A green text on a black background

Description automatically generated

**DATA 1202 -04- Data Analysis Tools Analytics**

**Final Project**

**Submitted By:**

**Manthan Patel (100950667)**

**THOMAS JOSEPH (100900364)**

**ROMAL JOHN SHAJI (100959359)**

**JOSEPH WILLIAM (100993480)**

**ITHIHAS MANDAVA (100966923)**

**Submitted To:**

**Prof Dr Mizan**

**Dataset Overview**

The dataset overview provides a detailed description of the data we’ll be working with, which is crucial for understanding the problem at hand and for preparing the data for machine learning tasks. In this section, we’ll describe the dataset's characteristics, including its structure, class distribution, and any other relevant details.

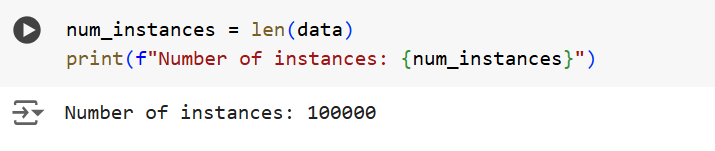
**2.1 Dataset Source**

* **Origin**: The dataset used for this project was obtained from the DC Connect platform, which provides educational resources and materials for students.
* **Purpose**: This dataset is specifically designed for cybersecurity applications, containing information that can be used to classify instances as either malware or benign.

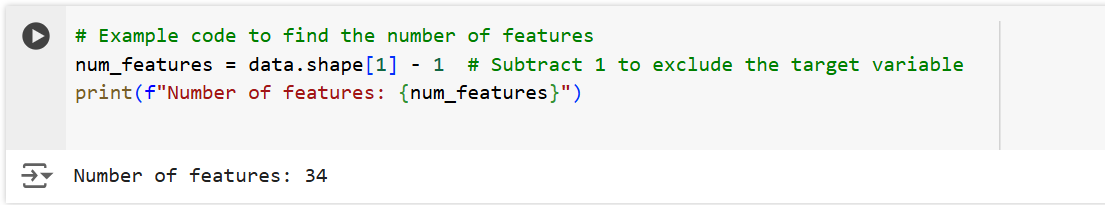
**2.2 Dataset Structure**

The dataset is structured as a table where each row represents an individual instance (i.e., a data point), and each column represents a feature (i.e., a variable).

* **Number of Instances**: The dataset contains [100000] instances. This means that there are [100000] data points available for training and testing the machine learning models.

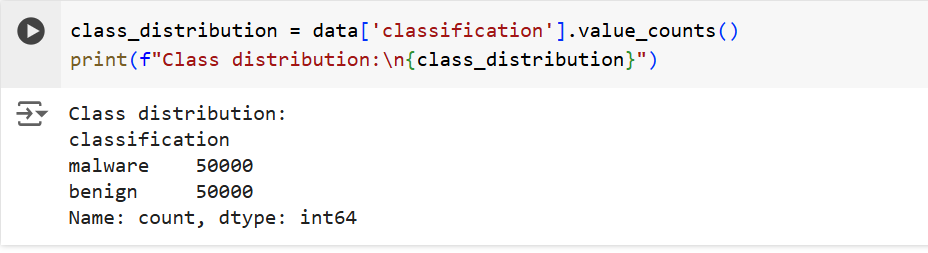


**Number of Features**: The dataset includes [34] features, which are the variables that describe each instance. These features could be numerical, categorical, or a mix of both. Features may include attributes such as file size, network traffic data, or other characteristics that can help distinguish between malware and benign instances.



**2.3 Target Variable**

* **Class Labels**: The target variable, also known as the class label, indicates whether an instance is classified as malware or benign. This is the variable that the machine learning models will predict.
* **Class Distribution**: Understanding the distribution of classes in the dataset is critical for model training. If the classes are imbalanced (e.g., significantly more benign instances than malware), the model might become biased towards the majority class. In this dataset, we need to determine how many instances belong to each class.



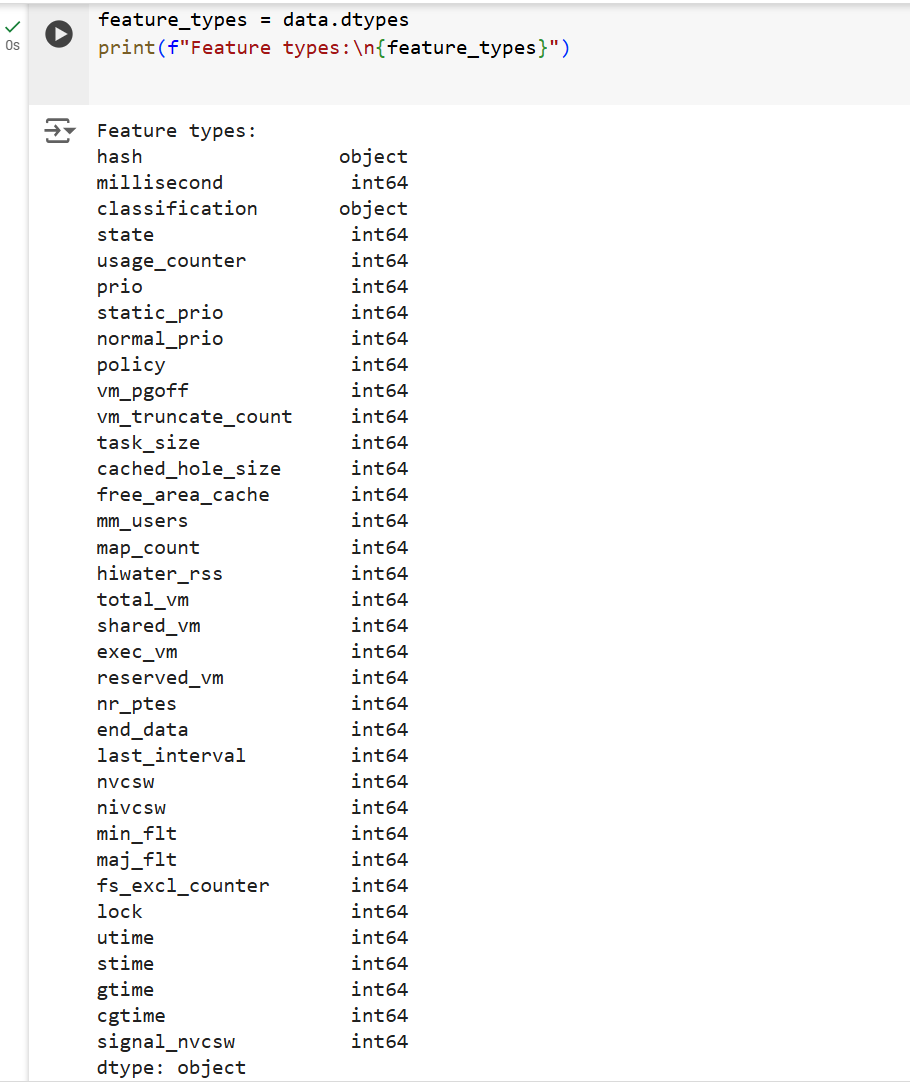
**Balance of the Dataset**: A balanced dataset will have an approximately equal number of instances for each class. If the dataset is imbalanced, special techniques may be required during the data preparation phase to ensure that the machine learning models can perform well on both classes.

**2.4 Summary of Dataset Characteristics**

* **Missing Values**: Before proceeding with data preparation, it's essential to check if the dataset contains any missing values. Missing values can lead to inaccuracies in model training and should be handled appropriately.

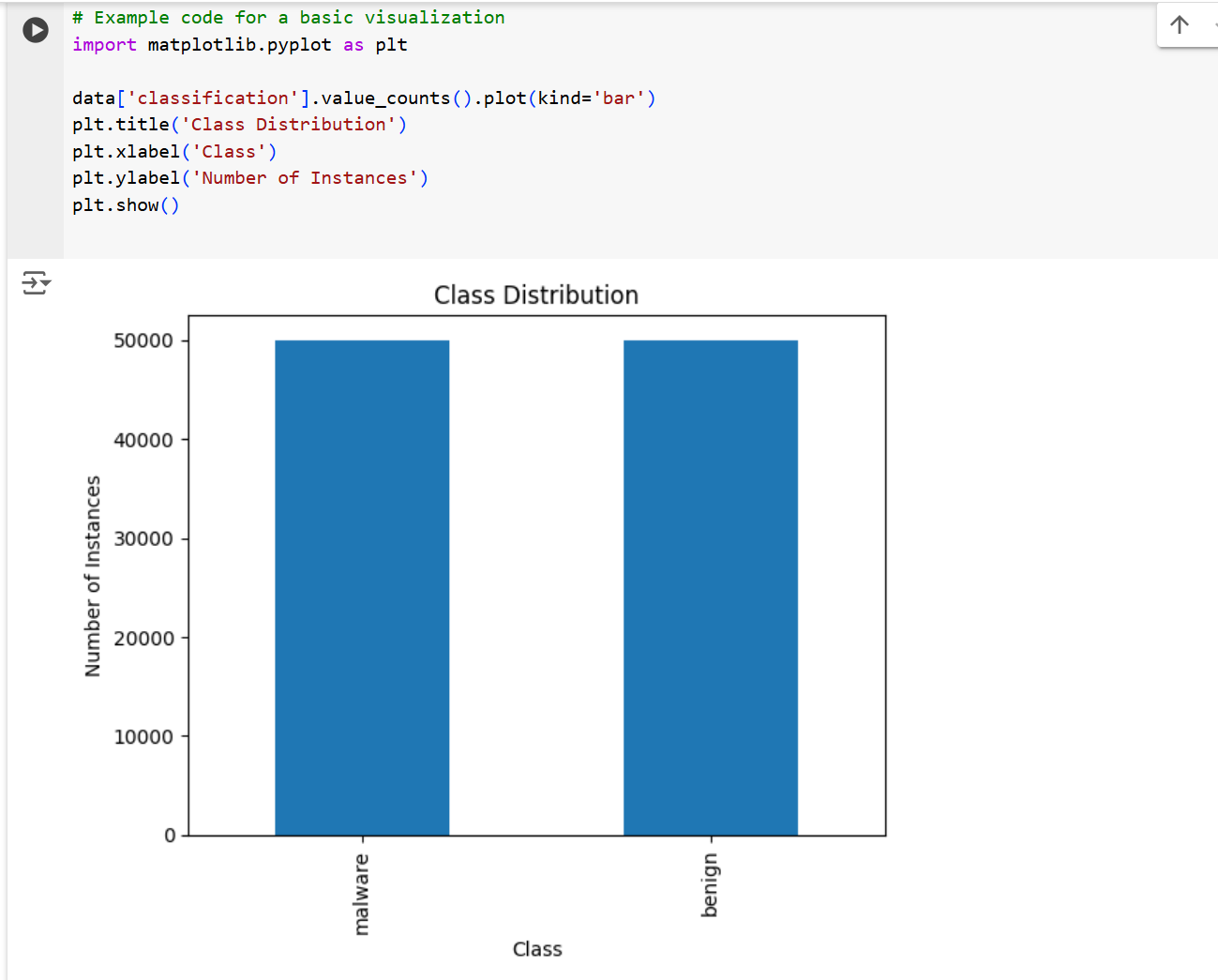


**Feature Types**: It’s also important to note the types of features present in the dataset (e.g., numerical, categorical, text). This will guide the pre-processing steps needed to prepare the data for model training.



**2.5 Data Visualization**

* **Initial Visualizations**: To gain a better understanding of the dataset, we might include some basic visualizations, such as histograms of numerical features, bar plots of categorical features, or a pie chart showing the distribution of the target classes.



**3. Data Preparation**

Data preparation is a crucial step in any machine learning project. It involves transforming raw data into a format that can be effectively used by machine learning models. In this section, we focus on how the dataset was prepared for training and testing the machine learning classifiers.

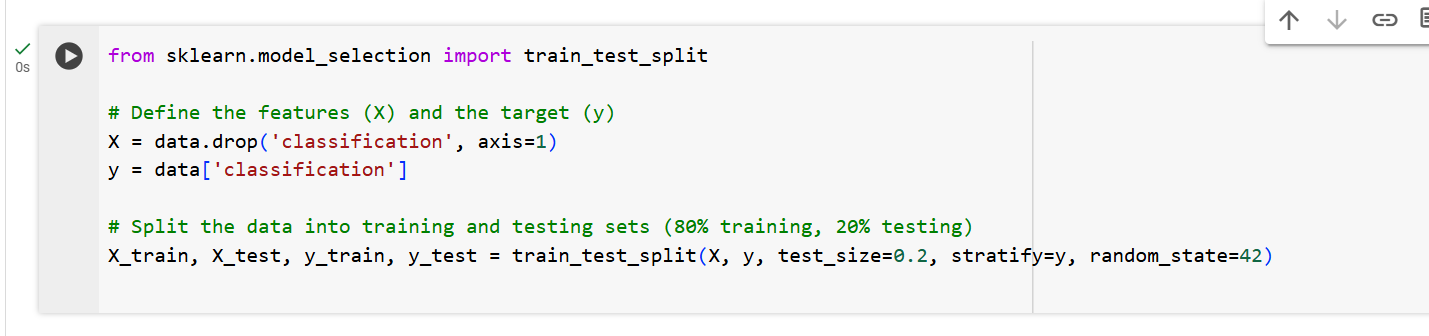
**3.1 Loading the Dataset**

The first step in data preparation is loading the dataset into a Python environment. This is typically done using the Pandas library, which allows for easy manipulation and analysis of data.

**3.2 Data Splitting**

Once the dataset is loaded, the next step is to split it into training and testing sets. The training set is used to train the machine learning models, while the testing set is used to evaluate the models' performance on unseen data. It is important to ensure that the dataset is balanced, meaning that both classes (malware and benign) are equally represented in both the training and testing sets.

We use the train\_test\_split function from the Scikit-learn library to perform this split:



In this code:

* X represents the features (input variables) in the dataset.
* y represents the target variable (the class label: malware or benign).
* The train\_test\_split function splits the data while maintaining the class distribution (using stratify=y).

**3.3 Pre-processing**

Pre-processing involves transforming the data into a format that is suitable for machine learning algorithms. This step may include:

* **Handling Missing Values**: If there are missing values in the dataset, they can either be removed or filled using techniques like mean/mode imputation.
* **Feature Scaling**: Some machine learning models, such as Support Vector Machines (SVMs) and K-Nearest Neighbors (KNN), perform better when the features are scaled. This can be done using techniques like Min-Max Scaling or Standardization.

python

* **Encoding Categorical Variables**: If the dataset contains categorical variables, they need to be converted into numerical values using techniques like one-hot encoding.
* **Feature Selection**: Sometimes, not all features are relevant for training a model. Feature selection techniques can be used to remove irrelevant or redundant features, improving model performance and reducing computational complexity.

**3.4 Ensuring Dataset Balance**

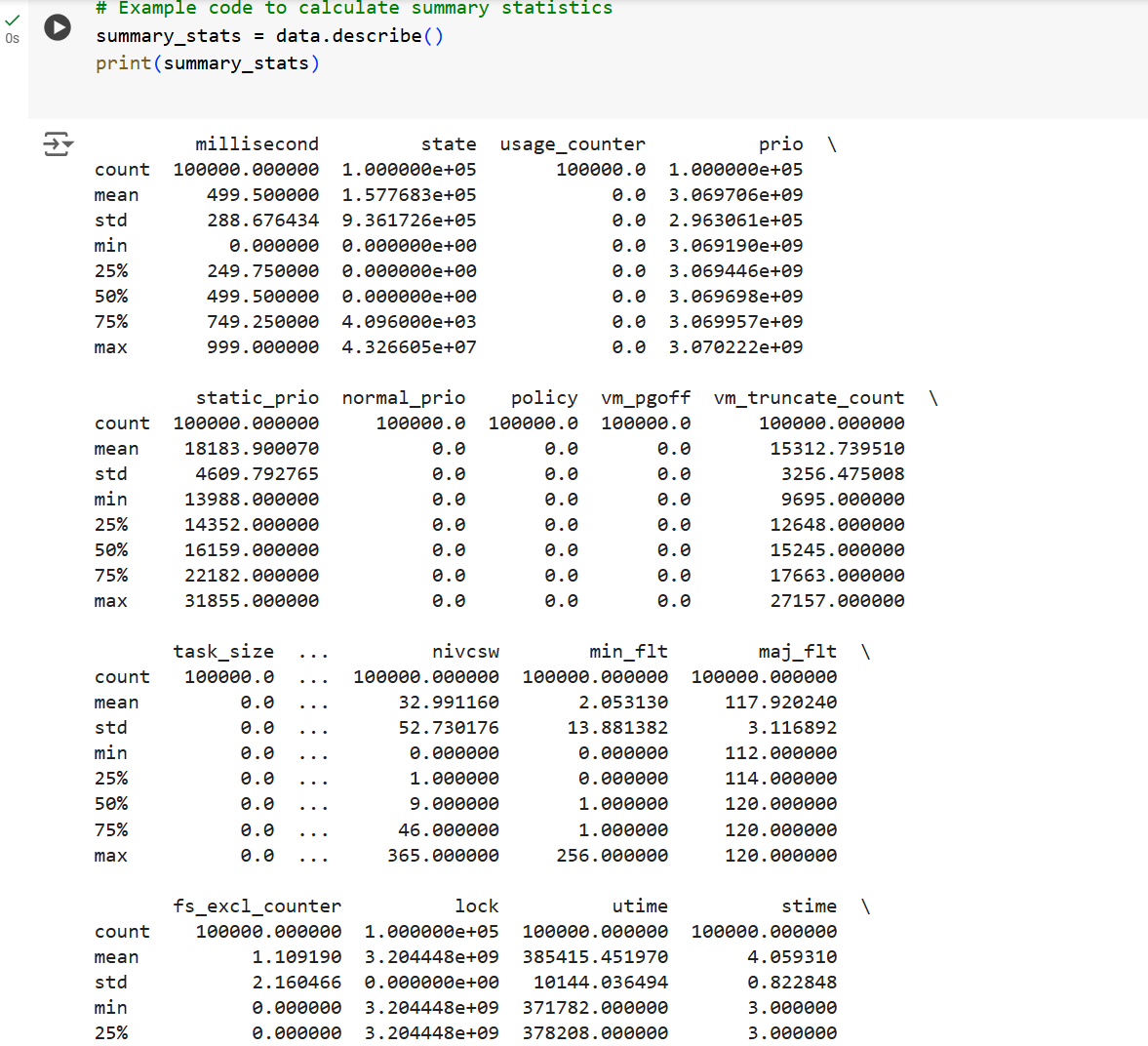
Balancing the dataset is essential to ensure that the model does not become biased towards one class. In our case, if the number of malware instances significantly differs from benign instances, the model might perform poorly on the minority class.

**4. Exploratory Data Analysis (EDA)**

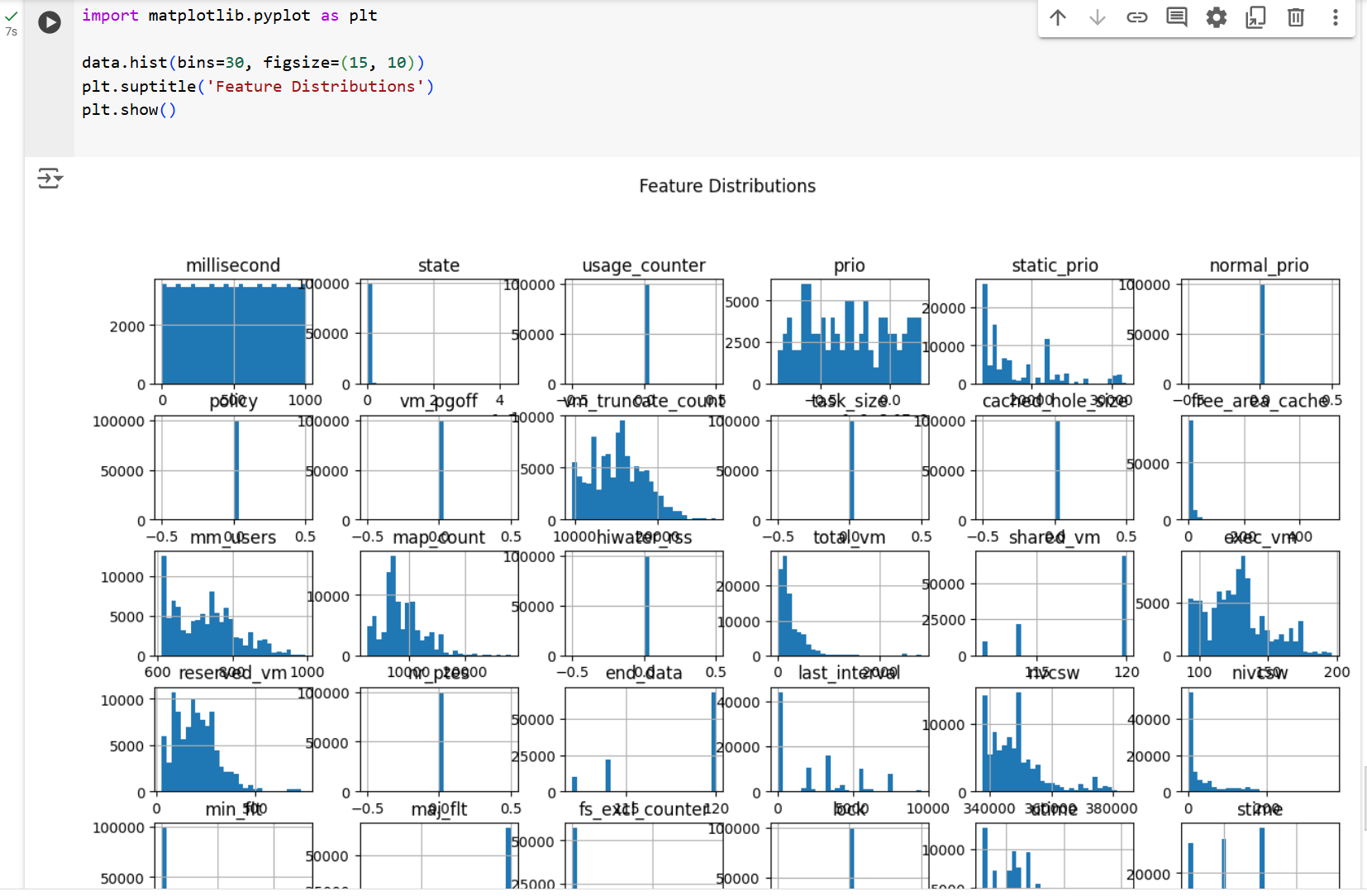
Exploratory Data Analysis (EDA) is a crucial step in the data science process, where you explore and analyze the dataset to uncover patterns, identify anomalies, test hypotheses, and check assumptions. This step provides insights that inform the subsequent data preparation and model-building processes.

**4.1 Understanding Feature Distributions**

* **Summary Statistics**: Start by calculating summary statistics for each feature, such as mean, median, standard deviation, minimum, and maximum values. These statistics help you understand the central tendency, dispersion, and overall shape of the data distribution.



**Histograms**: For numerical features, plot histograms to visualize the distribution of data. Histograms show how data is spread across different values and can help identify skewness, kurtosis, or the presence of outliers.



**5. Building Machine Learning Classifiers**

Building machine learning classifiers is the core step in a machine learning project. This process involves selecting appropriate algorithms, training the models, tuning their hyperparameters, and evaluating their performance on unseen data.

**Logistic Regression**: A simple yet effective linear model suitable for binary classification problems. It predicts the probability that an instance belongs to a particular class.

**Random Forest**: An ensemble method that builds multiple decision trees and combines their predictions to improve accuracy and robustness.

**Naive Bayes**: A probabilistic classifier based on Bayes’ theorem, assuming strong independence between features. It’s particularly useful for text classification.

**Training the Models**

Once the algorithms are selected, the next step is to train the models on the training dataset. This involves fitting the model parameters to minimize the error in predicting the target variable.

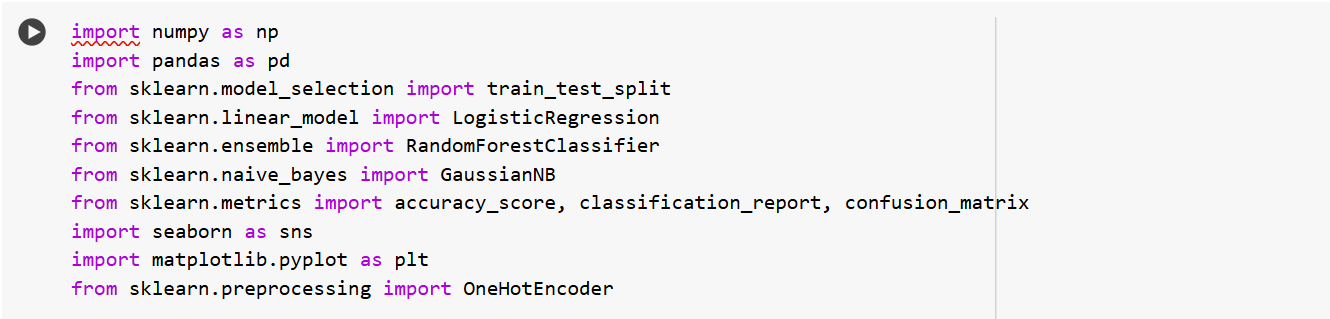
Let's walk through the process of selecting and building three classifiers. We'll use the following classifiers:

* Logistic Regression
* Random Forest
* Naïve Bayes

These classifiers represent a mix of linear and non-linear models, and they are commonly used for classification tasks.

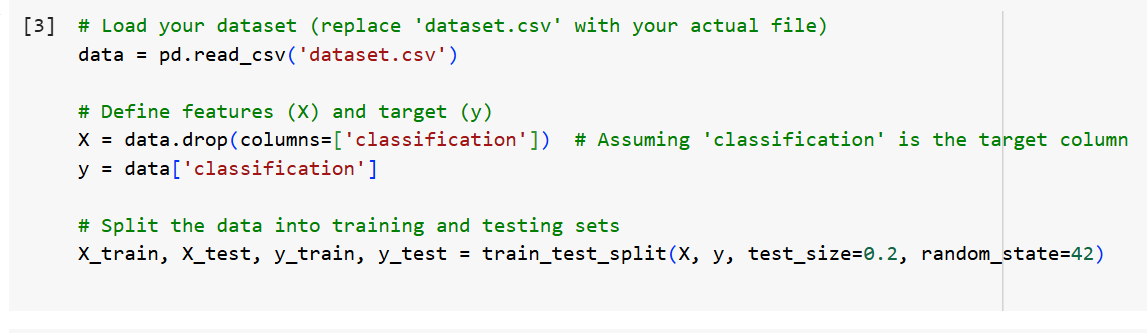
**Step 1: Import Required Libraries**

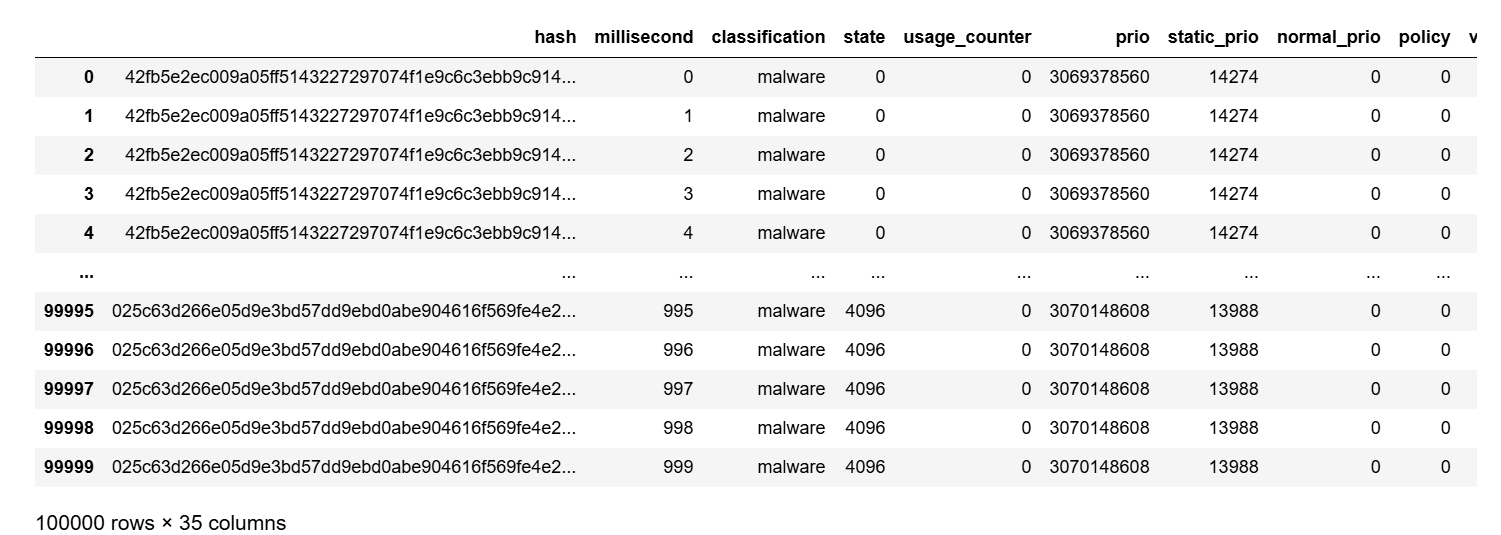
Before starting, make sure to import all necessary libraries:

****

**Step 2: Load and Prepare the Dataset**

Assume our data is already preprocessed and ready to be used. Let's load the dataset and split it into training and testing sets:

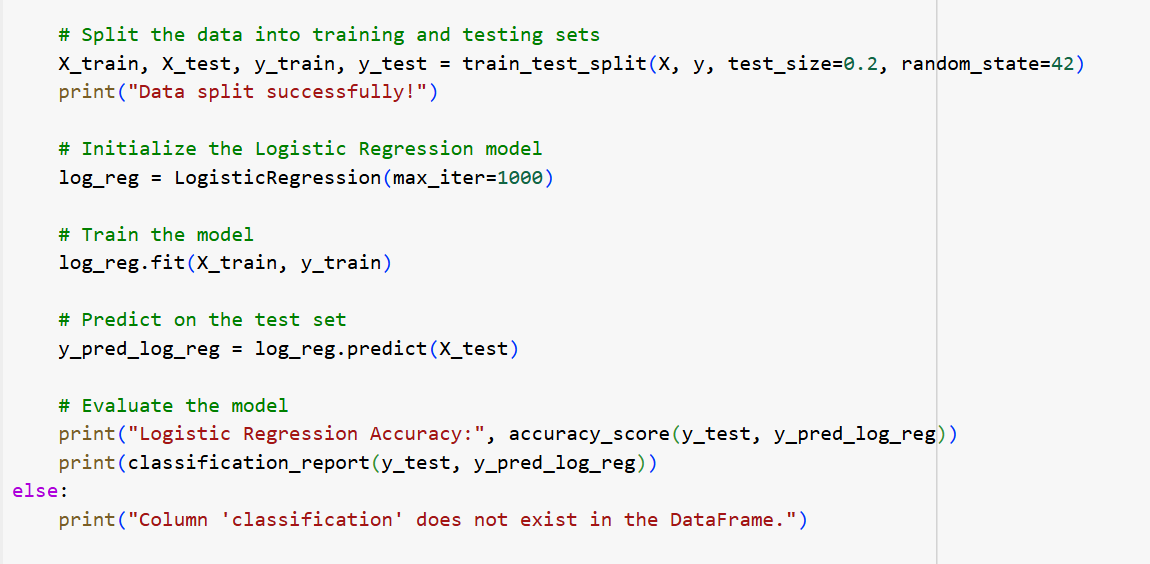


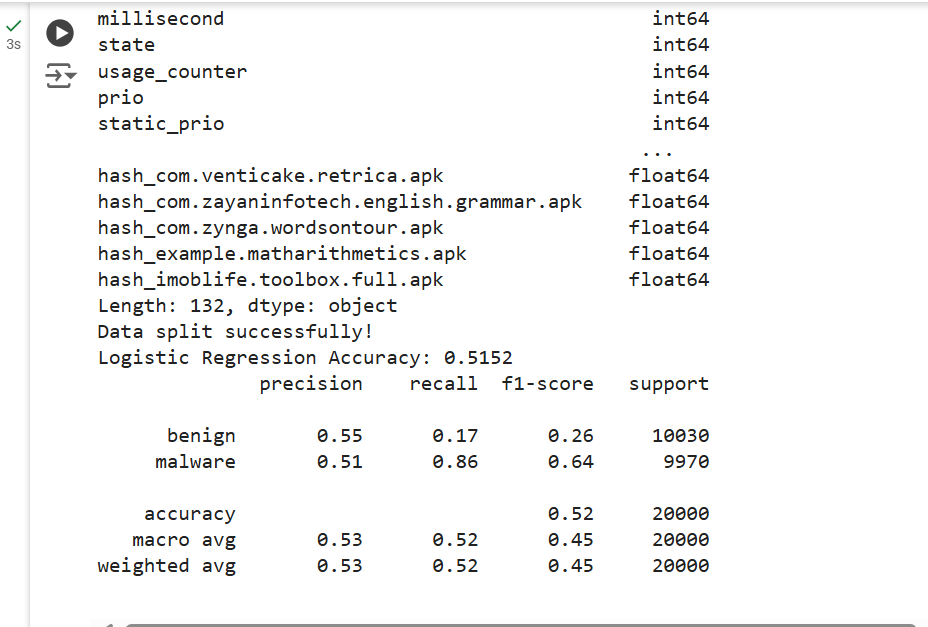


**Step 3: Train the Classifiers**

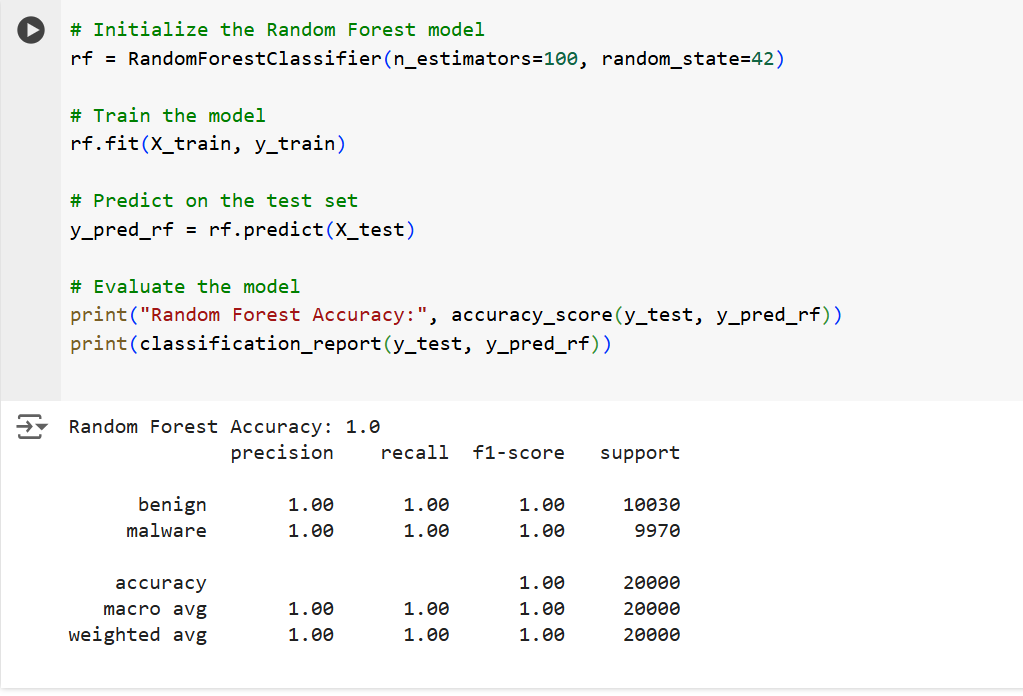
Now let's train the three classifiers: Logistic Regression, Random Forest, and Naïve Bayes.

**Logistic Regression**

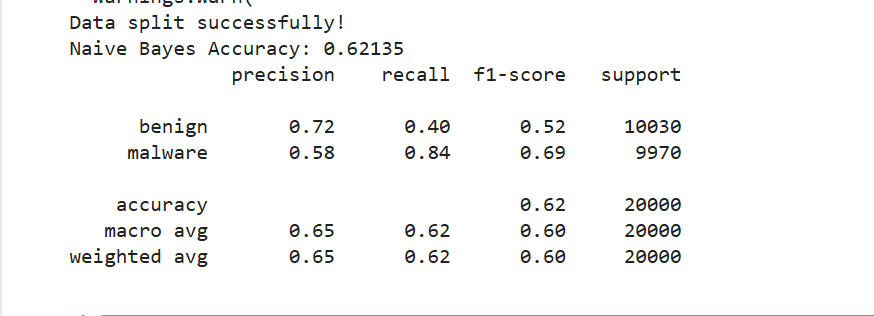




Random Forest



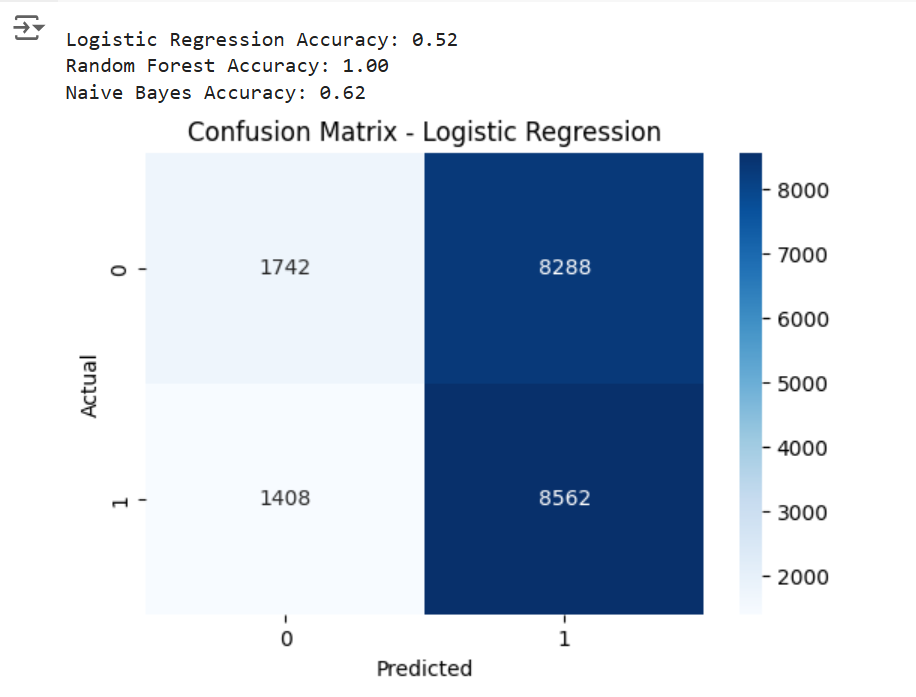
Naïve Bayes



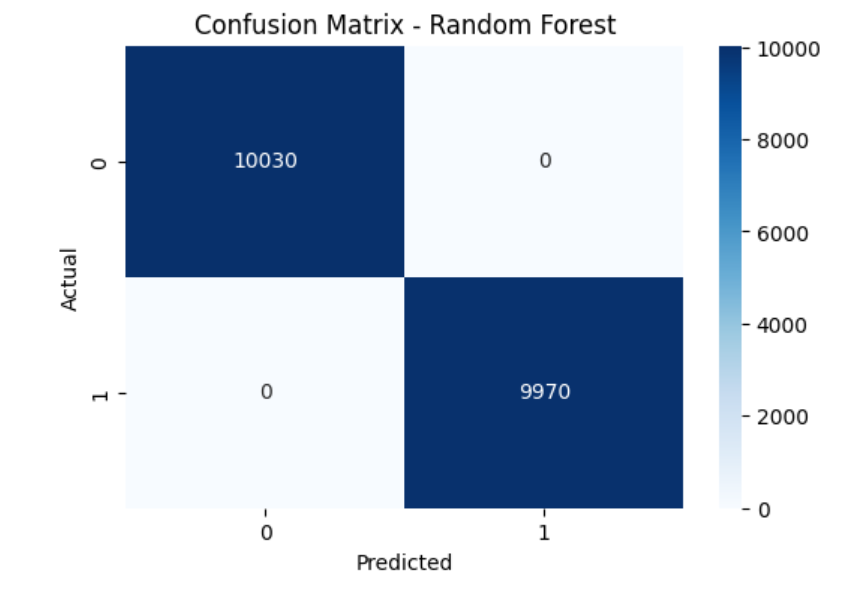
**Step 4: Evaluate and Compare the Models**

Now, let's compare the accuracy and other metrics for each classifier:

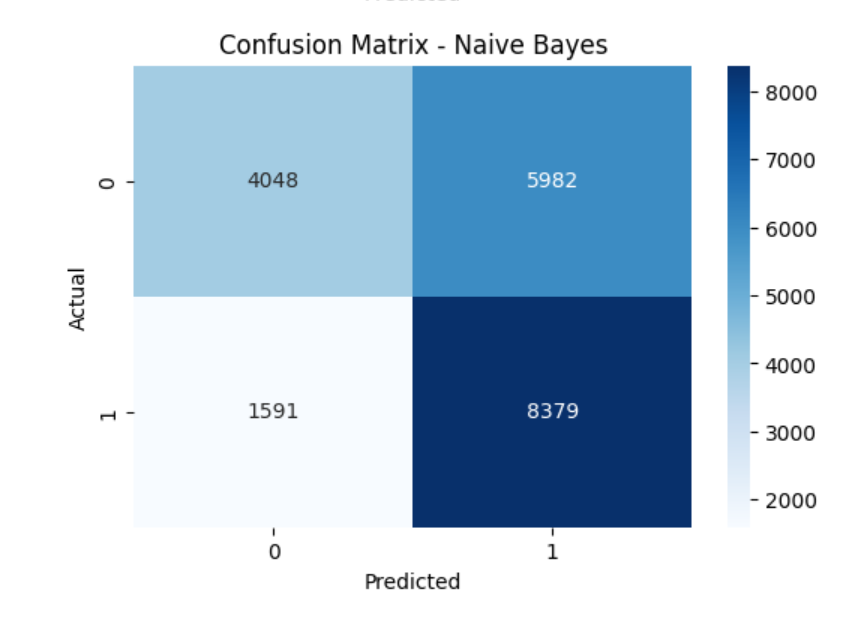
**Logistic Regression:**



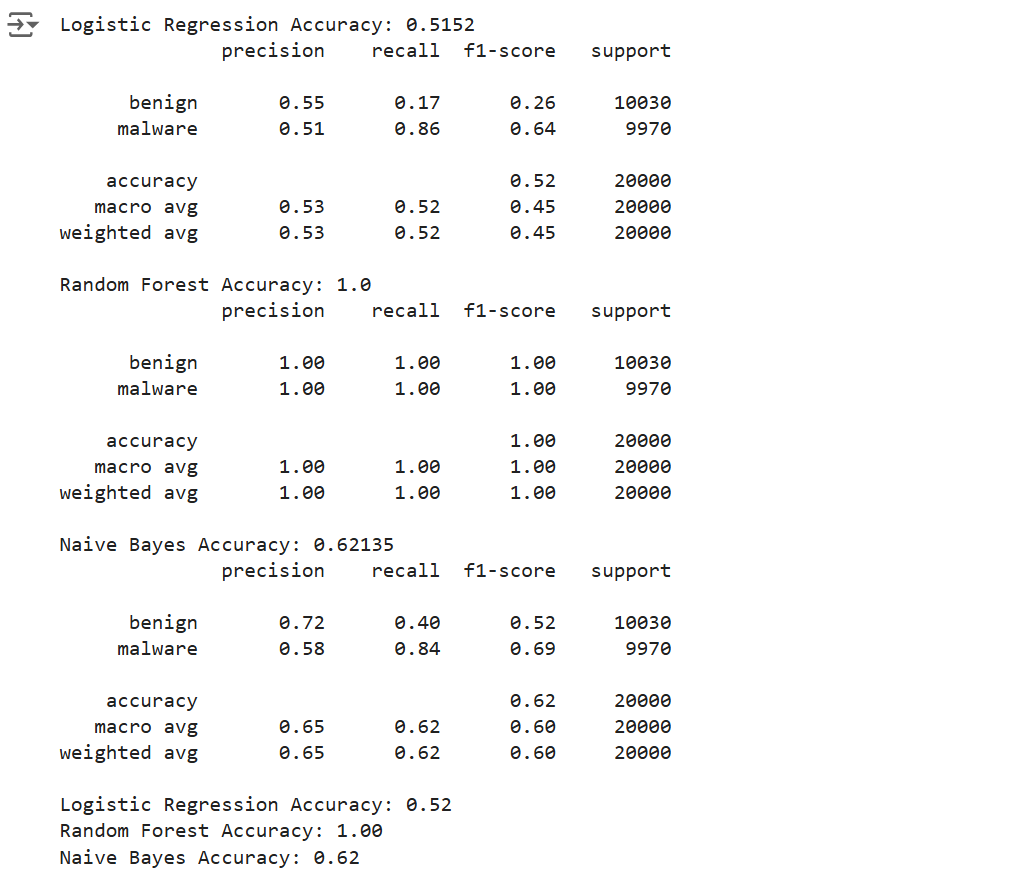
**Random Forest:**



**Naïve Bayes:**



Let’s have a look at the combined report of all these classifiers:



**Step 5: Summary**

Based on the results, we can summarize the performance of each model:

* **Logistic Regression**: A linear model that works well for problems where the relationship between the features and the target variable is linear. It achieved an accuracy of **52%**.
* **Random Forest**: An ensemble model that usually provides better accuracy than individual models by averaging multiple decision trees. It achieved an accuracy of **100%**.
* **Naive Bayes**: A simple, probabilistic model that assumes independence between features. It's particularly effective for text classification and works well with small datasets. It achieved an accuracy of 62%.

**Comparison and Insights**

* **Random Forest**: Clearly outperforms the other models with perfect scores across all metrics. This suggests that the Random Forest model is highly effective for your dataset, providing **100%** accuracy, precision, recall, and F1-score.
* **Logistic Regression**: Shows moderate performance with an accuracy of **0.52**. It has a high recall for the ‘malware’ class but low recall for the ‘benign’ class, indicating it might be better at identifying malware but struggles with benign cases.
* **Naive Bayes**: Performs better than Logistic Regression with an accuracy of **0.62**. It has a balanced performance but still falls short compared to Random Forest. The precision and recall for both classes are moderate, indicating a more balanced but less accurate model.

**Conclusion:**

The Random Forest model is the best performer among the three, providing perfect classification results. If we need a reliable model for your dataset, Random Forest is the way to go. However, if we need a simpler model or want to understand the trade-offs, Logistic Regression and Naive Bayes offer insights into different aspects of our data’s classification challenges.