**Machine Learning Project**

**CSAI 253**

**Section 1**

**Done By:**

**Farida Ali 202301940**

**Malak Khaled 202300369**

**Omar Mohamed 202300785**

**Azza Monir 202300067**

**Network Intrusion Detection**

***Problem Statement***

The aim of this project is to build a machine learning model to classify the dataset that contains both numerical and categorical features, with the target column "class". The significance of the project is the ability to generalize unseen data by developing classification models to detect intrusion in a network and specify it as "normal" or "anomaly".

***Data Exploration***

*1- Structure:*

Dataset contains categorical and numerical features,27 numerical columns and 3 categorical ones with the target column labeled **"class"**, which indicates whether the connection is normal or an anomaly.

*2- Missing Data:*

In the dataset provided there were no missing values labelled "NaN", so we handled the data as there are no nulls. (verified using isnull() method).

*3- Duplicates:*

Duplicated rows were removed to ensure data quality and uniqueness.

*4- Feature Division:*

Data were divided into numerical (int64 and float64) and categorical (object) features.

*5- Inconsistencies:*

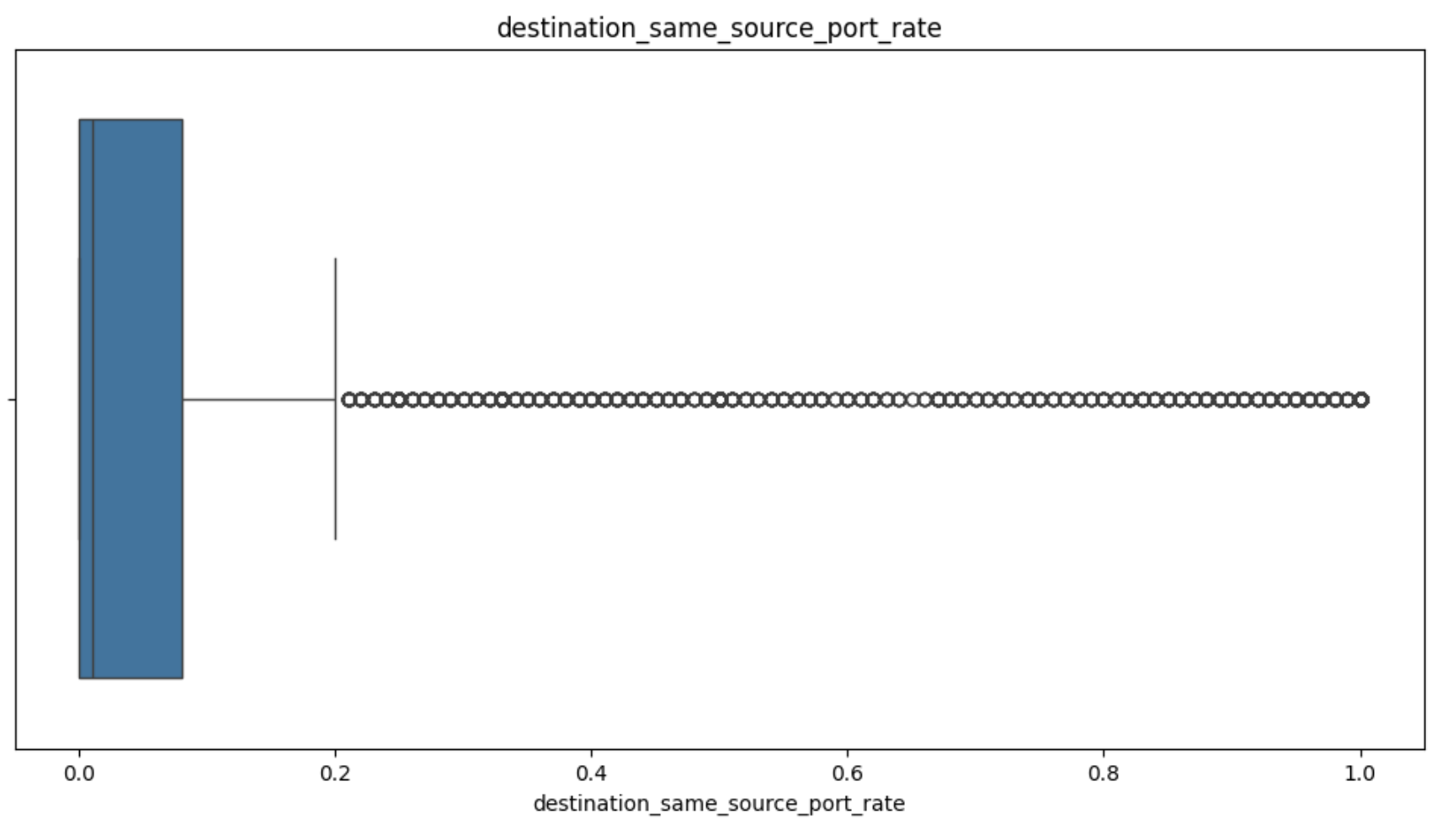
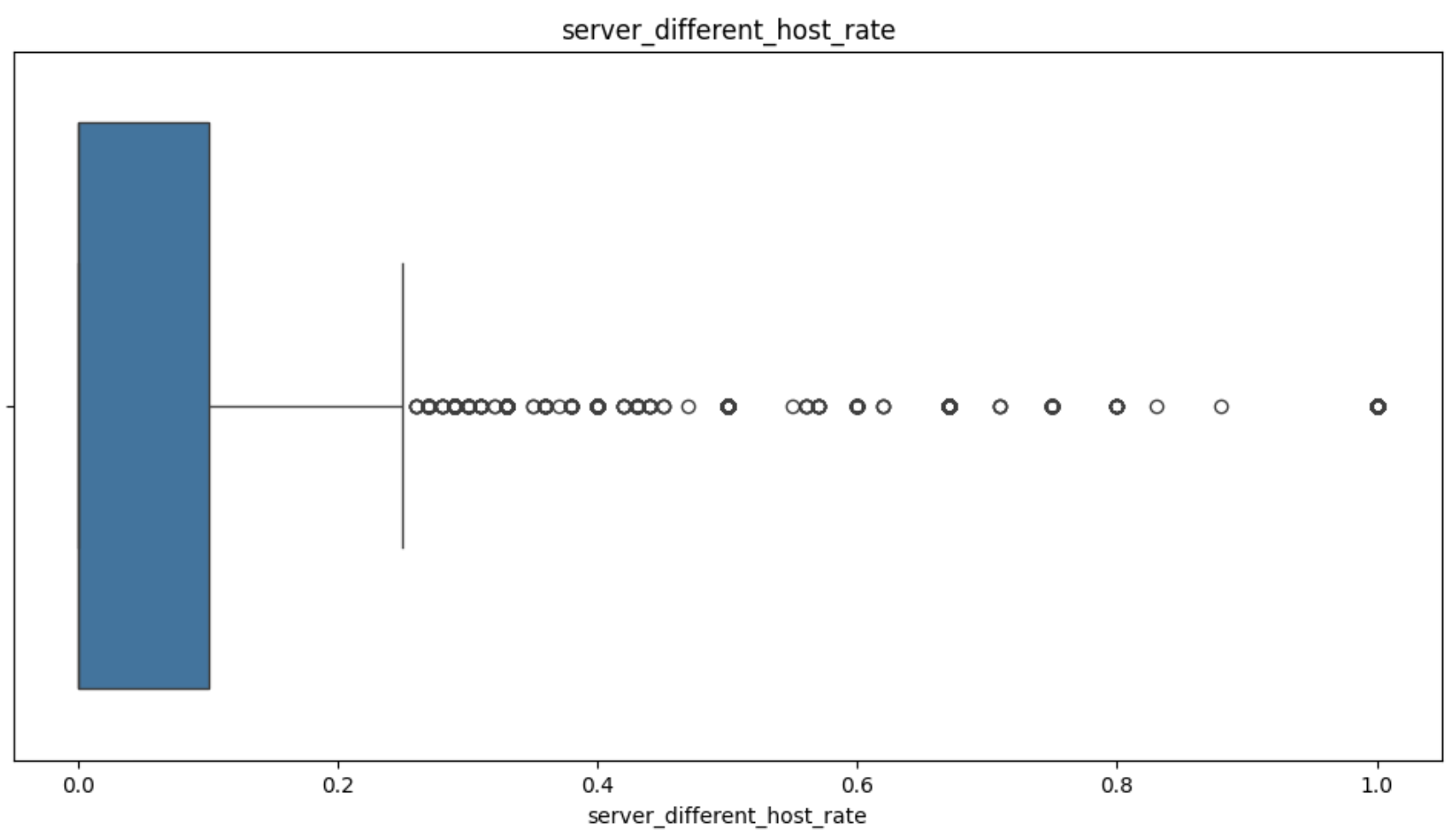
By checking the unique values in all categorical features, we checked that the dataset is consistent.

***Data Visualization***

*1- boxplots:*

Each numerical feature was visualized using a boxplot to detect outliers, which were handled using robust scaling techniques.

Boxplot samples:

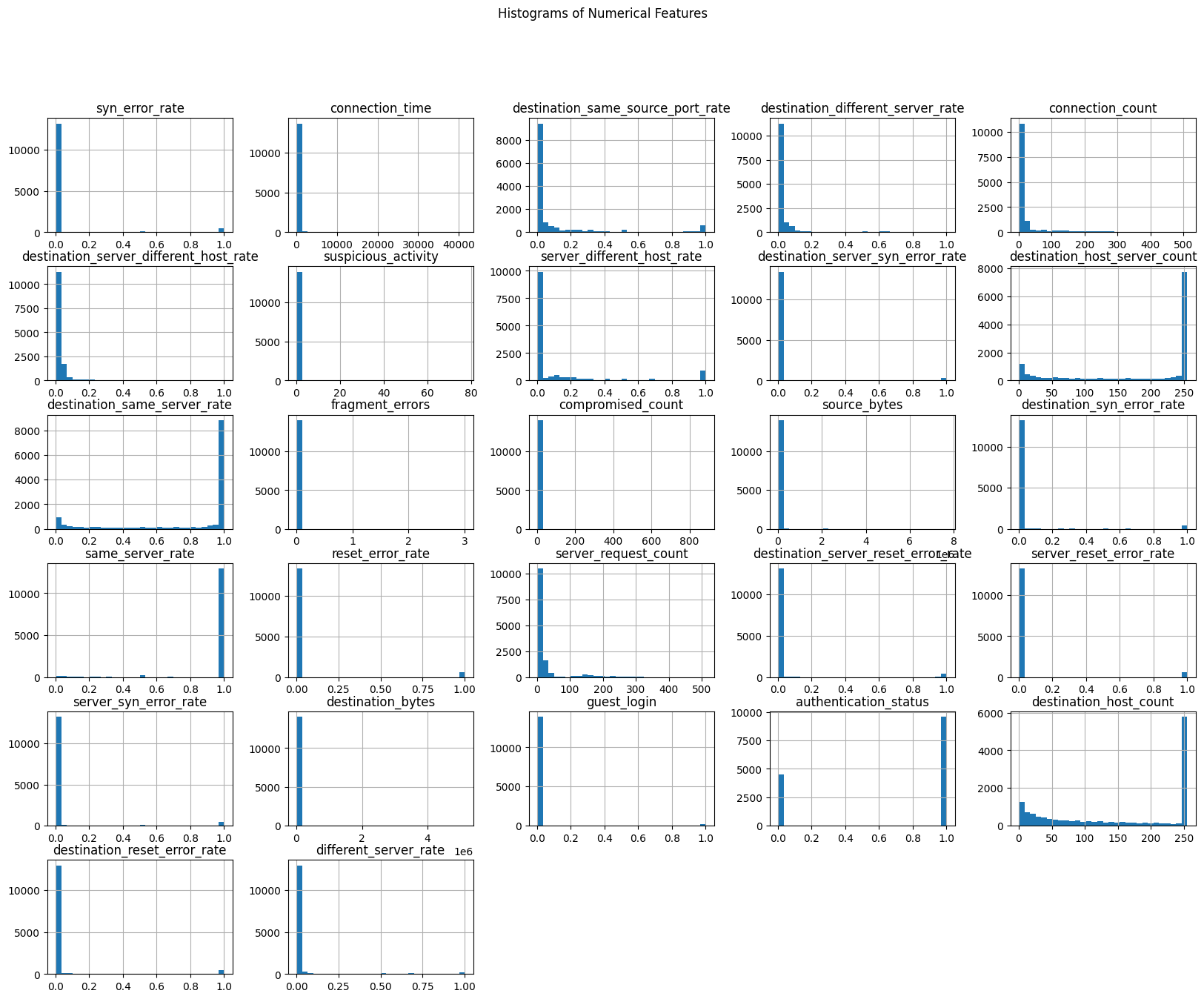


Results:

Most numerical features contain a huge number of outliers, and removing them will heavily reduce the dataset and affect training models; so, a robust scaler will be used to address this issue.

*2- Histograms:*

We created histograms for numerical features to check the data distribution. This helped identify skewed distributions that were addressed by scaling.



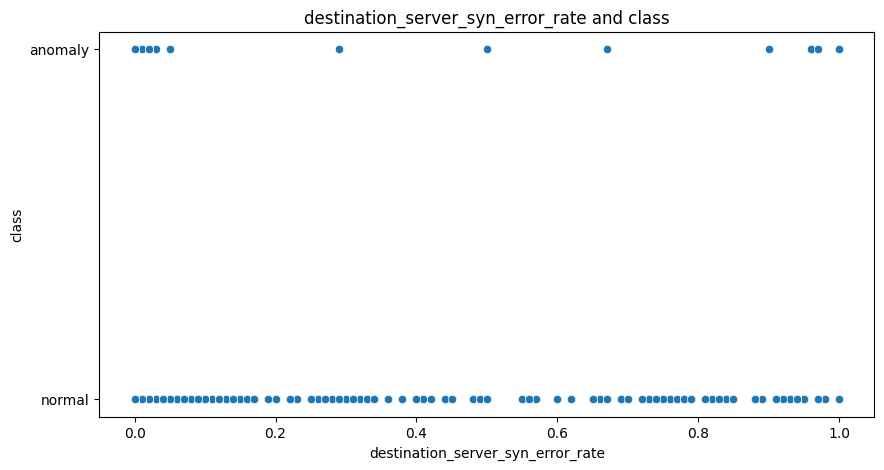
Results:

Most features were normal, but some were skewed; so, they will be handled by scaling.

*3- Scatterplots:*

Scatterplots were used to visualize relationships between features and the target. Features like **connection time** showed clear separation between normal and anomalous classes, aiding in classification.

Scatterplot samples:

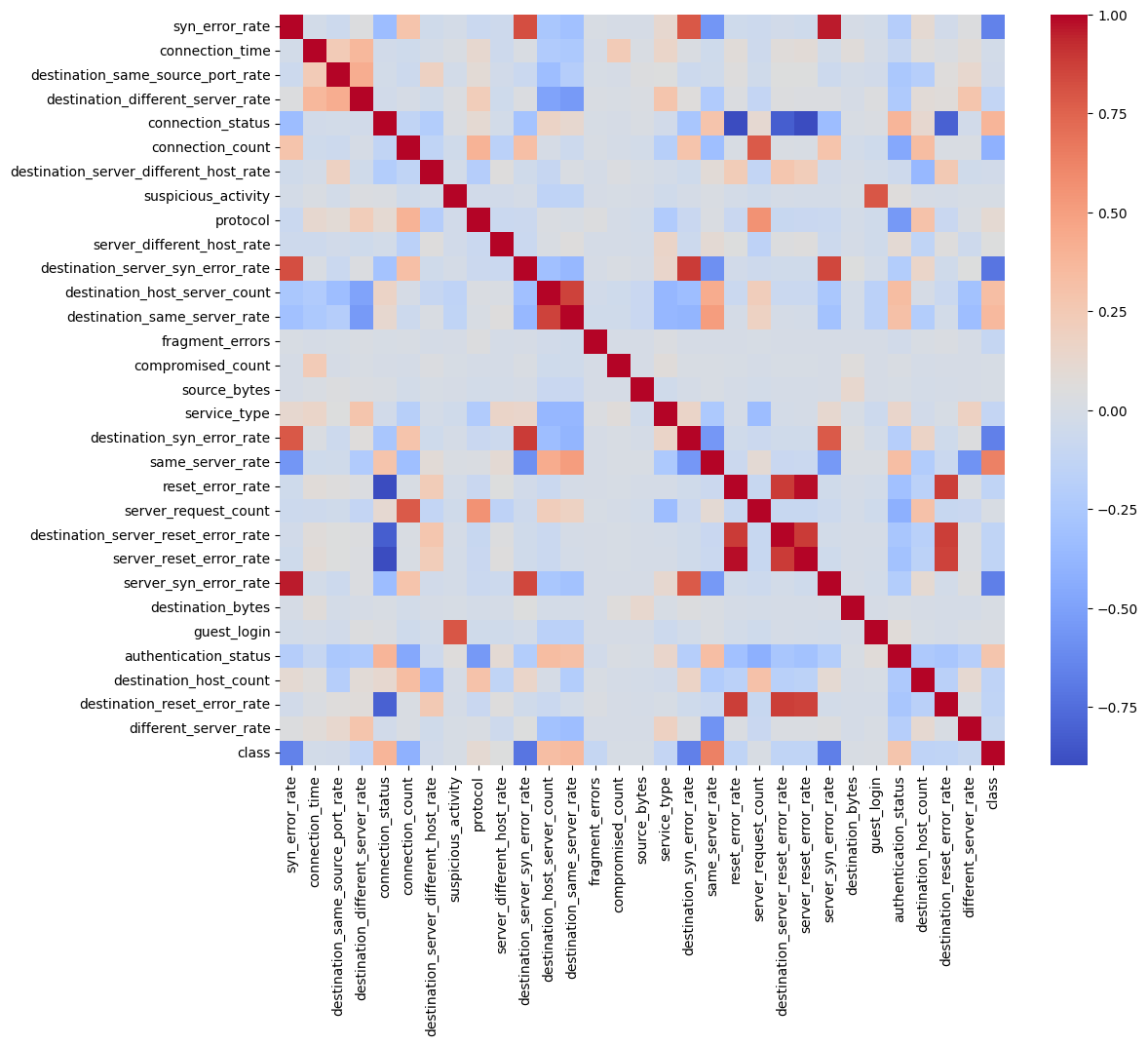


Results:

Some features displayed clear separation between classes, while other features showed significant overlap, indicating lower predictive power.

*4- Correlation Heatmap:*

A heatmap was used to visualize the correlation matrix and detect highly correlated features with the target.



Results:

Some features were moderately correlated with each other and the target variable, although none showed significantly high correlation.

***Data Preprocessing***

The data was cleaned and preprocessed before implementing the classification models.

Steps:

*1-* *Encoding Categorical Features:*

**Process**: All the categorical features were encoded using the Label Encoding technique to transform it to quantitative features suitable for machine learning models.

**Reason for Label Encoding**: The number of unique values per categorical feature was relatively low and ordinal relationships were not required, label encoding was sufficient.

This ensured compatibility with tree-based models like Random Forest and Decision Tree, which can work with integers representing categories.

*2- Scaling and Handling Outliers:*

**Process**: Visualized numerical features using boxplots to detect outliers and found that the dataset contains a large number of outliers.

**Handling Technique**: To avoid data loss, we used RobustScaler instead of removing outliers.

**Reason for using RobustScaler**: Unlike StandardScaler (which uses mean and variance), RobustScaler uses median and interquartile range (IQR), making it resistant to outliers. It scales features so that outliers have less influence on model training instead of completely removing them.

*3-* *Feature Selection:*

**Process**: We tried dropping unnecessary columns manually just by checking the heatmap, but this was inefficient. So, we decided to select the important features using a Random Forest Classifier.

**Technique:** We trained a Random Forest Classifier on all available features. Then, we extracted feature importance scores based on how useful each feature was. Finally, we selected the top 20 important features.

**Top Features:**



*4- Data Splitting:*

**Process**: The data was split into 80% training and 20% testing using the train\_test\_split function from scikit-learn.

*5- Imbalance Class:*

**Process:** We found that the target feature (class) was imbalanced, it contained more samples in one class (normal) and very few in the other (anomaly). So, we handled this issue by using the over-sampling technique, SMOTE. This ensured that the dataset is now balanced, containing an equal number of samples in both classes.

**Technique:** We applied SMOTE (Synthetic Minority Over-sampling Technique) on the training set. This technique creates synthetic samples of the minority class by interpolating between existing minority class samples.

***Data Exploration and Preprocessing Conclusion***

The dataset is now ready for applying machine learning models. We gained insights and statistics on the dataset, Then, the data was encoded and cleaned from duplicates, outliers, missing values, and inconsistencies. The data was also visualized and then scaled and split into training and testing sets. Finally, the training set was balanced through SMOTE oversampling.

***Model Selection, Implementation and Evaluation***

1. The following four models were implemented for binary classification and one ensampling technique:

* Decision Tree Classifier
* Stochastic Gradient Descent (SGD) Classifier
* Logistic Regression
* K-Nearest Neighbors (KNN)
* Soft Voting Classifier

***Model Evaluation***

1. In this phase, we evaluate the performance of the models implemented using various metrics to determine which model best classifies and predicts the network intrusions.

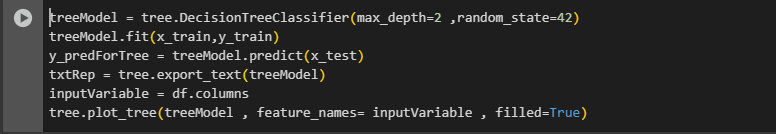
The models are evaluated based on the following metrics:

* Accuracy Score: The proportion of correct predictions (both normal and anomalous).
* Confusion Matrix: A visual representation of the number of true positives, false positives, true negatives, and false negatives.
* Precision Score: The proportion of true positive predictions (anomalies) out of all predicted anomalies.
* Recall Score: The proportion of true positive predictions (anomalies) out of all actual anomalies in the dataset.
* F1 Score: The harmonic mean of precision and recall, providing a balanced measure.
* ROC Score: This score evaluates the ability of the model to distinguish between the two classes (normal and anomaly), with higher values indicating better performance.
* Classification Report: Provides a comprehensive summary of the performance of a classification model

**1. Decision Tree rationale** :

1. Easy for understanding and visualization.
2. Capture the non-linear relationships.

**Decision tree implementation:**

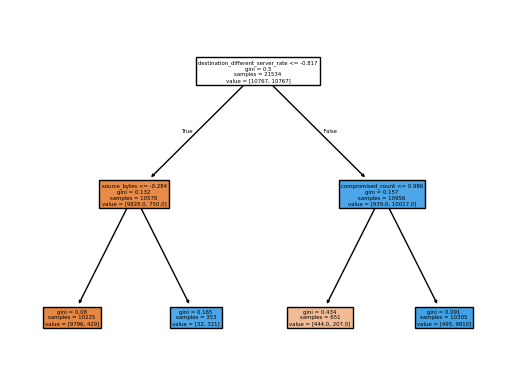


This is the model implementation after being imported from scikit-learn library. The 1st line initializes the model with maximum depth = 2 which limits the depth of the tree (prevents overfitting). The 2nd line fits the model with the x and y training data.

The 3rd line predicts the model data using the x\_test data. The 4th line is required for representing the output of the model in textual representation. The 5th and the 6th line are required for the visualization of the tree.

**Evaluation**:

* **Accuracy**: The Decision Tree showed moderate accuracy, but it tends to **overfit** when the tree depth is not constrained.
* **Precision**: Precision was satisfactory, but not the best, as the model struggles with classifying some anomalies.
* **Recall**: Recall is relatively high, but the model misses some anomalies, which could be dangerous for intrusion detection.
* **F1-Score**: The F1-Score was decent, but the overfitting issue lowered its generalization capability.
* **Confusion Matrix**: Shows an imbalance between true positives and false negatives. This is expected due to the overfitting nature of the tree.



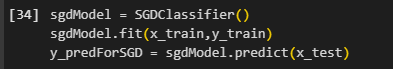
**Improvements**:

* Pruning the tree could help reduce overfitting and increase generalizability.

**2.Stochastic Gradient Descent rationale** :

1. Minimizes the cost function.
2. Faster and more suitable for the large dataset that is used

**SGD Implementation:**



The 1st line initializes the model that was previously imported from scikit-learn library. The 2nd line fits the data to the model using the x,y train data. The last line predicts the model data output that will later on be compared with the model actual data to calculate the model accuracy.

**Evaluation**:

* **Accuracy**: The model performed well on linear relationships but struggled with more complex decision boundaries.
* **Precision**: Precision was satisfactory for detecting normal connections but lower for detecting anomalies.
* **Recall**: Recall was moderate; the model had difficulty catching all anomalies.
* **F1-Score**: The F1-Score showed improvement compared to Logistic Regression but was still lower than that of the Soft Voting Classifier.
* **Confusion Matrix**: The matrix showed more false negatives, indicating that SGD might miss some anomalies.

**Improvements:**

* Hyperparameter tuning, specifically adjusting the learning rate and the number of iterations, could improve model performance.

**3.Logistic Regression rationale** :

1. Good for threshold tuning as its output is a probability of the sample belonging to a class.
2. Easy to interpret using coefficients.

**Logistic Regression Implementation:**

****

The 1st line initializes the model that was previously imported from scikit-learn library. The 2nd line fits the data to the model using the x,y train data. The last line predicts the model data output that will later on be compared with the model actual data to calculate the model accuracy.

**Evaluation**:

* **Accuracy**: Logistic regression performed well in identifying normal connections but struggled with anomalies.
* **Precision**: Precision was low because the model was more likely to classify normal connections as anomalies.
* **Recall**: The recall was also lower, as the model couldn't detect many anomalous connections.
* **F1-Score**: The F1-Score was lower compared to the Decision Tree and Soft Voting Classifier.
* **Confusion Matrix**: Logistic regression showed a higher number of false positives, which can be an issue when detecting intrusions.

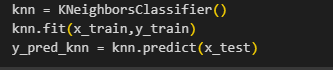
**Improvements**:

* Logistic regression may benefit from **feature interaction** terms, which could improve its performance in this non-linear task.

**4.K-Nearest Neighbour rationale** :

1. Determines the most relevant data points to predict unknown values accurately.

**KNN Implementation:**

****

The 1st line initializes the model that was previously imported from scikit-learn library(by setting the number of the k neighbours to 5 to prevent over/under fitting). The 2nd line fits the data to the model using the x,y train data. The last line predicts the model data output that will later on be compared with the model actual data to calculate the model accuracy.

* **Evaluation**:  
  + **Accuracy**: KNN performed reasonably well, but it was slow with larger datasets due to its reliance on calculating distances.
  + **Precision**: Precision was decent, with the model able to identify true anomalies in many cases.
  + **Recall**: Recall was low because KNN tends to favor the majority class and is computationally expensive as the dataset size grows.
  + **F1-Score**: The F1-Score was acceptable but did not outperform other models like Soft Voting.
  + **Confusion Matrix**: The model showed a mix of false positives and false negatives, but with fewer false positives compared to Decision Trees.

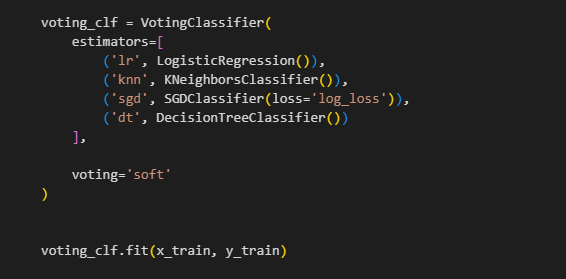
**Improvements**:

* The performance of KNN can be optimized by tuning the **k** value and applying **distance weighting** to give more importance to closer

**5.Soft Voting rationale (Ensemble Method):**

1. The **Soft Voting Classifier** combines the predictions of multiple models, allowing the ensemble to take the strengths of each model of them (Decision Tree, KNN, Logistic Regression, and SGD).

**Voting ensemble Implementation:**



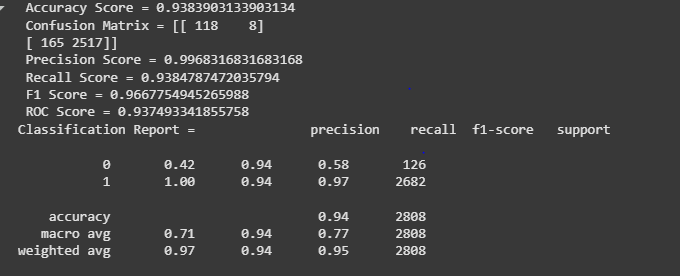
The 1st line initializes the model that was previously imported from scikit-learn library. Instead of relying on a single algorithm, it combines four different ones :

Logistic Regression, K-Nearest Neighbors (KNN), Stochastic Gradient Descent (SGD), and a Decision Tree to make more accurate predictions. Each model has its own way of learning from the data, and by blending them together, we can often get better results. The voting is set to **soft**, which means the model looks at the probability each classifier gives for a certain class and then averages them to decide the final output. After setting up the model, it's trained using the x,y train data. Once trained, this ensemble can be used to make predictions and evaluate how well it performs compared to using just one model on its own.

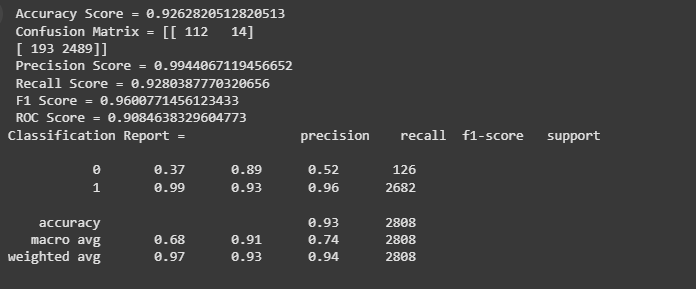
**Models performance measures :**

After each model implementation and evaluation these are the performance measure of each model :

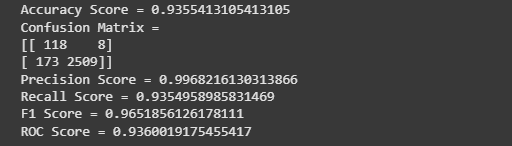
1.Decision Tree performance measures :



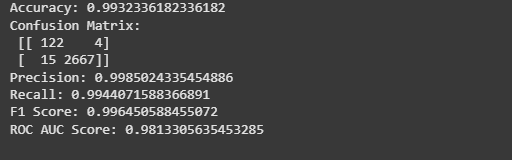
1. SGD performance measures :



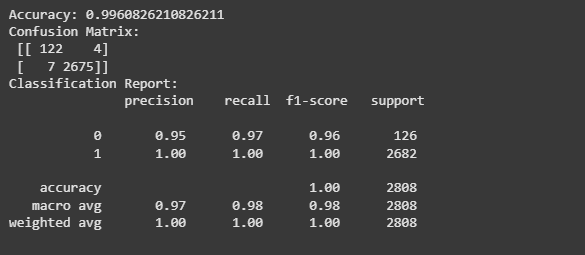
1. Logistic Regression performance measures :



1. KNN performance measures at k = 5 :



1. Voting performance measures :



From the provided statistics, it seems that the accuracy of the **voting ensemble model** is the highest of all the other models.

***Conclusion***

### 

### **Conclusion:**

In this project, we aimed to develop a **Network Intrusion Detection System** to classify network connections as **normal** or **anomalous**. After preprocessing the dataset, which included handling missing data, encoding categorical features, and scaling numerical values, we implemented multiple classification models: **Decision Tree**, **Logistic Regression**, **KNN**, **SGD**, and a **Soft Voting Classifier**.

The **Soft Voting Classifier** outperformed all other models, providing the best balance of **precision**, **recall**, and **F1-Score**, making it the most reliable for this task. While other models like **Decision Tree** and **KNN** showed decent performance, they faced issues like overfitting and computational inefficiency, respectively.

Overall, the project demonstrated that ensemble methods like **Soft Voting** can significantly improve model performance, making them a good choice for tasks such as network intrusion detection, where accuracy and reliability are crucial. Future improvements could focus on hyperparameter tuning and exploring more complex models for even better detection capabilities.