# **DDoS Detection Using Machine Learning**

## **Abstract**

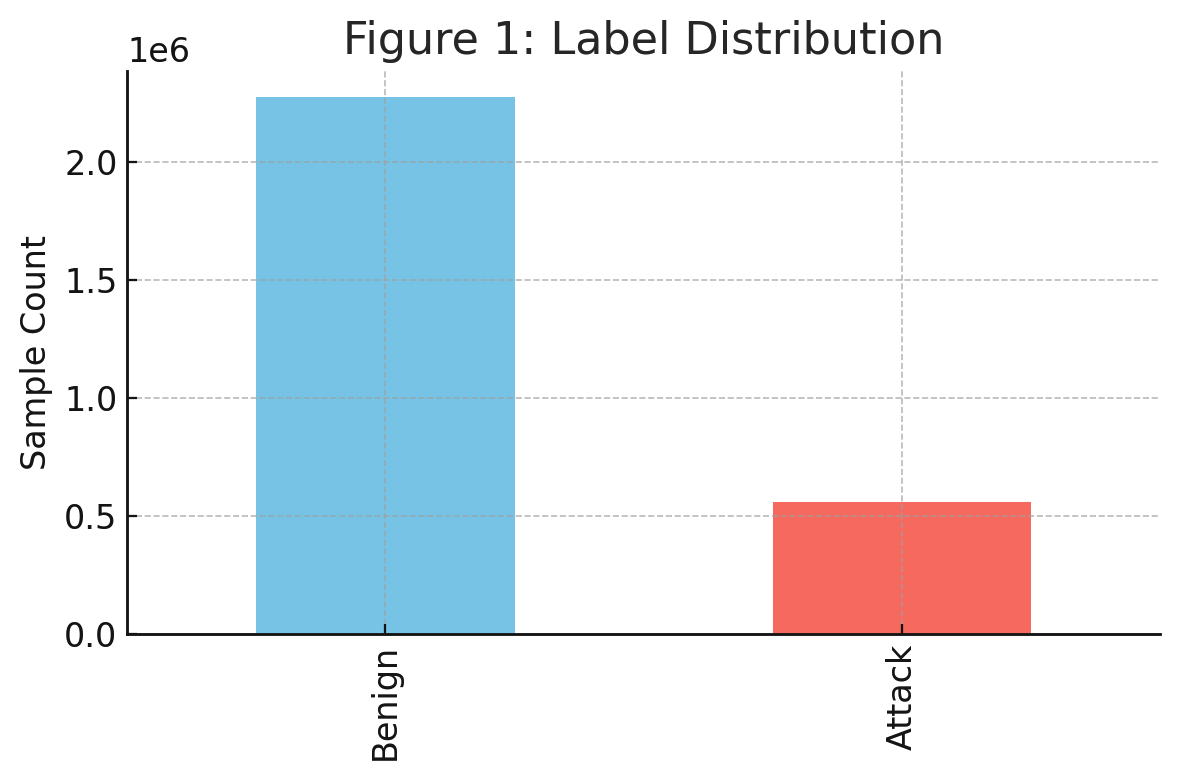
This project presents a machine learning pipeline for detecting Distributed Denial of Service (DDoS) attacks based on network traffic data. The pipeline includes outlier capping, domain-specific feature engineering, ensemble-based feature selection, and the evaluation of multiple classification models. Among the evaluated models, the Random Forest classifier achieved the highest F1 score of 0.9973. The final model, along with the data scaler, was saved for future deployment. The results demonstrate that machine learning can effectively and accurately distinguish DDoS attacks from normal network traffic.

## **1. Introduction**

DDoS attacks flood a target server or network with massive traffic to disrupt normal service. These attacks are often difficult to detect early due to traffic variability. This project aims to build a system that uses supervised learning to classify network traffic samples as either **benign** or **malicious (DDoS attack)**. By leveraging historical labeled data and automated feature engineering, the model learns to detect patterns associated with such attacks.

## **2. Dataset Description**

* **Source:** Kaggle (chethuhn/network-intrusion-dataset)
* **Files:** 8 CSV logs covering different days and scenarios
* **Total Samples:** 2,830,743
* **Features:** 79
* **Classes:**
  + Benign: 2,273,097
  + Attack: 557,646
  + Attack Rate: 19.7%

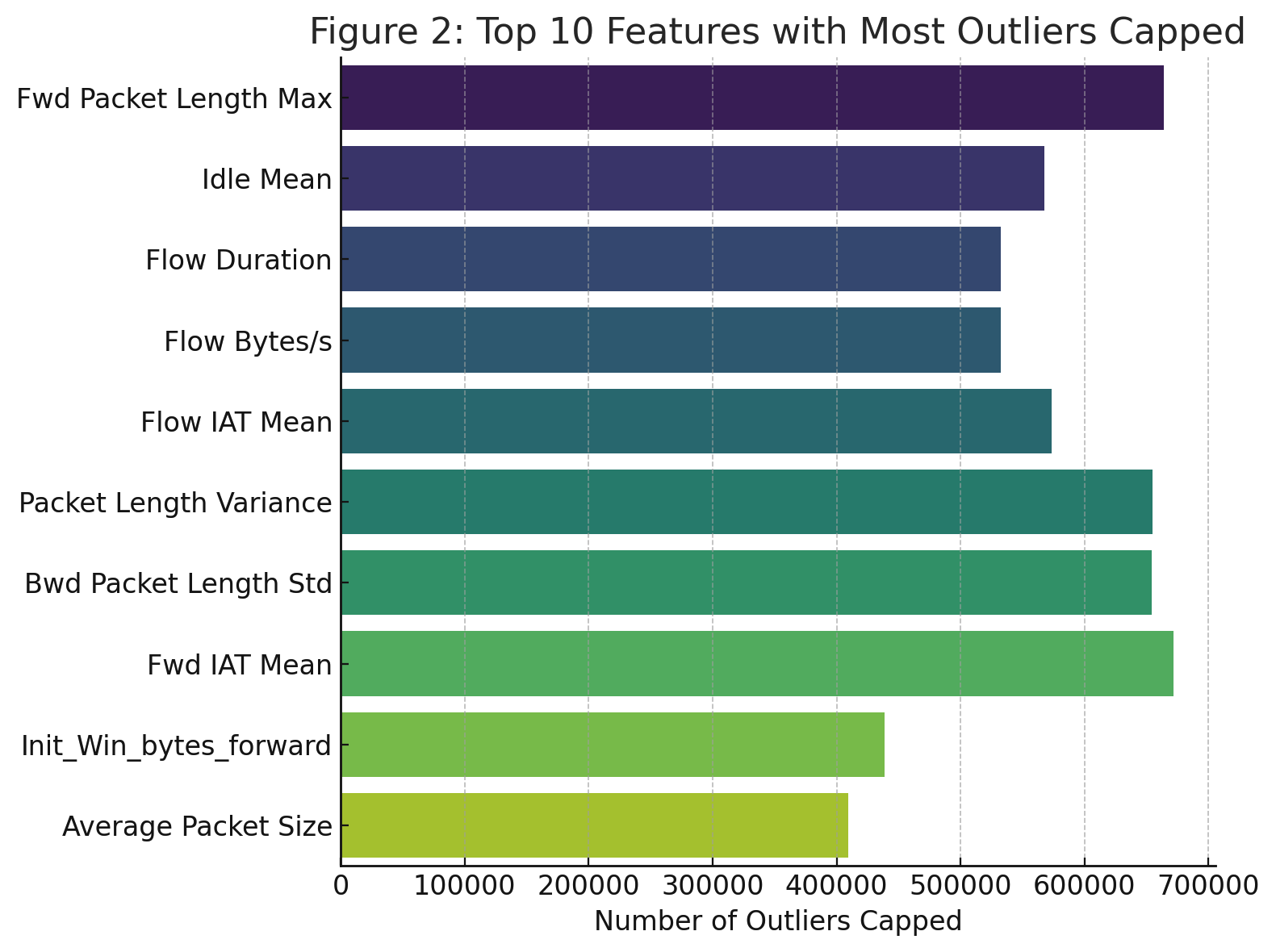


## **3. Preprocessing Pipeline**

### **3.1 Outlier Detection and Handling**

Used the **IQR method** to cap extreme values in numerical features. This preserved all data points while reducing the impact of extreme values.

* Over 20% of values in features like Fwd IAT Mean, Idle Mean, and Packet Length Variance were capped.
* Total features processed: 50+



### **3.2 Feature Engineering**

Added 17 domain-specific features to capture behavior patterns in the data, including:

* Byte-per-duration rates
* Directional traffic asymmetry
* Port classification (well-known, registered, dynamic)

### 

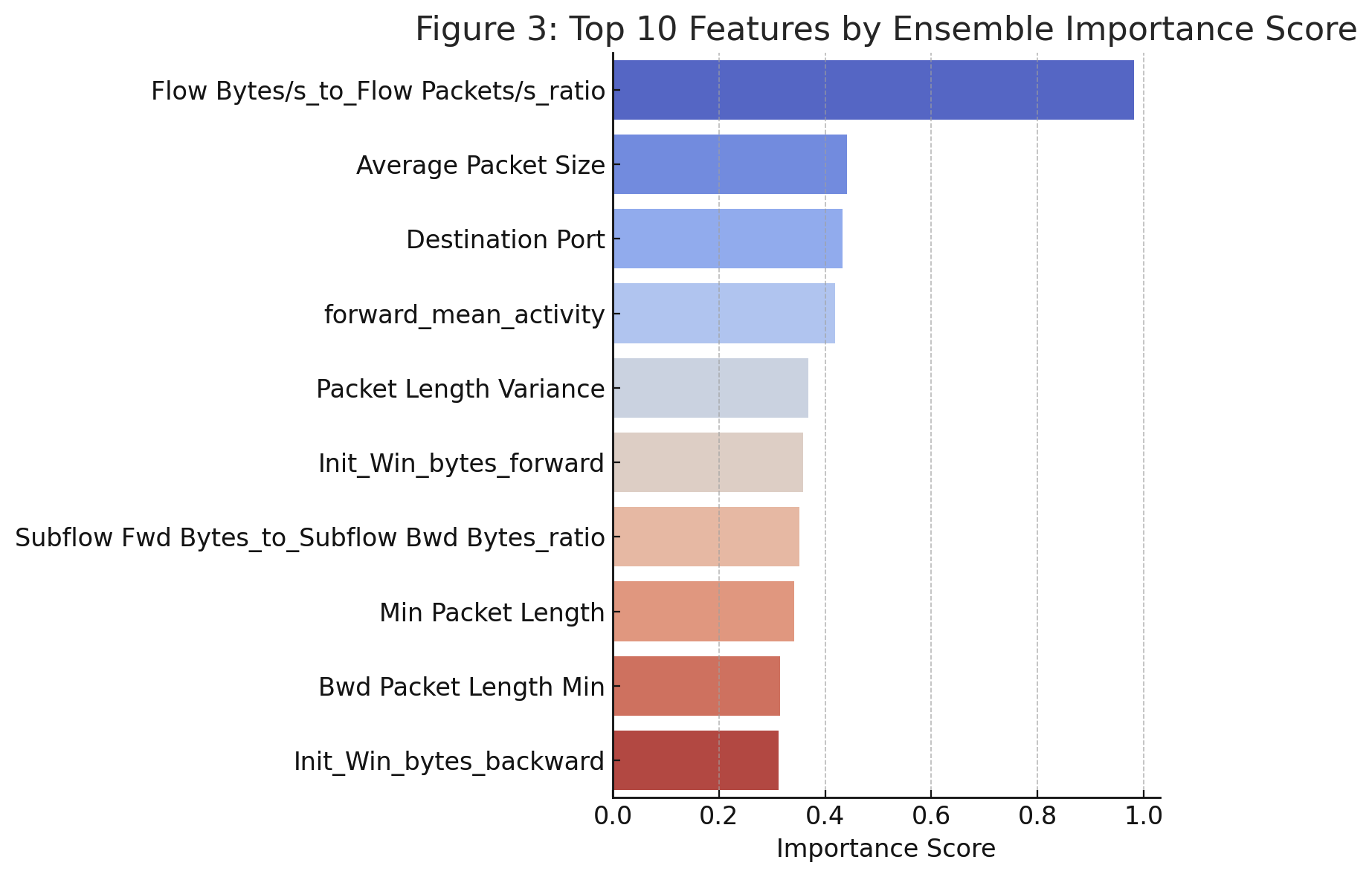
### **3.3 Feature Selection**

Applied an **ensemble approach** using:

* F-test (ANOVA)
* Mutual Information
* Random Forest importance

Top 10 features selected based on a weighted score:

1. **Flow Bytes/s\_to\_Flow Packets/s\_ratio**
2. **Average Packet Size**
3. **Destination Port**
4. **forward\_mean\_activity**
5. **Packet Length Variance**
6. **Init\_Win\_bytes\_forward**
7. **Subflow Fwd Bytes\_to\_Subflow Bwd Bytes\_ratio**
8. **Min Packet Length**
9. **Bwd Packet Length Min**
10. **Init\_Win\_bytes\_backward**

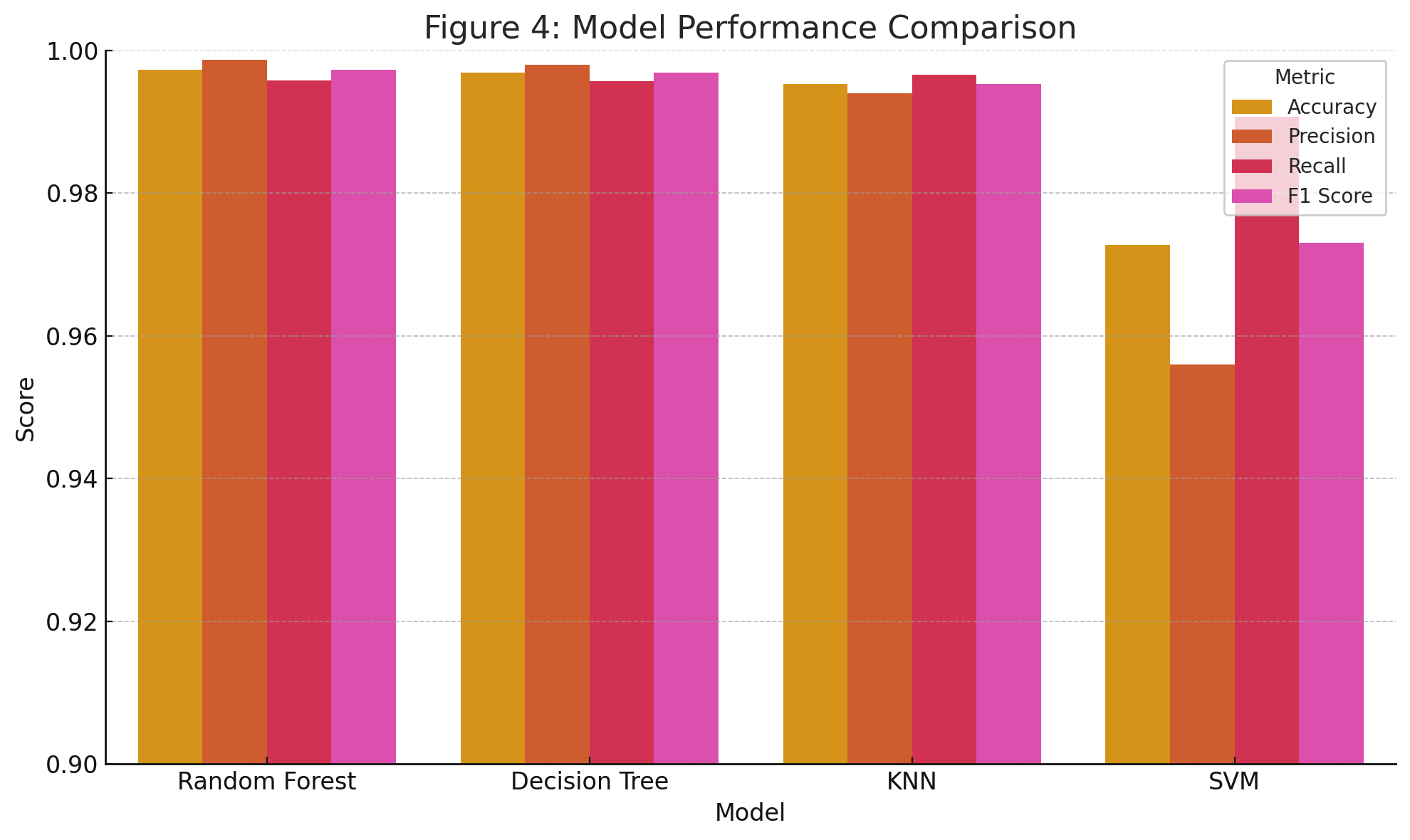


### **3.4 Standardization and Balancing**

* Used StandardScaler for normalization
* Applied **undersampling** to balance class sizes:  
  + 100,000 benign
  + 100,000 attack
* Final dataset for model training: 200,000 samples × 10 features

## **4. Model Training and Evaluation**

Tested four classifiers using 80/20 train-test split:



### **Confusion Matrices**

* Random Forest had the lowest false positives and false negatives.
* SVM, although accurate, showed a higher false positive rate.

### **4.1 Random Forest (Best Performing Model)**

* **How it works**: An ensemble of decision trees, each trained on different data subsets and features. Final prediction is made via majority voting.
* **Why it worked well**:  
  + Naturally resistant to overfitting.
  + Handles high-dimensional data and feature interactions.
  + Captures non-linear patterns in DDoS traffic.
* **Performance**:  
  + **F1 Score: 0.9973** — very high balance between precision and recall.
  + **Interpretation**: Only 25 benign samples were falsely flagged as attacks, and only 83 attacks went undetected.

### **4.2 Decision Tree**

* **How it works**: A single tree that makes decisions based on feature thresholds.
* **Pros**: Easy to interpret.
* **Cons**: Prone to overfitting on small changes in data.
* **Performance**: Nearly identical to Random Forest, but slightly lower generalization.

### **4.3 K-Nearest Neighbors (KNN)**

* **How it works**: Classifies based on majority label of K closest training samples in feature space.
* **Pros**: Intuitive, non-parametric.
* **Cons**: Computationally expensive at prediction time; requires good scaling.
* **Performance**:  
  + F1 Score: 0.9953
  + Slightly higher false positives than tree-based models.
* **Suitability**: Better for smaller datasets or low-latency offline inference.

### **4.4 Support Vector Machine (SVM)**

* **How it works**: Finds an optimal boundary (hyperplane) between two classes.
* **Pros**: Strong theoretical guarantees, especially for small or medium datasets.
* **Cons**: Slower training and less scalable on millions of rows.
* **Performance**:  
  + F1 Score: 0.9730
  + Precision was lowest (0.9560), meaning more false alarms.
* **Observation**: SVM struggled due to the dataset’s size and high dimensionality.

## **5. Final Model**

* The **Random Forest model** was selected as the final classifier.
* Model and scaler saved using joblib:  
  + ddos\_best\_model.joblib
  + ddos\_scaler.joblib

## **6. Conclusion**

This project successfully demonstrates an end-to-end machine learning pipeline for detecting DDoS attacks in network traffic. By preprocessing large-scale data, engineering meaningful features, and applying ensemble-based selection, the system achieved near-perfect performance using a Random Forest classifier. It is suitable for integration into real-world intrusion detection systems.