Assignment: Final Project Report

Course Section: 03

Date: 14 Aug 2024

Professor: Sk Md Mizanur Rahman

Nitin Vadhiya (100969422)

Dhruv (100968596)

Deepika (100963942)

Rahul (100806033)

Sidharth (100897868)

Table of Contents

[1. Information about the dataset 3](#_Toc174531949)

[1.1 Problem statement 3](#_Toc174531950)

[1.2 Objective 3](#_Toc174531951)

[2. Proper split of dataset 3](#_Toc174531952)

[3. Building, Training, and Testing the three classifiers 4](#_Toc174531953)

[3.1 Random forest classifier 4](#_Toc174531954)

[3.2 Decision tree classifier 6](#_Toc174531955)

[3.3 Logistic regression 7](#_Toc174531956)

[4. Comparing the results 8](#_Toc174531957)

Dataset URL: <https://www.kaggle.com/datasets/teamincribo/cyber-security-attacks/data>

# Information about the dataset

## Problem statement

In the domain of network security, accurately predicting the severity level of network incidents is crucial for timely and effective response. The dataset provided includes comprehensive network traffic data, capturing various attributes such as source and destination IP addresses, ports, protocols, packet lengths, packet types, payload data, malware indicators, anomaly scores, alerts/warnings, attack types, attack signatures, actions taken, and user and device information. By leveraging this data, the goal is to develop a machine learning model to predict the **severity level** of each network incident.

## Objective

To build a predictive model that classifies network incidents into different severity levels (e.g., Low, Medium, High) based on the provided attributes in the dataset.

# Proper split of dataset

A screenshot of a computer

Description automatically generated

* The dataset was split into 80% training and 20% testing data using train\_test\_split from sklearn library.
* **stratify=y**: When you pass ‘y’ to the ‘stratify’ parameter, ‘the train\_test\_split’ function will split the data in such a way that the distribution of classes in the training and testing datasets matches the distribution in the original dataset. This is particularly useful for imbalanced classes and want to ensure that both the training and testing sets are representative of the original class distribution.

A screenshot of a computer

Description automatically generated

A screenshot of a computer code

Description automatically generated

* The above code shows that our train-test split distribution is equal.

# Building, Training, and Testing the three classifiers

## Random forest classifier

The Random Forest Classifier is an ensemble learning method that constructs multiple decision trees during training and outputs the class that is the mode of the classes (classification) of the individual trees. It is known for its robustness and effectiveness in handling large datasets with numerous features.

A screenshot of a computer program

Description automatically generated

**After performing hyperparameter tuning to optimize model performance, the following results were obtained:**

A white background with black text

Description automatically generated

## Decision tree classifier

The Decision Tree Classifier builds a model in the form of a tree structure where nodes represent feature tests, and branches represent the outcome of these tests.

A screenshot of a computer program

Description automatically generated

**After performing hyperparameter tuning to optimize model performance, the following results were obtained:**

A white background with colorful text

Description automatically generated

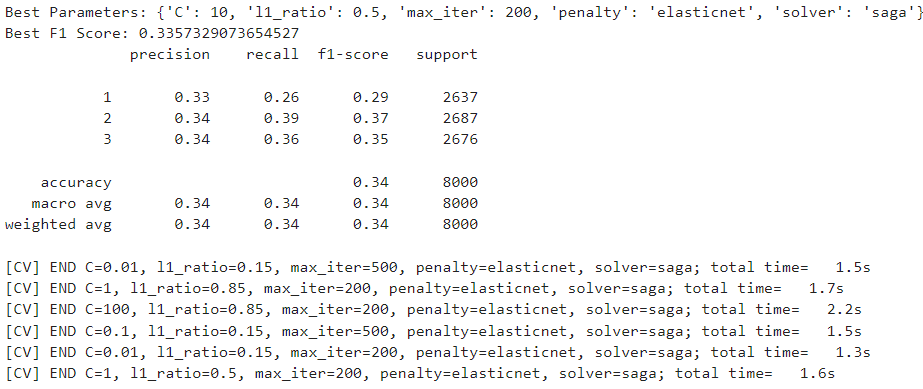
## Logistic regression

The Logistic Regression model is used for binary and multiclass classification problems. It estimates the probability of a class using a logistic function.

A screenshot of a computer program

Description automatically generated

**After performing hyperparameter tuning to optimize model performance, the following results were obtained:**



# Comparing the results

**Random forest classifier:**

* **Class 1:**
  + **Precision:** 0.34 (34% of the predicted instances for Class 1 were correct).
  + **Recall:** 0.31 (31% of the actual Class 1 instances were correctly identified).
  + **F1-Score:** 0.32 (The harmonic mean of precision and recall, indicating a moderate balance between them).
* **Class 2:**
  + **Precision:** 0.33 (33% of the predicted instances for Class 2 were correct).
  + **Recall:** 0.35 (35% of the actual Class 2 instances were correctly identified).
  + **F1-Score:** 0.34 (Shows a slight improvement in the balance between precision and recall compared to Class 1).
* **Class 3:**
  + **Precision:** 0.33 (33% of the predicted instances for Class 3 were correct).
  + **Recall:** 0.34 (34% of the actual Class 3 instances were correctly identified).
  + **F1-Score:** 0.33 (Similar balance between precision and recall as in Class 2).
* **Overall:**
  + **Accuracy:** 0.33 (The model correctly classified 33% of all instances).
  + **Macro Avg:** 0.33 (Average performance across all classes).
  + **Weighted Avg:** 0.33 (Weighted average considers the number of instances in each class, taking class imbalance into account).

**Decision tree classifier:**

* **Class 1:**
  + **Precision:** 0.34 (34% of the predicted instances for Class 1 were correct).
  + **Recall:** 0.38 (38% of the actual Class 1 instances were correctly identified).
  + **F1-Score:** 0.36 (Best performance for Class 1 among the three models).
* **Class 2:**
  + **Precision:** 0.33 (33% of the predicted instances for Class 2 were correct).
  + **Recall:** 0.32 (32% of the actual Class 2 instances were correctly identified).
  + **F1-Score:** 0.33 (Lower balance than in Class 1).
* **Class 3:**
  + **Precision:** 0.34 (34% of the predicted instances for Class 3 were correct).
  + **Recall:** 0.31 (31% of the actual Class 3 instances were correctly identified).
  + **F1-Score:** 0.32 (Slightly lower balance than in Class 2).
* **Overall:**
  + **Overall: Accuracy:** 0.34 (Matches Random Forest Classifier).
  + **Macro Avg:** 0.34 (Highest average across classes).
  + **Weighted Avg:** 0.34 (Shows consistent performance across all classes).

**Logistic regression:**

* **Class 1:**
  + **Precision:** 0.33 (33% of the predicted instances for Class 1 were correct).
  + **Recall:** 0.26 (Only 26% of the actual Class 1 instances were correctly identified).
  + **F1-Score:** 0.29 (Lower score due to the imbalance between precision and recall).
* **Class 2:**
  + **Precision:** 0.34 (34% of the predicted instances for Class 2 were correct).
  + **Recall:** 0.39 (39% of the actual Class 2 instances were correctly identified).
  + **F1-Score:** 0.37 (Higher score due to better recall).
* **Class 3:**
  + **Precision:** 0.34 (34% of the predicted instances for Class 3 were correct).
  + **Recall:** 0.36 (36% of the actual Class 3 instances were correctly identified).
  + **F1-Score:** 0.35 (Balanced but slightly higher than in Random Forest classifier).
* **Overall:**
  + **Accuracy:** 0.34 (Slightly better than Random Forest classifier).
  + **Macro Avg:** 0.34 (Indicates a general improvement).
  + **Weighted Avg:** 0.34 (Reflects better performance with class imbalances).

**Comparison:**

* **Random forest classifier** has balanced precision and recall across all classes but slightly lower F1 scores compared to the other models.
* **Decision tree classifier** achieves the best F1 score for Class 1 and maintains a balanced performance across all classes, making it the most consistent of the three.
* **Logistic regression** excels in recall for Class 2 and Class 3, leading to higher F1 scores for these classes, but it sacrifices performance for Class 1.

**Conclusion:**

* The **Random Forest Classifier** offers balanced precision and recall across all classes but has slightly lower F1 scores compared to the other models.
* **Decision tree classifier** is the best in terms of balanced precision, recall, and F1 scores, especially if consistency across all classes is desired.
* **Logistic regression** may be preferable if the priority is on better recall for certain classes, despite lower precision in others.