An Approach to Examine and Recognize Anomalies on Cloud Computing Platforms with Machine Learning Concepts

MPGK Jayaweera[1], WMCJT Kithulwatta[2,3#], and RMKT Rathnayaka[3,4]

[1]Department of Computing and Information Systems, Faculty of Computing, Sabaragamuwa University of Sri Lanka, Belihuloya, Sri Lanka

[2]Department of Information and Communication Technology, Faculty of Technological Studies, Uva Wellassa University of Sri Lanka, Badulla, Sri Lanka

[3]Research Center for Grey Systems and Uncertainty Analysis (GSUSL), Department of Physical Sciences & Technology, Faculty of Applied Sciences, Sabaragamuwa University of Sri Lanka, Belihuloya, Sri Lanka.

[4]Department of Physical Sciences and Technology, Faculty of Applied Sciences, Sabaragamuwa University of Sri Lanka, Belihuloya, Sri Lanka

#chirantha@uwu.ac.lk

ABSTRACT Cloud computing is one of the most rapidly growing computing concepts in today's information technology world. It connects data and applications from various geographical locations. A large number of transactions and the hidden infrastructure in cloud computing systems have presented the research community with several challenges. Among these, maintaining cloud network security has emerged as a major challenge. It is critical to address issues in the quickly changing cloud computing market in order to guarantee that businesses can fully utilize cutting-edge technology, uphold strong security protocols, and maximize operational effectiveness. Businesses that successfully navigate these obstacles can maintain their competitiveness in a dynamic digital ecosystem by improving scalability, leveraging the flexibility provided by the cloud, and adapting to technological changes with ease. Anomaly detection (or outlier detection) is the identification of unusual or suspicious data that differs significantly from the majority of the data. Research on anomaly detection in cloud network data is crucial because it enables businesses to more rapidly and efficiently recognize potential security threats, network performance concerns, and other issues. Recently, machine learning methods have demonstrated their efficacy in anomaly detection. This research aimed to introduce a novel hybrid model for anomaly detection in cloud network data and to investigate the performance of this model in comparison to other machine learning algorithms. The research was conducted with the UNSW-NB15 anomaly dataset and employed various feature selection and pre-processing techniques to prepare the data for model training. The hybrid model was built using a combination of Random Forest and SVM algorithms and the process was evaluated using metrics such as F1-Score, Recall, Precision, and Accuracy. The result showed that the hybrid model has 94.23% accuracy and a total time of 109.92s which is the combination of the train time of 100.45s and prediction time of 9.47s. The limitations of the study include the class imbalance problem in the dataset and the lack of real-world applications for testing. The research suggests future work in the application of hybrid models in anomaly detection and cloud network security and the need for further investigation into the potential benefits of such models.

KEYWORDS: Anomaly Detection, Cloud Computing, Machine Learning, Monitoring

I. INTRODUCTION

The technology of cloud computing virtualization provides efficient resources for end users. The characteristics of cloud computing include manageability, scalability, and availability. In addition, cloud computing has the advantages of economy, on-demand service, convenience, universality, multi-tenancy, flexibility, and stability [27]. Cloud computing mainly provides three service delivery models and four development patterns: infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS), public cloud, private cloud, hybrid cloud, community cloud, and virtual private cloud [29]. Today, cloud computing has integrated with other computing technologies like fog computing, grid computing, Docker containers, IoT, etc [28], [30], [31]. Cloud security is one of the most important aspects of cloud computing because it involves thousands of user transactions, information, and communication. The availability, integrity, and confidentiality of cloud computing platforms or services must be ensured to provide secure cloud computing platforms/services. Security vulnerabilities and challenges arise from the usage of cloud computing services. Currently, cloud computing models are the primary source of these challenges and vulnerabilities [32]. The intruders exploit the weakness of cloud models in accessing the users' private data, by attacking the processing power of computer systems [3].

An anomaly is an observation that differs so significantly from previous observations that it raises suspicion that it was caused by a distinct mechanism. It's frequently a sign of something unexpected or problematic happening. Anomaly detection is the identification of rare item events or observations that raise suspicion by differing significantly from the majority of data. They are slightly or majorly different from the majority of data and anomaly detection can help to find outliers and problems in data. In other words, anomalies are data points or patterns in a dataset that differ significantly from the expected or usual behavior. These anomalies can be produced by several things, including measurement errors, sensor malfunction, data corruption, or system failure, and they can happen spontaneously or as a result of mistakes in data collecting or processing. Finding these odd data points or patterns in a dataset that are frequently a sign of a deeper issue or problem is called anomaly detection. A dataset may contain a variety of anomalies, including point anomalies that only affect a single instance of data, contextual anomalies that only occur under certain circumstances, collective anomalies that involve multiple data points that behave similarly, and collective contextual anomalies that involve multiple data points that behave similarly only under certain circumstances. In several fields, including network intrusion detection, fraud detection, defect detection, and monitoring of complex systems, anomaly detection is a critical step [6][7].

Finding strange or unexpected data points or patterns in a dataset is the process of anomaly detection. Anomalies can be found using a variety of techniques, such as statistical techniques, clustering, classification, deep learning, distance-based techniques, and time-series-based techniques. Quantiles, standard deviation, and other statistical metrics are used in statistical procedures to detect data points that significantly depart from the norm. Anomalies are data points that do not belong to any cluster and are grouped by clustering algorithms. To categorize new data points as normal or abnormal, classification algorithms are trained on labeled data. Deep learning algorithms discover the underlying structure of the data and the location of data points that deviate from this pattern to find anomalies. Measures of the distance between data points are used by distance-based algorithms to detect data points that are far away from other ones. To identify anomalies, time-series-based algorithms employ techniques like moving average, exponential smoothing, ARIMA, and Prophet. A combination of several methods is frequently used to boost the robustness and accuracy of anomaly detection. The choice of the method relies on the nature of the data and the particular requirements of the application [2].

The connection between cloud network data and anomaly detection is it provides an analysis of unusual activities, and unexpected activities through anomaly detection algorithms. Effective monitoring and security procedures are becoming more and more important as more businesses shift their data and apps to the cloud. Anomaly detection can aid in the identification of potential security vulnerabilities and performance problems, enabling businesses to take preventative action to lessen these risks and their effects on operations. A wealth of knowledge regarding the functionality, security, and use of cloud-based systems is contained in cloud network data. Log files, performance indicators, network traffic, and other sorts of data are examples of this data. These data can be examined by anomaly detection algorithms to find patterns or anomalies that point to issues with the network or its elements, such as security breaches, performance issues, or other suspicious activities. Additionally, anomaly detection in cloud network data aids organizations in conforming to several legal standards about the security, privacy, and integrity of their data. Automated anomaly detection is a crucial tool for preserving the security and dependability of cloud-based systems since it gets more challenging to manually detect and react to anomalies as more data is stored and processed in the cloud [1][6].

II. MOTIVATION

The goal of anomaly detection is to use approaches that can discover relevant anomalies in data without producing a large number of false positives.

Cloud security is one of the most important aspects of cloud computing because it involves thousands of user transactions and information. The availability, integrity, and confidentiality of cloud computing platforms or services must be ensured to provide secure cloud computing platforms/services. Security vulnerabilities and challenges arise from the usage of cloud computing services. Currently, cloud computing models are the primary source of these challenges and vulnerabilities. The intruders exploit the weakness of cloud models in accessing the users' private data, by attacking the processing power of computer systems [8][10].

The detection of anomalies in data has a long history and a wide range of applications. An anomaly or outlier is an observation that differs so significantly from other observations that it raises the possibility that it was generated by a different mechanism. It can also be defined as an outlier observation that shows up to deviate significantly from the rest of the sample members in which it occurs.

Due to the complexity of modern systems, highly available cloud service requirements in a cloud environment are difficult to guarantee and can thus only be ensured with great effort. As a result of these trends, there is an increasing demand for intelligent applications that automatically detect anomalies and provide suggestions for solving or at least mitigating problems so that a negative impact on service quality does not cascade. What constitutes an anomaly in each case is determined by the sample and the methodology. Anomalies are classified into three types in general: Anomalies can be classified into three types namely *point anomalies*, *collective anomalies*, and *contextual anomalies*. There are primarily three approaches for detecting anomalies (machine learning, deep learning, and statistical approach). After reviewing previous studies, the study discovered that machine learning outperforms the other two methods in detecting abnormalities. Although the practice mentioned above provides ways to detect anomalies in a dataset. The research community still knows little about which is the most suitable algorithm for detecting anomalies within a cloud environment. The author is motivated to close this gap of knowledge and try to use a specific machine learning algorithm to detect anomalies using a data set. After analyzing the team can decide whether this algorithm is suitable or not for detecting anomalies within a cloud network [4][5][10][24].

A. Significance of the study

It is critical to address anomaly problems in cloud computing platforms because they have an immediate effect on the security and dependability of digital infrastructure. Anomalies can jeopardize data integrity and result in breaches and unauthorized access, regardless of whether they are caused by malevolent activity or system malfunctions. It is imperative to promptly identify and address irregularities in order to assurance the unceasing procedure of cloud-based services, protect confidential data, and uphold user confidence. In the quickly changing world of digital technology, proactive tactics for anomaly management not only improve the general resilience of cloud systems but also help to build a strong cybersecurity foundation.

B. Research objectives

To keep a clear direction within the research study, below research objectives (RO) were made.

RO1: *To introduce a novel hybrid model and compare the performance of the hybrid model to other machine learning models, such as single-algorithm models, in detecting anomalies in cloud network data.*

RO2: *To look into how different algorithmic combinations affect, how well the hybrid model performs while looking for anomalies in data from cloud networks.*

RO3: *To investigate how well the novel hybrid model handles various data kinds and investigate how various feature selection and pre-processing techniques affect the novel hybrid model's ability to detect anomalies in cloud network data.*

C. Contribution of the paper

By presenting a novel hybrid model that combines Random Forest (RF) and Support Vector Machine (SVM) techniques, the research significantly advances the subject of anomaly detection in cloud network data. This hybrid method offers a unique solution for anomaly detection problems, marking a significant deviation from the traditional application of single-algorithm models. In contrast to stand-alone RF models, the hybrid model aims to improve detection robustness and accuracy by combining the advantages of both RF and SVM.

One of the primary contributions is the extensive testing of the proposed hybrid model against multiple machine learning methods, including multiple RF and SVM configurations and an MLP model.

III. RESEARCH METHOD

Machine learning models such as Isolation Forests, One-Class SVM, and Autoencoders are frequently employed in anomaly identification. These models are significant because, in the absence of labeled training data, they are highly effective at identifying patterns and abnormalities in a variety of datasets. One-Class SVM is skilled at identifying outliers in high-dimensional spaces, Autoencoders learn intrinsic data representations, and Isolation Forests effectively isolate anomalies by building random decision trees. These tools are useful for detecting deviations from normal patterns in a variety of applications, including cybersecurity and system monitoring. This approach involves building up a hybrid model combining SVM and random forest algorithms. This research used the UNSW-NB15 dataset for the study. The methodology is concluded here after identifying and analyzing the comparisons between different algorithm models.

The combined strengths of Random Forest (RF) and Support Vector Machine (SVM) in handling different areas of anomaly detection in cloud network data led to their selection for the hybrid model. As an ensemble learning technique, RF is well-known for its stability and resistance to overfitting. It is particularly good at capturing complicated relationships within data. However, SVM is good at managing non-linear patterns by determining optimal decision boundaries, especially when employing non-linear kernels. The hybrid model combines the power of SVM's ability to identify distinct decision boundaries with the versatility of RF's modeling techniques to attempt to capitalize on the differences between the two approaches.

A. Gather Relevant Data

The UNSW NB15 dataset was used in this research study to study the usage of cloud network data to detect anomalies. The loading of the UNSW NB15 dataset was the first stage in the study procedure. The dataset included network traffic information that can be used to develop and test anomaly detection methods.

B. Pre-processing and Feature Selection

Preprocessing the dataset came after the data had been loaded. Make sure the data is prepared for usage in the feature selection process, this may involve cleaning and normalizing it. The process of choosing a subset of the features in a dataset that is most important for anomaly detection is known as feature selection. Techniques like correlation analysis or mutual information can be used for this.

C. Train the Model

The process of training models using the chosen features followed the feature selection phase. The dataset was divided into training and testing sets, and several anomaly detection models were trained and evaluated using these sets. In this study, models like Random Forest (Estimators = 100), Random Forest (Estimators = 50), Random Forest (Estimators = 150), SVM (Kernel - rbf, gamma-scale), SVM (Kernel - sigmoid, gamma-scale), SVM (Kernel - poly, gamma-scale), and a hybrid model that combined the best features of Random Forest and SVM models were used.

D. Analyze the Model

A comparison was done once the models had been trained and assessed to see which model performed the best on the UNSW NB15 dataset. The evaluation measures used in the comparison included accuracy, precision, recall, and F1-score. The comparison's findings were used to evaluate the performance of various models for finding anomalies in cloud network data.

E. Summary of the Methodology

In conclusion, the study used the UNSW NB15 dataset to evaluate the hybrid model through preprocessing, feature selection, model training using random forest models, SVM models, and a hybrid model, and comparing all the models to determine which is the best.

This research study recommended the following methodology step-wise to better understand:

- ***Data collection:*** The UNSW-NB15 anomaly dataset was used.
- ***Data preprocessing and feature selection:*** The data was preprocessed and features were selected for the training and testing sets.
- ***Model training:*** The model was trained using Random Forest and SVM algorithms [34].
- ***Hybrid model construction:*** A novel hybrid model was built due to their higher accuracy and other aspects.
- ***Model evaluation:*** The performance of the novel hybrid model was evaluated and compared to that of other machine learning models, such as single-algorithm models, Random Forest (Estimators = 100), Random Forest (Estimators = 50), Random Forest (Estimators = 150) and SVM (Kernel - rbf, gamma - scale), SVM (Kernel - sigmoid, gamma - scale), SVM (Kernel - poly, gamma - scale), and MLP(ANN) model.
- ***Data analysis:*** The results were analyzed and discussed in terms of the research objectives, including the impact of various algorithmic combinations on the performance of the hybrid model, the performance of the hybrid model compared to that of single-algorithm models, and the potential future research pathways for the application of hybrid models in anomaly detection and cloud network security.
- ***Limitations and recommendations:*** The limitations of the study were identified as the class imbalance problem in the dataset and future research recommendations were made to address the class imbalance problem in the dataset, further investigate the potential of hybrid models in anomaly detection and cloud network security, and investigate the rate of false positives and false negatives, computational resources and the ease of understanding of the hybrid model.

IV. DESIGN, IMPLEMENTATION, AND ANALYSIS OF THE RESULTS

This section describes the model's design comprehensively with the model's basic architecture and the proposed model's workflow. Here several diagrams are presented and discussed to explain model functions. The technologies, algorithms, special methods, and functions used in implementation were defined in this section. Further, this section discussed the findings of the phases of implementation.

A. Gathering the Relevant Data Set

The UNSW-15 dataset was a good option for the study since it offers a thorough assessment of the proposed approach's capacity to recognize various sorts of attacks. The dataset included both known and undiscovered attack types, allowing for the evaluation of the approach's capacity to identify several distinct attacks. Additionally, a thorough evaluation of the performance of the approach is possible due to the dataset's size and abundance of instances. The dataset also included real-world network traffic statistics, enhancing the relevance and applicability of the study's findings to real-world circumstances. Furthermore, the performance of the proposed technique in other current ways can be easily compared thanks to the UNSW-15 dataset, which is a well-known and often-used dataset in the field of network intrusion detection. A fair assessment of the performance of the suggested strategy is possible thanks to the dataset's balance, which includes a sufficient number of both normal and attack occurrences. The dataset is additionally current and contains up-to-date network traffic data, increasing its applicability to current real-world settings.

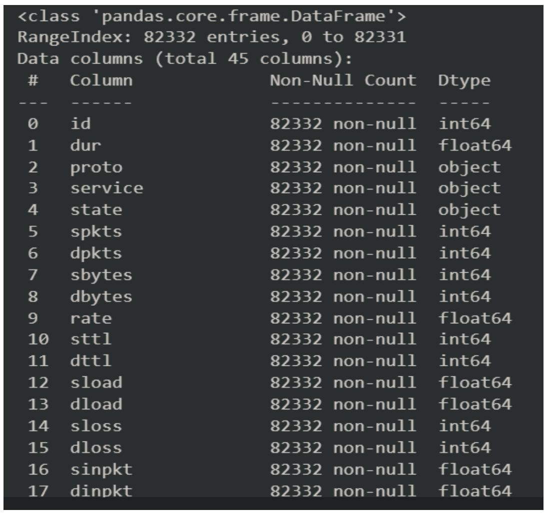
Loading Data

First, the author read a CSV file and created a DataFrame object in Python using the Pandas module. In particular, it loads the data from the CSV file at the supplied file path using the ***read csv***() function and stores it in the variable ***'df'***. The DataFrame is a strong and adaptable data structure that makes it simple to manipulate and analyze data presented in tabular form. The author then used to show the data frame's first five rows. This can be helpful for rapidly verifying that the data has been loaded properly and previewing the contents of the DataFrame. Table 1 presents the whole content for the loaded data in the study.

Table 1: The content of the loaded data

index	id	dur	proto	service	state	spkts	dpkts	sbytes
0	1	1.10E-05	udp	-	INT	2	0	496
1	2	8.00E-06	udp	-	INT	2	0	1762
2	3	5.00E-06	udp	-	INT	2	0	1068
3	4	6.00E-06	udp	-	INT	2	0	900
4	5	1.00E-05	udp	-	INT	2	0	2126

Further, figure 1 demonstrates the metadata of the loaded data comprehensively.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 82332 entries, 0 to 82331
Data columns (total 45 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   id                82332 non-null  int64
 1   dur               82332 non-null  float64
 2   proto             82332 non-null  object
 3   service           82332 non-null  object
 4   state             82332 non-null  object
 5   spkts             82332 non-null  int64
 6   dpkts             82332 non-null  int64
 7   sbytes            82332 non-null  int64
 8   dbytes            82332 non-null  int64
 9   rate              82332 non-null  float64
 10  sttl              82332 non-null  int64
 11  dttl              82332 non-null  int64
 12  sload             82332 non-null  float64
 13  dload             82332 non-null  float64
 14  sloss             82332 non-null  int64
 15  dloss             82332 non-null  int64
 16  sinpkt            82332 non-null  float64
 17  dinpkt            82332 non-null  float64
```

Figure 1: Information of the loaded data

Furthermore, table 2 displays a tabular description of the loaded data.

Table 2: Description of the loaded data

	id	dur	spkts	dpkts	sbytes	dbytes
count	82332	82332	82332	82332	82332	82332
mean	41166.5	1.006756	18.66647	17.54594	7993.908	13233.79
std	23767.35	4.710444	133.9164	115.5741	171642.3	151471.5
min	1	0	1	0	24	0
25%	20583.75	8.00E-06	2	0	114	0
50%	41166.5	0.014138	6	2	534	178
75%	61749.25	0.71936	12	10	1280	956
max	82332	59.99999	10646	11018	14355774	14657531

B. Data Pre-Processing and Feature Selection

Pre-processing the data is a crucial stage in the methodology of the study since it guarantees that the UNSW-15 dataset is in a format that the model can use. The UNSW-15 dataset's data pre-processing may entail several important procedures.

Removal of Irrelevant Columns

To remove particular columns from the DataFrame, the author used the ***DataFrame function drop()***. It starts by making a list of the columns that should be deleted, author dropped "id" and "attack cat." The ***drop()*** method was then called with this list as its first argument. When axis=1 is used as the second parameter, pandas is instructed to remove the columns. The third parameter, inplace=True, is set to mean that the original DataFrame should be used for the operation. As a result, this will delete the columns "id" and "attack cat" from the DataFrame "df," update the original DataFrame to reflect the deletion of those columns, and return no new DataFrame.

Clamping

Clamping is a preprocessing method for reducing the range of values in a dataset. It is usually applied to stop outliers from skewing the results of subsequent processes, including statistical analysis or machine learning. Putting a maximum and minimum threshold for the values in a dataset entails "clamping," or setting any values outside of this range to the threshold value closest to them. This can help prepare data for analysis and clean it, which can also help to increase the precision and stability of machine learning models. In this research, the author prunes extreme values to make distributions less skewed. Features are reduced to the 95th percentile when their maximum values exceed 10 times the median value.

In summary, the author produces descriptive statistics for the numeric columns after first filtering the original DataFrame to only include those columns. The outcome is a new DataFrame that gives an overview of the distribution of data in the original DataFrame's numerical columns. Then, the author determines whether the maximum value of any column is bigger than 10 times the median value and greater than 10, and if it is, it replaces the values in that column with the 95th percentile's value if they are higher, else the value is left alone. If the DEBUG setting is set to 1, each column will print some information; otherwise, nothing will be printed.

Apply the log function on skewed-right numerical numbers

The author added one to each value before applying the natural logarithm to the values of each column in the numeric DataFrame df numeric if that column's minimum value is zero and there are more than 50 unique values in that column. This avoids using the undefined log(0). If the DEBUG setting is set to 1, each column will print some information; otherwise, nothing will be printed.

Reduce labels in categorical features

Reducing the cardinality of features to 5 or 6. Take the top 5 occurring labels in the feature as labels and set the remainder to '-' as seldom used labels. In this, the author determines whether each given column has more than six distinct values. If this is true for any given column, the value in that column is replaced with a '-' if it is not one of the most frequent values there; otherwise, it is left alone. If the DEBUG setting is set to 1, each column will print some information; otherwise, nothing will be printed. The scenario for reducing the labels in categorical features is presented in Table 3 below.

Table 3: Reduce labels in categorical features

index	proto	service	state
count	82332	82332	82332
unique	131	13	7
top	tcp	-	FIN
freq	43095	47153	39339

Best Features

Univariate statistical tests to determine which features best predict the target feature. Utilizing Python's scikit-learn module, choose the best features from a DataFrame, and display the outcomes. For feature selection, it first imports the required modules SelectKBest and chi2. The SelectKBest class is then created with the chi2 scoring function and the input k='all', instructing it to select all characteristics. The best features object is fitted to the input data by taking into account the goal variable y and the input data X. The scores and feature names are concatenated to produce a new DataFrame. The new DataFrame's columns now go by the names "feature" and "score." The DataFrame is then sorted based on the feature scores, and a bar chart is generated to show the top 21 features.

Figure 2 presents a bar chart for the top features.

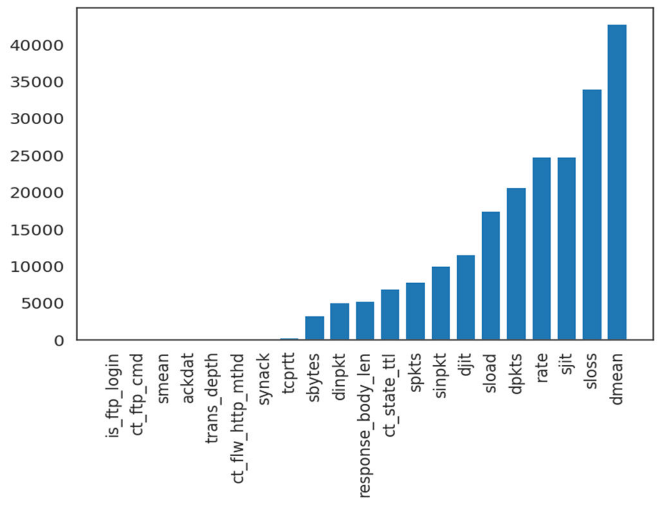


Figure 2: A bar chart for the top features

Encoding Categorical Features

One-hot encoding is used. None of the categorical features are ordinal. In this study, the author tried picking particular rows and columns from the original DataFrame "df" to create two new variables, "X" and "y," and it is displaying the first five rows of the DataFrame "X". The particular encoding categorical features are presented in table 4.

Table 4: Encoding Categorical Features

index	dur	proto	service	state	spkts	dpkts	sbytes
0	1.10E-05	udp	-	INT	0.693147	0	6.206575927
1	8.00E-06	udp	-	INT	0.693147	0	7.474204806
2	5.00E-06	udp	-	INT	0.693147	0	6.97354302
3	6.00E-06	udp	-	INT	0.693147	0	6.802394763
4	1.00E-05	udp	-	INT	0.693147	0	7.661997559

After that, the author used the "OneHotEncoder" class to apply the One-Hot-Encoding approach to columns 1, 2, and 3 of the DataFrame X while leaving the other columns alone to be handled by the "ColumnTransformer" class. Additionally, a ***numpy*** array was being created from the encoded DataFrame.

After that, unique values of a few columns in a DataFrame are extracted using Python's Pandas package, and they are then inserted into a list of feature names in a certain order. These three for loops iterate over the distinct values of the 'state', 'service', and 'proto' columns of the DataFrame 'df' and add them to the list of feature names in reverse order while excluding the first element. To facilitate additional analysis or model training, the author has included the distinctive values from these columns in the list of feature names.

C. Modeling and Evaluation

This entails training the SVM and random forest parts of the hybrid algorithm, training and test split, standardizing continuous features, training with random forest and SVM separately, and implementing a hybrid model and comparison.

Train Set Split

Using stratified sampling, the data in this part are divided into training and test sets. The input data "X" and the target variable "y" are divided into two datasets: the training set and the testing set, using the scikit-learn library. The dataset, the percentage of the dataset that should be given to the testing set, a random seed to assure repeatability, and the stratification of the data are all inputs to the "train test split" function, which was employed by the author in this study. The 'X train', 'X test', 'Y train', and 'Y test' datasets will be utilized for the models' training and testing, respectively. This split is an essential stage in the machine learning process because it enables the author to predict how well the model will perform on new data and avoid overfitting.

Standardize continuous features

The continuous features are scaled using a standard scaler to ensure that they are all in the same size order. To normalize the numerical features of the training and test datasets, use the scikit-learn library. It makes use of the "StandardScaler" class from the library's "preprocessing" module to scale the numerical features to unit variance and standardize them by removing the mean. By utilizing the 'fit transform()' method, which first fits the scaler to the data before transforming it, it generates an instance of the 'StandardScaler' class and applies it to the numerical characteristics of the training dataset. The test dataset's numerical features are then normalized using the transform() method using the same instance of the scaler. The efficiency and stability of the models can be enhanced by normalizing the numerical features, which is a crucial step because many Machine Learning methods are sensitive to the scale of the data.

Following that, the author constructs an empty dataframe called "model performance" and imports much time-related, performance-related metrics from the scikit-learn library. The dataframe comprises seven columns, including "Accuracy," "Recall," "Precision," "F1-Score," "train time," "pred time," and "total time." The time-related functions from the Python library were used to assess the time spent on training and prediction of the model, and the imported performance metrics from the scikit-learn library were used later to evaluate the performance of a machine learning model. This dataframe was used to store these measurements for later examination.

Random Forest

The author is making predictions on the test dataset while training a Random Forest classifier in Python using the scikit-learn library. It generates an instance of the class, imports the RandomForestClassifier class from the library's ensemble module, and then trains the model using the training dataset. Using a time module also keeps track of the length of time spent on training and prediction. After making the predictions, the author made predictions on the test dataset using the trained model's prediction approach. The Random Forest Classifier is an ensemble method that uses averaging to increase predicted accuracy and reduce over-fitting. It trains numerous decision trees on different subsamples of the dataset. The summary data for the Random Forest 100 prediction is presented in Figure 3.

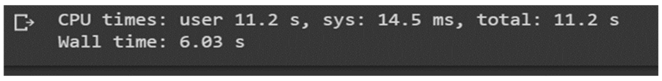

```
CPU times: user 11.2 s, sys: 14.5 ms, total: 11.2 s
Wall time: 6.03 s
```

Figure 3: Prediction (Random Forest 100)

On a test dataset, this algorithm assessed how well a Random Forest classifier model performs. It computes several performance metrics, including accuracy, recall, precision, and f1-score, using the scikit-learn module. It also determines how long training and prediction will take. Additionally, it prints the times and performance indicators in a more readable manner. The results are then saved in a dataframe for future study. Figure 4 presents the model performance data (Random Forest 100).

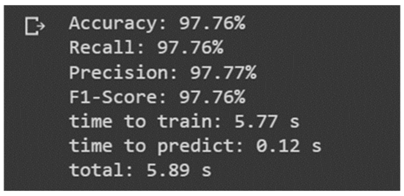

```
Accuracy: 97.76%
Recall: 97.76%
Precision: 97.77%
F1-Score: 97.76%
time to train: 5.77 s
time to predict: 0.12 s
total: 5.89 s
```

Figure 4: Model Performance (Random forest 100)

On a test dataset, the author plotted a confusion matrix for a Random Forest classifier model. A table called the confusion matrix is used to describe how well a classification algorithm performs. The matrix is generated using the scikit-learn library's plot confusion matrix function, which takes the model, test data, and true labels as inputs. The plot is displayed with a white background and a chosen size of 5 x 5. Figure 5 presents the Random Forest 100 Confusion Matrix.

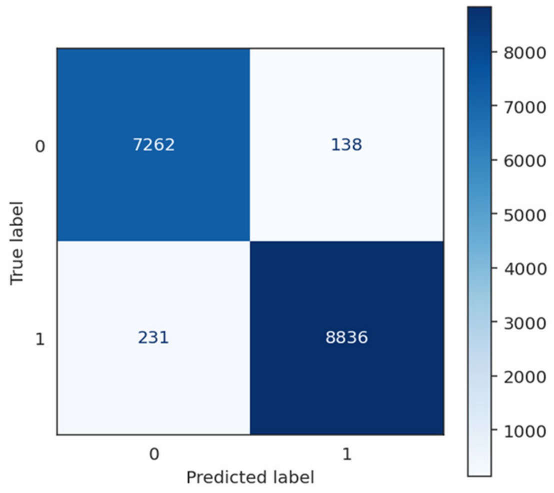


Figure 5: Confusion Matrix (Random Forest 100)

The top 20 features of the Random Forest classifier model are then plotted according to their importance using the scikit-learn package, and the plot was displayed in a 10 x 10-inch format with a white background. Additionally, it removed the top and right spines from the plot and flipped the y-axis such that the most significant feature was at the top.

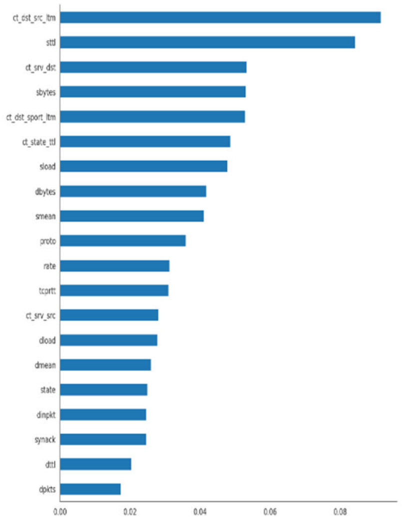
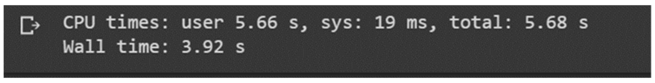


Figure 6 presents the top 20 features of Random Forest for 100 estimators.

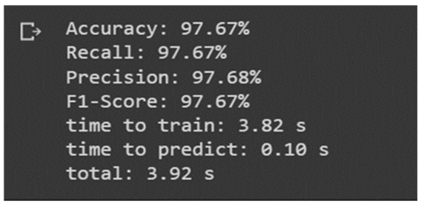
The author used the above concepts for Random Forest (Estimators = 50) and Random Forest (Estimators = 150) and got the following results. Figure 7 presents the prediction for Random Forest 50.



```
CPU times: user 5.66 s, sys: 19 ms, total: 5.68 s
Wall time: 3.92 s
```

Figure 7: Predictions (Random Forest 50)

Further, figure 8 presents the performance of the model for Random Forest 50.



```
Accuracy: 97.67%
Recall: 97.67%
Precision: 97.68%
F1-Score: 97.67%
time to train: 3.82 s
time to predict: 0.10 s
total: 3.92 s
```

Figure 8: Model Perfromance (Random Forest 50)

Figure 9 presents the Confusion Matrix for the Random Forest 50.

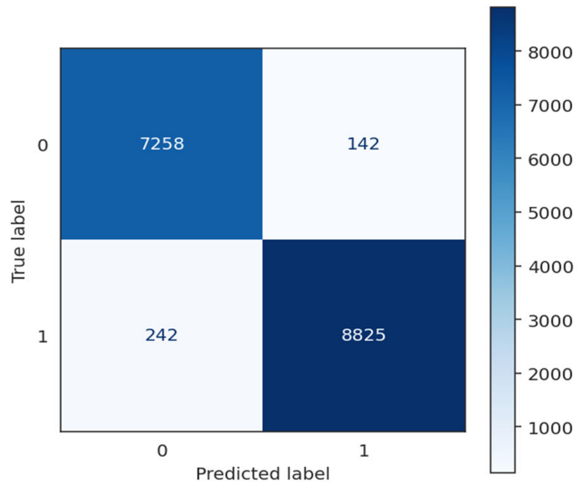
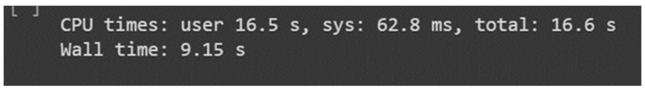


Figure 9: Confusion Matrix(Random Forest 50)

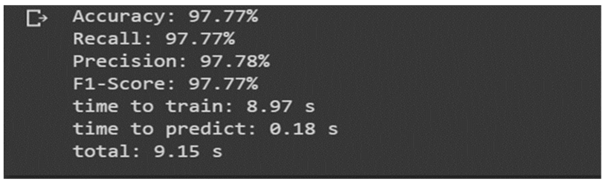
As presented in Figure 10, the Model Prediction for Radom Forest 150 is available.



```
CPU times: user 16.5 s, sys: 62.8 ms, total: 16.6 s
Wall time: 9.15 s
```

Figure 10: Predictions (Random Forest 150)

Further, figure 11 presents the model performance for Random Forest 150.



```
Accuracy: 97.77%
Recall: 97.77%
Precision: 97.78%
F1-Score: 97.77%
time to train: 8.97 s
time to predict: 0.18 s
total: 9.15 s
```

Figure 11: Model Performance (Random Forest 150)

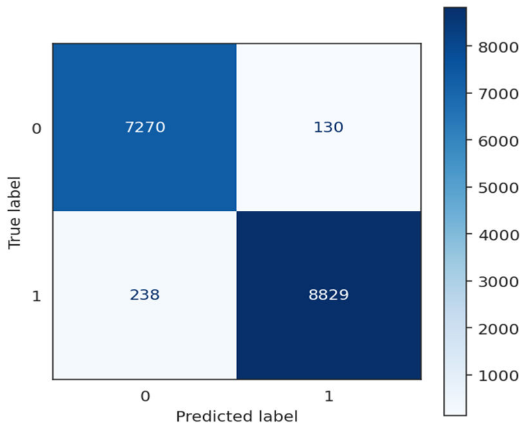


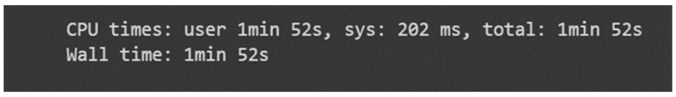
Figure 12: Confusion Matrix (Random Forest 150)

According to Figure 12, the Confusion Matrix for Random Forest 150 was presented.

SVM

After calculating results using the Random Forest algorithm, the author tried to apply these logics to the SVM algorithm.

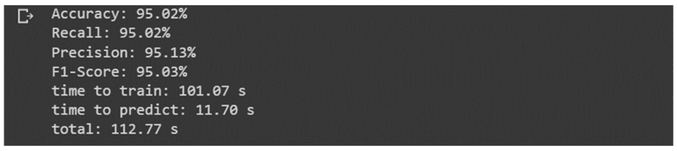
The scikit-learn library was used by the author to train and test a Support Vector Machine (SVM) classification model. The kernel = 'rbf' and gamma ='scale', which were parameters of the RBF kernel, were used to fit the model to the training data. This generated an instance of the SVM class. On the test dataset, it used the trained model to generate predictions. It also kept track of how long training and prediction take. The cell's execution time was also gauged. The gamma parameter was automatically scaled by ***1 / (n_features * X.var()),*** where n _features were the total number of features and ***X.var()*** was the variance of the training dataset, using the 'rbf' kernel and gamma ='scale' in the code. Figure 13 presents the SVM predictions for (Kernal – rbf, gamma – scale).



```
CPU times: user 1min 52s, sys: 202 ms, total: 1min 52s
Wall time: 1min 52s
```

Figure 13: SVM (Kernel - rbf, gamma - scale) Predictions

Then, using accuracy, recall, precision, F1-score, time for training, time for prediction, and total time, the author assessed the SVM model's performance on the test dataset and recorded the findings in a dataframe for later use. The model's performance is then recorded in a dataframe for future use, and the assessment metrics and time measurements are written out in a human-readable format. The SVM model performance for (Kernal – rbf, gamma – scale) was presented in Figure 14.



```
Accuracy: 95.02%
Recall: 95.02%
Precision: 95.13%
F1-Score: 95.03%
time to train: 101.07 s
time to predict: 11.70 s
total: 112.77 s
```

Figure 14: SVM (Kernel - rbf, gamma - scale) Model Performance

The predictions provided by the SVM model on the test dataset are then plotted as a confusion matrix by the author. The model, "X test," and "y test" were used as input arguments for the "plot confusion matrix" function, which creates the confusion matrix. The figurine has a 5.5-inch height and a blue color scheme. The ***'plt.show()'*** function was used to show the plot. It was used to visually assess the model's performance and determine which class the model successfully predicted and which class it incorrectly forecasted. Figure 15 presents the Confusion Matrix for SVM (Kernel - rbf, gamma - scale).

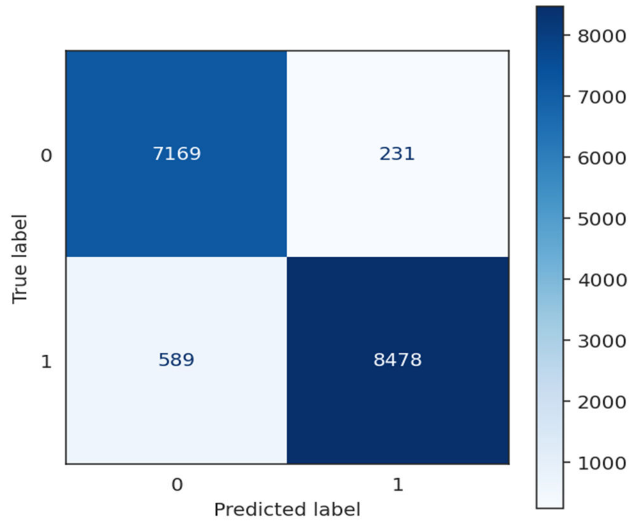
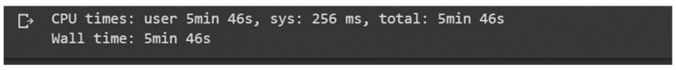


Figure 15: Confusion Matrix for SVM (Kernel - rbf, gamma - scale)

After that, the author used the above concepts to SVM (Kernel - sigmoid, gamma – scale) and SVM (Kernel - poly, gamma - scale). The following results were obtained from the study.

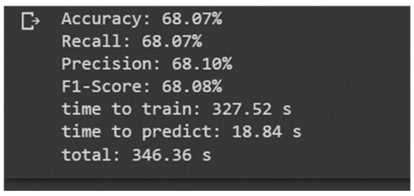
The SVM prediction for (Kernel - sigmoid, gamma - -scale) was presented in Figure 16.



```
CPU times: user 5min 46s, sys: 256 ms, total: 5min 46s
Wall time: 5min 46s
```

Figure 16: SVM (Kernel - sigmoid, gamma - scale) Predictions

The SVM method performance for (Kernel - sigmoid, gamma - -scale) was presented in Figure 17.



```
Accuracy: 68.07%
Recall: 68.07%
Precision: 68.10%
F1-Score: 68.08%
time to train: 327.52 s
time to predict: 18.84 s
total: 346.36 s
```

Figure 17: SVM method performance for (Kernel - sigmoid, gamma - scale)

Further, the Confusion Matrix for SVM (Kernel- - sigmoid, gamma - scale) was presented in Figure 18.

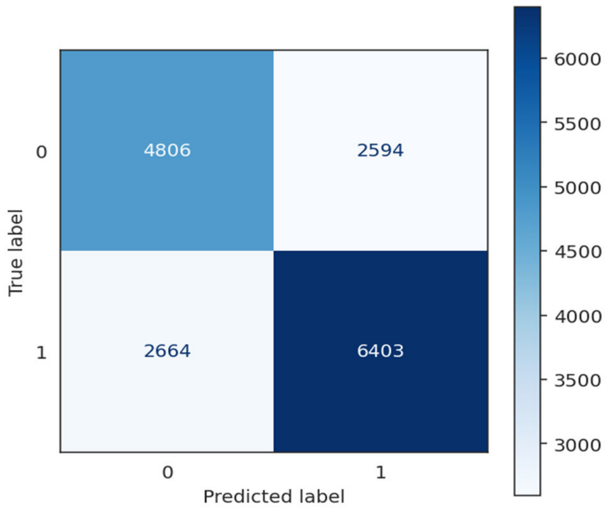
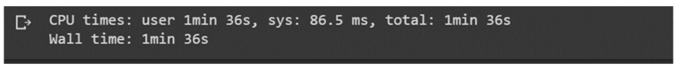


Figure 18: Confusion Matrix for SVM (Kernel - sigmoid, gamma - scale)

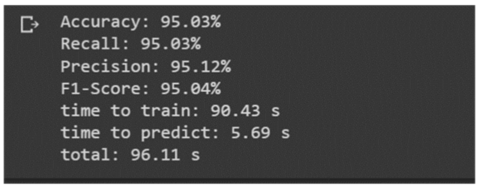
The SVM prediction for (Kernel - poly, gamma - -scale) was presented in Figure 19.



```
CPU times: user 1min 36s, sys: 86.5 ms, total: 1min 36s
Wall time: 1min 36s
```

Figure 19: SVM (Kernel - poly, gamma - scale) Predictions

The SVM method performance for (Kernel - poly, gamma - -scale) was presented in Figure 20.



```
Accuracy: 95.03%
Recall: 95.03%
Precision: 95.12%
F1-Score: 95.04%
time to train: 90.43 s
time to predict: 5.69 s
total: 96.11 s
```

Figure 20: SVM method performance for (Kernel - poly, gamma - scale)

Further, the Confusion Matrix for SVM (Kernel - poly, gamma - -scale) was presented in Figure 21.

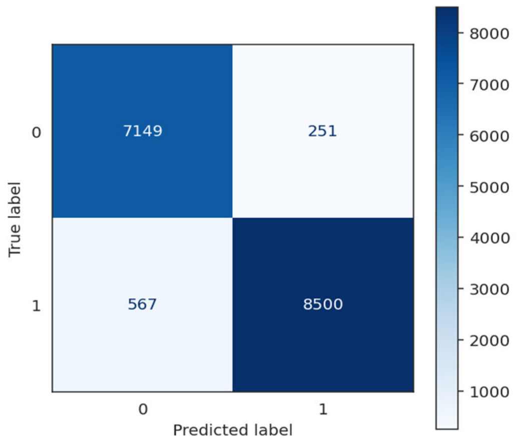
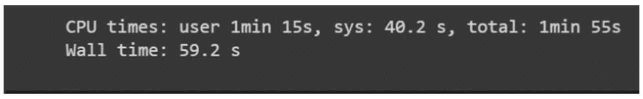


Figure 21: Confusion Matrix for SVM (Kernel - poly, gamma - scale)

ANN - MLP

The Multilayer Perceptron (MLP) is a Feedforward Neural Network (FNN). The MLP is trained using scikit-

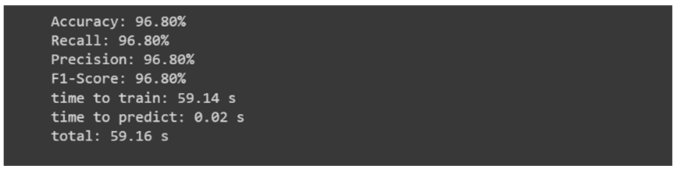
MLPClassifier learn on a dataset ('X train', 'y train') with certain hyperparameters defined, and the learned model is then used to make predictions on another dataset ('X test'). Additionally, it measured how long it takes to train the model and generate predictions using Python's time library. In conclusion, the author tested and trained an MLP classifier. As in Figure 22, the ANN – MLP prediction was captured.



```
CPU times: user 1min 15s, sys: 40.2 s, total: 1min 55s
Wall time: 59.2 s
```

Figure 22: ANN - MLP Predictions

The performance of a trained MLP model was assessed by the following. In this research study, the author used a variety of evaluation metrics, including accuracy, recall, precision, and F1-score. These evaluation metrics were then printed along with the time that it took to train the model, make predictions, and evaluate the performance overall. All evaluation metrics and time were then saved in a dataframe for comparison at a later time. It is a summary of the model's performance. The ANN – MLP method performance is presented in Figure 23.



```
Accuracy: 96.80%
Recall: 96.80%
Precision: 96.80%
F1-Score: 96.80%
time to train: 59.14 s
time to predict: 0.02 s
total: 59.16 s
```

Figure 23: ANN - MLP Method Performance

Next, the author used the scikit-learn library's 'plot confusion matrix' function to create and present a confusion matrix for the trained MLP model on the test dataset. Figure 24 presents the Confusion Matrix for ANN – MLP.

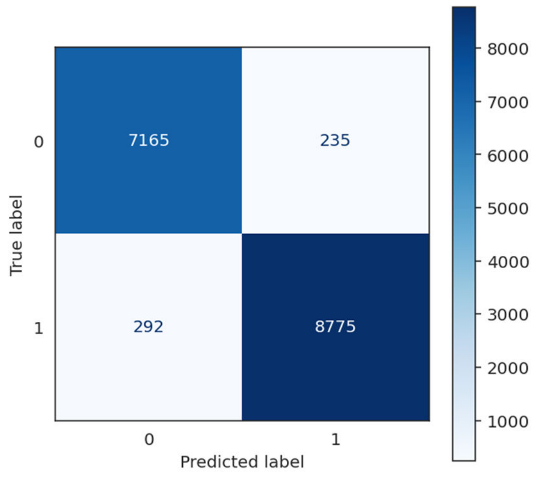


Figure 24: ANN - MLP Confusion Matrix

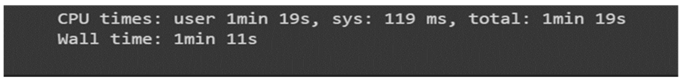
***Although this study used the ANN – MLP model for analyzing purposes. To build the hybrid model, the research team did not use the ANN-MLP model.*

Hybrid Model

The hybrid model was created by combining the Random Forest Model and the SVM model. The Random Forest was used to pre-process the data and to select the most relevant features, followed by the SVM model to classify the data based on the selected features.

- Random Forest Classifier with 150 estimators was used as it yielded the best results in all Random Forest models.
- SVM with poly kernel was used as it yielded the best result among SVM models.

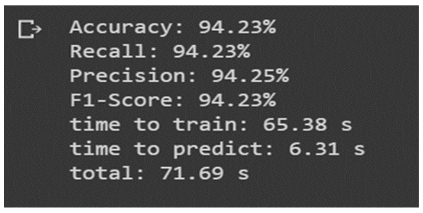
In this research study, a new model that combined the Random Forest Classifier and SVM Classifier was trained. It began by training a Random Forest Classifier with 150 estimators, then used the trained Random Forest model to select the most crucial features from the training data. It set a threshold of "median," which meant that features that were not crucial enough were eliminated from the dataset. The 'X train important' variable was used to keep the training data after it had been modified to include only the most crucial attributes. The test data, which was kept in the 'X test important' variable, went through the same procedure. The important features from the X_train_important data were then used to train an SVM model with a polynomial kernel. Then, using the X_test_important data and the trained SVM model, it made predictions. It also computed the accuracy, recall, precision, and F1-score of the predictions using the y_test data and measured the time required to train the model and make predictions. Figure 25 presents the predictions for the hybrid model.



```
CPU times: user 1min 19s, sys: 119 ms, total: 1min 19s
Wall time: 1min 11s
```

Figure 25: Predictions for Hybrid Model

The author evaluated the performance of a hybrid model that combined the Random Forest model's feature selection method with the Support Vector Machine's classification algorithm (SVM). The most crucial characteristics were chosen from the training set by the Random Forest model, and the SVM was subsequently trained using this smaller feature set. The accuracy, recall, precision, and F1-Score are then used to assess the hybrid model's performance, and the time it took to train and the forecast was also noted. For later comparison with different models, the outcomes were then saved in the "model_performance" dataframe with the label "Hybrid (Estimators - 150, Kernel - poly, gamma - scale)". Figure 26 presents the method performance for the hybrid method.



```
Accuracy: 94.23%
Recall: 94.23%
Precision: 94.25%
F1-Score: 94.23%
time to train: 65.38 s
time to predict: 6.31 s
total: 71.69 s
```

Figure 26: Method Performance - Hybrid Model

Then, using the 'SelectFromModel' feature selection technique, the author generated a confusion matrix for the SVM model that was fitted to the converted training data (X train important) and the transformed test data (X test important). The "Seaborn library's" "plot confusion matrix" method is used to display the confusion matrix as a 5x5-inch figure with a white background and a blue color map to represent it. By comparing the predicted values to the actual values in the test set, this matrix was used to assess the model's performance. Knowing how many false positives, false negatives, true positives, and true negatives the model produced is helpful. Figure 27 presents the Confusion Matrix for the Hybrid Model.

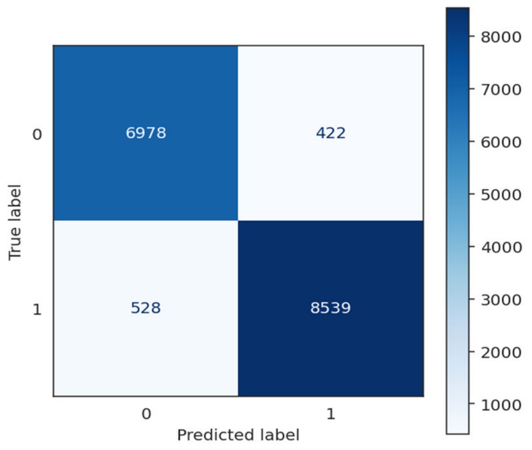


Figure 27: Confusion Matrix for Hybrid Method

Model Comparison

The author can anticipate seeing the performance measures (such as accuracy, recall, precision, and F1-score) of each model as well as their training and prediction timeframes from the model comparison. With the aid of this data, the author can compare the models and choose the one that offers the best overall performance or the best performance/computational efficiency trade-off. The confusion matrix for each model can also be used by the author to gauge how well it predicts the various classes. The overall model comparison is presented in Table 5.

Table 5: Overall model comparison

index	Accuracy	Recall	Precision	F1-Score	train_time	pred_time	total_time
Random Forest (Estimators - 100)	0.97759	0.97759	0.97767	0.97760	7.74432	0.19159	7.9359185
Random Forest (Estimators - 50)	0.97668	0.97668	0.97678	0.97669	4.00490	0.09696	4.1018745
Random Forest (Estimators = 150)	0.97765	0.97765	0.97776	0.97766	11.5436	0.27633	11.819950
SVM (Kernel - rbf, gamma - scale)	0.95020	0.95020	0.95127	0.95029	94.2105	17.3444	111.55504
SVM (Kernel - sigmoid, gamma - scale)	0.68069	0.68069	0.68098	0.68082	357.323	29.7981	387.12117
SVM (Kernel - poly, gamma - scale)	0.95032	0.95032	0.95118	0.95040	101.939	10.0080	111.94741
MLP	0.96799	0.96799	0.96804	0.96800	112.130	0.04786	112.17792
Hybrid (Estimators - 150, Kernel - poly, gamma - scale)	0.942309	0.942309	0.94245	0.94234	100.4496	9.477212	109.9267

Performance Measures:

The hybrid model, which used a combination of 150 estimators and a poly kernel with scale gamma, has an accuracy of 94.2309%. This accuracy is lower than that of the Random Forest algorithm with 100 estimators (97.7592%) and the Random Forest algorithm with 50 and 150 estimators (97.6681% and 97.7652%, respectively), but higher than that of the SVM algorithm with sigmoid kernel and scale gamma (68.0695%).

It could be argued that the specific combination of estimators and kernel used in the hybrid model may not be optimal and that a different combination may yield better performance. Figure 28 presents a comparative bar chart for the Model Performance under the accuracy.

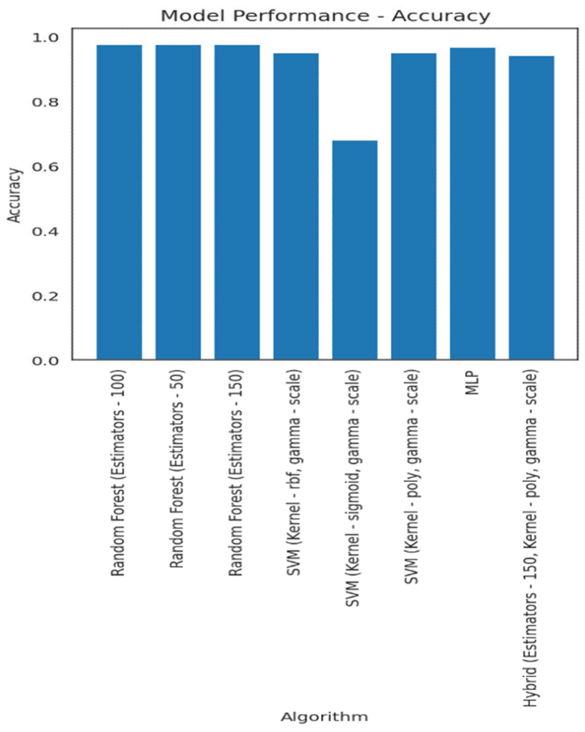


Figure 28: Model Performance - Accuracy

Figure 29 presents the model performance comparison according to the recall.

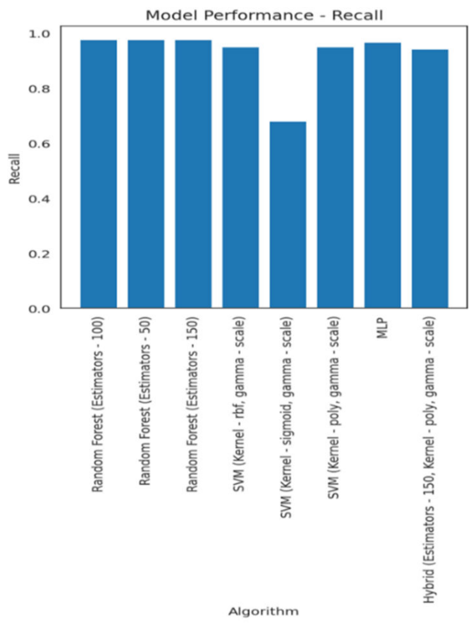


Figure 29: Model Performance - Recall

The hybrid model, which used a combination of 150 estimators and a poly kernel with scale gamma, has a recall value of 94.2309%. This recall is lower than that of the Random Forest algorithm with 100 estimators (97.7592%) and the Random Forest algorithm with 50 and 150 estimators (97.6681% and 97.7652%, respectively). This suggests that the hybrid model is not as good at detecting positive instances (i.e., it has a higher number of false negatives) compared to the Random Forest algorithm with 100 estimators and the Random Forest algorithm with 50 and 150 estimators.

Figure 30 presents the model performance comparison according to the precision.

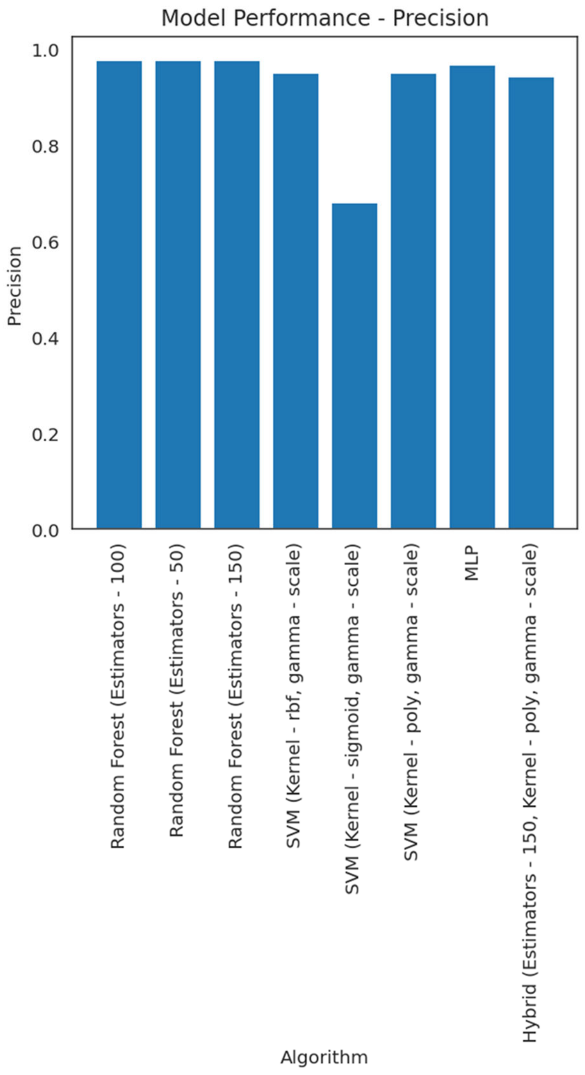


Figure 30: Model Performance - Precision

The hybrid model, which used a combination of 150 estimators and a poly kernel with scale gamma, had a precision value of 94.2459%. This precision was lower than that of the Random Forest algorithm with 100 estimators (97.7678%) and the Random Forest algorithm with 50 and 150 estimators (97.6780% and 97.7765%, respectively). This suggested that the hybrid model was not as good at detecting correct positive instances (i.e., it has a higher number of false positives) compared to the Random Forest algorithm with 100 estimators and the Random Forest algorithm with 50 and 150 estimators.

Figure 31 presents the model performance comparison according to the F1 Score.

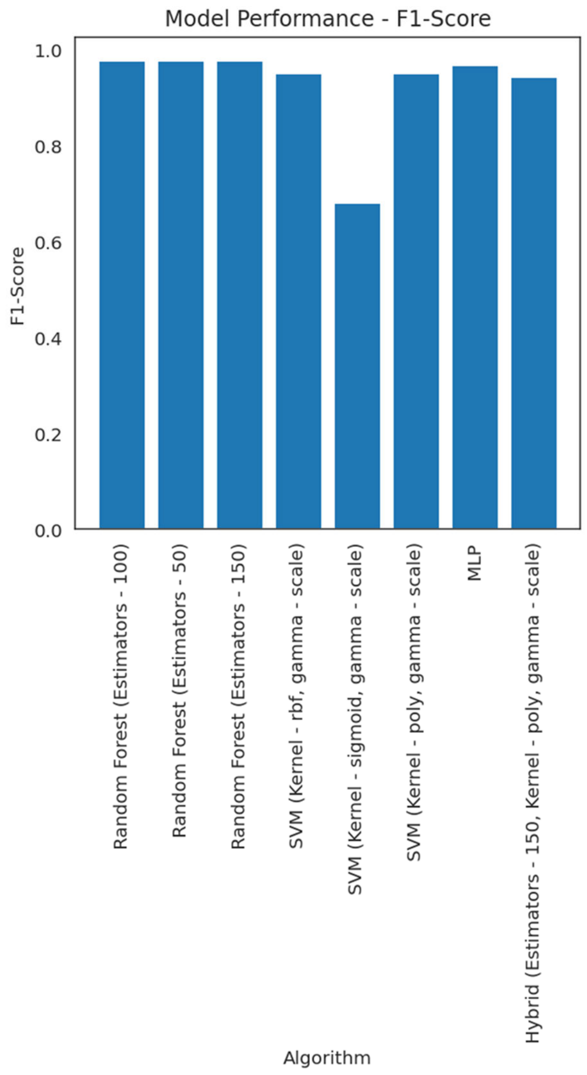


Figure 31: Model Performance - F1 Score

The hybrid model, which used a combination of 150 estimators and a poly kernel with scale gamma, has an F1-score of 94.2344%. This is slightly lower than the Random Forest algorithm with 100 estimators, but higher than the SVM algorithm with sigmoid kernel and scale gamma. This suggested that the hybrid model had a good balance of precision and recall, but not as good as the Random Forest algorithm with 100 estimators. It's also important to note that the F1-score was a measure that seeks a balance between precision and recall, so a higher F1-score means a better balance of precision and recall. In this case, it can be observed that the Hybrid model is not the best in terms of F1-score but it's still quite good.

Time Frames:

The Hybrid model, which used a combination of 150 estimators and a poly kernel with scale gamma, has a train time of 65.38 seconds. This train time was slower than the Random Forest algorithm with 100 estimators, but faster than the SVM algorithm with sigmoid kernel and scale gamma. This suggested that the Hybrid model had a relatively moderate train time compared to other models. However, it's important to consider the trade-off between train time and model performance. As we can see the Hybrid model had a good performance in terms of F1-score, the additional train time may be worth it if the performance gain was deemed significant for the specific application or domain.

Figure 32 presents the model comparison according to the train_time.

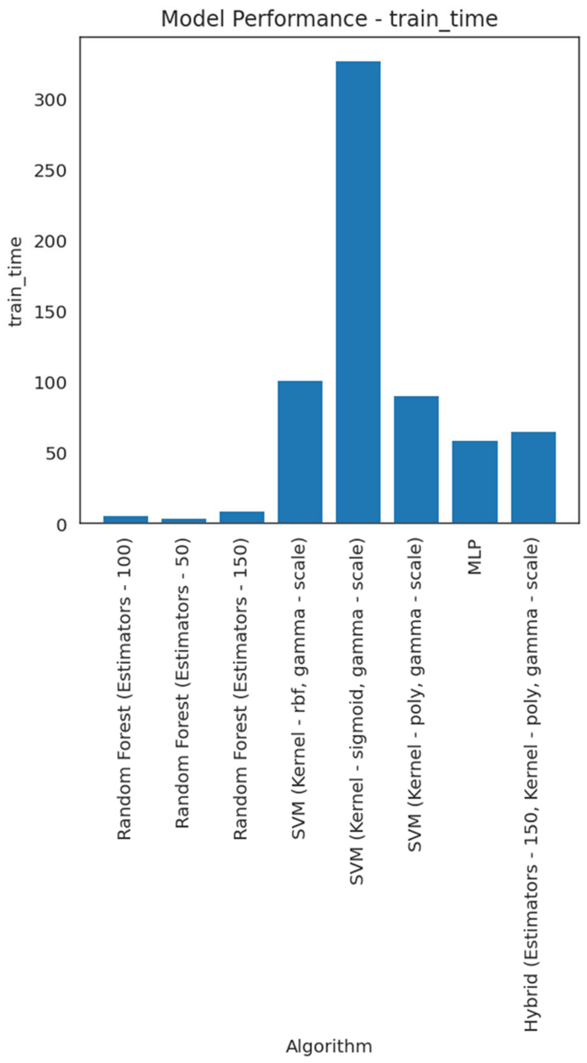


Figure 32: Model Comparison - train_time

Figure 33 presents the model performance comparison according to the pred_time.

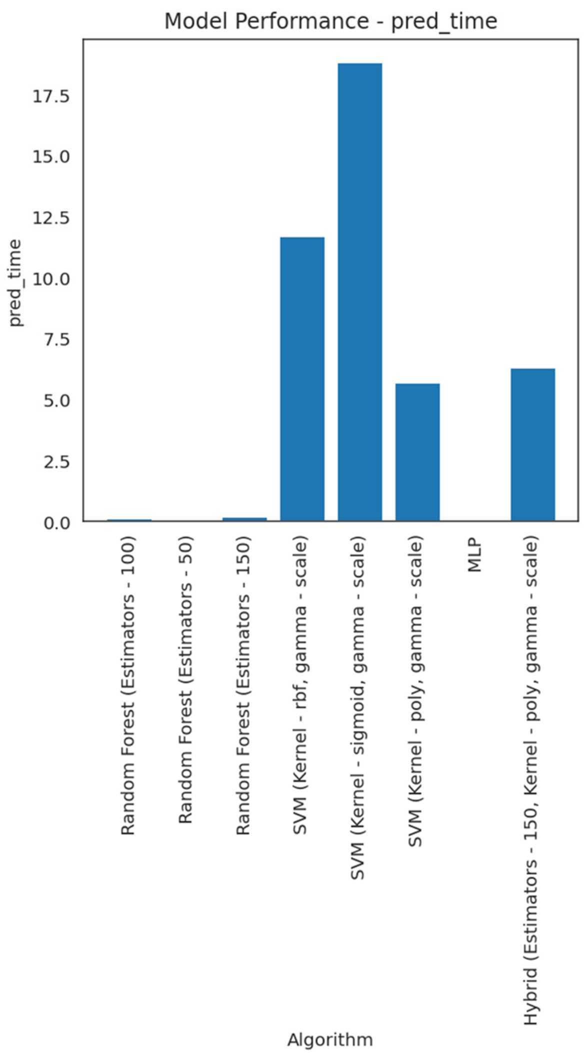


Figure 33: Model Comparison -pred_time

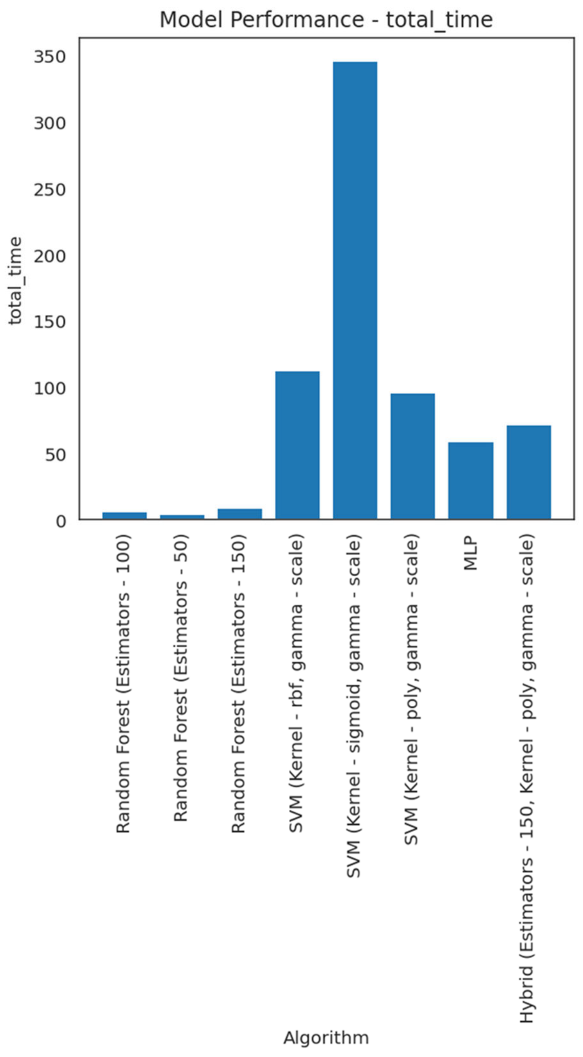


Figure 34: Model Performance - Total Time

The Hybrid model, which used a combination of 150 estimators and a poly kernel with scale gamma, had a prediction time of 6.31 seconds. This prediction time was slower than the Random Forest algorithm with 100 estimators and 50 estimators, but faster than the SVM algorithm with rbf kernel and sigmoid kernel with scale gamma. This suggests that the Hybrid model has a relatively moderate prediction time compared to other models. However, it's important to consider the trade-off between prediction time and model performance. As we can see the Hybrid model had a good performance in terms of F1-score, the additional prediction time may be worth it if the performance gain was deemed significant for the specific application or domain.

Figure 34 presents the model comparison according to the total time.

From the given output, it can be seen that the 'Hybrid (Estimators - 150, Kernel - poly, gamma - scale)' model has a total time of 71.691509 seconds, which was slower than most of the other models, particularly when compared to the Random Forest models and the MLP model. This suggested that the hybrid model may not be as computationally efficient as some of the other models in terms of the total time taken.

Class imbalance problem

The dataset's class imbalance problem poses a serious problem that affects the accuracy of the findings when it comes to anomaly detection in cloud network data. When the proportion of normal behavior to anomalous behavior is noticeably greater, an imbalance arises. This imbalance might, in practice, result in a model that performs well in predicting instances of the majority class (normal examples) but poorly in detecting cases of the minority class (anomalies). A model with great precision but low recall is one of the possible outcomes, which could lead to a system that fails to recognize real security threats and ignores vulnerabilities.

Looking ahead, resolving the issue of class imbalance becomes essential for subsequent studies. Investigating other balancing strategies, including oversampling, undersampling, or sophisticated approaches like the Synthetic Minority Over-sampling Technique (SMOTE), is one possible direction. The purpose of these methods is to lessen the effect of class imbalance on model performance. A more resilient anomaly detection system may also benefit from the application of ensemble methods and the creation of more sophisticated hybrid models, particularly when those models are specifically made to manage unbalanced datasets. To gain a deeper comprehension of hybrid model performance in real-world scenarios, future research should expand evaluations to a wider range of real-world datasets.

V. CONCLUSION & RECOMMENDATIONS

A. *Discussion*

The main objective of this research was to introduce a novel hybrid model for detecting anomalies in cloud network data and to compare its performance to other machine learning models. The study used the UNSW-NB15 anomaly dataset for the experiments and preprocessed and selected features for the training and testing sets. The model training was done using Random Forest and SVM algorithms, and a novel hybrid model was built with Hybrid RF(Estimators - 150) and SVM(Kernel - poly, gamma - scale) due to their higher accuracy and other aspects.

The results showed that the novel hybrid model performed somewhat poorly compared to the Random Forest models that were used alone, but the total time for the hybrid model was deemed acceptable. This was the first time that a hybrid model was used for the UNSW_NB15 dataset. The limitation of the study was the class imbalance problem in the dataset.

The results of this study contributed to the understanding of how different algorithmic combinations affect the performance of a hybrid model in detecting anomalies in cloud network data. The study also highlights the importance of feature selection and pre-processing techniques in improving the performance of a model. However, the study also highlighted the need for further research to address the class imbalance problem in the dataset.

One possible explanation for the poor performance of the hybrid model could be the combination of the two models. SVM and Random Forest used different approaches to solve classification problems, and combining them may not have resulted in an optimal solution. Another possible explanation could be the choice of parameters for the SVM, such as the kernel and gamma, which may not have been the best suited for the specific dataset used in this research.

Based on the information provided, the contribution of the study can be summarized as follows:

- Novel Hybrid Model: The study proposed a new hybrid model to detect anomalies in cloud network data. The model was built using two selected algorithms, SVM and Random Forest, and is compared to single-algorithm models to evaluate its performance.

- Algorithmic Combinations: The study investigated the impact of different algorithmic combinations on the performance of the hybrid model. This analysis provides insights into the effectiveness of various machine learning algorithms in detecting anomalies in cloud network data.

- Data Handling: The study also explored how well the hybrid model handles various types of data and how various feature selection and pre-processing techniques can affect its performance.

- Research Pathways: The study discusses potential future research pathways for the application of hybrid models in anomaly detection and cloud network security. It also highlights the importance of understanding the hybrid model and its security implications.

- Performance Evaluation: The study evaluates the hybrid model in terms of its computational resources, false positives, and false negatives, which can provide practical insights into its usefulness in real-world applications.

Overall, the study contributed to the field of anomaly detection and cloud network security by proposing a new hybrid model and evaluating its performance against other machine learning algorithms. It also provided insights into the impact of different algorithmic combinations, data handling techniques, and potential research pathways.

B. *Practical implication of the hybrid model*

Particularly in the area of anomaly detection in cloud network data, the hybrid model in this study has important real-world applications. The model provides a sophisticated approach to addressing the intricacies and nuances inherent in cloud network security by combining the benefits of Random Forest (RF) and Support Vector Machine (SVM). The robustness of the system is improved when managing cloud network anomalies because of its capacity to create precise decision limits with the help of SVM and to capture complex relationships within data, which is made possible by RF's ensemble learning. Put practically, this means that cloud settings will be able to recognize odd patterns or possible security concerns with greater precision.

C. *Conclusion and Recommendations*

This research aimed to introduce a novel hybrid model for detecting anomalies in cloud network data and to compare its performance to other machine learning models. The study used the UNSW-NB15 anomaly dataset and preprocessed and selected features for the training and testing sets. The results showed that the novel hybrid model performed somewhat poorly compared to the Random Forest models that were used alone, but the total time for the hybrid model was deemed acceptable. The study also highlighted the need for further research to address the class imbalance problem in the dataset. Overall, the study contributed to the understanding of how different algorithmic combinations affect the performance of a hybrid model in detecting anomalies in cloud network data and the importance of feature selection and pre-processing techniques in improving the performance of a model.

The practical implications of the findings suggest that hybrid models can be used for anomaly detection in cloud network data, but the performance may be impacted by the selection of algorithms and the dataset used. The study also recommends future research to address the class imbalance problem in the dataset and to further investigate the potential of hybrid models in anomaly detection and cloud network security. Additionally, the study recommends future research to investigate the rate of false positives and false negatives, computational resources, and the ease of understanding of the hybrid model.

In conclusion, this research has shown that a hybrid model of SVM and Random Forest can be used for anomaly identification in cloud network data using the UNSW-NB15 dataset. However, the results suggested that the performance of the hybrid model was not as good as the Random Forest models alone. Further research is needed to optimize the parameters of the SVM and Random Forest models to improve the performance of the hybrid model. Despite the limitations, this research provides valuable insights for future research in this area.

ACKNOWLEDGMENT

The first author is thanking the supervisors of this research study for their support.

FUNDING

Not applicable.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

REFERENCES

[1] A. Vervaet, "MONILOG: An Automated Log-based ANOMALY DETECTION SYSTEM FOR CLOUD computing infrastructures," 2021 IEEE 37th International Conference on Data Engineering (ICDE), 2021.

[2] Dingde Jiang, Yang Han, Xingwei Wang, Zhengzheng Xu, Hongwei Xu, and Zhenhua Chen, "A time-frequency detecting method for network traffic anomalies," International Conference on Computational Problem-Solving, Li Jiang, China, 2010, pp. 94-97.

[3] B. Wang, Q. Hua, H. Zhang, X. Tan, Y. Nan, R. Chen, and X. Shu, "Research on ANOMALY DETECTION and real-time reliability evaluation with the log of cloud platform," Alexandria Engineering Journal, vol. 61, no. 9, pp. 7183–7193, 2022.

[4] S. H. Haji and S. Y. Ameen, "Attack and anomaly detection in IOT networks using Machine Learning Techniques: A Review," Asian Journal of Research in Computer Science, pp. 30–46, 2021.

[5] A. B. Nassif, M. A. Talib, Q. Nasir, and F. M. Dakalbab, "Machine Learning for Anomaly Detection: A Systematic Review," in IEEE Access, vol. 9, pp. 78658-78700, 2021, doi: 10.1109/ACCESS.2021.3083060.

[6] T. Sureda Riera, J.-R. Bermejo Higuera, J. Bermejo Higuera, J.-J. Martínez Herraiz, and J.-A. Sicilia Montalvo, "Prevention and fighting against web attacks through anomaly detection technology. A systematic review," Sustainability, vol. 12, no. 12, p. 4945, 2020.

[7] M. Ozkan-Okay, R. Samet, Ö. Aslan and D. Gupta, "A Comprehensive Systematic Literature Review on Intrusion Detection Systems," in IEEE Access, vol. 9, pp. 157727-157760, 2021, doi: 10.1109/ACCESS.2021.3129336.

[8] J. Svacina, J. Raffety, C. Woodahl, B. Stone, T. Cerny, M. Bures, D. Shin, K. Frajtak, and P. Tisnovsky, "On vulnerability and Security Log Analysis," Proceedings of the International Conference on Research in Adaptive and Convergent Systems, 2020.

[9] T. L. Yasarathna and L. Munasinghe, "Anomaly detection in cloud network data," 2020 International Research Conference on Smart Computing and Systems Engineering (SCSE), Colombo, Sri Lanka, 2020, pp. 62-67, doi: 10.1109/SCSE49731.2020.9313014.

[10] T. Hagemann and K. Katsarou, "A systematic review on anomaly detection for cloud computing environments," 2020 3rd Artificial Intelligence and Cloud Computing Conference, 2020.

[11] A. Alshammari and A. Aldribi, "Apply machine learning techniques to detect malicious network traffic in cloud computing," Journal of Big Data, vol. 8, no. 1, 2021.

[12] S. Nedelkoski, J. Cardoso and O. Kao, "Anomaly Detection from System Tracing Data Using Multimodal Deep Learning," 2019 IEEE 12th International Conference on Cloud Computing (CLOUD), Milan, Italy, 2019, pp. 179-186, doi: 10.1109/CLOUD.2019.00038.

[13] M. S. Islam, W. Pourmajidi, L. Zhang, J. Steinbacher, T. Erwin and A. Miranskyy, "Anomaly Detection in a Large-Scale Cloud Platform," 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), Madrid, ES, 2021, pp. 150-159, doi: 10.1109/ICSE-SEIP52600.2021.00024.

[14] F. J. Schmidt, "Anomaly detection in cloud computing environments," thesis.

[15] T. Salman, D. Bhamare, A. Erbad, R. Jain and M. Samaka, "Machine Learning for Anomaly Detection and Categorization in

Multi-Cloud Environments," 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud), New York, NY, USA, 2017, pp. 97-103, doi: 10.1109/CSCloud.2017.15.

[16] S. E. Hajjami, J. Malki, M. Berrada and B. Fourka, "Machine Learning for anomaly detection. Performance study considering anomaly distribution in an imbalanced dataset," 2020 5th International Conference on Cloud Computing and Artificial Intelligence: Technologies and Applications (CloudTech), Marrakesh, Morocco, 2020, pp. 1-8, doi: 10.1109/CloudTech49835.2020.9365887.

[17] X. Qiu, Y. Dai, P. Sun and X. Jin, "PHM Technology for Memory Anomalies in Cloud Computing for IaaS," 2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS), Macau, China, 2020, pp. 41-51, doi: 10.1109/QRS51102.2020.00018.

[18] A. Gerard, R. Latif, S. Latif, W. Iqbal, T. Saba and N. Gerard, "MAD-Malicious Activity Detection Framework in Federated Cloud Computing," 2020 13th International Conference on Developments in eSystems Engineering (DeSE), Liverpool, United Kingdom, 2020, pp. 273-278, doi: 10.1109/DeSE51703.2020.9450728.

[19] J. Bogatinovski, S. Nedelkoski, J. Cardoso and O. Kao, "Self-Supervised Anomaly Detection from Distributed Traces," 2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC), Leicester, UK, 2020, pp. 342-347, doi: 10.1109/UCC48980.2020.00054.

[20] W. Wang, X. Du, D. Shan, R. Qin and N. Wang, "Cloud Intrusion Detection Method Based on Stacked Contractive Auto-Encoder and Support Vector Machine," in IEEE Transactions on Cloud Computing, vol. 10, no. 3, pp. 1634-1646, 1 July-Sept. 2022, doi: 10.1109/TCC.2020.3001017.

[21] C. Raj, L. Khular and G. Raj, "Clustering Based Incident Handling For Anomaly Detection in Cloud Infrastructures," 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2020, pp. 611-616, doi: 10.1109/Confluence47617.2020.9058314.

[22] Y. Yuan, H. Anu, W. Shi, B. Liang and B. Qin, "Learning-Based Anomaly Cause Tracing with Synthetic Analysis of Logs from Multiple Cloud Service Components," 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), Milwaukee, WI, USA, 2019, pp. 66-71, doi: 10.1109/COMPSAC.2019.00019.

[23] M. Thill, W. Konen and T. Bäck, "Online anomaly detection on the webscope S5 dataset: A comparative study," 2017 Evolving and Adaptive Intelligent Systems (EAIS), Ljubljana, Slovenia, 2017, pp. 1-8, doi: 10.1109/EAIS.2017.7954844.

[24] M. S. Islam and A. Miranskyy, "Anomaly Detection in Cloud Components," 2020 IEEE 13th International Conference on Cloud Computing (CLOUD), Beijing, China, 2020, pp. 1-3, doi: 10.1109/CLOUD49709.2020.00008.

[25] S. Eltanbouly, M. Bashendy, N. AlNaimi, Z. Chkirbene and A. Erbad, "Machine Learning Techniques for Network Anomaly Detection: A Survey," 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT), Doha, Qatar, 2020, pp. 156-162, doi: 10.1109/ICIoT48696.2020.9089465.

[26] I. Aljamal, A. Tekeoğlu, K. Bekiroglu and S. Sengupta, "Hybrid Intrusion Detection System Using Machine Learning Techniques in Cloud Computing Environments," 2019 IEEE 17th International Conference on Software Engineering Research, Management and Applications (SERA), Honolulu, HI, USA, 2019, pp. 84-89, doi: 10.1109/SERA.2019.8886794.

[27] Kithulwatta, W.M.C.J.T., Wickramaarachchi, W.U., Jayasena, K.P.N., Kumara, B.T.G.S., Rathnayaka, R.M.K.T. (2022). Adoption of Docker Containers as an Infrastructure for Deploying Software Applications: A Review. In: Saeed, F., Al-Hadhrami, T., Mohammed, E., Al-Sarem, M. (eds) Advances on Smart and Soft Computing. Advances in Intelligent Systems and Computing, vol 1399. Springer, Singapore. https://doi.org/10.1007/978-981-16-5559-3_21

[28] W. M. C. J. T. Kithulwatta, K. P. N. Jayasena, B. T. G. S. Kumara and R. M. K. T. Rathnayaka, "Docker incorporation is different from other computer system infrastructures: A review," 2021 International Research Conference on Smart Computing and Systems Engineering (SCSE), Colombo, Sri Lanka, 2021, pp. 230-236, doi: 10.1109/SCSE53661.2021.9568323.

[29] W. M. C. J. T. Kithulwatta, K. P. N. Jayasena, B. T. G. S. Kumara and R. M. K. T. Rathnayaka, "Docker Containerized Infrastructure Orchestration with Portainer Container-native Approach," 2022 3rd International Conference for Emerging Technology (INCET), Belgaum, India, 2022, pp. 1-6, doi: 10.1109/INCET54531.2022.9825257.

[30] W. M. C. J. T. Kithulwatta, K. P. N. Jayasena, B. T. G. S. Kumara and R. M. K. T. Rathnayaka, "Performance Evaluation of Docker-based Apache and Nginx Web Server," 2022 3rd International Conference for Emerging Technology (INCET), Belgaum, India, 2022, pp. 1-6, doi: 10.1109/INCET54531.2022.9824303.

[31] Kithulwatta, W.M.C.J.T., Jayasena, K.P.N., Kumara, B.T. and Rathnayaka, R.M.K.T., 2022. Integration With Docker Container Technologies for Distributed and Microservices Applications: A State-of-the-Art Review. International Journal of Systems and ServiceOriented Engineering (IJSSOE), 12(1), pp.1-22.

[32] Jayaweera, M.P.G.K., Kithulwatta, W.M.C.J.T. & Rathnayaka, R.M.K.T. Detect anomalies in cloud platforms by using network data: a review. Cluster Comput 26, 3279–3289 (2023). https://doi.org/10.1007/s10586-023-04055-1

[33] Gayantha, M. H., Kithulwatta, W. M. C. J. T., & Rathnayaka, R. M. K. T. (2022). The Interconnection of Internet of Things and Artificial Intelligence: A Review. In Sri Lankan Journal of Applied Sciences (Vol. 1, Issue 1). https://sljoas.uwu.ac.lk/index.php/sljoas/article/view/45/12

[34] M.H. Gayantha, W.M.C.J.T. Kithulwatta, R.M.K.T. Rathnayaka. Identification of a Machine Learning Architecture for Potato DiseaseClassification Using Leaf Images. Applied Sciences Undergraduate Research Symposium 2022 At: Sabaragamuwa University of Sri Lanka. p. 15.