IDENTIFICATION OF DDOS ATTACK

Name: Hariharan R

Reg. No.: 22011101036

Dataset Description:

A small sample of the CICIDS2017 dataset is used. It is a publicly available dataset designed for evaluating and benchmarking intrusion detection and prevention systems. It comprises a diverse range of network traffic scenarios, including benign traffic as well as various types of network attacks and anomalies. The dataset is derived from real-world traffic captured in a controlled lab environment, ensuring its relevance and applicability to real-world cybersecurity challenges. It comprises 78 features including the label.

Number of features: 78

The actual dataset : <https://www.unb.ca/cic/datasets/ids-2017.html>

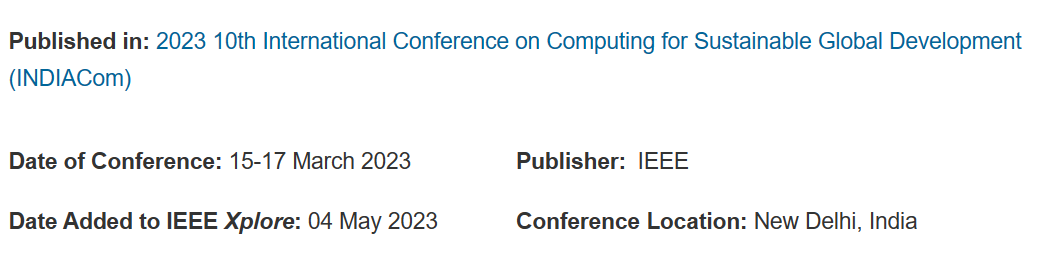
Sample Dataset: <https://www.kaggle.com/datasets/sweety18/cicids-2017-friday-noon-ddos>



RECENT RESEARCH PAPER USING THE DATASET:

[**Comparative Study of Machine Learning Techniques for Intrusion Detection on CICIDS-2017 Dataset | IEEE Conference Publication | IEEE Xplore**](https://ieeexplore.ieee.org/abstract/document/10112295)

https://ieeexplore.ieee.org/abstract/document/10112295



[Improving Multilayer-Perceptron(MLP)-based Network Anomaly Detection with Birch Clustering on CICIDS-2017 Dataset | IEEE Conference Publication | IEEE Xplore](https://ieeexplore.ieee.org/abstract/document/10152640)

https://ieeexplore.ieee.org/abstract/document/10152640

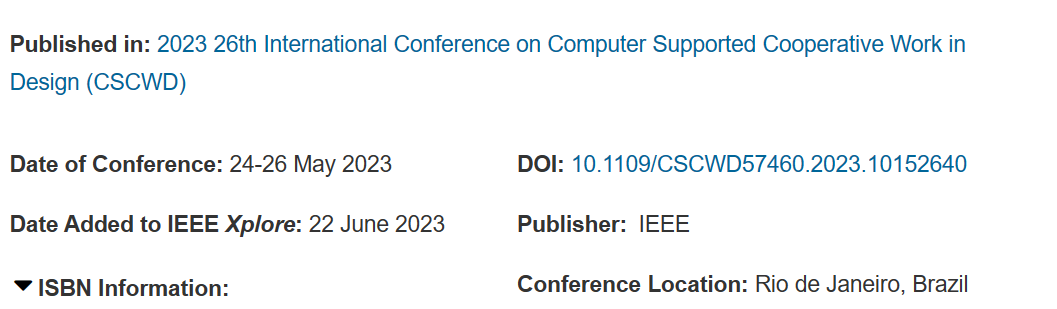


Diagram:

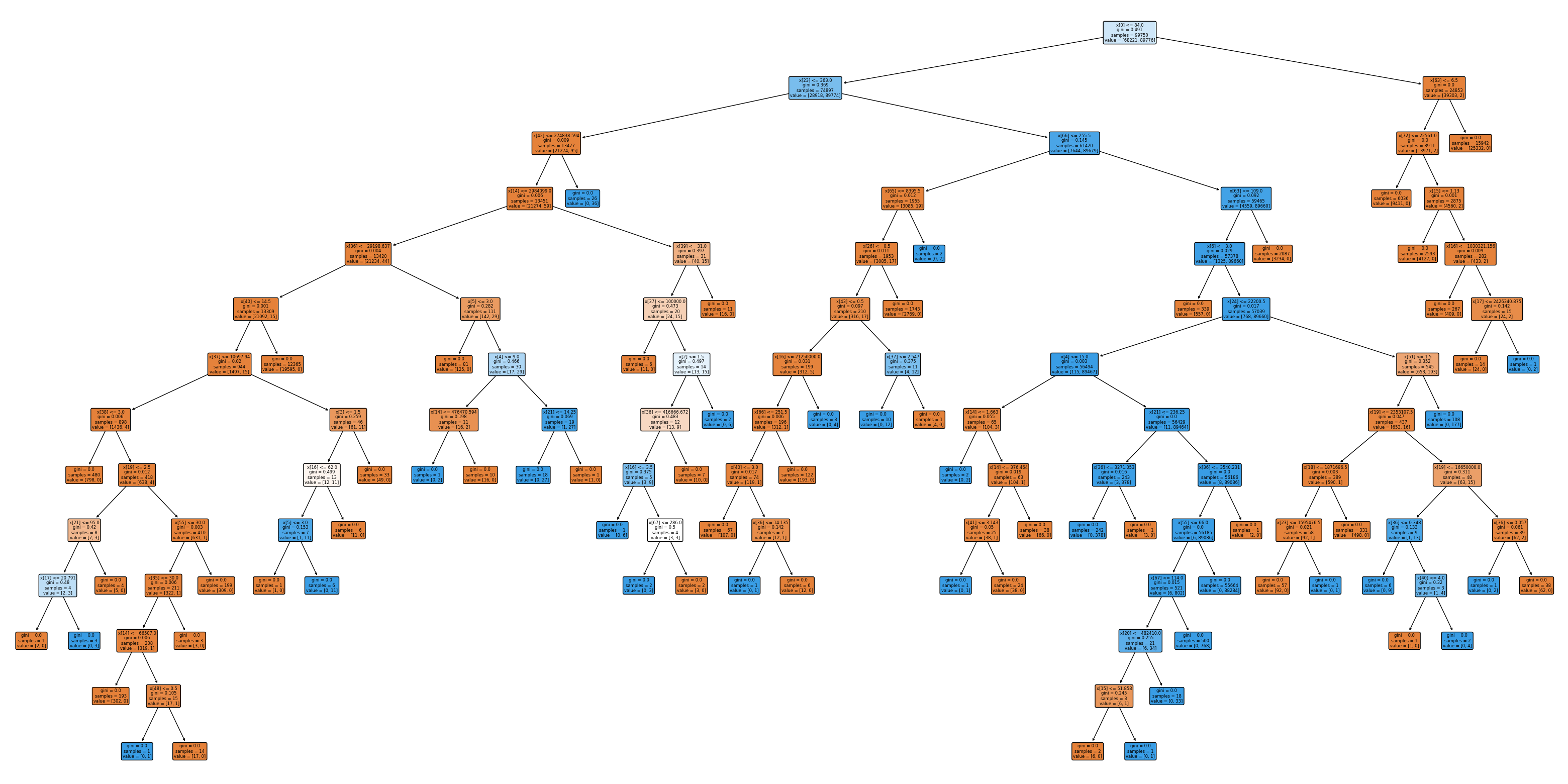
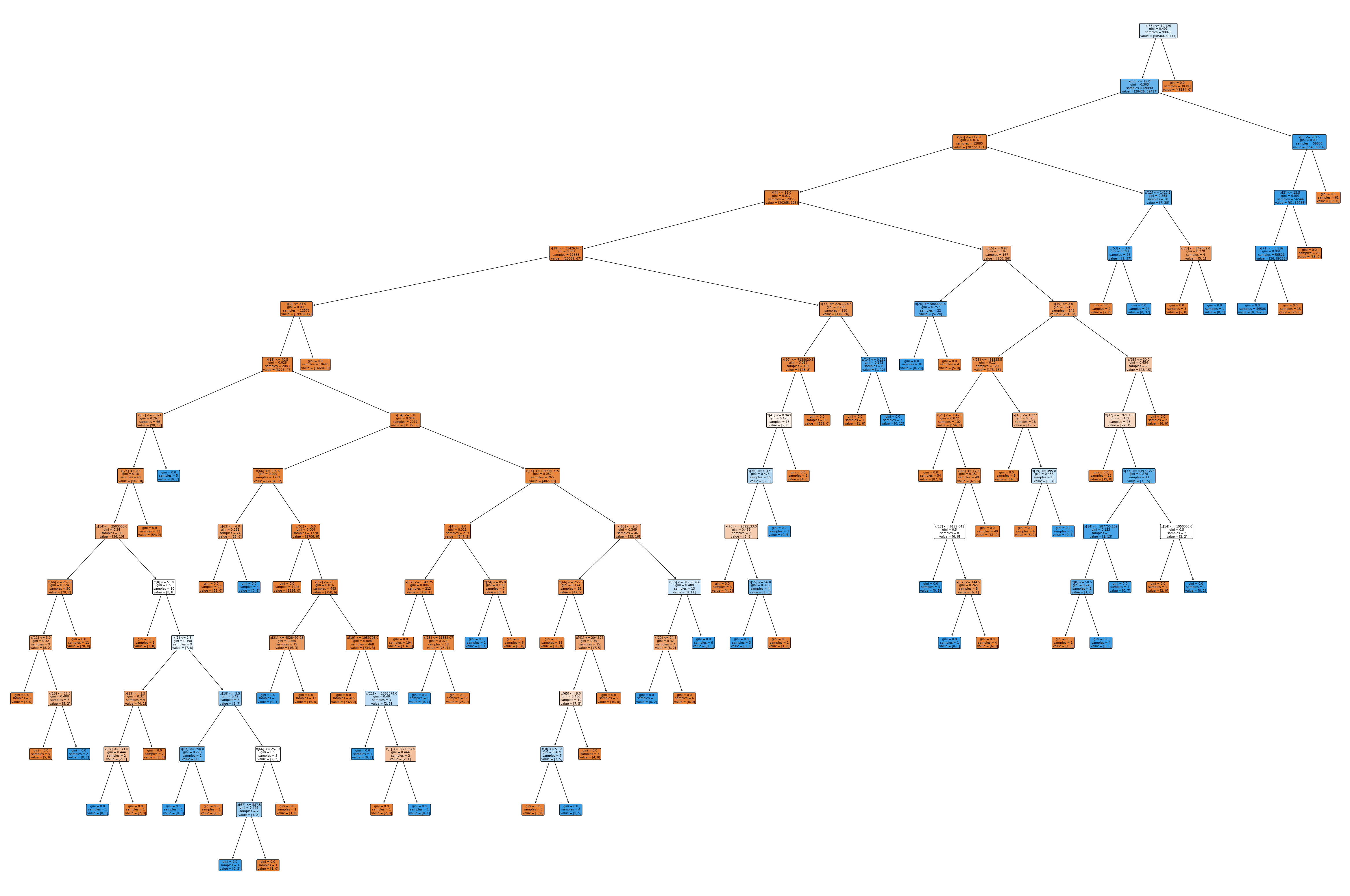


FIG 1: Tree 1 of random forest model (n=50)

Fig 2 : Tree 2 of random forest classifier(n=50)

Code:

**import** pandas **as** pd  
**import** matplotlib.pyplot **as** plt  
**import** seaborn **as** sns

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.ensemble **import** RandomForestClassifier  
**from** sklearn.linear\_model **import** LogisticRegression  
**from** sklearn.svm **import** SVC  
**from** sklearn.neural\_network **import** MLPClassifier

**from** sklearn.metrics **import** accuracy\_score, f1\_score, precision\_score, recall\_score, roc\_curve, auc, confusion\_matrix

path='/content/drive/MyDrive/Friday-WorkingHours-Afternoon-DDos.pcap\_ISCX.csv'  
 df= pd.read\_csv(path)  
 df.head()

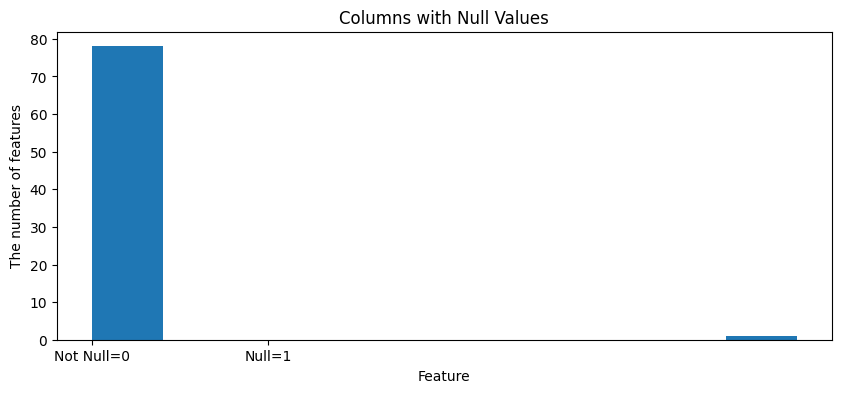
df.describe()

df.info()

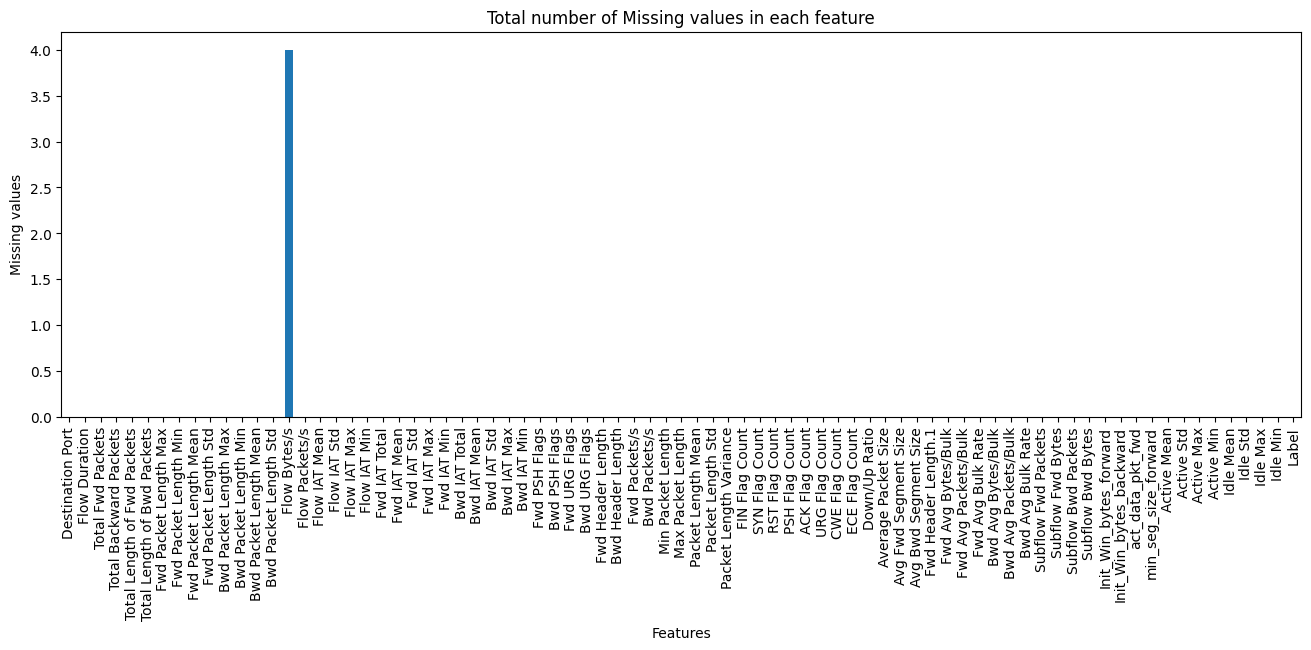
*#removing unwanted spaces infront of the column names*  
 df.columns=df.columns.str.strip()

df.info()

*#Checking for null values in the dataset.*  
plt.figure(1,figsize=( 10,4))  
plt.hist( df.isna().sum())  
   
plt.xticks([0, 1], labels=['Not Null=0', 'Null=1'])  
plt.title('Columns with Null Values')  
plt.xlabel('Feature')  
plt.ylabel('The number of features')  
   
plt.show()



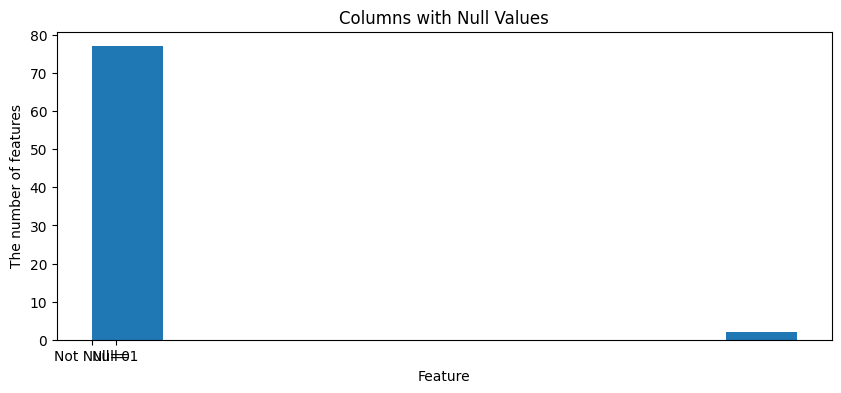
*#checking for the feature with null values*  
missing\_values = df.isnull().sum()  
fig = plt.figure(figsize=(16, 5))  
missing\_values.plot(kind='bar')  
plt.xlabel("Features")  
plt.ylabel("Missing values")  
 plt.title("Total number of Missing values in each feature")  
 plt.show()

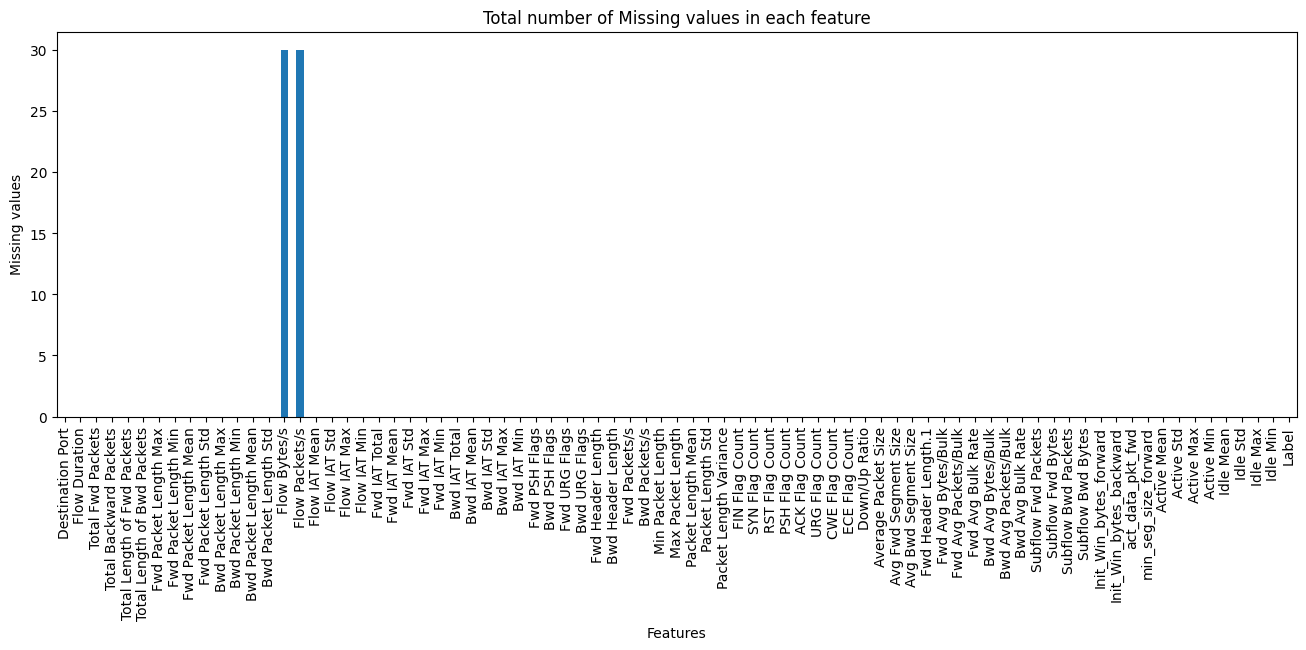


*#removing all the entries with null values*  
df=df.dropna()

pd.set\_option('use\_inf\_as\_na', True) *# Treat inf as NaN*  
null\_values=df.isnull().sum() *# Check for NaN values*

*#Checking for new null values in the dataset.*  
plt.figure(1,figsize=( 10,4))  
plt.hist( null\_values)  
   
plt.xticks([0, 1], labels=['Not Null=0', 'Null=1'])  
plt.title('Columns with Null Values')  
plt.xlabel('Feature')  
plt.ylabel('The number of features')  
   
plt.show()



*#checking for the feature with null values*  
missing\_values = null\_values  
fig = plt.figure(figsize=(16, 5))  
missing\_values.plot(kind='bar')  
plt.xlabel("Features")  
plt.ylabel("Missing values")  
plt.title("Total number of Missing values in each feature")  
plt.show()

*#dropping all the new null values*  
df=df.dropna()

df['Label'].unique()

array(['BENIGN', 'DDoS'], dtype=object)

*# Converting the labels in the DataFrame to numerical values*  
   
df['Label'] = df['Label'].map({'BENIGN': 0, 'DDoS': 1})

df.tail()

*# Split data into features and target variable*  
 X = df.drop('Label', axis=1)  
 y = df['Label']  
   
*# Split the data into training and testing sets*  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.30, random\_state=42)

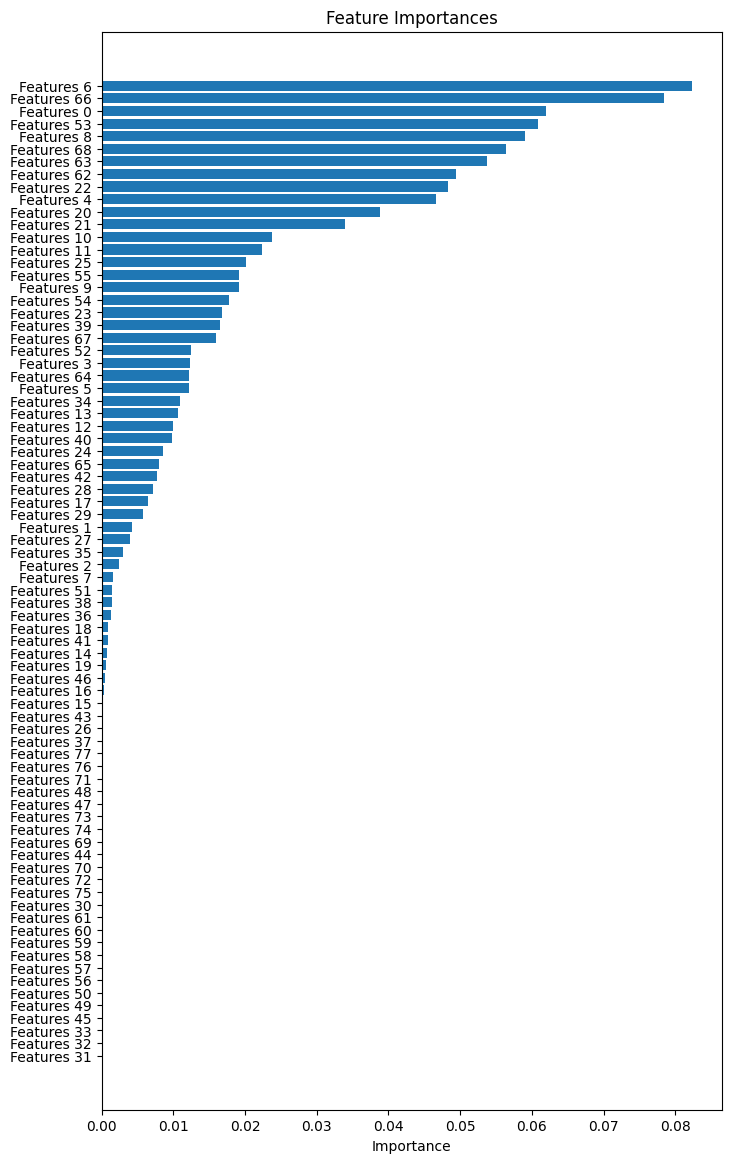
# **RANDOM FOREST**

*# Random Forest*  
 rf\_model = RandomForestClassifier(n\_estimators=50, random\_state=42)  
 rf\_model.fit(X\_train, y\_train)

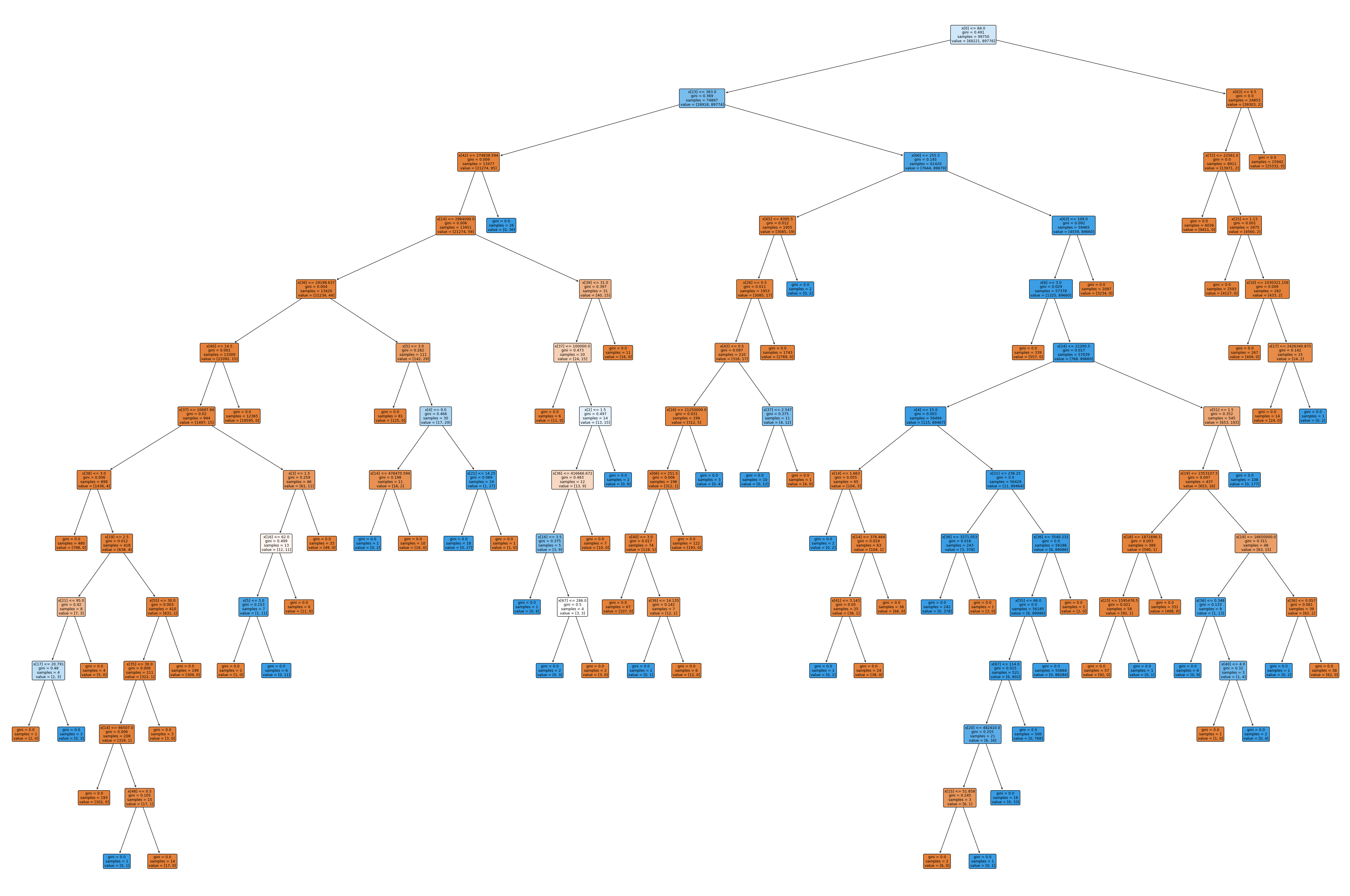
RandomForestClassifier(n\_estimators=50, random\_state=42)

rf\_pred = rf\_model.predict(X\_test)

*# Getting feature importances from the trained model*  
 importances = rf\_model.feature\_importances\_  
   
*# Getting the indices of features sorted by importance*  
 indices = sorted(range(len(importances)), key=**lambda** i: importances[i], reverse=False)  
 feature\_names = [f"Features {i}" **for** i **in** indices]  
*# Plotting feature importances horizontally*  
 plt.figure(figsize=(8, 14))  
 plt.barh(range(X\_train.shape[1]), importances[indices], align="center")  
 plt.yticks(range(X\_train.shape[1]), feature\_names)  
 plt.xlabel("Importance")  
 plt.title("Feature Importances")  
 plt.show()



**from** sklearn.tree **import** plot\_tree  
   
estimator = rf\_model.estimators\_[0] *# Selecting the first estimator from the random forest model*  
   
  
plt.figure(figsize=(60, 40))  
 plot\_tree(estimator, filled=True, rounded=True)  
 plt.show()



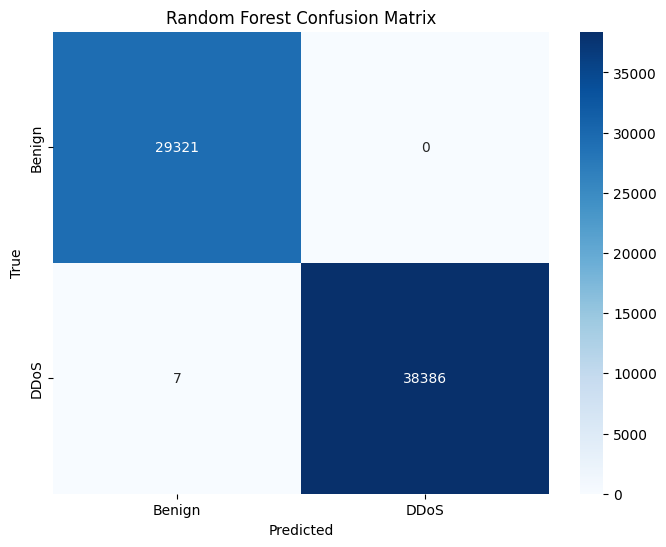
#RANDOM FOREST EVALUATION

*# Function to generate and display a detailed confusion matrix*  
 **def** plot\_confusion\_matrix(y\_true, y\_pred, classes, title):  
 cm = confusion\_matrix(y\_true, y\_pred)  
 plt.figure(figsize=(8, 6))  
 sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=classes, yticklabels=classes)  
 plt.title(title)  
 plt.xlabel('Predicted')  
 plt.ylabel('True')  
 plt.show()

*# Evaluate Random Forest*  
 rf\_accuracy = accuracy\_score(y\_test, rf\_pred)  
 rf\_f1 = f1\_score(y\_test, rf\_pred)  
 rf\_precision = precision\_score(y\_test, rf\_pred)  
 rf\_recall = recall\_score(y\_test, rf\_pred)  
   
print('\nRandom Forest Metrics:')  
 print(f'Accuracy: {rf\_accuracy:.4f}')  
 print(f'F1 Score: {rf\_f1:.4f}')  
 print(f'Precision: {rf\_precision:.4f}')  
 print(f'Recall: {rf\_recall:.4f}')

Random Forest Metrics:  
 Accuracy: 0.9999  
 F1 Score: 0.9999  
 Precision: 1.0000  
 Recall: 0.9998

*# Confusion Matrix for Random Forest*  
 plot\_confusion\_matrix(y\_test, rf\_pred, ['Benign', 'DDoS'], 'Random Forest Confusion Matrix')



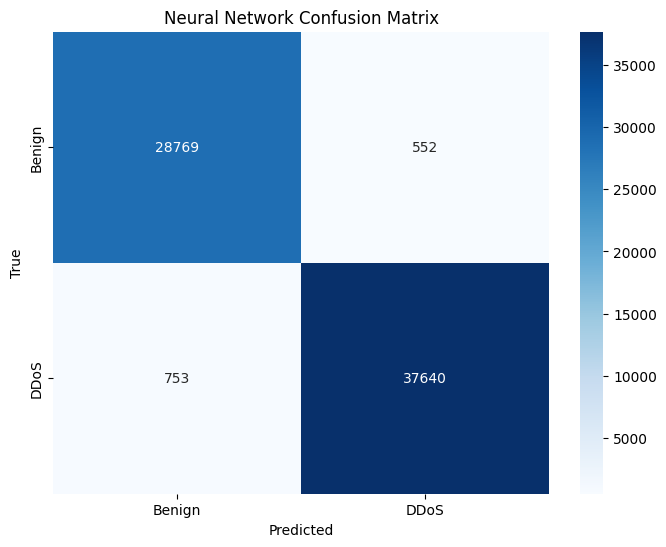
#**NEURAL NETWORK**

nn\_model = MLPClassifier(hidden\_layer\_sizes=(10,), max\_iter=10, random\_state=42)  
 nn\_model.fit(X\_train, y\_train)  
 nn\_pred = nn\_model.predict(X\_test)

nn\_accuracy = accuracy\_score(y\_test, nn\_pred)  
 nn\_f1 = f1\_score(y\_test, nn\_pred)  
 nn\_precision = precision\_score(y\_test, nn\_pred)  
 nn\_recall = recall\_score(y\_test, nn\_pred)  
   
print('\nNeural Network Metrics:')  
 print(f'Accuracy: {nn\_accuracy:.4f}')  
 print(f'F1 Score: {nn\_f1:.4f}')  
 print(f'Precision: {nn\_precision:.4f}')  
 print(f'Recall: {nn\_recall:.4f}')

Neural Network Metrics:  
 Accuracy: 0.9807  
 F1 Score: 0.9830  
 Precision: 0.9855  
 Recall: 0.9804

*# Confusion Matrix for Neural Network*  
 plot\_confusion\_matrix(y\_test, nn\_pred, ['Benign', 'DDoS'], 'Neural Network Confusion Matrix')



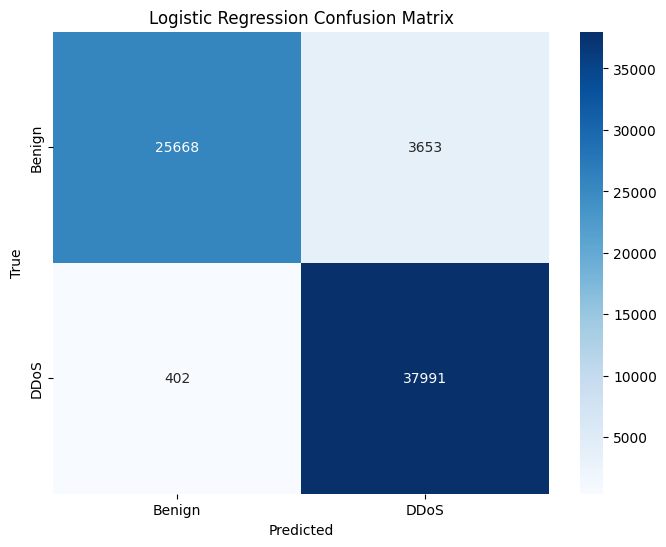
#**LOGISTIC REGRESSION**

lr\_model = LogisticRegression(random\_state=42)  
 lr\_model.fit(X\_train, y\_train)  
 lr\_pred = lr\_model.predict(X\_test)

#**EVALUATION**

lr\_accuracy = accuracy\_score(y\_test, lr\_pred)  
 lr\_f1 = f1\_score(y\_test, lr\_pred)  
 lr\_precision = precision\_score(y\_test, lr\_pred)  
 lr\_recall = recall\_score(y\_test, lr\_pred)

*# Confusion Matrix for Logistic Regression*  
 plot\_confusion\_matrix(y\_test, lr\_pred, ['Benign', 'DDoS'], 'Logistic Regression Confusion Matrix')



print('\nLogistic Regression Metrics:')  
 print(f'Accuracy: {lr\_accuracy:.4f}')  
 print(f'F1 Score: {lr\_f1:.4f}')  
 print(f'Precision: {lr\_precision:.4f}')  
 print(f'Recall: {lr\_recall:.4f}')

Logistic Regression Metrics:  
 Accuracy: 0.9401  
 F1 Score: 0.9493  
 Precision: 0.9123  
 Recall: 0.9895

Comparison:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Logistic Regression | Neural Network | Random Forest |
| Accuracy | 0.9401 | 0.9807 | 0.9999 |
| F1 score | 0.9493 | 0.9803 | 0.9999 |
| Precision | 0.9123 | 0.9855 | 1.0000 |
| Recall | 0.9895 | 0.9804 | 0.9998 |

Conclusion:

Out of all the three classifiers used, namely – Logistic Regression, Neural Network, Random Forest, the Random forest model proves to be highly precise and accurate than other three models.