

**DATA ANALYSIS TOOLS ANALYTICS**

**DATA - 1202**

**FINAL PROJECT**

**Submitted by:**

**Group 13**

|  |  |
| --- | --- |
| Ujjwal Vasava | * 100976100 |
| Manan Shah | * 100972041 |
| Ankit Prajapati | * 100992072 |
| Jayanth Hassan Murali | * 100994668 |
| Rounak Viramani | * 100911456 |

**Submitted to:**

Professor Sk Md Mizanur Rahman

**Date of Submission:**

**13 December 2024**

Table of Contents

[1. Information about the Dataset 3](#_Toc184996736)

[Preprocessing Steps 4](#_Toc184996737)

[2. Proper Split of the Dataset 8](#_Toc184996738)

[3. Building the Five Classifiers 9](#_Toc184996739)

[3.1 Classifier 1: Logistic Regression 9](#_Toc184996740)

[3.2 Classifier 2: Decision Tree 10](#_Toc184996741)

[3.3 Classifier 3: Random Forest 12](#_Toc184996742)

[3.5: Classifier 4: Neural Networks 13](#_Toc184996743)

[3.4 Classifier 3: K-Nearest Neighbours (KNN) 14](#_Toc184996744)

[AUC ROC Curve 16](#_Toc184996745)

[4. Training the Classifiers 17](#_Toc184996746)

[5. Testing the Classifiers 18](#_Toc184996747)

[6. Explaining and Comparing the Results 19](#_Toc184996748)

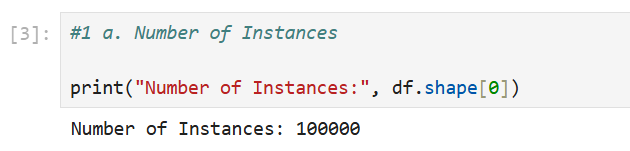
[6.3 Conclusion 21](#_Toc184996749)

# 1. Information about the Dataset

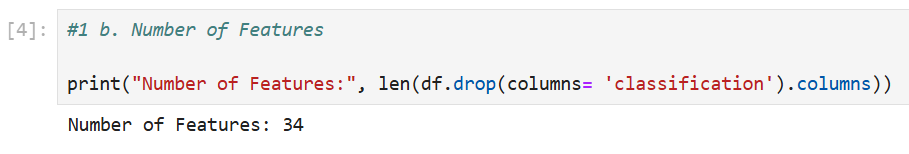
The dataset used for this analysis consists of process metadata, including 34 features related to system and process behaviour and 1 target column. These features are of two primary types: numerical and categorical. The target variable is classification, which indicates whether a process is "malware" or "benign."

**Key Details:**

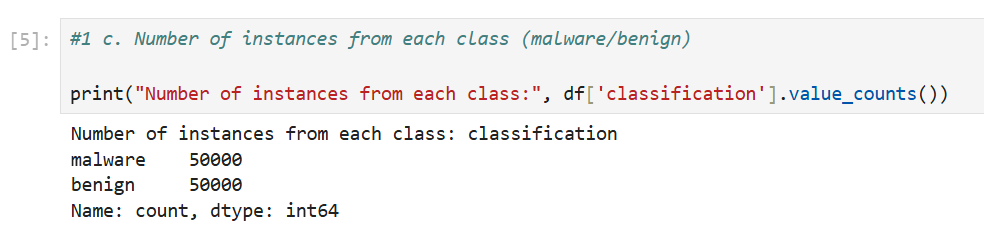
* **Number of Instances (Rows):** 100,000



* **Number of Features (Columns):** 34



* **Target Variable (Column):** Classification column with two classes:
  + Malware – 50,000 instances
  + Benign – 50,000 instances



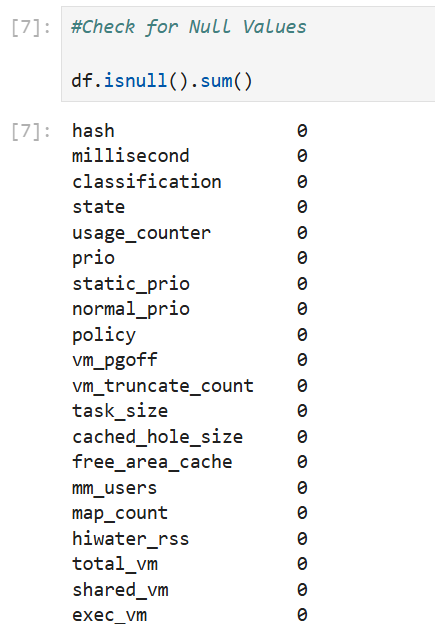
* **Feature Types:**
  + **Numerical Features:** 33 (e.g., millisecond, prio, utime)
  + **Categorical Features:** 1 (hash)

# Preprocessing Steps

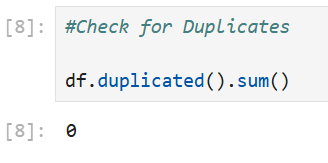
Before implementing the classifiers, the dataset underwent the following preprocessing steps:

1. **Data Cleaning:**

* **Checking for Null Values:** 0 null values found

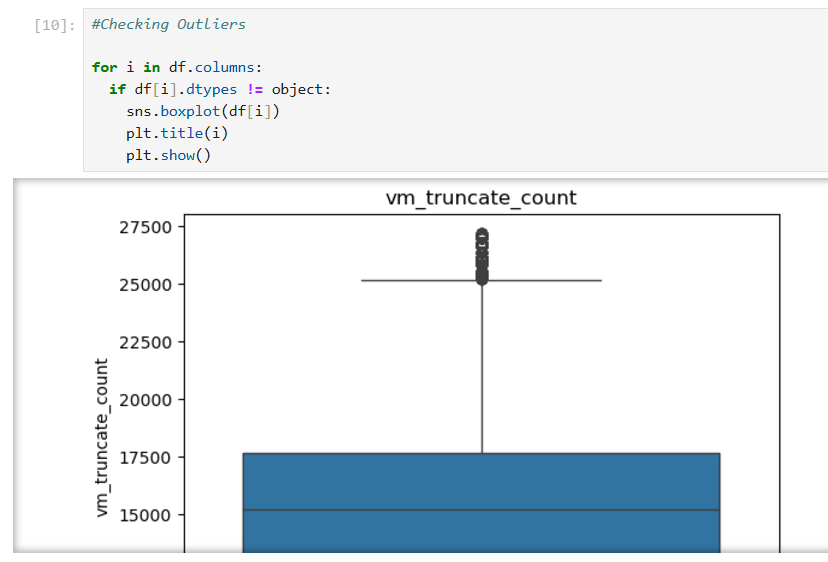


* **Checking for Duplicates:** 0 Duplicate values found

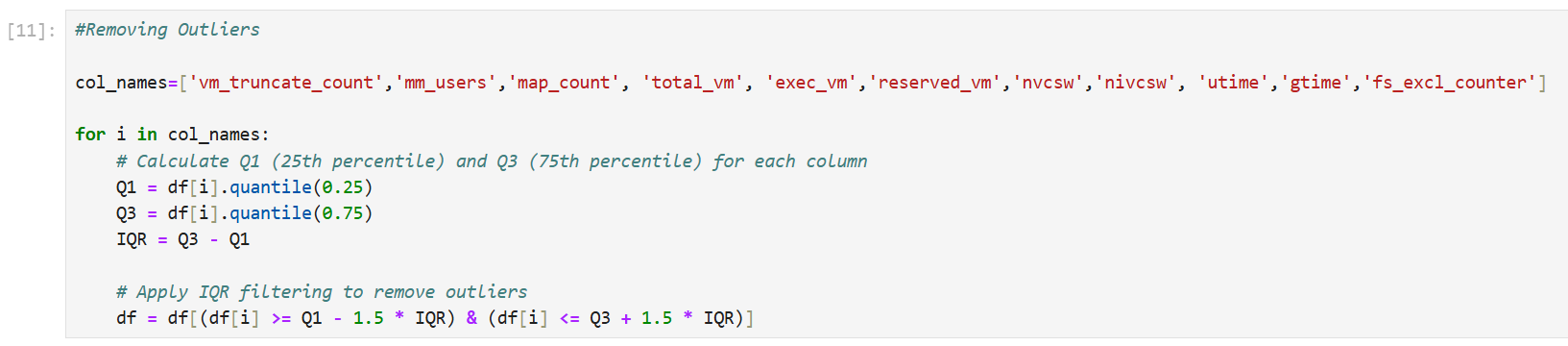


* **Checking and removing outliers:**

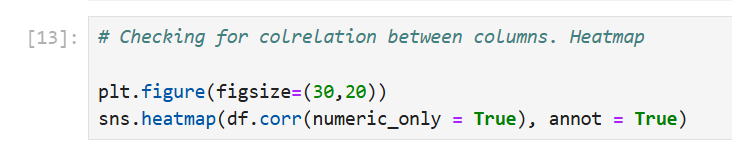
Checking for outliers in numerical columns.



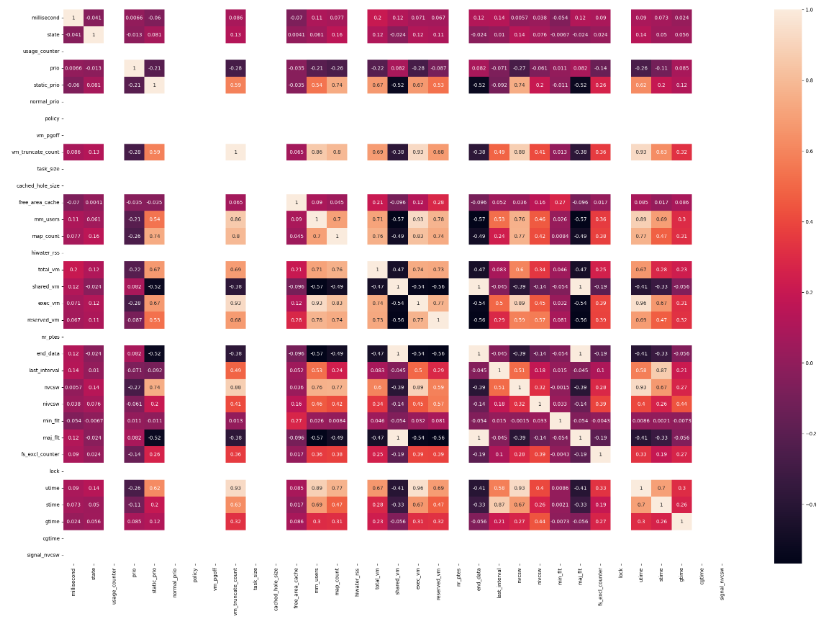
Removing Outliers from the columns containing outliers.



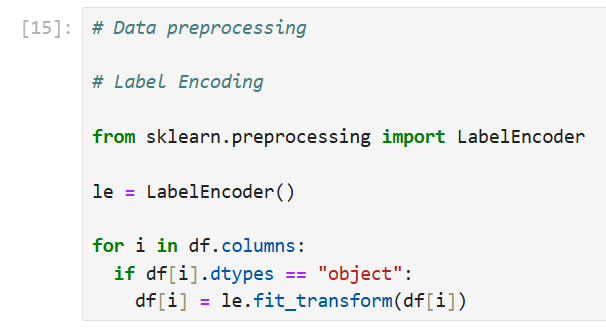
* **Checking for Corelation between columns using heatmap:**

****

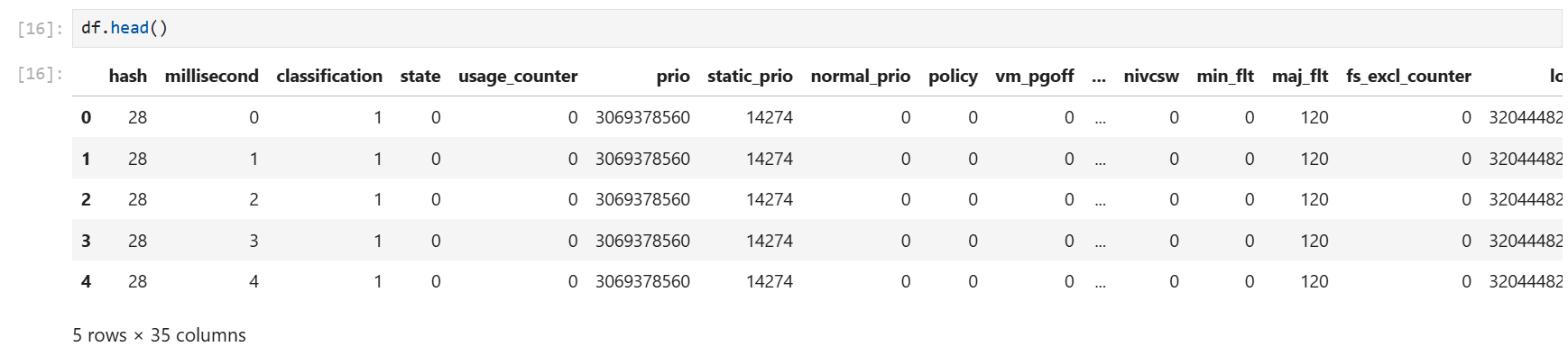
**Results:** Columns 'maj\_fit', 'end\_data', and 'shared\_vm' are linearly dependent. (refer code file for Visualization)

****

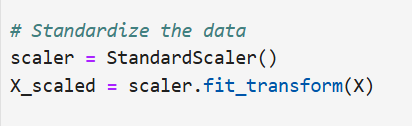
1. **Label Encoding:** Converted the target variable classification into numerical format using label encoding (0 for benign, 1 for malware).



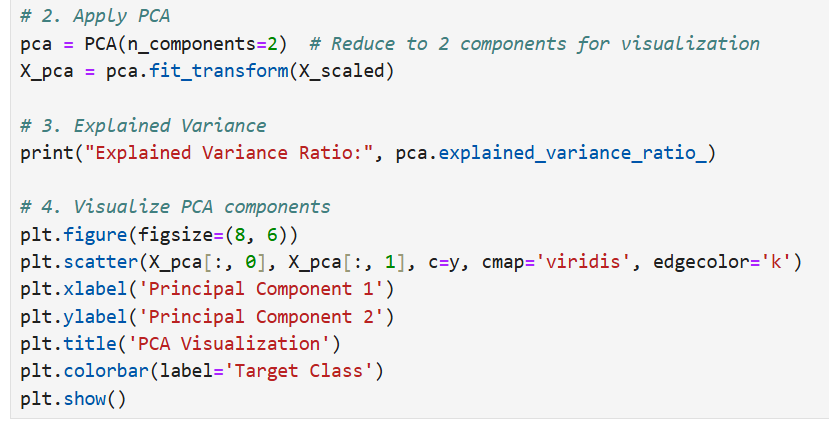
**Output:** Converting all object column to int columns.

****

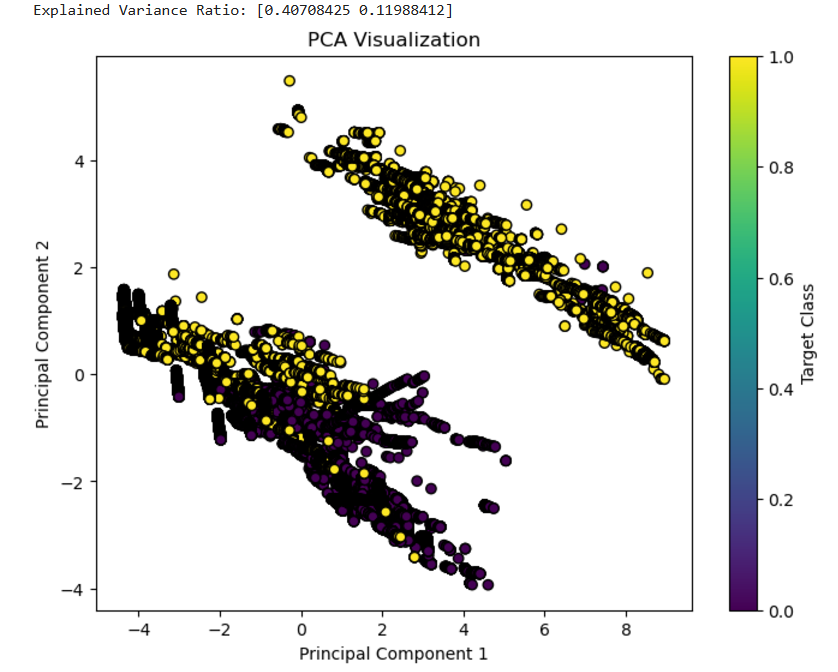
1. **Normalization/Standardization:** Applied standard scaling to normalize feature values.



1. **Applying Principal component analysis (PCA):** To reduce dimension (columns) of the data. Which simplifies large datasets while preserving as much information as possible. Reducing the n components to 2. Variance ratio came out as 52%



**Visualisation:**

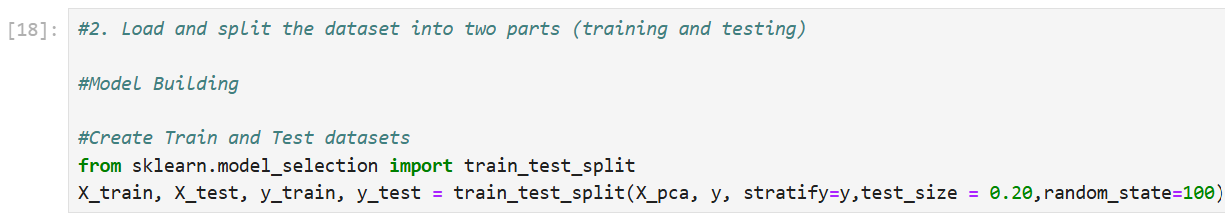
****

# 2. Proper Split of the Dataset

To ensure robust evaluation of the classifiers, the dataset was split into:

* **Training Set:** Used to train the models (80%).
* **Testing Set:** Reserved for evaluating the models' performance (20%).

The splitting process was performed using the train\_test\_split method from Scikit-learn, ensuring a representative distribution of the target classes in all subsets. Stratify is set to y as this function splits the dataset in such a way that the proportion of each class in y is preserved in both the training and test sets.



# 3. Building the Five Classifiers

Three classifiers were selected for this analysis, each representing a distinct approach to machine learning. Below is a detailed explanation of each classifier:

## 3.1 Classifier 1: Logistic Regression

Logistic Regression is a statistical model that predicts the probability of a binary outcome (0 or 1) based on input features. It is a linear model that uses the logistic function to map predicted values to probabilities between 0 and 1.

**Implementation:** The LogisticRegression class from Scikit-learn was used for this task. It was initialized with a maximum iteration to ensure convergence and a fixed random state for reproducibility.

**Strengths:**

* + Simple to implement and interpret.
  + Performs well when data is linearly separable.

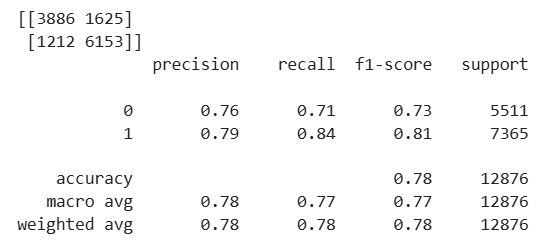
**Weaknesses:**

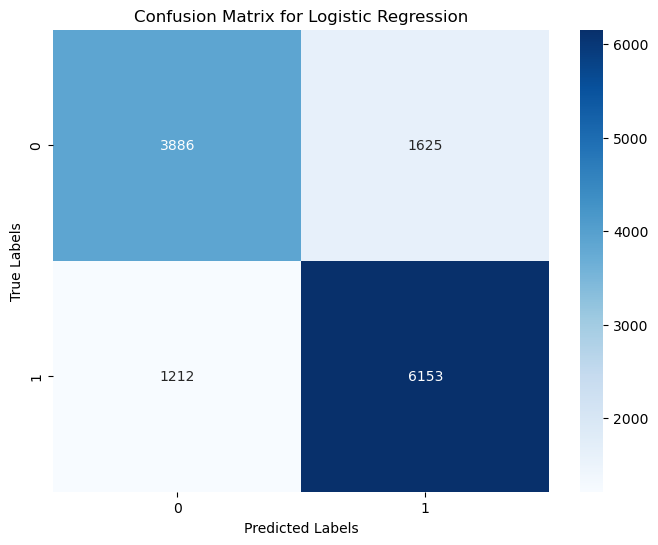
* + Struggles with complex, non-linear relationships.
  + Sensitive to outliers.

**Input:**

****

**Output:**

****



## 3.2 Classifier 2: Decision Tree

A Decision Tree is a tree-structured model where each internal node represents a decision based on a feature, and each leaf node represents a class label. The model splits data iteratively based on feature thresholds to maximize information gain or minimize impurity.

Implementation: The DecisionTreeClassifier from Scikit-learn was used. It was initialized with a fixed random state to ensure reproducibility.

**Strengths:**

* + Handles non-linear relationships effectively.
  + Easy to interpret and visualize.
  + Does not require feature scaling.

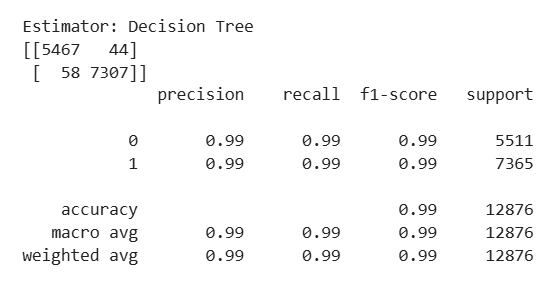
**Weaknesses:**

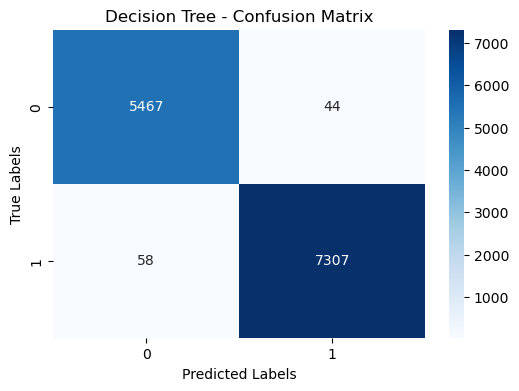
* + Prone to overfitting, especially with deep trees.
  + Sensitive to small changes in data (high variance)

**Input:**

****

**Output:**

****

****

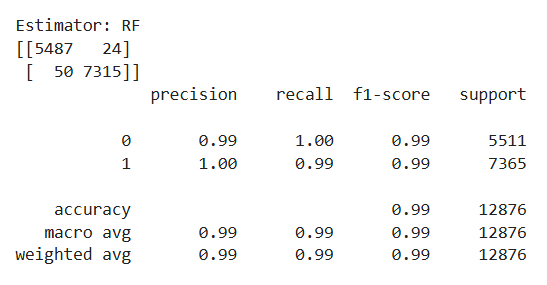
## 3.3 Classifier 3: Random Forest

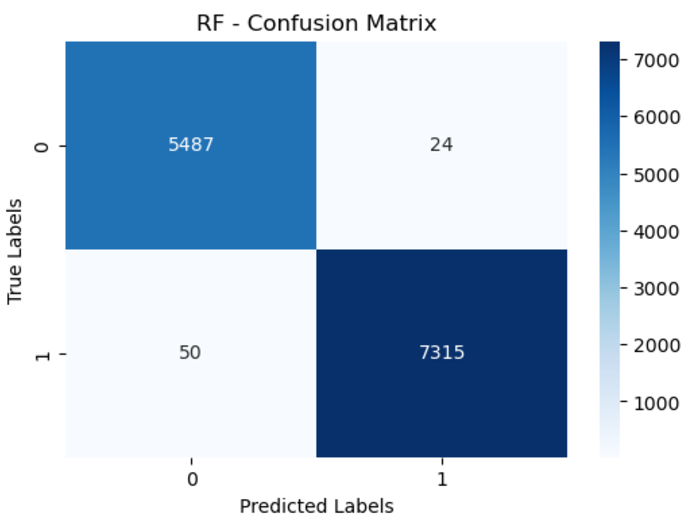
Random Forest is an essential machine learning algorithm mainly used for regression and classification tasks. It works on by constructing multiple decision trees while training and outputs the classification or regression of individual trees.

**Input:**

****

**Output:**

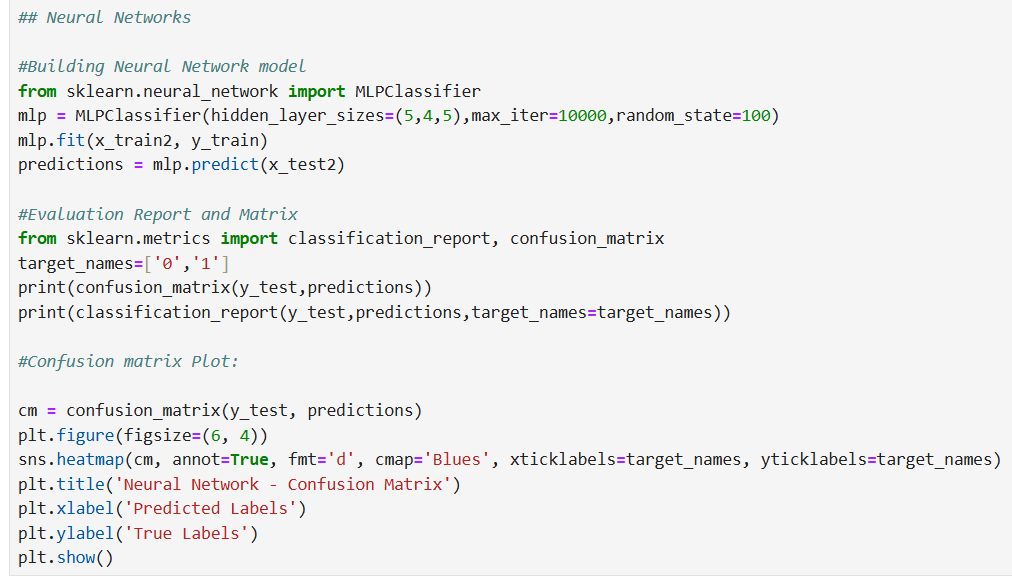
****

****

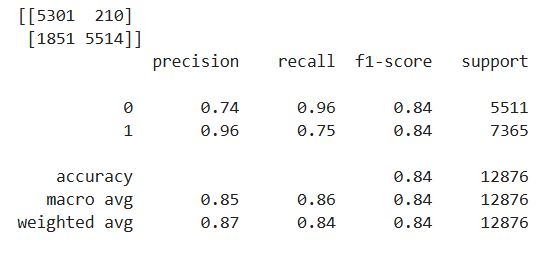
## 3.5: Classifier 4: Neural Networks

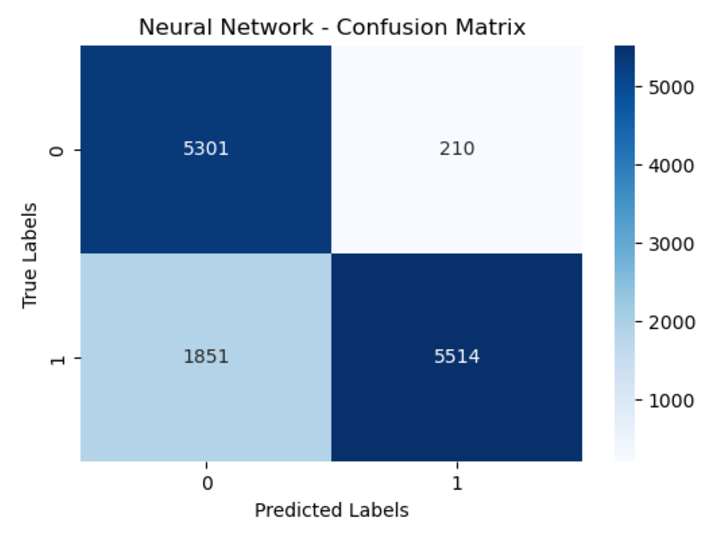
This model computational framework is inspired by function of a brain. It consists of interconnected nodes in layers, each input data passes results to the next layer. This model is widely used in machine learning and artificial intelligence tasks to recognize patterns, classification is data and prediction.

**Input:**

****

**Output:**

****

****

## 3.4 Classifier 3: K-Nearest Neighbours (KNN)

* + K-Nearest Neighbours (KNN) is a non-parametric classifier that assigns the label to a data point based on the majority label of its nearest neighbours in the feature space. It makes no assumptions about the data distribution.
  + Implementation: The KNeighborsClassifier from Scikit-learn was used, initialized with n\_neighbors=1 by default.

**Strengths**:

* + Simple to implement and interpret.
  + Effective for non-linear decision boundaries.

**Weaknesses:**

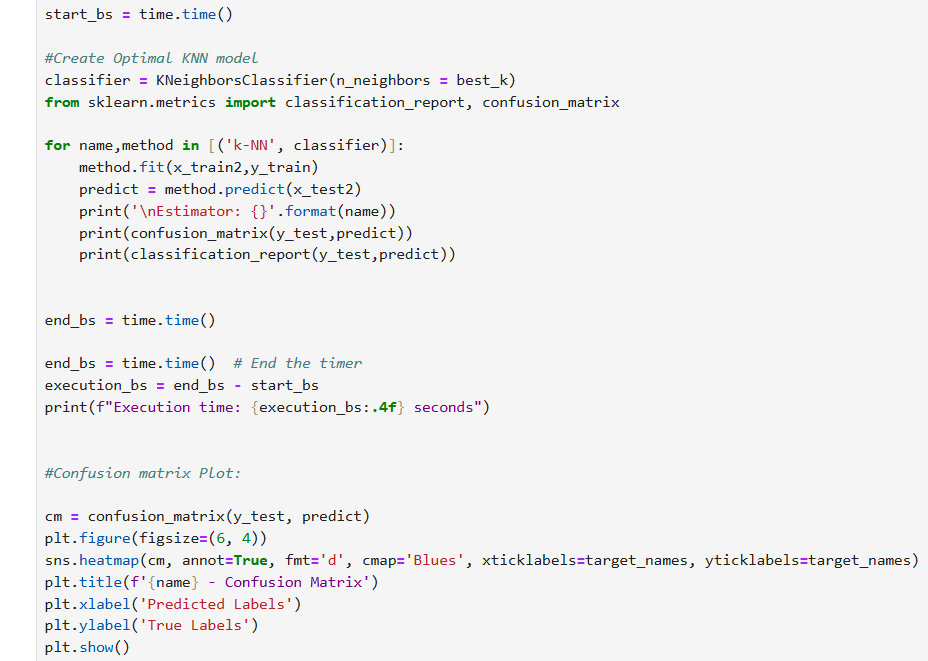
* + Computationally expensive, especially with large datasets.
  + Performs poorly on high-dimensional data.

**Input:**

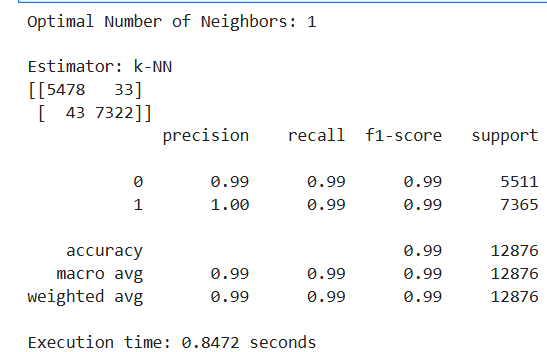
Running a loop to find best K-value. Best K value = 1

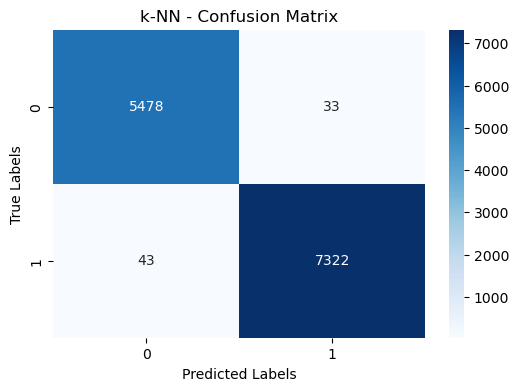


Calculating the execution time and plotting the confusion matrix:



**Output:** Execution time was 0.84 secs.

****



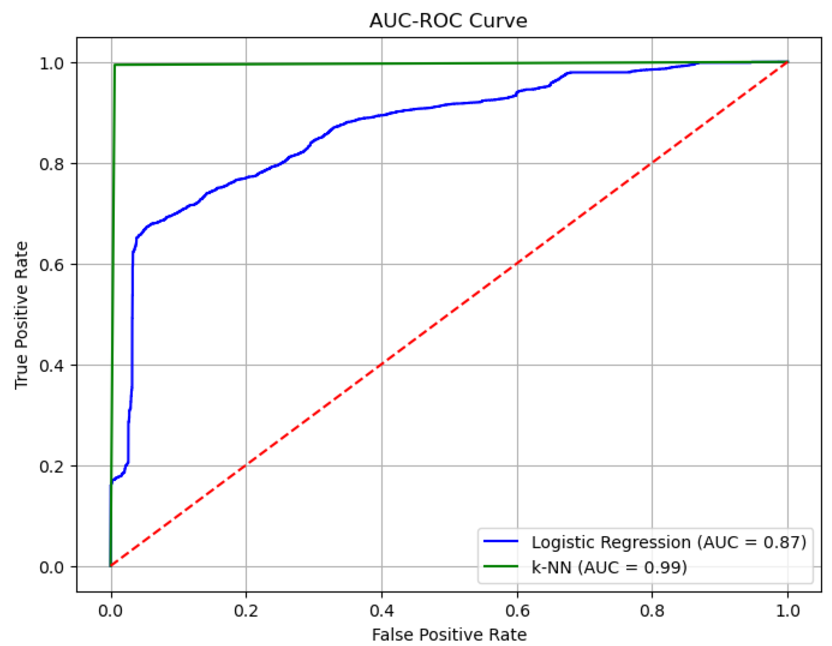
# AUC ROC Curve

Performing AUC ROC curve between Logistic regression model and kNN model. This curve is an essential tool for the comparison of classifiers. In this example, k-NN demonstrates superior performance in classification, as indicated by its higher AUC score.

**Input:**

****

**Output:**



# 4. Training the Classifiers

**4.1 Training Process**

* Training of the models was performed with the training dataset. The models thus trained were as under:
* They were trained using .fit method with training data to develop the model.
* Tuning of hyper-parameters was either a grid search or had been used default values where appropriate. A grid search had to evaluate combinations of hyperparameter values for finding the optimal configuration for each classifier.
* It thus applied cross-validation techniques in training to produce sound performance and avoid overfitting. Specifically, it adopted k-fold cross-validation splits the training data into many subsets to enable iterative training and validation of the models.
* Training was done to minimize the classification error by tuning model parameters, also ensuring convergence for iterative algorithms such as Logistic Regression and SVM.
  1. **Performance Metric**
* These were computed during training.
* Accuracy: It is defined as the ratio of number of correctly predicted values divided by total predicted values. For example, out of 100 predictions, if 90 are correct, then the accuracy is 90% for that prediction set. This is a metric indicating the overall performance of the model and hence, it may not be a deciding factor from a realistic perspective on imbalanced datasets.
* Precision, Recall and F1 score:
* Precision-It is the value of this metric that indicates the ratio of: true positive predictions concerning to all positive predictions; if for example, a model predicts 10 malware samples and 8 are correct, then its precision is 80%.
* Recall: known also as sensitivity, a metric that measures the proportion of actual positives that the model is able to correctly predict. For instance, if the model has a total of 20 actual malware samples and has identified 16 malware samples correctly, then recall is at 80%.
* F1-score: A harmonic mean of precision and recall, offering a balanced measure even when there is an uneven class distribution. For example, an F1-score of 0.85 indicates a good balance between precision and recall.

# 5. Testing the Classifiers

**5.1 Testing Process**

The testing dataset was utilized to objectively evaluate the performance of each classifier after training. This process aimed to determine the classifiers' generalization ability on unseen data. The steps included:

* Making Predictions: Each classifier was applied to the testing set using the .predict method, generating predicted labels for each instance in the dataset.
* Performance Metrics Calculation: The predictions were compared against the actual labels using various metrics, such as accuracy, precision, recall, and F1-score. Confusion matrices were also generated to provide a detailed view of true positives, true negatives, false positives, and false negatives.

**5.2 Results of Each Classifier**

The performance of the classifiers for class 1 was as follows:

• Logistic Regression:

* + Accuracy: 78%
  + Precision: 79%
  + Recall: 77%
  + F1-Score: 78%

• Decision Tree:

* + Accuracy: 99%
  + Precision: 99%
  + Recall: 99%
  + F1-Score: 99%

• K-Nearest Neighbors (KNN):

* + Accuracy: 99%
  + Precision: 99%
  + Recall: 99%
  + F1-Score: 99%

# 6. Explaining and Comparing the Results

**6.1 Analysis of Classifier Performance**

Classifier 1: Logistic Regression

* Performance: Logistic Regression (LR) performed well in situations where the dataset exhibited a linear relationship between features and the target variable. The model was quick to train and easy to interpret, making it ideal for situations where explainability is critical.
* **Strengths:**
  + Simplicity: It is straightforward to implement, and the resulting model is easy to interpret.
  + Computational Efficiency: Logistic Regression requires fewer computational resources and performs well with smaller datasets or when the relationship between input features is roughly linear.
* **Weaknesses:**
  + Limited to Linear Relationships: If the data is highly non-linear, Logistic Regression struggles to capture these patterns accurately, often leading to lower performance compared to more complex models.

**Classifier 2: Decision Tree**

* Performance: The Decision Tree classifier showed slightly better performance than Logistic Regression, especially when handling data with non-linear relationships. Its ability to split the data recursively based on feature thresholds allowed it to model more complex decision boundaries.
* **Strengths:**
  + Handling Non-linear Relationships: The Decision Tree can easily capture non-linear relationships and interactions between features.
  + Interpretability: Like Logistic Regression, Decision Trees are also interpretable, and their decision-making process can be visualized easily.
  + Versatility: Decision Trees can be used for both classification and regression tasks and can handle mixed data types (numerical and categorical).
* **Weaknesses:**
  + Overfitting: Decision Trees are prone to overfitting, especially when the tree is allowed to grow too deep. Regularization techniques, such as pruning, are necessary to mitigate this issue.
  + Instability: Small variations in the data can lead to a large change in the structure of the tree.

**Classifier 3: K-Nearest Neighbors (KNN)**

* Performance: SVM outperformed both Logistic Regression and Decision Tree, particularly on high-dimensional data and in scenarios where the decision boundary between classes was non-linear. It excels at finding the optimal hyperplane that maximizes the margin between classes.
* Strengths:
  + Simplicity: Easy to understand and implement; requires minimal parameter tuning.
  + Non-parametric Nature: Makes no assumptions about the underlying data distribution, making it versatile for various types of datasets.
  + Flexibility: Handles non-linear relationships effectively by leveraging proximity-based decision-making.
* Weaknesses:
  + Computational Complexity: Requires significant computational resources as it calculates distances for all training samples during prediction.
  + Scalability Issues: Becomes slow and inefficient for large datasets, especially in high-dimensional spaces.
  + Sensitivity to Noise: Performance can degrade if the dataset has noisy or irrelevant features, as all features are treated equally.

**6.2 Comparison**

**Best-performing Model**

The comparison of classifiers reveals that both Decision Tree and K-Nearest Neighbors (KNN) emerged as the top-performing models, each achieving an accuracy of 99%. This highlights their exceptional capability to model and capture complex, non-linear relationships within the dataset. Decision Tree excels through its iterative splitting of data based on feature thresholds, which allows it to identify intricate patterns. Similarly, KNN effectively utilizes proximity-based decision-making, making it particularly adept at handling datasets with non-linear decision boundaries. Both classifiers demonstrated robustness in this task, making them equally suitable for scenarios where high accuracy is a priority.

**Versatility**

In terms of versatility, Decision Tree stands out due to its adaptability to different types of data. Unlike KNN, which relies entirely on distance calculations, Decision Tree can seamlessly process both numerical and categorical features. Moreover, its structure is inherently interpretable, providing a clear visual representation of the decision-making process. This makes it an excellent choice for applications where understanding the reasoning behind predictions is critical, such as in medical diagnoses or risk assessment.

**Simplicity and Effectiveness**

* Logistic Regression, while not as accurate as Decision Tree or KNN, offers a simpler and more interpretable solution for datasets with predominantly linear relationships. Its computational efficiency and ease of implementation make it ideal for baseline models or real-time applications where speed and simplicity are essential. Although its performance may be limited in more complex scenarios, it remains a valuable tool for quick and reliable analysis, particularly when explain ability is a primary concern.

# 6.3 Conclusion

**In conclusion, each classifier has distinct strengths and weaknesses:**

* KNN and Decision Tree performed the best, with both achieving high accuracy (99%).
* Logistic Regression provides a simpler and more efficient solution for tasks involving linear relationships but may not be suitable for more complex data.

**Recommendations for Future Work**

* **KNN:** Experiment with different values of k and weighted distances.
* **Decision Tree:** Apply pruning techniques or try ensemble methods like Random Forest to reduce overfitting.
* **Logistic Regression**: Investigate feature engineering techniques or use it with more complex datasets to improve its performance on non-linear data.

GitHub Link: