Cyber Attack Detection – Grad Team 9

**Aravind Julakanti, Prasanna Dwarampudi, Ritesh Aduri**

**Project Description:** Intrusion Detection Systems are the most important defense tools against sophisticated and ever-increasing network attacks. Anomaly-based intrusion detection approaches suffer from consistent and accurate performance evolutions. In this project, we train a machine learning-based approach to detecting intrusions given a training dataset. This dataset contains the most recent and benign common attacks. According to the McAfee report, this includes the most common attacks, such as DoS, DDoS, Infiltration, Heart bleed, Bot.

Malicious activities are defined as any intentional and harmful actions carried out by individuals or groups with the intent of causing damage, disruption, or unauthorized access to computer systems, networks, or digital assets.

The dataset has 760,000 observations and 78 predictors. The predictors include various features such as packet length, idle time, IAT's (Inter-arrival time) minimum and maximum, PSH (Push) flag, URG (Urgent) flag, ACK (Acknowledgment) flag, and several other parameters. The time interval between two consecutive packets arriving at a destination is referred to as inter-arrival time (IAT). The PSH (Push) TCP flag indicates that the data should be sent to the receiver as soon as possible. Another TCP flag is the URG (Urgent) flag, which indicates that the data being transmitted is urgent and should be processed immediately. The TCP ACK (Acknowledgement) flag is used to acknowledge the receipt of data from the sender.

Observations include the flow FWD, BWD packet’s length, idle time, IAT’s MIN and MAX, PSH, URG, ACK flags and several other parameters along with the response variables. All these parameters provide us an information/insight about the data which is transferred inside a network. These insight helps in differentiating the malicious and benign cases.

# Project Objective:

The primary objective of a data science cyberattack detection project is to create a model or system that can identify and detect cyber threats or attacks in real-time or near-real-time. From the given large dataset, an accurate model is to be developed to classify real-time networking data into attacks or benign cases, and to have an accurate model that can detect cyberattacks with a low false positive rate and a high true positive rate.

# Data Visualization:

Data visualization is the process of visually representing data and information by using charts, graphs, maps, and other visual aids to convey complex information in an intuitive and understandable manner. The primary goal of data visualization is to communicate insights and patterns that may be hidden in data in order to assist individuals in making better decisions and identifying trends or opportunities.

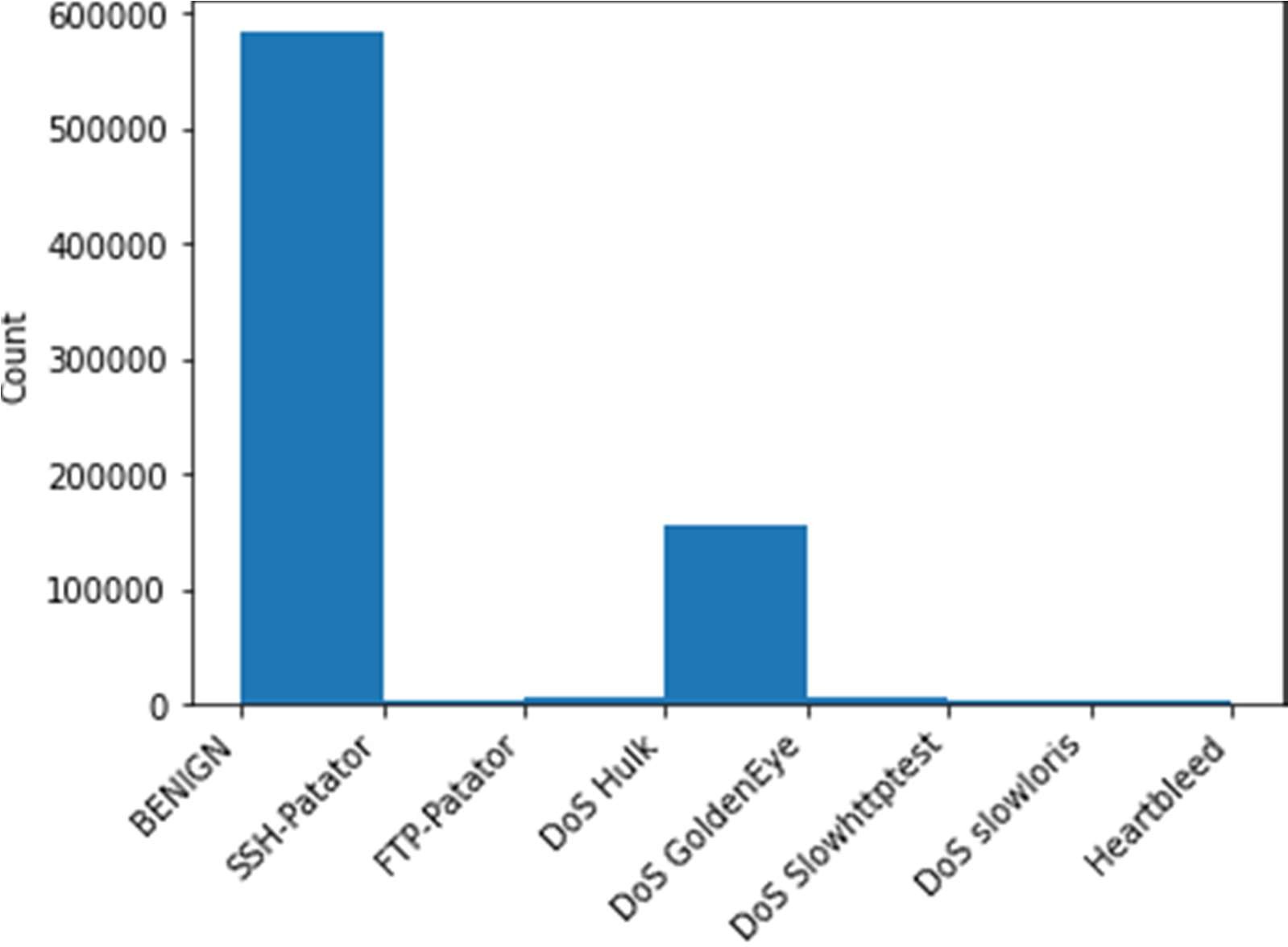


Fig 1.1 Histogram Visualization

The above figure depicts the information about the number of observations present in each of the class whereas the class heart bleed has only very few samples compared to other classes which leads to class imbalance, and it will be priority for the second deliverable.

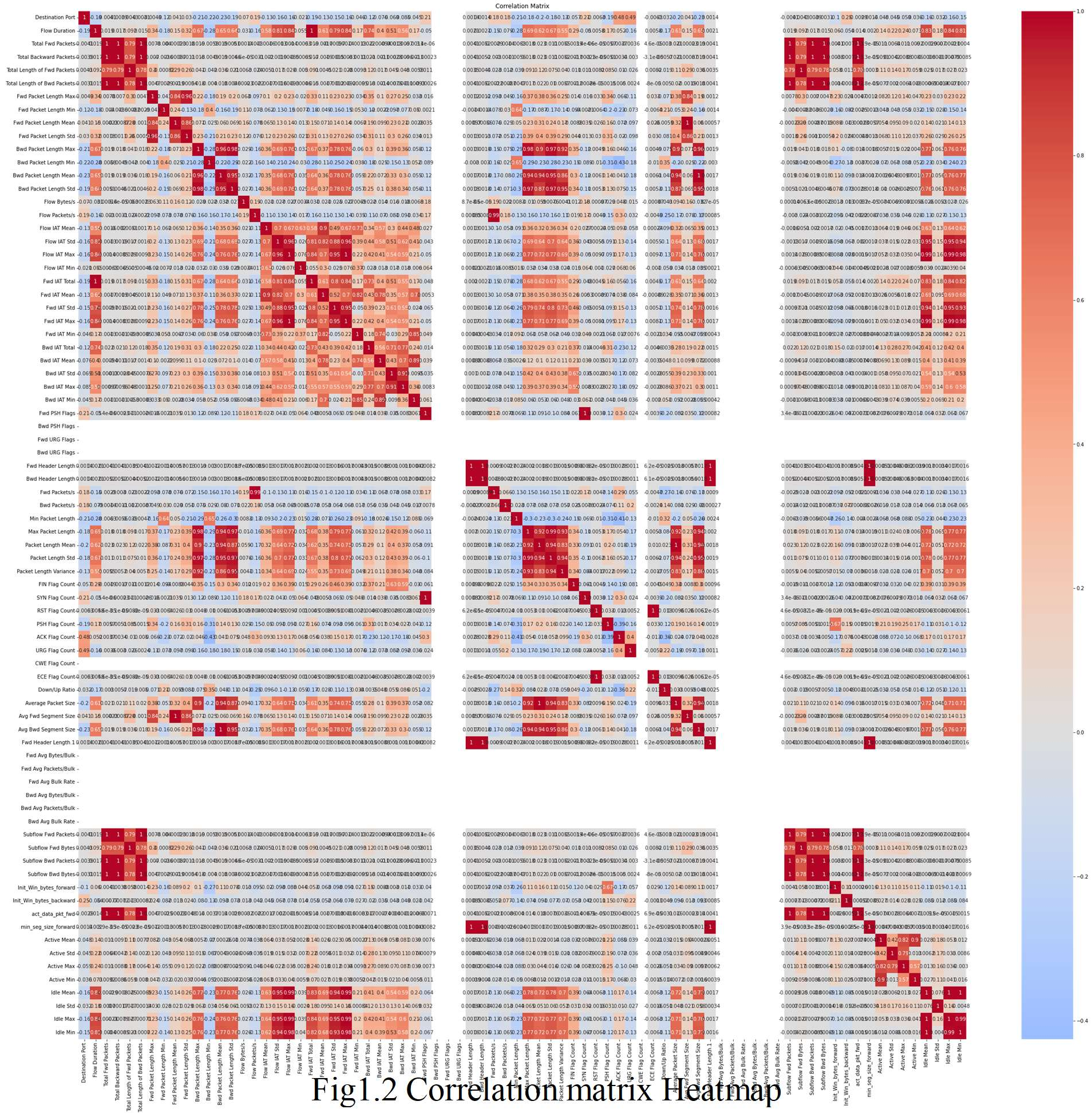


Fig1.2 Correlation matrix Heatmap

The above figure shows the correlation matrix. A correlation matrix's values range from -1 to 1, with -1 indicating perfect negative correlation, 0 indicating no correlation, and 1 indicating perfect positive correlation. As per the above heatmap, we do have a few predictor variables with positive and negative linearity. So, feature reduction will be prioritized for the upcoming deliverable.

# Data pre-processing and splitting technique:

For developing of the model, we have used train test split from the sklearn model library. The sklearn.model selection library's train test split module is used to split a dataset into training and testing subsets for machine learning applications. This enables us to evaluate the model's performance and accuracy.

We have bifurcated the data into 90 and 10 percent for training and testing respectively. Since, decision tree classifier technique is used which doesn’t require any additional pre-processing. Few of the rows which consists of extreme values on the either of the number spectrum are dropped while making sure this activity doesn’t imply in class imbalance. Without dropping the extreme values in the dataset, we have encountered run time error and computational complexity.

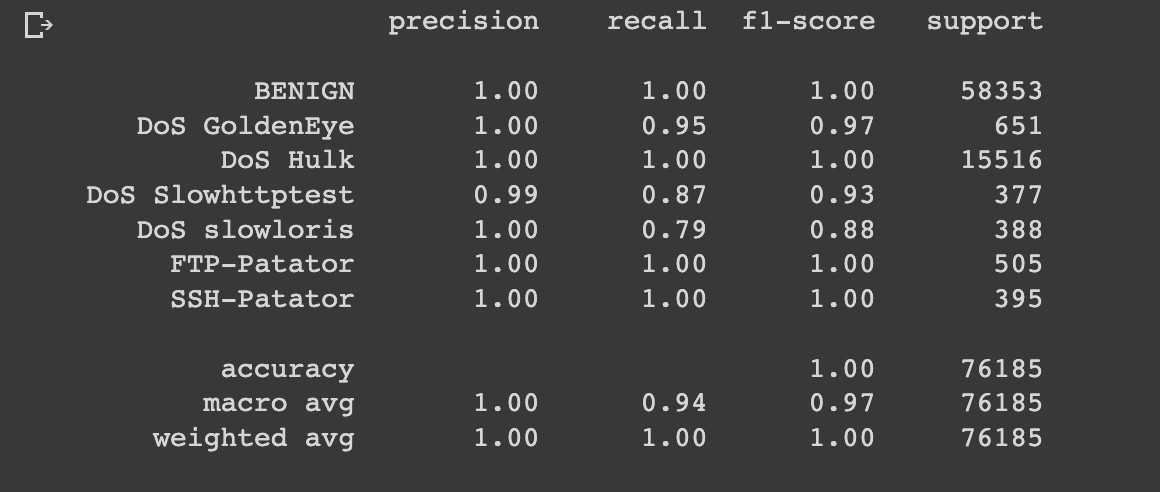
For the testing data, we have replaced the NaN values with mean values whereas positive and negative infinite with exponential values.

# Model Selection:

Decision tree classifier is used for the classification as we have various classes which are to be classified and segregated. The most important characteristics or qualities are utilized to iteratively divide the data into subsets in a decision tree classifier, a supervised learning algorithm used in machine learning. It creates a model that resembles a tree that may be used to categorize fresh instances according to their feature values.

A collection of decision rules that are drawn from the data are created by the decision tree classifier. Based on the feature values of the instance being classed, each node in the tree represents a decision point, and the branches represent potential outcomes. The ultimate decision or class classification given to the instance is provided by the leaf nodes. Decision tree classifiers often do not need resampling approaches, because they already have built-in mechanisms for dealing with over fitting and generalization.

# Results:



Classification Report.

An evaluation of a classification model's performance on test data is done using a classification report, which is a summary of the model's performance. Precision, recall, F1-score, and support for each class label are among the important metrics that are provided. Out of all the instances that were projected as positive, precision is the percentage of correctly predicted positive instances. It is calculated by dividing the total of genuine positives and false positives by the number of true positives.

Recall quantifies the share of accurately anticipated positive cases among all of the actual positive instances. It is calculated by dividing the total of true positives and false negatives by the number of true positives. The harmonic mean of recall and precision is known as the F1- score, and it provides a balance between the two measures. As 2 \* ((precision \* recall) / (precision + recall)), it is calculated.

# Conclusion:

Decision trees are helpful in many applications since they are adaptable enough to fresh data. Outliers and asymmetric data are dealt by the decision tree, which indeed helps in improving accuracy of the model. While evaluating the model performance for logistic regression we have received accuracy of 94%, however we have used the decision tree which yielded an accuracy of 99%. Hence, we have developed a model using Decision trees.

# Project Deliverable 2

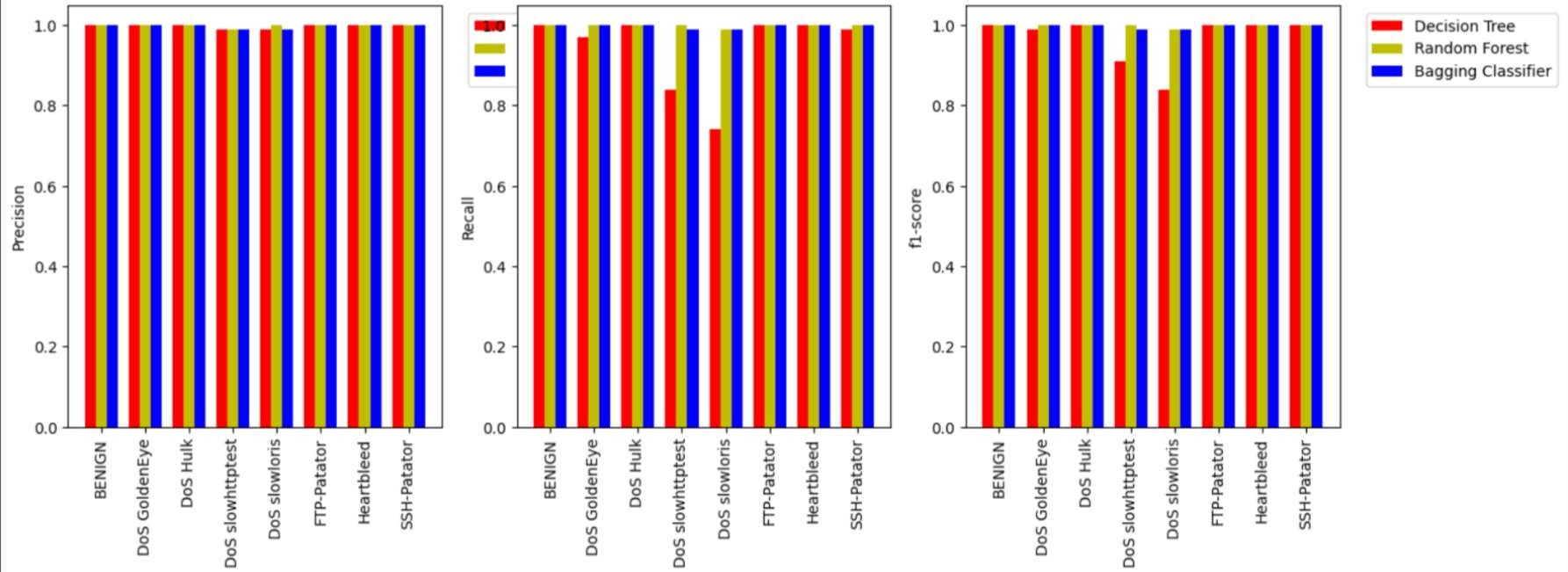
**Model Selection:**

Random forest has been used to the improve the accuracy and the classification rates for the classes which haven’t been classified properly in the previous deliverable. It’s an ensemble method, in random forest multiple decision trees are combined to create a more robust and accurate model.

Individual decision trees are constructed using a technique known as bagging, which involves sampling the data with replacement to generate multiple datasets for training each decision tree. The random forest algorithm's final prediction is obtained by aggregating the predictions of all decision trees.

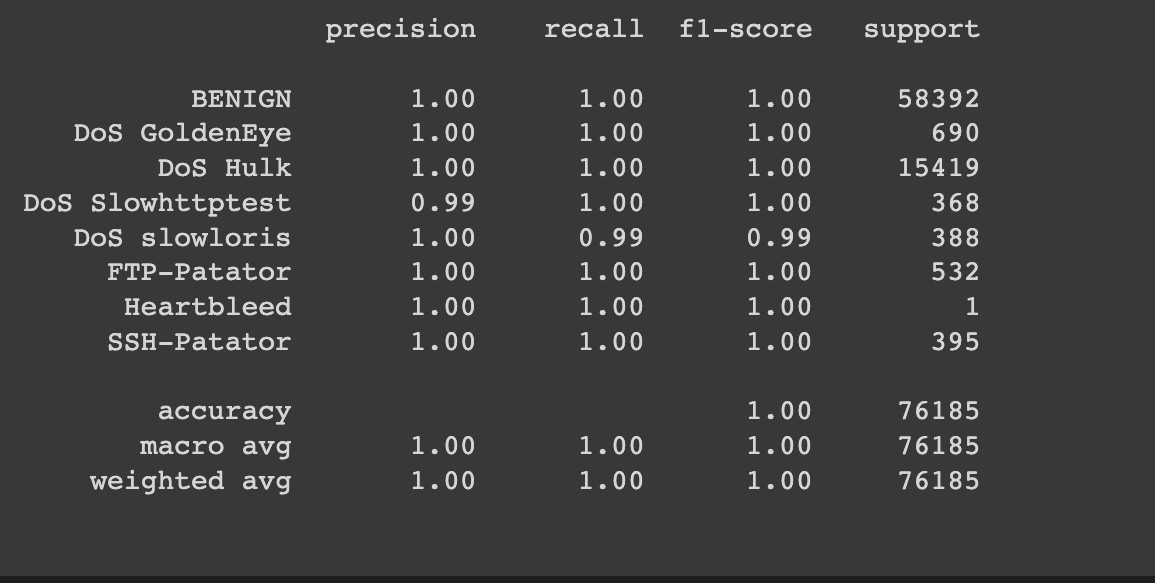
We have tried imputing data using data KNN, it has lead decrease in accuracy rate and classification rate across various classes. We have also used Grid Search, it has lead to spike in computational rate in the environment.

# Results:



Comparison of different Algorithms

Above image depicts the results across different algorithms such as Decision tree, Random Forest, Bagging Classifier.



Classification report across the class

# Conclusion:

By training each decision tree on a random subset of the features and data samples, random forest reduces over fitting. This method yields a diverse set of decision trees, each of which captures a unique aspect of the data.

Each decision tree is constructed by recursively partitioning the data into smaller subsets based on feature values. The algorithm selects the feature and threshold that best separates the data into classes or predicts the target variable at each node. The splitting process is repeated until a stopping criterion is reached, such as maximum tree depth or minimum number of samples per leaf node.

**Project Deliverable 3**

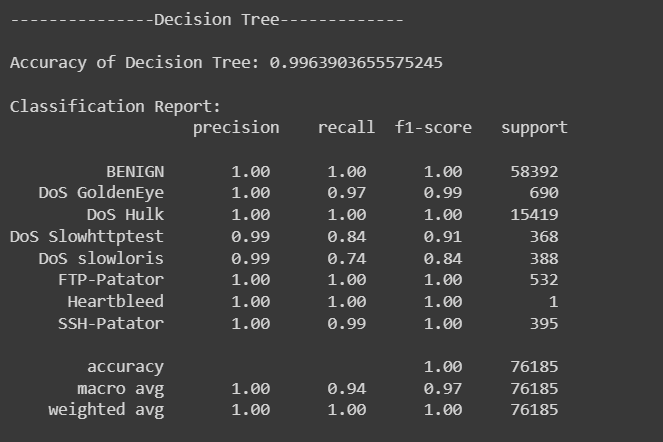
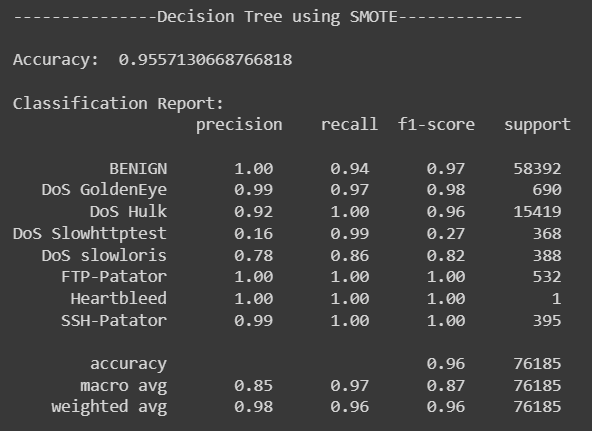
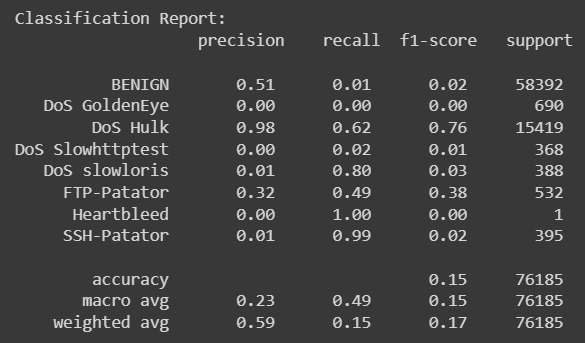
In Project Deliverable 3, we utilized various models including Decision Tree, Random Forest, AdBoost, KNN and CNN. Additionally, we applied SMOTE and NearMiss techniques to the models. Please find below a succinct report on their performance.

**Decision Tree:**

The classification performance of a decision tree model was evaluated using the provided dataset. The model achieved an accuracy of 0.996, indicating that it was able to correctly predict the class labels for almost all of the samples. The model was particularly effective at identifying the BENIGN class, with a precision, recall, and F1-score of 1.00.

The performance of the decision tree model was also evaluated after applying the SMOTE algorithm to balance the imbalanced dataset. The accuracy of the model decreased to 0.956, indicating that the SMOTE algorithm did not improve the classification performance of the model. The precision and recall of the BENIGN class decreased, indicating that the model was less effective at identifying this class after applying the SMOTE algorithm. However, the precision and recall of the DoS Slowhttptest class increased, indicating that the model was better able to identify this class after applying the SMOTE algorithm.

Finally, the decision tree model was evaluated after applying the NearMiss algorithm to balance the imbalanced dataset. The accuracy of the model decreased significantly to 0.15, indicating that the NearMiss algorithm did not improve the classification performance of the model. The precision, recall, and F1-score for most of the classes decreased significantly after applying the NearMiss algorithm. This indicates that the NearMiss algorithm was not effective in improving the classification performance of the decision tree model.

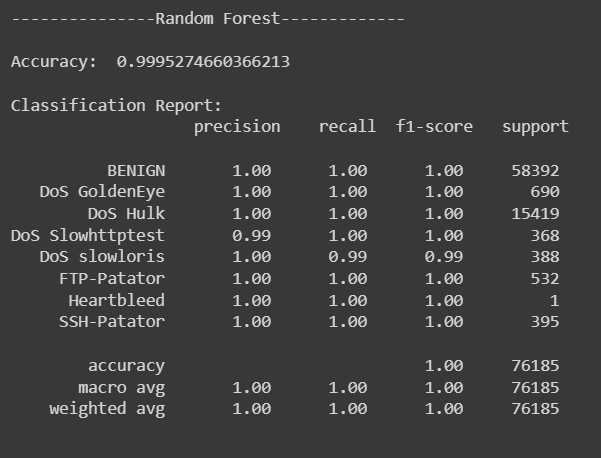
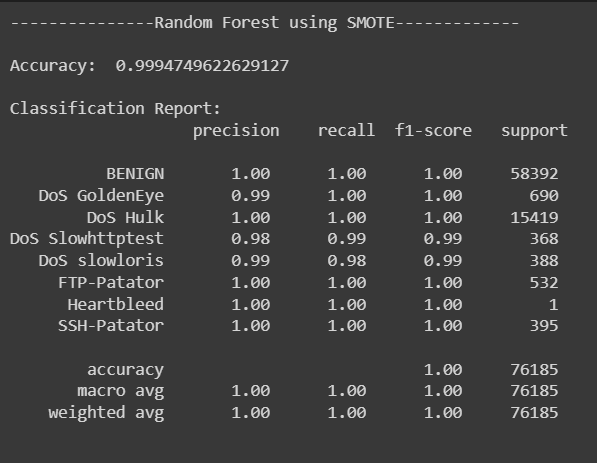
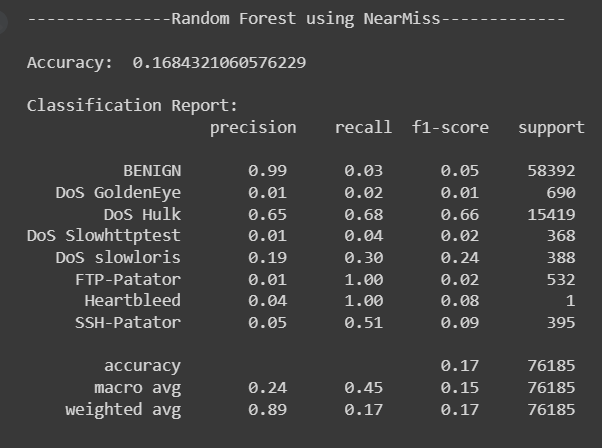
  

**Random Forest:**

In the case of Random Forest without any sampling, the accuracy achieved is 99.95%, and the classification report shows that the model performed well for all the classes with precision, recall, and f1-score values of 1.0 for most of the classes.

When using SMOTE for sampling, the accuracy achieved is 99.95%, which is similar to the model without any sampling. However, the classification report shows that the model has slightly lower precision, recall, and f1-score values for some of the minority classes like DoS GoldenEye, DoS Slowhttptest, and DoS slowloris. This might be because the SMOTE technique oversamples the minority class and tries to balance the data distribution.

When using NearMiss for sampling, the accuracy achieved is only 16.8%, which is significantly lower than the other two models. The classification report shows that the model has low precision, recall, and f1-score values for most of the classes, except for FTP-Patator, Heartbleed, and SSH-Patator. This might be because the NearMiss technique undersamples the majority class, and hence the model is not able to learn the patterns in the majority class properly.

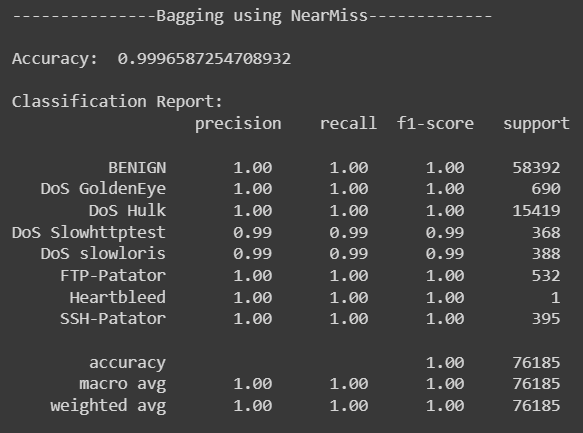
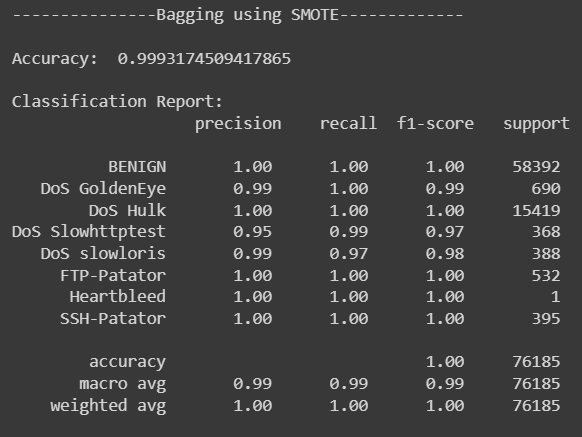
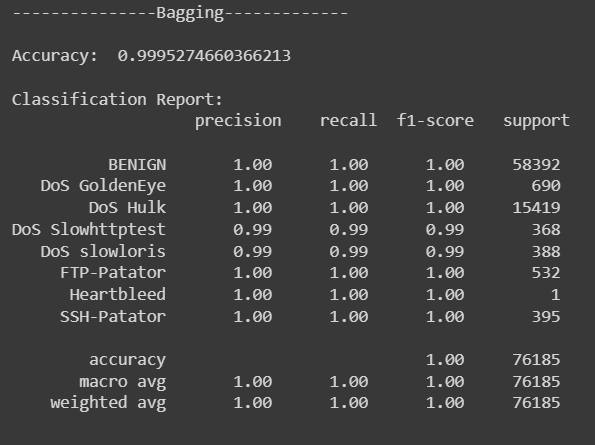
  

**Bagging**:

Bagging without any sampling technique resulted in an accuracy of 0.9995 and a weighted F1-score of 1.0. The classification report shows that all classes were predicted with high precision, recall, and F1-scores.

Bagging using SMOTE resulted in a slightly lower accuracy of 0.9993, but still performed well in predicting all classes. The model showed higher precision and recall for minority classes, such as DoS GoldenEye and DoS Slowhttptest, compared to Bagging without sampling.

Finally, Bagging using NearMiss resulted in the highest accuracy of 0.9996 and a weighted F1-score of 1.0. This technique showed similar performance to Bagging without sampling, indicating that it may be a suitable approach for this dataset.

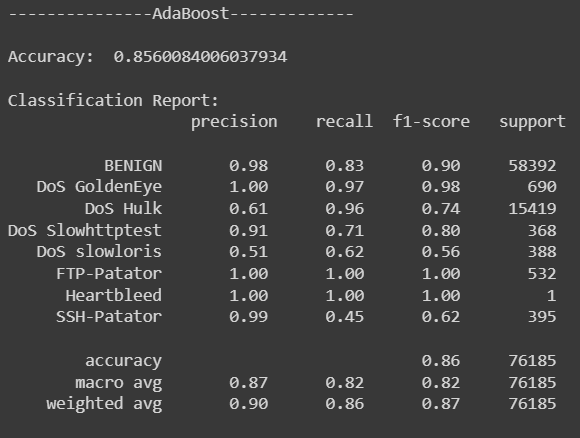
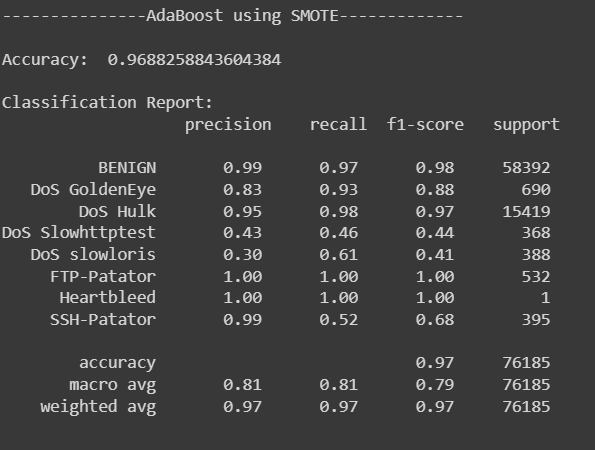
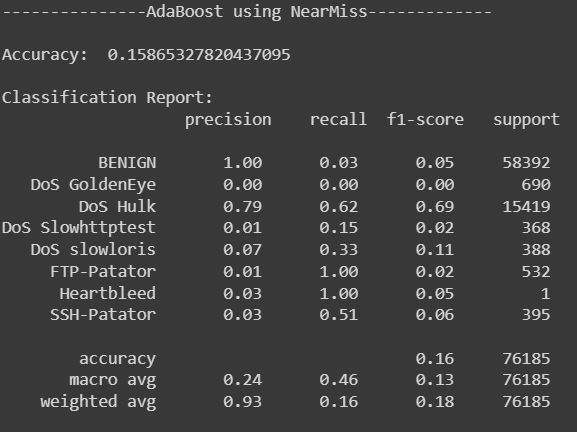


**AdaBoost:**

The results showed that the standard AdaBoost algorithm had an accuracy of 0.856 and performed reasonably well on the majority class (BENIGN) with a precision of 0.98 and a recall of 0.83. However, it performed poorly on the minority classes (DoS GoldenEye, DoS slowhttptest, and SSH-Patator) with a precision below 0.6 and a recall below 0.5.

On the other hand, the AdaBoost algorithm with SMOTE improved the overall accuracy to 0.969, indicating that the SMOTE technique helped balance the class distribution and improve the model's performance. However, this improvement came at the cost of lower precision and recall for some of the minority classes, especially DoS slowloris.

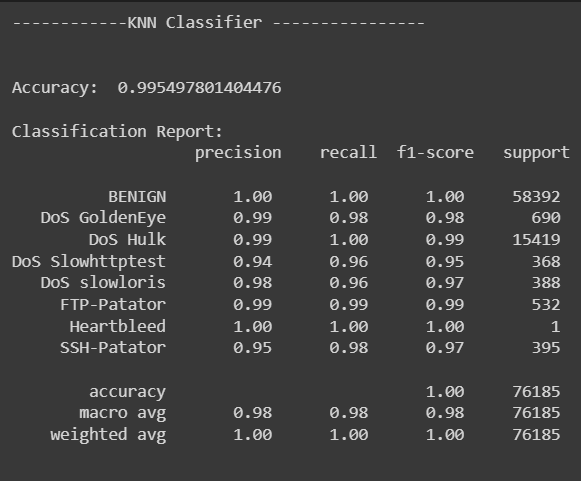
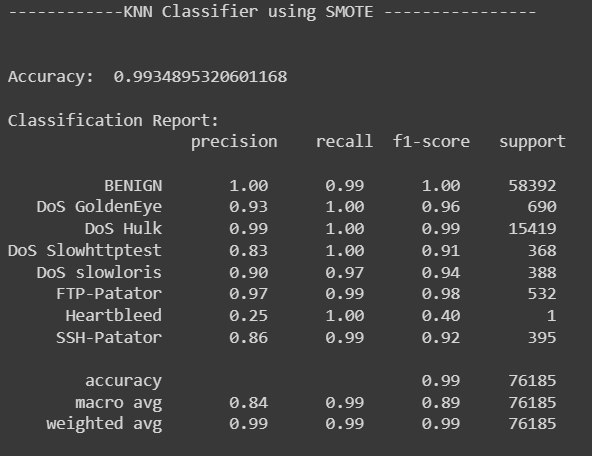
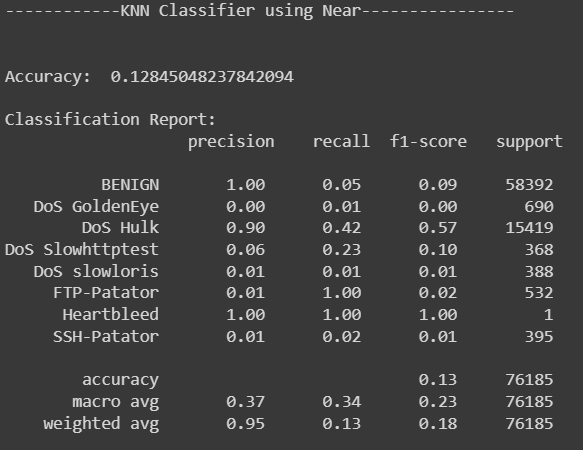
Finally, AdaBoost with NearMiss performed poorly with an accuracy of only 0.159. This indicates that the NearMiss technique did not effectively address the class imbalance problem and resulted in poor performance for all classes.

**KNN Classifier:**

KNN classifier and the KNN classifier using SMOTE performed similarly in terms of precision, recall, and f1-score for each class. However, the KNN classifier using SMOTE had lower precision, recall, and f1-score for the "DoS GoldenEye", "DoS Slowhttptest", "DoS slowloris", "Heartbleed", and "SSH-Patator" classes.

The KNN classifier using NearMiss performed poorly in all metrics, with extremely low precision, recall, and f1-score for all classes except "Heartbleed", which had a perfect score due to only having one sample in the dataset.

**Convolutional neural networks (CNNs):**

The Convolutional Neural Network (CNN) model was tested on the dataset, and the testing accuracy was found to be 0.02, which indicates poor performance. The testing classification report shows that the precision, recall, and F1-score for all classes are either zero or close to zero, except for class 0. This indicates that the model is not able to classify the network traffic accurately. The macro average F1-score for all classes is only 0.01, which suggests that the model is not able to generalize well on the dataset. The weighted average F1-score is also very low, indicating that the model's overall performance is not good. The model needs to be further optimized and trained to improve its performance on the dataset.

