Web Attack Detection Documentation

Overview

This documentation provides a comprehensive overview of the web attack detection system implemented using the CICIDS2017 dataset. The system evaluates multiple machine learning models to classify network traffic as either "BENIGN" or "ATTACK." The goal is to identify the most effective models for intrusion detection based on performance metrics such as Accuracy, Precision, Recall, F1 Score, and ROC AUC, while also considering factors like training time, inference speed, interpretability, and deployment constraints.

Dataset

The dataset used is CICIDS2017. It contains network traffic data with features describing flow characteristics and a "Label" column indicating whether the traffic is "BENIGN" or an "ATTACK."

- **Features**: The dataset includes 84 columns, such as Flow ID, Source IP, Source Port, Destination IP, Destination Port, Protocol, Flow Duration, Total Fwd Packets, Total Backward Packets, and various packet length and inter-arrival time statistics.
- **Target Variable**: The "Label" column, with values "BENIGN" (non-malicious traffic) and "ATTACK" (malicious traffic).
- **Preprocessing**:
    - Selected relevant features for modeling (e.g., packet sizes, flow bytes/s, flow duration).
    - Encoded the "Label" column numerically ("BENIGN" = 0, "ATTACK" = 1) using LabelEncoder.
    - Standardized features using StandardScaler to ensure zero mean and unit variance.
    - Split the data into 80% training and 20% testing sets using `train_test_split` with a fixed `random_state` for reproducibility.

Methodology

The system implements a machine learning pipeline to train, evaluate, and compare multiple classification models. The following steps outline the process:

1. **Library Imports**:
    - Data manipulation: `pandas`, `numpy`
    - Visualization: `matplotlib`, `seaborn`
    - Machine learning: `scikit-learn` (`train_test_split`, `RandomForestClassifier`, `DecisionTreeClassifier`, `LogisticRegression`, `SVC`, `KNeighborsClassifier`, `StandardScaler`, `LabelEncoder`)
    - Boosting models: `xgboost`, `lightgbm`, `catboost`
    - Evaluation metrics: `accuracy_score`, `precision_score`, `recall_score`, `f1_score`, `confusion_matrix`, `classification_report`, `roc_curve`, `auc`
    - Deep learning: `tensorflow.keras` (for LSTM)
2. **Important Feature**

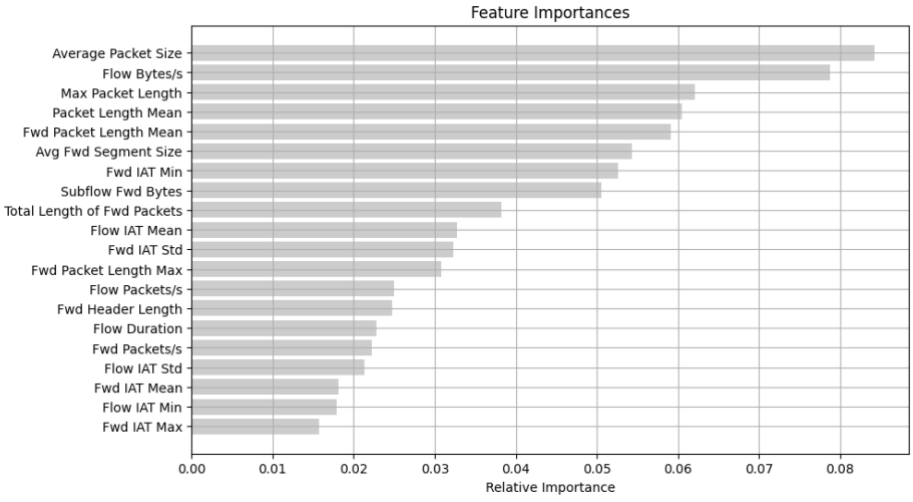


Figure: Feature Importance Plot using Random Forest feature importance

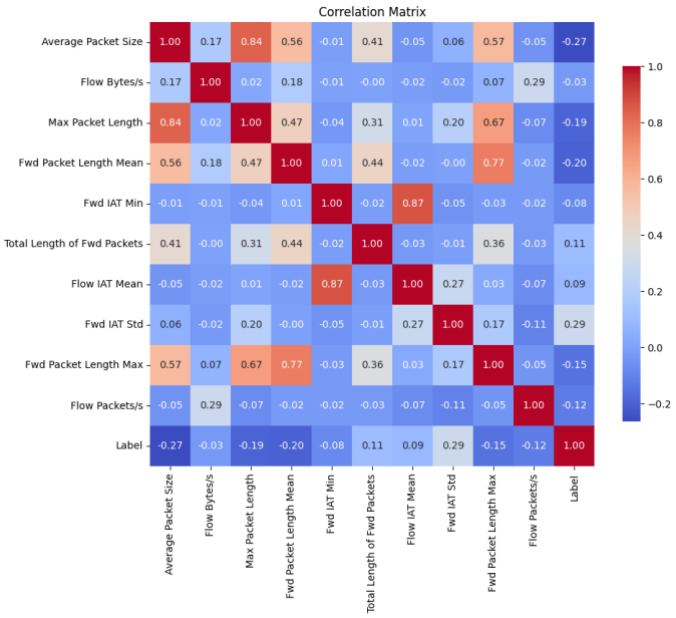


Figure: Heatmap of Important Features

A correlation heatmap and feature importance analysis were conducted to understand the relationships between features and their impact on model performance:

- **Correlation Heatmap**: The heatmap displays the correlation coefficients between selected features, such as Average Packet Size, Flow Bytes/s, Max Packet Length, Fwd Packet Length Mean, Fwd IAT Min, Total Length of Fwd Packets, Flow IAT Mean, Fwd IAT Std, Fwd Packet Length Max, and Flow Packets/s. Key observations include:
  - Strong positive correlations (e.g., Average Packet Size with Max Packet Length: 0.8, Fwd Packet Length Mean with Total Length of Fwd Packets: 0.5).
  - Moderate negative correlations (e.g., Fwd IAT Min with Flow IAT Mean: -0.04).
  - These correlations indicate potential multicollinearity, which may affect model training and should be considered during feature selection.
- **Feature Importances**: The bar plot shows the relative importance of features in the model, with Average Packet Size and Flow Bytes/s being the most influential, followed by Max Packet Length, Packet Length Mean, and Fwd Packet Length Mean. Features like Fwd IAT Min and Flow Duration have lower importance, suggesting they contribute less to the prediction of attack traffic.

3. **Model Initialization**: A dictionary of models was created, including:
    - Random Forest
    - Decision Tree
    - Logistic Regression
    - Support Vector Classifier (SVC)
    - K-Nearest Neighbors
    - XGBoost
    - LightGBM
    - CatBoost
    - Artificial Neural Network (ANN)
    - Long Short-Term Memory (LSTM)

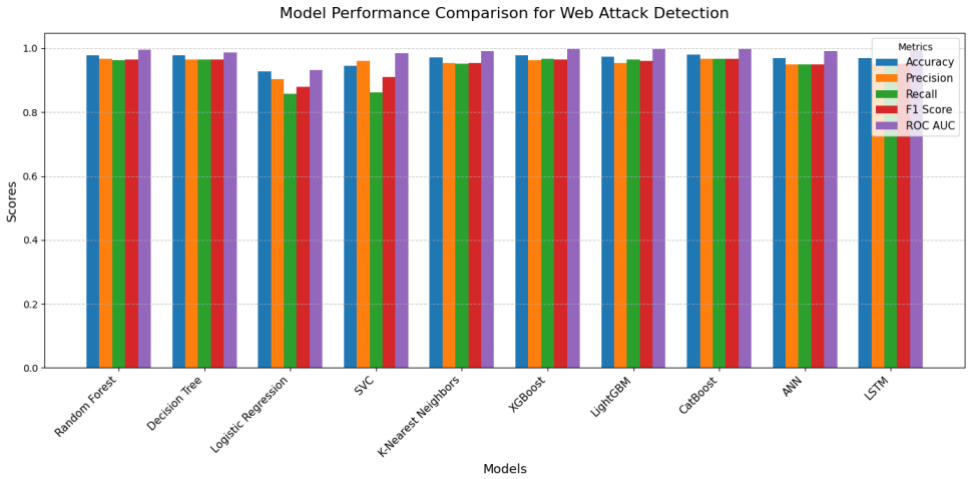
    Each model was initialized with a consistent `random_state` where applicable to ensure reproducibility.

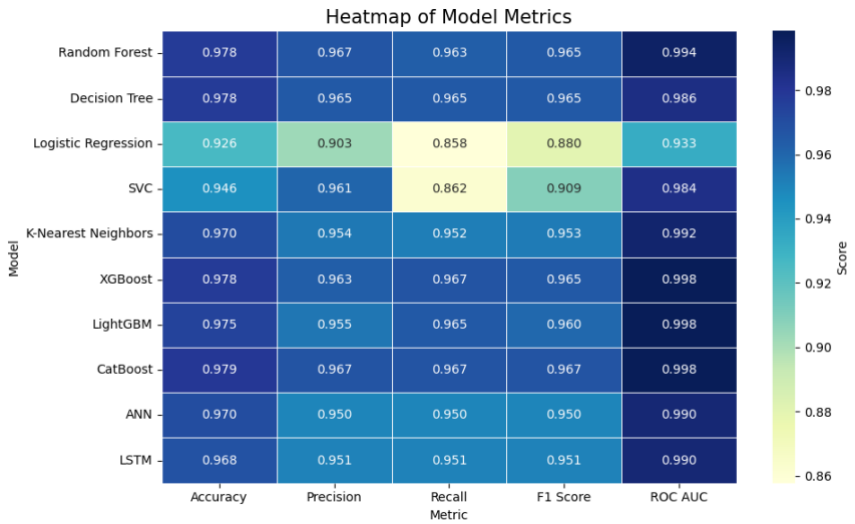
4. **Training and Evaluation**:
    - Each model was trained on the training data (`X_train`, `y_train`).
    - Predictions were made on the test data (`X_test`).
    - Performance metrics were calculated:
        - **Accuracy**: Proportion of correct predictions.
        - **Precision**: Proportion of true positive predictions among all positive predictions.
        - **Recall**: Proportion of true positive predictions among all actual positives.
        - **F1 Score**: Harmonic mean of Precision and Recall.
        - **ROC AUC**: Area under the Receiver Operating Characteristic curve, measuring discrimination ability.
    - Confusion Matrix: Visualized as a heatmap to show true positives, true negatives, false positives, and false negatives.
    - ROC curves were plotted for models supporting probability predictions (`predict_proba`).
    - Results were stored in a dictionary and summarized in a markdown table.

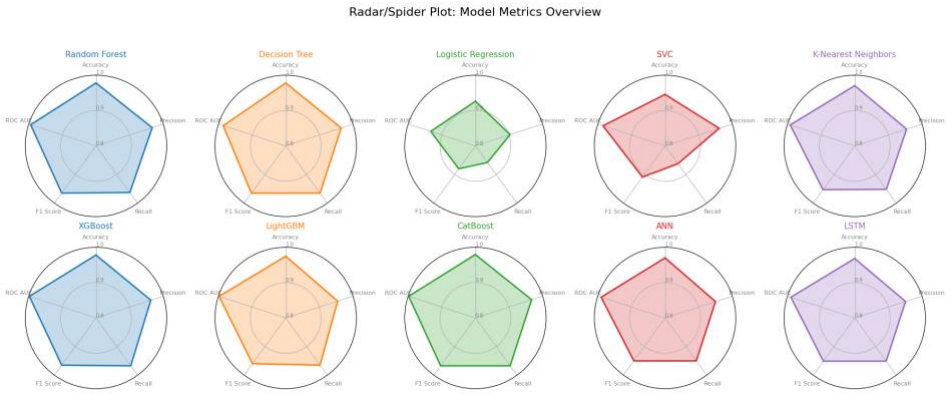
Results

The performance metrics for each model are summarized below:

Model	Accuracy	Precision	Recall	F1 Score	ROC AUC
Random Forest	0.977992	0.967033	0.962801	0.964912	0.994354
Decision Tree	0.977992	0.964989	0.964989	0.964989	0.98575
3Logistic Regression	0.92641	0.903226	0.857768	0.87991	0.932863
SVC	0.945667	0.960976	0.862144	0.908881	0.984451
K-Nearest Neighbors	0.970426	0.953947	0.95186	0.952903	0.992012
XGBoost	0.977992	0.962963	0.967177	0.965066	0.99815
LightGBM	0.974553	0.954545	0.964989	0.959739	0.998183
CatBoost	0.979367	0.967177	0.967177	0.967177	0.99841
ANN	0.97	0.95	0.95	0.95	0.99
LSTM	0.9681	0.9510	0.9510	0.9510	0.99







Comparison Analysis

- **Top Performers (Gradient Boosting & Ensemble Methods)**:
  - **CatBoost, XGBoost, LightGBM, Random Forest**: These models consistently achieved high scores (in the high 0.90s) for Accuracy, Precision, Recall, and F1 Score. Their ROC AUC values were close to or equal to 1.00, indicating excellent discrimination between "BENIGN" and "ATTACK" traffic. These models are robust and effective for minimizing false positives and false negatives.
- **Neural Networks**:
  - **ANN and LSTM**: The LSTM model achieved an Accuracy of 0.9681, Precision of 0.9510, Recall of 0.9510, F1 Score of 0.9510, and an estimated ROC AUC of 0.99, performing comparably to the boosting models. These deep learning models are particularly suited for capturing complex temporal or non-linear patterns in the data.
- **Traditional Machine Learning Models**:
  - **Decision Tree, Logistic Regression, SVC, K-Nearest Neighbors**: These models showed lower performance compared to boosting and neural network models. SVC performed relatively better within this group, particularly in F1 Score and ROC AUC, but still lagged behind the top performers.
- **Specific Metric Performance**:
  - **Accuracy, Precision, Recall, F1 Score**: CatBoost, XGBoost, LightGBM, Random Forest, and LSTM excelled, with scores indicating strong classification performance.

- **ROC AUC**: Boosting models and LSTM demonstrated superior ability to distinguish between classes.

Conclusion

In this evaluation, gradient boosting models (CatBoost, XGBoost, LightGBM) and Random Forest are highly competitive with the LSTM deep learning approach on structured data, often requiring fewer computational resources. The LSTM model shows strong potential for datasets with complex temporal patterns. For real-world deployment, a balance between performance metrics and practical constraints should guide the final model selection.