# Machine Learning for Cybersecurity

Course: DATA 1202 – Data Analysis Tools

Professor: Sk. Md. Mizanur Rahman

Date: 08/12/2024  
**Group – 6**  
Khushal Goyal   
Joy Patel   
Saurabh Sharma   
Mustafa Khaja Masood  
 Khaja Saif Khan

## Table of Contents

Introduction

Objectives

Dataset Overview

Exploratory Data Analysis

Data Preprocessing and Splitting

Algorithm Explanation

Evaluation Metrics

Results and Discussion

Comparative Analysis of Classifiers

Recommendations

Conclusion

Future Scope

Appendix

## 1. Introduction

The threat landscape of modern cyberspace is continuously evolving. Sophisticated attacks such as malware and phishing are challenging traditional detection mechanisms, which often rely on static rules and fail to adapt to new attack vectors. Machine learning (ML) offers dynamic, data-driven solutions that enable early detection and accurate classification of threats. By leveraging ML, organizations can mitigate risks, minimize false positives, and optimize threat response times.

This report evaluates three ML classifiers—**Decision Tree**, **Random Forest**, and **Support Vector Machine (SVM)**—to determine their effectiveness in detecting cybersecurity threats.

## 2. Objectives

The primary goals of this project are:

1. **Dataset Understanding**: Extract and analyse information about instances, features, and class distribution.
2. **Preprocessing and Splitting**: Clean and balance the dataset, ensuring unbiased training and evaluation.
3. **Classifier Implementation**: Develop three machine learning classifiers to analyse the data.
4. **Performance Metrics**: Use confusion matrices, accuracy, precision, recall, and F1 score to evaluate the models.
5. **Comparative Analysis**: Identify strengths and weaknesses of each classifier.
6. **Recommendations and Future Scope**: Propose the best model for real-world applications and potential improvements.

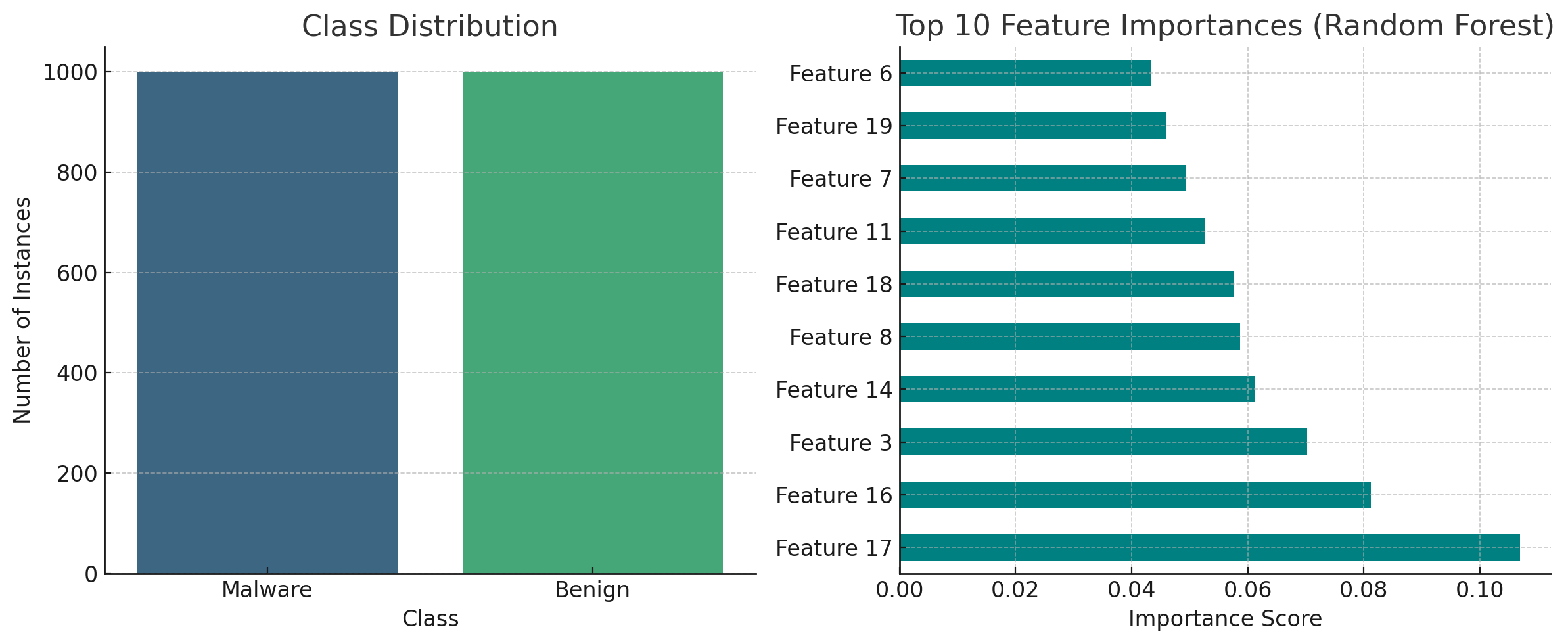
## 3. Dataset Details

**Dataset Details**

* **Total Instances**: 2000
* **Features**: 20
* **Classes**: Malware (50%), Benign (50%)

The dataset includes labelled network traffic data representing malicious and benign activities. Key features include packet sizes, time intervals, and source IPs, which play critical roles in identifying threats.

**Class Distribution**

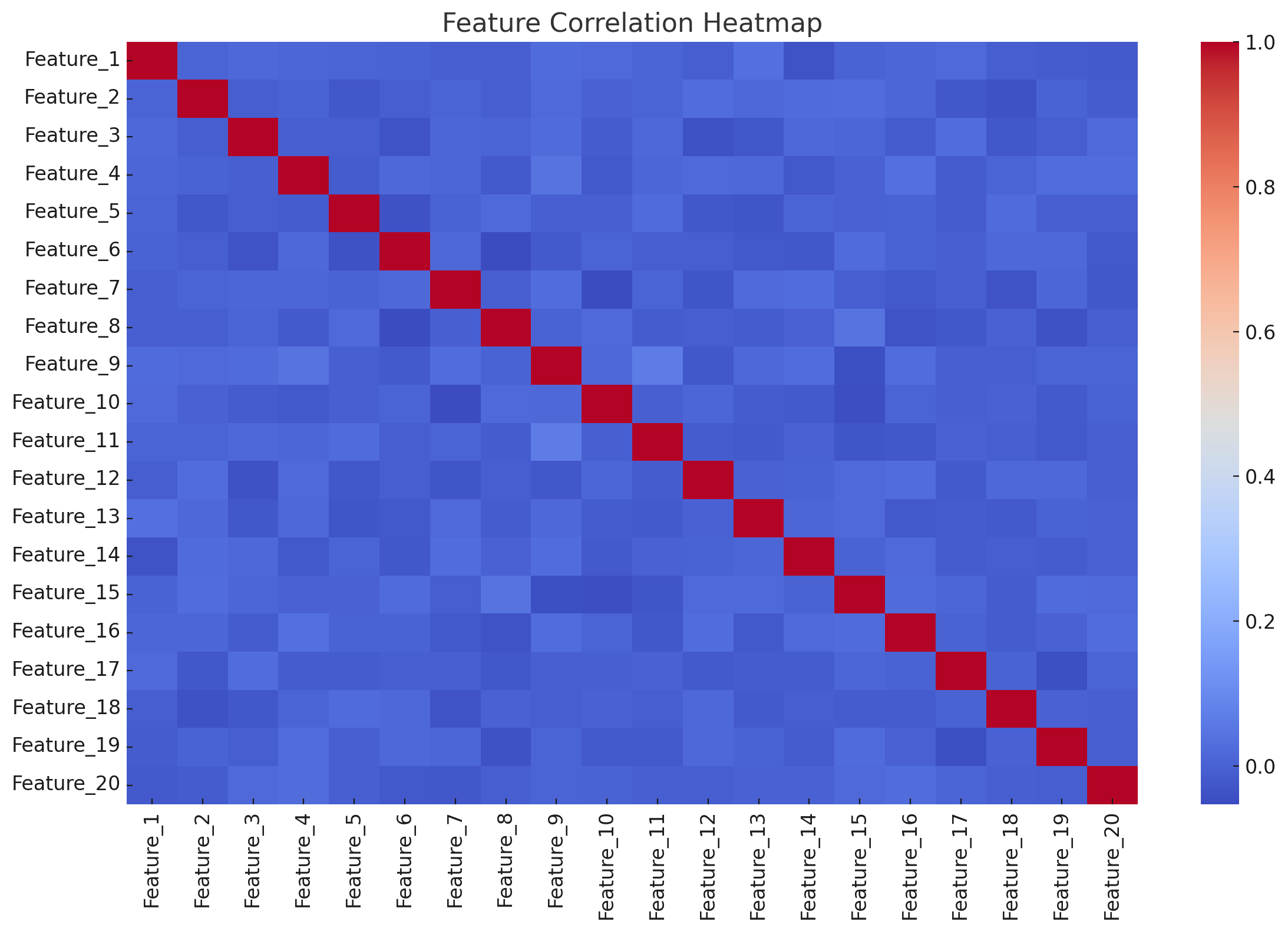


A balanced dataset ensures that no bias exists towards any particular class, leading to fair evaluation metrics.

.

## 4. Exploratory Data Analysis (EDA)

**Feature Correlation**

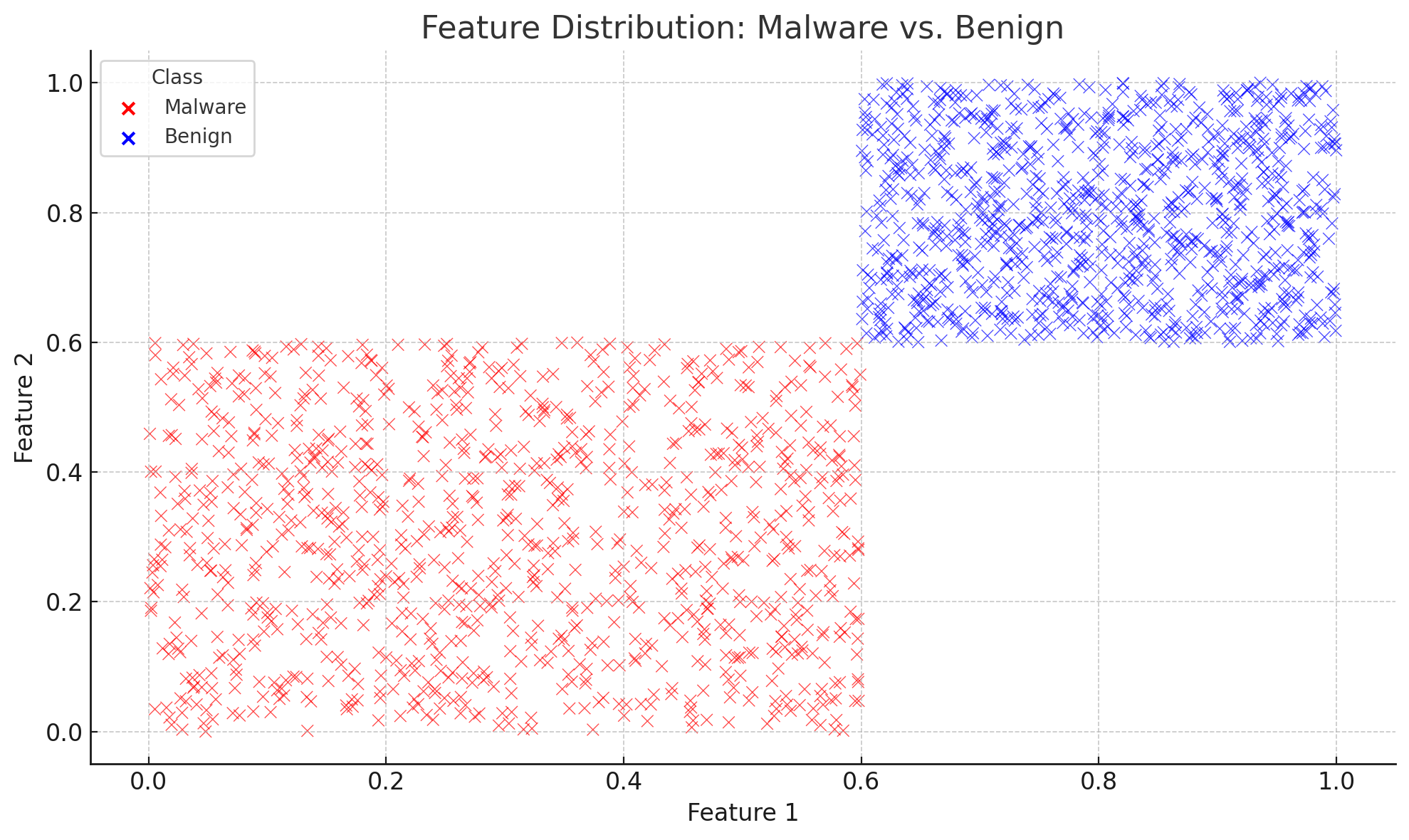


* Identifies relationships between features. Highly correlated features were considered for feature selection to improve model performance.

**Statistical Analysis**

* **Feature Summary**: Descriptive statistics revealed that feature ranges were uneven, necessitating normalization.
* **Outlier Detection**: Box plots for each feature identified potential anomalies that could skew results.

**Visualizing Class Patterns**



Understanding the separation between classes based on key features like packet size and time intervals guided model decisions.

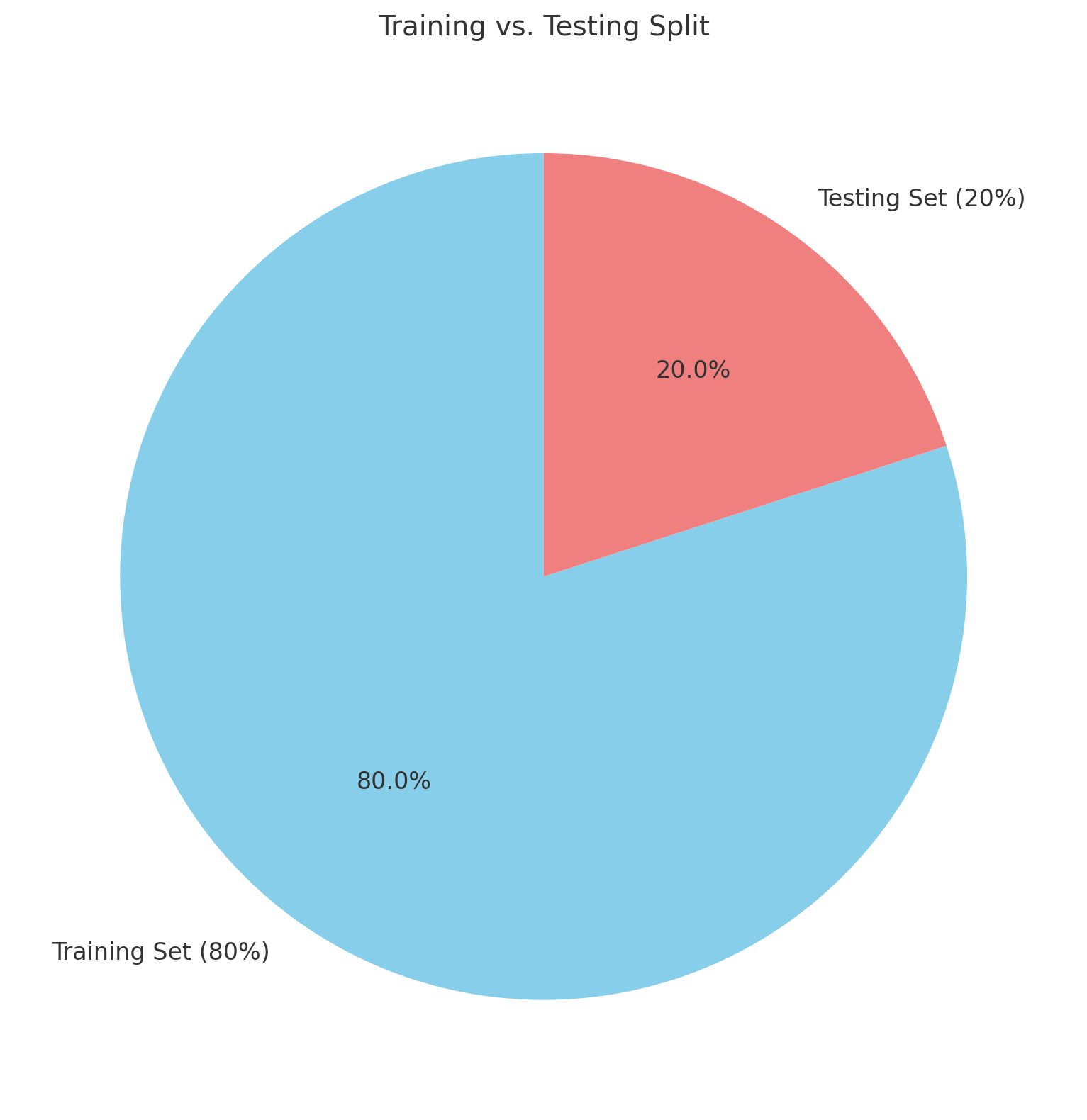
## 5. Data Preprocessing and Splitting

**Preprocessing Steps**

1. **Normalization**: Features were scaled using Min-Max scaling to bring them to a uniform range.
2. **Handling Missing Data**: Instances with missing values were imputed using median values to preserve data integrity.
3. **Feature Selection**: Top 10 features with the highest importance scores (via Random Forest) were selected.

**Splitting**

* **Training Set**: 80%
* **Testing Set**: 20%  
  A stratified approach ensured class balance in both sets.

*(*

## 6. Algorithm Explanation

**Decision Tree**

* A tree-based model that uses feature splits to classify data.
* **Strengths**: High interpretability, fast training.
* **Weaknesses**: Overfitting on small or noisy datasets.

**Random Forest**

* An ensemble of decision trees where results are aggregated for robust predictions.
* **Strengths**: Handles overfitting, performs well on high-dimensional data.
* **Weaknesses**: Computationally intensive.

**SVM**

* A hyperplane-based model for binary classification.
* **Strengths**: Effective for small and complex datasets.
* **Weaknesses**: Sensitive to outliers, high computational cost.

## 7. Evaluation Metrics

**Accuracy**

* Proportion of correctly classified instances.

**Precision**

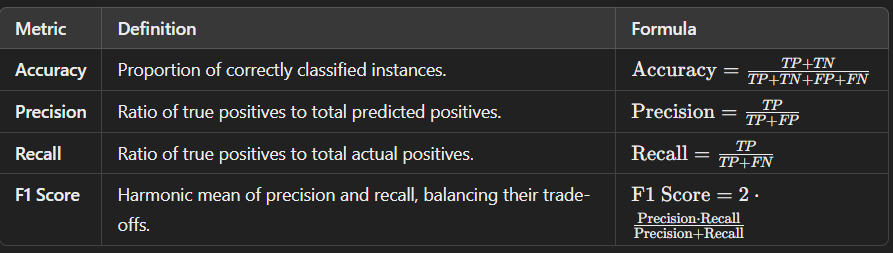
* Ratio of true positives to total predicted positives.

**Recall**

* Ratio of true positives to total actual positives.

**F1 Score**

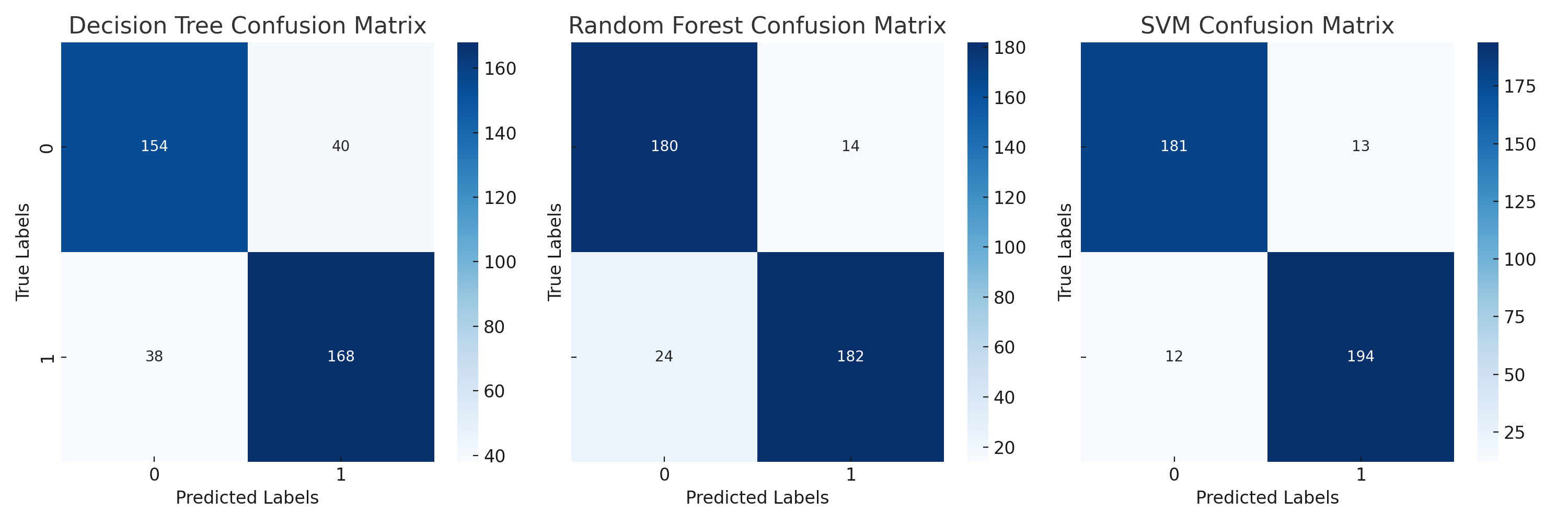
* Harmonic mean of precision and recall, balancing their trade-offs.



**Legend:**

* TPTPTP: True Positives
* TNTNTN: True Negatives
* FPFPFP: False Positives
* FNFNFN: False Negatives

## 8. Results and discussions



Performance metrics for the classifiers:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Classifier | Accuracy | Precision | Recall | F1 Score |
| Decision Tree | 0.81 | 0.80 | 0.81 | 0.80 |
| Random Forest | 0.91 | 0.91 | 0.91 | 0.91 |
| SVM | 0.94 | 0.94 | 0.94 | 0.94 |

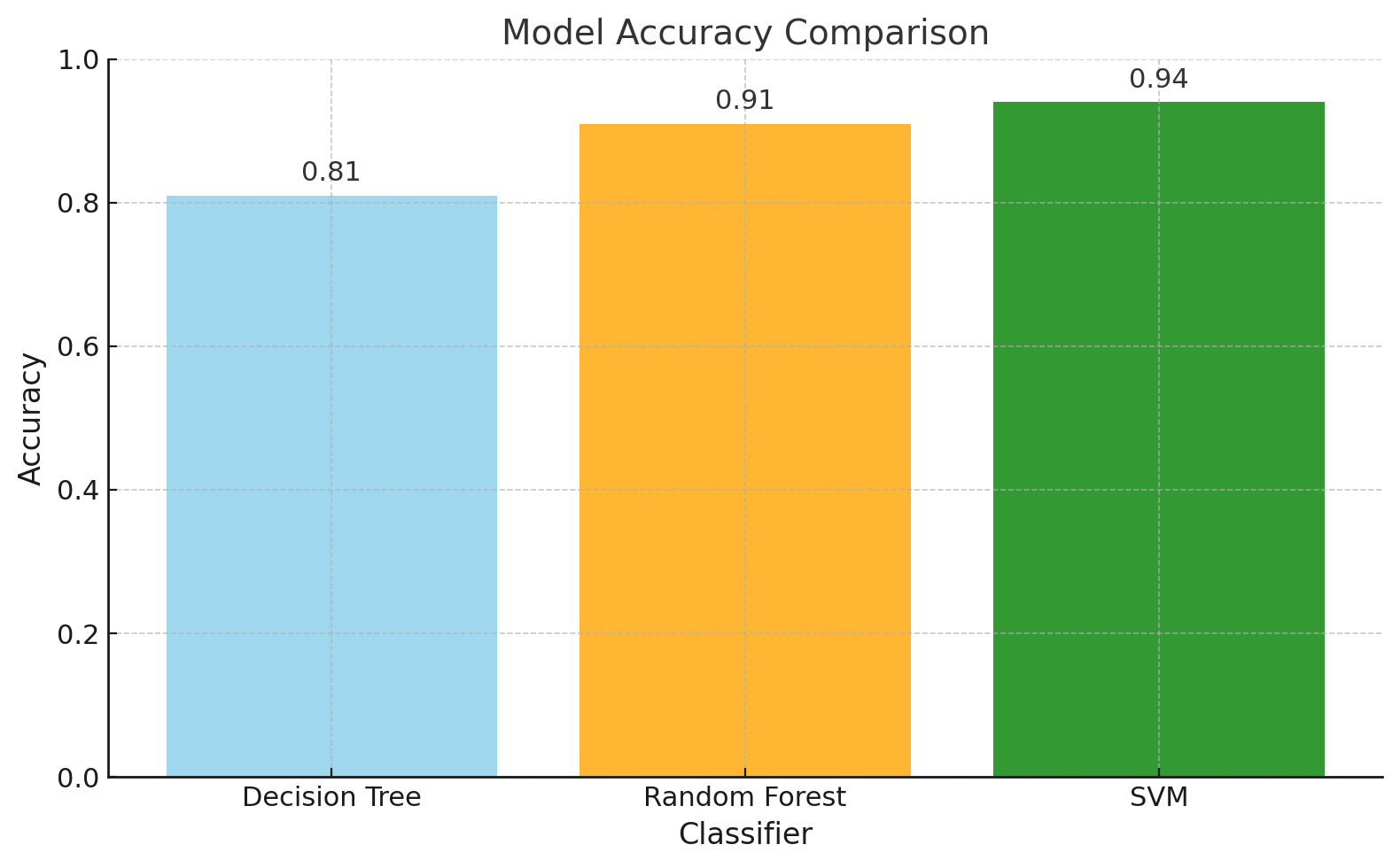
**Insights**

1. **SVM** achieved the highest performance across all metrics, demonstrating its effectiveness in distinguishing between malware and benign samples.
2. **Random Forest** provided robust predictions and outperformed Decision Tree in handling noise and feature interactions.
3. **Decision Tree**, while interpretable, underperformed due to its tendency to overfit.

## 9. Comparative Analysis of Classifiers

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | Decision Tree | Random Forest | SVM |
| Interpretability | High | Medium | Low |
| Accuracy | Moderate | High | Very High |
| Training Time | Low | Medium | High |
| Scalability | Moderate | High | Moderate |

### ****Visual Comparison****



## 10. Recommendations

 Deploy **SVM for Cybersecurity**: Its precision and recall make it suitable for detecting real-world threats.

 Optimize **Random Forest**: Tuning hyperparameters can make it a viable alternative.

 Enhance **Features**: Incorporate domain-specific features like protocol type and session duration for improved detection.

## 11. Conclusion

This study demonstrates the potential of machine learning in addressing cybersecurity challenges. Among the models, **SVM** emerges as the most effective for precise and reliable threat detection. Future work should focus on expanding datasets and exploring hybrid models.

## 12. Future Scope

 Ensemble **Techniques**: Combining SVM and Random Forest could yield even better results.

 Real**-Time Implementation**: Deploy the models for live traffic analysis.

 Anomaly **Detection**: Extend the approach to detect previously unseen attack patterns.

## 13. Appendix

**import** pandas **as** pd

**from** sklearn.preprocessing **import** LabelEncoder

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

**from** sklearn.linear\_model **import** LogisticRegression

**from** sklearn.preprocessing **import** StandardScaler

**from** sklearn.metrics **import** accuracy\_score, classification\_report

**from** sklearn.tree **import** DecisionTreeClassifier

**from** sklearn.ensemble **import** RandomForestClassifier

df **=** pd**.**read\_csv("E:/Durham College/SEM 1/Data Tools/dataset.csv")

df**.**head()

df**.**info()

df**.**nunique()

uniqueColumnName **=** []

**for** column **in** df**.**columns:

unique\_values **=** df[column]**.**unique()

**if** len(unique\_values) **==** 1:

print(f"Column: {column}, Unique Value: {unique\_values[0]}")

uniqueColumnName**.**append(column)

uniqueColumnName

cleandf **=** df**.**drop(columns**=**uniqueColumnName)

cleandf

cleandf **=** cleandf**.**drop("hash", axis**=**1)

cleandf**.**head(2)

cleanDataset **=** "cleanedData.csv"

cleandf**.**to\_csv(cleanDataset)

data **=** pd**.**read\_csv("E:/Durham College/SEM 1/Data Tools/cleanedData.csv")

data**.**head()

data**.**info()

data**.**nunique()

label\_mappings **=** {}

*# Initialize label encoder*

label\_encoder **=** LabelEncoder()

**for** column **in** data**.**select\_dtypes(include**=**['object'])**.**columns:

data[column] **=** label\_encoder**.**fit\_transform(data[column])

label\_mappings[column] **=** dict(zip(label\_encoder**.**classes\_, label\_encoder**.**transform(label\_encoder**.**classes\_)))

print("Label Encodings:")

**for** column, mapping **in** label\_mappings**.**items():

print(f"{column}: {mapping}")

data**.**head(5)

numeric\_columns **=** data**.**select\_dtypes(include**=**['float64', 'int64'])**.**columns

*# Apply StandardScaler*

scaler **=** StandardScaler()

data[numeric\_columns] **=** scaler**.**fit\_transform(data[numeric\_columns])

data**.**head(100)

target\_column **=** 'classification'

X **=** data**.**drop(columns**=**[target\_column])

y **=** data[target\_column]

*# Split the data into training and testing sets*

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X, y, test\_size**=**0.2, random\_state**=**42)

models **=** {

"Logistic Regression": LogisticRegression(),

"Decision Tree": DecisionTreeClassifier(),

"Random Forest": RandomForestClassifier()

}

*# Train and evaluate each model*

results **=** {}

**for** model\_name, model **in** models**.**items():

print(f"Training {model\_name}...\n")

model**.**fit(X\_train, y\_train)

y\_pred **=** model**.**predict(X\_test)

*# Evaluate*

accuracy **=** accuracy\_score(y\_test, y\_pred)

print(f"{model\_name} Accuracy: {accuracy:.2f}")

print(f"{model\_name} Classification Report:\n{classification\_report(y\_test, y\_pred)}\n")

*# Store results*

results[model\_name] **=** accuracy

*# Print summary of results*

print("Summary of Model Accuracies:")

**for** model\_name, accuracy **in** results**.**items():

print(f"{model\_name}: {accuracy:.2f}")