0.996	0.169	0.007 ms	0.727 ms
OCSVM-SGD-sigm.-0.6-0.9	0.093	0.961	0.170	0.011 ms	0.004 ms
OCSVM-SGD-ad.-sigm.-0.4-0.9	0.093	1.000	0.170	0.007 ms	0.726 ms
OCSVM-SGD-ad.-sigm.-0.3-0.9	0.094	1.000	0.172	0.007 ms	0.723 ms
OCSVM-SGD-ad.-rbf-0.8-0.9	0.094	1.000	0.172	0.008 ms	0.843 ms
OCSVM-SGD-ad.-rbf-0.6-0.9	0.095	1.000	0.173	0.008 ms	0.835 ms
OCSVM-SGD-rbf-0.8-0.9	0.095	1.000	0.173	0.015 ms	0.005 ms
OCSVM-SGD-ad.-rbf-0.4-0.9	0.096	1.000	0.174	0.008 ms	0.824 ms
OCSVM-SGD-ad.-sigm.-0.2-0.9	0.096	1.000	0.174	0.007 ms	0.709 ms
OCSVM-SGD-rbf-0.6-0.9	0.096	1.000	0.175	0.014 ms	0.005 ms
OCSVM-SGD-ad.-rbf-0.3-0.9	0.096	1.000	0.175	0.008 ms	0.819 ms
OCSVM-SGD-ad.-sigm.-0.1-0.9	0.096	1.000	0.176	0.007 ms	0.700 ms
OCSVM-SGD-rbf-0.4-0.9	0.097	1.000	0.177	0.013 ms	0.005 ms
OCSVM-SGD-ad.-rbf-0.2-0.9	0.097	1.000	0.177	0.008 ms	0.802 ms
OCSVM-SGD-rbf-0.3-0.9	0.097	1.000	0.178	0.012 ms	0.005 ms
OCSVM-SGD-sigm.-0.4-0.9	0.098	0.996	0.179	0.010 ms	0.004 ms
OCSVM-SGD-rbf-0.2-0.9	0.098	1.000	0.179	0.011 ms	0.005 ms
OCSVM-SGD-ad.-rbf-0.1-0.9	0.098	1.000	0.179	0.008 ms	0.790 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
OCSVM-SGD-rbf-0.1-0.9	0.099	1.000	0.180	0.011 ms	0.005 ms
OCSVM-SGD-ad.-sigm.-0.4-0.8	0.099	0.988	0.180	0.007 ms	0.678 ms
OCSVM-SGD-ad.-rbf-0.8-0.8	0.099	1.000	0.180	0.008 ms	0.810 ms
OCSVM-SGD-sigm.-0.3-0.9	0.100	1.000	0.181	0.010 ms	0.004 ms
OCSVM-SGD-sigm.-0.1-0.9	0.100	1.000	0.182	0.010 ms	0.004 ms
OCSVM-SGD-sigm.-0.2-0.9	0.100	1.000	0.183	0.010 ms	0.004 ms
OCSVM-SGD-ad.-rbf-0.6-0.8	0.101	1.000	0.183	0.008 ms	0.788 ms
OCSVM-poly-0.8-0.9	0.101	1.000	0.184	0.019 ms	0.010 ms
OCSVM-sigm.-0.6-0.9	0.101	1.000	0.184	0.038 ms	0.018 ms
OCSVM-sigm.-0.8-0.9	0.101	1.000	0.184	0.038 ms	0.018 ms
OCSVM-SGD-ad.-rbf-0.4-0.8	0.102	1.000	0.185	0.008 ms	0.778 ms
OCSVM-sigm.-0.3-0.9	0.102	1.000	0.186	0.038 ms	0.018 ms
OCSVM-poly-0.4-0.9	0.102	1.000	0.186	0.019 ms	0.010 ms
OCSVM-sigm.-0.4-0.9	0.102	1.000	0.186	0.038 ms	0.018 ms
OCSVM-SGD-sigm.-0.6-0.8	0.103	0.957	0.186	0.011 ms	0.004 ms
OCSVM-sigm.-0.2-0.9	0.103	1.000	0.186	0.038 ms	0.018 ms
OCSVM-rbf-scale-0.9	0.103	1.000	0.186	0.028 ms	0.019 ms
OCSVM-rbf-0.4-0.9	0.103	1.000	0.186	0.030 ms	0.020 ms
OCSVM-sigm.-0.1-0.9	0.103	1.000	0.188	0.034 ms	0.017 ms
OCSVM-SGD-ad.-rbf-0.3-0.8	0.104	1.000	0.188	0.008 ms	0.762 ms
OCSVM-poly-0.2-0.9	0.104	1.000	0.188	0.019 ms	0.010 ms
OCSVM-rbf-0.3-0.9	0.104	1.000	0.188	0.030 ms	0.020 ms
OCSVM-sigm.-scale-0.9	0.104	1.000	0.189	0.038 ms	0.018 ms
OCSVM-poly-auto-0.9	0.104	1.000	0.189	0.019 ms	0.010 ms
OCSVM-SGD-rbf-0.8-0.8	0.105	1.000	0.189	0.013 ms	0.005 ms
OCSVM-SGD-rbf-0.4-0.8	0.105	1.000	0.190	0.012 ms	0.005 ms
OCSVM-SGD-rbf-0.6-0.8	0.105	1.000	0.190	0.012 ms	0.005 ms
OCSVM-sigm.-auto-0.9	0.105	1.000	0.190	0.029 ms	0.014 ms
OCSVM-poly-scale-0.9	0.105	1.000	0.191	0.019 ms	0.010 ms
OCSVM-SGD-rbf-0.3-0.8	0.106	1.000	0.191	0.012 ms	0.005 ms
OCSVM-SGD-rbf-0.2-0.8	0.106	1.000	0.192	0.011 ms	0.005 ms
OCSVM-SGD-ad.-sigm.-0.3-0.8	0.106	0.992	0.192	0.007 ms	0.641 ms
OCSVM-rbf-0.2-0.9	0.106	1.000	0.192	0.028 ms	0.020 ms
OCSVM-SGD-ad.-rbf-0.2-0.8	0.106	1.000	0.192	0.008 ms	0.740 ms
OCSVM-SGD-ad.-sigm.-0.1-0.8	0.106	1.000	0.192	0.007 ms	0.636 ms
OCSVM-lin.-scale-0.9	0.107	1.000	0.193	0.017 ms	0.009 ms
OCSVM-lin.-auto-0.9	0.107	1.000	0.193	0.017 ms	0.009 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
OCSVM-lin.-0.1-0.9	0.107	1.000	0.193	0.017 ms	0.009 ms
OCSVM-lin.-0.2-0.9	0.107	1.000	0.193	0.017 ms	0.009 ms
OCSVM-lin.-0.3-0.9	0.107	1.000	0.193	0.017 ms	0.009 ms
OCSVM-lin.-0.4-0.9	0.107	1.000	0.193	0.017 ms	0.009 ms
OCSVM-lin.-0.6-0.9	0.107	1.000	0.193	0.017 ms	0.009 ms
OCSVM-lin.-0.8-0.9	0.107	1.000	0.193	0.017 ms	0.009 ms
OCSVM-rbf-auto-0.9	0.107	1.000	0.193	0.027 ms	0.019 ms
OCSVM-poly-0.3-0.9	0.107	1.000	0.193	0.019 ms	0.010 ms
OCSVM-rbf-0.8-0.9	0.107	1.000	0.194	0.030 ms	0.021 ms
OCSVM-poly-0.1-0.9	0.107	1.000	0.194	0.020 ms	0.010 ms
OCSVM-rbf-0.6-0.9	0.107	1.000	0.194	0.029 ms	0.021 ms
OCSVM-rbf-0.1-0.9	0.108	1.000	0.194	0.027 ms	0.019 ms
OCSVM-SGD-rbf-0.1-0.8	0.109	1.000	0.196	0.011 ms	0.005 ms
OCSVM-SGD-ad.-rbf-0.8-0.7	0.109	1.000	0.196	0.008 ms	0.742 ms
OCSVM-poly-0.8-0.8	0.109	1.000	0.197	0.018 ms	0.009 ms
OCSVM-SGD-ad.-rbf-0.1-0.8	0.110	1.000	0.197	0.008 ms	0.715 ms
OCSVM-poly-0.2-0.8	0.110	1.000	0.198	0.019 ms	0.009 ms
OCSVM-poly-0.4-0.8	0.110	1.000	0.198	0.018 ms	0.009 ms
OCSVM-poly-0.6-0.9	0.110	1.000	0.198	0.019 ms	0.010 ms
OCSVM-poly-0.1-0.8	0.110	1.000	0.199	0.019 ms	0.009 ms
OCSVM-SGD-sigm.-0.8-0.7	0.113	0.891	0.200	0.012 ms	0.004 ms
OCSVM-rbf-0.4-0.8	0.111	1.000	0.200	0.029 ms	0.018 ms
OCSVM-SGD-ad.-rbf-0.6-0.7	0.111	1.000	0.200	0.008 ms	0.720 ms
OCSVM-rbf-0.3-0.8	0.111	1.000	0.200	0.029 ms	0.018 ms
OCSVM-sigm.-0.8-0.8	0.111	0.996	0.200	0.037 ms	0.016 ms
OCSVM-SGD-sigm.-0.2-0.8	0.112	0.996	0.201	0.010 ms	0.004 ms
OCSVM-SGD-sigm.-0.1-0.8	0.112	1.000	0.201	0.010 ms	0.004 ms
OCSVM-SGD-ad.-sigm.-0.2-0.8	0.112	0.996	0.202	0.007 ms	0.609 ms
OCSVM-SGD-sigm.-0.4-0.8	0.113	0.996	0.202	0.010 ms	0.004 ms
OCSVM-rbf-0.6-0.8	0.113	1.000	0.203	0.028 ms	0.019 ms
OCSVM-sigm.-0.6-0.8	0.113	0.996	0.203	0.037 ms	0.017 ms
OCSVM-SGD-ad.-sigm.-0.4-0.1	1.000	0.113	0.204	0.007 ms	0.089 ms
OCSVM-rbf-0.1-0.8	0.113	1.000	0.204	0.026 ms	0.017 ms
OCSVM-SGD-sigm.-0.3-0.8	0.113	0.996	0.204	0.010 ms	0.004 ms
OCSVM-sigm.-0.4-0.8	0.114	1.000	0.204	0.037 ms	0.016 ms
OCSVM-SGD-ad.-rbf-0.4-0.7	0.114	1.000	0.205	0.008 ms	0.704 ms
OCSVM-rbf-auto-0.8	0.114	1.000	0.205	0.026 ms	0.017 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
OCSVM-rbf-0.8-0.8	0.114	1.000	0.205	0.028 ms	0.019 ms
OCSVM-sigm.-0.1-0.8	0.114	1.000	0.205	0.032 ms	0.015 ms
OCSVM-poly-auto-0.8	0.115	1.000	0.206	0.018 ms	0.009 ms
OCSVM-sigm.-0.2-0.8	0.115	1.000	0.207	0.036 ms	0.016 ms
OCSVM-SGD-rbf-0.8-0.7	0.116	1.000	0.207	0.012 ms	0.005 ms
OCSVM-rbf-0.2-0.8	0.116	1.000	0.208	0.027 ms	0.018 ms
OCSVM-sigm.-0.3-0.8	0.116	1.000	0.208	0.037 ms	0.016 ms
OCSVM-lin.-scale-0.8	0.116	1.000	0.208	0.017 ms	0.008 ms
OCSVM-lin.-auto-0.8	0.116	1.000	0.208	0.016 ms	0.008 ms
OCSVM-lin.-0.1-0.8	0.116	1.000	0.208	0.016 ms	0.008 ms
OCSVM-lin.-0.2-0.8	0.116	1.000	0.208	0.016 ms	0.008 ms
OCSVM-lin.-0.3-0.8	0.116	1.000	0.208	0.016 ms	0.008 ms
OCSVM-lin.-0.4-0.8	0.116	1.000	0.208	0.016 ms	0.008 ms
OCSVM-lin.-0.6-0.8	0.116	1.000	0.208	0.016 ms	0.008 ms
OCSVM-lin.-0.8-0.8	0.116	1.000	0.208	0.016 ms	0.008 ms
OCSVM-SGD-rbf-0.6-0.7	0.116	1.000	0.208	0.012 ms	0.005 ms
OCSVM-SGD-ad.-rbf-0.3-0.7	0.117	1.000	0.209	0.008 ms	0.685 ms
OCSVM-sigm.-scale-0.8	0.117	1.000	0.210	0.036 ms	0.016 ms
OCSVM-poly-scale-0.8	0.117	1.000	0.210	0.019 ms	0.009 ms
OCSVM-sigm.-auto-0.8	0.117	1.000	0.210	0.027 ms	0.013 ms
OCSVM-rbf-scale-0.8	0.117	1.000	0.210	0.027 ms	0.017 ms
OCSVM-rbf-0.8-0.7	0.118	1.000	0.212	0.027 ms	0.017 ms
OCSVM-poly-0.8-0.7	0.118	1.000	0.212	0.017 ms	0.008 ms
OCSVM-rbf-0.4-0.7	0.119	1.000	0.213	0.027 ms	0.016 ms
OCSVM-rbf-0.6-0.7	0.121	1.000	0.216	0.027 ms	0.017 ms
OCSVM-rbf-scale-0.7	0.122	1.000	0.217	0.026 ms	0.016 ms
OCSVM-poly-0.3-0.8	0.122	1.000	0.217	0.018 ms	0.009 ms
OCSVM-poly-0.6-0.8	0.122	1.000	0.218	0.018 ms	0.009 ms
OCSVM-SGD-ad.-rbf-0.2-0.7	0.123	1.000	0.219	0.008 ms	0.651 ms
OCSVM-SGD-rbf-0.3-0.7	0.124	1.000	0.220	0.012 ms	0.005 ms
OCSVM-SGD-ad.-rbf-0.8-0.6	0.124	1.000	0.221	0.008 ms	0.658 ms
OCSVM-sigm.-0.1-0.7	0.124	1.000	0.221	0.031 ms	0.013 ms
OCSVM-poly-0.2-0.7	0.124	1.000	0.221	0.018 ms	0.008 ms
OCSVM-rbf-0.2-0.7	0.125	1.000	0.221	0.026 ms	0.016 ms
OCSVM-SGD-rbf-0.4-0.7	0.125	1.000	0.222	0.012 ms	0.005 ms
OCSVM-SGD-rbf-0.2-0.7	0.125	1.000	0.222	0.011 ms	0.005 ms
OCSVM-poly-0.4-0.7	0.125	1.000	0.222	0.017 ms	0.008 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
OCSVM-rbf-0.6-0.6	0.125	1.000	0.223	0.025 ms	0.015 ms
OCSVM-sigm.-0.8-0.7	0.126	0.992	0.223	0.035 ms	0.014 ms
OCSVM-sigm.-0.3-0.7	0.126	0.996	0.224	0.035 ms	0.014 ms
OCSVM-SGD-sigm.-0.6-0.7	0.127	0.934	0.224	0.011 ms	0.004 ms
OCSVM-SGD-poly-0.1-0.9	0.126	1.000	0.224	0.012 ms	0.005 ms
OCSVM-rbf-0.8-0.6	0.127	1.000	0.225	0.025 ms	0.015 ms
OCSVM-sigm.-auto-0.7	0.127	1.000	0.225	0.026 ms	0.011 ms
OCSVM-poly-0.1-0.7	0.127	1.000	0.226	0.017 ms	0.008 ms
OCSVM-sigm.-0.6-0.7	0.129	0.992	0.228	0.035 ms	0.014 ms
OCSVM-sigm.-0.4-0.7	0.129	1.000	0.228	0.034 ms	0.014 ms
OCSVM-SGD-ad.-rbf-0.6-0.6	0.129	1.000	0.228	0.008 ms	0.632 ms
OCSVM-rbf-0.4-0.6	0.129	1.000	0.229	0.025 ms	0.015 ms
OCSVM-SGD-ad.-poly-0.1-0.9	0.129	0.996	0.229	0.011 ms	0.577 ms
OCSVM-rbf-auto-0.7	0.130	1.000	0.230	0.025 ms	0.015 ms
OCSVM-SGD-rbf-0.1-0.7	0.130	1.000	0.230	0.011 ms	0.005 ms
OCSVM-rbf-0.3-0.7	0.130	1.000	0.230	0.027 ms	0.016 ms
OCSVM-lin.-scale-0.7	0.130	1.000	0.231	0.016 ms	0.007 ms
OCSVM-lin.-auto-0.7	0.130	1.000	0.231	0.016 ms	0.008 ms
OCSVM-lin.-0.1-0.7	0.130	1.000	0.231	0.016 ms	0.007 ms
OCSVM-lin.-0.2-0.7	0.130	1.000	0.231	0.016 ms	0.007 ms
OCSVM-lin.-0.3-0.7	0.130	1.000	0.231	0.016 ms	0.007 ms
OCSVM-lin.-0.4-0.7	0.130	1.000	0.231	0.015 ms	0.007 ms
OCSVM-lin.-0.6-0.7	0.130	1.000	0.231	0.016 ms	0.007 ms
OCSVM-lin.-0.8-0.7	0.130	1.000	0.231	0.016 ms	0.007 ms
OCSVM-poly-0.8-0.6	0.131	1.000	0.231	0.016 ms	0.007 ms
OCSVM-sigm.-0.2-0.7	0.131	0.996	0.231	0.034 ms	0.014 ms
OCSVM-SGD-sigm.-0.2-0.7	0.132	0.996	0.232	0.009 ms	0.004 ms
OCSVM-SGD-sigm.-0.3-0.7	0.132	0.992	0.232	0.010 ms	0.004 ms
OCSVM-sigm.-scale-0.7	0.132	0.996	0.233	0.034 ms	0.014 ms
OCSVM-poly-scale-0.7	0.132	1.000	0.233	0.017 ms	0.008 ms
OCSVM-SGD-sigm.-0.4-0.7	0.133	0.988	0.234	0.010 ms	0.004 ms
OCSVM-rbf-0.1-0.7	0.133	1.000	0.235	0.025 ms	0.016 ms
OCSVM-SGD-sigm.-0.1-0.7	0.133	1.000	0.236	0.010 ms	0.004 ms
OCSVM-poly-0.4-0.6	0.134	1.000	0.237	0.016 ms	0.007 ms
OCSVM-SGD-ad.-rbf-0.1-0.7	0.134	1.000	0.237	0.008 ms	0.597 ms
OCSVM-poly-0.3-0.7	0.135	1.000	0.237	0.017 ms	0.008 ms
OCSVM-rbf-0.8-0.5	0.135	1.000	0.238	0.023 ms	0.013 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
OCSVM-SGD-ad.-rbf-0.4-0.6	0.135	1.000	0.238	0.008 ms	0.603 ms
OCSVM-rbf-scale-0.6	0.136	1.000	0.239	0.024 ms	0.014 ms
OCSVM-poly-auto-0.7	0.136	1.000	0.239	0.017 ms	0.008 ms
OCSVM-poly-0.6-0.7	0.137	1.000	0.240	0.017 ms	0.008 ms
OCSVM-rbf-0.3-0.6	0.137	1.000	0.242	0.026 ms	0.015 ms
OCSVM-rbf-0.6-0.4	0.138	1.000	0.243	0.021 ms	0.011 ms
OCSVM-rbf-0.2-0.6	0.139	1.000	0.243	0.024 ms	0.014 ms
OCSVM-poly-scale-0.6	0.139	1.000	0.245	0.016 ms	0.007 ms
OCSVM-SGD-lin.-0.1-0.9	0.140	1.000	0.246	0.007 ms	0.004 ms
OCSVM-SGD-lin.-0.2-0.9	0.140	1.000	0.246	0.007 ms	0.004 ms
OCSVM-SGD-lin.-0.3-0.9	0.140	1.000	0.246	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.4-0.9	0.140	1.000	0.246	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.6-0.9	0.140	1.000	0.246	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.8-0.9	0.140	1.000	0.246	0.007 ms	0.003 ms
OCSVM-SGD-poly-0.1-0.8	0.141	1.000	0.247	0.012 ms	0.005 ms
OCSVM-sigm.-0.8-0.6	0.141	0.988	0.247	0.032 ms	0.012 ms
OCSVM-SGD-rbf-0.6-0.6	0.142	1.000	0.249	0.011 ms	0.005 ms
OCSVM-rbf-0.1-0.6	0.142	1.000	0.249	0.023 ms	0.014 ms
OCSVM-lin.-scale-0.6	0.142	0.996	0.249	0.015 ms	0.006 ms
OCSVM-lin.-auto-0.6	0.142	0.996	0.249	0.015 ms	0.006 ms
OCSVM-lin.-0.1-0.6	0.142	0.996	0.249	0.015 ms	0.006 ms
OCSVM-lin.-0.2-0.6	0.142	0.996	0.249	0.015 ms	0.006 ms
OCSVM-lin.-0.3-0.6	0.142	0.996	0.249	0.014 ms	0.006 ms
OCSVM-lin.-0.4-0.6	0.142	0.996	0.249	0.014 ms	0.006 ms
OCSVM-lin.-0.6-0.6	0.142	0.996	0.249	0.014 ms	0.006 ms
OCSVM-lin.-0.8-0.6	0.142	0.996	0.249	0.015 ms	0.006 ms
OCSVM-SGD-rbf-0.8-0.6	0.142	1.000	0.249	0.012 ms	0.005 ms
OCSVM-rbf-auto-0.6	0.142	1.000	0.249	0.023 ms	0.013 ms
OCSVM-rbf-0.6-0.5	0.143	1.000	0.251	0.023 ms	0.013 ms
OCSVM-rbf-0.4-0.4	0.143	1.000	0.251	0.021 ms	0.011 ms
OCSVM-poly-0.1-0.6	0.143	1.000	0.251	0.016 ms	0.007 ms
OCSVM-poly-0.8-0.5	0.144	0.996	0.251	0.015 ms	0.006 ms
OCSVM-rbf-0.8-0.4	0.144	1.000	0.252	0.021 ms	0.011 ms
OCSVM-SGD-ad.-sigm.-0.1-0.1	1.000	0.145	0.253	0.007 ms	0.088 ms
OCSVM-poly-0.3-0.6	0.145	1.000	0.253	0.016 ms	0.007 ms
OCSVM-SGD-ad.-rbf-0.3-0.6	0.145	1.000	0.253	0.008 ms	0.565 ms
OCSVM-SGD-ad.-lin.-0.1-0.9	0.145	0.996	0.253	0.006 ms	0.429 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
OCSVM-SGD-ad.-lin.-0.2-0.9	0.145	0.996	0.253	0.006 ms	0.431 ms
OCSVM-SGD-ad.-lin.-0.3-0.9	0.145	0.996	0.253	0.006 ms	0.426 ms
OCSVM-SGD-ad.-lin.-0.4-0.9	0.145	0.996	0.253	0.006 ms	0.426 ms
OCSVM-SGD-ad.-lin.-0.6-0.9	0.145	0.996	0.253	0.006 ms	0.426 ms
OCSVM-SGD-ad.-lin.-0.8-0.9	0.145	0.996	0.253	0.006 ms	0.427 ms
OCSVM-sigm.-0.1-0.6	0.145	0.996	0.253	0.029 ms	0.011 ms
OCSVM-rbf-0.4-0.5	0.145	1.000	0.254	0.024 ms	0.013 ms
iForest-1000-0.5	0.146	0.969	0.254	6.832 ms	3.639 ms
OCSVM-sigm.-auto-0.6	0.146	0.996	0.254	0.024 ms	0.010 ms
iForest-500-0.5	0.146	0.965	0.254	3.420 ms	1.828 ms
OCSVM-sigm.-0.4-0.6	0.146	0.992	0.254	0.031 ms	0.012 ms
iForest-200-0.5	0.147	0.965	0.254	1.375 ms	0.735 ms
OCSVM-SGD-ad.-poly-0.1-0.8	0.146	0.992	0.255	0.010 ms	0.513 ms
iForest-30-0.5	0.147	0.957	0.255	0.210 ms	0.112 ms
OCSVM-poly-auto-0.6	0.146	0.992	0.255	0.015 ms	0.007 ms
OCSVM-sigm.-0.2-0.6	0.146	1.000	0.255	0.032 ms	0.012 ms
OCSVM-rbf-scale-0.5	0.147	1.000	0.256	0.022 ms	0.012 ms
iForest-10-0.5	0.148	0.957	0.256	0.073 ms	0.039 ms
OCSVM-rbf-0.8-0.3	0.147	1.000	0.257	0.017 ms	0.010 ms
OCSVM-SGD-rbf-0.4-0.6	0.148	1.000	0.257	0.011 ms	0.005 ms
iForest-50-0.5	0.149	0.969	0.258	0.346 ms	0.185 ms
OCSVM-sigm.-0.3-0.6	0.148	0.992	0.258	0.032 ms	0.012 ms
iForest-100-0.5	0.149	0.969	0.258	0.689 ms	0.370 ms
OCSVM-sigm.-0.6-0.6	0.148	0.992	0.258	0.033 ms	0.012 ms
iForest-20-0.5	0.149	0.957	0.258	0.142 ms	0.076 ms
OCSVM-poly-0.2-0.6	0.148	1.000	0.258	0.016 ms	0.007 ms
OCSVM-sigm.-scale-0.6	0.150	1.000	0.260	0.032 ms	0.012 ms
OCSVM-poly-0.2-0.5	0.150	0.996	0.260	0.015 ms	0.006 ms
OCSVM-rbf-0.1-0.5	0.150	1.000	0.261	0.021 ms	0.012 ms
OCSVM-poly-0.4-0.5	0.150	0.996	0.261	0.015 ms	0.006 ms
OCSVM-SGD-rbf-0.3-0.6	0.151	1.000	0.262	0.011 ms	0.005 ms
OCSVM-SGD-ad.-rbf-0.2-0.6	0.151	1.000	0.262	0.008 ms	0.541 ms
OCSVM-poly-0.1-0.5	0.152	0.996	0.264	0.014 ms	0.006 ms
OCSVM-poly-0.6-0.6	0.153	1.000	0.265	0.016 ms	0.007 ms
OCSVM-rbf-0.6-0.3	0.153	1.000	0.265	0.018 ms	0.009 ms
OCSVM-poly-scale-0.5	0.153	0.996	0.265	0.015 ms	0.006 ms
OCSVM-SGD-rbf-0.2-0.6	0.153	1.000	0.265	0.011 ms	0.005 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
OCSVM-rbf-0.2-0.5	0.154	1.000	0.266	0.022 ms	0.012 ms
OCSVM-rbf-0.8-0.2	0.154	1.000	0.267	0.014 ms	0.008 ms
OCSVM-rbf-0.3-0.5	0.154	1.000	0.267	0.024 ms	0.013 ms
OCSVM-rbf-0.8-0.1	0.154	1.000	0.268	0.010 ms	0.006 ms
OCSVM-poly-auto-0.5	0.155	0.992	0.268	0.014 ms	0.006 ms
OCSVM-SGD-sigm.-0.1-0.6	0.157	0.988	0.271	0.010 ms	0.004 ms
OCSVM-SGD-lin.-0.1-0.8	0.158	0.996	0.272	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.2-0.8	0.158	0.996	0.272	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.3-0.8	0.158	0.996	0.272	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.4-0.8	0.158	0.996	0.272	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.6-0.8	0.158	0.996	0.272	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.8-0.8	0.158	0.996	0.272	0.007 ms	0.003 ms
OCSVM-sigm.-auto-0.5	0.158	0.988	0.272	0.022 ms	0.008 ms
OCSVM-SGD-rbf-0.1-0.6	0.158	1.000	0.273	0.011 ms	0.005 ms
OCSVM-SGD-ad.-sigm.-0.8-0.2	0.932	0.160	0.273	0.007 ms	0.090 ms
OCSVM-SGD-sigm.-0.2-0.6	0.159	0.988	0.273	0.010 ms	0.004 ms
OCSVM-poly-0.3-0.5	0.159	0.996	0.274	0.015 ms	0.006 ms
OCSVM-sigm.-0.1-0.5	0.160	0.988	0.275	0.026 ms	0.010 ms
OCSVM-SGD-poly-0.1-0.7	0.160	0.992	0.275	0.012 ms	0.005 ms
OCSVM-rbf-auto-0.5	0.160	1.000	0.276	0.022 ms	0.011 ms
OCSVM-SGD-ad.-lin.-0.1-0.8	0.161	0.984	0.277	0.006 ms	0.384 ms
OCSVM-SGD-ad.-lin.-0.2-0.8	0.161	0.984	0.277	0.006 ms	0.385 ms
OCSVM-SGD-ad.-lin.-0.3-0.8	0.161	0.984	0.277	0.006 ms	0.381 ms
OCSVM-SGD-ad.-lin.-0.4-0.8	0.161	0.984	0.277	0.006 ms	0.383 ms
OCSVM-SGD-ad.-lin.-0.6-0.8	0.161	0.984	0.277	0.006 ms	0.381 ms
OCSVM-SGD-ad.-lin.-0.8-0.8	0.161	0.984	0.277	0.006 ms	0.382 ms
OCSVM-sigm.-0.3-0.5	0.162	0.980	0.279	0.029 ms	0.011 ms
OCSVM-SGD-ad.-rbf-0.8-0.5	0.162	1.000	0.279	0.008 ms	0.518 ms
OCSVM-SGD-poly-0.2-0.9	0.164	1.000	0.282	0.012 ms	0.005 ms
iForest-100-0.4	0.166	0.949	0.283	0.689 ms	0.369 ms
OCSVM-sigm.-0.8-0.5	0.165	0.988	0.283	0.029 ms	0.010 ms
OCSVM-rbf-0.3-0.4	0.165	1.000	0.283	0.021 ms	0.011 ms
iForest-10-0.4	0.168	0.926	0.284	0.073 ms	0.039 ms
OCSVM-sigm.-0.2-0.5	0.166	0.984	0.284	0.029 ms	0.011 ms
OCSVM-sigm.-0.6-0.5	0.166	0.984	0.284	0.029 ms	0.011 ms
OCSVM-SGD-sigm.-0.3-0.6	0.166	0.984	0.284	0.010 ms	0.004 ms
OCSVM-SGD-ad.-rbf-0.6-0.5	0.166	1.000	0.284	0.008 ms	0.506 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
OCSVM-sigm.-scale-0.5	0.166	0.988	0.284	0.029 ms	0.010 ms
OCSVM-sigm.-0.4-0.5	0.166	0.984	0.285	0.028 ms	0.011 ms
iForest-200-0.4	0.168	0.949	0.285	1.373 ms	0.735 ms
OCSVM-lin.-scale-0.5	0.167	0.988	0.285	0.014 ms	0.006 ms
OCSVM-lin.-auto-0.5	0.167	0.988	0.285	0.013 ms	0.006 ms
OCSVM-lin.-0.1-0.5	0.167	0.988	0.285	0.013 ms	0.006 ms
OCSVM-lin.-0.2-0.5	0.167	0.988	0.285	0.013 ms	0.006 ms
OCSVM-lin.-0.3-0.5	0.167	0.988	0.285	0.013 ms	0.006 ms
OCSVM-lin.-0.4-0.5	0.167	0.988	0.285	0.013 ms	0.006 ms
OCSVM-lin.-0.6-0.5	0.167	0.988	0.285	0.013 ms	0.006 ms
OCSVM-lin.-0.8-0.5	0.167	0.988	0.285	0.013 ms	0.006 ms
OCSVM-SGD-sigm.-0.6-0.6	0.169	0.930	0.285	0.011 ms	0.004 ms
iForest-50-0.4	0.168	0.938	0.285	0.347 ms	0.185 ms
iForest-500-0.4	0.168	0.953	0.286	3.420 ms	1.823 ms
OCSVM-rbf-0.4-0.3	0.167	1.000	0.286	0.018 ms	0.009 ms
OCSVM-SGD-ad.-sigm.-0.6-0.2	0.956	0.168	0.286	0.007 ms	0.089 ms
iForest-1000-0.4	0.168	0.953	0.286	6.828 ms	3.642 ms
OCSVM-rbf-0.6-0.2	0.167	1.000	0.287	0.014 ms	0.007 ms
OCSVM-SGD-ad.-sigm.-0.3-0.1	1.000	0.168	0.288	0.007 ms	0.089 ms
iForest-30-0.4	0.170	0.938	0.288	0.210 ms	0.112 ms
OCSVM-SGD-sigm.-0.4-0.6	0.169	0.984	0.288	0.010 ms	0.004 ms
OCSVM-SGD-ad.-sigm.-0.1-0.7	0.169	0.977	0.289	0.007 ms	0.407 ms
OCSVM-rbf-0.2-0.4	0.169	1.000	0.290	0.019 ms	0.010 ms
iForest-20-0.4	0.171	0.941	0.290	0.141 ms	0.076 ms
OCSVM-poly-0.8-0.4	0.171	0.996	0.292	0.013 ms	0.005 ms
OCSVM-SGD-ad.-poly-0.1-0.7	0.171	0.984	0.292	0.010 ms	0.438 ms
OCSVM-poly-auto-0.4	0.172	0.988	0.293	0.012 ms	0.005 ms
OCSVM-poly-0.6-0.5	0.172	0.996	0.294	0.015 ms	0.006 ms
OCSVM-SGD-ad.-sigm.-0.8-0.7	0.178	0.844	0.294	0.007 ms	0.355 ms
OCSVM-SGD-ad.-poly-0.2-0.9	0.173	0.984	0.295	0.010 ms	0.433 ms
OCSVM-rbf-0.6-0.1	0.173	1.000	0.295	0.011 ms	0.006 ms
OCSVM-SGD-poly-0.2-0.8	0.174	0.988	0.295	0.012 ms	0.005 ms
OCSVM-rbf-0.3-0.3	0.173	1.000	0.295	0.018 ms	0.009 ms
OCSVM-SGD-rbf-0.8-0.5	0.173	1.000	0.295	0.011 ms	0.005 ms
OCSVM-poly-0.3-0.4	0.174	0.996	0.297	0.013 ms	0.005 ms
OCSVM-SGD-sigm.-0.8-0.6	0.178	0.887	0.297	0.012 ms	0.004 ms
OCSVM-SGD-lin.-0.1-0.7	0.175	0.988	0.297	0.007 ms	0.004 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
OCSVM-SGD-lin.-0.2-0.7	0.175	0.988	0.297	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.3-0.7	0.175	0.988	0.297	0.007 ms	0.004 ms
OCSVM-SGD-lin.-0.4-0.7	0.175	0.988	0.297	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.6-0.7	0.175	0.988	0.297	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.8-0.7	0.175	0.988	0.297	0.007 ms	0.003 ms
OCSVM-rbf-0.1-0.4	0.175	1.000	0.298	0.019 ms	0.010 ms
OCSVM-rbf-0.4-0.2	0.176	1.000	0.299	0.014 ms	0.007 ms
OCSVM-SGD-rbf-0.6-0.5	0.177	1.000	0.300	0.011 ms	0.005 ms
OCSVM-SGD-ad.-sigm.-0.2-0.7	0.177	0.984	0.301	0.007 ms	0.394 ms
OCSVM-SGD-rbf-0.4-0.5	0.179	1.000	0.303	0.011 ms	0.005 ms
OCSVM-rbf-0.1-0.3	0.179	1.000	0.304	0.016 ms	0.008 ms
OCSVM-rbf-auto-0.4	0.179	1.000	0.304	0.019 ms	0.009 ms
OCSVM-SGD-poly-0.1-0.6	0.180	0.988	0.305	0.012 ms	0.005 ms
OCSVM-poly-0.2-0.4	0.180	0.996	0.305	0.013 ms	0.005 ms
OCSVM-SGD-ad.-rbf-0.1-0.6	0.181	1.000	0.306	0.008 ms	0.464 ms
OCSVM-SGD-ad.-sigm.-0.3-0.7	0.182	0.977	0.307	0.007 ms	0.384 ms
OCSVM-SGD-ad.-rbf-0.4-0.5	0.181	1.000	0.307	0.008 ms	0.465 ms
OCSVM-sigm.-scale-0.4	0.182	0.980	0.307	0.025 ms	0.009 ms
OCSVM-rbf-scale-0.4	0.182	1.000	0.308	0.019 ms	0.010 ms
OCSVM-SGD-rbf-0.3-0.5	0.182	1.000	0.308	0.011 ms	0.005 ms
OCSVM-rbf-scale-0.3	0.184	1.000	0.311	0.016 ms	0.008 ms
OCSVM-poly-scale-0.4	0.187	0.996	0.315	0.013 ms	0.005 ms
OCSVM-poly-0.1-0.4	0.187	0.996	0.315	0.013 ms	0.005 ms
OCSVM-rbf-0.2-0.3	0.187	1.000	0.315	0.017 ms	0.008 ms
OCSVM-rbf-auto-0.3	0.188	1.000	0.317	0.015 ms	0.008 ms
iForest-200-0.3	0.192	0.906	0.317	1.374 ms	0.734 ms
OCSVM-SGD-rbf-0.2-0.5	0.189	1.000	0.317	0.011 ms	0.005 ms
OCSVM-lin.-scale-0.4	0.189	0.980	0.318	0.012 ms	0.005 ms
OCSVM-lin.-auto-0.4	0.189	0.980	0.318	0.012 ms	0.005 ms
OCSVM-lin.-0.1-0.4	0.189	0.980	0.318	0.012 ms	0.005 ms
OCSVM-lin.-0.2-0.4	0.189	0.980	0.318	0.012 ms	0.005 ms
OCSVM-lin.-0.3-0.4	0.189	0.980	0.318	0.012 ms	0.005 ms
OCSVM-lin.-0.4-0.4	0.189	0.980	0.318	0.012 ms	0.005 ms
OCSVM-lin.-0.6-0.4	0.189	0.980	0.318	0.012 ms	0.005 ms
OCSVM-lin.-0.8-0.4	0.189	0.980	0.318	0.012 ms	0.005 ms
OCSVM-poly-0.4-0.4	0.189	0.996	0.318	0.013 ms	0.005 ms
OCSVM-SGD-ad.-rbf-0.3-0.5	0.189	1.000	0.318	0.008 ms	0.447 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
iForest-100-0.3	0.194	0.902	0.319	0.688 ms	0.369 ms
iForest-10-0.3	0.195	0.895	0.320	0.073 ms	0.039 ms
iForest-50-0.3	0.194	0.910	0.320	0.346 ms	0.185 ms
OCSVM-SGD-ad.-sigm.-0.4-0.7	0.192	0.965	0.320	0.007 ms	0.366 ms
iForest-1000-0.3	0.194	0.918	0.321	6.829 ms	3.639 ms
OCSVM-poly-auto-0.3	0.191	0.988	0.321	0.010 ms	0.004 ms
OCSVM-sigm.-0.2-0.4	0.192	0.980	0.321	0.025 ms	0.009 ms
iForest-500-0.3	0.195	0.918	0.321	3.420 ms	1.821 ms
OCSVM-SGD-ad.-sigm.-0.2-0.1	1.000	0.191	0.321	0.007 ms	0.089 ms
OCSVM-poly-0.8-0.3	0.192	0.992	0.321	0.011 ms	0.004 ms
OCSVM-SGD-poly-0.2-0.7	0.192	0.988	0.321	0.012 ms	0.005 ms
OCSVM-sigm.-auto-0.4	0.192	0.980	0.322	0.019 ms	0.007 ms
iForest-20-0.3	0.196	0.914	0.323	0.141 ms	0.076 ms
OCSVM-sigm.-0.1-0.4	0.193	0.980	0.323	0.022 ms	0.008 ms
OCSVM-sigm.-0.8-0.4	0.193	0.980	0.323	0.025 ms	0.009 ms
OCSVM-SGD-lin.-0.1-0.6	0.194	0.980	0.324	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.2-0.6	0.194	0.980	0.324	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.3-0.6	0.194	0.980	0.324	0.007 ms	0.004 ms
OCSVM-SGD-lin.-0.4-0.6	0.194	0.980	0.324	0.007 ms	0.004 ms
OCSVM-SGD-lin.-0.6-0.6	0.194	0.980	0.324	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.8-0.6	0.194	0.980	0.324	0.007 ms	0.003 ms
OCSVM-SGD-ad.-sigm.-0.6-0.7	0.198	0.898	0.324	0.007 ms	0.342 ms
iForest-30-0.3	0.198	0.906	0.325	0.210 ms	0.112 ms
OCSVM-SGD-poly-0.3-0.9	0.195	0.988	0.326	0.011 ms	0.005 ms
OCSVM-sigm.-0.4-0.4	0.197	0.980	0.328	0.025 ms	0.009 ms
OCSVM-SGD-ad.-lin.-0.1-0.7	0.197	0.984	0.328	0.006 ms	0.318 ms
OCSVM-SGD-ad.-lin.-0.2-0.7	0.197	0.984	0.328	0.006 ms	0.318 ms
OCSVM-SGD-ad.-lin.-0.3-0.7	0.197	0.984	0.328	0.006 ms	0.317 ms
OCSVM-SGD-ad.-lin.-0.4-0.7	0.197	0.984	0.328	0.006 ms	0.318 ms
OCSVM-SGD-ad.-lin.-0.6-0.7	0.197	0.984	0.328	0.006 ms	0.317 ms
OCSVM-SGD-ad.-lin.-0.8-0.7	0.197	0.984	0.328	0.006 ms	0.318 ms
OCSVM-SGD-ad.-poly-0.2-0.8	0.197	0.984	0.328	0.010 ms	0.386 ms
OCSVM-sigm.-0.6-0.4	0.198	0.980	0.329	0.025 ms	0.009 ms
OCSVM-SGD-sigm.-0.1-0.5	0.199	0.980	0.330	0.009 ms	0.004 ms
OCSVM-sigm.-0.3-0.4	0.199	0.980	0.331	0.026 ms	0.009 ms
OCSVM-poly-0.6-0.4	0.199	0.996	0.331	0.013 ms	0.005 ms
OCSVM-SGD-sigm.-0.2-0.5	0.200	0.988	0.333	0.009 ms	0.004 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
OCSVM-rbf-0.4-0.1	0.200	1.000	0.334	0.010 ms	0.005 ms
OCSVM-rbf-0.3-0.2	0.201	1.000	0.334	0.014 ms	0.007 ms
OCSVM-SGD-rbf-0.8-0.4	0.201	1.000	0.335	0.011 ms	0.005 ms
OCSVM-SGD-ad.-rbf-0.2-0.5	0.206	1.000	0.342	0.008 ms	0.413 ms
OCSVM-SGD-rbf-0.1-0.5	0.206	1.000	0.342	0.011 ms	0.005 ms
OCSVM-SGD-rbf-0.6-0.4	0.207	1.000	0.343	0.011 ms	0.005 ms
OCSVM-SGD-sigm.-0.3-0.5	0.209	0.980	0.345	0.010 ms	0.004 ms
OCSVM-rbf-0.2-0.2	0.209	1.000	0.346	0.013 ms	0.007 ms
OCSVM-SGD-ad.-poly-0.1-0.6	0.212	0.980	0.349	0.010 ms	0.356 ms
OCSVM-SGD-poly-0.1-0.5	0.213	0.980	0.350	0.012 ms	0.005 ms
OCSVM-poly-0.4-0.3	0.213	0.992	0.351	0.011 ms	0.004 ms
OCSVM-rbf-scale-0.2	0.214	1.000	0.352	0.013 ms	0.007 ms
OCSVM-SGD-poly-0.3-0.8	0.218	0.984	0.357	0.011 ms	0.005 ms
OCSVM-rbf-0.3-0.1	0.218	1.000	0.357	0.010 ms	0.005 ms
OCSVM-lin.-scale-0.3	0.219	0.984	0.358	0.010 ms	0.004 ms
OCSVM-lin.-auto-0.3	0.219	0.984	0.358	0.009 ms	0.004 ms
OCSVM-lin.-0.1-0.3	0.219	0.984	0.358	0.009 ms	0.004 ms
OCSVM-lin.-0.2-0.3	0.219	0.984	0.358	0.009 ms	0.004 ms
OCSVM-lin.-0.3-0.3	0.219	0.984	0.358	0.009 ms	0.004 ms
OCSVM-lin.-0.4-0.3	0.219	0.984	0.358	0.009 ms	0.004 ms
OCSVM-lin.-0.6-0.3	0.219	0.984	0.358	0.009 ms	0.004 ms
OCSVM-lin.-0.8-0.3	0.219	0.984	0.358	0.009 ms	0.004 ms
OCSVM-poly-auto-0.1	0.219	0.984	0.358	0.004 ms	0.002 ms
OCSVM-poly-0.1-0.3	0.219	0.992	0.359	0.010 ms	0.004 ms
OCSVM-poly-0.2-0.3	0.220	0.992	0.360	0.011 ms	0.004 ms
OCSVM-SGD-rbf-0.4-0.4	0.220	1.000	0.360	0.011 ms	0.005 ms
OCSVM-SGD-sigm.-0.4-0.5	0.222	0.977	0.362	0.010 ms	0.004 ms
OCSVM-SGD-poly-0.2-0.6	0.222	0.988	0.362	0.011 ms	0.005 ms
OCSVM-poly-scale-0.3	0.222	0.992	0.363	0.011 ms	0.004 ms
OCSVM-SGD-ad.-sigm.-0.1-0.2	0.576	0.266	0.364	0.007 ms	0.103 ms
iForest-10-0.2	0.236	0.805	0.365	0.073 ms	0.039 ms
OCSVM-poly-0.3-0.3	0.225	0.992	0.367	0.011 ms	0.004 ms
iForest-20-0.2	0.237	0.812	0.367	0.142 ms	0.076 ms
iForest-500-0.2	0.235	0.848	0.368	3.419 ms	1.822 ms
iForest-1000-0.2	0.236	0.852	0.369	6.832 ms	3.646 ms
OCSVM-poly-0.6-0.3	0.227	0.992	0.370	0.011 ms	0.004 ms
OCSVM-SGD-lin.-0.1-0.5	0.228	0.980	0.370	0.007 ms	0.003 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
OCSVM-SGD-lin.-0.2-0.5	0.228	0.980	0.370	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.3-0.5	0.228	0.980	0.370	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.4-0.5	0.228	0.980	0.370	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.6-0.5	0.228	0.980	0.370	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.8-0.5	0.228	0.980	0.370	0.007 ms	0.003 ms
iForest-100-0.2	0.237	0.848	0.371	0.688 ms	0.368 ms
iForest-200-0.2	0.237	0.852	0.371	1.375 ms	0.738 ms
OCSVM-SGD-rbf-0.3-0.4	0.228	1.000	0.372	0.011 ms	0.005 ms
OCSVM-SGD-sigm.-0.6-0.5	0.233	0.930	0.373	0.011 ms	0.004 ms
iForest-50-0.2	0.241	0.844	0.374	0.347 ms	0.192 ms
OCSVM-SGD-ad.-sigm.-0.2-0.6	0.233	0.957	0.375	0.007 ms	0.299 ms
OCSVM-SGD-ad.-rbf-0.8-0.1	0.670	0.262	0.376	0.008 ms	0.139 ms
iForest-30-0.2	0.243	0.836	0.376	0.210 ms	0.112 ms
OCSVM-rbf-0.1-0.2	0.232	1.000	0.377	0.013 ms	0.006 ms
OCSVM-SGD-ad.-lin.-0.1-0.6	0.233	0.980	0.377	0.006 ms	0.272 ms
OCSVM-SGD-ad.-lin.-0.2-0.6	0.233	0.980	0.377	0.006 ms	0.271 ms
OCSVM-SGD-ad.-lin.-0.3-0.6	0.233	0.980	0.377	0.006 ms	0.271 ms
OCSVM-SGD-ad.-lin.-0.4-0.6	0.233	0.980	0.377	0.006 ms	0.271 ms
OCSVM-SGD-ad.-lin.-0.6-0.6	0.233	0.980	0.377	0.006 ms	0.271 ms
OCSVM-SGD-ad.-lin.-0.8-0.6	0.233	0.980	0.377	0.006 ms	0.271 ms
OCSVM-SGD-ad.-poly-0.2-0.7	0.234	0.984	0.378	0.010 ms	0.327 ms
OCSVM-SGD-ad.-poly-0.3-0.9	0.235	0.980	0.379	0.010 ms	0.322 ms
OCSVM-poly-auto-0.2	0.235	0.988	0.379	0.008 ms	0.003 ms
OCSVM-SGD-ad.-rbf-0.6-0.1	0.556	0.289	0.380	0.008 ms	0.147 ms
OCSVM-sigm.-0.8-0.3	0.238	0.965	0.382	0.020 ms	0.007 ms
OCSVM-SGD-rbf-0.2-0.4	0.237	1.000	0.383	0.011 ms	0.005 ms
OCSVM-SGD-poly-0.3-0.7	0.241	0.980	0.387	0.011 ms	0.005 ms
OCSVM-sigm.-0.1-0.3	0.241	0.984	0.388	0.018 ms	0.006 ms
OCSVM-sigm.-scale-0.3	0.242	0.980	0.389	0.020 ms	0.007 ms
OCSVM-sigm.-0.2-0.3	0.243	0.980	0.390	0.020 ms	0.007 ms
OCSVM-SGD-ad.-rbf-0.8-0.4	0.243	1.000	0.391	0.008 ms	0.365 ms
OCSVM-sigm.-auto-0.3	0.246	0.984	0.393	0.015 ms	0.006 ms
OCSVM-SGD-ad.-rbf-0.6-0.4	0.245	1.000	0.394	0.008 ms	0.360 ms
OCSVM-SGD-poly-0.4-0.9	0.247	0.984	0.394	0.011 ms	0.005 ms
OCSVM-sigm.-0.6-0.3	0.248	0.969	0.395	0.020 ms	0.007 ms
OCSVM-SGD-ad.-sigm.-0.1-0.6	0.249	0.965	0.396	0.007 ms	0.284 ms
OCSVM-SGD-rbf-0.8-0.3	0.251	1.000	0.401	0.010 ms	0.005 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
OCSVM-rbf-0.2-0.1	0.253	1.000	0.404	0.009 ms	0.005 ms
OCSVM-sigm.-0.3-0.3	0.255	0.977	0.405	0.020 ms	0.007 ms
OCSVM-SGD-poly-0.2-0.5	0.256	0.980	0.406	0.011 ms	0.005 ms
OCSVM-rbf-scale-0.1	0.255	1.000	0.407	0.009 ms	0.005 ms
OCSVM-SGD-rbf-0.6-0.3	0.256	1.000	0.408	0.011 ms	0.005 ms
OCSVM-sigm.-0.4-0.3	0.259	0.973	0.409	0.020 ms	0.007 ms
OCSVM-SGD-ad.-poly-0.1-0.5	0.259	0.973	0.410	0.010 ms	0.294 ms
OCSVM-rbf-auto-0.2	0.259	1.000	0.411	0.012 ms	0.006 ms
OCSVM-SGD-poly-0.1-0.4	0.261	0.980	0.412	0.012 ms	0.005 ms
OCSVM-SGD-sigm.-0.1-0.4	0.264	0.980	0.416	0.010 ms	0.004 ms
OCSVM-SGD-ad.-sigm.-0.4-0.2	0.897	0.273	0.419	0.007 ms	0.091 ms
OCSVM-SGD-ad.-poly-0.2-0.6	0.268	0.980	0.421	0.010 ms	0.288 ms
OCSVM-SGD-sigm.-0.2-0.4	0.269	0.980	0.422	0.009 ms	0.004 ms
OCSVM-SGD-ad.-poly-0.3-0.8	0.270	0.984	0.424	0.010 ms	0.284 ms
OCSVM-SGD-rbf-0.1-0.4	0.269	1.000	0.424	0.011 ms	0.005 ms
OCSVM-SGD-poly-0.4-0.8	0.274	0.980	0.428	0.011 ms	0.005 ms
iForest-200-0.1	0.310	0.695	0.429	1.374 ms	0.734 ms
iForest-10-0.1	0.322	0.652	0.431	0.075 ms	0.040 ms
OCSVM-poly-0.8-0.2	0.276	0.988	0.432	0.008 ms	0.003 ms
OCSVM-SGD-ad.-rbf-0.1-0.5	0.276	1.000	0.432	0.008 ms	0.324 ms
OCSVM-poly-0.6-0.2	0.277	0.988	0.433	0.008 ms	0.003 ms
OCSVM-SGD-poly-0.3-0.6	0.278	0.980	0.434	0.011 ms	0.005 ms
iForest-500-0.1	0.314	0.703	0.434	3.426 ms	1.826 ms
OCSVM-SGD-rbf-0.4-0.3	0.278	1.000	0.435	0.011 ms	0.005 ms
iForest-100-0.1	0.316	0.699	0.436	0.687 ms	0.368 ms
OCSVM-SGD-ad.-rbf-0.4-0.4	0.279	1.000	0.436	0.008 ms	0.322 ms
OCSVM-SGD-sigm.-0.8-0.5	0.297	0.832	0.438	0.011 ms	0.004 ms
OCSVM-SGD-sigm.-0.3-0.4	0.284	0.965	0.439	0.009 ms	0.004 ms
OCSVM-SGD-ad.-sigm.-0.8-0.3	0.961	0.285	0.440	0.007 ms	0.089 ms
iForest-1000-0.1	0.318	0.715	0.440	6.838 ms	3.645 ms
OCSVM-SGD-lin.-0.1-0.4	0.284	0.980	0.440	0.007 ms	0.004 ms
OCSVM-SGD-lin.-0.2-0.4	0.284	0.980	0.440	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.3-0.4	0.284	0.980	0.440	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.4-0.4	0.284	0.980	0.440	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.6-0.4	0.284	0.980	0.440	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.8-0.4	0.284	0.980	0.440	0.007 ms	0.003 ms
OCSVM-rbf-auto-0.1	0.283	1.000	0.441	0.008 ms	0.004 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
iForest-30-0.1	0.325	0.688	0.441	0.210 ms	0.112 ms
iForest-20-0.1	0.328	0.676	0.442	0.141 ms	0.076 ms
OCSVM-SGD-ad.-sigm.-0.2-0.2	0.683	0.328	0.443	0.007 ms	0.100 ms
OCSVM-SGD-ad.-lin.-0.1-0.5	0.287	0.980	0.444	0.006 ms	0.226 ms
OCSVM-SGD-ad.-lin.-0.2-0.5	0.287	0.980	0.444	0.006 ms	0.226 ms
OCSVM-SGD-ad.-lin.-0.3-0.5	0.287	0.980	0.444	0.006 ms	0.226 ms
OCSVM-SGD-ad.-lin.-0.4-0.5	0.287	0.980	0.444	0.006 ms	0.225 ms
OCSVM-SGD-ad.-lin.-0.6-0.5	0.287	0.980	0.444	0.006 ms	0.225 ms
OCSVM-SGD-ad.-lin.-0.8-0.5	0.287	0.980	0.444	0.006 ms	0.226 ms
OCSVM-SGD-ad.-sigm.-0.3-0.6	0.292	0.941	0.446	0.007 ms	0.243 ms
OCSVM-poly-scale-0.2	0.289	0.988	0.447	0.008 ms	0.003 ms
OCSVM-poly-0.4-0.2	0.290	0.988	0.448	0.008 ms	0.003 ms
OCSVM-rbf-0.1-0.1	0.289	1.000	0.449	0.008 ms	0.004 ms
OCSVM-SGD-ad.-rbf-0.3-0.4	0.289	1.000	0.449	0.008 ms	0.313 ms
OCSVM-SGD-rbf-0.3-0.3	0.290	1.000	0.450	0.011 ms	0.005 ms
OCSVM-poly-0.3-0.2	0.292	0.988	0.451	0.008 ms	0.003 ms
iForest-50-0.1	0.337	0.711	0.457	0.346 ms	0.185 ms
OCSVM-poly-0.2-0.2	0.299	0.988	0.459	0.008 ms	0.003 ms
OCSVM-poly-0.1-0.2	0.301	0.988	0.462	0.008 ms	0.003 ms
OCSVM-SGD-ad.-sigm.-0.3-0.2	0.741	0.336	0.462	0.007 ms	0.097 ms
OCSVM-sigm.-scale-0.2	0.304	0.980	0.464	0.015 ms	0.005 ms
OCSVM-sigm.-auto-0.2	0.304	0.984	0.465	0.011 ms	0.004 ms
OCSVM-SGD-poly-0.4-0.7	0.304	0.984	0.465	0.011 ms	0.005 ms
OCSVM-SGD-ad.-poly-0.3-0.7	0.306	0.980	0.467	0.010 ms	0.253 ms
OCSVM-sigm.-0.2-0.2	0.307	0.980	0.468	0.014 ms	0.005 ms
OCSVM-sigm.-0.8-0.2	0.314	0.941	0.471	0.015 ms	0.005 ms
OCSVM-SGD-poly-0.2-0.4	0.313	0.984	0.475	0.011 ms	0.005 ms
OCSVM-lin.-scale-0.2	0.314	0.984	0.476	0.008 ms	0.003 ms
OCSVM-lin.-auto-0.2	0.314	0.984	0.476	0.007 ms	0.003 ms
OCSVM-lin.-0.1-0.2	0.314	0.984	0.476	0.007 ms	0.003 ms
OCSVM-lin.-0.2-0.2	0.314	0.984	0.476	0.007 ms	0.003 ms
OCSVM-lin.-0.3-0.2	0.314	0.984	0.476	0.007 ms	0.003 ms
OCSVM-lin.-0.4-0.2	0.314	0.984	0.476	0.007 ms	0.003 ms
OCSVM-lin.-0.6-0.2	0.314	0.984	0.476	0.007 ms	0.003 ms
OCSVM-lin.-0.8-0.2	0.314	0.984	0.476	0.007 ms	0.003 ms
OCSVM-SGD-sigm.-0.4-0.4	0.319	0.953	0.477	0.009 ms	0.004 ms
OCSVM-SGD-ad.-rbf-0.2-0.4	0.314	1.000	0.478	0.008 ms	0.291 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
OCSVM-SGD-ad.-poly-0.4-0.9	0.320	0.980	0.482	0.010 ms	0.244 ms
OCSVM-SGD-ad.-sigm.-0.6-0.3	0.943	0.324	0.483	0.007 ms	0.090 ms
OCSVM-SGD-rbf-0.2-0.3	0.319	1.000	0.484	0.011 ms	0.005 ms
OCSVM-sigm.-0.6-0.2	0.325	0.965	0.486	0.015 ms	0.005 ms
OCSVM-sigm.-0.1-0.2	0.323	0.984	0.486	0.013 ms	0.005 ms
OCSVM-SGD-poly-0.3-0.5	0.327	0.984	0.491	0.011 ms	0.005 ms
OCSVM-sigm.-0.4-0.2	0.330	0.977	0.493	0.014 ms	0.005 ms
OCSVM-SGD-poly-0.1-0.3	0.331	0.984	0.495	0.012 ms	0.005 ms
OCSVM-SGD-ad.-poly-0.2-0.5	0.334	0.969	0.497	0.010 ms	0.234 ms
OCSVM-SGD-ad.-rbf-0.1-0.1	0.660	0.402	0.500	0.008 ms	0.145 ms
OCSVM-sigm.-0.3-0.2	0.337	0.977	0.501	0.014 ms	0.005 ms
OCSVM-SGD-poly-0.4-0.6	0.340	0.984	0.506	0.011 ms	0.005 ms
OCSVM-SGD-ad.-sigm.-0.8-0.4	0.901	0.355	0.510	0.007 ms	0.091 ms
OCSVM-SGD-lin.-0.1-0.3	0.350	0.984	0.516	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.2-0.3	0.350	0.984	0.516	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.3-0.3	0.350	0.984	0.516	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.4-0.3	0.350	0.984	0.516	0.007 ms	0.004 ms
OCSVM-SGD-lin.-0.6-0.3	0.350	0.984	0.516	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.8-0.3	0.350	0.984	0.516	0.007 ms	0.003 ms
OCSVM-SGD-ad.-poly-0.1-0.4	0.353	0.965	0.517	0.010 ms	0.222 ms
OCSVM-SGD-ad.-poly-0.8-0.1	0.948	0.355	0.517	0.010 ms	0.093 ms
OCSVM-SGD-ad.-sigm.-0.4-0.6	0.366	0.895	0.519	0.007 ms	0.194 ms
OCSVM-SGD-ad.-poly-0.4-0.8	0.355	0.969	0.519	0.010 ms	0.221 ms
OCSVM-SGD-rbf-0.8-0.2	0.352	1.000	0.520	0.011 ms	0.005 ms
OCSVM-SGD-ad.-poly-0.3-0.6	0.356	0.969	0.520	0.010 ms	0.220 ms
OCSVM-SGD-sigm.-0.6-0.4	0.376	0.867	0.524	0.010 ms	0.004 ms
OCSVM-SGD-ad.-poly-0.6-0.1	0.779	0.398	0.527	0.010 ms	0.100 ms
OCSVM-SGD-poly-0.6-0.9	0.362	0.984	0.529	0.011 ms	0.005 ms
OCSVM-SGD-ad.-lin.-0.1-0.4	0.364	0.965	0.529	0.006 ms	0.184 ms
OCSVM-SGD-ad.-lin.-0.2-0.4	0.364	0.965	0.529	0.006 ms	0.185 ms
OCSVM-SGD-ad.-lin.-0.3-0.4	0.364	0.965	0.529	0.006 ms	0.183 ms
OCSVM-SGD-ad.-lin.-0.4-0.4	0.364	0.965	0.529	0.006 ms	0.183 ms
OCSVM-SGD-ad.-lin.-0.6-0.4	0.364	0.965	0.529	0.006 ms	0.183 ms
OCSVM-SGD-ad.-lin.-0.8-0.4	0.364	0.965	0.529	0.006 ms	0.183 ms
OCSVM-SGD-rbf-0.6-0.2	0.369	1.000	0.539	0.011 ms	0.005 ms
OCSVM-SGD-sigm.-0.1-0.3	0.373	0.973	0.540	0.009 ms	0.004 ms
OCSVM-SGD-rbf-0.1-0.3	0.370	1.000	0.540	0.011 ms	0.005 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
OCSVM-SGD-sigm.-0.8-0.1	0.970	0.379	0.545	0.010 ms	0.004 ms
OCSVM-SGD-sigm.-0.2-0.3	0.385	0.957	0.549	0.009 ms	0.004 ms
OCSVM-SGD-ad.-poly-0.4-0.7	0.386	0.957	0.551	0.010 ms	0.203 ms
OCSVM-SGD-poly-0.3-0.4	0.384	0.984	0.552	0.011 ms	0.005 ms
OCSVM-SGD-poly-0.4-0.5	0.384	0.984	0.553	0.011 ms	0.005 ms
OCSVM-SGD-poly-0.6-0.8	0.387	0.984	0.555	0.011 ms	0.005 ms
OCSVM-SGD-ad.-sigm.-0.4-0.3	0.758	0.441	0.558	0.007 ms	0.099 ms
OCSVM-SGD-poly-0.2-0.3	0.389	0.984	0.558	0.011 ms	0.005 ms
OCSVM-SGD-ad.-sigm.-0.1-0.5	0.406	0.906	0.561	0.007 ms	0.179 ms
OCSVM-SGD-ad.-sigm.-0.2-0.5	0.407	0.902	0.561	0.007 ms	0.179 ms
OCSVM-SGD-ad.-sigm.-0.1-0.3	0.640	0.500	0.561	0.007 ms	0.109 ms
OCSVM-SGD-rbf-0.4-0.2	0.399	1.000	0.571	0.011 ms	0.005 ms
OCSVM-SGD-ad.-poly-0.3-0.5	0.411	0.941	0.572	0.010 ms	0.190 ms
OCSVM-SGD-ad.-rbf-0.4-0.1	0.644	0.516	0.573	0.008 ms	0.152 ms
OCSVM-SGD-sigm.-0.3-0.3	0.412	0.949	0.574	0.009 ms	0.004 ms
OCSVM-poly-0.8-0.1	0.405	0.988	0.575	0.005 ms	0.002 ms
OCSVM-SGD-ad.-poly-0.2-0.4	0.411	0.957	0.575	0.010 ms	0.194 ms
OCSVM-SGD-ad.-poly-0.1-0.1	0.700	0.492	0.578	0.010 ms	0.109 ms
OCSVM-poly-0.1-0.1	0.411	0.988	0.580	0.005 ms	0.002 ms
OCSVM-poly-scale-0.1	0.411	0.988	0.581	0.005 ms	0.002 ms
OCSVM-poly-0.4-0.1	0.412	0.988	0.582	0.005 ms	0.002 ms
OCSVM-poly-0.2-0.1	0.415	0.988	0.584	0.005 ms	0.002 ms
OCSVM-poly-0.6-0.1	0.415	0.988	0.584	0.005 ms	0.002 ms
OCSVM-SGD-ad.-sigm.-0.8-0.5	0.788	0.465	0.585	0.007 ms	0.097 ms
OCSVM-SGD-ad.-sigm.-0.6-0.6	0.477	0.758	0.585	0.007 ms	0.146 ms
OCSVM-SGD-sigm.-0.8-0.4	0.465	0.789	0.586	0.011 ms	0.004 ms
OCSVM-SGD-ad.-poly-0.4-0.6	0.428	0.938	0.588	0.010 ms	0.184 ms
OCSVM-SGD-poly-0.6-0.7	0.420	0.984	0.589	0.011 ms	0.005 ms
OCSVM-SGD-ad.-rbf-0.4-0.3	0.419	0.996	0.590	0.008 ms	0.233 ms
OCSVM-SGD-ad.-rbf-0.6-0.3	0.419	0.996	0.590	0.008 ms	0.234 ms
OCSVM-poly-0.3-0.1	0.425	0.988	0.595	0.005 ms	0.002 ms
OCSVM-SGD-ad.-rbf-0.1-0.4	0.423	1.000	0.595	0.008 ms	0.233 ms
OCSVM-SGD-ad.-sigm.-0.6-0.4	0.783	0.480	0.596	0.007 ms	0.098 ms
OCSVM-SGD-sigm.-0.4-0.3	0.443	0.930	0.600	0.010 ms	0.004 ms
OCSVM-SGD-ad.-sigm.-0.2-0.3	0.662	0.551	0.601	0.007 ms	0.109 ms
OCSVM-SGD-ad.-sigm.-0.8-0.6	0.592	0.613	0.603	0.007 ms	0.119 ms
OCSVM-SGD-ad.-poly-0.4-0.1	0.762	0.500	0.604	0.010 ms	0.104 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
OCSVM-SGD-rbf-0.3-0.2	0.434	1.000	0.605	0.010 ms	0.005 ms
OCSVM-SGD-ad.-sigm.-0.3-0.3	0.724	0.523	0.608	0.007 ms	0.103 ms
OCSVM-SGD-poly-0.4-0.4	0.440	0.984	0.608	0.011 ms	0.005 ms
OCSVM-SGD-ad.-rbf-0.3-0.3	0.438	1.000	0.609	0.008 ms	0.227 ms
OCSVM-SGD-ad.-sigm.-0.3-0.5	0.469	0.887	0.614	0.007 ms	0.159 ms
OCSVM-SGD-ad.-poly-0.6-0.9	0.462	0.938	0.619	0.010 ms	0.172 ms
OCSVM-SGD-poly-0.3-0.3	0.455	0.984	0.622	0.011 ms	0.005 ms
OCSVM-SGD-ad.-rbf-0.2-0.1	0.694	0.566	0.624	0.008 ms	0.149 ms
OCSVM-SGD-ad.-poly-0.2-0.1	0.719	0.551	0.624	0.010 ms	0.109 ms
OCSVM-SGD-ad.-rbf-0.8-0.3	0.457	0.988	0.625	0.008 ms	0.220 ms
OCSVM-SGD-ad.-poly-0.1-0.3	0.472	0.926	0.625	0.010 ms	0.170 ms
OCSVM-SGD-ad.-poly-0.3-0.1	0.771	0.527	0.626	0.010 ms	0.104 ms
OCSVM-SGD-poly-0.1-0.2	0.461	0.977	0.627	0.012 ms	0.005 ms
OCSVM-lin.-scale-0.1	0.461	0.980	0.627	0.007 ms	0.002 ms
OCSVM-lin.-auto-0.1	0.461	0.980	0.627	0.005 ms	0.002 ms
OCSVM-lin.-0.1-0.1	0.461	0.980	0.627	0.005 ms	0.002 ms
OCSVM-lin.-0.2-0.1	0.461	0.980	0.627	0.005 ms	0.002 ms
OCSVM-lin.-0.3-0.1	0.461	0.980	0.627	0.005 ms	0.002 ms
OCSVM-lin.-0.4-0.1	0.461	0.980	0.627	0.004 ms	0.002 ms
OCSVM-lin.-0.6-0.1	0.461	0.980	0.627	0.005 ms	0.002 ms
OCSVM-lin.-0.8-0.1	0.461	0.980	0.627	0.004 ms	0.002 ms
OCSVM-SGD-poly-0.6-0.6	0.465	0.984	0.632	0.011 ms	0.005 ms
OCSVM-SGD-poly-0.8-0.9	0.466	0.984	0.632	0.011 ms	0.005 ms
OCSVM-SGD-ad.-lin.-0.1-0.3	0.481	0.926	0.633	0.006 ms	0.144 ms
OCSVM-SGD-ad.-lin.-0.2-0.3	0.481	0.926	0.633	0.006 ms	0.144 ms
OCSVM-SGD-ad.-lin.-0.3-0.3	0.481	0.926	0.633	0.006 ms	0.143 ms
OCSVM-SGD-ad.-lin.-0.4-0.3	0.481	0.926	0.633	0.006 ms	0.143 ms
OCSVM-SGD-ad.-lin.-0.6-0.3	0.481	0.926	0.633	0.006 ms	0.143 ms
OCSVM-SGD-ad.-lin.-0.8-0.3	0.481	0.926	0.633	0.006 ms	0.143 ms
OCSVM-SGD-ad.-sigm.-0.4-0.5	0.520	0.812	0.634	0.007 ms	0.141 ms
OCSVM-SGD-ad.-sigm.-0.1-0.4	0.583	0.699	0.636	0.007 ms	0.124 ms
OCSVM-SGD-ad.-poly-0.8-0.2	0.808	0.527	0.638	0.010 ms	0.101 ms
OCSVM-sigm.-auto-0.1	0.474	0.980	0.639	0.007 ms	0.003 ms
OCSVM-SGD-rbf-0.2-0.2	0.470	1.000	0.639	0.011 ms	0.005 ms
OCSVM-SGD-ad.-poly-0.3-0.4	0.491	0.922	0.640	0.010 ms	0.162 ms
OCSVM-SGD-ad.-poly-0.6-0.8	0.497	0.910	0.643	0.010 ms	0.160 ms
OCSVM-SGD-lin.-0.1-0.2	0.481	0.980	0.645	0.007 ms	0.003 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
OCSVM-SGD-lin.-0.2-0.2	0.481	0.980	0.645	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.3-0.2	0.481	0.980	0.645	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.4-0.2	0.481	0.980	0.645	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.6-0.2	0.481	0.980	0.645	0.007 ms	0.003 ms
OCSVM-SGD-lin.-0.8-0.2	0.481	0.980	0.645	0.007 ms	0.003 ms
OCSVM-sigm.-0.1-0.1	0.482	0.980	0.646	0.008 ms	0.003 ms
OCSVM-SGD-ad.-rbf-0.3-0.1	0.683	0.613	0.646	0.008 ms	0.152 ms
OCSVM-sigm.-0.8-0.1	0.494	0.941	0.648	0.008 ms	0.003 ms
OCSVM-sigm.-0.6-0.1	0.492	0.949	0.648	0.008 ms	0.003 ms
OCSVM-SGD-ad.-sigm.-0.3-0.4	0.616	0.684	0.648	0.007 ms	0.119 ms
OCSVM-SGD-ad.-poly-0.4-0.5	0.501	0.918	0.648	0.010 ms	0.159 ms
OCSVM-SGD-ad.-sigm.-0.6-0.5	0.650	0.652	0.651	0.007 ms	0.114 ms
OCSVM-SGD-ad.-sigm.-0.2-0.4	0.585	0.738	0.653	0.007 ms	0.126 ms
OCSVM-SGD-ad.-sigm.-0.4-0.4	0.691	0.621	0.654	0.007 ms	0.109 ms
OCSVM-SGD-ad.-rbf-0.2-0.3	0.490	0.992	0.656	0.008 ms	0.207 ms
OCSVM-SGD-ad.-poly-0.2-0.3	0.516	0.906	0.657	0.010 ms	0.156 ms
OCSVM-sigm.-0.2-0.1	0.497	0.973	0.658	0.009 ms	0.003 ms
OCSVM-sigm.-0.3-0.1	0.497	0.973	0.658	0.009 ms	0.003 ms
OCSVM-sigm.-scale-0.1	0.497	0.977	0.659	0.009 ms	0.003 ms
OCSVM-sigm.-0.4-0.1	0.502	0.973	0.662	0.009 ms	0.003 ms
OCSVM-SGD-ad.-poly-0.6-0.7	0.528	0.898	0.665	0.010 ms	0.152 ms
OCSVM-SGD-ad.-poly-0.6-0.2	0.761	0.598	0.670	0.010 ms	0.106 ms
OCSVM-SGD-ad.-rbf-0.8-0.2	0.631	0.723	0.674	0.008 ms	0.163 ms
OCSVM-SGD-sigm.-0.6-0.3	0.571	0.828	0.676	0.010 ms	0.004 ms
OCSVM-SGD-sigm.-0.6-0.1	0.971	0.520	0.677	0.010 ms	0.004 ms
OCSVM-SGD-poly-0.6-0.5	0.516	0.984	0.677	0.011 ms	0.005 ms
OCSVM-SGD-poly-0.8-0.8	0.516	0.984	0.677	0.011 ms	0.005 ms
OCSVM-SGD-poly-0.2-0.2	0.520	0.980	0.679	0.011 ms	0.005 ms
OCSVM-SGD-sigm.-0.8-0.3	0.669	0.703	0.686	0.011 ms	0.004 ms
OCSVM-SGD-ad.-poly-0.8-0.3	0.776	0.621	0.690	0.010 ms	0.106 ms
OCSVM-SGD-ad.-lin.-0.1-0.1	0.761	0.633	0.691	0.006 ms	0.097 ms
OCSVM-SGD-ad.-lin.-0.2-0.1	0.761	0.633	0.691	0.006 ms	0.100 ms
OCSVM-SGD-ad.-lin.-0.3-0.1	0.761	0.633	0.691	0.006 ms	0.097 ms
OCSVM-SGD-ad.-lin.-0.4-0.1	0.761	0.633	0.691	0.006 ms	0.096 ms
OCSVM-SGD-ad.-lin.-0.6-0.1	0.761	0.633	0.691	0.006 ms	0.096 ms
OCSVM-SGD-ad.-lin.-0.8-0.1	0.761	0.633	0.691	0.006 ms	0.096 ms
OCSVM-SGD-ad.-poly-0.4-0.4	0.562	0.898	0.692	0.010 ms	0.144 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
OCSVM-SGD-poly-0.4-0.3	0.534	0.984	0.692	0.011 ms	0.005 ms
OCSVM-SGD-sigm.-0.8-0.2	0.871	0.578	0.695	0.010 ms	0.004 ms
OCSVM-SGD-poly-0.8-0.7	0.538	0.984	0.696	0.012 ms	0.005 ms
OCSVM-SGD-ad.-poly-0.3-0.3	0.588	0.891	0.708	0.010 ms	0.139 ms
OCSVM-SGD-rbf-0.1-0.2	0.551	1.000	0.710	0.011 ms	0.005 ms
OCSVM-SGD-ad.-poly-0.8-0.9	0.593	0.895	0.713	0.010 ms	0.138 ms
OCSVM-SGD-sigm.-0.1-0.2	0.580	0.930	0.715	0.010 ms	0.004 ms
OCSVM-SGD-ad.-poly-0.6-0.6	0.605	0.891	0.720	0.010 ms	0.135 ms
OCSVM-SGD-ad.-poly-0.6-0.3	0.723	0.723	0.723	0.010 ms	0.113 ms
OCSVM-SGD-ad.-rbf-0.1-0.3	0.576	0.980	0.725	0.008 ms	0.184 ms
OCSVM-SGD-sigm.-0.2-0.2	0.593	0.938	0.726	0.009 ms	0.004 ms
OCSVM-SGD-ad.-poly-0.8-0.4	0.739	0.719	0.729	0.010 ms	0.112 ms
OCSVM-SGD-ad.-poly-0.8-0.8	0.626	0.871	0.729	0.010 ms	0.131 ms
OCSVM-SGD-ad.-poly-0.4-0.2	0.728	0.730	0.729	0.010 ms	0.113 ms
OCSVM-SGD-poly-0.6-0.4	0.583	0.977	0.730	0.011 ms	0.005 ms
OCSVM-SGD-poly-0.3-0.2	0.585	0.980	0.733	0.011 ms	0.005 ms
OCSVM-SGD-ad.-poly-0.1-0.2	0.659	0.832	0.736	0.010 ms	0.125 ms
OCSVM-SGD-ad.-poly-0.3-0.2	0.697	0.781	0.737	0.010 ms	0.118 ms
OCSVM-SGD-poly-0.8-0.6	0.594	0.977	0.739	0.011 ms	0.005 ms
OCSVM-SGD-ad.-poly-0.6-0.5	0.637	0.879	0.739	0.010 ms	0.129 ms
OCSVM-SGD-ad.-poly-0.6-0.4	0.687	0.805	0.741	0.010 ms	0.120 ms
OCSVM-SGD-ad.-poly-0.8-0.7	0.664	0.848	0.744	0.010 ms	0.124 ms
OCSVM-SGD-ad.-poly-0.8-0.6	0.685	0.816	0.745	0.010 ms	0.120 ms
OCSVM-SGD-ad.-poly-0.8-0.5	0.714	0.781	0.746	0.010 ms	0.116 ms
OCSVM-SGD-sigm.-0.3-0.2	0.633	0.910	0.747	0.009 ms	0.004 ms
OCSVM-SGD-ad.-lin.-0.1-0.2	0.650	0.887	0.750	0.006 ms	0.113 ms
OCSVM-SGD-ad.-lin.-0.2-0.2	0.650	0.887	0.750	0.006 ms	0.113 ms
OCSVM-SGD-ad.-lin.-0.3-0.2	0.650	0.887	0.750	0.006 ms	0.114 ms
OCSVM-SGD-ad.-lin.-0.4-0.2	0.650	0.887	0.750	0.006 ms	0.113 ms
OCSVM-SGD-ad.-lin.-0.6-0.2	0.650	0.887	0.750	0.006 ms	0.113 ms
OCSVM-SGD-ad.-lin.-0.8-0.2	0.650	0.887	0.750	0.006 ms	0.113 ms
OCSVM-SGD-ad.-poly-0.4-0.3	0.670	0.855	0.751	0.010 ms	0.124 ms
OCSVM-SGD-ad.-rbf-0.6-0.2	0.636	0.922	0.753	0.008 ms	0.170 ms
OCSVM-SGD-sigm.-0.4-0.2	0.674	0.855	0.754	0.010 ms	0.004 ms
OCSVM-SGD-ad.-poly-0.2-0.2	0.679	0.852	0.756	0.010 ms	0.123 ms
OCSVM-SGD-sigm.-0.6-0.2	0.776	0.742	0.758	0.010 ms	0.004 ms
OCSVM-SGD-ad.-rbf-0.1-0.2	0.699	0.836	0.762	0.008 ms	0.157 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
OCSVM-SGD-ad.-rbf-0.3-0.2	0.630	0.965	0.762	0.008 ms	0.173 ms
OCSVM-SGD-ad.-rbf-0.4-0.2	0.637	0.980	0.772	0.008 ms	0.173 ms
OCSVM-SGD-poly-0.8-0.5	0.649	0.969	0.777	0.011 ms	0.005 ms
OCSVM-SGD-sigm.-0.4-0.1	0.907	0.688	0.782	0.009 ms	0.004 ms
OCSVM-SGD-ad.-rbf-0.2-0.2	0.664	0.965	0.787	0.008 ms	0.169 ms
OCSVM-SGD-poly-0.4-0.2	0.670	0.969	0.792	0.011 ms	0.005 ms
OCSVM-SGD-sigm.-0.1-0.1	0.857	0.746	0.797	0.010 ms	0.004 ms
OCSVM-SGD-rbf-0.6-0.1	0.663	1.000	0.798	0.010 ms	0.005 ms
OCSVM-SGD-rbf-0.8-0.1	0.678	0.988	0.804	0.011 ms	0.005 ms
OCSVM-SGD-rbf-0.3-0.1	0.674	1.000	0.805	0.010 ms	0.005 ms
OCSVM-SGD-rbf-0.4-0.1	0.677	1.000	0.808	0.010 ms	0.005 ms
OCSVM-SGD-sigm.-0.3-0.1	0.873	0.754	0.809	0.009 ms	0.004 ms
OCSVM-SGD-poly-0.6-0.3	0.698	0.977	0.814	0.011 ms	0.005 ms
OCSVM-SGD-poly-0.8-0.4	0.729	0.957	0.828	0.011 ms	0.005 ms
OCSVM-SGD-sigm.-0.2-0.1	0.860	0.812	0.835	0.009 ms	0.004 ms
OCSVM-SGD-poly-0.1-0.1	0.763	0.945	0.845	0.012 ms	0.005 ms
OCSVM-SGD-lin.-0.1-0.1	0.756	0.957	0.845	0.050 ms	0.004 ms
OCSVM-SGD-lin.-0.2-0.1	0.756	0.957	0.845	0.007 ms	0.004 ms
OCSVM-SGD-lin.-0.3-0.1	0.756	0.957	0.845	0.007 ms	0.004 ms
OCSVM-SGD-lin.-0.4-0.1	0.756	0.957	0.845	0.007 ms	0.004 ms
OCSVM-SGD-lin.-0.6-0.1	0.756	0.957	0.845	0.008 ms	0.004 ms
OCSVM-SGD-lin.-0.8-0.1	0.756	0.957	0.845	0.007 ms	0.004 ms
OCSVM-SGD-poly-0.8-0.3	0.775	0.941	0.850	0.011 ms	0.005 ms
OCSVM-SGD-poly-0.8-0.1	0.935	0.781	0.851	0.011 ms	0.005 ms
OCSVM-SGD-rbf-0.2-0.1	0.751	1.000	0.858	0.010 ms	0.005 ms
OCSVM-SGD-poly-0.2-0.1	0.778	0.961	0.860	0.012 ms	0.005 ms
OCSVM-SGD-poly-0.3-0.1	0.795	0.941	0.862	0.011 ms	0.005 ms
OCSVM-SGD-poly-0.6-0.2	0.798	0.941	0.864	0.011 ms	0.005 ms
OCSVM-SGD-poly-0.4-0.1	0.834	0.922	0.876	0.012 ms	0.005 ms
OCSVM-SGD-poly-0.8-0.2	0.839	0.918	0.877	0.012 ms	0.005 ms
OCSVM-SGD-poly-0.6-0.1	0.887	0.887	0.887	0.011 ms	0.005 ms
OCSVM-SGD-rbf-0.1-0.1	0.848	1.000	0.918	0.011 ms	0.005 ms

Table A.2: Performance of different novelty detection methods and hyper-parameter combinations for detecting anomalous logins. t_{train} and $t_{predict}$ are *per data point*.

A.2 ANOMALOUS LOG MESSAGES DETECTOR

This Section contains results obtained during hyper-parameter optimisation for various models for detection of anomalous log messages:

- DeepLog-{h}-{L}-{α}-{g}
- PCA-{k}-{β}
- Invariants Mining-{s}-{ϵ}
- LogCluster-{θ}-{γ}

Table A.3 contains the number of true/false positives/negatives; Table A.4 contains the derived metrics and efficiency.

METHOD	TP	FP	FP	TN
PCA-6-0.000010	4976	54575	935	0
PCA-6-0.000100	4976	54321	1189	0
PCA-6-0.000500	4976	14916	40594	0
PCA-6-0.001000	4976	12811	42699	0
LogCluster-0.4-0.1	4976	10861	44649	0
PCA-18-0.000010	4976	9747	45763	0
PCA-16-0.000010	4976	9483	46027	0
PCA-17-0.000010	4976	9466	46044	0
PCA-15-0.000010	4976	9199	46311	0
PCA-18-0.000100	4976	9181	46329	0
PCA-6-0.003000	4976	9098	46412	0
PCA-17-0.000100	4976	9058	46452	0
PCA-18-0.000500	4976	8850	46660	0
PCA-16-0.000100	4976	8818	46692	0
PCA-15-0.000100	4976	8751	46759	0
PCA-18-0.001000	4976	8676	46834	0
PCA-17-0.000500	4976	8604	46906	0
PCA-6-0.005000	4976	8423	47087	0
PCA-17-0.001000	4976	8396	47114	0
PCA-16-0.000500	4976	8376	47134	0
PCA-10-0.000010	4976	8219	47291	0
PCA-18-0.003000	4976	8172	47338	0
Invariants Mining-0.96-0.2	4976	7877	47633	0
Invariants Mining-0.96-0.9	4976	7877	47633	0
Invariants Mining-0.96-0.4	4976	7877	47633	0
Invariants Mining-0.96-0.3	4976	7877	47633	0

METHOD	TP	FP	TN	FN
Invariants Mining-0.96-0.5	4976	7877	47633	0
Invariants Mining-0.96-0.7	4976	7877	47633	0
Invariants Mining-0.96-0.6	4976	7877	47633	0
Invariants Mining-0.96-0.8	4976	7877	47633	0
Invariants Mining-0.96-0.1	4976	7860	47650	0
PCA-16-0.001000	4976	7840	47670	0
PCA-6-0.010000	4976	7820	47690	0
PCA-18-0.005000	4976	7801	47709	0
PCA-15-0.000500	4976	7626	47884	0
PCA-17-0.003000	4976	7612	47898	0
PCA-18-0.010000	4976	7411	48099	0
PCA-15-0.001000	4976	7397	48113	0
PCA-17-0.005000	4976	7370	48140	0
PCA-16-0.003000	4976	7364	48146	0
PCA-16-0.005000	4976	7216	48294	0
PCA-17-0.010000	4976	7192	48318	0
PCA-10-0.000100	4976	7101	48409	0
PCA-18-0.050000	4976	6875	48635	0
PCA-15-0.003000	4976	6833	48677	0
PCA-16-0.010000	4976	6710	48800	0
PCA-15-0.005000	4976	6665	48845	0
PCA-18-0.080000	4976	6483	49027	0
PCA-10-0.000500	4976	6444	49066	0
PCA-17-0.050000	4976	6320	49190	0
PCA-15-0.010000	4976	6297	49213	0
PCA-10-0.001000	4976	6159	49351	0
PCA-6-0.050000	4976	6070	49440	0
PCA-17-0.080000	4976	6044	49466	0
PCA-16-0.050000	4976	5933	49577	0
PCA-6-0.080000	4976	5760	49750	0
PCA-16-0.080000	4976	5681	49829	0
PCA-15-0.050000	4976	5431	50079	0
PCA-10-0.003000	4976	5429	50081	0
PCA-10-0.005000	4976	5305	50205	0
Invariants Mining-0.97-0.5	4976	5216	50294	0
Invariants Mining-0.97-0.3	4976	5216	50294	0
Invariants Mining-0.97-0.9	4976	5216	50294	0

METHOD	TP	FP	TN	FN
Invariants Mining-0.97-0.1	4976	5216	50294	0
Invariants Mining-0.98-0.1	4976	5216	50294	0
Invariants Mining-0.97-0.8	4976	5216	50294	0
Invariants Mining-0.97-0.4	4976	5216	50294	0
Invariants Mining-0.97-0.2	4976	5216	50294	0
Invariants Mining-0.98-0.7	4976	5216	50294	0
Invariants Mining-0.98-0.5	4976	5216	50294	0
Invariants Mining-0.97-0.6	4976	5216	50294	0
Invariants Mining-0.98-0.4	4976	5216	50294	0
Invariants Mining-0.97-0.7	4976	5216	50294	0
Invariants Mining-0.98-0.3	4976	5216	50294	0
Invariants Mining-0.98-0.8	4976	5216	50294	0
Invariants Mining-0.98-0.9	4976	5216	50294	0
Invariants Mining-0.98-0.2	4976	5216	50294	0
Invariants Mining-0.98-0.6	4976	5216	50294	0
PCA-10-0.010000	4976	5078	50432	0
PCA-15-0.080000	4976	4743	50767	0
PCA-10-0.050000	4976	4043	51467	0
PCA-10-0.080000	4976	3876	51634	0
Invariants Mining-0.99-0.7	4976	3499	52011	0
Invariants Mining-0.99-0.6	4976	3499	52011	0
Invariants Mining-0.99-0.8	4976	3499	52011	0
Invariants Mining-0.99-0.4	4976	3499	52011	0
Invariants Mining-0.99-0.9	4976	3499	52011	0
Invariants Mining-0.99-0.5	4976	3499	52011	0
Invariants Mining-0.99-0.3	4976	3259	52251	0
Invariants Mining-0.99-0.1	4976	3259	52251	0
Invariants Mining-0.99-0.2	4976	3259	52251	0
PCA-5-0.080000	4973	2814	52696	3
PCA-5-0.050000	4973	2698	52812	3
PCA-4-0.080000	4973	2533	52977	3
PCA-12-0.080000	4976	2051	53459	0
LogCluster-0.3-0.1	4976	1932	53578	0
PCA-12-0.050000	4976	1906	53604	0
PCA-13-0.080000	4976	1871	53639	0
PCA-13-0.050000	4976	1786	53724	0
PCA-11-0.080000	4976	1767	53743	0

METHOD	TP	FP	TN	FN
PCA-11-0.050000	4976	1640	53870	0
PCA-12-0.010000	4976	1592	53918	0
PCA-13-0.010000	4976	1539	53971	0
PCA-12-0.005000	4976	1526	53984	0
PCA-12-0.003000	4976	1492	54018	0
PCA-14-0.080000	4976	1475	54035	0
PCA-13-0.005000	4976	1408	54102	0
PCA-14-0.050000	4976	1400	54110	0
PCA-11-0.010000	4976	1382	54128	0
PCA-12-0.001000	4976	1336	54174	0
PCA-11-0.005000	4976	1316	54194	0
PCA-13-0.003000	4976	1299	54211	0
PCA-12-0.000500	4976	1277	54233	0
PCA-11-0.003000	4976	1257	54253	0
PCA-13-0.001000	4976	1214	54296	0
PCA-12-0.000100	4976	1184	54326	0
PCA-11-0.001000	4976	1183	54327	0
PCA-14-0.010000	4976	1168	54342	0
PCA-9-0.080000	4976	1150	54360	0
PCA-11-0.000500	4976	1140	54370	0
PCA-13-0.000500	4976	1139	54371	0
PCA-14-0.005000	4976	1062	54448	0
PCA-12-0.000010	4976	1058	54452	0
PCA-9-0.050000	4976	1042	54468	0
PCA-14-0.003000	4976	1032	54478	0
PCA-4-0.050000	4973	1017	54493	3
PCA-13-0.000100	4976	1009	54501	0
PCA-11-0.000100	4976	1008	54502	0
PCA-14-0.001000	4976	964	54546	0
PCA-8-0.080000	4976	960	54550	0
PCA-11-0.000010	4976	929	54581	0
PCA-14-0.000500	4976	924	54586	0
PCA-5-0.010000	4973	909	54601	3
PCA-13-0.000010	4976	891	54619	0
PCA-9-0.010000	4976	880	54630	0
PCA-8-0.050000	4976	868	54642	0
PCA-7-0.080000	4976	842	54668	0

METHOD	TP	FP	TN	FN
LogCluster-0.4-0.2	4976	840	54670	0
PCA-9-0.005000	4976	829	54681	0
PCA-9-0.003000	4976	813	54697	0
PCA-7-0.050000	4976	777	54733	0
PCA-14-0.000100	4976	769	54741	0
PCA-5-0.005000	4973	747	54763	3
PCA-9-0.001000	4976	737	54773	0
PCA-8-0.010000	4976	734	54776	0
Invariants Mining-0.995-0.6	4976	719	54791	0
Invariants Mining-0.995-0.7	4976	719	54791	0
Invariants Mining-0.995-0.8	4976	719	54791	0
Invariants Mining-0.995-0.9	4976	719	54791	0
Invariants Mining-0.995-0.5	4976	719	54791	0
PCA-14-0.000010	4976	701	54809	0
PCA-4-0.010000	4973	684	54826	3
PCA-5-0.003000	4973	677	54833	3
PCA-9-0.000500	4976	667	54843	0
LogCluster-0.2-0.1	4976	666	54844	0
PCA-8-0.005000	4976	662	54848	0
Invariants Mining-0.995-0.3	4976	646	54864	0
Invariants Mining-0.995-0.4	4976	646	54864	0
PCA-4-0.005000	4973	642	54868	3
PCA-4-0.003000	4973	608	54902	3
PCA-5-0.001000	4973	594	54916	3
PCA-8-0.003000	4976	577	54933	0
PCA-7-0.010000	4976	549	54961	0
PCA-5-0.000500	4973	545	54965	3
PCA-4-0.001000	4972	542	54968	4
PCA-4-0.000500	4972	506	55004	4
PCA-9-0.000100	4976	505	55005	0
PCA-7-0.005000	4975	502	55008	1
PCA-8-0.001000	4975	484	55026	1
PCA-1-0.005000	4807	292	55218	169
PCA-0-0.005000	4807	292	55218	169
PCA-0-0.010000	4824	298	55212	152
PCA-1-0.010000	4824	298	55212	152
PCA-5-0.000100	4972	457	55053	4

METHOD	TP	FP	TN	FN
PCA-7-0.003000	4975	459	55051	1
PCA-4-0.000100	4970	438	55072	6
PCA-0-0.080000	4872	329	55181	104
PCA-1-0.080000	4872	329	55181	104
PCA-1-0.050000	4865	320	55190	111
PCA-0-0.050000	4865	320	55190	111
PCA-8-0.000500	4975	431	55079	1
LogCluster-0.3-0.2	4976	425	55085	0
PCA-7-0.001000	4975	416	55094	1
PCA-9-0.000010	4975	410	55100	1
PCA-7-0.000500	4975	405	55105	1
PCA-1-0.000010	4695	78	55432	281
PCA-0-0.000010	4695	78	55432	281
PCA-8-0.000100	4975	379	55131	1
Invariants Mining-0.995-0.2	4976	378	55132	0
Invariants Mining-0.995-0.1	4976	378	55132	0
PCA-4-0.000010	4970	370	55140	6
PCA-5-0.000010	4970	361	55149	6
PCA-7-0.000100	4975	362	55148	1
PCA-1-0.000100	4733	89	55421	243
PCA-0-0.000100	4733	89	55421	243
PCA-1-0.003000	4795	153	55357	181
PCA-0-0.003000	4795	153	55357	181
LogCluster-0.1-0.1	4976	332	55178	0
PCA-1-0.000500	4760	97	55413	216
PCA-0-0.000500	4760	97	55413	216
PCA-7-0.000010	4975	325	55185	1
PCA-8-0.000010	4975	323	55187	1
PCA-1-0.001000	4779	105	55405	197
PCA-0-0.001000	4779	105	55405	197
PCA-3-0.080000	4961	285	55225	15
PCA-2-0.080000	4949	257	55253	27
PCA-3-0.050000	4958	259	55251	18
PCA-2-0.050000	4946	240	55270	30
PCA-2-0.010000	4938	212	55298	38
PCA-2-0.003000	4927	194	55316	49
PCA-2-0.005000	4933	200	55310	43

METHOD	TP	FP	TN	FN
PCA-2-0.001000	4920	185	55325	56
PCA-2-0.000500	4918	165	55345	58
PCA-2-0.000010	4901	142	55368	75
PCA-2-0.000100	4910	149	55361	66
PCA-3-0.010000	4951	180	55330	25
PCA-3-0.005000	4948	163	55347	28
PCA-3-0.003000	4946	157	55353	30
PCA-3-0.001000	4941	146	55364	35
PCA-3-0.000500	4938	141	55369	38
Invariants Mining-0.999-0.1	4976	180	55330	0
Invariants Mining-0.999-0.2	4976	180	55330	0
Invariants Mining-0.999-0.5	4976	180	55330	0
Invariants Mining-0.999-0.3	4976	180	55330	0
Invariants Mining-0.999-0.4	4976	180	55330	0
Invariants Mining-0.999-0.6	4976	180	55330	0
Invariants Mining-0.999-0.8	4976	180	55330	0
Invariants Mining-0.999-0.9	4976	180	55330	0
Invariants Mining-0.999-0.7	4976	180	55330	0
PCA-3-0.000100	4931	129	55381	45
LogCluster-0.4-0.3	4976	169	55341	0
PCA-3-0.000010	4919	110	55400	57
LogCluster-0.2-0.2	4976	138	55372	0
LogCluster-0.1-0.2	4976	103	55407	0
LogCluster-0.3-0.3	4976	83	55427	0
LogCluster-0.4-0.4	4976	59	55451	0
Invariants Mining-1.0-0.7	4976	57	55453	0
Invariants Mining-1.0-0.1	4976	57	55453	0
Invariants Mining-1.0-0.2	4976	57	55453	0
Invariants Mining-1.0-0.4	4976	57	55453	0
Invariants Mining-1.0-0.8	4976	57	55453	0
Invariants Mining-1.0-0.6	4976	57	55453	0
Invariants Mining-1.0-0.3	4976	57	55453	0
Invariants Mining-1.0-0.5	4976	57	55453	0
Invariants Mining-1.0-0.9	4976	57	55453	0
LogCluster-0.2-0.3	4976	54	55456	0
LogCluster-0.1-0.3	4976	52	55458	0
LogCluster-0.3-0.4	4976	50	55460	0

METHOD	TP	FP	TN	FN
LogCluster-0.1-0.4	4976	42	55468	0
LogCluster-0.2-0.4	4976	42	55468	0

Table A.3: Number of true/false positive/negative predictions on log sequences of the novelty detection methods and hyper-parameter combinations under test. The dataset used for this evaluation contains 256 positive and 2499 negative examples. The number of true/false positives/negatives of DeepLog is not available.

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-12-3-128-3	0.071	1.000	0.133	188.157 ms	1.255 ms
DeepLog-12-2-64-3	0.071	1.000	0.133	137.153 ms	0.980 ms
DeepLog-12-5-64-3	0.071	1.000	0.133	180.319 ms	1.558 ms
DeepLog-12-1-256-3	0.071	1.000	0.133	165.648 ms	0.770 ms
DeepLog-12-4-32-3	0.071	1.000	0.133	149.117 ms	1.436 ms
DeepLog-10-1-256-3	0.072	1.000	0.134	160.003 ms	0.781 ms
DeepLog-12-4-64-7	0.072	1.000	0.134	166.874 ms	1.899 ms
DeepLog-12-1-128-4	0.072	1.000	0.134	133.948 ms	0.959 ms
DeepLog-12-1-32-4	0.072	1.000	0.134	116.737 ms	0.902 ms
DeepLog-12-4-64-5	0.072	1.000	0.134	166.874 ms	1.515 ms
DeepLog-11-3-128-5	0.072	1.000	0.134	185.024 ms	1.266 ms
DeepLog-11-1-32-3	0.072	1.000	0.134	113.983 ms	0.845 ms
DeepLog-11-4-64-3	0.072	1.000	0.134	170.506 ms	1.487 ms
DeepLog-12-1-32-5	0.072	1.000	0.134	116.737 ms	0.920 ms
DeepLog-11-4-32-5	0.072	1.000	0.134	151.351 ms	1.805 ms
DeepLog-11-1-256-3	0.072	1.000	0.134	163.117 ms	0.871 ms
DeepLog-10-4-64-3	0.072	1.000	0.134	177.515 ms	1.320 ms
DeepLog-11-3-128-4	0.072	1.000	0.134	185.024 ms	1.253 ms
DeepLog-10-1-32-3	0.072	1.000	0.134	115.760 ms	0.919 ms
DeepLog-12-4-64-4	0.072	1.000	0.134	166.874 ms	1.432 ms
DeepLog-12-1-64-3	0.072	1.000	0.134	120.618 ms	0.780 ms
DeepLog-11-2-64-4	0.072	1.000	0.134	135.333 ms	1.123 ms
DeepLog-9-4-32-3	0.072	1.000	0.134	159.380 ms	1.378 ms
DeepLog-10-3-32-3	0.072	1.000	0.134	136.619 ms	1.220 ms
DeepLog-9-1-256-3	0.072	1.000	0.134	159.246 ms	0.786 ms
DeepLog-10-2-64-4	0.072	1.000	0.134	138.626 ms	1.040 ms
DeepLog-12-2-128-6	0.072	1.000	0.134	160.307 ms	1.155 ms
DeepLog-11-1-64-3	0.072	1.000	0.134	120.211 ms	0.810 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-12-5-32-3	0.072	1.000	0.134	161.609 ms	1.593 ms
DeepLog-11-4-32-4	0.072	1.000	0.134	151.351 ms	1.526 ms
DeepLog-10-2-32-3	0.072	1.000	0.134	126.327 ms	0.993 ms
DeepLog-12-1-256-4	0.072	1.000	0.134	165.648 ms	0.819 ms
DeepLog-12-3-32-5	0.072	1.000	0.134	137.887 ms	4.209 ms
DeepLog-11-3-32-4	0.072	1.000	0.134	137.162 ms	1.347 ms
DeepLog-12-3-32-6	0.072	1.000	0.134	137.887 ms	4.282 ms
DeepLog-12-1-32-6	0.072	1.000	0.134	116.737 ms	0.984 ms
DeepLog-12-1-256-7	0.072	1.000	0.134	165.648 ms	1.056 ms
DeepLog-11-5-32-3	0.072	1.000	0.134	163.355 ms	1.629 ms
DeepLog-12-1-256-5	0.072	1.000	0.134	165.648 ms	0.865 ms
DeepLog-11-3-128-3	0.072	1.000	0.134	185.024 ms	1.151 ms
DeepLog-12-3-32-4	0.072	1.000	0.134	137.887 ms	1.434 ms
DeepLog-11-2-32-3	0.072	1.000	0.134	125.380 ms	1.073 ms
DeepLog-12-3-128-5	0.072	1.000	0.134	188.157 ms	1.327 ms
DeepLog-11-4-64-4	0.072	1.000	0.134	170.506 ms	1.668 ms
DeepLog-12-2-32-4	0.072	1.000	0.134	128.497 ms	1.036 ms
DeepLog-12-3-128-4	0.072	1.000	0.134	188.157 ms	1.317 ms
DeepLog-12-3-32-3	0.072	1.000	0.134	137.887 ms	1.212 ms
DeepLog-12-4-128-5	0.072	1.000	0.134	212.681 ms	1.458 ms
DeepLog-11-5-32-4	0.072	1.000	0.134	163.355 ms	1.774 ms
DeepLog-12-4-128-4	0.072	1.000	0.134	212.681 ms	1.432 ms
DeepLog-12-5-32-4	0.072	1.000	0.134	161.609 ms	1.597 ms
DeepLog-12-4-32-4	0.072	1.000	0.134	149.117 ms	1.695 ms
DeepLog-10-2-128-3	0.072	1.000	0.134	157.738 ms	0.996 ms
DeepLog-12-2-128-5	0.072	1.000	0.134	160.307 ms	1.102 ms
DeepLog-10-4-32-3	0.072	1.000	0.134	152.722 ms	1.330 ms
DeepLog-12-2-128-3	0.072	1.000	0.134	160.307 ms	1.015 ms
DeepLog-12-5-32-7	0.072	1.000	0.134	161.609 ms	1.894 ms
DeepLog-12-2-64-4	0.072	1.000	0.134	137.153 ms	1.036 ms
DeepLog-11-3-32-3	0.072	1.000	0.134	137.162 ms	1.341 ms
DeepLog-12-4-128-6	0.072	1.000	0.134	212.681 ms	1.493 ms
DeepLog-12-1-128-3	0.072	1.000	0.134	133.948 ms	0.786 ms
DeepLog-10-4-64-4	0.072	1.000	0.134	177.515 ms	1.538 ms
DeepLog-12-2-32-3	0.072	1.000	0.134	128.497 ms	0.973 ms
DeepLog-11-1-64-5	0.072	1.000	0.134	120.211 ms	1.001 ms
DeepLog-10-2-64-3	0.072	1.000	0.134	138.626 ms	1.022 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-12-4-64-3	0.072	1.000	0.134	166.874 ms	1.412 ms
DeepLog-12-4-64-8	0.072	1.000	0.134	166.874 ms	1.949 ms
DeepLog-11-2-64-3	0.072	1.000	0.134	135.333 ms	1.024 ms
DeepLog-11-3-32-5	0.072	1.000	0.134	137.162 ms	1.367 ms
DeepLog-12-2-128-4	0.072	1.000	0.134	160.307 ms	1.047 ms
DeepLog-12-1-32-3	0.072	1.000	0.134	116.737 ms	0.776 ms
DeepLog-9-4-64-3	0.072	1.000	0.134	170.110 ms	1.389 ms
DeepLog-11-3-64-3	0.072	1.000	0.134	150.869 ms	1.310 ms
DeepLog-11-2-32-4	0.072	1.000	0.134	125.380 ms	1.087 ms
DeepLog-10-4-32-4	0.072	1.000	0.134	152.722 ms	1.398 ms
DeepLog-10-1-128-3	0.072	1.000	0.134	133.389 ms	0.763 ms
DeepLog-12-5-32-5	0.072	1.000	0.134	161.609 ms	1.619 ms
DeepLog-11-1-128-3	0.072	1.000	0.134	132.552 ms	0.833 ms
DeepLog-10-2-32-6	0.072	1.000	0.134	126.327 ms	1.103 ms
DeepLog-11-1-64-4	0.072	1.000	0.134	120.211 ms	0.950 ms
DeepLog-12-1-64-5	0.072	1.000	0.134	120.618 ms	0.924 ms
DeepLog-12-2-32-5	0.072	1.000	0.134	128.497 ms	1.086 ms
DeepLog-11-4-32-3	0.072	1.000	0.134	151.351 ms	1.406 ms
DeepLog-12-1-128-5	0.072	1.000	0.134	133.948 ms	1.024 ms
DeepLog-12-4-64-6	0.072	1.000	0.134	166.874 ms	1.805 ms
DeepLog-12-4-128-3	0.072	1.000	0.134	212.681 ms	1.368 ms
DeepLog-12-1-64-4	0.072	1.000	0.134	120.618 ms	0.801 ms
DeepLog-9-3-32-3	0.072	1.000	0.134	137.238 ms	1.102 ms
DeepLog-12-1-256-6	0.072	1.000	0.134	165.648 ms	1.027 ms
DeepLog-12-5-32-6	0.072	1.000	0.134	161.609 ms	1.639 ms
DeepLog-10-2-32-4	0.072	1.000	0.134	126.327 ms	1.027 ms
DeepLog-11-2-128-3	0.072	1.000	0.134	160.080 ms	1.106 ms
DeepLog-11-5-32-5	0.072	1.000	0.134	163.355 ms	1.827 ms
DeepLog-10-2-32-5	0.072	1.000	0.134	126.327 ms	1.098 ms
DeepLog-12-4-128-8	0.072	1.000	0.135	212.681 ms	1.593 ms
DeepLog-12-5-32-8	0.072	1.000	0.135	161.609 ms	1.916 ms
DeepLog-12-3-128-7	0.073	1.000	0.135	188.157 ms	1.438 ms
DeepLog-12-2-64-7	0.073	1.000	0.135	137.153 ms	1.349 ms
DeepLog-11-2-64-7	0.072	1.000	0.135	135.333 ms	1.310 ms
DeepLog-8-1-32-4	0.072	1.000	0.135	116.696 ms	0.844 ms
DeepLog-12-3-32-7	0.073	1.000	0.135	137.887 ms	4.702 ms
DeepLog-12-5-64-5	0.072	1.000	0.135	180.319 ms	1.605 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-10-1-32-4	0.072	1.000	0.135	115.760 ms	0.964 ms
DeepLog-11-5-32-9	0.072	1.000	0.135	163.355 ms	2.062 ms
DeepLog-12-5-64-4	0.072	1.000	0.135	180.319 ms	1.580 ms
DeepLog-12-4-128-12	0.073	1.000	0.135	212.681 ms	1.985 ms
DeepLog-11-2-128-4	0.072	1.000	0.135	160.080 ms	1.157 ms
DeepLog-11-1-64-8	0.073	1.000	0.135	120.211 ms	1.079 ms
DeepLog-11-3-128-8	0.072	1.000	0.135	185.024 ms	1.413 ms
DeepLog-9-1-256-4	0.072	1.000	0.135	159.246 ms	0.848 ms
DeepLog-7-5-32-3	0.073	1.000	0.135	171.960 ms	1.511 ms
DeepLog-11-2-64-6	0.072	1.000	0.135	135.333 ms	1.296 ms
DeepLog-8-1-256-4	0.072	1.000	0.135	166.658 ms	0.876 ms
DeepLog-8-2-64-4	0.072	1.000	0.135	137.269 ms	1.250 ms
DeepLog-10-2-128-5	0.073	1.000	0.135	157.738 ms	1.148 ms
DeepLog-11-1-256-4	0.072	1.000	0.135	163.117 ms	0.907 ms
DeepLog-12-1-128-6	0.073	1.000	0.135	133.948 ms	1.035 ms
DeepLog-11-4-32-9	0.073	1.000	0.135	151.351 ms	1.900 ms
DeepLog-9-4-32-4	0.072	1.000	0.135	159.380 ms	1.367 ms
DeepLog-10-3-64-4	0.072	1.000	0.135	150.441 ms	1.181 ms
DeepLog-9-1-64-3	0.073	1.000	0.135	128.810 ms	0.906 ms
DeepLog-12-3-64-6	0.073	1.000	0.135	150.436 ms	1.358 ms
DeepLog-9-1-128-3	0.072	1.000	0.135	132.636 ms	0.783 ms
DeepLog-10-2-32-7	0.073	1.000	0.135	126.327 ms	1.124 ms
DeepLog-11-3-64-5	0.072	1.000	0.135	150.869 ms	1.335 ms
DeepLog-12-4-32-6	0.073	1.000	0.135	149.117 ms	1.752 ms
DeepLog-8-2-128-4	0.072	1.000	0.135	169.714 ms	1.117 ms
DeepLog-11-4-32-6	0.072	1.000	0.135	151.351 ms	1.801 ms
DeepLog-8-3-32-5	0.072	1.000	0.135	137.352 ms	1.228 ms
DeepLog-11-2-32-7	0.073	1.000	0.135	125.380 ms	1.403 ms
DeepLog-8-4-32-6	0.073	1.000	0.135	161.176 ms	1.691 ms
DeepLog-10-4-64-6	0.072	1.000	0.135	177.515 ms	1.743 ms
DeepLog-12-4-128-7	0.072	1.000	0.135	212.681 ms	1.580 ms
DeepLog-8-1-64-5	0.072	1.000	0.135	120.598 ms	0.890 ms
DeepLog-11-3-64-6	0.073	1.000	0.135	150.869 ms	1.371 ms
DeepLog-12-5-64-7	0.072	1.000	0.135	180.319 ms	1.725 ms
DeepLog-11-2-64-9	0.073	1.000	0.135	135.333 ms	1.469 ms
DeepLog-10-2-64-7	0.073	1.000	0.135	138.626 ms	1.132 ms
DeepLog-12-2-32-8	0.073	1.000	0.135	128.497 ms	1.137 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-10-2-64-8	0.073	1.000	0.135	138.626 ms	1.167 ms
DeepLog-8-2-128-6	0.073	1.000	0.135	169.714 ms	1.313 ms
DeepLog-10-4-32-6	0.073	1.000	0.135	152.722 ms	1.797 ms
DeepLog-12-1-64-6	0.073	1.000	0.135	120.618 ms	0.971 ms
DeepLog-10-1-32-5	0.072	1.000	0.135	115.760 ms	0.986 ms
DeepLog-10-2-128-4	0.072	1.000	0.135	157.738 ms	1.081 ms
DeepLog-11-1-128-5	0.072	1.000	0.135	132.552 ms	0.908 ms
DeepLog-10-1-128-6	0.073	1.000	0.135	133.389 ms	0.903 ms
DeepLog-8-1-32-3	0.072	1.000	0.135	116.696 ms	0.822 ms
DeepLog-12-3-64-3	0.072	1.000	0.135	150.436 ms	1.255 ms
DeepLog-10-1-64-5	0.072	1.000	0.135	121.028 ms	0.871 ms
DeepLog-12-3-128-6	0.072	1.000	0.135	188.157 ms	1.394 ms
DeepLog-8-1-64-3	0.072	1.000	0.135	120.598 ms	0.833 ms
DeepLog-9-4-64-6	0.073	1.000	0.135	170.110 ms	4.956 ms
DeepLog-11-5-32-6	0.072	1.000	0.135	163.355 ms	1.942 ms
DeepLog-8-4-32-7	0.073	1.000	0.135	161.176 ms	1.738 ms
DeepLog-8-4-32-4	0.072	1.000	0.135	161.176 ms	1.468 ms
DeepLog-10-2-64-5	0.072	1.000	0.135	138.626 ms	1.105 ms
DeepLog-11-5-32-7	0.072	1.000	0.135	163.355 ms	2.007 ms
DeepLog-11-2-32-5	0.072	1.000	0.135	125.380 ms	1.183 ms
DeepLog-9-1-256-5	0.073	1.000	0.135	159.246 ms	0.867 ms
DeepLog-9-1-128-6	0.073	1.000	0.135	132.636 ms	1.036 ms
DeepLog-12-5-64-8	0.072	1.000	0.135	180.319 ms	1.829 ms
DeepLog-10-1-128-5	0.072	1.000	0.135	133.389 ms	0.856 ms
DeepLog-10-4-32-5	0.072	1.000	0.135	152.722 ms	1.669 ms
DeepLog-10-3-64-3	0.072	1.000	0.135	150.441 ms	1.174 ms
DeepLog-12-5-64-9	0.073	1.000	0.135	180.319 ms	1.863 ms
DeepLog-8-1-128-3	0.072	1.000	0.135	140.468 ms	0.790 ms
DeepLog-9-3-32-4	0.072	1.000	0.135	137.238 ms	1.135 ms
DeepLog-11-1-32-4	0.072	1.000	0.135	113.983 ms	0.863 ms
DeepLog-9-4-64-4	0.072	1.000	0.135	170.110 ms	1.417 ms
DeepLog-7-1-64-3	0.072	1.000	0.135	122.034 ms	0.792 ms
DeepLog-8-3-32-3	0.072	1.000	0.135	137.352 ms	1.100 ms
DeepLog-10-1-256-6	0.073	1.000	0.135	160.003 ms	0.900 ms
DeepLog-10-1-64-4	0.072	1.000	0.135	121.028 ms	0.849 ms
DeepLog-8-1-256-3	0.072	1.000	0.135	166.658 ms	0.798 ms
DeepLog-10-4-64-7	0.073	1.000	0.135	177.515 ms	1.782 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-9-1-128-5	0.072	1.000	0.135	132.636 ms	0.866 ms
DeepLog-9-2-32-3	0.072	1.000	0.135	135.313 ms	0.986 ms
DeepLog-11-1-64-9	0.073	1.000	0.135	120.211 ms	1.078 ms
DeepLog-10-3-64-5	0.073	1.000	0.135	150.441 ms	1.447 ms
DeepLog-12-2-32-7	0.073	1.000	0.135	128.497 ms	1.125 ms
DeepLog-11-4-32-7	0.072	1.000	0.135	151.351 ms	1.808 ms
DeepLog-11-3-32-6	0.072	1.000	0.135	137.162 ms	1.449 ms
DeepLog-12-3-64-4	0.072	1.000	0.135	150.436 ms	1.268 ms
DeepLog-8-2-32-3	0.072	1.000	0.135	127.653 ms	1.054 ms
DeepLog-10-1-64-3	0.072	1.000	0.135	121.028 ms	0.832 ms
DeepLog-9-2-64-4	0.072	1.000	0.135	146.191 ms	1.160 ms
DeepLog-12-2-128-7	0.073	1.000	0.135	160.307 ms	1.167 ms
DeepLog-12-4-32-7	0.073	1.000	0.135	149.117 ms	1.838 ms
DeepLog-11-3-128-7	0.072	1.000	0.135	185.024 ms	1.369 ms
DeepLog-12-3-64-7	0.073	1.000	0.135	150.436 ms	1.368 ms
DeepLog-8-4-64-4	0.072	1.000	0.135	176.597 ms	1.464 ms
DeepLog-11-5-32-8	0.072	1.000	0.135	163.355 ms	1.996 ms
DeepLog-8-2-128-3	0.072	1.000	0.135	169.714 ms	1.077 ms
DeepLog-11-4-32-8	0.073	1.000	0.135	151.351 ms	1.884 ms
DeepLog-8-1-128-4	0.072	1.000	0.135	140.468 ms	0.826 ms
DeepLog-7-2-64-3	0.073	1.000	0.135	137.791 ms	1.114 ms
DeepLog-11-2-32-6	0.073	1.000	0.135	125.380 ms	1.387 ms
DeepLog-12-2-64-6	0.073	1.000	0.135	137.153 ms	1.315 ms
DeepLog-12-3-64-5	0.073	1.000	0.135	150.436 ms	1.330 ms
DeepLog-10-2-64-6	0.073	1.000	0.135	138.626 ms	1.119 ms
DeepLog-12-4-128-10	0.072	1.000	0.135	212.681 ms	1.726 ms
DeepLog-8-2-32-7	0.073	1.000	0.135	127.653 ms	3.466 ms
DeepLog-8-4-32-3	0.072	1.000	0.135	161.176 ms	1.312 ms
DeepLog-8-4-64-3	0.072	1.000	0.135	176.597 ms	1.436 ms
DeepLog-11-5-32-11	0.072	1.000	0.135	163.355 ms	2.090 ms
DeepLog-10-3-32-5	0.073	1.000	0.135	136.619 ms	1.349 ms
DeepLog-11-3-64-4	0.072	1.000	0.135	150.869 ms	1.318 ms
DeepLog-11-2-64-5	0.072	1.000	0.135	135.333 ms	1.222 ms
DeepLog-12-2-64-5	0.072	1.000	0.135	137.153 ms	1.091 ms
DeepLog-12-4-128-11	0.072	1.000	0.135	212.681 ms	1.746 ms
DeepLog-9-3-32-5	0.072	1.000	0.135	137.238 ms	1.192 ms
DeepLog-10-3-32-4	0.072	1.000	0.135	136.619 ms	1.283 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-11-1-32-5	0.072	1.000	0.135	113.983 ms	1.038 ms
DeepLog-11-1-64-7	0.072	1.000	0.135	120.211 ms	1.064 ms
DeepLog-8-3-64-3	0.072	1.000	0.135	151.216 ms	1.138 ms
DeepLog-11-1-128-6	0.072	1.000	0.135	132.552 ms	0.913 ms
DeepLog-11-5-32-10	0.072	1.000	0.135	163.355 ms	2.084 ms
DeepLog-11-3-128-9	0.073	1.000	0.135	185.024 ms	1.474 ms
DeepLog-8-3-64-4	0.072	1.000	0.135	151.216 ms	1.157 ms
DeepLog-8-4-32-5	0.073	1.000	0.135	161.176 ms	1.669 ms
DeepLog-10-4-64-5	0.072	1.000	0.135	177.515 ms	1.645 ms
DeepLog-11-3-128-6	0.072	1.000	0.135	185.024 ms	1.279 ms
DeepLog-10-1-256-4	0.072	1.000	0.135	160.003 ms	0.831 ms
DeepLog-8-1-64-6	0.072	1.000	0.135	120.598 ms	1.038 ms
DeepLog-8-3-32-4	0.072	1.000	0.135	137.352 ms	1.136 ms
DeepLog-7-1-64-4	0.072	1.000	0.135	122.034 ms	0.839 ms
DeepLog-11-2-128-5	0.072	1.000	0.135	160.080 ms	1.251 ms
DeepLog-7-1-32-4	0.073	1.000	0.135	117.337 ms	0.888 ms
DeepLog-9-2-32-4	0.073	1.000	0.135	135.313 ms	1.033 ms
DeepLog-10-1-256-5	0.073	1.000	0.135	160.003 ms	0.869 ms
DeepLog-8-3-64-5	0.072	1.000	0.135	151.216 ms	1.321 ms
DeepLog-9-4-32-6	0.073	1.000	0.135	159.380 ms	1.447 ms
DeepLog-11-1-128-7	0.073	1.000	0.135	132.552 ms	1.048 ms
DeepLog-9-4-32-5	0.072	1.000	0.135	159.380 ms	1.390 ms
DeepLog-8-2-32-5	0.072	1.000	0.135	127.653 ms	3.447 ms
DeepLog-12-4-32-5	0.072	1.000	0.135	149.117 ms	1.759 ms
DeepLog-11-2-64-8	0.073	1.000	0.135	135.333 ms	1.462 ms
DeepLog-12-1-32-7	0.072	1.000	0.135	116.737 ms	1.022 ms
DeepLog-9-2-32-5	0.073	1.000	0.135	135.313 ms	1.198 ms
DeepLog-8-2-64-3	0.072	1.000	0.135	137.269 ms	0.969 ms
DeepLog-12-2-32-6	0.073	1.000	0.135	128.497 ms	1.119 ms
DeepLog-8-2-128-5	0.072	1.000	0.135	169.714 ms	1.309 ms
DeepLog-8-1-128-5	0.072	1.000	0.135	140.468 ms	1.025 ms
DeepLog-10-3-64-6	0.073	1.000	0.135	150.441 ms	1.525 ms
DeepLog-12-1-32-8	0.072	1.000	0.135	116.737 ms	1.032 ms
DeepLog-8-2-32-4	0.072	1.000	0.135	127.653 ms	1.101 ms
DeepLog-8-2-32-6	0.073	1.000	0.135	127.653 ms	3.428 ms
DeepLog-8-1-64-4	0.072	1.000	0.135	120.598 ms	0.847 ms
DeepLog-8-4-64-5	0.072	1.000	0.135	176.597 ms	1.496 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-11-1-256-5	0.072	1.000	0.135	163.117 ms	0.916 ms
DeepLog-11-1-128-4	0.072	1.000	0.135	132.552 ms	0.873 ms
DeepLog-7-1-32-3	0.072	1.000	0.135	117.337 ms	0.874 ms
DeepLog-9-1-128-4	0.072	1.000	0.135	132.636 ms	0.809 ms
DeepLog-9-4-64-5	0.072	1.000	0.135	170.110 ms	1.491 ms
DeepLog-12-4-128-9	0.072	1.000	0.135	212.681 ms	1.656 ms
DeepLog-8-1-256-5	0.073	1.000	0.135	166.658 ms	0.959 ms
DeepLog-9-2-64-3	0.072	1.000	0.135	146.191 ms	0.990 ms
DeepLog-10-1-128-4	0.072	1.000	0.135	133.389 ms	0.792 ms
DeepLog-11-1-64-6	0.072	1.000	0.135	120.211 ms	1.036 ms
DeepLog-12-5-64-6	0.072	1.000	0.135	180.319 ms	1.619 ms
DeepLog-12-3-64-13	0.073	1.000	0.136	150.436 ms	1.469 ms
DeepLog-12-1-128-9	0.073	1.000	0.136	133.948 ms	1.077 ms
DeepLog-12-3-64-9	0.073	1.000	0.136	150.436 ms	1.425 ms
DeepLog-12-3-64-10	0.073	1.000	0.136	150.436 ms	1.448 ms
DeepLog-12-4-128-16	0.073	1.000	0.136	212.681 ms	2.122 ms
DeepLog-12-2-128-9	0.073	1.000	0.136	160.307 ms	1.425 ms
DeepLog-11-5-32-15	0.073	1.000	0.136	163.355 ms	2.382 ms
DeepLog-11-4-64-6	0.073	1.000	0.136	170.506 ms	1.795 ms
DeepLog-11-5-32-16	0.073	1.000	0.136	163.355 ms	2.492 ms
DeepLog-10-4-32-8	0.073	1.000	0.136	152.722 ms	2.117 ms
DeepLog-11-5-32-14	0.073	1.000	0.136	163.355 ms	2.099 ms
DeepLog-8-2-128-7	0.073	1.000	0.136	169.714 ms	1.364 ms
DeepLog-8-2-64-9	0.073	1.000	0.136	137.269 ms	1.403 ms
DeepLog-10-4-64-10	0.073	1.000	0.136	177.515 ms	2.167 ms
DeepLog-12-3-32-13	0.073	1.000	0.136	137.887 ms	4.818 ms
DeepLog-12-3-32-11	0.073	1.000	0.136	137.887 ms	4.771 ms
DeepLog-9-3-64-4	0.073	1.000	0.136	158.282 ms	1.152 ms
DeepLog-11-2-64-14	0.073	1.000	0.136	135.333 ms	1.572 ms
DeepLog-9-2-64-5	0.073	1.000	0.136	146.191 ms	1.236 ms
DeepLog-10-4-32-9	0.073	1.000	0.136	152.722 ms	2.129 ms
DeepLog-10-1-256-8	0.073	1.000	0.136	160.003 ms	1.051 ms
DeepLog-12-4-64-9	0.073	1.000	0.136	166.874 ms	1.983 ms
DeepLog-8-2-64-7	0.073	1.000	0.136	137.269 ms	1.356 ms
DeepLog-12-1-256-9	0.073	1.000	0.136	165.648 ms	1.104 ms
DeepLog-11-2-64-13	0.073	1.000	0.136	135.333 ms	1.552 ms
DeepLog-10-3-64-9	0.073	1.000	0.136	150.441 ms	1.778 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-12-4-128-19	0.073	1.000	0.136	212.681 ms	2.209 ms
DeepLog-11-2-64-12	0.073	1.000	0.136	135.333 ms	1.532 ms
DeepLog-12-1-64-9	0.073	1.000	0.136	120.618 ms	1.094 ms
DeepLog-10-2-32-8	0.073	1.000	0.136	126.327 ms	1.366 ms
DeepLog-11-2-64-10	0.073	1.000	0.136	135.333 ms	1.534 ms
DeepLog-10-1-256-7	0.073	1.000	0.136	160.003 ms	0.929 ms
DeepLog-12-1-64-7	0.073	1.000	0.136	120.618 ms	1.010 ms
DeepLog-10-1-64-6	0.073	1.000	0.136	121.028 ms	1.136 ms
DeepLog-12-1-128-10	0.073	1.000	0.136	133.948 ms	1.100 ms
DeepLog-11-3-32-8	0.073	1.000	0.136	137.162 ms	1.615 ms
DeepLog-12-4-32-12	0.073	1.000	0.136	149.117 ms	5.701 ms
DeepLog-11-1-256-6	0.073	1.000	0.136	163.117 ms	0.936 ms
DeepLog-10-4-32-7	0.073	1.000	0.136	152.722 ms	1.999 ms
DeepLog-9-1-64-7	0.073	1.000	0.136	128.810 ms	1.038 ms
DeepLog-12-5-32-11	0.073	1.000	0.136	161.609 ms	2.161 ms
DeepLog-12-3-32-14	0.073	1.000	0.136	137.887 ms	4.696 ms
DeepLog-10-2-32-10	0.073	1.000	0.136	126.327 ms	1.571 ms
DeepLog-9-4-64-7	0.073	1.000	0.136	170.110 ms	5.032 ms
DeepLog-12-2-64-9	0.073	1.000	0.136	137.153 ms	1.372 ms
DeepLog-11-1-64-10	0.073	1.000	0.136	120.211 ms	2.721 ms
DeepLog-12-3-32-9	0.073	1.000	0.136	137.887 ms	4.692 ms
DeepLog-8-2-64-6	0.073	1.000	0.136	137.269 ms	1.333 ms
DeepLog-11-4-64-7	0.073	1.000	0.136	170.506 ms	1.820 ms
DeepLog-11-2-32-10	0.073	1.000	0.136	125.380 ms	1.512 ms
DeepLog-8-2-32-8	0.073	1.000	0.136	127.653 ms	3.537 ms
DeepLog-10-4-64-8	0.073	1.000	0.136	177.515 ms	2.006 ms
DeepLog-12-1-128-7	0.073	1.000	0.136	133.948 ms	1.062 ms
DeepLog-12-3-64-11	0.073	1.000	0.136	150.436 ms	1.452 ms
DeepLog-8-2-64-8	0.073	1.000	0.136	137.269 ms	1.376 ms
DeepLog-12-4-128-17	0.073	1.000	0.136	212.681 ms	2.209 ms
DeepLog-12-2-64-10	0.073	1.000	0.136	137.153 ms	1.413 ms
DeepLog-8-1-256-6	0.073	1.000	0.136	166.658 ms	0.961 ms
DeepLog-7-1-64-5	0.073	1.000	0.136	122.034 ms	0.983 ms
DeepLog-12-4-32-9	0.073	1.000	0.136	149.117 ms	1.997 ms
DeepLog-11-5-32-13	0.073	1.000	0.136	163.355 ms	2.132 ms
DeepLog-12-4-32-11	0.073	1.000	0.136	149.117 ms	5.680 ms
DeepLog-10-2-64-10	0.073	1.000	0.136	138.626 ms	1.202 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-10-1-256-9	0.073	1.000	0.136	160.003 ms	1.091 ms
DeepLog-12-2-32-10	0.073	1.000	0.136	128.497 ms	1.184 ms
DeepLog-9-1-64-9	0.073	1.000	0.136	128.810 ms	1.086 ms
DeepLog-12-5-32-12	0.073	1.000	0.136	161.609 ms	2.196 ms
DeepLog-9-1-64-6	0.073	1.000	0.136	128.810 ms	0.993 ms
DeepLog-9-3-64-6	0.073	1.000	0.136	158.282 ms	1.165 ms
DeepLog-9-1-64-4	0.073	1.000	0.136	128.810 ms	0.934 ms
DeepLog-11-1-128-8	0.073	1.000	0.136	132.552 ms	1.115 ms
DeepLog-11-1-32-6	0.073	1.000	0.136	113.983 ms	1.080 ms
DeepLog-12-1-128-12	0.073	1.000	0.136	133.948 ms	1.123 ms
DeepLog-12-1-32-12	0.073	1.000	0.136	116.737 ms	1.116 ms
DeepLog-10-1-32-7	0.073	1.000	0.136	115.760 ms	1.245 ms
DeepLog-10-1-128-8	0.073	1.000	0.136	133.389 ms	1.122 ms
DeepLog-12-1-128-11	0.073	1.000	0.136	133.948 ms	1.115 ms
DeepLog-12-4-32-8	0.073	1.000	0.136	149.117 ms	1.913 ms
DeepLog-12-5-32-10	0.073	1.000	0.136	161.609 ms	2.056 ms
DeepLog-12-5-64-10	0.073	1.000	0.136	180.319 ms	2.212 ms
DeepLog-12-2-128-8	0.073	1.000	0.136	160.307 ms	1.364 ms
DeepLog-12-3-32-10	0.073	1.000	0.136	137.887 ms	4.732 ms
DeepLog-12-2-64-11	0.073	1.000	0.136	137.153 ms	1.419 ms
DeepLog-11-4-64-5	0.073	1.000	0.136	170.506 ms	1.701 ms
DeepLog-11-4-32-10	0.073	1.000	0.136	151.351 ms	1.919 ms
DeepLog-12-4-128-18	0.073	1.000	0.136	212.681 ms	2.196 ms
DeepLog-12-1-64-10	0.073	1.000	0.136	120.618 ms	1.113 ms
DeepLog-12-1-128-8	0.073	1.000	0.136	133.948 ms	1.073 ms
DeepLog-10-2-64-9	0.073	1.000	0.136	138.626 ms	1.193 ms
DeepLog-10-3-32-7	0.073	1.000	0.136	136.619 ms	1.784 ms
DeepLog-7-1-64-6	0.073	1.000	0.136	122.034 ms	1.087 ms
DeepLog-10-2-128-6	0.073	1.000	0.136	157.738 ms	1.156 ms
DeepLog-12-2-64-8	0.073	1.000	0.136	137.153 ms	1.364 ms
DeepLog-12-4-128-13	0.073	1.000	0.136	212.681 ms	2.033 ms
DeepLog-12-4-32-10	0.073	1.000	0.136	149.117 ms	2.073 ms
DeepLog-10-4-64-9	0.073	1.000	0.136	177.515 ms	2.065 ms
DeepLog-12-1-32-10	0.073	1.000	0.136	116.737 ms	1.120 ms
DeepLog-11-2-32-8	0.073	1.000	0.136	125.380 ms	1.462 ms
DeepLog-12-3-32-15	0.073	1.000	0.136	137.887 ms	4.729 ms
DeepLog-9-3-32-6	0.073	1.000	0.136	137.238 ms	1.197 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-12-5-32-13	0.073	1.000	0.136	161.609 ms	2.180 ms
DeepLog-9-1-64-8	0.073	1.000	0.136	128.810 ms	1.047 ms
DeepLog-12-3-32-12	0.073	1.000	0.136	137.887 ms	4.748 ms
DeepLog-12-4-128-14	0.073	1.000	0.136	212.681 ms	2.091 ms
DeepLog-12-3-64-12	0.073	1.000	0.136	150.436 ms	1.484 ms
DeepLog-12-3-64-8	0.073	1.000	0.136	150.436 ms	1.402 ms
DeepLog-12-2-32-9	0.073	1.000	0.136	128.497 ms	1.172 ms
DeepLog-10-1-32-6	0.073	1.000	0.136	115.760 ms	1.088 ms
DeepLog-10-2-32-9	0.073	1.000	0.136	126.327 ms	1.572 ms
DeepLog-10-3-32-8	0.073	1.000	0.136	136.619 ms	1.802 ms
DeepLog-10-4-32-10	0.073	1.000	0.136	152.722 ms	2.160 ms
DeepLog-10-3-64-8	0.073	1.000	0.136	150.441 ms	1.582 ms
DeepLog-12-1-32-11	0.073	1.000	0.136	116.737 ms	1.104 ms
DeepLog-7-5-32-4	0.073	1.000	0.136	171.960 ms	1.590 ms
DeepLog-8-3-32-6	0.073	1.000	0.136	137.352 ms	1.400 ms
DeepLog-12-4-32-13	0.073	1.000	0.136	149.117 ms	5.599 ms
DeepLog-7-1-64-7	0.073	1.000	0.136	122.034 ms	1.096 ms
DeepLog-8-3-64-6	0.073	1.000	0.136	151.216 ms	1.561 ms
DeepLog-12-5-32-9	0.073	1.000	0.136	161.609 ms	1.995 ms
DeepLog-9-3-64-5	0.073	1.000	0.136	158.282 ms	1.159 ms
DeepLog-10-3-64-7	0.073	1.000	0.136	150.441 ms	1.543 ms
DeepLog-11-5-32-12	0.073	1.000	0.136	163.355 ms	2.098 ms
DeepLog-10-3-32-6	0.073	1.000	0.136	136.619 ms	1.608 ms
DeepLog-12-1-64-8	0.073	1.000	0.136	120.618 ms	1.078 ms
DeepLog-12-3-128-8	0.073	1.000	0.136	188.157 ms	1.467 ms
DeepLog-12-3-32-8	0.073	1.000	0.136	137.887 ms	4.649 ms
DeepLog-10-1-64-7	0.073	1.000	0.136	121.028 ms	1.173 ms
DeepLog-12-1-256-10	0.073	1.000	0.136	165.648 ms	1.125 ms
DeepLog-12-1-256-8	0.073	1.000	0.136	165.648 ms	1.080 ms
DeepLog-11-2-32-9	0.073	1.000	0.136	125.380 ms	1.476 ms
DeepLog-9-1-64-5	0.073	1.000	0.136	128.810 ms	0.980 ms
DeepLog-8-2-64-5	0.073	1.000	0.136	137.269 ms	1.317 ms
DeepLog-9-3-64-3	0.073	1.000	0.136	158.282 ms	1.137 ms
DeepLog-11-2-64-11	0.073	1.000	0.136	135.333 ms	1.529 ms
DeepLog-12-4-64-10	0.073	1.000	0.136	166.874 ms	2.011 ms
DeepLog-11-3-32-7	0.073	1.000	0.136	137.162 ms	1.626 ms
DeepLog-10-1-128-7	0.073	1.000	0.136	133.389 ms	1.037 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-12-1-32-9	0.073	1.000	0.136	116.737 ms	1.059 ms
DeepLog-12-4-128-15	0.073	1.000	0.136	212.681 ms	2.094 ms
DeepLog-8-4-64-6	0.073	1.000	0.136	176.597 ms	1.648 ms
DeepLog-12-5-32-19	0.074	1.000	0.137	161.609 ms	2.214 ms
DeepLog-11-5-32-18	0.073	1.000	0.137	163.355 ms	2.519 ms
DeepLog-12-2-128-15	0.074	1.000	0.137	160.307 ms	1.493 ms
DeepLog-12-5-32-20	0.074	1.000	0.137	161.609 ms	2.253 ms
DeepLog-12-2-128-13	0.074	1.000	0.137	160.307 ms	1.465 ms
DeepLog-12-5-32-15	0.073	1.000	0.137	161.609 ms	2.228 ms
DeepLog-12-1-128-15	0.074	1.000	0.137	133.948 ms	3.037 ms
DeepLog-12-2-128-14	0.074	1.000	0.137	160.307 ms	1.487 ms
DeepLog-11-3-32-14	0.074	1.000	0.137	137.162 ms	1.902 ms
DeepLog-11-2-128-6	0.073	1.000	0.137	160.080 ms	1.449 ms
DeepLog-10-2-64-14	0.073	1.000	0.137	138.626 ms	1.436 ms
DeepLog-8-4-64-9	0.074	1.000	0.137	176.597 ms	1.674 ms
DeepLog-12-5-64-17	0.074	1.000	0.137	180.319 ms	2.413 ms
DeepLog-10-2-64-15	0.074	1.000	0.137	138.626 ms	1.480 ms
DeepLog-7-4-32-3	0.074	1.000	0.137	160.527 ms	1.644 ms
DeepLog-9-4-32-7	0.074	1.000	0.137	159.380 ms	1.758 ms
DeepLog-8-3-64-7	0.073	1.000	0.137	151.216 ms	1.578 ms
DeepLog-12-1-32-17	0.074	1.000	0.137	116.737 ms	1.185 ms
DeepLog-11-1-64-12	0.074	1.000	0.137	120.211 ms	2.823 ms
DeepLog-11-1-64-13	0.074	1.000	0.137	120.211 ms	2.875 ms
DeepLog-11-2-128-7	0.074	1.000	0.137	160.080 ms	1.485 ms
DeepLog-12-3-32-19	0.074	1.000	0.137	137.887 ms	4.826 ms
DeepLog-12-4-64-13	0.074	1.000	0.137	166.874 ms	2.035 ms
DeepLog-12-1-256-11	0.073	1.000	0.137	165.648 ms	1.130 ms
DeepLog-12-1-128-13	0.074	1.000	0.137	133.948 ms	2.751 ms
DeepLog-12-1-32-13	0.074	1.000	0.137	116.737 ms	1.158 ms
DeepLog-12-3-128-9	0.073	1.000	0.137	188.157 ms	1.522 ms
DeepLog-11-4-32-12	0.074	1.000	0.137	151.351 ms	1.960 ms
DeepLog-12-4-32-17	0.073	1.000	0.137	149.117 ms	5.809 ms
DeepLog-11-5-32-17	0.073	1.000	0.137	163.355 ms	2.568 ms
DeepLog-12-2-32-12	0.073	1.000	0.137	128.497 ms	1.428 ms
DeepLog-12-3-32-18	0.074	1.000	0.137	137.887 ms	4.783 ms
DeepLog-12-1-32-16	0.074	1.000	0.137	116.737 ms	1.163 ms
DeepLog-10-4-32-13	0.073	1.000	0.137	152.722 ms	5.118 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-7-2-128-4	0.074	1.000	0.137	159.838 ms	1.259 ms
DeepLog-12-4-32-14	0.073	1.000	0.137	149.117 ms	5.612 ms
DeepLog-11-3-64-10	0.074	1.000	0.137	150.869 ms	1.523 ms
DeepLog-12-3-32-17	0.073	1.000	0.137	137.887 ms	4.771 ms
DeepLog-7-3-32-6	0.074	1.000	0.137	147.040 ms	1.460 ms
DeepLog-12-4-64-12	0.073	1.000	0.137	166.874 ms	2.028 ms
DeepLog-9-1-64-10	0.073	1.000	0.137	128.810 ms	1.086 ms
DeepLog-7-2-128-3	0.073	1.000	0.137	159.838 ms	1.078 ms
DeepLog-10-4-32-12	0.073	1.000	0.137	152.722 ms	5.163 ms
DeepLog-12-4-64-15	0.074	1.000	0.137	166.874 ms	2.043 ms
DeepLog-7-2-128-5	0.074	1.000	0.137	159.838 ms	1.286 ms
DeepLog-11-4-32-13	0.074	1.000	0.137	151.351 ms	1.971 ms
DeepLog-9-2-64-6	0.074	1.000	0.137	146.191 ms	1.253 ms
DeepLog-11-3-64-7	0.073	1.000	0.137	150.869 ms	1.413 ms
DeepLog-11-4-32-14	0.074	1.000	0.137	151.351 ms	2.001 ms
DeepLog-11-4-64-9	0.074	1.000	0.137	170.506 ms	1.880 ms
DeepLog-8-4-64-8	0.073	1.000	0.137	176.597 ms	1.654 ms
DeepLog-11-3-32-11	0.074	1.000	0.137	137.162 ms	1.834 ms
DeepLog-7-3-32-8	0.074	1.000	0.137	147.040 ms	1.547 ms
DeepLog-12-1-32-14	0.074	1.000	0.137	116.737 ms	1.162 ms
DeepLog-11-3-32-12	0.074	1.000	0.137	137.162 ms	1.863 ms
DeepLog-12-5-64-12	0.074	1.000	0.137	180.319 ms	2.319 ms
DeepLog-12-5-64-16	0.074	1.000	0.137	180.319 ms	2.370 ms
DeepLog-12-4-64-14	0.074	1.000	0.137	166.874 ms	2.055 ms
DeepLog-10-3-32-11	0.073	1.000	0.137	136.619 ms	1.908 ms
DeepLog-11-4-32-11	0.074	1.000	0.137	151.351 ms	1.941 ms
DeepLog-12-2-64-16	0.074	1.000	0.137	137.153 ms	1.446 ms
DeepLog-10-3-32-10	0.073	1.000	0.137	136.619 ms	1.870 ms
DeepLog-12-2-32-16	0.074	1.000	0.137	128.497 ms	3.981 ms
DeepLog-10-1-64-8	0.073	1.000	0.137	121.028 ms	1.185 ms
DeepLog-12-2-32-11	0.073	1.000	0.137	128.497 ms	1.410 ms
DeepLog-11-3-128-10	0.074	1.000	0.137	185.024 ms	1.542 ms
DeepLog-10-1-64-11	0.074	1.000	0.137	121.028 ms	1.254 ms
DeepLog-10-4-32-14	0.074	1.000	0.137	152.722 ms	5.235 ms
DeepLog-11-3-32-10	0.073	1.000	0.137	137.162 ms	1.776 ms
DeepLog-12-1-32-15	0.074	1.000	0.137	116.737 ms	1.153 ms
DeepLog-12-2-64-13	0.073	1.000	0.137	137.153 ms	1.443 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-11-3-32-9	0.073	1.000	0.137	137.162 ms	1.743 ms
DeepLog-12-5-64-14	0.074	1.000	0.137	180.319 ms	2.353 ms
DeepLog-9-2-64-7	0.074	1.000	0.137	146.191 ms	1.273 ms
DeepLog-12-2-64-17	0.074	1.000	0.137	137.153 ms	1.452 ms
DeepLog-10-2-64-12	0.073	1.000	0.137	138.626 ms	1.407 ms
DeepLog-12-1-64-14	0.074	1.000	0.137	120.618 ms	1.141 ms
DeepLog-12-2-64-15	0.074	1.000	0.137	137.153 ms	1.452 ms
DeepLog-12-3-128-10	0.074	1.000	0.137	188.157 ms	1.571 ms
DeepLog-10-4-32-11	0.073	1.000	0.137	152.722 ms	5.215 ms
DeepLog-10-1-256-10	0.073	1.000	0.137	160.003 ms	1.105 ms
DeepLog-11-1-64-11	0.074	1.000	0.137	120.211 ms	2.784 ms
DeepLog-7-3-32-3	0.073	1.000	0.137	147.040 ms	1.197 ms
DeepLog-10-1-256-11	0.074	1.000	0.137	160.003 ms	1.150 ms
DeepLog-10-2-64-16	0.074	1.000	0.137	138.626 ms	1.490 ms
DeepLog-7-3-32-4	0.073	1.000	0.137	147.040 ms	1.343 ms
DeepLog-12-2-32-15	0.074	1.000	0.137	128.497 ms	4.020 ms
DeepLog-11-1-256-8	0.073	1.000	0.137	163.117 ms	1.059 ms
DeepLog-12-1-64-12	0.074	1.000	0.137	120.618 ms	1.126 ms
DeepLog-12-2-32-14	0.073	1.000	0.137	128.497 ms	1.437 ms
DeepLog-12-1-64-11	0.073	1.000	0.137	120.618 ms	1.118 ms
DeepLog-12-5-32-14	0.073	1.000	0.137	161.609 ms	2.200 ms
DeepLog-11-3-64-8	0.073	1.000	0.137	150.869 ms	1.398 ms
DeepLog-10-3-32-9	0.073	1.000	0.137	136.619 ms	1.877 ms
DeepLog-8-4-64-7	0.073	1.000	0.137	176.597 ms	1.673 ms
DeepLog-11-2-128-8	0.074	1.000	0.137	160.080 ms	1.485 ms
DeepLog-12-2-128-11	0.074	1.000	0.137	160.307 ms	1.451 ms
DeepLog-12-2-64-12	0.073	1.000	0.137	137.153 ms	1.423 ms
DeepLog-11-4-64-10	0.074	1.000	0.137	170.506 ms	1.945 ms
DeepLog-7-3-32-5	0.074	1.000	0.137	147.040 ms	1.402 ms
DeepLog-12-1-64-13	0.074	1.000	0.137	120.618 ms	1.135 ms
DeepLog-11-1-256-7	0.073	1.000	0.137	163.117 ms	1.007 ms
DeepLog-12-5-64-15	0.074	1.000	0.137	180.319 ms	2.373 ms
DeepLog-12-3-128-11	0.074	1.000	0.137	188.157 ms	1.581 ms
DeepLog-12-1-256-12	0.073	1.000	0.137	165.648 ms	1.142 ms
DeepLog-12-5-32-17	0.074	1.000	0.137	161.609 ms	2.212 ms
DeepLog-9-3-64-9	0.074	1.000	0.137	158.282 ms	1.370 ms
DeepLog-11-3-32-13	0.074	1.000	0.137	137.162 ms	1.843 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-10-1-64-10	0.074	1.000	0.137	121.028 ms	1.240 ms
DeepLog-12-3-32-16	0.073	1.000	0.137	137.887 ms	4.738 ms
DeepLog-12-2-64-14	0.074	1.000	0.137	137.153 ms	1.441 ms
DeepLog-11-1-128-10	0.074	1.000	0.137	132.552 ms	1.212 ms
DeepLog-12-4-32-16	0.073	1.000	0.137	149.117 ms	5.927 ms
DeepLog-12-4-64-11	0.073	1.000	0.137	166.874 ms	2.011 ms
DeepLog-12-3-64-14	0.073	1.000	0.137	150.436 ms	1.657 ms
DeepLog-9-3-64-7	0.073	1.000	0.137	158.282 ms	1.205 ms
DeepLog-12-2-128-10	0.073	1.000	0.137	160.307 ms	1.441 ms
DeepLog-10-2-64-11	0.073	1.000	0.137	138.626 ms	1.376 ms
DeepLog-12-1-128-14	0.074	1.000	0.137	133.948 ms	3.026 ms
DeepLog-12-4-32-18	0.074	1.000	0.137	149.117 ms	5.856 ms
DeepLog-10-2-64-13	0.073	1.000	0.137	138.626 ms	1.438 ms
DeepLog-12-1-256-13	0.074	1.000	0.137	165.648 ms	2.874 ms
DeepLog-11-1-128-9	0.074	1.000	0.137	132.552 ms	1.198 ms
DeepLog-11-2-128-9	0.074	1.000	0.137	160.080 ms	1.495 ms
DeepLog-12-2-32-13	0.073	1.000	0.137	128.497 ms	1.450 ms
DeepLog-12-5-32-18	0.074	1.000	0.137	161.609 ms	2.222 ms
DeepLog-12-2-128-12	0.074	1.000	0.137	160.307 ms	1.461 ms
DeepLog-9-3-64-8	0.074	1.000	0.137	158.282 ms	1.345 ms
DeepLog-11-3-64-9	0.074	1.000	0.137	150.869 ms	1.415 ms
DeepLog-11-5-32-19	0.073	1.000	0.137	163.355 ms	2.569 ms
DeepLog-8-3-32-7	0.073	1.000	0.137	137.352 ms	1.446 ms
DeepLog-10-2-64-17	0.074	1.000	0.137	138.626 ms	1.544 ms
DeepLog-11-2-128-10	0.074	1.000	0.137	160.080 ms	1.537 ms
DeepLog-10-1-64-9	0.073	1.000	0.137	121.028 ms	1.226 ms
DeepLog-12-5-32-16	0.074	1.000	0.137	161.609 ms	2.196 ms
DeepLog-12-5-64-13	0.074	1.000	0.137	180.319 ms	2.276 ms
DeepLog-12-5-64-11	0.073	1.000	0.137	180.319 ms	2.230 ms
DeepLog-10-4-32-15	0.074	1.000	0.137	152.722 ms	5.252 ms
DeepLog-11-4-64-8	0.073	1.000	0.137	170.506 ms	1.837 ms
DeepLog-7-3-32-7	0.074	1.000	0.137	147.040 ms	1.520 ms
DeepLog-8-3-32-8	0.073	1.000	0.137	137.352 ms	1.523 ms
DeepLog-12-4-32-15	0.073	1.000	0.137	149.117 ms	5.570 ms
DeepLog-12-3-128-12	0.074	1.000	0.137	188.157 ms	1.629 ms
DeepLog-11-1-128-14	0.074	1.000	0.138	132.552 ms	1.230 ms
DeepLog-11-4-64-18	0.074	1.000	0.138	170.506 ms	2.168 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-11-5-32-20	0.074	1.000	0.138	163.355 ms	2.575 ms
DeepLog-11-1-128-13	0.074	1.000	0.138	132.552 ms	1.246 ms
DeepLog-11-1-256-9	0.074	1.000	0.138	163.117 ms	1.206 ms
DeepLog-10-1-128-10	0.074	1.000	0.138	133.389 ms	1.190 ms
DeepLog-11-3-128-14	0.074	1.000	0.138	185.024 ms	1.837 ms
DeepLog-8-3-64-8	0.074	1.000	0.138	151.216 ms	1.621 ms
DeepLog-12-2-64-20	0.074	1.000	0.138	137.153 ms	1.527 ms
DeepLog-11-2-128-13	0.074	1.000	0.138	160.080 ms	1.611 ms
DeepLog-10-4-64-11	0.074	1.000	0.138	177.515 ms	2.318 ms
DeepLog-12-2-128-20	0.074	1.000	0.138	160.307 ms	1.570 ms
DeepLog-10-2-32-12	0.074	1.000	0.138	126.327 ms	1.647 ms
DeepLog-12-4-64-16	0.074	1.000	0.138	166.874 ms	2.003 ms
DeepLog-12-4-64-18	0.074	1.000	0.138	166.874 ms	2.038 ms
DeepLog-12-3-128-14	0.074	1.000	0.138	188.157 ms	1.790 ms
DeepLog-12-2-128-17	0.074	1.000	0.138	160.307 ms	1.509 ms
DeepLog-10-1-256-12	0.074	1.000	0.138	160.003 ms	1.250 ms
DeepLog-11-1-64-18	0.074	1.000	0.138	120.211 ms	2.936 ms
DeepLog-11-4-32-18	0.074	1.000	0.138	151.351 ms	2.087 ms
DeepLog-12-1-32-18	0.074	1.000	0.138	116.737 ms	1.184 ms
DeepLog-12-3-64-18	0.074	1.000	0.138	150.436 ms	1.766 ms
DeepLog-7-4-32-7	0.074	1.000	0.138	160.527 ms	1.928 ms
DeepLog-7-2-128-6	0.074	1.000	0.138	159.838 ms	1.312 ms
DeepLog-11-1-256-10	0.074	1.000	0.138	163.117 ms	1.231 ms
DeepLog-9-2-64-8	0.074	1.000	0.138	146.191 ms	1.288 ms
DeepLog-11-4-64-13	0.074	1.000	0.138	170.506 ms	1.951 ms
DeepLog-7-4-32-6	0.074	1.000	0.138	160.527 ms	1.726 ms
DeepLog-11-4-32-17	0.074	1.000	0.138	151.351 ms	2.064 ms
DeepLog-10-1-256-14	0.074	1.000	0.138	160.003 ms	1.274 ms
DeepLog-12-2-64-18	0.074	1.000	0.138	137.153 ms	1.496 ms
DeepLog-12-1-128-17	0.074	1.000	0.138	133.948 ms	3.055 ms
DeepLog-11-4-32-16	0.074	1.000	0.138	151.351 ms	2.031 ms
DeepLog-11-3-32-15	0.074	1.000	0.138	137.162 ms	1.982 ms
DeepLog-11-4-64-16	0.074	1.000	0.138	170.506 ms	1.967 ms
DeepLog-11-1-32-10	0.074	1.000	0.138	113.983 ms	1.113 ms
DeepLog-11-2-32-14	0.074	1.000	0.138	125.380 ms	1.633 ms
DeepLog-12-3-128-17	0.074	1.000	0.138	188.157 ms	1.835 ms
DeepLog-11-2-64-15	0.074	1.000	0.138	135.333 ms	1.612 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-11-4-32-19	0.074	1.000	0.138	151.351 ms	2.089 ms
DeepLog-10-2-128-8	0.074	1.000	0.138	157.738 ms	1.471 ms
DeepLog-7-4-32-4	0.074	1.000	0.138	160.527 ms	1.679 ms
DeepLog-12-1-256-18	0.074	1.000	0.138	165.648 ms	2.862 ms
DeepLog-9-4-32-8	0.074	1.000	0.138	159.380 ms	1.846 ms
DeepLog-12-1-32-20	0.074	1.000	0.138	116.737 ms	1.189 ms
DeepLog-11-1-128-11	0.074	1.000	0.138	132.552 ms	1.224 ms
DeepLog-10-1-128-11	0.074	1.000	0.138	133.389 ms	1.211 ms
DeepLog-11-2-32-12	0.074	1.000	0.138	125.380 ms	1.605 ms
DeepLog-12-2-32-17	0.074	1.000	0.138	128.497 ms	4.001 ms
DeepLog-11-3-64-12	0.074	1.000	0.138	150.869 ms	1.715 ms
DeepLog-12-3-32-20	0.074	1.000	0.138	137.887 ms	4.925 ms
DeepLog-11-3-64-11	0.074	1.000	0.138	150.869 ms	1.695 ms
DeepLog-12-3-128-18	0.074	1.000	0.138	188.157 ms	1.854 ms
DeepLog-11-3-128-13	0.074	1.000	0.138	185.024 ms	1.809 ms
DeepLog-12-3-128-19	0.074	1.000	0.138	188.157 ms	1.844 ms
DeepLog-10-1-32-9	0.074	1.000	0.138	115.760 ms	1.336 ms
DeepLog-7-4-32-8	0.074	1.000	0.138	160.527 ms	1.929 ms
DeepLog-11-3-64-14	0.074	1.000	0.138	150.869 ms	1.757 ms
DeepLog-12-1-128-20	0.074	1.000	0.138	133.948 ms	3.093 ms
DeepLog-11-4-32-20	0.074	1.000	0.138	151.351 ms	2.081 ms
DeepLog-10-2-128-10	0.074	1.000	0.138	157.738 ms	1.490 ms
DeepLog-12-1-64-20	0.074	1.000	0.138	120.618 ms	3.085 ms
DeepLog-12-2-32-18	0.074	1.000	0.138	128.497 ms	4.042 ms
DeepLog-11-4-64-17	0.074	1.000	0.138	170.506 ms	2.146 ms
DeepLog-12-1-256-17	0.074	1.000	0.138	165.648 ms	2.836 ms
DeepLog-12-2-32-20	0.074	1.000	0.138	128.497 ms	4.092 ms
DeepLog-12-1-256-20	0.074	1.000	0.138	165.648 ms	3.099 ms
DeepLog-11-1-32-7	0.074	1.000	0.138	113.983 ms	1.092 ms
DeepLog-10-1-128-9	0.074	1.000	0.138	133.389 ms	1.181 ms
DeepLog-11-1-64-19	0.074	1.000	0.138	120.211 ms	2.972 ms
DeepLog-11-3-128-12	0.074	1.000	0.138	185.024 ms	1.783 ms
DeepLog-11-2-32-15	0.074	1.000	0.138	125.380 ms	1.607 ms
DeepLog-10-3-64-10	0.074	1.000	0.138	150.441 ms	1.882 ms
DeepLog-11-3-128-17	0.074	1.000	0.138	185.024 ms	1.885 ms
DeepLog-12-3-64-15	0.074	1.000	0.138	150.436 ms	1.717 ms
DeepLog-10-1-256-13	0.074	1.000	0.138	160.003 ms	1.265 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-12-1-256-19	0.074	1.000	0.138	165.648 ms	2.835 ms
DeepLog-11-2-32-13	0.074	1.000	0.138	125.380 ms	1.580 ms
DeepLog-12-3-64-19	0.074	1.000	0.138	150.436 ms	1.799 ms
DeepLog-11-4-64-12	0.074	1.000	0.138	170.506 ms	1.912 ms
DeepLog-7-2-128-8	0.074	1.000	0.138	159.838 ms	1.515 ms
DeepLog-11-3-128-15	0.074	1.000	0.138	185.024 ms	1.810 ms
DeepLog-12-2-32-19	0.074	1.000	0.138	128.497 ms	4.046 ms
DeepLog-10-2-128-7	0.074	1.000	0.138	157.738 ms	1.419 ms
DeepLog-7-4-32-5	0.074	1.000	0.138	160.527 ms	1.664 ms
DeepLog-11-1-32-8	0.074	1.000	0.138	113.983 ms	1.095 ms
DeepLog-11-4-32-15	0.074	1.000	0.138	151.351 ms	2.018 ms
DeepLog-10-1-32-8	0.074	1.000	0.138	115.760 ms	1.332 ms
DeepLog-12-5-64-20	0.074	1.000	0.138	180.319 ms	2.407 ms
DeepLog-10-1-256-16	0.074	1.000	0.138	160.003 ms	1.287 ms
DeepLog-11-4-64-15	0.074	1.000	0.138	170.506 ms	1.934 ms
DeepLog-12-4-32-19	0.074	1.000	0.138	149.117 ms	5.799 ms
DeepLog-11-2-128-11	0.074	1.000	0.138	160.080 ms	1.600 ms
DeepLog-10-2-32-11	0.074	1.000	0.138	126.327 ms	1.697 ms
DeepLog-10-1-256-15	0.074	1.000	0.138	160.003 ms	1.274 ms
DeepLog-11-3-64-13	0.074	1.000	0.138	150.869 ms	1.741 ms
DeepLog-10-3-64-11	0.074	1.000	0.138	150.441 ms	1.881 ms
DeepLog-12-1-256-15	0.074	1.000	0.138	165.648 ms	2.919 ms
DeepLog-12-4-128-20	0.074	1.000	0.138	212.681 ms	2.242 ms
DeepLog-12-1-64-19	0.074	1.000	0.138	120.618 ms	3.107 ms
DeepLog-12-1-32-19	0.074	1.000	0.138	116.737 ms	1.171 ms
DeepLog-12-1-128-16	0.074	1.000	0.138	133.948 ms	3.041 ms
DeepLog-12-4-64-19	0.074	1.000	0.138	166.874 ms	2.079 ms
DeepLog-11-1-128-15	0.074	1.000	0.138	132.552 ms	1.244 ms
DeepLog-12-2-128-19	0.074	1.000	0.138	160.307 ms	1.558 ms
DeepLog-12-5-64-19	0.074	1.000	0.138	180.319 ms	2.392 ms
DeepLog-12-3-64-20	0.074	1.000	0.138	150.436 ms	1.823 ms
DeepLog-12-1-256-14	0.074	1.000	0.138	165.648 ms	2.910 ms
DeepLog-12-1-64-17	0.074	1.000	0.138	120.618 ms	1.165 ms
DeepLog-11-1-32-9	0.074	1.000	0.138	113.983 ms	1.109 ms
DeepLog-11-1-256-11	0.074	1.000	0.138	163.117 ms	1.245 ms
DeepLog-12-5-64-18	0.074	1.000	0.138	180.319 ms	2.412 ms
DeepLog-12-2-64-19	0.074	1.000	0.138	137.153 ms	1.498 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-10-2-128-9	0.074	1.000	0.138	157.738 ms	1.480 ms
DeepLog-11-3-128-16	0.074	1.000	0.138	185.024 ms	1.843 ms
DeepLog-12-3-64-16	0.074	1.000	0.138	150.436 ms	1.765 ms
DeepLog-12-4-64-17	0.074	1.000	0.138	166.874 ms	2.032 ms
DeepLog-11-1-128-16	0.074	1.000	0.138	132.552 ms	1.250 ms
DeepLog-12-1-256-16	0.074	1.000	0.138	165.648 ms	2.824 ms
DeepLog-12-3-128-16	0.074	1.000	0.138	188.157 ms	1.832 ms
DeepLog-11-1-128-17	0.074	1.000	0.138	132.552 ms	2.912 ms
DeepLog-12-1-64-15	0.074	1.000	0.138	120.618 ms	1.155 ms
DeepLog-11-1-128-12	0.074	1.000	0.138	132.552 ms	1.235 ms
DeepLog-12-4-32-20	0.074	1.000	0.138	149.117 ms	5.817 ms
DeepLog-11-3-128-11	0.074	1.000	0.138	185.024 ms	1.757 ms
DeepLog-12-1-64-16	0.074	1.000	0.138	120.618 ms	1.152 ms
DeepLog-11-2-128-12	0.074	1.000	0.138	160.080 ms	1.600 ms
DeepLog-11-4-64-14	0.074	1.000	0.138	170.506 ms	1.940 ms
DeepLog-12-3-128-20	0.074	1.000	0.138	188.157 ms	1.880 ms
DeepLog-11-1-64-14	0.074	1.000	0.138	120.211 ms	2.905 ms
DeepLog-12-2-128-18	0.074	1.000	0.138	160.307 ms	1.515 ms
DeepLog-12-3-128-15	0.074	1.000	0.138	188.157 ms	1.786 ms
DeepLog-12-3-128-13	0.074	1.000	0.138	188.157 ms	1.800 ms
DeepLog-11-1-64-15	0.074	1.000	0.138	120.211 ms	2.951 ms
DeepLog-12-1-128-18	0.074	1.000	0.138	133.948 ms	3.078 ms
DeepLog-7-4-32-9	0.074	1.000	0.138	160.527 ms	1.969 ms
DeepLog-12-3-64-17	0.074	1.000	0.138	150.436 ms	1.783 ms
DeepLog-12-2-128-16	0.074	1.000	0.138	160.307 ms	1.487 ms
DeepLog-12-1-64-18	0.074	1.000	0.138	120.618 ms	1.167 ms
DeepLog-7-2-128-7	0.074	1.000	0.138	159.838 ms	1.328 ms
DeepLog-11-1-64-16	0.074	1.000	0.138	120.211 ms	2.941 ms
DeepLog-11-2-32-11	0.074	1.000	0.138	125.380 ms	1.565 ms
DeepLog-11-4-64-11	0.074	1.000	0.138	170.506 ms	1.962 ms
DeepLog-11-1-64-17	0.074	1.000	0.138	120.211 ms	2.925 ms
DeepLog-12-1-128-19	0.074	1.000	0.138	133.948 ms	3.094 ms
DeepLog-12-4-64-20	0.074	1.000	0.138	166.874 ms	2.071 ms
DeepLog-10-2-128-12	0.075	1.000	0.139	157.738 ms	1.527 ms
DeepLog-10-2-128-16	0.075	1.000	0.139	157.738 ms	1.596 ms
DeepLog-10-2-64-18	0.075	1.000	0.139	138.626 ms	1.603 ms
DeepLog-11-2-128-14	0.075	1.000	0.139	160.080 ms	1.655 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-10-1-64-17	0.075	1.000	0.139	121.028 ms	1.352 ms
DeepLog-10-4-64-12	0.074	1.000	0.139	177.515 ms	2.314 ms
DeepLog-11-2-128-15	0.075	1.000	0.139	160.080 ms	1.643 ms
DeepLog-9-4-32-9	0.075	1.000	0.139	159.380 ms	1.945 ms
DeepLog-8-1-128-6	0.075	1.000	0.139	140.468 ms	1.054 ms
DeepLog-11-2-128-17	0.075	1.000	0.139	160.080 ms	1.634 ms
DeepLog-10-2-64-20	0.075	1.000	0.139	138.626 ms	1.619 ms
DeepLog-10-3-32-14	0.075	1.000	0.139	136.619 ms	2.080 ms
DeepLog-10-3-32-16	0.075	1.000	0.139	136.619 ms	2.131 ms
DeepLog-11-2-128-18	0.075	1.000	0.139	160.080 ms	1.647 ms
DeepLog-8-1-128-7	0.075	1.000	0.139	140.468 ms	1.083 ms
DeepLog-10-1-64-12	0.075	1.000	0.139	121.028 ms	1.292 ms
DeepLog-11-1-256-15	0.075	1.000	0.139	163.117 ms	1.255 ms
DeepLog-11-1-128-20	0.075	1.000	0.139	132.552 ms	3.120 ms
DeepLog-11-2-64-16	0.074	1.000	0.139	135.333 ms	1.597 ms
DeepLog-11-3-64-16	0.075	1.000	0.139	150.869 ms	4.582 ms
DeepLog-11-1-256-17	0.075	1.000	0.139	163.117 ms	1.286 ms
DeepLog-10-1-128-12	0.075	1.000	0.139	133.389 ms	1.328 ms
DeepLog-11-4-64-19	0.074	1.000	0.139	170.506 ms	2.201 ms
DeepLog-10-3-64-12	0.075	1.000	0.139	150.441 ms	1.917 ms
DeepLog-10-3-64-14	0.075	1.000	0.139	150.441 ms	1.947 ms
DeepLog-10-2-128-15	0.075	1.000	0.139	157.738 ms	1.540 ms
DeepLog-11-2-64-17	0.075	1.000	0.139	135.333 ms	1.627 ms
DeepLog-10-4-32-18	0.075	1.000	0.139	152.722 ms	5.411 ms
DeepLog-10-2-32-14	0.075	1.000	0.139	126.327 ms	3.719 ms
DeepLog-8-3-32-10	0.075	1.000	0.139	137.352 ms	1.653 ms
DeepLog-11-3-32-17	0.075	1.000	0.139	137.162 ms	2.011 ms
DeepLog-11-3-32-19	0.075	1.000	0.139	137.162 ms	2.029 ms
DeepLog-8-1-32-5	0.075	1.000	0.139	116.696 ms	0.954 ms
DeepLog-9-1-32-4	0.075	1.000	0.139	124.475 ms	0.812 ms
DeepLog-10-2-32-13	0.075	1.000	0.139	126.327 ms	3.738 ms
DeepLog-11-1-256-20	0.075	1.000	0.139	163.117 ms	3.225 ms
DeepLog-11-2-64-20	0.075	1.000	0.139	135.333 ms	1.664 ms
DeepLog-11-2-128-16	0.075	1.000	0.139	160.080 ms	1.647 ms
DeepLog-7-4-32-13	0.075	1.000	0.139	160.527 ms	2.022 ms
DeepLog-10-1-64-15	0.075	1.000	0.139	121.028 ms	1.304 ms
DeepLog-11-1-256-13	0.075	1.000	0.139	163.117 ms	1.252 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-11-1-32-17	0.075	1.000	0.139	113.983 ms	1.235 ms
DeepLog-11-1-32-18	0.075	1.000	0.139	113.983 ms	1.263 ms
DeepLog-11-1-32-13	0.075	1.000	0.139	113.983 ms	1.193 ms
DeepLog-10-1-64-16	0.075	1.000	0.139	121.028 ms	1.338 ms
DeepLog-11-3-64-15	0.075	1.000	0.139	150.869 ms	4.489 ms
DeepLog-10-4-32-17	0.075	1.000	0.139	152.722 ms	5.458 ms
DeepLog-10-1-128-14	0.075	1.000	0.139	133.389 ms	1.347 ms
DeepLog-11-2-128-19	0.075	1.000	0.139	160.080 ms	1.630 ms
DeepLog-11-1-256-14	0.075	1.000	0.139	163.117 ms	1.252 ms
DeepLog-11-3-32-20	0.075	1.000	0.139	137.162 ms	2.004 ms
DeepLog-11-1-128-18	0.074	1.000	0.139	132.552 ms	2.896 ms
DeepLog-11-2-32-16	0.075	1.000	0.139	125.380 ms	1.648 ms
DeepLog-10-2-128-14	0.075	1.000	0.139	157.738 ms	1.575 ms
DeepLog-10-3-64-15	0.075	1.000	0.139	150.441 ms	1.924 ms
DeepLog-10-1-64-20	0.075	1.000	0.139	121.028 ms	1.403 ms
DeepLog-7-4-32-11	0.074	1.000	0.139	160.527 ms	2.024 ms
DeepLog-10-2-128-11	0.074	1.000	0.139	157.738 ms	1.519 ms
DeepLog-11-1-32-12	0.074	1.000	0.139	113.983 ms	1.176 ms
DeepLog-10-4-64-15	0.075	1.000	0.139	177.515 ms	2.364 ms
DeepLog-11-1-256-18	0.075	1.000	0.139	163.117 ms	1.282 ms
DeepLog-11-3-32-18	0.075	1.000	0.139	137.162 ms	2.026 ms
DeepLog-11-3-128-18	0.075	1.000	0.139	185.024 ms	1.903 ms
DeepLog-10-2-128-13	0.075	1.000	0.139	157.738 ms	1.561 ms
DeepLog-11-2-64-19	0.075	1.000	0.139	135.333 ms	1.627 ms
DeepLog-10-2-32-15	0.075	1.000	0.139	126.327 ms	3.741 ms
DeepLog-10-3-32-15	0.075	1.000	0.139	136.619 ms	2.130 ms
DeepLog-11-1-256-12	0.074	1.000	0.139	163.117 ms	1.254 ms
DeepLog-11-3-64-17	0.075	1.000	0.139	150.869 ms	4.876 ms
DeepLog-7-4-32-10	0.074	1.000	0.139	160.527 ms	2.018 ms
DeepLog-11-1-256-16	0.075	1.000	0.139	163.117 ms	1.267 ms
DeepLog-10-1-32-11	0.075	1.000	0.139	115.760 ms	1.353 ms
DeepLog-10-1-32-12	0.075	1.000	0.139	115.760 ms	1.351 ms
DeepLog-10-1-32-10	0.075	1.000	0.139	115.760 ms	1.353 ms
DeepLog-11-1-128-19	0.075	1.000	0.139	132.552 ms	2.923 ms
DeepLog-10-4-64-13	0.075	1.000	0.139	177.515 ms	2.303 ms
DeepLog-11-4-64-20	0.075	1.000	0.139	170.506 ms	2.218 ms
DeepLog-10-1-64-13	0.075	1.000	0.139	121.028 ms	1.291 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-7-4-32-14	0.075	1.000	0.139	160.527 ms	2.047 ms
DeepLog-10-1-64-19	0.075	1.000	0.139	121.028 ms	1.361 ms
DeepLog-10-1-64-14	0.075	1.000	0.139	121.028 ms	1.307 ms
DeepLog-11-2-64-18	0.075	1.000	0.139	135.333 ms	1.644 ms
DeepLog-10-4-64-14	0.075	1.000	0.139	177.515 ms	2.381 ms
DeepLog-11-3-32-16	0.074	1.000	0.139	137.162 ms	1.980 ms
DeepLog-10-3-64-13	0.075	1.000	0.139	150.441 ms	1.963 ms
DeepLog-11-3-128-19	0.075	1.000	0.139	185.024 ms	1.917 ms
DeepLog-11-1-32-16	0.075	1.000	0.139	113.983 ms	1.227 ms
DeepLog-10-4-64-16	0.075	1.000	0.139	177.515 ms	2.390 ms
DeepLog-7-4-32-15	0.075	1.000	0.139	160.527 ms	2.061 ms
DeepLog-10-3-32-13	0.075	1.000	0.139	136.619 ms	2.081 ms
DeepLog-10-1-128-13	0.075	1.000	0.139	133.389 ms	1.340 ms
DeepLog-11-2-32-17	0.075	1.000	0.139	125.380 ms	1.644 ms
DeepLog-11-2-128-20	0.075	1.000	0.139	160.080 ms	1.713 ms
DeepLog-10-1-64-18	0.075	1.000	0.139	121.028 ms	1.351 ms
DeepLog-9-1-32-3	0.075	1.000	0.139	124.475 ms	0.799 ms
DeepLog-11-2-32-18	0.075	1.000	0.139	125.380 ms	1.654 ms
DeepLog-8-3-32-9	0.074	1.000	0.139	137.352 ms	1.610 ms
DeepLog-10-4-32-16	0.075	1.000	0.139	152.722 ms	5.449 ms
DeepLog-10-2-64-19	0.075	1.000	0.139	138.626 ms	1.635 ms
DeepLog-11-3-128-20	0.075	1.000	0.139	185.024 ms	4.952 ms
DeepLog-11-1-256-19	0.075	1.000	0.139	163.117 ms	1.309 ms
DeepLog-11-1-32-11	0.074	1.000	0.139	113.983 ms	1.171 ms
DeepLog-10-3-32-12	0.075	1.000	0.139	136.619 ms	2.030 ms
DeepLog-11-1-32-15	0.075	1.000	0.139	113.983 ms	1.214 ms
DeepLog-7-4-32-12	0.075	1.000	0.139	160.527 ms	2.021 ms
DeepLog-11-1-32-14	0.075	1.000	0.139	113.983 ms	1.214 ms
DeepLog-10-2-128-17	0.075	1.000	0.140	157.738 ms	1.646 ms
DeepLog-10-2-128-19	0.075	1.000	0.140	157.738 ms	1.649 ms
DeepLog-8-4-32-8	0.075	1.000	0.140	161.176 ms	1.769 ms
DeepLog-10-2-128-18	0.075	1.000	0.140	157.738 ms	1.633 ms
DeepLog-8-2-128-9	0.076	1.000	0.140	169.714 ms	1.353 ms
DeepLog-11-2-32-19	0.075	1.000	0.140	125.380 ms	1.664 ms
DeepLog-9-1-128-7	0.075	1.000	0.140	132.636 ms	1.082 ms
DeepLog-8-1-256-9	0.075	1.000	0.140	166.658 ms	1.143 ms
DeepLog-10-1-32-15	0.075	1.000	0.140	115.760 ms	1.409 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-10-4-64-19	0.075	1.000	0.140	177.515 ms	5.824 ms
DeepLog-8-4-32-9	0.075	1.000	0.140	161.176 ms	1.767 ms
DeepLog-10-1-256-20	0.075	1.000	0.140	160.003 ms	1.385 ms
DeepLog-10-4-64-20	0.076	1.000	0.140	177.515 ms	5.731 ms
DeepLog-10-2-32-19	0.075	1.000	0.140	126.327 ms	3.876 ms
DeepLog-10-2-32-16	0.075	1.000	0.140	126.327 ms	3.794 ms
DeepLog-8-2-128-8	0.075	1.000	0.140	169.714 ms	1.373 ms
DeepLog-9-1-32-5	0.075	1.000	0.140	124.475 ms	0.831 ms
DeepLog-9-1-256-6	0.075	1.000	0.140	159.246 ms	1.001 ms
DeepLog-8-1-32-6	0.075	1.000	0.140	116.696 ms	0.976 ms
DeepLog-10-4-32-20	0.075	1.000	0.140	152.722 ms	5.501 ms
DeepLog-7-4-32-17	0.075	1.000	0.140	160.527 ms	2.117 ms
DeepLog-10-1-32-13	0.075	1.000	0.140	115.760 ms	1.375 ms
DeepLog-10-1-256-19	0.075	1.000	0.140	160.003 ms	1.369 ms
DeepLog-8-1-64-7	0.075	1.000	0.140	120.598 ms	1.059 ms
DeepLog-10-1-128-17	0.075	1.000	0.140	133.389 ms	1.378 ms
DeepLog-9-4-64-8	0.075	1.000	0.140	170.110 ms	5.227 ms
DeepLog-10-2-32-20	0.075	1.000	0.140	126.327 ms	3.890 ms
DeepLog-8-1-64-8	0.075	1.000	0.140	120.598 ms	1.053 ms
DeepLog-10-1-128-16	0.075	1.000	0.140	133.389 ms	1.388 ms
DeepLog-10-3-32-17	0.075	1.000	0.140	136.619 ms	2.141 ms
DeepLog-7-4-32-16	0.075	1.000	0.140	160.527 ms	2.109 ms
DeepLog-11-1-32-19	0.075	1.000	0.140	113.983 ms	1.285 ms
DeepLog-11-3-64-20	0.075	1.000	0.140	150.869 ms	5.039 ms
DeepLog-10-1-256-17	0.075	1.000	0.140	160.003 ms	1.355 ms
DeepLog-7-4-32-18	0.075	1.000	0.140	160.527 ms	2.144 ms
DeepLog-10-2-32-17	0.075	1.000	0.140	126.327 ms	3.884 ms
DeepLog-10-1-128-15	0.075	1.000	0.140	133.389 ms	1.371 ms
DeepLog-9-2-32-6	0.075	1.000	0.140	135.313 ms	1.237 ms
DeepLog-11-3-64-19	0.075	1.000	0.140	150.869 ms	4.974 ms
DeepLog-8-1-64-9	0.075	1.000	0.140	120.598 ms	1.059 ms
DeepLog-10-1-32-14	0.075	1.000	0.140	115.760 ms	1.399 ms
DeepLog-10-3-64-16	0.075	1.000	0.140	150.441 ms	1.982 ms
DeepLog-8-1-64-11	0.076	1.000	0.140	120.598 ms	1.129 ms
DeepLog-10-4-64-18	0.075	1.000	0.140	177.515 ms	5.748 ms
DeepLog-10-3-64-17	0.075	1.000	0.140	150.441 ms	1.986 ms
DeepLog-10-1-128-19	0.076	1.000	0.140	133.389 ms	1.435 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-10-4-64-17	0.075	1.000	0.140	177.515 ms	5.806 ms
DeepLog-8-2-64-10	0.075	1.000	0.140	137.269 ms	1.417 ms
DeepLog-8-1-256-8	0.075	1.000	0.140	166.658 ms	1.114 ms
DeepLog-10-1-256-18	0.075	1.000	0.140	160.003 ms	1.349 ms
DeepLog-7-4-32-19	0.075	1.000	0.140	160.527 ms	2.140 ms
DeepLog-10-1-128-18	0.075	1.000	0.140	133.389 ms	1.393 ms
DeepLog-11-1-32-20	0.075	1.000	0.140	113.983 ms	1.292 ms
DeepLog-8-2-64-11	0.076	1.000	0.140	137.269 ms	1.414 ms
DeepLog-11-2-32-20	0.075	1.000	0.140	125.380 ms	1.692 ms
DeepLog-10-2-32-18	0.075	1.000	0.140	126.327 ms	3.852 ms
DeepLog-9-1-256-7	0.075	1.000	0.140	159.246 ms	1.037 ms
DeepLog-11-3-64-18	0.075	1.000	0.140	150.869 ms	4.897 ms
DeepLog-9-1-128-8	0.075	1.000	0.140	132.636 ms	1.091 ms
DeepLog-10-3-32-18	0.075	1.000	0.140	136.619 ms	2.144 ms
DeepLog-8-2-32-9	0.075	1.000	0.140	127.653 ms	3.535 ms
DeepLog-10-2-128-20	0.076	1.000	0.140	157.738 ms	1.798 ms
DeepLog-9-1-128-9	0.075	1.000	0.140	132.636 ms	1.091 ms
DeepLog-10-4-32-19	0.075	1.000	0.140	152.722 ms	5.430 ms
DeepLog-9-1-32-6	0.075	1.000	0.140	124.475 ms	0.953 ms
DeepLog-11-1-64-20	0.075	1.000	0.140	120.211 ms	3.013 ms
DeepLog-8-1-256-7	0.075	1.000	0.140	166.658 ms	0.987 ms
DeepLog-10-3-64-18	0.075	1.000	0.140	150.441 ms	1.995 ms
DeepLog-8-1-64-10	0.075	1.000	0.140	120.598 ms	1.133 ms
DeepLog-10-1-32-20	0.076	1.000	0.141	115.760 ms	3.274 ms
DeepLog-9-1-128-10	0.076	1.000	0.141	132.636 ms	1.107 ms
DeepLog-9-3-32-7	0.076	1.000	0.141	137.238 ms	1.445 ms
DeepLog-9-4-32-10	0.076	1.000	0.141	159.380 ms	2.156 ms
DeepLog-9-4-32-11	0.076	1.000	0.141	159.380 ms	2.172 ms
DeepLog-10-1-128-20	0.076	1.000	0.141	133.389 ms	1.411 ms
DeepLog-7-1-32-5	0.076	1.000	0.141	117.337 ms	1.123 ms
DeepLog-9-1-128-11	0.076	1.000	0.141	132.636 ms	1.115 ms
DeepLog-8-1-32-8	0.076	1.000	0.141	116.696 ms	1.025 ms
DeepLog-9-1-256-8	0.076	1.000	0.141	159.246 ms	1.088 ms
DeepLog-8-1-32-7	0.076	1.000	0.141	116.696 ms	1.016 ms
DeepLog-9-3-32-8	0.076	1.000	0.141	137.238 ms	1.487 ms
DeepLog-9-1-256-9	0.076	1.000	0.141	159.246 ms	1.093 ms
DeepLog-10-1-32-17	0.076	1.000	0.141	115.760 ms	3.281 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-8-2-128-10	0.076	1.000	0.141	169.714 ms	1.386 ms
DeepLog-10-3-64-19	0.076	1.000	0.141	150.441 ms	2.022 ms
DeepLog-8-1-64-12	0.076	1.000	0.141	120.598 ms	1.159 ms
DeepLog-10-3-64-20	0.076	1.000	0.141	150.441 ms	2.029 ms
DeepLog-8-1-32-9	0.076	1.000	0.141	116.696 ms	1.068 ms
DeepLog-10-1-32-16	0.076	1.000	0.141	115.760 ms	3.244 ms
DeepLog-9-3-32-9	0.076	1.000	0.141	137.238 ms	1.526 ms
DeepLog-9-2-32-7	0.076	1.000	0.141	135.313 ms	1.345 ms
DeepLog-8-2-32-10	0.076	1.000	0.141	127.653 ms	3.567 ms
DeepLog-10-1-32-18	0.076	1.000	0.141	115.760 ms	3.259 ms
DeepLog-8-1-64-13	0.076	1.000	0.141	120.598 ms	1.162 ms
DeepLog-10-1-32-19	0.076	1.000	0.141	115.760 ms	3.199 ms
DeepLog-8-1-128-8	0.076	1.000	0.141	140.468 ms	1.108 ms
DeepLog-9-3-32-10	0.076	1.000	0.141	137.238 ms	1.601 ms
DeepLog-8-4-32-10	0.076	1.000	0.142	161.176 ms	1.869 ms
DeepLog-8-1-64-14	0.076	1.000	0.142	120.598 ms	1.151 ms
DeepLog-8-4-32-13	0.077	1.000	0.142	161.176 ms	1.906 ms
DeepLog-9-4-64-10	0.077	1.000	0.142	170.110 ms	5.288 ms
DeepLog-9-4-64-9	0.076	1.000	0.142	170.110 ms	5.284 ms
DeepLog-8-4-32-12	0.077	1.000	0.142	161.176 ms	1.906 ms
DeepLog-9-4-32-14	0.076	1.000	0.142	159.380 ms	2.206 ms
DeepLog-9-4-32-12	0.076	1.000	0.142	159.380 ms	2.179 ms
DeepLog-8-2-64-15	0.077	1.000	0.142	137.269 ms	1.480 ms
DeepLog-9-1-64-11	0.077	1.000	0.142	128.810 ms	1.119 ms
DeepLog-8-2-128-11	0.077	1.000	0.142	169.714 ms	1.405 ms
DeepLog-8-1-32-14	0.077	1.000	0.142	116.696 ms	1.155 ms
DeepLog-9-3-32-14	0.076	1.000	0.142	137.238 ms	1.626 ms
DeepLog-8-2-64-12	0.077	1.000	0.142	137.269 ms	1.489 ms
DeepLog-8-4-64-11	0.076	1.000	0.142	176.597 ms	1.930 ms
DeepLog-9-3-32-12	0.076	1.000	0.142	137.238 ms	1.624 ms
DeepLog-8-4-32-11	0.077	1.000	0.142	161.176 ms	1.897 ms
DeepLog-9-2-32-9	0.077	1.000	0.142	135.313 ms	1.439 ms
DeepLog-9-3-32-11	0.076	1.000	0.142	137.238 ms	1.608 ms
DeepLog-9-3-32-13	0.076	1.000	0.142	137.238 ms	1.614 ms
DeepLog-8-1-256-10	0.076	1.000	0.142	166.658 ms	1.147 ms
DeepLog-9-1-32-7	0.076	1.000	0.142	124.475 ms	1.001 ms
DeepLog-8-1-32-15	0.077	1.000	0.142	116.696 ms	1.185 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-9-2-32-8	0.077	1.000	0.142	135.313 ms	1.413 ms
DeepLog-7-1-32-6	0.076	1.000	0.142	117.337 ms	1.140 ms
DeepLog-9-4-32-13	0.076	1.000	0.142	159.380 ms	2.181 ms
DeepLog-7-1-32-9	0.077	1.000	0.142	117.337 ms	1.194 ms
DeepLog-8-4-64-10	0.076	1.000	0.142	176.597 ms	1.933 ms
DeepLog-8-1-32-10	0.076	1.000	0.142	116.696 ms	1.115 ms
DeepLog-8-1-32-12	0.076	1.000	0.142	116.696 ms	1.135 ms
DeepLog-9-4-32-16	0.077	1.000	0.142	159.380 ms	2.194 ms
DeepLog-8-1-256-11	0.076	1.000	0.142	166.658 ms	1.158 ms
DeepLog-9-4-32-15	0.076	1.000	0.142	159.380 ms	2.191 ms
DeepLog-8-1-128-9	0.076	1.000	0.142	140.468 ms	1.130 ms
DeepLog-8-2-64-13	0.077	1.000	0.142	137.269 ms	1.475 ms
DeepLog-9-1-64-12	0.077	1.000	0.142	128.810 ms	1.180 ms
DeepLog-8-2-64-14	0.077	1.000	0.142	137.269 ms	1.487 ms
DeepLog-7-1-32-7	0.076	1.000	0.142	117.337 ms	1.167 ms
DeepLog-8-1-32-13	0.076	1.000	0.142	116.696 ms	1.155 ms
DeepLog-7-1-32-8	0.077	1.000	0.142	117.337 ms	1.182 ms
DeepLog-8-1-32-11	0.076	1.000	0.142	116.696 ms	1.126 ms
DeepLog-9-4-32-18	0.077	1.000	0.143	159.380 ms	2.201 ms
DeepLog-9-3-64-10	0.077	1.000	0.143	158.282 ms	1.418 ms
DeepLog-9-3-64-12	0.077	1.000	0.143	158.282 ms	1.583 ms
DeepLog-9-4-32-17	0.077	1.000	0.143	159.380 ms	2.229 ms
DeepLog-8-3-64-9	0.077	1.000	0.143	151.216 ms	1.659 ms
DeepLog-8-2-32-11	0.077	1.000	0.143	127.653 ms	3.620 ms
DeepLog-9-2-64-11	0.077	1.000	0.143	146.191 ms	1.515 ms
DeepLog-10-3-32-19	0.077	1.000	0.143	136.619 ms	2.308 ms
DeepLog-8-2-64-16	0.077	1.000	0.143	137.269 ms	1.491 ms
DeepLog-9-4-32-20	0.077	1.000	0.143	159.380 ms	2.261 ms
DeepLog-8-3-64-12	0.077	1.000	0.143	151.216 ms	1.745 ms
DeepLog-9-3-32-16	0.077	1.000	0.143	137.238 ms	4.560 ms
DeepLog-8-1-32-16	0.077	1.000	0.143	116.696 ms	1.196 ms
DeepLog-9-3-64-11	0.077	1.000	0.143	158.282 ms	1.417 ms
DeepLog-9-4-32-19	0.077	1.000	0.143	159.380 ms	2.212 ms
DeepLog-8-3-32-11	0.077	1.000	0.143	137.352 ms	1.653 ms
DeepLog-7-2-128-10	0.077	1.000	0.143	159.838 ms	1.533 ms
DeepLog-10-3-32-20	0.077	1.000	0.143	136.619 ms	2.360 ms
DeepLog-7-2-128-9	0.077	1.000	0.143	159.838 ms	1.528 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-9-1-256-10	0.077	1.000	0.143	159.246 ms	1.122 ms
DeepLog-9-2-64-10	0.077	1.000	0.143	146.191 ms	1.523 ms
DeepLog-9-3-64-13	0.077	1.000	0.143	158.282 ms	1.601 ms
DeepLog-8-3-64-11	0.077	1.000	0.143	151.216 ms	1.730 ms
DeepLog-9-3-32-15	0.077	1.000	0.143	137.238 ms	4.676 ms
DeepLog-8-3-64-10	0.077	1.000	0.143	151.216 ms	1.687 ms
DeepLog-9-1-256-12	0.077	1.000	0.143	159.246 ms	2.873 ms
DeepLog-9-1-256-11	0.077	1.000	0.143	159.246 ms	2.883 ms
DeepLog-7-2-128-13	0.077	1.000	0.143	159.838 ms	1.541 ms
DeepLog-7-2-128-12	0.077	1.000	0.143	159.838 ms	1.550 ms
DeepLog-7-2-128-11	0.077	1.000	0.143	159.838 ms	1.541 ms
DeepLog-8-2-32-13	0.077	1.000	0.143	127.653 ms	3.622 ms
DeepLog-9-2-64-9	0.077	1.000	0.143	146.191 ms	1.503 ms
DeepLog-9-1-64-13	0.077	1.000	0.143	128.810 ms	1.185 ms
DeepLog-8-2-32-12	0.077	1.000	0.143	127.653 ms	3.600 ms
DeepLog-9-1-256-13	0.077	1.000	0.143	159.246 ms	2.862 ms
DeepLog-9-3-32-17	0.077	1.000	0.143	137.238 ms	4.624 ms
DeepLog-9-1-128-12	0.077	1.000	0.143	132.636 ms	1.209 ms
DeepLog-8-1-256-12	0.078	1.000	0.144	166.658 ms	1.212 ms
DeepLog-8-2-32-15	0.078	1.000	0.144	127.653 ms	3.572 ms
DeepLog-9-3-32-18	0.078	1.000	0.144	137.238 ms	4.700 ms
DeepLog-9-1-32-10	0.078	1.000	0.144	124.475 ms	1.079 ms
DeepLog-8-2-32-14	0.078	1.000	0.144	127.653 ms	3.605 ms
DeepLog-8-1-128-11	0.077	1.000	0.144	140.468 ms	1.167 ms
DeepLog-9-3-64-14	0.077	1.000	0.144	158.282 ms	1.605 ms
DeepLog-9-1-32-14	0.078	1.000	0.144	124.475 ms	1.139 ms
DeepLog-8-3-32-13	0.077	1.000	0.144	137.352 ms	1.706 ms
DeepLog-8-4-32-16	0.078	1.000	0.144	161.176 ms	1.999 ms
DeepLog-9-2-32-10	0.078	1.000	0.144	135.313 ms	1.468 ms
DeepLog-9-2-32-12	0.078	1.000	0.144	135.313 ms	1.507 ms
DeepLog-9-1-32-9	0.078	1.000	0.144	124.475 ms	1.027 ms
DeepLog-9-2-32-13	0.078	1.000	0.144	135.313 ms	1.516 ms
DeepLog-8-1-32-17	0.078	1.000	0.144	116.696 ms	1.213 ms
DeepLog-9-1-64-14	0.078	1.000	0.144	128.810 ms	1.240 ms
DeepLog-9-1-32-11	0.078	1.000	0.144	124.475 ms	1.107 ms
DeepLog-9-3-64-15	0.078	1.000	0.144	158.282 ms	1.681 ms
DeepLog-8-2-128-13	0.078	1.000	0.144	169.714 ms	1.472 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-9-2-32-11	0.078	1.000	0.144	135.313 ms	1.469 ms
DeepLog-9-3-32-20	0.078	1.000	0.144	137.238 ms	4.687 ms
DeepLog-8-1-256-13	0.078	1.000	0.144	166.658 ms	1.216 ms
DeepLog-8-4-32-15	0.078	1.000	0.144	161.176 ms	1.978 ms
DeepLog-8-4-32-14	0.078	1.000	0.144	161.176 ms	1.903 ms
DeepLog-9-1-32-12	0.078	1.000	0.144	124.475 ms	1.129 ms
DeepLog-8-1-64-15	0.078	1.000	0.144	120.598 ms	1.183 ms
DeepLog-9-3-32-19	0.078	1.000	0.144	137.238 ms	4.693 ms
DeepLog-8-2-128-12	0.078	1.000	0.144	169.714 ms	1.471 ms
DeepLog-9-2-64-12	0.077	1.000	0.144	146.191 ms	1.534 ms
DeepLog-9-2-64-13	0.078	1.000	0.144	146.191 ms	1.537 ms
DeepLog-8-1-128-10	0.077	1.000	0.144	140.468 ms	1.147 ms
DeepLog-9-1-32-8	0.077	1.000	0.144	124.475 ms	1.017 ms
DeepLog-8-3-32-12	0.077	1.000	0.144	137.352 ms	1.691 ms
DeepLog-9-1-64-15	0.078	1.000	0.144	128.810 ms	1.217 ms
DeepLog-8-3-32-14	0.077	1.000	0.144	137.352 ms	1.726 ms
DeepLog-8-1-256-14	0.078	1.000	0.144	166.658 ms	1.221 ms
DeepLog-9-1-32-13	0.078	1.000	0.144	124.475 ms	1.134 ms
DeepLog-9-2-32-14	0.078	1.000	0.145	135.313 ms	1.570 ms
DeepLog-9-3-64-18	0.078	1.000	0.145	158.282 ms	1.749 ms
DeepLog-9-2-32-16	0.078	1.000	0.145	135.313 ms	1.567 ms
DeepLog-9-4-64-14	0.078	1.000	0.145	170.110 ms	5.361 ms
DeepLog-8-1-128-14	0.078	1.000	0.145	140.468 ms	1.161 ms
DeepLog-9-3-64-17	0.078	1.000	0.145	158.282 ms	1.734 ms
DeepLog-9-3-64-19	0.078	1.000	0.145	158.282 ms	1.757 ms
DeepLog-8-1-64-18	0.078	1.000	0.145	120.598 ms	1.192 ms
DeepLog-8-1-64-17	0.078	1.000	0.145	120.598 ms	1.190 ms
DeepLog-8-1-128-12	0.078	1.000	0.145	140.468 ms	1.165 ms
DeepLog-8-1-64-16	0.078	1.000	0.145	120.598 ms	1.186 ms
DeepLog-9-1-256-16	0.078	1.000	0.145	159.246 ms	3.014 ms
DeepLog-8-2-64-17	0.078	1.000	0.145	137.269 ms	1.535 ms
DeepLog-8-3-32-15	0.078	1.000	0.145	137.352 ms	1.744 ms
DeepLog-9-2-32-15	0.078	1.000	0.145	135.313 ms	1.569 ms
DeepLog-8-2-64-18	0.078	1.000	0.145	137.269 ms	1.559 ms
DeepLog-8-4-64-12	0.078	1.000	0.145	176.597 ms	2.027 ms
DeepLog-9-4-64-12	0.078	1.000	0.145	170.110 ms	5.260 ms
DeepLog-9-1-256-15	0.078	1.000	0.145	159.246 ms	2.979 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-9-1-128-13	0.078	1.000	0.145	132.636 ms	1.243 ms
DeepLog-9-4-64-13	0.078	1.000	0.145	170.110 ms	5.394 ms
DeepLog-9-3-64-16	0.078	1.000	0.145	158.282 ms	1.750 ms
DeepLog-9-1-256-14	0.078	1.000	0.145	159.246 ms	2.895 ms
DeepLog-9-1-256-18	0.078	1.000	0.145	159.246 ms	3.276 ms
DeepLog-9-4-64-11	0.078	1.000	0.145	170.110 ms	5.331 ms
DeepLog-8-1-128-13	0.078	1.000	0.145	140.468 ms	1.168 ms
DeepLog-9-1-64-16	0.078	1.000	0.145	128.810 ms	1.237 ms
DeepLog-9-1-256-17	0.078	1.000	0.145	159.246 ms	3.008 ms
DeepLog-3-1-64-5	0.079	1.000	0.146	142.055 ms	0.763 ms
DeepLog-9-4-64-18	0.079	1.000	0.146	170.110 ms	5.355 ms
DeepLog-8-1-256-17	0.079	1.000	0.146	166.658 ms	1.253 ms
DeepLog-8-2-32-18	0.079	1.000	0.146	127.653 ms	3.610 ms
DeepLog-8-4-64-13	0.079	1.000	0.146	176.597 ms	5.019 ms
DeepLog-9-4-64-16	0.079	1.000	0.146	170.110 ms	5.432 ms
DeepLog-8-3-32-18	0.079	1.000	0.146	137.352 ms	1.824 ms
DeepLog-9-2-64-15	0.079	1.000	0.146	146.191 ms	1.612 ms
DeepLog-8-1-256-16	0.079	1.000	0.146	166.658 ms	1.249 ms
DeepLog-8-3-64-16	0.079	1.000	0.146	151.216 ms	1.812 ms
DeepLog-8-2-32-16	0.079	1.000	0.146	127.653 ms	3.599 ms
DeepLog-8-1-256-20	0.079	1.000	0.146	166.658 ms	1.303 ms
DeepLog-9-4-64-19	0.079	1.000	0.146	170.110 ms	5.588 ms
DeepLog-3-1-64-9	0.079	1.000	0.146	142.055 ms	0.773 ms
DeepLog-8-2-128-16	0.079	1.000	0.146	169.714 ms	1.498 ms
DeepLog-8-1-256-19	0.079	1.000	0.146	166.658 ms	1.271 ms
DeepLog-9-1-32-15	0.079	1.000	0.146	124.475 ms	1.172 ms
DeepLog-3-1-32-5	0.079	1.000	0.146	121.034 ms	0.775 ms
DeepLog-9-2-64-14	0.079	1.000	0.146	146.191 ms	1.620 ms
DeepLog-8-2-32-19	0.079	1.000	0.146	127.653 ms	3.602 ms
DeepLog-3-1-64-4	0.079	1.000	0.146	142.055 ms	0.756 ms
DeepLog-3-3-64-4	0.079	1.000	0.146	158.154 ms	1.000 ms
DeepLog-8-3-64-15	0.079	1.000	0.146	151.216 ms	1.799 ms
DeepLog-8-3-64-17	0.079	1.000	0.146	151.216 ms	1.836 ms
DeepLog-8-1-64-19	0.079	1.000	0.146	120.598 ms	1.210 ms
DeepLog-3-1-32-9	0.079	1.000	0.146	121.034 ms	0.790 ms
DeepLog-3-1-64-6	0.079	1.000	0.146	142.055 ms	0.761 ms
DeepLog-8-3-32-17	0.079	1.000	0.146	137.352 ms	1.829 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-8-4-32-17	0.079	1.000	0.146	161.176 ms	2.050 ms
DeepLog-3-3-32-5	0.079	1.000	0.146	141.391 ms	0.986 ms
DeepLog-3-2-32-4	0.079	1.000	0.146	135.844 ms	0.886 ms
DeepLog-8-1-256-18	0.079	1.000	0.146	166.658 ms	1.250 ms
DeepLog-3-3-64-5	0.079	1.000	0.146	158.154 ms	1.010 ms
DeepLog-8-1-256-15	0.079	1.000	0.146	166.658 ms	1.240 ms
DeepLog-8-3-64-14	0.079	1.000	0.146	151.216 ms	1.791 ms
DeepLog-9-1-32-16	0.079	1.000	0.146	124.475 ms	1.192 ms
DeepLog-3-3-64-3	0.079	1.000	0.146	158.154 ms	0.997 ms
DeepLog-8-2-32-17	0.079	1.000	0.146	127.653 ms	3.599 ms
DeepLog-9-2-32-18	0.079	1.000	0.146	135.313 ms	1.577 ms
DeepLog-9-4-64-17	0.079	1.000	0.146	170.110 ms	5.309 ms
DeepLog-3-3-64-6	0.079	1.000	0.146	158.154 ms	1.010 ms
DeepLog-3-1-32-7	0.079	1.000	0.146	121.034 ms	0.781 ms
DeepLog-3-1-32-10	0.079	1.000	0.146	121.034 ms	0.797 ms
DeepLog-8-2-128-17	0.079	1.000	0.146	169.714 ms	1.498 ms
DeepLog-3-1-32-6	0.079	1.000	0.146	121.034 ms	0.774 ms
DeepLog-3-2-32-6	0.079	1.000	0.146	135.844 ms	0.891 ms
DeepLog-8-2-128-14	0.079	1.000	0.146	169.714 ms	1.501 ms
DeepLog-3-1-64-7	0.079	1.000	0.146	142.055 ms	0.760 ms
DeepLog-3-2-32-5	0.079	1.000	0.146	135.844 ms	0.891 ms
DeepLog-3-3-32-4	0.079	1.000	0.146	141.391 ms	0.995 ms
DeepLog-3-3-32-7	0.079	1.000	0.146	141.391 ms	1.009 ms
DeepLog-8-2-32-20	0.079	1.000	0.146	127.653 ms	3.581 ms
DeepLog-3-3-64-8	0.079	1.000	0.146	158.154 ms	1.019 ms
DeepLog-8-3-64-13	0.079	1.000	0.146	151.216 ms	1.753 ms
DeepLog-3-2-32-3	0.079	1.000	0.146	135.844 ms	0.866 ms
DeepLog-3-1-64-3	0.079	1.000	0.146	142.055 ms	0.742 ms
DeepLog-9-2-32-17	0.078	1.000	0.146	135.313 ms	1.579 ms
DeepLog-3-1-32-8	0.079	1.000	0.146	121.034 ms	0.787 ms
DeepLog-3-1-32-3	0.079	1.000	0.146	121.034 ms	0.757 ms
DeepLog-9-4-64-20	0.079	1.000	0.146	170.110 ms	5.605 ms
DeepLog-3-3-64-9	0.079	1.000	0.146	158.154 ms	1.023 ms
DeepLog-3-3-32-6	0.079	1.000	0.146	141.391 ms	1.003 ms
DeepLog-3-1-64-8	0.079	1.000	0.146	142.055 ms	0.775 ms
DeepLog-8-3-32-16	0.079	1.000	0.146	137.352 ms	1.826 ms
DeepLog-8-4-64-14	0.079	1.000	0.146	176.597 ms	5.032 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-9-1-128-14	0.079	1.000	0.146	132.636 ms	1.254 ms
DeepLog-8-4-32-18	0.079	1.000	0.146	161.176 ms	2.081 ms
DeepLog-8-2-128-15	0.079	1.000	0.146	169.714 ms	1.495 ms
DeepLog-3-3-64-7	0.079	1.000	0.146	158.154 ms	1.017 ms
DeepLog-9-1-64-17	0.079	1.000	0.146	128.810 ms	1.266 ms
DeepLog-3-3-32-3	0.079	1.000	0.146	141.391 ms	0.971 ms
DeepLog-3-1-32-4	0.079	1.000	0.146	121.034 ms	0.763 ms
DeepLog-9-4-64-15	0.078	1.000	0.146	170.110 ms	5.334 ms
DeepLog-8-1-64-20	0.079	1.000	0.146	120.598 ms	1.210 ms
DeepLog-3-3-64-10	0.079	1.000	0.146	158.154 ms	1.027 ms
DeepLog-8-2-64-19	0.079	1.000	0.147	137.269 ms	1.577 ms
DeepLog-3-1-32-12	0.079	1.000	0.147	121.034 ms	0.792 ms
DeepLog-3-3-64-15	0.079	1.000	0.147	158.154 ms	1.073 ms
DeepLog-3-3-64-20	0.080	1.000	0.147	158.154 ms	1.084 ms
DeepLog-3-3-32-19	0.079	1.000	0.147	141.391 ms	1.024 ms
DeepLog-8-1-128-16	0.079	1.000	0.147	140.468 ms	1.222 ms
DeepLog-3-1-64-17	0.080	1.000	0.147	142.055 ms	2.929 ms
DeepLog-9-1-64-19	0.079	1.000	0.147	128.810 ms	1.275 ms
DeepLog-3-3-32-20	0.079	1.000	0.147	141.391 ms	1.053 ms
DeepLog-9-1-32-18	0.079	1.000	0.147	124.475 ms	3.094 ms
DeepLog-8-4-32-19	0.079	1.000	0.147	161.176 ms	2.080 ms
DeepLog-3-3-64-13	0.079	1.000	0.147	158.154 ms	1.068 ms
DeepLog-3-3-32-17	0.079	1.000	0.147	141.391 ms	1.049 ms
DeepLog-8-1-32-19	0.079	1.000	0.147	116.696 ms	1.269 ms
DeepLog-3-1-64-15	0.080	1.000	0.147	142.055 ms	2.913 ms
DeepLog-3-1-32-11	0.079	1.000	0.147	121.034 ms	0.793 ms
DeepLog-3-2-32-14	0.079	1.000	0.147	135.844 ms	0.939 ms
DeepLog-8-3-32-20	0.079	1.000	0.147	137.352 ms	1.884 ms
DeepLog-3-1-64-16	0.080	1.000	0.147	142.055 ms	2.942 ms
DeepLog-3-3-32-9	0.079	1.000	0.147	141.391 ms	1.017 ms
DeepLog-3-2-32-17	0.079	1.000	0.147	135.844 ms	0.946 ms
DeepLog-9-1-32-17	0.079	1.000	0.147	124.475 ms	3.166 ms
DeepLog-8-1-128-18	0.079	1.000	0.147	140.468 ms	1.210 ms
DeepLog-9-1-128-19	0.079	1.000	0.147	132.636 ms	1.253 ms
DeepLog-3-2-32-18	0.079	1.000	0.147	135.844 ms	0.956 ms
DeepLog-3-3-32-12	0.079	1.000	0.147	141.391 ms	1.024 ms
DeepLog-9-2-32-19	0.079	1.000	0.147	135.313 ms	1.623 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-9-1-32-20	0.080	1.000	0.147	124.475 ms	3.157 ms
DeepLog-3-2-32-12	0.079	1.000	0.147	135.844 ms	0.936 ms
DeepLog-8-4-64-16	0.079	1.000	0.147	176.597 ms	5.075 ms
DeepLog-9-1-64-20	0.079	1.000	0.147	128.810 ms	1.281 ms
DeepLog-9-1-128-18	0.079	1.000	0.147	132.636 ms	1.269 ms
DeepLog-3-3-64-14	0.079	1.000	0.147	158.154 ms	1.076 ms
DeepLog-3-1-32-15	0.079	1.000	0.147	121.034 ms	0.802 ms
DeepLog-3-3-64-11	0.079	1.000	0.147	158.154 ms	1.073 ms
DeepLog-3-1-64-11	0.079	1.000	0.147	142.055 ms	0.823 ms
DeepLog-8-3-64-18	0.079	1.000	0.147	151.216 ms	1.819 ms
DeepLog-3-3-32-10	0.079	1.000	0.147	141.391 ms	1.024 ms
DeepLog-9-1-256-19	0.080	1.000	0.147	159.246 ms	3.680 ms
DeepLog-3-1-64-14	0.079	1.000	0.147	142.055 ms	2.938 ms
DeepLog-8-2-128-19	0.079	1.000	0.147	169.714 ms	1.512 ms
DeepLog-3-3-32-8	0.079	1.000	0.147	141.391 ms	1.022 ms
DeepLog-8-1-128-17	0.079	1.000	0.147	140.468 ms	1.234 ms
DeepLog-3-2-32-8	0.079	1.000	0.147	135.844 ms	0.890 ms
DeepLog-8-1-32-20	0.079	1.000	0.147	116.696 ms	1.296 ms
DeepLog-9-1-32-19	0.079	1.000	0.147	124.475 ms	3.125 ms
DeepLog-8-3-32-19	0.079	1.000	0.147	137.352 ms	1.840 ms
DeepLog-8-4-32-20	0.079	1.000	0.147	161.176 ms	2.125 ms
DeepLog-3-3-32-15	0.079	1.000	0.147	141.391 ms	1.036 ms
DeepLog-3-1-64-10	0.079	1.000	0.147	142.055 ms	0.803 ms
DeepLog-3-3-32-13	0.079	1.000	0.147	141.391 ms	1.042 ms
DeepLog-3-2-32-7	0.079	1.000	0.147	135.844 ms	0.889 ms
DeepLog-3-2-32-9	0.079	1.000	0.147	135.844 ms	0.900 ms
DeepLog-8-4-64-20	0.079	1.000	0.147	176.597 ms	5.363 ms
DeepLog-8-2-128-18	0.079	1.000	0.147	169.714 ms	1.519 ms
DeepLog-3-1-32-13	0.079	1.000	0.147	121.034 ms	0.798 ms
DeepLog-3-3-64-12	0.079	1.000	0.147	158.154 ms	1.079 ms
DeepLog-9-1-128-16	0.079	1.000	0.147	132.636 ms	1.273 ms
DeepLog-9-1-64-18	0.079	1.000	0.147	128.810 ms	1.241 ms
DeepLog-3-3-64-19	0.080	1.000	0.147	158.154 ms	1.087 ms
DeepLog-3-2-32-15	0.079	1.000	0.147	135.844 ms	0.926 ms
DeepLog-9-1-128-17	0.079	1.000	0.147	132.636 ms	1.268 ms
DeepLog-8-1-32-18	0.079	1.000	0.147	116.696 ms	1.274 ms
DeepLog-8-4-64-15	0.079	1.000	0.147	176.597 ms	4.993 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-3-3-32-16	0.079	1.000	0.147	141.391 ms	1.048 ms
DeepLog-3-2-32-10	0.079	1.000	0.147	135.844 ms	0.906 ms
DeepLog-3-2-32-13	0.079	1.000	0.147	135.844 ms	0.946 ms
DeepLog-8-1-128-15	0.079	1.000	0.147	140.468 ms	1.197 ms
DeepLog-3-2-32-16	0.079	1.000	0.147	135.844 ms	0.930 ms
DeepLog-8-4-64-19	0.079	1.000	0.147	176.597 ms	5.478 ms
DeepLog-3-1-32-16	0.079	1.000	0.147	121.034 ms	0.809 ms
DeepLog-8-2-64-20	0.079	1.000	0.147	137.269 ms	1.569 ms
DeepLog-3-2-32-11	0.079	1.000	0.147	135.844 ms	0.915 ms
DeepLog-3-3-32-18	0.079	1.000	0.147	141.391 ms	1.022 ms
DeepLog-3-3-64-17	0.080	1.000	0.147	158.154 ms	1.055 ms
DeepLog-8-3-64-19	0.079	1.000	0.147	151.216 ms	1.812 ms
DeepLog-8-4-64-17	0.079	1.000	0.147	176.597 ms	5.043 ms
DeepLog-3-3-32-11	0.079	1.000	0.147	141.391 ms	1.015 ms
DeepLog-3-1-32-20	0.080	1.000	0.147	121.034 ms	0.812 ms
DeepLog-9-1-128-20	0.079	1.000	0.147	132.636 ms	1.267 ms
DeepLog-3-1-64-13	0.079	1.000	0.147	142.055 ms	2.981 ms
DeepLog-8-3-64-20	0.079	1.000	0.147	151.216 ms	1.830 ms
DeepLog-3-1-32-17	0.079	1.000	0.147	121.034 ms	0.806 ms
DeepLog-8-4-64-18	0.079	1.000	0.147	176.597 ms	5.079 ms
DeepLog-8-1-128-19	0.080	1.000	0.147	140.468 ms	1.250 ms
DeepLog-9-3-64-20	0.079	1.000	0.147	158.282 ms	1.798 ms
DeepLog-3-1-32-19	0.080	1.000	0.147	121.034 ms	0.832 ms
DeepLog-9-1-128-15	0.079	1.000	0.147	132.636 ms	1.267 ms
DeepLog-8-2-128-20	0.079	1.000	0.147	169.714 ms	1.516 ms
DeepLog-3-1-64-12	0.079	1.000	0.147	142.055 ms	0.820 ms
DeepLog-3-3-64-16	0.079	1.000	0.147	158.154 ms	1.058 ms
DeepLog-3-3-64-18	0.080	1.000	0.147	158.154 ms	1.058 ms
DeepLog-3-1-32-18	0.080	1.000	0.147	121.034 ms	0.825 ms
DeepLog-3-3-32-14	0.079	1.000	0.147	141.391 ms	1.035 ms
DeepLog-3-1-32-14	0.079	1.000	0.147	121.034 ms	0.801 ms
DeepLog-9-2-32-20	0.080	1.000	0.148	135.313 ms	1.712 ms
DeepLog-9-2-64-18	0.080	1.000	0.148	146.191 ms	1.632 ms
DeepLog-3-1-64-20	0.080	1.000	0.148	142.055 ms	2.989 ms
DeepLog-3-1-64-18	0.080	1.000	0.148	142.055 ms	3.005 ms
DeepLog-9-2-64-16	0.080	1.000	0.148	146.191 ms	1.640 ms
DeepLog-9-2-64-19	0.080	1.000	0.148	146.191 ms	1.657 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-9-2-64-20	0.080	1.000	0.148	146.191 ms	1.729 ms
DeepLog-9-1-256-20	0.080	1.000	0.148	159.246 ms	3.635 ms
DeepLog-3-1-64-19	0.080	1.000	0.148	142.055 ms	3.021 ms
DeepLog-3-2-32-19	0.080	1.000	0.148	135.844 ms	0.966 ms
DeepLog-9-2-64-17	0.080	1.000	0.148	146.191 ms	1.624 ms
DeepLog-3-2-32-20	0.080	1.000	0.148	135.844 ms	0.967 ms
DeepLog-8-1-128-20	0.080	1.000	0.148	140.468 ms	2.978 ms
DeepLog-4-3-32-3	0.082	1.000	0.151	139.900 ms	1.215 ms
DeepLog-4-2-32-5	0.082	1.000	0.151	132.329 ms	1.041 ms
DeepLog-4-2-32-4	0.082	1.000	0.151	132.329 ms	1.048 ms
DeepLog-4-1-32-6	0.082	1.000	0.151	119.678 ms	1.004 ms
DeepLog-4-3-32-5	0.082	1.000	0.151	139.900 ms	1.204 ms
DeepLog-4-1-64-4	0.082	1.000	0.151	128.681 ms	0.721 ms
DeepLog-4-2-32-6	0.082	1.000	0.151	132.329 ms	1.162 ms
DeepLog-4-3-32-4	0.082	1.000	0.151	139.900 ms	1.202 ms
DeepLog-4-2-32-8	0.082	1.000	0.151	132.329 ms	1.167 ms
DeepLog-4-2-32-3	0.082	1.000	0.151	132.329 ms	0.827 ms
DeepLog-4-1-32-3	0.082	1.000	0.151	119.678 ms	0.719 ms
DeepLog-4-1-64-3	0.082	1.000	0.151	128.681 ms	0.711 ms
DeepLog-4-1-64-6	0.082	1.000	0.151	128.681 ms	0.849 ms
DeepLog-4-1-32-4	0.082	1.000	0.151	119.678 ms	0.884 ms
DeepLog-4-1-64-5	0.082	1.000	0.151	128.681 ms	0.727 ms
DeepLog-4-2-32-7	0.082	1.000	0.151	132.329 ms	1.163 ms
DeepLog-4-1-32-5	0.082	1.000	0.151	119.678 ms	0.891 ms
DeepLog-4-1-32-7	0.082	1.000	0.152	119.678 ms	1.016 ms
DeepLog-7-4-64-5	0.083	1.000	0.154	178.839 ms	1.507 ms
DeepLog-7-3-64-3	0.084	1.000	0.154	152.963 ms	1.141 ms
DeepLog-7-4-64-7	0.084	1.000	0.154	178.839 ms	1.529 ms
DeepLog-7-4-64-4	0.083	1.000	0.154	178.839 ms	1.461 ms
DeepLog-7-1-128-3	0.084	1.000	0.154	132.439 ms	0.786 ms
DeepLog-7-4-64-3	0.083	1.000	0.154	178.839 ms	1.265 ms
DeepLog-7-4-64-6	0.084	1.000	0.154	178.839 ms	1.524 ms
DeepLog-7-1-256-3	0.084	1.000	0.154	162.946 ms	0.806 ms
PCA-6-0.000010	0.084	1.000	0.154	0.004 ms	0.051 ms
PCA-6-0.000100	0.084	1.000	0.155	0.004 ms	0.051 ms
DeepLog-4-2-32-9	0.084	1.000	0.155	132.329 ms	1.199 ms
DeepLog-5-1-32-5	0.084	1.000	0.155	119.373 ms	0.786 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-7-5-32-7	0.084	1.000	0.155	171.960 ms	1.771 ms
DeepLog-5-3-32-4	0.084	1.000	0.155	141.431 ms	0.955 ms
DeepLog-7-5-32-6	0.084	1.000	0.155	171.960 ms	1.758 ms
DeepLog-7-2-64-4	0.084	1.000	0.155	137.791 ms	1.151 ms
DeepLog-7-3-64-4	0.084	1.000	0.155	152.963 ms	1.157 ms
DeepLog-5-1-32-4	0.084	1.000	0.155	119.373 ms	0.772 ms
DeepLog-5-1-32-3	0.084	1.000	0.155	119.373 ms	0.755 ms
DeepLog-5-3-32-3	0.084	1.000	0.155	141.431 ms	0.940 ms
DeepLog-5-1-64-4	0.084	1.000	0.155	124.154 ms	0.786 ms
DeepLog-5-2-64-3	0.084	1.000	0.155	141.661 ms	0.930 ms
DeepLog-7-2-64-5	0.084	1.000	0.155	137.791 ms	1.143 ms
DeepLog-7-5-32-5	0.084	1.000	0.155	171.960 ms	1.762 ms
DeepLog-5-1-64-3	0.084	1.000	0.155	124.154 ms	0.771 ms
DeepLog-5-1-128-4	0.084	1.000	0.156	134.074 ms	0.886 ms
DeepLog-5-2-32-4	0.084	1.000	0.156	129.330 ms	0.920 ms
DeepLog-5-1-64-5	0.084	1.000	0.156	124.154 ms	0.799 ms
DeepLog-5-2-32-5	0.085	1.000	0.156	129.330 ms	1.040 ms
DeepLog-5-2-64-8	0.085	1.000	0.156	141.661 ms	1.084 ms
DeepLog-5-2-64-9	0.085	1.000	0.156	141.661 ms	1.146 ms
DeepLog-5-1-128-6	0.085	1.000	0.156	134.074 ms	0.895 ms
DeepLog-5-3-32-7	0.085	1.000	0.156	141.431 ms	1.102 ms
DeepLog-5-2-64-5	0.084	1.000	0.156	141.661 ms	0.957 ms
DeepLog-5-3-32-5	0.084	1.000	0.156	141.431 ms	0.962 ms
DeepLog-5-3-32-8	0.085	1.000	0.156	141.431 ms	1.144 ms
DeepLog-5-1-128-5	0.085	1.000	0.156	134.074 ms	0.889 ms
DeepLog-7-1-64-8	0.084	1.000	0.156	122.034 ms	1.091 ms
DeepLog-5-1-128-3	0.084	1.000	0.156	134.074 ms	0.767 ms
DeepLog-5-1-64-7	0.085	1.000	0.156	124.154 ms	0.897 ms
DeepLog-5-3-32-6	0.084	1.000	0.156	141.431 ms	1.094 ms
DeepLog-5-2-32-3	0.084	1.000	0.156	129.330 ms	0.914 ms
DeepLog-5-1-128-8	0.085	1.000	0.156	134.074 ms	0.962 ms
DeepLog-5-1-32-6	0.084	1.000	0.156	119.373 ms	0.877 ms
DeepLog-5-1-128-7	0.085	1.000	0.156	134.074 ms	0.931 ms
DeepLog-5-1-64-6	0.085	1.000	0.156	124.154 ms	0.895 ms
DeepLog-5-2-64-7	0.084	1.000	0.156	141.661 ms	1.082 ms
DeepLog-5-2-64-4	0.084	1.000	0.156	141.661 ms	0.941 ms
DeepLog-5-2-64-6	0.084	1.000	0.156	141.661 ms	0.957 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-5-2-32-6	0.085	1.000	0.157	129.330 ms	1.077 ms
DeepLog-5-1-128-10	0.085	1.000	0.157	134.074 ms	0.968 ms
DeepLog-7-2-32-5	0.085	1.000	0.157	128.480 ms	1.249 ms
DeepLog-7-1-64-9	0.085	1.000	0.157	122.034 ms	1.094 ms
DeepLog-5-3-32-9	0.085	1.000	0.157	141.431 ms	1.283 ms
DeepLog-7-1-64-11	0.085	1.000	0.157	122.034 ms	1.111 ms
DeepLog-7-1-64-10	0.085	1.000	0.157	122.034 ms	1.118 ms
DeepLog-5-2-32-7	0.085	1.000	0.157	129.330 ms	1.081 ms
DeepLog-5-2-64-10	0.085	1.000	0.157	141.661 ms	1.155 ms
DeepLog-5-2-32-8	0.085	1.000	0.157	129.330 ms	1.100 ms
DeepLog-7-2-64-6	0.085	1.000	0.157	137.791 ms	1.295 ms
DeepLog-7-2-32-4	0.085	1.000	0.157	128.480 ms	1.058 ms
DeepLog-7-3-32-9	0.085	1.000	0.157	147.040 ms	1.560 ms
DeepLog-7-4-64-8	0.085	1.000	0.157	178.839 ms	1.749 ms
DeepLog-7-2-32-3	0.085	1.000	0.157	128.480 ms	1.039 ms
DeepLog-5-1-32-7	0.085	1.000	0.157	119.373 ms	0.890 ms
DeepLog-7-2-32-6	0.085	1.000	0.157	128.480 ms	1.425 ms
DeepLog-5-1-128-9	0.085	1.000	0.157	134.074 ms	0.955 ms
DeepLog-7-1-256-4	0.085	1.000	0.157	162.946 ms	0.923 ms
DeepLog-7-1-256-5	0.085	1.000	0.157	162.946 ms	1.053 ms
DeepLog-7-5-32-8	0.086	1.000	0.158	171.960 ms	2.066 ms
DeepLog-7-5-32-9	0.086	1.000	0.159	171.960 ms	2.339 ms
DeepLog-4-1-64-7	0.086	1.000	0.159	128.681 ms	0.861 ms
DeepLog-4-3-32-7	0.086	1.000	0.159	139.900 ms	1.225 ms
DeepLog-4-3-32-8	0.086	1.000	0.159	139.900 ms	1.229 ms
DeepLog-4-1-64-8	0.086	1.000	0.159	128.681 ms	0.859 ms
DeepLog-4-1-32-9	0.087	1.000	0.159	119.678 ms	1.051 ms
DeepLog-4-1-32-8	0.086	1.000	0.159	119.678 ms	1.019 ms
DeepLog-4-3-32-6	0.086	1.000	0.159	139.900 ms	1.221 ms
DeepLog-7-4-32-20	0.087	1.000	0.160	160.527 ms	2.136 ms
DeepLog-7-5-32-10	0.087	1.000	0.161	171.960 ms	2.371 ms
DeepLog-7-3-64-5	0.088	1.000	0.162	152.963 ms	1.317 ms
DeepLog-7-4-64-9	0.088	1.000	0.162	178.839 ms	1.787 ms
DeepLog-7-3-64-7	0.089	1.000	0.163	152.963 ms	1.491 ms
DeepLog-7-1-256-6	0.089	1.000	0.163	162.946 ms	1.082 ms
DeepLog-7-3-32-12	0.089	1.000	0.163	147.040 ms	1.604 ms
DeepLog-4-3-32-9	0.089	1.000	0.163	139.900 ms	1.308 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-7-2-32-8	0.089	1.000	0.163	128.480 ms	1.456 ms
DeepLog-7-3-64-6	0.089	1.000	0.163	152.963 ms	1.373 ms
DeepLog-7-3-32-10	0.088	1.000	0.163	147.040 ms	1.563 ms
DeepLog-7-3-32-11	0.089	1.000	0.163	147.040 ms	1.596 ms
DeepLog-7-2-32-7	0.089	1.000	0.163	128.480 ms	1.451 ms
DeepLog-7-4-64-11	0.089	1.000	0.163	178.839 ms	1.803 ms
DeepLog-7-1-32-11	0.089	1.000	0.163	117.337 ms	1.197 ms
DeepLog-4-2-32-12	0.089	1.000	0.163	132.329 ms	1.213 ms
DeepLog-7-1-256-7	0.089	1.000	0.163	162.946 ms	1.106 ms
DeepLog-7-1-128-4	0.088	1.000	0.163	132.439 ms	0.888 ms
DeepLog-4-3-32-10	0.089	1.000	0.163	139.900 ms	1.338 ms
DeepLog-4-2-32-13	0.089	1.000	0.163	132.329 ms	3.310 ms
DeepLog-4-2-32-10	0.089	1.000	0.163	132.329 ms	1.199 ms
DeepLog-7-4-64-10	0.089	1.000	0.163	178.839 ms	1.801 ms
DeepLog-7-1-32-10	0.089	1.000	0.163	117.337 ms	1.193 ms
DeepLog-4-2-32-11	0.089	1.000	0.163	132.329 ms	1.202 ms
DeepLog-7-1-32-12	0.089	1.000	0.163	117.337 ms	1.209 ms
DeepLog-7-3-32-13	0.089	1.000	0.163	147.040 ms	1.621 ms
DeepLog-7-1-128-5	0.089	1.000	0.163	132.439 ms	1.019 ms
DeepLog-4-2-32-14	0.089	1.000	0.163	132.329 ms	3.331 ms
DeepLog-7-1-128-6	0.089	1.000	0.163	132.439 ms	1.146 ms
DeepLog-7-4-64-13	0.089	1.000	0.164	178.839 ms	1.882 ms
DeepLog-7-2-64-7	0.089	1.000	0.164	137.791 ms	1.369 ms
DeepLog-4-3-32-15	0.089	1.000	0.164	139.900 ms	1.324 ms
DeepLog-4-1-32-11	0.089	1.000	0.164	119.678 ms	1.113 ms
DeepLog-4-1-64-15	0.089	1.000	0.164	128.681 ms	0.917 ms
DeepLog-4-1-64-9	0.089	1.000	0.164	128.681 ms	0.887 ms
DeepLog-4-1-32-10	0.089	1.000	0.164	119.678 ms	1.121 ms
DeepLog-4-3-32-18	0.089	1.000	0.164	139.900 ms	1.361 ms
DeepLog-4-2-32-18	0.089	1.000	0.164	132.329 ms	3.352 ms
DeepLog-7-3-64-9	0.090	1.000	0.164	152.963 ms	1.693 ms
DeepLog-7-3-64-8	0.089	1.000	0.164	152.963 ms	1.679 ms
DeepLog-4-1-64-12	0.089	1.000	0.164	128.681 ms	0.892 ms
DeepLog-4-3-32-20	0.089	1.000	0.164	139.900 ms	3.730 ms
DeepLog-4-3-32-11	0.089	1.000	0.164	139.900 ms	1.327 ms
DeepLog-4-2-32-17	0.089	1.000	0.164	132.329 ms	3.328 ms
DeepLog-4-3-32-14	0.089	1.000	0.164	139.900 ms	1.332 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-4-1-32-12	0.089	1.000	0.164	119.678 ms	1.114 ms
DeepLog-4-1-64-11	0.089	1.000	0.164	128.681 ms	0.885 ms
DeepLog-4-1-64-14	0.089	1.000	0.164	128.681 ms	0.893 ms
DeepLog-4-3-32-16	0.089	1.000	0.164	139.900 ms	1.337 ms
DeepLog-7-2-32-9	0.089	1.000	0.164	128.480 ms	1.516 ms
DeepLog-4-2-32-15	0.089	1.000	0.164	132.329 ms	3.269 ms
DeepLog-4-1-32-15	0.090	1.000	0.164	119.678 ms	2.847 ms
DeepLog-4-3-32-12	0.089	1.000	0.164	139.900 ms	1.336 ms
DeepLog-4-3-32-17	0.089	1.000	0.164	139.900 ms	1.381 ms
DeepLog-7-1-32-15	0.089	1.000	0.164	117.337 ms	1.256 ms
DeepLog-7-2-32-11	0.090	1.000	0.164	128.480 ms	1.529 ms
DeepLog-4-1-32-13	0.089	1.000	0.164	119.678 ms	1.114 ms
DeepLog-7-1-32-13	0.089	1.000	0.164	117.337 ms	1.197 ms
DeepLog-4-1-64-10	0.089	1.000	0.164	128.681 ms	0.891 ms
DeepLog-4-2-32-19	0.089	1.000	0.164	132.329 ms	3.314 ms
DeepLog-4-1-64-16	0.090	1.000	0.164	128.681 ms	0.928 ms
DeepLog-7-1-32-16	0.089	1.000	0.164	117.337 ms	1.266 ms
DeepLog-7-1-32-14	0.089	1.000	0.164	117.337 ms	1.256 ms
DeepLog-4-1-32-16	0.090	1.000	0.164	119.678 ms	2.795 ms
DeepLog-4-3-32-19	0.089	1.000	0.164	139.900 ms	3.763 ms
DeepLog-7-4-64-12	0.089	1.000	0.164	178.839 ms	1.865 ms
DeepLog-4-1-32-14	0.090	1.000	0.164	119.678 ms	2.846 ms
DeepLog-4-2-32-20	0.090	1.000	0.164	132.329 ms	3.376 ms
DeepLog-4-3-32-13	0.089	1.000	0.164	139.900 ms	1.340 ms
DeepLog-4-2-32-16	0.089	1.000	0.164	132.329 ms	3.326 ms
DeepLog-4-1-64-13	0.089	1.000	0.164	128.681 ms	0.894 ms
DeepLog-7-2-32-10	0.090	1.000	0.164	128.480 ms	1.525 ms
DeepLog-4-1-64-19	0.090	1.000	0.165	128.681 ms	0.937 ms
DeepLog-5-3-32-12	0.090	1.000	0.165	141.431 ms	1.296 ms
DeepLog-7-2-32-15	0.090	1.000	0.165	128.480 ms	1.572 ms
DeepLog-5-3-32-13	0.090	1.000	0.165	141.431 ms	1.305 ms
DeepLog-7-2-32-14	0.090	1.000	0.165	128.480 ms	1.526 ms
DeepLog-7-3-64-11	0.090	1.000	0.165	152.963 ms	1.866 ms
DeepLog-5-1-32-8	0.090	1.000	0.165	119.373 ms	0.891 ms
DeepLog-7-2-32-16	0.090	1.000	0.165	128.480 ms	1.608 ms
DeepLog-5-3-32-16	0.090	1.000	0.165	141.431 ms	1.308 ms
DeepLog-5-3-32-17	0.090	1.000	0.165	141.431 ms	1.315 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-5-3-32-10	0.090	1.000	0.165	141.431 ms	1.294 ms
DeepLog-5-1-64-9	0.090	1.000	0.165	124.154 ms	0.957 ms
DeepLog-5-3-32-14	0.090	1.000	0.165	141.431 ms	1.299 ms
DeepLog-5-1-32-12	0.090	1.000	0.165	119.373 ms	0.900 ms
DeepLog-5-3-32-15	0.090	1.000	0.165	141.431 ms	1.300 ms
DeepLog-5-2-32-9	0.090	1.000	0.165	129.330 ms	1.119 ms
DeepLog-5-1-32-9	0.090	1.000	0.165	119.373 ms	0.899 ms
DeepLog-5-1-32-13	0.090	1.000	0.165	119.373 ms	0.909 ms
DeepLog-7-3-64-10	0.090	1.000	0.165	152.963 ms	1.841 ms
DeepLog-7-3-64-12	0.090	1.000	0.165	152.963 ms	1.901 ms
DeepLog-7-4-64-16	0.090	1.000	0.165	178.839 ms	2.122 ms
DeepLog-5-1-128-11	0.090	1.000	0.165	134.074 ms	0.974 ms
DeepLog-5-1-32-11	0.090	1.000	0.165	119.373 ms	0.904 ms
DeepLog-4-1-32-18	0.090	1.000	0.165	119.678 ms	3.109 ms
DeepLog-7-2-32-13	0.090	1.000	0.165	128.480 ms	1.531 ms
DeepLog-7-4-64-14	0.090	1.000	0.165	178.839 ms	1.860 ms
DeepLog-5-3-32-11	0.090	1.000	0.165	141.431 ms	1.294 ms
DeepLog-4-1-64-17	0.090	1.000	0.165	128.681 ms	0.934 ms
DeepLog-7-4-64-15	0.090	1.000	0.165	178.839 ms	2.078 ms
DeepLog-5-1-32-10	0.090	1.000	0.165	119.373 ms	0.902 ms
DeepLog-4-1-32-19	0.090	1.000	0.165	119.678 ms	3.134 ms
DeepLog-4-1-32-20	0.090	1.000	0.165	119.678 ms	3.109 ms
DeepLog-7-2-32-12	0.090	1.000	0.165	128.480 ms	1.518 ms
DeepLog-7-1-128-7	0.090	1.000	0.165	132.439 ms	1.215 ms
DeepLog-5-1-64-8	0.090	1.000	0.165	124.154 ms	0.906 ms
DeepLog-4-1-64-18	0.090	1.000	0.165	128.681 ms	0.940 ms
DeepLog-4-1-32-17	0.090	1.000	0.165	119.678 ms	3.050 ms
DeepLog-7-1-256-8	0.090	1.000	0.165	162.946 ms	1.120 ms
DeepLog-4-1-64-20	0.090	1.000	0.165	128.681 ms	0.942 ms
DeepLog-5-2-32-12	0.090	1.000	0.166	129.330 ms	1.168 ms
DeepLog-7-2-128-14	0.091	1.000	0.166	159.838 ms	1.575 ms
DeepLog-7-1-128-9	0.090	1.000	0.166	132.439 ms	1.247 ms
DeepLog-7-1-128-12	0.091	1.000	0.166	132.439 ms	1.280 ms
DeepLog-7-1-256-11	0.090	1.000	0.166	162.946 ms	1.149 ms
DeepLog-7-1-256-9	0.090	1.000	0.166	162.946 ms	1.134 ms
DeepLog-7-5-32-11	0.091	1.000	0.166	171.960 ms	2.376 ms
DeepLog-5-1-128-13	0.090	1.000	0.166	134.074 ms	0.980 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-7-1-128-11	0.091	1.000	0.166	132.439 ms	1.275 ms
DeepLog-7-2-64-8	0.091	1.000	0.166	137.791 ms	1.500 ms
DeepLog-7-1-64-13	0.090	1.000	0.166	122.034 ms	1.251 ms
DeepLog-7-3-32-15	0.091	1.000	0.166	147.040 ms	1.669 ms
DeepLog-5-2-32-10	0.090	1.000	0.166	129.330 ms	1.170 ms
DeepLog-7-1-256-12	0.090	1.000	0.166	162.946 ms	1.149 ms
DeepLog-7-3-32-16	0.091	1.000	0.166	147.040 ms	1.642 ms
DeepLog-7-4-64-17	0.090	1.000	0.166	178.839 ms	2.147 ms
DeepLog-7-1-64-12	0.090	1.000	0.166	122.034 ms	1.226 ms
DeepLog-7-1-128-8	0.090	1.000	0.166	132.439 ms	1.226 ms
DeepLog-7-2-32-18	0.091	1.000	0.166	128.480 ms	1.591 ms
DeepLog-7-1-256-14	0.091	1.000	0.166	162.946 ms	1.153 ms
DeepLog-5-3-32-18	0.090	1.000	0.166	141.431 ms	1.375 ms
DeepLog-7-1-256-10	0.090	1.000	0.166	162.946 ms	1.132 ms
DeepLog-7-2-64-9	0.091	1.000	0.166	137.791 ms	1.506 ms
DeepLog-7-1-64-15	0.091	1.000	0.166	122.034 ms	1.264 ms
DeepLog-7-3-32-14	0.091	1.000	0.166	147.040 ms	1.645 ms
DeepLog-7-1-128-10	0.091	1.000	0.166	132.439 ms	1.275 ms
DeepLog-5-2-32-11	0.090	1.000	0.166	129.330 ms	1.165 ms
DeepLog-7-4-64-18	0.091	1.000	0.166	178.839 ms	2.209 ms
DeepLog-5-2-32-13	0.090	1.000	0.166	129.330 ms	1.185 ms
DeepLog-7-1-64-14	0.091	1.000	0.166	122.034 ms	1.268 ms
DeepLog-5-1-128-12	0.090	1.000	0.166	134.074 ms	0.971 ms
DeepLog-7-2-32-17	0.090	1.000	0.166	128.480 ms	1.585 ms
DeepLog-7-1-256-13	0.091	1.000	0.166	162.946 ms	1.144 ms
DeepLog-7-2-128-16	0.091	1.000	0.167	159.838 ms	1.599 ms
DeepLog-7-5-32-13	0.091	1.000	0.167	171.960 ms	2.516 ms
DeepLog-7-2-128-15	0.091	1.000	0.167	159.838 ms	1.557 ms
DeepLog-7-3-32-18	0.091	1.000	0.167	147.040 ms	1.638 ms
DeepLog-7-3-32-19	0.091	1.000	0.167	147.040 ms	1.688 ms
DeepLog-7-5-32-12	0.091	1.000	0.167	171.960 ms	2.416 ms
DeepLog-7-1-128-14	0.091	1.000	0.167	132.439 ms	1.288 ms
DeepLog-7-1-32-17	0.091	1.000	0.167	117.337 ms	1.282 ms
DeepLog-7-2-128-17	0.091	1.000	0.167	159.838 ms	1.613 ms
DeepLog-7-1-128-13	0.091	1.000	0.167	132.439 ms	1.282 ms
DeepLog-7-2-64-11	0.091	1.000	0.167	137.791 ms	3.666 ms
DeepLog-7-2-64-10	0.091	1.000	0.167	137.791 ms	3.569 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-7-3-32-17	0.091	1.000	0.167	147.040 ms	1.638 ms
DeepLog-7-2-64-12	0.092	1.000	0.168	137.791 ms	3.668 ms
DeepLog-7-2-64-15	0.092	1.000	0.168	137.791 ms	3.634 ms
DeepLog-7-1-256-15	0.092	1.000	0.168	162.946 ms	1.200 ms
DeepLog-7-4-64-19	0.092	1.000	0.168	178.839 ms	2.225 ms
DeepLog-7-2-32-20	0.092	1.000	0.168	128.480 ms	1.682 ms
DeepLog-7-2-32-19	0.092	1.000	0.168	128.480 ms	1.635 ms
DeepLog-7-2-64-13	0.092	1.000	0.168	137.791 ms	3.700 ms
DeepLog-7-4-64-20	0.092	1.000	0.168	178.839 ms	2.251 ms
DeepLog-7-2-64-14	0.092	1.000	0.168	137.791 ms	3.730 ms
DeepLog-7-2-64-16	0.092	1.000	0.168	137.791 ms	3.658 ms
DeepLog-7-1-256-17	0.093	1.000	0.169	162.946 ms	1.204 ms
DeepLog-7-1-32-18	0.092	1.000	0.169	117.337 ms	1.353 ms
DeepLog-7-2-64-17	0.092	1.000	0.169	137.791 ms	3.687 ms
DeepLog-7-3-64-13	0.092	1.000	0.169	152.963 ms	1.927 ms
DeepLog-7-1-256-16	0.092	1.000	0.169	162.946 ms	1.196 ms
DeepLog-7-2-128-18	0.092	1.000	0.169	159.838 ms	3.763 ms
DeepLog-7-1-64-16	0.092	1.000	0.169	122.034 ms	1.296 ms
DeepLog-7-2-64-18	0.092	1.000	0.169	137.791 ms	3.722 ms
DeepLog-7-2-64-19	0.092	1.000	0.169	137.791 ms	3.795 ms
DeepLog-7-2-128-19	0.092	1.000	0.169	159.838 ms	3.761 ms
DeepLog-7-2-64-20	0.092	1.000	0.169	137.791 ms	3.757 ms
DeepLog-7-3-32-20	0.093	1.000	0.170	147.040 ms	1.817 ms
DeepLog-7-5-32-15	0.093	1.000	0.170	171.960 ms	2.551 ms
DeepLog-7-1-128-15	0.093	1.000	0.170	132.439 ms	1.350 ms
DeepLog-7-1-64-19	0.093	1.000	0.170	122.034 ms	3.064 ms
DeepLog-7-1-32-20	0.093	1.000	0.170	117.337 ms	1.383 ms
DeepLog-7-1-64-18	0.093	1.000	0.170	122.034 ms	3.038 ms
DeepLog-7-1-256-18	0.093	1.000	0.170	162.946 ms	1.235 ms
DeepLog-7-1-128-17	0.093	1.000	0.170	132.439 ms	1.362 ms
DeepLog-7-5-32-16	0.093	1.000	0.170	171.960 ms	2.497 ms
DeepLog-7-1-256-19	0.093	1.000	0.170	162.946 ms	1.266 ms
DeepLog-7-1-32-19	0.093	1.000	0.170	117.337 ms	1.372 ms
DeepLog-7-1-64-20	0.093	1.000	0.170	122.034 ms	3.062 ms
DeepLog-7-3-64-14	0.093	1.000	0.170	152.963 ms	1.936 ms
DeepLog-7-5-32-14	0.093	1.000	0.170	171.960 ms	2.551 ms
DeepLog-7-3-64-15	0.093	1.000	0.170	152.963 ms	1.934 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-7-1-128-16	0.093	1.000	0.170	132.439 ms	1.355 ms
DeepLog-7-1-64-17	0.093	1.000	0.170	122.034 ms	3.027 ms
DeepLog-6-3-32-3	0.093	1.000	0.171	138.385 ms	1.045 ms
DeepLog-7-3-64-20	0.094	1.000	0.171	152.963 ms	1.998 ms
DeepLog-7-5-32-20	0.094	1.000	0.171	171.960 ms	2.575 ms
DeepLog-6-2-64-3	0.093	1.000	0.171	138.691 ms	0.823 ms
DeepLog-7-1-256-20	0.093	1.000	0.171	162.946 ms	1.267 ms
DeepLog-7-3-64-19	0.094	1.000	0.171	152.963 ms	2.000 ms
DeepLog-7-5-32-18	0.093	1.000	0.171	171.960 ms	2.558 ms
DeepLog-6-1-32-5	0.094	1.000	0.171	117.614 ms	0.788 ms
DeepLog-7-3-64-16	0.093	1.000	0.171	152.963 ms	1.934 ms
DeepLog-6-3-32-4	0.094	1.000	0.171	138.385 ms	1.197 ms
DeepLog-6-1-64-3	0.094	1.000	0.171	122.782 ms	0.672 ms
DeepLog-7-3-64-17	0.093	1.000	0.171	152.963 ms	1.980 ms
DeepLog-7-5-32-17	0.093	1.000	0.171	171.960 ms	2.553 ms
DeepLog-6-2-64-5	0.093	1.000	0.171	138.691 ms	0.922 ms
DeepLog-7-1-128-19	0.093	1.000	0.171	132.439 ms	3.241 ms
DeepLog-6-3-64-4	0.094	1.000	0.171	152.999 ms	1.013 ms
DeepLog-6-1-128-4	0.094	1.000	0.171	133.033 ms	0.694 ms
DeepLog-7-5-32-19	0.094	1.000	0.171	171.960 ms	2.579 ms
DeepLog-7-1-64-21	0.094	1.000	0.171	122.034 ms	3.090 ms
DeepLog-6-3-64-3	0.093	1.000	0.171	152.999 ms	0.929 ms
DeepLog-6-1-64-4	0.094	1.000	0.171	122.782 ms	0.749 ms
DeepLog-7-3-64-18	0.093	1.000	0.171	152.963 ms	1.983 ms
DeepLog-6-3-32-5	0.094	1.000	0.171	138.385 ms	1.224 ms
DeepLog-6-1-32-4	0.094	1.000	0.171	117.614 ms	0.785 ms
DeepLog-6-1-32-3	0.093	1.000	0.171	117.614 ms	0.987 ms
DeepLog-6-2-64-4	0.093	1.000	0.171	138.691 ms	0.917 ms
DeepLog-6-1-256-3	0.093	1.000	0.171	154.270 ms	0.654 ms
DeepLog-7-1-128-20	0.094	1.000	0.171	132.439 ms	3.230 ms
DeepLog-6-1-128-3	0.093	1.000	0.171	133.033 ms	0.667 ms
DeepLog-7-1-128-18	0.093	1.000	0.171	132.439 ms	3.272 ms
DeepLog-6-2-32-3	0.093	1.000	0.171	129.500 ms	0.787 ms
DeepLog-7-2-128-20	0.094	1.000	0.171	159.838 ms	3.820 ms
DeepLog-6-3-32-7	0.094	1.000	0.172	138.385 ms	1.246 ms
DeepLog-6-3-32-6	0.094	1.000	0.172	138.385 ms	1.246 ms
DeepLog-7-1-64-25	0.094	1.000	0.172	122.034 ms	3.077 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-7-1-64-23	0.094	1.000	0.172	122.034 ms	3.088 ms
DeepLog-6-3-32-8	0.094	1.000	0.172	138.385 ms	1.269 ms
DeepLog-7-1-64-26	0.094	1.000	0.172	122.034 ms	3.079 ms
DeepLog-6-2-128-4	0.094	1.000	0.172	157.070 ms	0.867 ms
DeepLog-6-3-64-5	0.094	1.000	0.172	152.999 ms	1.198 ms
DeepLog-6-2-128-3	0.094	1.000	0.172	157.070 ms	0.839 ms
DeepLog-7-1-64-24	0.094	1.000	0.172	122.034 ms	3.136 ms
DeepLog-7-1-64-22	0.094	1.000	0.172	122.034 ms	3.098 ms
DeepLog-6-2-128-5	0.094	1.000	0.172	157.070 ms	0.980 ms
DeepLog-5-2-32-14	0.094	1.000	0.173	129.330 ms	1.408 ms
DeepLog-5-2-32-18	0.095	1.000	0.173	129.330 ms	1.427 ms
DeepLog-7-1-64-28	0.095	1.000	0.173	122.034 ms	3.079 ms
DeepLog-7-1-64-27	0.094	1.000	0.173	122.034 ms	3.075 ms
DeepLog-5-2-32-17	0.095	1.000	0.173	129.330 ms	1.417 ms
DeepLog-5-2-32-15	0.095	1.000	0.173	129.330 ms	1.437 ms
DeepLog-5-2-32-16	0.095	1.000	0.173	129.330 ms	1.436 ms
DeepLog-7-1-64-29	0.095	1.000	0.174	122.034 ms	3.086 ms
DeepLog-5-3-32-19	0.096	1.000	0.175	141.431 ms	1.752 ms
DeepLog-6-3-32-9	0.096	1.000	0.175	138.385 ms	1.345 ms
DeepLog-5-3-32-20	0.096	1.000	0.175	141.431 ms	1.738 ms
DeepLog-7-1-64-30	0.098	1.000	0.178	122.034 ms	4.432 ms
DeepLog-6-3-32-10	0.098	1.000	0.179	138.385 ms	1.638 ms
DeepLog-6-3-32-11	0.099	1.000	0.180	138.385 ms	1.646 ms
DeepLog-6-3-32-12	0.099	1.000	0.180	138.385 ms	4.549 ms
DeepLog-6-3-32-13	0.103	1.000	0.187	138.385 ms	4.546 ms
DeepLog-7-1-64-31	0.104	1.000	0.188	122.034 ms	4.778 ms
DeepLog-6-3-32-14	0.105	1.000	0.191	138.385 ms	4.639 ms
DeepLog-6-3-32-16	0.106	1.000	0.191	138.385 ms	4.655 ms
DeepLog-6-3-32-18	0.106	1.000	0.191	138.385 ms	4.717 ms
DeepLog-6-3-32-17	0.106	1.000	0.191	138.385 ms	4.708 ms
DeepLog-6-3-32-19	0.106	1.000	0.191	138.385 ms	4.782 ms
DeepLog-5-2-64-11	0.105	1.000	0.191	141.661 ms	1.390 ms
DeepLog-6-3-32-15	0.105	1.000	0.191	138.385 ms	4.639 ms
DeepLog-5-2-64-12	0.105	1.000	0.191	141.661 ms	1.396 ms
DeepLog-7-1-64-32	0.108	1.000	0.195	122.034 ms	4.866 ms
DeepLog-7-1-64-33	0.108	1.000	0.195	122.034 ms	4.857 ms
DeepLog-7-1-64-34	0.109	1.000	0.196	122.034 ms	4.983 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-7-1-64-35	0.109	1.000	0.197	122.034 ms	4.990 ms
DeepLog-7-1-64-36	0.112	1.000	0.201	122.034 ms	5.422 ms
DeepLog-7-1-64-38	0.113	1.000	0.202	122.034 ms	5.477 ms
DeepLog-6-1-256-4	0.113	1.000	0.202	154.270 ms	0.730 ms
DeepLog-7-1-64-37	0.112	1.000	0.202	122.034 ms	5.386 ms
DeepLog-6-2-64-6	0.113	1.000	0.203	138.691 ms	0.999 ms
DeepLog-6-2-64-8	0.113	1.000	0.203	138.691 ms	1.021 ms
DeepLog-5-1-32-14	0.113	1.000	0.203	119.373 ms	1.099 ms
DeepLog-6-1-256-5	0.113	1.000	0.203	154.270 ms	0.753 ms
DeepLog-6-1-32-6	0.113	1.000	0.203	117.614 ms	0.886 ms
DeepLog-5-1-32-15	0.113	1.000	0.203	119.373 ms	1.102 ms
DeepLog-5-1-64-10	0.113	1.000	0.203	124.154 ms	1.140 ms
DeepLog-6-1-64-5	0.113	1.000	0.203	122.782 ms	0.840 ms
DeepLog-6-2-64-7	0.113	1.000	0.203	138.691 ms	1.012 ms
DeepLog-5-2-64-13	0.113	1.000	0.203	141.661 ms	1.389 ms
DeepLog-5-1-64-11	0.113	1.000	0.203	124.154 ms	1.141 ms
DeepLog-5-1-32-17	0.114	1.000	0.204	119.373 ms	1.122 ms
DeepLog-5-1-32-18	0.114	1.000	0.204	119.373 ms	1.126 ms
DeepLog-5-2-64-16	0.114	1.000	0.204	141.661 ms	1.388 ms
DeepLog-6-1-32-7	0.114	1.000	0.204	117.614 ms	0.893 ms
DeepLog-5-1-32-16	0.114	1.000	0.204	119.373 ms	1.115 ms
DeepLog-6-1-128-7	0.114	1.000	0.204	133.033 ms	0.914 ms
DeepLog-6-1-256-6	0.113	1.000	0.204	154.270 ms	0.876 ms
DeepLog-6-1-32-8	0.114	1.000	0.204	117.614 ms	0.921 ms
DeepLog-5-1-64-13	0.113	1.000	0.204	124.154 ms	1.153 ms
DeepLog-6-1-128-6	0.113	1.000	0.204	133.033 ms	0.785 ms
DeepLog-6-1-256-7	0.114	1.000	0.204	154.270 ms	0.885 ms
DeepLog-5-1-128-14	0.114	1.000	0.204	134.074 ms	1.172 ms
DeepLog-6-3-64-6	0.113	1.000	0.204	152.999 ms	1.228 ms
DeepLog-5-1-64-14	0.114	1.000	0.204	124.154 ms	1.214 ms
DeepLog-6-1-64-7	0.114	1.000	0.204	122.782 ms	1.005 ms
DeepLog-5-2-64-14	0.114	1.000	0.204	141.661 ms	1.401 ms
DeepLog-6-1-64-6	0.113	1.000	0.204	122.782 ms	0.972 ms
DeepLog-5-2-64-15	0.114	1.000	0.204	141.661 ms	1.377 ms
DeepLog-5-1-64-12	0.113	1.000	0.204	124.154 ms	1.129 ms
DeepLog-6-1-128-5	0.113	1.000	0.204	133.033 ms	0.761 ms
DeepLog-5-2-64-17	0.114	1.000	0.204	141.661 ms	1.414 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-6-3-64-7	0.114	1.000	0.205	152.999 ms	1.274 ms
DeepLog-5-2-64-18	0.114	1.000	0.205	141.661 ms	1.418 ms
DeepLog-5-1-128-17	0.114	1.000	0.205	134.074 ms	1.210 ms
DeepLog-6-1-128-9	0.114	1.000	0.205	133.033 ms	0.931 ms
DeepLog-5-1-128-15	0.114	1.000	0.205	134.074 ms	1.185 ms
DeepLog-5-2-64-19	0.114	1.000	0.205	141.661 ms	1.421 ms
DeepLog-5-1-64-18	0.115	1.000	0.205	124.154 ms	1.249 ms
DeepLog-5-1-32-19	0.114	1.000	0.205	119.373 ms	1.127 ms
DeepLog-6-1-64-9	0.114	1.000	0.205	122.782 ms	1.037 ms
DeepLog-6-2-128-6	0.114	1.000	0.205	157.070 ms	1.090 ms
DeepLog-6-1-128-8	0.114	1.000	0.205	133.033 ms	0.915 ms
DeepLog-5-1-128-18	0.115	1.000	0.205	134.074 ms	1.225 ms
DeepLog-5-1-32-20	0.114	1.000	0.205	119.373 ms	1.121 ms
DeepLog-5-1-64-17	0.114	1.000	0.205	124.154 ms	1.234 ms
DeepLog-5-1-128-16	0.114	1.000	0.205	134.074 ms	1.191 ms
DeepLog-6-1-256-8	0.114	1.000	0.205	154.270 ms	0.876 ms
DeepLog-6-1-32-9	0.114	1.000	0.205	117.614 ms	0.923 ms
DeepLog-6-2-128-7	0.114	1.000	0.205	157.070 ms	1.105 ms
DeepLog-5-1-64-15	0.114	1.000	0.205	124.154 ms	1.228 ms
DeepLog-6-3-64-9	0.114	1.000	0.205	152.999 ms	1.308 ms
DeepLog-6-1-64-8	0.114	1.000	0.205	122.782 ms	1.018 ms
DeepLog-6-1-256-9	0.114	1.000	0.205	154.270 ms	0.923 ms
DeepLog-5-1-64-16	0.114	1.000	0.205	124.154 ms	1.242 ms
DeepLog-6-3-64-8	0.114	1.000	0.205	152.999 ms	1.281 ms
DeepLog-6-1-256-11	0.115	1.000	0.206	154.270 ms	0.950 ms
DeepLog-6-1-32-16	0.115	1.000	0.206	117.614 ms	0.970 ms
DeepLog-5-2-32-20	0.115	1.000	0.206	129.330 ms	1.448 ms
DeepLog-6-1-32-14	0.115	1.000	0.206	117.614 ms	0.970 ms
DeepLog-6-1-128-11	0.115	1.000	0.206	133.033 ms	3.017 ms
DeepLog-6-1-32-13	0.115	1.000	0.206	117.614 ms	0.962 ms
DeepLog-5-2-64-20	0.115	1.000	0.206	141.661 ms	1.444 ms
DeepLog-5-1-128-20	0.115	1.000	0.206	134.074 ms	1.227 ms
DeepLog-6-1-256-10	0.115	1.000	0.206	154.270 ms	0.946 ms
DeepLog-6-1-256-12	0.115	1.000	0.206	154.270 ms	0.965 ms
DeepLog-6-3-64-11	0.115	1.000	0.206	152.999 ms	1.326 ms
DeepLog-6-1-32-10	0.115	1.000	0.206	117.614 ms	0.936 ms
DeepLog-5-1-128-19	0.115	1.000	0.206	134.074 ms	1.205 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-5-2-32-19	0.115	1.000	0.206	129.330 ms	1.442 ms
DeepLog-5-1-64-19	0.115	1.000	0.206	124.154 ms	1.248 ms
DeepLog-6-3-64-14	0.115	1.000	0.206	152.999 ms	1.358 ms
DeepLog-6-1-32-12	0.115	1.000	0.206	117.614 ms	0.956 ms
DeepLog-6-2-32-4	0.115	1.000	0.206	129.500 ms	1.072 ms
DeepLog-6-1-128-10	0.115	1.000	0.206	133.033 ms	0.948 ms
DeepLog-6-3-64-13	0.115	1.000	0.206	152.999 ms	1.343 ms
DeepLog-6-1-32-17	0.115	1.000	0.206	117.614 ms	0.965 ms
DeepLog-6-2-32-5	0.115	1.000	0.206	129.500 ms	1.121 ms
DeepLog-6-3-64-10	0.115	1.000	0.206	152.999 ms	1.329 ms
DeepLog-6-1-32-11	0.115	1.000	0.206	117.614 ms	0.941 ms
DeepLog-5-1-64-20	0.115	1.000	0.206	124.154 ms	1.254 ms
DeepLog-6-3-64-12	0.115	1.000	0.206	152.999 ms	1.342 ms
DeepLog-6-1-32-15	0.115	1.000	0.206	117.614 ms	0.973 ms
DeepLog-6-1-128-18	0.116	1.000	0.207	133.033 ms	2.985 ms
DeepLog-6-1-128-13	0.115	1.000	0.207	133.033 ms	3.017 ms
DeepLog-6-2-32-6	0.116	1.000	0.207	129.500 ms	1.269 ms
DeepLog-6-3-64-15	0.115	1.000	0.207	152.999 ms	1.369 ms
DeepLog-6-1-32-18	0.115	1.000	0.207	117.614 ms	0.971 ms
DeepLog-6-1-128-12	0.115	1.000	0.207	133.033 ms	3.001 ms
DeepLog-6-1-128-15	0.115	1.000	0.207	133.033 ms	2.957 ms
DeepLog-6-1-256-13	0.116	1.000	0.207	154.270 ms	3.046 ms
DeepLog-6-1-256-14	0.116	1.000	0.207	154.270 ms	3.042 ms
DeepLog-6-1-128-14	0.115	1.000	0.207	133.033 ms	3.048 ms
DeepLog-6-1-128-17	0.116	1.000	0.207	133.033 ms	2.995 ms
DeepLog-6-1-128-16	0.116	1.000	0.207	133.033 ms	2.977 ms
DeepLog-6-1-256-15	0.116	1.000	0.208	154.270 ms	3.074 ms
DeepLog-6-1-256-18	0.116	1.000	0.208	154.270 ms	3.019 ms
DeepLog-6-1-128-20	0.116	1.000	0.208	133.033 ms	3.062 ms
DeepLog-6-1-256-19	0.116	1.000	0.208	154.270 ms	2.998 ms
DeepLog-6-1-256-20	0.116	1.000	0.208	154.270 ms	3.022 ms
DeepLog-6-1-256-16	0.116	1.000	0.208	154.270 ms	2.997 ms
DeepLog-6-2-32-7	0.116	1.000	0.208	129.500 ms	1.323 ms
DeepLog-6-2-128-8	0.116	1.000	0.208	157.070 ms	1.112 ms
DeepLog-6-1-256-17	0.116	1.000	0.208	154.270 ms	3.013 ms
DeepLog-6-1-128-19	0.116	1.000	0.208	133.033 ms	3.046 ms
DeepLog-6-2-128-10	0.117	1.000	0.209	157.070 ms	1.192 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-6-2-128-9	0.117	1.000	0.209	157.070 ms	1.159 ms
DeepLog-6-1-64-10	0.117	1.000	0.209	122.782 ms	1.153 ms
DeepLog-6-2-128-13	0.118	1.000	0.210	157.070 ms	1.222 ms
DeepLog-6-2-128-12	0.117	1.000	0.210	157.070 ms	1.205 ms
DeepLog-6-2-128-11	0.117	1.000	0.210	157.070 ms	1.199 ms
DeepLog-6-2-128-19	0.118	1.000	0.211	157.070 ms	1.265 ms
DeepLog-6-2-128-17	0.118	1.000	0.211	157.070 ms	1.259 ms
DeepLog-6-2-128-18	0.118	1.000	0.211	157.070 ms	1.266 ms
DeepLog-6-2-128-20	0.118	1.000	0.211	157.070 ms	1.268 ms
DeepLog-6-2-128-14	0.118	1.000	0.211	157.070 ms	1.238 ms
DeepLog-6-2-128-15	0.118	1.000	0.211	157.070 ms	1.242 ms
DeepLog-6-2-128-16	0.118	1.000	0.211	157.070 ms	1.251 ms
DeepLog-6-2-32-8	0.119	1.000	0.212	129.500 ms	1.344 ms
DeepLog-6-1-64-11	0.120	1.000	0.213	122.782 ms	1.189 ms
DeepLog-6-1-64-12	0.120	1.000	0.214	122.782 ms	1.203 ms
DeepLog-6-1-64-13	0.120	1.000	0.214	122.782 ms	1.199 ms
DeepLog-6-1-64-14	0.120	1.000	0.215	122.782 ms	3.253 ms
DeepLog-6-1-64-15	0.120	1.000	0.215	122.782 ms	3.261 ms
DeepLog-6-2-64-10	0.122	1.000	0.217	138.691 ms	1.279 ms
DeepLog-6-2-64-9	0.122	1.000	0.217	138.691 ms	1.296 ms
DeepLog-6-2-64-12	0.123	1.000	0.218	138.691 ms	1.318 ms
DeepLog-6-2-64-11	0.122	1.000	0.218	138.691 ms	1.323 ms
DeepLog-6-1-64-16	0.123	1.000	0.220	122.782 ms	3.251 ms
DeepLog-6-2-32-11	0.126	1.000	0.223	129.500 ms	1.392 ms
DeepLog-6-2-32-10	0.125	1.000	0.223	129.500 ms	1.361 ms
DeepLog-6-2-64-13	0.126	1.000	0.223	138.691 ms	1.384 ms
DeepLog-6-2-32-9	0.125	1.000	0.223	129.500 ms	1.348 ms
DeepLog-6-2-32-12	0.126	1.000	0.224	129.500 ms	1.438 ms
DeepLog-6-1-32-19	0.127	1.000	0.225	117.614 ms	1.222 ms
DeepLog-6-2-64-14	0.127	1.000	0.225	138.691 ms	1.376 ms
DeepLog-6-2-64-15	0.127	1.000	0.225	138.691 ms	1.400 ms
DeepLog-6-1-32-20	0.127	1.000	0.225	117.614 ms	1.213 ms
DeepLog-6-3-64-16	0.127	1.000	0.225	152.999 ms	1.714 ms
DeepLog-6-2-64-17	0.127	1.000	0.226	138.691 ms	1.414 ms
DeepLog-6-3-64-17	0.127	1.000	0.226	152.999 ms	1.703 ms
DeepLog-6-2-64-16	0.127	1.000	0.226	138.691 ms	1.405 ms
DeepLog-6-3-64-18	0.127	1.000	0.226	152.999 ms	1.723 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
DeepLog-6-3-64-19	0.127	1.000	0.226	152.999 ms	1.730 ms
DeepLog-6-3-64-20	0.128	1.000	0.227	152.999 ms	1.775 ms
DeepLog-6-2-32-13	0.128	1.000	0.227	129.500 ms	1.490 ms
DeepLog-6-2-32-14	0.128	1.000	0.227	129.500 ms	1.478 ms
DeepLog-6-2-64-20	0.130	1.000	0.230	138.691 ms	1.523 ms
DeepLog-6-2-64-18	0.130	1.000	0.230	138.691 ms	1.523 ms
DeepLog-6-2-64-19	0.130	1.000	0.230	138.691 ms	1.514 ms
DeepLog-6-2-32-15	0.130	1.000	0.231	129.500 ms	1.501 ms
DeepLog-6-1-64-17	0.131	1.000	0.231	122.782 ms	3.249 ms
DeepLog-6-2-32-16	0.131	1.000	0.231	129.500 ms	1.515 ms
DeepLog-6-3-32-20	0.131	1.000	0.231	138.385 ms	4.853 ms
DeepLog-6-1-64-19	0.131	1.000	0.232	122.782 ms	3.334 ms
DeepLog-6-1-64-18	0.131	1.000	0.232	122.782 ms	3.268 ms
DeepLog-6-2-32-20	0.131	1.000	0.232	129.500 ms	1.572 ms
DeepLog-6-1-64-20	0.131	1.000	0.232	122.782 ms	3.321 ms
DeepLog-6-2-32-19	0.131	1.000	0.232	129.500 ms	1.547 ms
DeepLog-6-2-32-18	0.131	1.000	0.232	129.500 ms	1.524 ms
DeepLog-6-2-32-17	0.131	1.000	0.232	129.500 ms	1.490 ms
PCA-6-0.000500	0.250	1.000	0.400	0.004 ms	0.051 ms
PCA-6-0.001000	0.280	1.000	0.437	0.004 ms	0.051 ms
LogCluster-0.4-0.1	0.314	1.000	0.478	0.788 ms	0.711 ms
PCA-18-0.000010	0.338	1.000	0.505	0.004 ms	0.051 ms
PCA-16-0.000010	0.344	1.000	0.512	0.004 ms	0.051 ms
PCA-17-0.000010	0.345	1.000	0.513	0.004 ms	0.051 ms
PCA-15-0.000010	0.351	1.000	0.520	0.004 ms	0.051 ms
PCA-18-0.000100	0.351	1.000	0.520	0.004 ms	0.051 ms
PCA-6-0.003000	0.354	1.000	0.522	0.004 ms	0.051 ms
PCA-17-0.000100	0.355	1.000	0.524	0.004 ms	0.051 ms
PCA-18-0.000500	0.360	1.000	0.529	0.004 ms	0.051 ms
PCA-16-0.000100	0.361	1.000	0.530	0.004 ms	0.051 ms
PCA-15-0.000100	0.362	1.000	0.532	0.004 ms	0.051 ms
PCA-18-0.001000	0.364	1.000	0.534	0.004 ms	0.051 ms
PCA-17-0.000500	0.366	1.000	0.536	0.004 ms	0.051 ms
PCA-6-0.005000	0.371	1.000	0.542	0.004 ms	0.051 ms
PCA-17-0.001000	0.372	1.000	0.542	0.004 ms	0.051 ms
PCA-16-0.000500	0.373	1.000	0.543	0.004 ms	0.051 ms
PCA-10-0.000010	0.377	1.000	0.548	0.004 ms	0.051 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
PCA-18-0.003000	0.378	1.000	0.549	0.004 ms	0.051 ms
Invariants Mining-0.96-0.2	0.387	1.000	0.558	8.652 ms	0.002 ms
Invariants Mining-0.96-0.3	0.387	1.000	0.558	8.733 ms	0.002 ms
Invariants Mining-0.96-0.4	0.387	1.000	0.558	8.723 ms	0.002 ms
Invariants Mining-0.96-0.5	0.387	1.000	0.558	8.734 ms	0.002 ms
Invariants Mining-0.96-0.6	0.387	1.000	0.558	8.721 ms	0.002 ms
Invariants Mining-0.96-0.7	0.387	1.000	0.558	8.730 ms	0.002 ms
Invariants Mining-0.96-0.8	0.387	1.000	0.558	8.717 ms	0.002 ms
Invariants Mining-0.96-0.9	0.387	1.000	0.558	8.753 ms	0.002 ms
Invariants Mining-0.96-0.1	0.388	1.000	0.559	8.664 ms	0.002 ms
PCA-16-0.001000	0.388	1.000	0.559	0.004 ms	0.051 ms
PCA-6-0.010000	0.389	1.000	0.560	0.004 ms	0.050 ms
PCA-18-0.005000	0.389	1.000	0.561	0.004 ms	0.051 ms
PCA-15-0.000500	0.395	1.000	0.566	0.004 ms	0.051 ms
PCA-17-0.003000	0.395	1.000	0.567	0.004 ms	0.051 ms
PCA-18-0.010000	0.402	1.000	0.573	0.004 ms	0.051 ms
PCA-15-0.001000	0.402	1.000	0.574	0.004 ms	0.051 ms
PCA-17-0.005000	0.403	1.000	0.575	0.004 ms	0.051 ms
PCA-16-0.003000	0.403	1.000	0.575	0.004 ms	0.051 ms
PCA-16-0.005000	0.408	1.000	0.580	0.004 ms	0.051 ms
PCA-17-0.010000	0.409	1.000	0.580	0.004 ms	0.051 ms
PCA-10-0.000100	0.412	1.000	0.584	0.004 ms	0.051 ms
PCA-18-0.050000	0.420	1.000	0.591	0.004 ms	0.051 ms
PCA-15-0.003000	0.421	1.000	0.593	0.004 ms	0.051 ms
PCA-16-0.010000	0.426	1.000	0.597	0.004 ms	0.051 ms
PCA-15-0.005000	0.427	1.000	0.599	0.004 ms	0.051 ms
PCA-18-0.080000	0.434	1.000	0.606	0.004 ms	0.051 ms
PCA-10-0.000500	0.436	1.000	0.607	0.004 ms	0.051 ms
PCA-17-0.050000	0.441	1.000	0.612	0.004 ms	0.051 ms
PCA-15-0.010000	0.441	1.000	0.612	0.004 ms	0.051 ms
PCA-10-0.001000	0.447	1.000	0.618	0.004 ms	0.051 ms
PCA-6-0.050000	0.450	1.000	0.621	0.004 ms	0.051 ms
PCA-17-0.080000	0.452	1.000	0.622	0.004 ms	0.050 ms
PCA-16-0.050000	0.456	1.000	0.627	0.004 ms	0.051 ms
PCA-6-0.080000	0.463	1.000	0.633	0.004 ms	0.051 ms
PCA-16-0.080000	0.467	1.000	0.637	0.004 ms	0.051 ms
PCA-15-0.050000	0.478	1.000	0.647	0.004 ms	0.051 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
PCA-10-0.003000	0.478	1.000	0.647	0.004 ms	0.051 ms
PCA-10-0.005000	0.484	1.000	0.652	0.004 ms	0.051 ms
Invariants Mining-0.97-0.1	0.488	1.000	0.656	8.685 ms	0.002 ms
Invariants Mining-0.97-0.2	0.488	1.000	0.656	8.695 ms	0.002 ms
Invariants Mining-0.97-0.3	0.488	1.000	0.656	8.694 ms	0.002 ms
Invariants Mining-0.97-0.4	0.488	1.000	0.656	8.709 ms	0.002 ms
Invariants Mining-0.97-0.5	0.488	1.000	0.656	8.741 ms	0.002 ms
Invariants Mining-0.97-0.6	0.488	1.000	0.656	8.759 ms	0.002 ms
Invariants Mining-0.97-0.7	0.488	1.000	0.656	8.744 ms	0.002 ms
Invariants Mining-0.97-0.8	0.488	1.000	0.656	8.817 ms	0.002 ms
Invariants Mining-0.97-0.9	0.488	1.000	0.656	8.772 ms	0.002 ms
Invariants Mining-0.98-0.1	0.488	1.000	0.656	8.656 ms	0.002 ms
Invariants Mining-0.98-0.2	0.488	1.000	0.656	8.739 ms	0.002 ms
Invariants Mining-0.98-0.3	0.488	1.000	0.656	8.743 ms	0.002 ms
Invariants Mining-0.98-0.4	0.488	1.000	0.656	8.749 ms	0.002 ms
Invariants Mining-0.98-0.5	0.488	1.000	0.656	8.746 ms	0.002 ms
Invariants Mining-0.98-0.6	0.488	1.000	0.656	8.748 ms	0.002 ms
Invariants Mining-0.98-0.7	0.488	1.000	0.656	8.805 ms	0.002 ms
Invariants Mining-0.98-0.8	0.488	1.000	0.656	8.796 ms	0.002 ms
Invariants Mining-0.98-0.9	0.488	1.000	0.656	8.789 ms	0.002 ms
PCA-10-0.010000	0.495	1.000	0.662	0.004 ms	0.051 ms
PCA-15-0.080000	0.512	1.000	0.677	0.004 ms	0.051 ms
PCA-10-0.050000	0.552	1.000	0.711	0.004 ms	0.051 ms
PCA-10-0.080000	0.562	1.000	0.720	0.004 ms	0.051 ms
Invariants Mining-0.99-0.4	0.587	1.000	0.740	9.836 ms	0.002 ms
Invariants Mining-0.99-0.5	0.587	1.000	0.740	9.836 ms	0.002 ms
Invariants Mining-0.99-0.6	0.587	1.000	0.740	9.837 ms	0.002 ms
Invariants Mining-0.99-0.7	0.587	1.000	0.740	9.870 ms	0.002 ms
Invariants Mining-0.99-0.8	0.587	1.000	0.740	9.848 ms	0.002 ms
Invariants Mining-0.99-0.9	0.587	1.000	0.740	9.842 ms	0.002 ms
Invariants Mining-0.99-0.1	0.604	1.000	0.753	9.793 ms	0.002 ms
Invariants Mining-0.99-0.2	0.604	1.000	0.753	9.803 ms	0.002 ms
Invariants Mining-0.99-0.3	0.604	1.000	0.753	9.819 ms	0.002 ms
PCA-5-0.080000	0.639	0.999	0.779	0.004 ms	0.051 ms
PCA-5-0.050000	0.648	0.999	0.786	0.004 ms	0.051 ms
PCA-4-0.080000	0.663	0.999	0.797	0.004 ms	0.050 ms
PCA-12-0.080000	0.708	1.000	0.829	0.004 ms	0.051 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
LogCluster-0.3-0.1	0.720	1.000	0.837	0.990 ms	0.976 ms
PCA-12-0.050000	0.723	1.000	0.839	0.004 ms	0.051 ms
PCA-13-0.080000	0.727	1.000	0.842	0.004 ms	0.050 ms
PCA-13-0.050000	0.736	1.000	0.848	0.004 ms	0.051 ms
PCA-11-0.080000	0.738	1.000	0.849	0.004 ms	0.051 ms
PCA-11-0.050000	0.752	1.000	0.859	0.004 ms	0.050 ms
PCA-12-0.010000	0.758	1.000	0.862	0.004 ms	0.051 ms
PCA-13-0.010000	0.764	1.000	0.866	0.004 ms	0.050 ms
PCA-12-0.005000	0.765	1.000	0.867	0.004 ms	0.050 ms
PCA-12-0.003000	0.769	1.000	0.870	0.004 ms	0.051 ms
PCA-14-0.080000	0.771	1.000	0.871	0.004 ms	0.051 ms
PCA-13-0.005000	0.779	1.000	0.876	0.004 ms	0.051 ms
PCA-14-0.050000	0.780	1.000	0.877	0.004 ms	0.051 ms
PCA-11-0.010000	0.783	1.000	0.878	0.004 ms	0.051 ms
PCA-12-0.001000	0.788	1.000	0.882	0.004 ms	0.051 ms
PCA-11-0.005000	0.791	1.000	0.883	0.004 ms	0.051 ms
PCA-13-0.003000	0.793	1.000	0.885	0.004 ms	0.051 ms
PCA-12-0.000500	0.796	1.000	0.886	0.004 ms	0.051 ms
PCA-11-0.003000	0.798	1.000	0.888	0.004 ms	0.051 ms
PCA-13-0.001000	0.804	1.000	0.891	0.004 ms	0.051 ms
PCA-12-0.000100	0.808	1.000	0.894	0.004 ms	0.051 ms
PCA-11-0.001000	0.808	1.000	0.894	0.004 ms	0.051 ms
PCA-14-0.010000	0.810	1.000	0.895	0.004 ms	0.050 ms
PCA-9-0.080000	0.812	1.000	0.896	0.004 ms	0.051 ms
PCA-11-0.000500	0.814	1.000	0.897	0.004 ms	0.051 ms
PCA-13-0.000500	0.814	1.000	0.897	0.004 ms	0.051 ms
PCA-14-0.005000	0.824	1.000	0.904	0.004 ms	0.050 ms
PCA-12-0.000010	0.825	1.000	0.904	0.004 ms	0.050 ms
PCA-9-0.050000	0.827	1.000	0.905	0.005 ms	0.050 ms
PCA-14-0.003000	0.828	1.000	0.906	0.004 ms	0.051 ms
PCA-4-0.050000	0.830	0.999	0.907	0.004 ms	0.051 ms
PCA-13-0.000100	0.831	1.000	0.908	0.004 ms	0.051 ms
PCA-11-0.000100	0.832	1.000	0.908	0.004 ms	0.051 ms
PCA-14-0.001000	0.838	1.000	0.912	0.004 ms	0.050 ms
PCA-8-0.080000	0.838	1.000	0.912	0.004 ms	0.051 ms
PCA-11-0.000010	0.843	1.000	0.915	0.004 ms	0.051 ms
PCA-14-0.000500	0.843	1.000	0.915	0.004 ms	0.051 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
PCA-5-0.010000	0.845	0.999	0.916	0.004 ms	0.051 ms
PCA-13-0.000010	0.848	1.000	0.918	0.004 ms	0.050 ms
PCA-9-0.010000	0.850	1.000	0.919	0.004 ms	0.051 ms
PCA-8-0.050000	0.851	1.000	0.920	0.004 ms	0.050 ms
PCA-7-0.080000	0.855	1.000	0.922	0.004 ms	0.051 ms
LogCluster-0.4-0.2	0.856	1.000	0.922	0.789 ms	0.713 ms
PCA-9-0.005000	0.857	1.000	0.923	0.004 ms	0.051 ms
PCA-9-0.003000	0.860	1.000	0.924	0.004 ms	0.051 ms
PCA-7-0.050000	0.865	1.000	0.928	0.004 ms	0.051 ms
PCA-14-0.000100	0.866	1.000	0.928	0.004 ms	0.050 ms
PCA-5-0.005000	0.869	0.999	0.930	0.004 ms	0.051 ms
PCA-9-0.001000	0.871	1.000	0.931	0.004 ms	0.050 ms
PCA-8-0.010000	0.871	1.000	0.931	0.004 ms	0.051 ms
Invariants Mining-0.995-0.5	0.874	1.000	0.933	10.258 ms	0.002 ms
Invariants Mining-0.995-0.6	0.874	1.000	0.933	10.285 ms	0.002 ms
Invariants Mining-0.995-0.7	0.874	1.000	0.933	10.264 ms	0.002 ms
Invariants Mining-0.995-0.8	0.874	1.000	0.933	10.285 ms	0.002 ms
Invariants Mining-0.995-0.9	0.874	1.000	0.933	10.261 ms	0.002 ms
PCA-14-0.000010	0.877	1.000	0.934	0.004 ms	0.051 ms
PCA-4-0.010000	0.879	0.999	0.935	0.004 ms	0.051 ms
PCA-5-0.003000	0.880	0.999	0.936	0.004 ms	0.050 ms
PCA-9-0.000500	0.882	1.000	0.937	0.004 ms	0.051 ms
LogCluster-0.2-0.1	0.882	1.000	0.937	1.569 ms	1.812 ms
PCA-8-0.005000	0.883	1.000	0.938	0.004 ms	0.050 ms
Invariants Mining-0.995-0.3	0.885	1.000	0.939	10.224 ms	0.002 ms
Invariants Mining-0.995-0.4	0.885	1.000	0.939	10.251 ms	0.002 ms
PCA-4-0.005000	0.886	0.999	0.939	0.004 ms	0.051 ms
PCA-4-0.003000	0.891	0.999	0.942	0.004 ms	0.050 ms
PCA-5-0.001000	0.893	0.999	0.943	0.005 ms	0.050 ms
PCA-8-0.003000	0.896	1.000	0.945	0.004 ms	0.051 ms
PCA-7-0.010000	0.901	1.000	0.948	0.004 ms	0.051 ms
PCA-5-0.000500	0.901	0.999	0.948	0.004 ms	0.050 ms
PCA-4-0.001000	0.902	0.999	0.948	0.004 ms	0.051 ms
PCA-4-0.000500	0.908	0.999	0.951	0.004 ms	0.050 ms
PCA-9-0.000100	0.908	1.000	0.952	0.004 ms	0.050 ms
PCA-7-0.005000	0.908	1.000	0.952	0.004 ms	0.051 ms
PCA-8-0.001000	0.911	1.000	0.954	0.004 ms	0.050 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
PCA-0-0.005000	0.943	0.966	0.954	0.004 ms	0.050 ms
PCA-1-0.005000	0.943	0.966	0.954	0.004 ms	0.050 ms
PCA-0-0.010000	0.942	0.969	0.955	0.004 ms	0.050 ms
PCA-1-0.010000	0.942	0.969	0.955	0.004 ms	0.050 ms
PCA-5-0.000100	0.916	0.999	0.956	0.004 ms	0.051 ms
PCA-7-0.003000	0.916	1.000	0.956	0.005 ms	0.050 ms
PCA-4-0.000100	0.919	0.999	0.957	0.004 ms	0.050 ms
PCA-0-0.080000	0.937	0.979	0.957	0.004 ms	0.051 ms
PCA-1-0.080000	0.937	0.979	0.957	0.004 ms	0.050 ms
PCA-0-0.050000	0.938	0.978	0.958	0.004 ms	0.051 ms
PCA-1-0.050000	0.938	0.978	0.958	0.004 ms	0.051 ms
PCA-8-0.000500	0.920	1.000	0.958	0.004 ms	0.050 ms
LogCluster-0.3-0.2	0.921	1.000	0.959	0.990 ms	0.987 ms
PCA-7-0.001000	0.923	1.000	0.960	0.004 ms	0.051 ms
PCA-9-0.000010	0.924	1.000	0.960	0.004 ms	0.050 ms
PCA-7-0.000500	0.925	1.000	0.961	0.004 ms	0.051 ms
PCA-0-0.000010	0.984	0.944	0.963	0.004 ms	0.051 ms
PCA-1-0.000010	0.984	0.944	0.963	0.004 ms	0.051 ms
PCA-8-0.000100	0.929	1.000	0.963	0.004 ms	0.051 ms
Invariants Mining-0.995-0.1	0.929	1.000	0.963	10.238 ms	0.002 ms
Invariants Mining-0.995-0.2	0.929	1.000	0.963	10.211 ms	0.002 ms
PCA-4-0.000010	0.931	0.999	0.964	0.004 ms	0.051 ms
PCA-5-0.000010	0.932	0.999	0.964	0.004 ms	0.050 ms
PCA-7-0.000100	0.932	1.000	0.965	0.004 ms	0.051 ms
PCA-0-0.000100	0.982	0.951	0.966	0.004 ms	0.050 ms
PCA-1-0.000100	0.982	0.951	0.966	0.004 ms	0.050 ms
PCA-0-0.003000	0.969	0.964	0.966	0.004 ms	0.050 ms
PCA-1-0.003000	0.969	0.964	0.966	0.004 ms	0.051 ms
LogCluster-0.1-0.1	0.937	1.000	0.968	2.583 ms	3.388 ms
PCA-0-0.000500	0.980	0.957	0.968	0.004 ms	0.051 ms
PCA-1-0.000500	0.980	0.957	0.968	0.004 ms	0.051 ms
PCA-7-0.000010	0.939	1.000	0.968	0.004 ms	0.050 ms
PCA-8-0.000010	0.939	1.000	0.968	0.004 ms	0.050 ms
PCA-0-0.001000	0.979	0.960	0.969	0.004 ms	0.050 ms
PCA-1-0.001000	0.979	0.960	0.969	0.004 ms	0.050 ms
PCA-3-0.080000	0.946	0.997	0.971	0.004 ms	0.051 ms
PCA-2-0.080000	0.951	0.995	0.972	0.004 ms	0.051 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
PCA-3-0.050000	0.950	0.996	0.973	0.004 ms	0.050 ms
PCA-2-0.050000	0.954	0.994	0.973	0.004 ms	0.050 ms
PCA-2-0.010000	0.959	0.992	0.975	0.004 ms	0.050 ms
PCA-2-0.003000	0.962	0.990	0.976	0.004 ms	0.050 ms
PCA-2-0.005000	0.961	0.991	0.976	0.004 ms	0.051 ms
PCA-2-0.001000	0.964	0.989	0.976	0.004 ms	0.050 ms
PCA-2-0.000500	0.968	0.988	0.978	0.004 ms	0.051 ms
PCA-2-0.000010	0.972	0.985	0.978	0.004 ms	0.050 ms
PCA-2-0.000100	0.971	0.987	0.979	0.004 ms	0.050 ms
PCA-3-0.010000	0.965	0.995	0.980	0.004 ms	0.050 ms
PCA-3-0.005000	0.968	0.994	0.981	0.004 ms	0.051 ms
PCA-3-0.003000	0.969	0.994	0.981	0.004 ms	0.051 ms
PCA-3-0.001000	0.971	0.993	0.982	0.004 ms	0.050 ms
PCA-3-0.000500	0.972	0.992	0.982	0.005 ms	0.050 ms
Invariants Mining-0.999-0.1	0.965	1.000	0.982	12.762 ms	0.002 ms
Invariants Mining-0.999-0.2	0.965	1.000	0.982	12.972 ms	0.002 ms
Invariants Mining-0.999-0.3	0.965	1.000	0.982	13.020 ms	0.002 ms
Invariants Mining-0.999-0.4	0.965	1.000	0.982	12.984 ms	0.002 ms
Invariants Mining-0.999-0.5	0.965	1.000	0.982	13.024 ms	0.002 ms
Invariants Mining-0.999-0.6	0.965	1.000	0.982	13.273 ms	0.003 ms
Invariants Mining-0.999-0.7	0.965	1.000	0.982	13.268 ms	0.002 ms
Invariants Mining-0.999-0.8	0.965	1.000	0.982	13.258 ms	0.002 ms
Invariants Mining-0.999-0.9	0.965	1.000	0.982	13.271 ms	0.002 ms
PCA-3-0.000100	0.975	0.991	0.983	0.004 ms	0.050 ms
LogCluster-0.4-0.3	0.967	1.000	0.983	0.791 ms	0.713 ms
PCA-3-0.000010	0.978	0.989	0.983	0.004 ms	0.050 ms
LogCluster-0.2-0.2	0.973	1.000	0.986	1.550 ms	1.806 ms
LogCluster-0.1-0.2	0.980	1.000	0.990	2.593 ms	3.394 ms
LogCluster-0.3-0.3	0.984	1.000	0.992	0.982 ms	0.981 ms
LogCluster-0.4-0.4	0.988	1.000	0.994	0.784 ms	0.712 ms
Invariants Mining-1.0-0.1	0.989	1.000	0.994	15.094 ms	0.001 ms
Invariants Mining-1.0-0.2	0.989	1.000	0.994	15.095 ms	0.001 ms
Invariants Mining-1.0-0.3	0.989	1.000	0.994	15.099 ms	0.001 ms
Invariants Mining-1.0-0.4	0.989	1.000	0.994	15.097 ms	0.001 ms
Invariants Mining-1.0-0.5	0.989	1.000	0.994	15.262 ms	0.001 ms
Invariants Mining-1.0-0.6	0.989	1.000	0.994	15.269 ms	0.002 ms
Invariants Mining-1.0-0.7	0.989	1.000	0.994	15.306 ms	0.002 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
Invariants Mining-1.0-0.8	0.989	1.000	0.994	15.265 ms	0.001 ms
Invariants Mining-1.0-0.9	0.989	1.000	0.994	15.296 ms	0.001 ms
LogCluster-0.2-0.3	0.989	1.000	0.995	1.558 ms	1.811 ms
LogCluster-0.1-0.3	0.990	1.000	0.995	2.581 ms	3.396 ms
LogCluster-0.3-0.4	0.990	1.000	0.995	0.989 ms	0.981 ms
LogCluster-0.1-0.4	0.992	1.000	0.996	2.582 ms	3.381 ms
LogCluster-0.2-0.4	0.992	1.000	0.996	1.556 ms	1.806 ms

Table A.4: Performance of different novelty detection methods and hyper-parameter combinations for detecting anomalous logins. t_{train} and $t_{predict}$ are *per data point*.

A.3 ANOMALOUS JOBS DETECTOR

In this Section, we present the results we obtained during hyper-parameter optimisation of the candidate models for the anomalous login detector. The naming of the methods reflects the hyper-parameter combination tested:

- IsolationForest-{t}
- LOF-{*distance metric*}
- OCSVM-{*kernel function*}-{γ}-{ν}
- OCSVM-SGD-{*kernel function*}-{γ}-{ν}

Table A.5 contains the number of true/false positives/negatives; Table A.6 contains the derived metrics and efficiency.

METHOD	TP	FP	FP	TN
OCSVM-sigm.-0.1-0.4	30893	104122	157355	8133
OCSVM-SGD-sigm.-0.2-0.4	29525	97684	163793	9501
OCSVM-sigm.-0.3-0.4	30925	104009	157468	8101
OCSVM-sigm.-0.4-0.4	30921	103906	157571	8105
OCSVM-sigm.-0.2-0.4	30885	103702	157775	8141
OCSVM-sigm.-auto-0.4	31377	104083	157394	7649
OCSVM-sigm.-scale-0.4	31392	104015	157462	7634
OCSVM-SGD-sigm.-0.3-0.4	29739	96429	165048	9287
OCSVM-lin.-scale-0.4	31861	104035	157442	7165
OCSVM-lin.-auto-0.4	31861	104035	157442	7165
OCSVM-lin.-0.1-0.4	31861	104035	157442	7165

METHOD	TP	FP	TN	FN
OCSVM-lin.-0.2-0.4	31861	104035	157442	7165
OCSVM-lin.-0.3-0.4	31861	104035	157442	7165
OCSVM-lin.-0.4-0.4	31861	104035	157442	7165
OCSVM-SGD-sigm.-0.1-0.4	29767	92901	168576	9259
OCSVM-poly-scale-0.4	32743	104952	156525	6283
OCSVM-poly-0.3-0.4	32733	104491	156986	6293
OCSVM-poly-auto-0.4	32733	104452	157025	6293
OCSVM-poly-0.1-0.4	32734	104455	157022	6292
OCSVM-poly-0.4-0.4	32733	104443	157034	6293
OCSVM-poly-0.2-0.4	32734	104417	157060	6292
OCSVM-SGD-lin.-0.1-0.4	31728	97593	163884	7298
OCSVM-SGD-lin.-0.2-0.4	31728	97593	163884	7298
OCSVM-SGD-lin.-0.3-0.4	31728	97593	163884	7298
OCSVM-SGD-lin.-0.4-0.4	31728	97593	163884	7298
OCSVM-SGD-sigm.-0.4-0.4	30354	90989	170488	8672
OCSVM-SGD-poly-0.1-0.4	33661	97494	163983	5365
OCSVM-rbf-auto-0.4	36067	105533	155944	2959
OCSVM-SGD-poly-0.2-0.4	33587	94518	166959	5439
OCSVM-poly-auto-0.3	32679	88116	173361	6347
OCSVM-SGD-poly-0.3-0.4	33531	89906	171571	5495
OCSVM-sigm.-0.3-0.3	30643	78384	183093	8383
OCSVM-sigm.-0.2-0.3	30663	78401	183076	8363
OCSVM-sigm.-0.4-0.3	30640	78270	183207	8386
OCSVM-sigm.-0.1-0.3	30844	78466	183011	8182
OCSVM-sigm.-auto-0.3	30963	78483	182994	8063
OCSVM-rbf-scale-0.4	38129	105377	156100	897
OCSVM-sigm.-scale-0.3	31329	78860	182617	7697
OCSVM-rbf-0.4-0.4	38816	105829	155648	210
OCSVM-rbf-0.3-0.4	38791	105713	155764	235
OCSVM-rbf-0.2-0.4	38791	105516	155961	235
OCSVM-rbf-0.1-0.4	38775	105450	156027	251
OCSVM-lin.-scale-0.3	31510	78366	183111	7516
OCSVM-lin.-auto-0.3	31510	78366	183111	7516
OCSVM-lin.-0.1-0.3	31510	78366	183111	7516
OCSVM-lin.-0.2-0.3	31510	78366	183111	7516
OCSVM-lin.-0.3-0.3	31510	78366	183111	7516
OCSVM-lin.-0.4-0.3	31510	78366	183111	7516

METHOD	TP	FP	TN	FN
OCSVM-SGD-poly-0.4-0.4	33470	84964	176513	5556
OCSVM-poly-0.1-0.3	32678	79672	181805	6348
OCSVM-poly-scale-0.3	32678	78617	182860	6348
OCSVM-poly-0.2-0.3	32679	78410	183067	6347
OCSVM-poly-0.4-0.3	32678	78379	183098	6348
OCSVM-poly-0.3-0.3	32679	78339	183138	6347
OCSVM-SGD-lin.-0.1-0.3	31468	70607	190870	7558
OCSVM-SGD-lin.-0.2-0.3	31468	70607	190870	7558
OCSVM-SGD-lin.-0.3-0.3	31468	70607	190870	7558
OCSVM-SGD-lin.-0.4-0.3	31468	70607	190870	7558
OCSVM-SGD-sigm.-0.3-0.3	29400	62831	198646	9626
OCSVM-SGD-sigm.-0.2-0.3	29326	62412	199065	9700
OCSVM-SGD-rbf-0.4-0.4	38711	93113	168364	315
OCSVM-SGD-sigm.-0.4-0.3	30148	63426	198051	8878
OCSVM-SGD-rbf-0.3-0.4	38713	91608	169869	313
OCSVM-rbf-auto-0.3	36025	79429	182048	3001
OCSVM-SGD-poly-0.1-0.3	33586	69844	191633	5440
OCSVM-SGD-poly-0.2-0.3	33544	69043	192434	5482
OCSVM-SGD-rbf-0.2-0.4	38747	84386	177091	279
OCSVM-rbf-scale-0.3	37513	79285	182192	1513
OCSVM-SGD-poly-0.3-0.3	33465	65715	195762	5561
OCSVM-rbf-0.1-0.3	38326	79516	181961	700
OCSVM-rbf-0.4-0.3	38769	79751	181726	257
OCSVM-rbf-0.3-0.3	38746	79589	181888	280
OCSVM-rbf-0.2-0.3	38712	79452	182025	314
OCSVM-SGD-sigm.-0.1-0.3	29674	51331	210146	9352
OCSVM-SGD-poly-0.4-0.3	33413	61300	200177	5613
OCSVM-sigm.-0.2-0.2	30603	52423	209054	8423
OCSVM-sigm.-0.3-0.2	30550	52183	209294	8476
OCSVM-sigm.-0.1-0.2	30727	52672	208805	8299
OCSVM-sigm.-0.4-0.2	30516	52041	209436	8510
OCSVM-sigm.-auto-0.2	30874	52598	208879	8152
OCSVM-sigm.-scale-0.2	31206	52792	208685	7820
OCSVM-lin.-scale-0.2	31455	52312	209165	7571
OCSVM-lin.-auto-0.2	31455	52312	209165	7571
OCSVM-lin.-0.1-0.2	31455	52312	209165	7571
OCSVM-lin.-0.2-0.2	31455	52312	209165	7571

METHOD	TP	FP	TN	FN
OCSVM-lin.-0.3-0.2	31455	52312	209165	7571
OCSVM-lin.-0.4-0.2	31455	52312	209165	7571
OCSVM-poly-scale-0.2	32635	53218	208259	6391
OCSVM-SGD-rbf-0.1-0.4	38710	69755	191722	316
OCSVM-poly-0.2-0.2	32618	52247	209230	6408
OCSVM-poly-0.3-0.2	32619	52121	209356	6407
OCSVM-poly-0.4-0.2	32618	52112	209365	6408
OCSVM-poly-auto-0.2	32618	51902	209575	6408
OCSVM-SGD-rbf-0.4-0.3	38650	66624	194853	376
OCSVM-poly-0.1-0.2	32619	48650	212827	6407
OCSVM-SGD-lin.-0.1-0.2	31378	44364	217113	7648
OCSVM-SGD-lin.-0.2-0.2	31378	44364	217113	7648
OCSVM-SGD-lin.-0.3-0.2	31378	44364	217113	7648
OCSVM-SGD-lin.-0.4-0.2	31378	44364	217113	7648
OCSVM-SGD-rbf-0.3-0.3	38637	62320	199157	389
OCSVM-poly-auto-0.1	32628	44771	216706	6398
OCSVM-rbf-auto-0.2	35950	53266	208211	3076
OCSVM-SGD-rbf-0.2-0.3	38571	56776	204701	455
OCSVM-rbf-scale-0.2	37202	53194	208283	1824
OCSVM-SGD-poly-0.2-0.2	33493	42386	219091	5533
OCSVM-SGD-sigm.-0.4-0.2	30025	33717	227760	9001
OCSVM-rbf-0.1-0.2	38260	53207	208270	766
OCSVM-rbf-0.4-0.2	38732	54095	207382	294
OCSVM-SGD-poly-0.1-0.2	33538	41505	219972	5488
OCSVM-rbf-0.3-0.2	38704	53902	207575	322
OCSVM-rbf-0.2-0.2	38609	53515	207962	417
OCSVM-SGD-poly-0.3-0.2	33413	39131	222346	5613
OCSVM-SGD-poly-0.4-0.2	33297	37467	224010	5729
OCSVM-SGD-sigm.-0.3-0.2	29306	27155	234322	9720
OCSVM-sigm.-0.3-0.1	30385	26131	235346	8641
OCSVM-sigm.-0.4-0.1	30374	25972	235505	8652
OCSVM-sigm.-0.2-0.1	30382	25967	235510	8644
OCSVM-sigm.-0.1-0.1	30458	26100	235377	8568
OCSVM-sigm.-auto-0.1	30650	26232	235245	8376
OCSVM-sigm.-scale-0.1	31023	26280	235197	8003
OCSVM-lin.-scale-0.1	31401	26151	235326	7625
OCSVM-lin.-auto-0.1	31401	26151	235326	7625

METHOD	TP	FP	TN	FN
OCSVM-lin.-0.1-0.1	31401	26151	235326	7625
OCSVM-lin.-0.2-0.1	31401	26151	235326	7625
OCSVM-lin.-0.3-0.1	31401	26151	235326	7625
OCSVM-lin.-0.4-0.1	31401	26151	235326	7625
OCSVM-SGD-sigm.-0.2-0.2	29130	21162	240315	9896
OCSVM-SGD-rbf-0.1-0.3	38208	39765	221712	818
OCSVM-poly-scale-0.1	32581	28074	233403	6445
OCSVM-poly-0.1-0.1	32576	27062	234415	6450
OCSVM-poly-0.2-0.1	32576	26015	235462	6450
OCSVM-poly-0.3-0.1	32577	26012	235465	6449
OCSVM-poly-0.4-0.1	32576	25987	235490	6450
iForest-20	37392	34922	226555	1634
iForest-30	37145	32298	229179	1881
LOF-cosine	76640	68412	454805	2031
iForest-50	37176	32223	229254	1850
OCSVM-rbf-scale-0.1	34231	26577	234900	4795
LOF-correlation	76648	68073	455144	2023
OCSVM-SGD-rbf-0.4-0.2	38031	33702	227775	995
iForest-10	37145	31738	229739	1881
iForest-100	37458	31257	230220	1568
OCSVM-SGD-lin.-0.1-0.1	31241	18965	242512	7785
OCSVM-SGD-lin.-0.2-0.1	31241	18965	242512	7785
OCSVM-SGD-lin.-0.3-0.1	31241	18965	242512	7785
OCSVM-SGD-lin.-0.4-0.1	31241	18965	242512	7785
OCSVM-rbf-auto-0.1	35854	27108	234369	3172
OCSVM-SGD-sigm.-0.1-0.2	29345	13391	248086	9681
OCSVM-SGD-rbf-0.3-0.2	37983	27676	233801	1043
OCSVM-rbf-0.4-0.1	38686	28349	233128	340
OCSVM-rbf-0.1-0.1	38182	27316	234161	844
OCSVM-rbf-0.3-0.1	38629	28005	233472	397
OCSVM-rbf-0.2-0.1	38563	27493	233984	463
OCSVM-SGD-poly-0.4-0.1	32608	16193	245284	6418
OCSVM-SGD-poly-0.3-0.1	32872	16271	245206	6154
OCSVM-SGD-poly-0.2-0.1	33279	14361	247116	5747
OCSVM-SGD-rbf-0.2-0.2	37979	21684	239793	1047
OCSVM-SGD-sigm.-0.3-0.1	28277	5855	255622	10749
OCSVM-SGD-sigm.-0.2-0.1	28602	5941	255536	10424

METHOD	TP	FP	TN	FN
OCSVM-SGD-rbf-0.1-0.2	37961	19731	241746	1065
OCSVM-SGD-poly-0.1-0.1	33377	10355	251122	5649
OCSVM-SGD-sigm.-0.4-0.1	29338	4349	257128	9688
OCSVM-SGD-sigm.-0.1-0.1	29171	3319	258158	9855
LOF-minkowski	71068	20818	502399	7603
LOF-chebyshev	71405	20274	502943	7266
LOF-braycurtis	76530	19958	503259	2141
OCSVM-SGD-rbf-0.1-0.1	37279	8951	252526	1747
OCSVM-SGD-rbf-0.4-0.1	37311	8845	252632	1715
OCSVM-SGD-rbf-0.3-0.1	37270	8781	252696	1756
OCSVM-SGD-rbf-0.2-0.1	37266	8581	252896	1760

Table A.5: Number of true/false positive/negative predictions detecting anomalous jobs using various hyper-parameter combinations.

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
OCSVM-sigm.-0.1-0.4	0.229	0.792	0.355	1.632 ms	0.184 ms
OCSVM-SGD-sigm.-0.2-0.4	0.232	0.757	0.355	0.003 ms	0.000 ms
OCSVM-sigm.-0.3-0.4	0.229	0.792	0.356	1.692 ms	0.194 ms
OCSVM-sigm.-0.4-0.4	0.229	0.792	0.356	1.692 ms	0.194 ms
OCSVM-sigm.-0.2-0.4	0.229	0.791	0.356	1.682 ms	0.192 ms
OCSVM-sigm.-auto-0.4	0.232	0.804	0.360	1.624 ms	0.181 ms
OCSVM-sigm.-scale-0.4	0.232	0.804	0.360	2.164 ms	0.264 ms
OCSVM-SGD-sigm.-0.3-0.4	0.236	0.762	0.360	0.003 ms	0.000 ms
OCSVM-lin.-scale-0.4	0.234	0.816	0.364	0.846 ms	0.084 ms
OCSVM-lin.-auto-0.4	0.234	0.816	0.364	0.848 ms	0.084 ms
OCSVM-lin.-0.1-0.4	0.234	0.816	0.364	0.847 ms	0.084 ms
OCSVM-lin.-0.2-0.4	0.234	0.816	0.364	0.848 ms	0.084 ms
OCSVM-lin.-0.3-0.4	0.234	0.816	0.364	0.892 ms	0.086 ms
OCSVM-lin.-0.4-0.4	0.234	0.816	0.364	0.847 ms	0.084 ms
OCSVM-SGD-sigm.-0.1-0.4	0.243	0.763	0.368	0.003 ms	0.000 ms
OCSVM-poly-scale-0.4	0.238	0.839	0.371	1.006 ms	0.101 ms
OCSVM-poly-0.3-0.4	0.239	0.839	0.371	1.007 ms	0.101 ms
OCSVM-poly-auto-0.4	0.239	0.839	0.372	1.006 ms	0.101 ms
OCSVM-poly-0.1-0.4	0.239	0.839	0.372	1.006 ms	0.101 ms
OCSVM-poly-0.4-0.4	0.239	0.839	0.372	1.006 ms	0.101 ms
OCSVM-poly-0.2-0.4	0.239	0.839	0.372	1.007 ms	0.101 ms
OCSVM-SGD-lin.-0.1-0.4	0.245	0.813	0.377	0.002 ms	0.000 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
OCSVM-SGD-lin.-0.2-0.4	0.245	0.813	0.377	0.002 ms	0.000 ms
OCSVM-SGD-lin.-0.3-0.4	0.245	0.813	0.377	0.002 ms	0.000 ms
OCSVM-SGD-lin.-0.4-0.4	0.245	0.813	0.377	0.002 ms	0.000 ms
OCSVM-SGD-sigm.-0.4-0.4	0.250	0.778	0.379	0.003 ms	0.000 ms
OCSVM-SGD-poly-0.1-0.4	0.257	0.863	0.396	0.005 ms	0.001 ms
OCSVM-rbf-auto-0.4	0.255	0.924	0.399	1.454 ms	0.223 ms
OCSVM-SGD-poly-0.2-0.4	0.262	0.861	0.402	0.005 ms	0.001 ms
OCSVM-poly-auto-0.3	0.271	0.837	0.409	0.753 ms	0.076 ms
OCSVM-SGD-poly-0.3-0.4	0.272	0.859	0.413	0.005 ms	0.001 ms
OCSVM-sigm.-0.3-0.3	0.281	0.785	0.414	1.277 ms	0.145 ms
OCSVM-sigm.-0.2-0.3	0.281	0.786	0.414	1.270 ms	0.144 ms
OCSVM-sigm.-0.4-0.3	0.281	0.785	0.414	1.275 ms	0.146 ms
OCSVM-sigm.-0.1-0.3	0.282	0.790	0.416	1.240 ms	0.138 ms
OCSVM-sigm.-auto-0.3	0.283	0.793	0.417	1.216 ms	0.136 ms
OCSVM-rbf-scale-0.4	0.266	0.977	0.418	1.345 ms	0.221 ms
OCSVM-sigm.-scale-0.3	0.284	0.803	0.420	1.655 ms	0.197 ms
OCSVM-rbf-0.4-0.4	0.268	0.995	0.423	1.356 ms	0.220 ms
OCSVM-rbf-0.3-0.4	0.268	0.994	0.423	1.362 ms	0.221 ms
OCSVM-rbf-0.2-0.4	0.269	0.994	0.423	1.368 ms	0.221 ms
OCSVM-rbf-0.1-0.4	0.269	0.994	0.423	1.400 ms	0.221 ms
OCSVM-lin.-scale-0.3	0.287	0.807	0.423	0.630 ms	0.063 ms
OCSVM-lin.-auto-0.3	0.287	0.807	0.423	0.629 ms	0.063 ms
OCSVM-lin.-0.1-0.3	0.287	0.807	0.423	0.632 ms	0.063 ms
OCSVM-lin.-0.2-0.3	0.287	0.807	0.423	0.631 ms	0.063 ms
OCSVM-lin.-0.3-0.3	0.287	0.807	0.423	0.632 ms	0.076 ms
OCSVM-lin.-0.4-0.3	0.287	0.807	0.423	0.631 ms	0.063 ms
OCSVM-SGD-poly-0.4-0.4	0.283	0.858	0.425	0.004 ms	0.001 ms
OCSVM-poly-0.1-0.3	0.291	0.837	0.432	0.752 ms	0.076 ms
OCSVM-poly-scale-0.3	0.294	0.837	0.435	0.754 ms	0.076 ms
OCSVM-poly-0.2-0.3	0.294	0.837	0.435	0.753 ms	0.076 ms
OCSVM-poly-0.4-0.3	0.294	0.837	0.435	0.754 ms	0.076 ms
OCSVM-poly-0.3-0.3	0.294	0.837	0.436	0.754 ms	0.076 ms
OCSVM-SGD-lin.-0.1-0.3	0.308	0.806	0.446	0.002 ms	0.000 ms
OCSVM-SGD-lin.-0.2-0.3	0.308	0.806	0.446	0.002 ms	0.000 ms
OCSVM-SGD-lin.-0.3-0.3	0.308	0.806	0.446	0.002 ms	0.000 ms
OCSVM-SGD-lin.-0.4-0.3	0.308	0.806	0.446	0.002 ms	0.000 ms
OCSVM-SGD-sigm.-0.3-0.3	0.319	0.753	0.448	0.003 ms	0.000 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
OCSVM-SGD-sigm.-0.2-0.3	0.320	0.751	0.449	0.003 ms	0.000 ms
OCSVM-SGD-rbf-0.4-0.4	0.294	0.992	0.453	0.002 ms	0.000 ms
OCSVM-SGD-sigm.-0.4-0.3	0.322	0.773	0.455	0.003 ms	0.000 ms
OCSVM-SGD-rbf-0.3-0.4	0.297	0.992	0.457	0.002 ms	0.000 ms
OCSVM-rbf-auto-0.3	0.312	0.923	0.466	1.061 ms	0.168 ms
OCSVM-SGD-poly-0.1-0.3	0.325	0.861	0.472	0.005 ms	0.001 ms
OCSVM-SGD-poly-0.2-0.3	0.327	0.860	0.474	0.005 ms	0.001 ms
OCSVM-SGD-rbf-0.2-0.4	0.315	0.993	0.478	0.002 ms	0.000 ms
OCSVM-rbf-scale-0.3	0.321	0.961	0.481	0.984 ms	0.166 ms
OCSVM-SGD-poly-0.3-0.3	0.337	0.858	0.484	0.005 ms	0.001 ms
OCSVM-rbf-0.1-0.3	0.325	0.982	0.489	1.024 ms	0.166 ms
OCSVM-rbf-0.4-0.3	0.327	0.993	0.492	0.994 ms	0.166 ms
OCSVM-rbf-0.3-0.3	0.327	0.993	0.492	0.995 ms	0.165 ms
OCSVM-rbf-0.2-0.3	0.328	0.992	0.493	1.001 ms	0.166 ms
OCSVM-SGD-sigm.-0.1-0.3	0.366	0.760	0.494	0.003 ms	0.000 ms
OCSVM-SGD-poly-0.4-0.3	0.353	0.856	0.500	0.004 ms	0.001 ms
OCSVM-sigm.-0.2-0.2	0.369	0.784	0.501	0.848 ms	0.096 ms
OCSVM-sigm.-0.3-0.2	0.369	0.783	0.502	0.850 ms	0.097 ms
OCSVM-sigm.-0.1-0.2	0.368	0.787	0.502	0.827 ms	0.092 ms
OCSVM-sigm.-0.4-0.2	0.370	0.782	0.502	0.847 ms	0.097 ms
OCSVM-sigm.-auto-0.2	0.370	0.791	0.504	0.814 ms	0.090 ms
OCSVM-sigm.-scale-0.2	0.372	0.800	0.507	1.112 ms	0.132 ms
OCSVM-lin.-scale-0.2	0.376	0.806	0.512	0.416 ms	0.042 ms
OCSVM-lin.-auto-0.2	0.376	0.806	0.512	0.414 ms	0.042 ms
OCSVM-lin.-0.1-0.2	0.376	0.806	0.512	0.414 ms	0.042 ms
OCSVM-lin.-0.2-0.2	0.376	0.806	0.512	0.415 ms	0.042 ms
OCSVM-lin.-0.3-0.2	0.376	0.806	0.512	0.415 ms	0.042 ms
OCSVM-lin.-0.4-0.2	0.376	0.806	0.512	0.415 ms	0.042 ms
OCSVM-poly-scale-0.2	0.380	0.836	0.523	0.502 ms	0.051 ms
OCSVM-SGD-rbf-0.1-0.4	0.357	0.992	0.525	0.002 ms	0.000 ms
OCSVM-poly-0.2-0.2	0.384	0.836	0.527	0.503 ms	0.050 ms
OCSVM-poly-0.3-0.2	0.385	0.836	0.527	0.503 ms	0.051 ms
OCSVM-poly-0.4-0.2	0.385	0.836	0.527	0.504 ms	0.051 ms
OCSVM-poly-auto-0.2	0.386	0.836	0.528	0.503 ms	0.051 ms
OCSVM-SGD-rbf-0.4-0.3	0.367	0.990	0.536	0.002 ms	0.000 ms
OCSVM-poly-0.1-0.2	0.401	0.836	0.542	0.504 ms	0.050 ms
OCSVM-SGD-lin.-0.1-0.2	0.414	0.804	0.547	0.002 ms	0.000 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
OCSVM-SGD-lin.-0.2-0.2	0.414	0.804	0.547	0.002 ms	0.000 ms
OCSVM-SGD-lin.-0.3-0.2	0.414	0.804	0.547	0.002 ms	0.000 ms
OCSVM-SGD-lin.-0.4-0.2	0.414	0.804	0.547	0.002 ms	0.000 ms
OCSVM-SGD-rbf-0.3-0.3	0.383	0.990	0.552	0.002 ms	0.000 ms
OCSVM-poly-auto-0.1	0.422	0.836	0.560	0.245 ms	0.025 ms
OCSVM-rbf-auto-0.2	0.403	0.921	0.561	0.682 ms	0.111 ms
OCSVM-SGD-rbf-0.2-0.3	0.405	0.988	0.574	0.002 ms	0.000 ms
OCSVM-rbf-scale-0.2	0.412	0.953	0.575	0.631 ms	0.111 ms
OCSVM-SGD-poly-0.2-0.2	0.441	0.858	0.583	0.005 ms	0.001 ms
OCSVM-SGD-sigm.-0.4-0.2	0.471	0.769	0.584	0.003 ms	0.000 ms
OCSVM-rbf-0.1-0.2	0.418	0.980	0.586	0.658 ms	0.111 ms
OCSVM-rbf-0.4-0.2	0.417	0.992	0.588	0.637 ms	0.111 ms
OCSVM-SGD-poly-0.1-0.2	0.447	0.859	0.588	0.005 ms	0.001 ms
OCSVM-rbf-0.3-0.2	0.418	0.992	0.588	0.639 ms	0.111 ms
OCSVM-rbf-0.2-0.2	0.419	0.989	0.589	0.644 ms	0.111 ms
OCSVM-SGD-poly-0.3-0.2	0.461	0.856	0.599	0.004 ms	0.001 ms
OCSVM-SGD-poly-0.4-0.2	0.471	0.853	0.607	0.004 ms	0.001 ms
OCSVM-SGD-sigm.-0.3-0.2	0.519	0.751	0.614	0.003 ms	0.000 ms
OCSVM-sigm.-0.3-0.1	0.538	0.779	0.636	0.418 ms	0.049 ms
OCSVM-sigm.-0.4-0.1	0.539	0.778	0.637	0.414 ms	0.049 ms
OCSVM-sigm.-0.2-0.1	0.539	0.779	0.637	0.415 ms	0.048 ms
OCSVM-sigm.-0.1-0.1	0.539	0.780	0.637	0.408 ms	0.046 ms
OCSVM-sigm.-auto-0.1	0.539	0.785	0.639	0.406 ms	0.045 ms
OCSVM-sigm.-scale-0.1	0.541	0.795	0.644	0.552 ms	0.066 ms
OCSVM-lin.-scale-0.1	0.546	0.805	0.650	0.198 ms	0.021 ms
OCSVM-lin.-auto-0.1	0.546	0.805	0.650	0.198 ms	0.021 ms
OCSVM-lin.-0.1-0.1	0.546	0.805	0.650	0.199 ms	0.021 ms
OCSVM-lin.-0.2-0.1	0.546	0.805	0.650	0.198 ms	0.021 ms
OCSVM-lin.-0.3-0.1	0.546	0.805	0.650	0.198 ms	0.021 ms
OCSVM-lin.-0.4-0.1	0.546	0.805	0.650	0.198 ms	0.021 ms
OCSVM-SGD-sigm.-0.2-0.2	0.579	0.746	0.652	0.003 ms	0.000 ms
OCSVM-SGD-rbf-0.1-0.3	0.490	0.979	0.653	0.002 ms	0.000 ms
OCSVM-poly-scale-0.1	0.537	0.835	0.654	0.246 ms	0.025 ms
OCSVM-poly-0.1-0.1	0.546	0.835	0.660	0.245 ms	0.025 ms
OCSVM-poly-0.2-0.1	0.556	0.835	0.667	0.245 ms	0.025 ms
OCSVM-poly-0.3-0.1	0.556	0.835	0.667	0.247 ms	0.025 ms
OCSVM-poly-0.4-0.1	0.556	0.835	0.668	0.246 ms	0.025 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$
iForest-20	0.517	0.958	0.672	0.005 ms	0.002 ms
iForest-30	0.535	0.952	0.685	0.007 ms	0.002 ms
LOF-cosine	0.528	0.974	0.685	0.820 ms	0.545 ms
iForest-50	0.536	0.953	0.686	0.012 ms	0.004 ms
OCSVM-rbf-scale-0.1	0.563	0.877	0.686	0.300 ms	0.057 ms
LOF-correlation	0.530	0.974	0.686	0.967 ms	0.645 ms
OCSVM-SGD-rbf-0.4-0.2	0.530	0.975	0.687	0.002 ms	0.000 ms
iForest-10	0.539	0.952	0.688	0.003 ms	0.001 ms
iForest-100	0.545	0.960	0.695	0.023 ms	0.007 ms
OCSVM-SGD-lin.-0.1-0.1	0.622	0.801	0.700	0.023 ms	0.000 ms
OCSVM-SGD-lin.-0.2-0.1	0.622	0.801	0.700	0.003 ms	0.000 ms
OCSVM-SGD-lin.-0.3-0.1	0.622	0.801	0.700	0.003 ms	0.000 ms
OCSVM-SGD-lin.-0.4-0.1	0.622	0.801	0.700	0.003 ms	0.000 ms
OCSVM-rbf-auto-0.1	0.569	0.919	0.703	0.326 ms	0.056 ms
OCSVM-SGD-sigm.-0.1-0.2	0.687	0.752	0.718	0.003 ms	0.000 ms
OCSVM-SGD-rbf-0.3-0.2	0.578	0.973	0.726	0.002 ms	0.000 ms
OCSVM-rbf-0.4-0.1	0.577	0.991	0.730	0.304 ms	0.056 ms
OCSVM-rbf-0.1-0.1	0.583	0.978	0.731	0.315 ms	0.055 ms
OCSVM-rbf-0.3-0.1	0.580	0.990	0.731	0.303 ms	0.056 ms
OCSVM-rbf-0.2-0.1	0.584	0.988	0.734	0.306 ms	0.056 ms
OCSVM-SGD-poly-0.4-0.1	0.668	0.836	0.743	0.005 ms	0.001 ms
OCSVM-SGD-poly-0.3-0.1	0.669	0.842	0.746	0.005 ms	0.001 ms
OCSVM-SGD-poly-0.2-0.1	0.699	0.853	0.768	0.004 ms	0.001 ms
OCSVM-SGD-rbf-0.2-0.2	0.637	0.973	0.770	0.002 ms	0.000 ms
OCSVM-SGD-sigm.-0.3-0.1	0.828	0.725	0.773	0.003 ms	0.000 ms
OCSVM-SGD-sigm.-0.2-0.1	0.828	0.733	0.778	0.002 ms	0.000 ms
OCSVM-SGD-rbf-0.1-0.2	0.658	0.973	0.785	0.002 ms	0.000 ms
OCSVM-SGD-poly-0.1-0.1	0.763	0.855	0.807	0.005 ms	0.001 ms
OCSVM-SGD-sigm.-0.4-0.1	0.871	0.752	0.807	0.002 ms	0.000 ms
OCSVM-SGD-sigm.-0.1-0.1	0.898	0.747	0.816	0.002 ms	0.000 ms
LOF-minkowski	0.773	0.903	0.833	0.126 ms	0.082 ms
LOF-chebyshev	0.779	0.908	0.838	0.734 ms	0.488 ms
LOF-braycurtis	0.793	0.973	0.874	1.010 ms	0.666 ms
OCSVM-SGD-rbf-0.1-0.1	0.806	0.955	0.875	0.002 ms	0.000 ms
OCSVM-SGD-rbf-0.4-0.1	0.808	0.956	0.876	0.002 ms	0.000 ms
OCSVM-SGD-rbf-0.3-0.1	0.809	0.955	0.876	0.002 ms	0.000 ms
OCSVM-SGD-rbf-0.2-0.1	0.813	0.955	0.878	0.002 ms	0.000 ms

METHOD	PREC.	RECALL	F1	t_{train}	$t_{predict}$

Table A.6: Performance of different novelty detection methods and hyper-parameter combinations for detecting anomalous jobs. t_{train} and $t_{predict}$ are *per data point*.

B ANOMALY SCORES

The anomaly score $p_1(s)$ denotes the degree of anomaly of the initial login of a user session s. We determine the $p_1(s)$ as laid out in Section 4.4.1 using a OCSVM-SGD. This OCSVM-SGD calculates a login's signed distance to a hyperplane separating normal and anomalous logins. Figure B.1 shows the signed distance of logins $d(s)$ plotted against the resulting anomaly score $p_1(s)$. We sampled 20 normal logins from the user sessions collected on our test system and 20 anomalous logins we generated randomly.

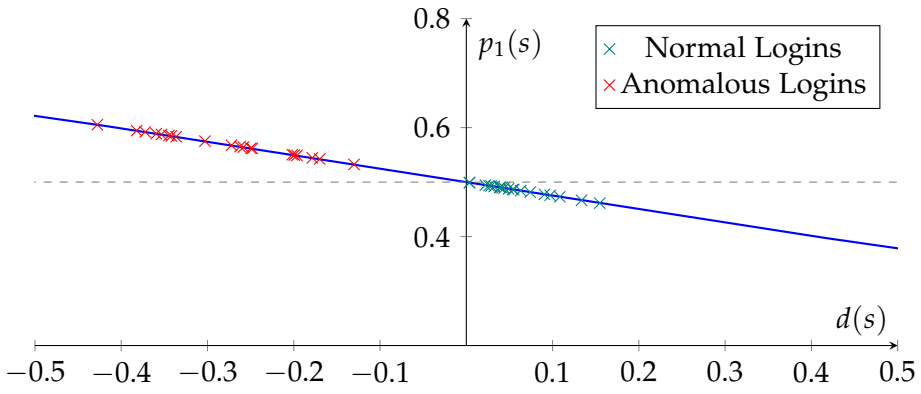


Figure B.1: A random sample of normal/anomalous logins. The plot shows the logins' signed distance to the separating hyperplane and the resulting anomaly score.

BIBLIOGRAPHY

[1] Burak Aksar, Yijia Zhang, Emre Ates, Benjamin Schwaller, Omar Aaziz, Vitus J. Leung, Jim Brandt, Manuel Egele, and Ayse K. Coskun. "Proctor: A Semi-Supervised Performance Anomaly Diagnosis Framework for Production HPC Systems." en. In: *High Performance Computing*. Ed. by Bradford L. Chamberlain, Ana-Lucia Varbanescu, Hatem Ltaief, and Piotr Luszczek. Vol. 12728. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2021, pp. 195–214. ISBN: 978-3-030-78712-7 978-3-030-78713-4. DOI: 10.1007/978-3-030-78713-4_11. URL: https://link.springer.com/10.1007/978-3-030-78713-4_11 (visited on 01/03/2022).

[2] Ethem Alpaydin. *Introduction to machine learning*. 2nd ed. Adaptive computation and machine learning. OCLC: ocn317698631. Cambridge, Mass: MIT Press, 2010. ISBN: 978-0-262-01243-0.

[3] Emre Ates, Ozan Tuncer, Ata Turk, Vitus J. Leung, Jim Brandt, Manuel Egele, and Ayse K. Coskun. "Taxonomist: Application Detection Through Rich Monitoring Data." In: *Euro-Par 2018: Parallel Processing*. Ed. by Marco Aldinucci, Luca Padovani, and Massimo Torquati. Vol. 11014. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 92–105. ISBN: 978-3-319-96982-4 978-3-319-96983-1. DOI: 10.1007/978-3-319-96983-1_7. URL: http://link.springer.com/10.1007/978-3-319-96983-1_7 (visited on 01/03/2022).

[4] Andrea Borghesi, Andrea Bartolini, Michele Lombardi, Michela Milano, and Luca Benini. "Anomaly Detection using Autoencoders in High Performance Computing Systems." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (July 2019). arXiv: 1811.05269, pp. 9428–9433. ISSN: 2374-3468, 2159-5399. DOI: 10.1609/aaai.v33i01.33019428. URL: http://arxiv.org/abs/1811.05269 (visited on 05/05/2022).

[5] Andrea Borghesi, Antonio Libri, Luca Benini, and Andrea Bartolini. "Online Anomaly Detection in HPC Systems." In: *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. Hsinchu, Taiwan: IEEE, Mar. 2019, pp. 229–233. ISBN: 978-1-5386-7884-8. DOI: 10.1109/AICAS.2019.8771527. URL: https://ieeexplore.ieee.org/document/8771527/ (visited on 01/03/2022).

[6] J. R. Bray and J. T. Curtis. "An Ordination of the Upland Forest Communities of Southern Wisconsin." In: (1957). DOI: 10.2307/1942268.

[7] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. "LOF: identifying density-based local outliers." In: *ACM SIGMOD Record* 29.2 (May 2000), pp. 93–104. ISSN: 0163-5808. DOI: 10.1145/335191.335388. URL: https://doi.org/10.1145/335191.335388 (visited on 04/28/2022).

[8] Pascal Brückner. *Analyzing a compromised HPC cluster*. Feb. 2021. URL: https://www.educv.de/blog/post-2021-02-17-analyzing-a-compromised-hpc-cluster/ (visited on 03/15/2022).

[9] Zhuangbin Chen, Jinyang Liu, Wenwei Gu, Yuxin Su, and Michael R. Lyu. "Experience Report: Deep Learning-based System Log Analysis for Anomaly Detection." In: *arXiv:2107.05908 [cs]* (Jan. 2022). arXiv: 2107.05908. URL: http://arxiv.org/abs/2107.05908 (visited on 05/02/2022).

[10] Hetong Dai, Heng Li, Che-Shao Chen, Weiyi Shang, and Tse-Hsun Chen. "Logram: Efficient Log Parsing Using nn-Gram Dictionaries." In: *IEEE Transactions on Software Engineering* 48.3 (Mar. 2022). Conference Name: IEEE Transactions on Software Engineering, pp. 879–892. ISSN: 1939-3520. DOI: 10.1109/TSE.2020.3007554.

[11] Mohamed Cherif Dani, Henri Doreau, and Samantha Alt. "K-means Application for Anomaly Detection and Log Classification in HPC." en. In: *Advances in Artificial Intelligence: From Theory to Practice*. Ed. by Salem Benferhat, Karim Tabia, and Moonis Ali. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 201–210. ISBN: 978-3-319-60045-1. DOI: 10.1007/978-3-319-60045-1_23.

[12] Petros Drineas and Michael W. Mahoney. "On the Nyström Method for Approximating a Gram Matrix for Improved Kernel-Based Learning." In: *The Journal of Machine Learning Research* 6 (Dec. 2005), pp. 2153–2175. ISSN: 1532-4435.

[13] Min Du and Feifei Li. "Spell: Online Streaming Parsing of Large Unstructured System Logs." In: *IEEE Transactions on Knowledge and Data Engineering* 31.11 (Nov. 2019), pp. 2213–2227. ISSN: 1041-4347, 1558-2191, 2326-3865. DOI: 10.1109/TKDE.2018.2875442. URL: https://ieeexplore.ieee.org/document/8489912/ (visited on 03/14/2022).

[14] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. "DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning." en. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. Dallas Texas USA: ACM, Oct. 2017, pp. 1285–1298. ISBN: 978-1-4503-4946-8. DOI: 10.1145/3133956.3134015. URL: https://dl.acm.org/doi/10.1145/3133956.3134015 (visited on 01/03/2022).

[15] Thijs van Ede, Hojjat Aghakhani, Noah Spahn, Riccardo Bortolameotti, Marco Cova, Andrea Continella, Maarten van Steen, Andreas Peter, Christopher Kruegel, and Giovanni Vigna. "DeepCASE: Semi-Supervised Contextual Analysis of Security Events." In: *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2022.

[16] Sarah M. Erfani, Sutharshan Rajasegarar, Shanika Karunasekera, and Christopher Leckie. "High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning." en. In: *Pattern Recognition* 58 (Oct. 2016), pp. 121–134. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2016.03.028. URL: https://www.sciencedirect.com/science/article/pii/S0031320316300267 (visited on 05/12/2022).

[17] Qiang Fu, Jian-Guang Lou, Yi Wang, and Jiang Li. "Execution Anomaly Detection in Distributed Systems through Unstructured Log Analysis." In: *2009 Ninth IEEE International Conference on Data Mining*. Miami Beach, FL, USA: IEEE, Dec. 2009, pp. 149–158. ISBN: 978-1-4244-5242-2. DOI: 10.1109/ICDM.2009.60. URL: http://ieeexplore.ieee.org/document/5360240/ (visited on 03/14/2022).

[18] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st. USA: Addison-Wesley Longman Publishing Co., Inc., 1989. ISBN: 978-0-201-15767-3.

[19] J. C. Gower and G. J. S. Ross. "Minimum Spanning Trees and Single Linkage Cluster Analysis." In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 18.1 (1969). Publisher: [Wiley, Royal Statistical Society], pp. 54–64. ISSN: 0035-9254. DOI: 10.2307/2346439. URL: https://www.jstor.org/stable/2346439 (visited on 05/24/2022).

[20] Frank Grey. "Pulse code communication." Pat. US2632058A. Mar. 1953.

[21] Hossein Hamooni, Biplob Debnath, Jianwu Xu, Hui Zhang, Guofei Jiang, and Abdullah Mueen. "LogMine: Fast Pattern Recognition for Log Analytics." en. In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. Indianapolis Indiana USA: ACM, Oct. 2016, pp. 1573–1582. ISBN: 978-1-4503-4073-1. DOI: 10.1145/2983323.2983358. URL: https://dl.acm.org/doi/10.1145/2983323.2983358 (visited on 03/14/2022).

[22] *Hardware and Architecture (bwForCluster JUSTUS 2) - bwHPC Wiki*. URL: https://wiki.bwhpc.de/e/Hardware_and_Architecture_(bwForCluster_JUSTUS_2) (visited on 03/31/2022).

[23] Wajih Hassan, Shengjian Guo, Ding Li, Zhengzhang Chen, Kangkook Jee, Zhichun Li, and Adam Bates. "NoDoze: Combatting Threat Alert Fatigue with Automated Provenance Triage." In: Jan. 2019. DOI: 10.14722/ndss.2019.23349.

[24] Pinjia He, Jieming Zhu, Shilin He, Jian Li, and Michael R. Lyu. "An Evaluation Study on Log Parsing and Its Use in Log Mining." In: *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. Toulouse, France: IEEE, June 2016, pp. 654–661. ISBN: 978-1-4673-8891-7. DOI: 10.1109/DSN.2016.66. URL: http://ieeexplore.ieee.org/document/7579781/ (visited on 01/08/2022).

[25] Pinjia He, Jieming Zhu, Zibin Zheng, and Michael R. Lyu. "Drain: An Online Log Parsing Approach with Fixed Depth Tree." In: *2017 IEEE International Conference on Web Services (ICWS)*. Honolulu, HI, USA: IEEE, June 2017, pp. 33–40. ISBN: 978-1-5386-0752-7. DOI: 10.1109/ICWS.2017.13. URL: http://ieeexplore.ieee.org/document/8029742/ (visited on 01/10/2022).

[26] Shilin He, Jieming Zhu, Pinjia He, and Michael R. Lyu. "Experience Report: System Log Analysis for Anomaly Detection." In: *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*. Ottawa, ON, Canada: IEEE, Oct. 2016, pp. 207–218. ISBN: 978-1-4673-9002-6. DOI: 10.1109/ISSRE.2016.21. URL: http://ieeexplore.ieee.org/document/7774521/ (visited on 03/04/2022).

[27] S. Hochreiter and J. Schmidhuber. "Long short-term memory." eng. In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.

[28] Matthew Horak, Sowmya Chandrasekaran, and Giovanni Tobar. *NLP Based Anomaly Detection for Categorical Time Series*. Number: arXiv:2204.10483 arXiv:2204.10483 [cs]. Apr. 2022. DOI: 10.48550/arXiv.2204.10483. URL: http://arxiv.org/abs/2204.10483 (visited on 06/27/2022).

[29] Zhen Ming Jiang, Ahmed E. Hassan, Parminder Flora, and Gilbert Hamann. "Abstracting Execution Logs to Execution Events for Enterprise Applications (Short Paper)." In: *2008 The Eighth International Conference on Quality Software*. Oxford, United Kingdom: IEEE, Aug. 2008, pp. 181–186. ISBN: 978-0-7695-3312-4. DOI: 10.1109/QSIC.2008.50. URL: http://ieeexplore.ieee.org/document/4601543/ (visited on 03/14/2022).

[30] Zhen Ming Jiang, Ahmed E. Hassan, Gilbert Hamann, and Parminder Flora. "An automated approach for abstracting execution logs to execution events." en. In: *Journal of Software Maintenance and Evolution: Research and Practice* 20.4 (July 2008), pp. 249–267. ISSN: 1532060X, 15320618. DOI: 10.1002/smr.374. URL: https:

//onlinelibrary.wiley.com/doi/10.1002/smr.374 (visited on 03/14/2022).

[31] L.O. Jimenez and D.A. Landgrebe. "Supervised classification in high-dimensional space: geometrical, statistical, and asymptotical properties of multivariate data." In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 28.1 (Feb. 1998). Conference Name: IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), pp. 39–54. ISSN: 1558-2442. DOI: 10.1109/5326.661089.

[32] Dieter Kranzlmüller. *Supercomputers offline across Europe*. June 2020. URL: https://www.lrz.de/presse/ereignisse/2020-06-03_InterviewSecurityIncident/ (visited on 03/15/2022).

[33] J. B. Kruskal. "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis." en. In: *Psychometrika* 29.1 (Mar. 1964), pp. 1–27. ISSN: 1860-0980. DOI: 10.1007/BF02289565. URL: https://doi.org/10.1007/BF02289565 (visited on 05/16/2022).

[34] Shiyu Liang, Yixuan Li, and R. Srikant. *Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks*. Tech. rep. arXiv:1706.02690. arXiv:1706.02690 [cs, stat] type: article. arXiv, Aug. 2020. DOI: 10.48550/arXiv.1706.02690. URL: http://arxiv.org/abs/1706.02690 (visited on 05/25/2022).

[35] Qingwei Lin, Hongyu Zhang, Jian-Guang Lou, Yu Zhang, and Xuewei Chen. "Log Clustering Based Problem Identification for Online Service Systems." In: *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*. May 2016, pp. 102–111.

[36] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation Forest." In: *2008 Eighth IEEE International Conference on Data Mining*. ISSN: 2374-8486. Dec. 2008, pp. 413–422. DOI: 10.1109/ICDM.2008.17.

[37] Jian-Guang Lou, Qiang Fu, Shengqi Yang, Ye Xu, and Jiang Li. "Mining invariants from console logs for system problem detection." In: *Proceedings of the 2010 USENIX conference on USENIX annual technical conference*. USENIXATC'10. USA: USENIX Association, June 2010, p. 24. (Visited on 05/05/2022).

[38] Scott M Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions." In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. URL: https://papers.nips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html (visited on 07/11/2022).

[39] Zhengping Luo, Tao Hou, Tung Thanh Nguyen, Hui Zeng, and Zhuo Lu. "Log Analytics in HPC: A Data-driven Reinforcement Learning Framework." In: *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. Toronto, ON, Canada: IEEE, July 2020, pp. 550–555. ISBN: 978-1-72818-695-5. DOI: 10.1109/INFOCOMWKSHPS50562.2020.9162664. URL: https://ieeexplore.ieee.org/document/9162664/ (visited on 01/03/2022).

[40] Zhengping Luo, Zhe Qu, Tung Nguyen, Hui Zeng, and Zhuo Lu. "Security of HPC Systems: From a Log-analyzing Perspective." en. In: *ICST Transactions on Security and Safety* 6.21 (Aug. 2019), p. 163134. ISSN: 2032-9393. DOI: 10.4108/eai.19-8-2019.163134. URL: http://eudl.eu/doi/10.4108/eai.19-8-2019.163134 (visited on 01/03/2022).

[41] Laurens van der Maaten, Eric Postma, and H. Herik. "Dimensionality Reduction: A Comparative Review." In: *Journal of Machine Learning Research - JMLR* 10 (Jan. 2007).

[42] Adetokunbo A.O. Makanju, A. Nur Zincir-Heywood, and Evangelos E. Milios. "Clustering event logs using iterative partitioning." en. In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09*. Paris, France: ACM Press, 2009, p. 1255. ISBN: 978-1-60558-495-9. DOI: 10.1145/1557019.1557154. URL: http://portal.acm.org/citation.cfm?doid=1557019.1557154 (visited on 03/14/2022).

[43] Adetokunbo Makanju, A. Nur Zincir-Heywood, and Evangelos E. Milios. "A Lightweight Algorithm for Message Type Extraction in System Application Logs." In: *IEEE Transactions on Knowledge and Data Engineering* 24.11 (Nov. 2012), pp. 1921–1936. ISSN: 1041-4347. DOI: 10.1109/TKDE.2011.138. URL: http://ieeexplore.ieee.org/document/5936060/ (visited on 03/14/2022).

[44] Weibin Meng et al. "Loganomaly: unsupervised detection of sequential and quantitative anomalies in unstructured logs." In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. IJCAI'19. Macao, China: AAAI Press, Aug. 2019, pp. 4739–4745. ISBN: 978-0-9992411-4-1. (Visited on 05/23/2022).

[45] Salma Messaoudi, Annibale Panichella, Domenico Bianculli, Lionel Briand, and Raimondas Sasnauskas. "A search-based approach for accurate identification of log message formats." en. In: *Proceedings of the 26th Conference on Program Comprehension*. Gothenburg Sweden: ACM, May 2018, pp. 167–177. ISBN: 978-1-4503-5714-2. DOI: 10.1145/3196321.3196340. URL: https://dl.acm.org/doi/10.1145/3196321.3196340 (visited on 03/14/2022).

[46] Thomas Minka. "Automatic choice of dimensionality for PCA." In: *Technical Report 514, MIT Media Lab Vision and Modeling Group* (Feb. 2001).

[47] Masayoshi Mizutani. "Incremental Mining of System Log Format." In: *2013 IEEE International Conference on Services Computing*. Santa Clara, CA, USA: IEEE, June 2013, pp. 595–602. ISBN: 978-0-7695-5026-8. DOI: 10.1109/SCC.2013.73. URL: http://ieeexplore.ieee.org/document/6649746/ (visited on 03/14/2022).

[48] Fionn Murtagh. "Clustering in Massive Data Sets." en. In: *Handbook of Massive Data Sets*. Ed. by James Abello, Panos M. Pardalos, and Mauricio G. C. Resende. Massive Computing. Boston, MA: Springer US, 2002, pp. 501–543. ISBN: 978-1-4615-0005-6. DOI: 10.1007/978-1-4615-0005-6_14. URL: https://doi.org/10.1007/978-1-4615-0005-6_14 (visited on 05/16/2022).

[49] Meiyappan Nagappan and Mladen A. Vouk. "Abstracting log lines to log event types for mining software system logs." In: *2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*. Cape Town, South Africa: IEEE, May 2010, pp. 114–117. ISBN: 978-1-4244-6802-7. DOI: 10.1109/MSR.2010.5463281. URL: http://ieeexplore.ieee.org/document/5463281/ (visited on 03/14/2022).

[50] Aum Patil, Amey Wadekar, Tanishq Gupta, Rohit Vijan, and Faruk Kazi. "Explainable LSTM Model for Anomaly Detection in HDFS Log File using Layerwise Relevance Propagation." In: *2019 IEEE Bombay Section Signature Conference (IBSSC)*. July 2019, pp. 1–6. DOI: 10.1109/IBSSC47189.2019.8973044.

[51] Karl Pearson. "Note on Regression and Inheritance in the Case of Two Parents." In: *Proceedings of the Royal Society of London Series I* 58 (Jan. 1895). ADS Bibcode: 1895RSPS...58..240P, pp. 240–242. URL: https://ui.adsabs.harvard.edu/abs/1895RSPS...58..240P (visited on 05/16/2022).

[52] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python." In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[53] Marco A. F. Pimentel, David A. Clifton, Lei Clifton, and Lionel Tarassenko. "A review of novelty detection." en. In: *Signal Processing* 99 (June 2014), pp. 215–249. ISSN: 0165-1684. DOI: 10.1016/j.sigpro.2013.12.026. URL: https://www.sciencedirect.com/science/article/pii/S016516841300515X (visited on 04/26/2022).

[54] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier." In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. New York, NY, USA: Association for Computing Machinery, Aug. 2016,

pp. 1135–1144. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939778. URL: https://doi.org/10.1145/2939672.2939778 (visited on 07/13/2022).

[55] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. en. Ed. by Francis Bach. Adaptive Computation and Machine Learning series. Cambridge, MA, USA: MIT Press, Dec. 2001. ISBN: 978-0-262-19475-4.

[56] Bernhard Schölkopf, Robert C Williamson, Alex Smola, John Shawe-Taylor, and John Platt. "Support Vector Method for Novelty Detection." In: *Advances in Neural Information Processing Systems*. Vol. 12. MIT Press, 1999. URL: https://proceedings.neurips.cc/paper/1999/hash/8725fb777f25776ffa9076e44fcfd776-Abstract.html (visited on 05/12/2022).

[57] Issam Sedki, Abdelwahab Hamou-Lhadj, Otmane Ait-Mohamed, and Mohammed A Shehab. "An Effective Approach for Parsing Large Log Files." In: ().

[58] David H. von Seggern. *CRC Standard Curves and Surfaces with Mathematica, Second Edition*. English. 2nd edition. Boca Raton, Fla.: Chapman and Hall/CRC, Oct. 2006. ISBN: 978-1-58488-599-3.

[59] Yun Shen, Enrico Mariconti, Pierre Antoine Vervier, and Gianluca Stringhini. "Tiresias: Predicting Security Events Through Deep Learning." en. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. Toronto Canada: ACM, Oct. 2018, pp. 592–605. ISBN: 978-1-4503-5693-0. DOI: 10.1145/3243734.3243811. URL: https://dl.acm.org/doi/10.1145/3243734.3243811 (visited on 01/20/2022).

[60] Keiichi Shima. "Length Matters: Clustering System Log Messages using Length of Words." In: *arXiv:1611.03213 [cs]* (Nov. 2016). arXiv: 1611.03213. URL: http://arxiv.org/abs/1611.03213 (visited on 03/14/2022).

[61] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. *Learning Important Features Through Propagating Activation Differences*. arXiv:1704.02685 [cs]. Oct. 2019. DOI: 10.48550/arXiv.1704.02685. URL: http://arxiv.org/abs/1704.02685 (visited on 07/13/2022).

[62] Amit Singhal. "Modern Information Retrieval: A Brief Overview." In: *IEEE Data Engineering Bulletin* 24 (Jan. 2001).

[63] Ayman Taha and Ali Hadi. "Anomaly Detection Methods for Categorical Data: A Review." In: *ACM Computing Surveys* 52 (May 2019), pp. 1–35. DOI: 10.1145/3312739.

[64] Liang Tang, Tao Li, and Chang-Shing Perng. "LogSig: generating system events from raw textual logs." en. In: *Proceedings of the 20th ACM international conference on Information and knowledge management - CIKM '11*. Glasgow, Scotland, UK: ACM Press, 2011, p. 785. ISBN: 978-1-4503-0717-8. DOI: 10.1145/2063576.2063690. URL: http://dl.acm.org/citation.cfm?doid=2063576.2063690 (visited on 03/14/2022).

[65] Andy Turner. *Software usage data on ARCHER2*. Feb. 2022. URL: https://www.archer2.ac.uk/news/2022/02/07/software-usage-data.html (visited on 06/21/2022).

[66] R. Vaarandi. "A data clustering algorithm for mining patterns from event logs." In: *Proceedings of the 3rd IEEE Workshop on IP Operations & Management (IPOM 2003) (IEEE Cat. No.03EX764)*. Kansas City, MO, USA: IEEE, 2003, pp. 119–126. ISBN: 978-0-7803-8199-5. DOI: 10.1109/IPOM.2003.1251233. URL: http://ieeexplore.ieee.org/document/1251233/ (visited on 03/14/2022).

[67] Risto Vaarandi and Mauno Pihelgas. "LogCluster - A data clustering and pattern mining algorithm for event logs." In: *2015 11th International Conference on Network and Service Management (CNSM)*. Barcelona, Spain: IEEE, Nov. 2015, pp. 1–7. ISBN: 978-3-901882-77-7. DOI: 10.1109/CNSM.2015.7367331. URL: http://ieeexplore.ieee.org/document/7367331/ (visited on 03/14/2022).

[68] William M. Waggener. *Pulse Code Modulation Techniques*. en. Springer International Publishing, 1995. ISBN: 978-0-442-01436-0. URL: https://link.springer.com/book/9780442014360 (visited on 05/16/2022).

[69] Y. Wang, J. Wong, and A. Miner. "Anomaly intrusion detection using one class SVM." In: *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004.* June 2004, pp. 358–364. DOI: 10.1109/IAW.2004.1437839.

[70] Sven Weinzierl, Sandra Zilker, Jens Brunk, Kate Revoredo, Martin Matzner, and Jörg Becker. "XNAP: Making LSTM-Based Next Activity Predictions Explainable by Using LRP." en. In: *Business Process Management Workshops*. Ed. by Adela Del Río Ortega, Henrik Leopold, and Flávia Maria Santoro. Lecture Notes in Business Information Processing. Cham: Springer International Publishing, 2020, pp. 129–141. ISBN: 978-3-030-66498-5. DOI: 10.1007/978-3-030-66498-5_10.

[71] Christopher K. I. Williams and Matthias Seeger. "Using the Nyström method to speed up kernel machines." In: *Proceedings of the 13th International Conference on Neural Information Processing Systems*. NIPS'00. Cambridge, MA, USA: MIT Press, Jan. 2000, pp. 661–667. (Visited on 05/17/2022).

[72] Yifan Wu. *DeepLog*. original-date: 2018-12-27T11:53:58Z. June 2022. URL: https://github.com/wuyifan18/DeepLog (visited on 06/08/2022).

[73] Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael Jordan. "Mining Console Logs for Large-Scale System Problem Detection." In: Jan. 2008.

[74] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. "Generalized Out-of-Distribution Detection: A Survey." In: *arXiv:2110.11334 [cs]* (Oct. 2021). arXiv: 2110.11334 version: 1. URL: http://arxiv.org/abs/2110.11334 (visited on 04/26/2022).

[75] Tianbao Yang, Yu-feng Li, Mehrdad Mahdavi, Rong Jin, and Zhi-Hua Zhou. "Nyström Method vs Random Fourier Features: A Theoretical and Empirical Comparison." In: *Advances in Neural Information Processing Systems*. Vol. 25. Curran Associates, Inc., 2012. URL: https://papers.nips.cc/paper/2012/hash/621bf66ddb7c962aa0d22ac97d69b793-Abstract.html (visited on 05/16/2022).

[76] Jieming Zhu, Shilin He, Jinyang Liu, Pinjia He, Qi Xie, Zibin Zheng, and Michael R. Lyu. "Tools and Benchmarks for Automated Log Parsing." In: *arXiv:1811.03509 [cs]* (Jan. 2019). arXiv: 1811.03509. URL: http://arxiv.org/abs/1811.03509 (visited on 01/08/2022).

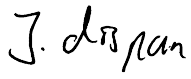
DECLARATION

I herewith declare that I have composed the present thesis myself and without use of any other than the cited sources and aids. Sentences or parts of sentences quoted literally are marked as such; other references with regard to the statement and scope are indicated by full details of the publications concerned. The thesis in the same or similar form has not been submitted to any examination body and has not been published. This thesis was not yet, even in part, used in another examination or as a course performance.

ERKLÄRUNG

Hiermit erkläre ich, dass ich die vorliegende Ausarbeitung selbst und ohne Verwendung anderer als der zitierten Quellen und Hilfsmittel verfasst habe. Wörtlich zitierte Sätze oder Satzteile sind als solche kenntlich gemacht; andere Hinweise zur Aussage und zum Umfang sind durch vollständige Angaben zu den betreffenden Publikationen gekennzeichnet. Die Ausarbeitung wurde in gleicher oder ähnlicher Form keiner Prüfungsstelle vorgelegt und ist nicht veröffentlicht worden. Diese Arbeit wurde noch nicht, auch nicht teilweise, in einer anderen Prüfung oder als Lehrveranstaltungsleistung verwendet.

Ulm, July 2022



Juri Dispan

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede and Ivo Pletikosić. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/

Thank you very much for your feedback and contribution.

Final Version as of July 16, 2022 (classicthesis v4.6).