Multi-class Classification of Internet Firewall Data: A Comparative Study

Kay Royo, Dhanusha Pathakota, Rishika Garg
University of California, Davis
STA 221

Abstract

The analysis of firewall logs plays an important role in monitoring network traffic and making informed decisions about allowing or blocking specific traffic. It plays a significant role in evaluating the impact of network activity. Our study focuses on analyzing firewall logs by building multi-class machine learning (ML) and deep learning (DL) models that are capable of classifying the incoming traffic into four distinct classes- "Allow, " "Drop, " "Deny, " or "Reset-both". Several ML and DL algorithms were used for evaluation , including, Random Forest (RF), Support Vector Machine (SVM), Deep Neural Network (DNN), Logistic Regression (LR) and k-Means Clustering. After our analysis, Random Forest was found to be the most effective algorithm for classifying the internet firewall data.

1 Introduction

Internet firewalls are an integral part of our daily lives, protecting our digital devices and networks from unauthorized access, cyber threats, and ensuring a safer online experience. The analysis and classification of firewall log data is crucial for several key reasons. Firstly, it enables the identification and categorization of various network traffic and activities, providing insights into firewall actions and the effectiveness of security measures. Secondly, classification facilitates the detection and mitigation of security threats by training models to identify known attack patterns and classify real-time attacks. Additionally, the classification of firewall log data supports network monitoring and optimization, enabling the analysis of traffic patterns, identification of bottlenecks, and efficient resource allocation. Overall, the classification of firewall logs enhances network security, aids in threat detection, optimizes network performance, and ensures the protection of critical network infrastructure. [13]

2 Problem Definition

Finding the most effective classification method for internet firewall log datasets can be challenging due to several factors. The data itself may possess complexity and high-dimensionality, making it difficult to identify the optimal approach for accurate classification. Additionally, the presence of various classes and imbalances in class distribution can pose challenges in training a robust classifier. Moreover, the dynamic nature of network activity and the emergence of new types of threats necessitate constant adaptation and evaluation of classification methods. Lastly, the effectiveness of a classification method may depend on the specific characteristics of the firewall log data and the desired objectives, highlighting the importance of exploring and comparing multiple techniques to determine the most suitable approach.

2.1 Goal

This study aims to identify the most effective machine learning method to employ for the classifying the internet firewall data provided by UCI Machine Learning Repository [1]. By determining the best model, we aim to select the most appropriate techniques based on the specific characteristics and requirements of the data. Through evaluation and comparison of different methods, we can identify the ones that yield the highest accuracy, performance, and interpretability for the given dataset. This ensures that the classification models are optimized for accurately identifying and categorizing network traffic, detecting security threats, and making informed decisions. Additionally, understanding the strengths and limitations of different methods helps in tailoring the classification approach to the unique challenges and complexities of internet firewall data, ultimately enhancing network security and operational efficiency.

In this study, we focus on examining the performance of the chosen algorithms, namely Random Forest (RF), Support Vector Machine (SVM), Deep Neural Network (DNN), Logistic regression (LR) and k-Means Clustering on the given dataset. Following the classification, the study evaluates the performance of different classifiers on the data by analyzing metrics such as accuracy, F1 score, precision, and recall, with the F1 score being a key metric for evaluating classifier performance.

2.2 Data Description

The internet firewall data used in this study is available in the UCI Machine Learning Repository, which was provided by Fatih Ertam from Firat University in Turkey [1]. This data set was collected from the internet traffic records on a university's firewall and it includes 12 distinct features, with the "Action" attribute serving as the class identifier (the other 11 being input features).These features encompass a wide range of information that are defined in Table 1, including the source port, destination port, NAT source port, NAT destination port, action, bytes, bytes sent, bytes received, packets, elapsed time (in seconds), packets sent, and packets received. With its diverse set of attributes, this comprehensive dataset is useful for delving into various aspects of network activity. Its application holds significant potential for the development of analytical models and algorithms aimed at enhancing our knowledge of network dynamics.

Table 1: Feature Description

Variable	Description
Source Port	Client source port
Destination Port	Client destination port
NAT Source Port	Modified source port number assigned during Network Address Translation (NAT) processes
NAT Destination Port	Modified destination port number assigned during Network Address Translation (NAT) processes
Action	Action taken by the firewall when processing a specific network connection or packet
Bytes	Size of data transmitted over a network
Bytes Sent	Total size of data that has been transferred from the sender to the recipient
Bytes Received	Total size of data that has been recieved by the recipient
Packets	Discrete units of data that are transmitted over a network
Elapsed Time (sec)	Time elapsed for a specific network connection
pkts_sent	Number of packets that have been transmitted from a source device in a network connection or session
pkts_received	Number of packets that have been received by a destination device in a network connection

2.3 Literature Review

The analysis and classification of internet firewall data and the examination of firewall logs have been subjects of extensive research in the field of network security and threat detection. Many studies have focused on developing effective models and methods to extract valuable insights from firewall logs

and enhance network management and monitoring. This literature review discusses relevant works related to feature selection and model choices in the classification of internet firewall data.

Feature selection is a critical factor in improving the efficiency and accuracy of classification models. Several papers, such as "High dimensional data classification and feature selection using support vector machines" [11] and "Selecting critical features for data classification based on machine learning methods, " [5] have emphasized the importance of feature selection. The former focuses on the classification of high-dimensional data using Support Vector Machines (SVMs) and the selection of critical features for improving the classification performance. And, the latter deals with identifying the most critical features for several machine learning techniques, including Decision Tree, Random Forest, and Support Vector Machine.

The paper [2] compares the performance of three machine learning techniques, namely Support Vector Machine (SVM), Random Forest (RF), and Extreme Learning Machine (ELM), for intrusion detection. The study used the NSL-KDD dataset and found that ELM outperformed the other approaches in terms of accuracy, precision, and recall. Additionally, As-Suhbani et al. [4] proposed a meta classifier model employing four binary classifiers that analyzed a network log dataset. They utilized Six features obtained after feature selection to train ML classifiers, including kNN, NB, J48, RF. The data was classified into "Allow" or "Drop" classes. The performance of the four algorithms was compared based on accuracy, F-measure, and Receiver Operating Characteristics (ROC) values. The kNN classifier achieved the highest accuracy of 99.87%.

Ertam et al. [9] conducted a study classifying the network log using the SVM algorithm. They evaluated the model's performance by using different SVM activation functions and performing multi-class classification of the action attribute, which could be "allow, " "drop, " "deny, " or "reset-both. " Radial Basis Function (RBF) activation function obtained the highest F-measure of 76.4%. Similarly, AL-Behadili [3] developed a Decision Tree (DT) classification algorithm for network log analysis to predict the action. They used the same 11 features and six benchmark classification algorithms, with the DT model achieving the best accuracy of 99.839%. Sharma et al. [16] employed the same dataset and set of 11 features to train their models. The DT classifier achieved the highest performance with a precision of 87%. Additionally, when they used a stacking ensemble with Random Forest (RF) as its meta, they achieved a precision of 91% and an accuracy of 99.8%. In our study, we used common algorithms that were applied in similar papers.

3 Methodology

3.1 Intuition

While our metrics may not outperform the existing benchmarks , our focus on identifying the most suitable techniques for extracting insights from firewall logs makes our approach valuable for network security and threat detection. By guiding the selection of features and suitable models, our findings contribute to enhancing the overall understanding and management of network traffic.
This project provides a significant advancement in the domain of internet firewall data classification when compared to state-of-the-art systems. Here are the key reasons why it is considered superior:

1. Evaluation of Feature Selection Techniques: The project extensively evaluates various feature selection methods specific to internet firewall data. By analyzing and comparing their results, it identifies the dominant features that can improve the performance of classification models. This ensures that the relevant features are selected, leading to enhanced classification accuracy and robustness.

2. Comparison of ML and DL Models: The project goes beyond traditional machine learning (ML) models and explores the effectiveness of deep learning (DL) models for internet firewall data classification. DL models, with their ability to automatically learn intricate patterns, offer potential advantages over traditional ML models. By thoroughly comparing and benchmarking the performance of both approaches, the project helps in determining the most suitable model for

accurate classification of firewall data.

3. Assessment of Supervised and Unsupervised Methods: The project investigates both supervised and unsupervised methods for firewall data classification. It examines how well these methods handle the unique challenges and characteristics of firewall logs. By evaluating their strengths and limitations, the project enables a comprehensive understanding of the trade-offs involved in using either approach. This knowledge assists in selecting the appropriate method based on the availability of labeled data and the desired level of interpretability.

By addressing these crucial aspects, this project stands out by providing insights into the effectiveness of various preprocessing techniques, the comparison between ML and DL models, and the evaluation of supervised and unsupervised methods. Through its comprehensive analysis, it offers valuable guidance to network security professionals, enabling them to make informed decisions and improve network monitoring and management for enhanced security and threat detection.

Figure 7(in the appendix section) describes the proposed methodology in an illustrative diagram.

3.2 Data Preprocessing

Preprocessing allows the raw data extracted from the dataset to be transformed through a series of steps such as deleting redundant records and normalising data, to get the data into the form required for learning. Datasets occasionally contain missing values; this can occur from errors in recording or extracting the features. To deal with missing values in the dataset, we decided to drop any rows that contained NaN, Null or Inf values, as the dataset is large enough this has almost no effect on the results. The table below shows the distribution of data obtained after data cleaning:

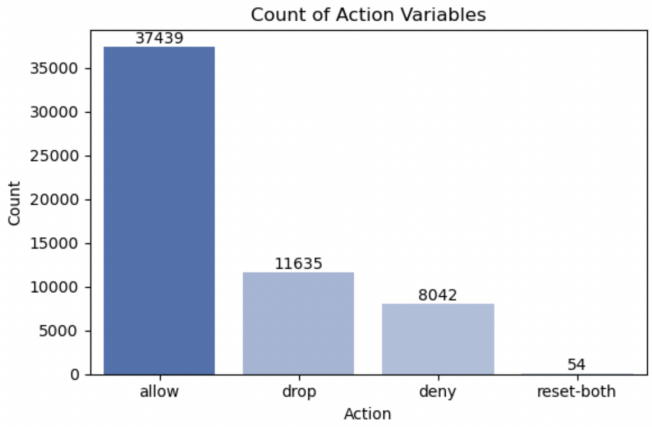


Figure 1: Target Distribution

The next step in preprocessing is normalisation. Normalisation is essential since the scale of the feature values differs. So, by bringing all the features within the same range, we ensure that they contribute an equal amount towards the classification. [7] We performed min-max normalisation using MinMaxScaler() from scikit-learn library. MinMaxScaler() re-scales all of the features into the [0, 1] range, using the following formula:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

where x is the original value.

We also carried out one-hot encoding. One-hot encoding is necessary when dealing with categorical variables or features with discrete values. It converts each categorical value into a binary vector

representation, allowing machine learning algorithms to process them effectively. We performed one-hot encoding using the get_dummies() function from the pandas library. This function creates binary columns for each unique value in the categorical feature. Each column represents a category, and the value is either 0 or 1, indicating whether the original value matches that category. By using one-hot encoding, we ensure that the categorical features are appropriately represented in a format that can be understood by machine learning algorithms, improving the performance and accuracy of our models.

3.3 Feature Selection

To improve the performance and efficiency of machine learning models used in this study, we focus on using the most informative features identified using feature selection. Feature selection is valuable for classification models as it improves performance by focusing on the most relevant features, reduces over-fitting by removing irrelevant features, and enhances interpretability by providing a clearer understanding of the relationship between features and class labels. It also contributes to computational efficiency by reducing dimensionality and improves noise tolerance by eliminating irrelevant information. [6]

Various feature selection methods are utilized including Chi-square test, forward selection, backward elimination, and recursive feature elimination using Logistic Regression.

- **Chi-square Test:** The Chi-square test is a statistical test used to determine whether or not two variables are related. In this case, we're utilizing it to determine whether there is a relationship between each feature and the target variable ('Action'). We can examine the features that have a strong correlation with the target variable by computing the chi-square statistic and p-values for each feature. The greater the chi-square value, the more important the feature is in determining the target variable. The formula for the chi-square test statistic is $\chi^2 = \frac{\sum(O-E)^2}{E}$ where O = Observed frequency and E = Expected frequency.
- **Recursive Feature Elimination (RFE) with Logistic Regression (LR):** RFE is a technique that assists us in selecting the most important features by recursively eliminating the less important ones. Here, we're using Logistic Regression (LR) to determine the importance of features. The algorithm starts with all the features and then removes the least important one in each iteration, until we reach the desired number of features. The importance of features is determined by the coefficients assigned by the Logistic Regression model.
- **Forward Feature Selection (FFS) with Logistic Regression (LR):** Forward feature selection begins with an empty set of features and adds one feature at a time based on importance. The algorithm selects features based on the highest score achieved, which is determined by the performance of the Logistic Regression model. By adding features one by one, FFS identifies the most important features that contribute to accurate classification.
- **Backward Elimination (BE) with Logistic Regression (LR):** Backward elimination is similar to forward feature selection but starts with all the features and removes the least important one in each iteration. Again, we're using Logistic Regression (LR) to estimate the importance of features. The algorithm removes features based on their importance, which is determined by the coefficients assigned by the Logistic Regression model. By iteratively eliminating features, BE identifies a subset of features that are most important for the classification.

Based on the above results, we have made the decision to select the top four features for classification, namely **Destination Port, NAT Source Port, Elapsed Time (sec), and Packets Received**. These features demonstrate a cumulative importance of nearly 90%, indicating their significance in the classification task.

Table 2: Feature Selection Result (Scaled Features)

Feature	Chi-square	RFE (LR)	FFS (LR)	BE (LR)	Total
Destination Port	True	True	True	True	4
NAT Source Port	True	True	True	True	4
Elapsed Time (sec)	True	True	True	True	4
pkts_received	True	True	False	True	3
Source Port	True	True	False	False	2
Bytes Received	True	True	False	False	2
Bytes	False	False	True	False	1
Bytes Sent	False	False	True	False	1
pkts_sent	False	False	False	True	1
Packets	False	False	False	False	0

True = Selected, RFE = Recursive feature elimination, FFS = Forward feature selection, BE = Backward elimination, LR = Logistic regression

3.4 Implemented models

The following supervised and unsupervised models are utilized as part of our implementation.The data is split it into training and testing sets in the ratio 80:20. [12]

3.4.1 Supervised Methods

The four supervised models that we utilized for the classification of Internet firewall data are Random Forest, Deep Neural Network, Support Vector Machines, and Logistic Regression with PCA.

Random Forest

Random Forest is a popular machine learning algorithm known for its effectiveness in both classification and regression tasks. The random forest is a collection of decision trees, building each tree by randomly sampling the features and the training data, and using majority voting amongst the trees to assign the class label. As a result, they tend to outperform decision trees since they overcome the over-fitting problem that decision trees may suffer from and are less sensitive to outlier data. [14]

Algorithm 1 : Random Forest

Precondition: A training set S := (x1, y1), ... , (xn, yn), features F, and number of trees in forest B.
1. function RandomForest(S, F)
2. H $\leftarrow \phi$
3. for i $\in$ 1, ..., B do:
4. S(i) $\leftarrow$ A bootstrap sample from S
5. hi $\leftarrow$ RandomizedTreeLearn(S(i), F)
6. H $\leftarrow$ H $\cup$ hi
7. end for
8. return H
9. end function
10. function RandomizedTreeLearn(S , F)
11. At each node:
12. f $\leftarrow$ very small subset of F
13. Split on best feature in f
14. return The learned tree
15. end function

Support Vector Machines

A support vector machine (SVM) is a supervised, linear classifier. It creates a hyperplane, and the goal is to find the maximum-margin hyperplane that separates the feature vectors of one class from those of another class. This separation forms a decision boundary for classifying the data points. [15]

There are different kernel functions available for SVM, each with its own characteristics and suitability for different scenarios. Some commonly used kernels include the linear kernel, polynomial kernel, sigmoid kernel, and radial basis function (RBF) kernel. The linear kernel is suitable for linearly separable data, while the polynomial kernel can handle moderately complex decision boundaries. The sigmoid kernel is often used in binary classification tasks. In our case, we selected the RBF kernel for our SVM classifier due to its effectiveness in capturing complex relationships and handling non-linear decision boundaries.

We chose the radial basis function (RBF) kernel for our SVM classifier based on the process of hyperparameter selection. Our selection was based on evaluating different hyperparameter configurations and determining the kernel that was most effective for our classification task.

The decision function for multi-class classification with RBF kernel can be expressed as:

$$f(x) = \text{argmax}(f(i,j)(x))$$

where $f(i,j)(x)$ is the decision function of the binary classifier SVM(i, j), and argmax selects the class label with the highest decision function value.

During training, the binary SVM classifiers are trained to maximize the margin between the two classes using the following objective function:

$$minimize\left(\frac{1}{2}\|\mathbf{w}\|^2 + C\sum\left(\max(0, 1 - y(i,j)(\mathbf{w}^T\varphi(x(i,j)) + b))\right)\right)$$

subject to $y(i,j)(\mathbf{w}^T\varphi(x(i,j)) + b) \geq 1$ for all training examples $(x(i,j), y(i,j))$.

where $\varphi(x)$ represents the transformation of the input feature vector x into a higher-dimensional space using the RBF kernel function. The parameters $\mathbf{w}$ and b represent the weight vector and bias term of the SVM classifier.

The regularization parameter C controls the trade-off between achieving a large margin and allowing some misclassifications. It determines the penalty for misclassified examples.

Deep Neural Network

Artificial Neural Networks (ANN) are advanced versions of traditional machine learning techniques. They use hidden layers to assess the importance of each input for multiple outputs. The hidden layer considers both individual inputs and groups of inputs to determine their significance. Deep Neural Networks (DNN) are a specific type of ANN that have multiple hidden layers. Each layer consists of many neurons that produce outputs called activations. DNNs take advantage of this layered structure to analyze complex patterns and make more accurate predictions. The DNN algorithm has been used in this study due to its enormous computing power, especially in nonlinear datasets. [17]

Initially, the input layer is examined to determine the appropriate number of input features, which in this case is 4 for the given dataset. To determine the optimal network structure, an iterative process of trial and error is employed since there is no standard or universally optimal number of layers. In this experiment, the first hidden layer consists of 20 nodes with the rectified linear unit (ReLU) activation function, and the same configuration is applied to the second hidden layer. The output layer consists of 4 nodes(number of classes) utilizing the softmax activation function.

To optimize the model's performance, the Adam optimizer is utilized. Adam is well-known for its effectiveness in gradually descending towards the minimum of the loss function by adaptively adjusting its learning rate. To enhance training efficiency, the training process is divided into batches, where each batch represents a subset of data points processed before updating the model parameters. The overall training process involves a fixed number of epochs, which corresponds to the complete passes over the training dataset.

In this study, the model was trained using a relatively modest number of epochs, specifically 100, and a batch size of 20.

Logistic Regression

Another method used for classifying the Internet Firewall data is Logistic regression. Logistic regression is a widely used classification algorithm that predicts the probability of an event or an instance belonging to a specific class. It is a linear model that estimates the relationship between the input features and the log-odds of the target variable using the logistic function. Logistic regression is particularly useful when dealing with binary classification problems but can also be extended to handle multi-class classification.

Logistic Regression, combined with PCA for dimensionality reduction, is used for multi-class classification. PCA is performed on the input features to transform them into a lower-dimensional space while preserving the most important information. This reduces the complexity of the logistic regression model and improves its performance. The following steps are conducted to implement LR with PCA.

Algorithm 2 : Logistic Regression + PCA

1. Input: d-dimensional principal components $C = \{c_1, \cdots, c_N\}$ of original data $X \in \mathbb{R}^D$ obtained using PCA
2. for $i, \cdots, N$:
3. Compute response $y_i = \frac{y_i - P(x_i)}{P(x_i)(1-P(x_i))}$
4. Compute weights $w_i = P(x_i)(1 - P(x_i))$
5. Fit weighted least-squares regression of z_i to x_i with weights w_i
6. Assign x_i to class label using highest $P(x_i)$ value

3.4.2 Unsupervised Method(s)

For this study, the most commonly used unsupervised machine learning method, i. e. , k-Means Clustering used to group the internet firewall data into supposed classes.

k-Means Clustering

K-means clustering is an unsupervised machine learning algorithm used to partition a dataset into distinct groups or clusters based on their similarities. The algorithm aims to minimize the within-cluster sum of squares (WCSS) by iteratively assigning data points to the cluster with the closest centroid and updating the centroids based on the newly assigned points. [20] The algorithm follows these steps:

Algorithm 3 : k-Means Clustering

1. Initialization: Randomly select K centroids as the initial cluster centers.
2. Assignment: Assign each data point to the nearest centroid based on a distance metric, typically Euclidean distance.
3. Update: Recalculate the centroid of each cluster based on the mean of the data points assigned to it. The objective function of K-means clustering can be expressed as:
J=$\sum_{i=1}^{K} \sum_{x} \in_{Ci} \|\text{x - } \mu_i\|^2$
(where J: The total sum of squares (TSS) or WCSS, K: The number of clusters, C_i: The i^{th} cluster, x: A data point in the dataset, μ_i: The centroid of the i^{th} cluster)
4. Repeat Steps 2 and 3 until convergence or for a maximum number of iterations.

Unlike all of the supervised methods we have previously discussed, k-means clustering is unsupervised, meaning that there are no labels for it to predict. As a result, we have no clear notion of what is a correct result. Instead, we can try to map the clusters to the different label groupings to see if there is a correlation.

Deciding the value of k for k-means clustering was more natural since the problem predefined it; the number of clusters is the same as the number of labels in the dataset since we want the clusters to represent the different classification labels i. e, k=4 in our case.

We tried adding fitting the model on all of the selected input features and then predicting the labels for every data point. The results we obtained after this clustering and the visualizations of the clusters formed by the model are discussed in Section 4.3.

3.4.3 Dimensionality Reduction

Dimensionality reduction is a fundamental technique used in data analysis and machine learning to address the challenge of high-dimensional data. As the number of features or variables increases, the complexity of the data also grows, leading to increased computational costs, overfitting, and difficulties in visualization.

In our project, we employed dimensionality reduction methods such as Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and t-distributed Stochastic Neighbor Embedding (t-SNE) to mitigate these issues and extract meaningful insights from our data.

Principal Component Analysis (PCA)

PCA is a dimensionality reduction technique that reduces the number of variables, which helps in reducing dimensionality while preserving important information. The number of components used in PCA for this dataset is 3 because it captures a substantial amount of the variance in the original data while keeping the dimensionality manageable and retaining the most important information. [8]

Algorithm 4 : PCA

1. Input: D-dimensional scaled training set $X = \{x_1, \cdots, x_N\}$
2. Compute mean: $\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$
3. Compute covariance matrix: $\text{Cov}(x) = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x_i})(x_i - \bar{x_i})^T$
4. Compute eigenvectors $\epsilon_1, \cdots, \epsilon_D$ and corresponding eigenvalues $\lambda_1, \cdots, \lambda_D$ of $\text{Cov}(x)$
5. Obtain d largest eigenvalues and define transformation matrix $T = [\epsilon_1, \cdots, \epsilon_d]$ of their associated eigenvectors
6. Project the data X into the PCA subspace: $y_i = Tx_i, \;\; i = 1, \cdots, n$
7. return Y

The result of PCA for dimensionality reduction on this dataset is shown in Figure 8, which is attached in the Appendix. The first three principal components (3D) contribute approximately 99.97% of the total variation in the dataset and thus provides a good approximation of the variation present in the original 10-dimensional dataset.

Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a dimensionality reduction technique that aims to find a linear combination of features that maximally separates different classes in a dataset. LDA is commonly used for classification tasks where the goal is to find a projection of the data that maximizes class separability. [10]

t-distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE is a dimensionality reduction technique commonly used for visualizing high-dimensional data in a lower-dimensional space. It is particularly effective in preserving the local structure and capturing complex relationships between data points. The algorithm aims to map each data point from the high-dimensional space to a two- or three-dimensional space, while maintaining the similarity relationships between the points. [18]

Algorithm 5 : LDA

1. Compute the mean vectors for each class in the dataset.
2. Compute the scatter matrices: within-class scatter matrix (Sw) and between-class scatter matrix (Sb).
    - Sw measures the spread of data within each class.
    - Sb measures the separation between different classes.
3. Compute the eigenvectors and eigenvalues of the matrix $\text{Sw}^{-1} * \text{Sb}$.
    - The eigenvectors represent the directions of maximum class separation.
    - The eigenvalues indicate the importance of each eigenvector in the projection.
4. Select the k eigenvectors corresponding to the k largest eigenvalues to form a transformation matrix.
5. Project the original data onto the new feature subspace using the transformation matrix.
    - This reduces the dimensionality of the data while maximizing class separability.
6. Optionally, apply a classification algorithm (e. g. , logistic regression, support vector machines) to the transformed data for classification purposes.

Algorithm 6 : t-SNE

1. Compute pairwise similarities: For each data point, measure its similarity to other points using a Gaussian kernel based on the Euclidean distance.
$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2/(2\sigma_i^2))}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2/(2\sigma_i^2))}$

2. Compute conditional probabilities: Convert the similarities into conditional probabilities that represent the likelihood of choosing a neighboring point as a neighbor, given its similarity to the current point.
$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$

3. Define the joint probability distribution: Construct the joint probability distribution that reflects the similarities between all pairs of points. This is done by symmetrizing the conditional probabilities.
$q_{ij} = \frac{(1+\|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l}(1+\|y_k - y_l\|^2)^{-1}}$

4. Optimize the low-dimensional embedding: Use gradient descent to minimize the Kullback-Leibler divergence between the joint distribution in the high-dimensional space and the joint distribution in the low-dimensional space. The algorithm aims to find an embedding that accurately represents the pairwise relationships between the data points.
$KL(P||Q) = \sum_{i \neq j} p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right)$

3.5 Hyperparameters

Hyperparameters are the settings of an algorithm that can be adjusted to optimize performance, and need to be set before training the model. The process of hyperparameter tuning involves trying different combinations of hyperparameter values and evaluating the performance of each model. To avoid over-fitting, cross-validation is used, where the training set is split into multiple subsets, and the model is trained and evaluated on different subsets iteratively.

We utilized the RandomizedSearchCV and GridSearchCV from Scikit-Learn, which automate the process of trying different hyperparameter combinations and evaluating the models. RandomizedSearchCV randomly samples from a grid of hyperparameters, while GridSearchCV evaluates all possible combinations. The following table shows the hyperparameters used for the chosen models:

Table 3: Hyperparameters

Model	Parameter	Value
RF	bootstrap	True
	max_depth	8
	max_features	sqrt
	min_samples_leaf	1
	min_samples_split	3
	n_estimators	1000
SVM	C(Regularization parameter)	1000
	gamma	1
	kernel	rbf
DNN	Number of Hidden Layers	2
	Number of Neurons in Hidden Layers	20
	Activation function in Hidden Layers	ReLU
	Number of Neurons in Output Layer	4
	Activation function in Output Layer	SoftMax
k-Means	n_clusters	4
LR	n_components(PCA)	3
	solver	saga
	penalty	l1
	C	10

3.6 Evaluation Metrics

Evaluation metrics are used to assess the performance and effectiveness of machine learning models. These metrics provide quantitative measures that enable comparisons between different models or variations of the same model.

3.6.1 Supervised models

We have used the following metrics to judge the quality of the different machine learning algorithms that we have presented.

Accuracy is the number of correct classifications, divided by the total number of classifications.

$$Accuracy = \frac{\#Correctly\ Classified\ Samples}{\#Samples}$$

When dealing with highly imbalanced classes, using the accuracy metric may not provide an accurate assessment of the model's performance. [19]

We define *Precision* as the ratio of the number of class predictions that truly were that class, over the number that was predicted that class. [15]

$$Precision = \frac{TruePositives(c)}{TruePositives(c) + FalsePositives(c)}$$

A low precision indicates that we have a large number of false positives in our results.

Recall is the ratio of the number of class predictions that truly were that class, over the number of true occurrences of that class.

$$Recall = \frac{TruePositives(c)}{TruePositives(c) + FalsePositives(c)}$$

A low recall indicates that we have a large number of false negatives in our results.

The *f1 score* allows us to combine the precision and recall values, and is their harmonic mean. By combining both the precision and recall into a single metric, it provides a balance between the two values.

$$f1 = \frac{2 * Precision(c) * Recall(c)}{Precision(c) + Recall(c)}$$

In all of the equations above, c represents each class in the multi-class classification problem, and the True Positives, False Positives, and False Negatives are calculated for each class separately.

When dealing with imbalanced class distributions, traditional evaluation metrics like accuracy can be misleading. This is because accuracy does not take into account the class imbalance and can be dominated by the majority class, leading to inflated performance results. On the other hand, evaluation metrics such as F1-score provide a more balanced assessment of model performance in the presence of imbalanced classes. As shown in Figure 1, we have an imbalanced class distribution, with 'allow' being the dominant class. So, we are using **F1-score** as the key evaluation metric for our dataset.

3.6.2 Unsupervised models

Unsupervised methods lack well-defined evaluation metrics, unlike supervised methods, as they do not have access to correct labels for assessing the results. However, visual representations can serve as valuable tools for understanding the outcomes of unsupervised classification algorithms. Therefore, we are are mostly relying on visualization as a metric to interpret the performance of unsupervised models.

However, since it is not feasible for a machine to judge model performance based on visual results, we did use 2 metrics to study the performance of the k-means clustering model.

Inertia measures the compactness of the clusters produced by k-means. The lower the inertia, the better the clustering, as it indicates that the data points within each cluster are closer to their centroid. Inertia tends to decrease as the number of clusters (k) increases.
It is calculated as the sum of squared distances between each data point and its nearest cluster centroid.

$Inertia = \sum_{i=1}^{n} \min_{c \in C} ||x_i - c||^2$

The *Silhouette score* measures the cohesion and separation of clusters. It evaluates how well each data point fits into its assigned cluster compared to other clusters. The silhouette score ranges from -1 to 1, where a value close to 1 indicates a well-clustered data point, a value close to 0 indicates a data point on or near the decision boundary between clusters, and a value close to -1 indicates a data point that may have been assigned to the wrong cluster. A higher average silhouette score indicates better clustering performance.
The equation for the silhouette score of a data point $x_i$is:

$s_i = \frac{b_i - a_i}{\max(a_i, b_i)}$

Both inertia and silhouette score provide insights into the quality and separation of the clusters obtained by k-means. However, since these metrics need to be used in combination with domain knowledge and other evaluation techniques to assess the effectiveness and interpretability of the clustering results, we also incorporated our understanding of the clusters we finally obtained.

4 Data Analysis

4.1 Test Bed

The testbed for this project is designed to assess the performance of various classification algorithms. As mentioned in the previous sections, our dataset consists of 57170 that have been divided into training and test sets.

The testbed environment is implemented using Python. Python seemed the most appropriate language of choice for the project; its use is common for machine learning programs due to the availability of many libraries, which makes implementation easier. Jupyter Notebooks have been employed for implementation, providing the advantage of running individual cells for efficient testing. It is beneficial when performing machine learning since you do not have to run the preprocessing each time. Also, the notebook provides visual outputs so that you can see graphs or tables directly.

Popular machine learning libraries such as numpy, pandas, seaborn, TensorFlow, and scikit-learn have been used for data manipulation, visualization, model training, and evaluation.

4.2 Questions of Interest

- Question 1: To what degree does the choice of feature selection affect the performance of a model?
- Question 2: Which supervised method demonstrates the best performance in classifying this data?
- Question 3: Do other supervised methods (DNN, SVM, and LR) perform better than the benchmark algorithm, Random Forest for classifying the Internet Firewall data in any scenario?
- Question 4: How successful are unsupervised algorithms in accurately clustering the firewall data ?
- Question 5: Did deep learning outperform machine learning in terms of classification?

4.3 Results

Unsupervised Models
Although it is not conventional to obtain typical metrics like accuracy, precision, recall, and f1-score for an unsupervised learning/clustering method such as k-means clustering, in our project, we have tried to do so to supplement our findings through visualization so that a machine is also able to draw comparisons amongst all methods.
Thus, our metric table looks like:

Table 4: Performance of supervised models using selected features

Model	Accuracy	Precision (macro avg)	Recall (macro avg)	f1-score (macro avg)	Inertia	Silhouette score
k-MC	55.76%	0.58	0.52	0.49	1881.9128	0.5778

From this table, it can be seen that while the typical metrics are as expected from a model that has never seen the labels, the inertia value of nearly 1882 suggests that the data points in our clustering result have relatively high distances from their assigned centroids. This could indicate that the clusters are not well-separated or that there is a high level of variation within the clusters. However, the silhouette score of about 0.58 (which is generally considered decent) suggests that the clusters in your k-means clustering result have a moderate level of separation and cohesion. Since the silhouette score is in line with the other metrics measured, it can be considered to be the most reliable judge of our model.

However, since this is an unsupervised model, the discussion of its results would be incomplete without visualizations of its clusters. We were able to visualize 4 distinct clusters using 3 different dimensionality reduction techniques (as discussed above); PCA, LDA and t-SNE. The results were as follows.

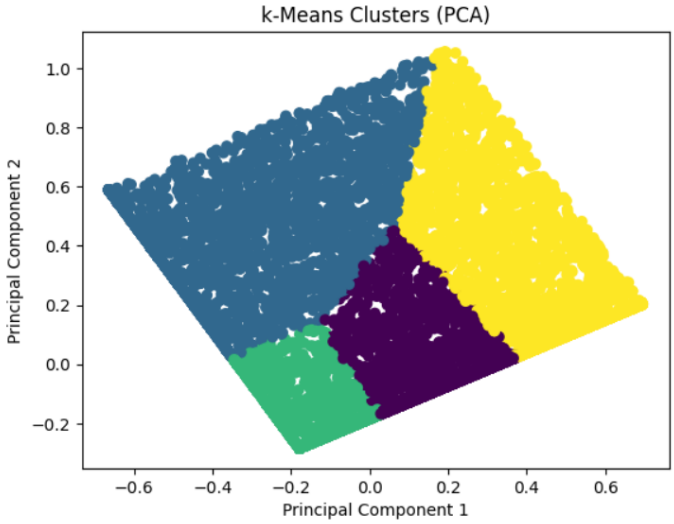


Figure 2: Visualization of k-means clusters using PCA

The PCA-based graph reveals 4 distinct clusters obtained from the k-means algorithm. These clusters demonstrate clear separation, indicating significant differences among the data points within each cluster. The absence of overlap or ambiguity suggests successful clustering, where each data point is appropriately assigned based on proximity to the cluster centroid. This visualization provides valuable insights into the underlying patterns and structures in the dataset, validating the effectiveness of the k-means algorithm for clustering analysis.

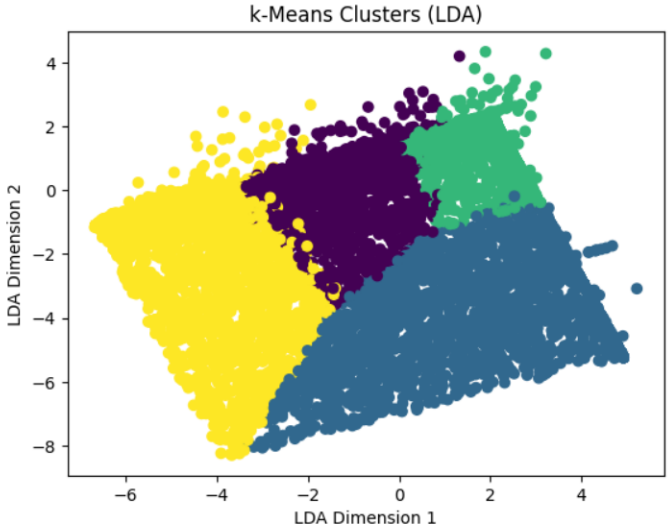


Figure 3: Visualization of k-means clusters using LDA

The LDA-based graph showcases 4 clusters derived from the LDA algorithm. Although the clusters are not as tightly packed as in the PCA visualization, they still exhibit discernible separation. The slight overlap among clusters suggests some degree of similarity or overlap in the feature space. Despite this, the LDA-based clustering highlights meaningful differences between the clusters, indicating the algorithm's ability to capture distinct patterns and discriminative information in the data. This visualization further supports the effectiveness of LDA in identifying relevant features and classifying data points into separate groups.

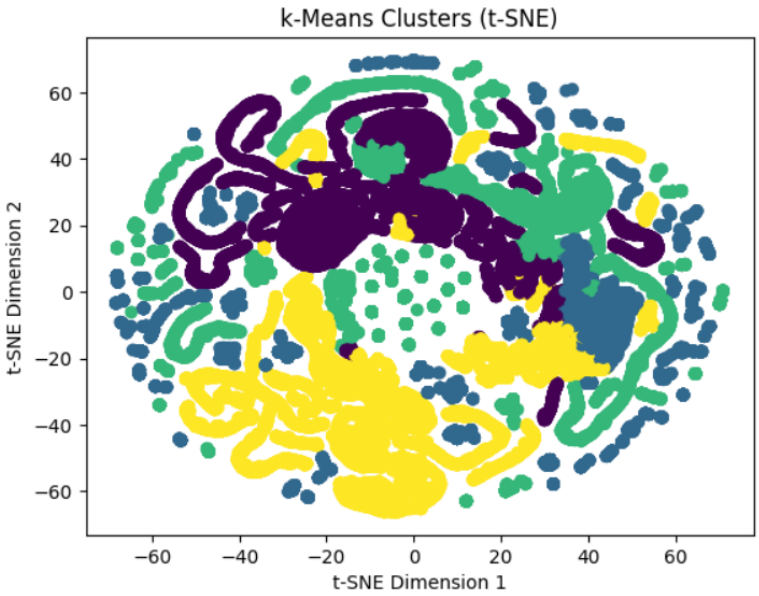


Figure 4: Visualization of k-means clusters using t-SNE

The t-SNE visualization, on the other hand, reveals a complex arrangement of data points, where clusters are scattered throughout the graph without clear boundaries. While some clusters appear to be relatively well-defined, others exhibit a more dispersed and intertwined nature. The proximity of certain clusters suggests a degree of similarity or shared characteristics, although they remain distinct entities. Despite the lack of well-defined boundaries, the t-SNE representation captures intricate patterns and relationships within the data. This unique visualization highlights the challenge of separating certain clusters due to their close proximity, indicating the presence of intricate and complex underlying structures in the dataset.

Overall, it can be seen the model successfully identified 4 distinct clusters in the data with performance evaluation metric scores decent for an unsupervised algorithm.

Supervised Models

After classifying the internet firewall data using supervised models, the performance was evaluated using accuracy, precision, recall, and F-measure metrics. The table below presents the results obtained as part of this evaluation:

Table 5: Performance of supervised models using selected features

Model	Accuracy	Precision (macro avg)	Recall (macro avg)	f1-score (macro avg)
RF	99.86 %	0.998	0.873	**0.8913**
SVM	99.36%	0.7421	0.7405	**0.7412**
DNN	99.78%	0.7470	0.7480	**0.7475**
LR + PCA	98.25%	0.7208	0.7319	**0.7298**

From the above table, it is evident that the Random Forest model performs better for the classification of internet firewall data, with an accuracy = 99.86%, precision = 0.92, recall = 0.87, and an f1-score = 0.89.

Now, let us examine the performance of each model using the main evaluation metric, which is the F1-score. The graph below illustrates the F1-scores achieved by each classification model:

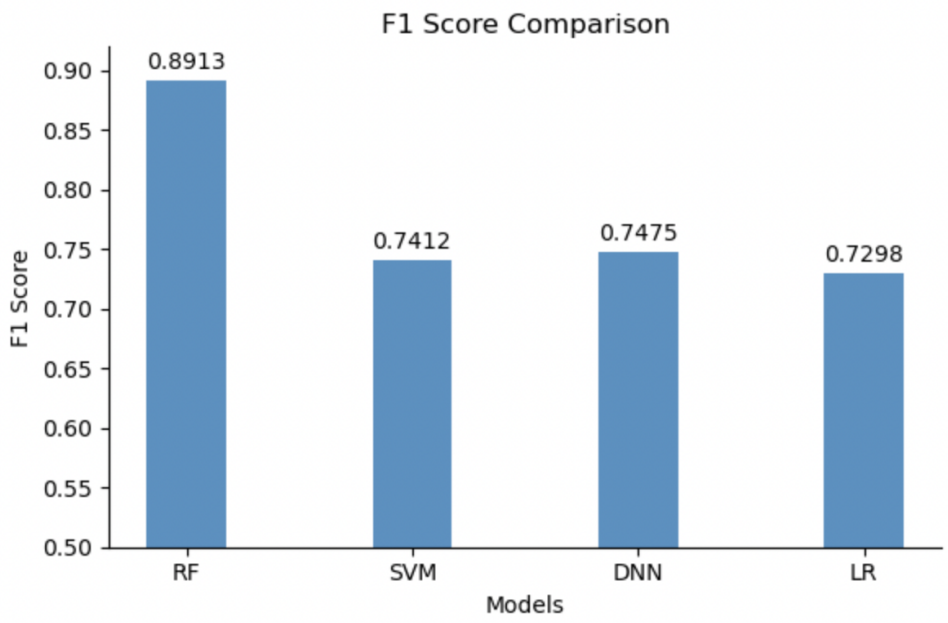
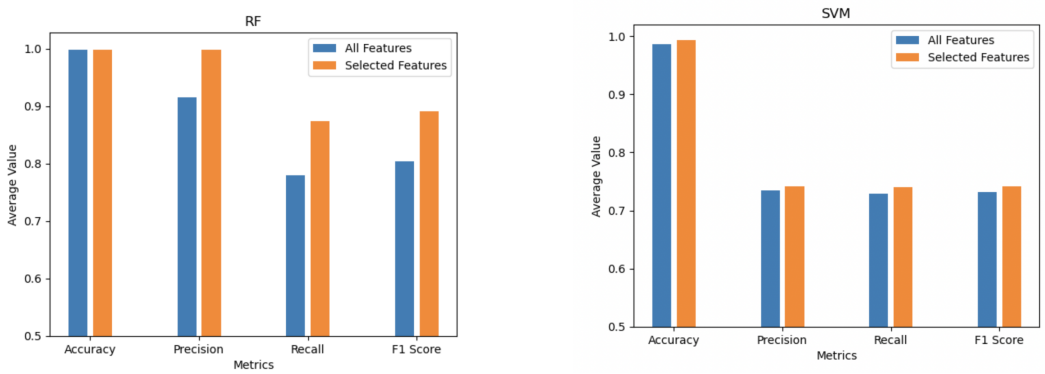


Figure 5: Comparison of F1-scores for each model. RF=Random Forest, SVM=Support Vector Machine, DNN=Deep Neural Network, LR=Logistic Regression

The F1-scores obtained for the models are as follows: RF=0.89, SVM=0.7412, DNN=0.7475, LR=0.7298. These results demonstrate that the **Random Forest classifier** outperformed the other models, followed by the Deep Neural Network classifier.

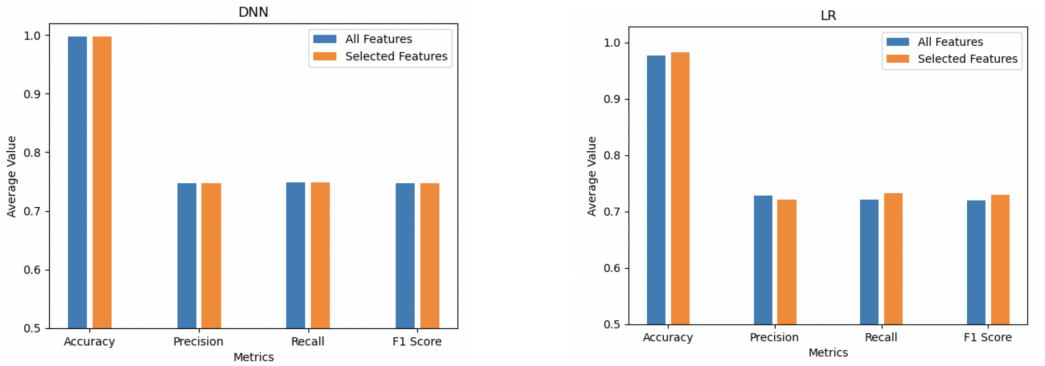
Furthermore,we would like to include graphs that show the performance difference between classifiers using all features and classifiers using selected features.

As shown in the below figure, RF achieved the best F1-score in both cases, with F1-scores of 0.8038 and 0.8913, respectively. Moreover, all the models except DNN (shown no difference) achieved better results using selected features compared to using all the features. This highlights the impact of feature selection, especially for the RF algorithm, whose F1-score increased by approximately 10.875% after feature selection.



(a) Random Forest

(b) Support Vector Machine



(c) Deep Neural Network

(d) Logistic Regression

Figure 6: Comparison between results achieved with selected features and all features by each of the classifiers. Blue=All Features and Orange=Selected Features

5 Conclusion and Discussion

Considering the crucial role of firewalls in safeguarding system security, this study focused on developing a range of machine learning (ML) and deep learning (DL) models with the ability to determine the suitable action for sessions in firewall logs. A comprehensive analysis was conducted on a dataset consisting of 57,170 logs, evaluating the performance of five multi-class classification algorithms in classifying actions as "Allow," "Drop," "Deny," or "Reset-both." The modeling process involved essential steps such as feature selection and hyperparameter tuning which aided in improving the performances of the classification models. The Random Forest classifier demonstrated the highest accuracy and F1-score, achieving an accuracy of 99.86% and an F1-score of 0.89 with the selected four features: Destination Port, NAT Source Port, Elapsed Time (sec), and Packets Received. These findings highlight the efficacy of the Random Forest classifier in accurately classifying the internet firewall data.

Discussion

Various learning methods were employed to classify the Internet Firewall data. Based on the results obtained, it appears that Deep Neural Network, Support Vector Machine, and Logistic Regression do not outperform the benchmark method, Random Forest. In the classification task performed on the firewall data, Random Forest consistently demonstrated superior performance in terms of accuracy, precision, recall, and f1-score. While DNN, SVM, and LR showed competitive performance, they were unable to surpass the classification performance achieved by RF. These findings suggest that for this particular dataset, Random Forest is a robust and reliable classification method that outperforms

alternative supervised techniques. This supports the findings of previous studies conducted using the same data.

As for the unsupervised k-Means Clustering, the performance it gave was quite decent, especially when we consider its relevance in the real-world scenario where most data is unlabelled but categorizable. With a silhouette score of nearly 0.58, it turned out to be a fairly dependable grouping algorithm for unlabelled data.

Additionally, the choice of feature selection appears to have some impact on the performance of the models, particularly Random Forest. In the case of RF, the feature selection technique influences the subsets of features considered at each split, potentially affecting the model's overall accuracy. However, for other supervised models such as LR and SVM, the impact of feature selection on performance is not as pronounced. This suggests that these models are more robust to the inclusion or exclusion of specific features, potentially due to their ability to learn and adapt to the available data regardless of feature selection. Nevertheless, further investigation and experimentation with different feature selection approaches are recommended to fully understand the potential impact on model performance across various classification tasks and datasets.

While this study provides valuable insights into the classification of Internet Firewall data, there are several limitations that should be acknowledged. The performance comparison is limited to a specific set of supervised methods, namely Deep Neural Network, Support Vector Machine, Logistic Regression, and Random Forest. Other classification algorithms or ensemble methods could yield different results. Additionally, the study focuses on a specific dataset, and the findings may not generalize to other firewall datasets with different characteristics. Furthermore, the performance evaluation metrics used, such as accuracy, precision, recall, and f1-score, may not capture all aspects of model performance. It is important to consider other evaluation metrics or conduct additional analysis to gain a comprehensive understanding of the models' effectiveness. Lastly, the impact of hyperparameter tuning and data preprocessing techniques on the performance of the models is not explored in detail, and further investigation in these areas could provide additional insights.

Possible future work includes:

- Explore alternative classification algorithms and ensemble methods to compare their performance with the evaluated models.
- Investigate the impact of different feature selection techniques on model performance across various classification tasks and datasets.
- Test the generalizability of findings to other firewall datasets with different characteristics.
- Incorporate additional evaluation metrics such as AUC-ROC, specificity, and balanced accuracy for a comprehensive assessment of model performance.
- Optimize hyperparameter tuning and data preprocessing techniques to enhance model accuracy and generalization capabilities.
- Implement and deploy the developed models in real-time firewall systems, considering factors like computational efficiency and scalability.

References

[1] Internet firewall data. *UCI Machine Learning Repository*, 2019.

[2] I. Ahmad, M. Basheri, M.J. Iqbal, and A. Raheem. Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection. *Computational Intelligence and Neuroscience*, 2018, 2018.

[3] H.N.K. AL-Behadili. Decision tree for multiclass classification of firewall access. *International Journal of Intelligent Engineering Systems*, 14(2):294–302, 2021.

[4] H.E. As-Suhbani and S.D. Khamitkar. Classification of firewall logs using supervised machine learning algorithms. *International Journal of Computer Science and Engineering*, 7(2):301–304, 2019.

[5] Rung-Ching Chen, Christine Dewi, Su-Wen Huang, and Rezzy Eko Caraka. Selecting critical features for data classification based on machine learning methods. *SN Computer Science*, 1(1):31, 2020.

[6] Clarence Chio and David Freeman. *Machine Learning and Security: Protecting Systems with Data and Algorithms*. O'Reilly, 1st edition, 2018.

[7] Pratap Dangeti. *Statistics for Machine Learning*. Packt Publishing Ltd, 2017.

[8] Chris Ding and Xiaofeng He. K-means clustering via principal component analysis. In *Proceedings of the Twenty-First International Conference on Machine Learning*. Association for Computing Machinery, 2004.

[9] F. Ertam and M. Kaya. Classification of firewall log files with multiclass support vector machine. In *Proceedings of the 6th International Symposium on Digital Forensic and Security (ISDFS)*, volume 2018, Antalya, Turkey, March 2018.

[10] Samantha Gardner-Lubbe. Linear discriminant analysis for multiple functional data analysis. *Journal of applied statistics*, 48(11):1917–1933, 2020.

[11] Bissan Ghaddar and Joe Naoum-Sawaya. High dimensional data classification and feature selection using support vector machines. *Decision Support Systems*, 56:438–449, 2013.

[12] A. Gholamy, V. Kreinovich, and O. Kosheleva. Why 70/30 or 80/20 relation between training and testing sets: A pedagogical explanation. 2018.

[13] A. K. Meena and N. Hubballi. Nviz: An interactive visualization of network security systems logs. In *2020 International Conference onCOMmunication Systems NETworkS (COMSNETS)*, pages 685–687, 2020.

[14] scikit-learn. Ensemble methods, n.d. Retrieved June 15, 2022, from `https://scikitlearn.org/stable/modules/ensemble.html?highlight=why+use+random+forest+classifier`.

[15] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[16] D. Sharma, V. Wason, and P. Johri. Optimized classification of firewall log data using heterogeneous ensemble techniques. In *Proceedings of the 2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, Greater Noida, India, 2021.

[17] A. Shrestha and A. Mahmood. Review of deep learning algorithms and architectures. *IEEE Access*, 7:53040–53065, 2019.

[18] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[19] Jeremy Watt, Reza Borhani, and Aggelos K. Katsaggelos. Linear two-class classification. In *Machine Learning Refined: Foundations, Algorithms, and Applications*, pages 125–173. Cambridge University Press, 2nd edition, 2020.

[20] Mohammed J. Zaki and Wagner Meira Jr. Representative-based clustering. In *Data Mining and Machine Learning: Fundamental Concepts and Algorithms*, pages 334–363. Cambridge University Press, 2nd edition, 2020.

Appendix

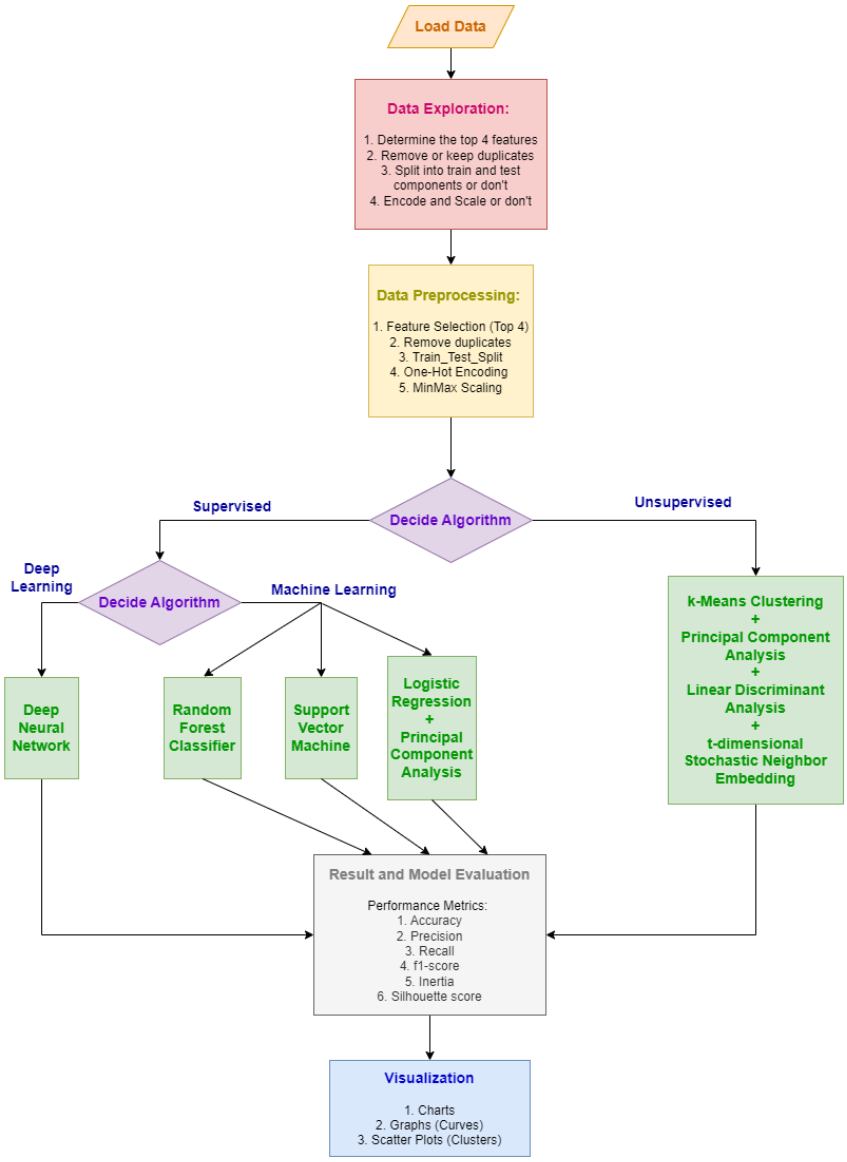


Figure 7: A flowchart summarizing what we did in our project

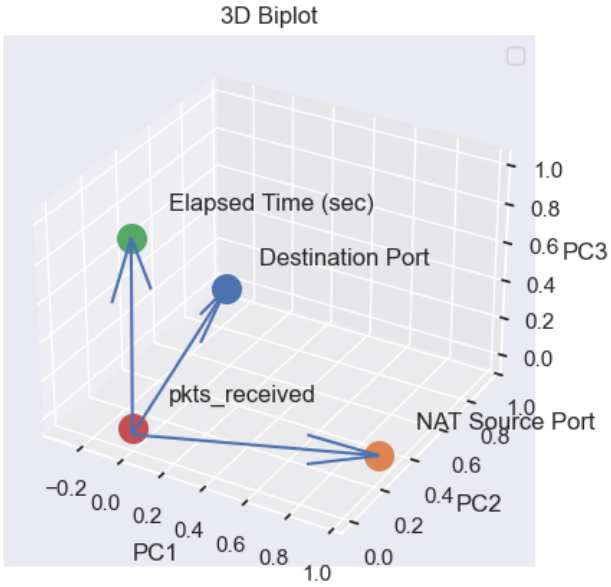


Figure 8: 3D PCA

Team

Table 6: Team Members & Tasks

Name	Task Completed
Kay Royo	Feature Selection, PCA + LR
Rishika Garg	RF, k-Means Clustering
Dhanusha Pathakota	DNN, SVM

Links

Repository

`https://github.com/kayannr/internet_firewall`

Dataset

`https://archive.ics.uci.edu/ml/datasets/Internet+Firewall+Data#`