19CSE304

FOUNDATIONS OF DATA SCIENCE

XSS VULNERABILITY DETECTION

Gadha Saji Menon

AM.EN.U4CSE22123

CONTENTS

1 Introduction

2 Data Generation

3 Training Data

4 Preprocessing

5 Model Training and Evaluation

6 Interactive Dashboard

7 Visualization

8 Inference and Conclusion

PROBLEM STATEMENT

Objective:

- Build and evaluate machine learning models using data processing techniques that can accurately classify whether a feature set represents an XSS vulnerability or not.

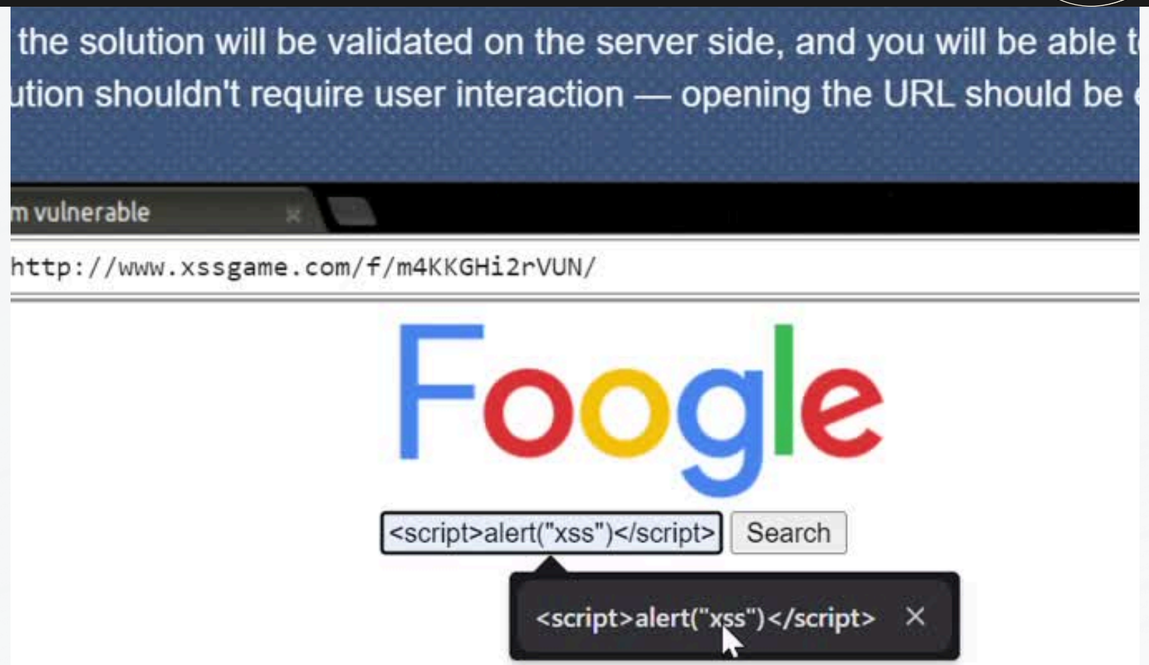
Approach:

- Generate a synthetic dataset that resembles real-world xss datasets.
- Train the raw data using **random forest**, preprocess the data and then re-train with the processed data.
- Generate an interactive dashboard for the model.
- Analyze the metrics and draw conclusions.

WHAT IS XSS?

Definition:

- Cross-Site Scripting (XSS) is a web vulnerability where attackers inject malicious scripts into webpages, executed in users' browsers.



Some famous attacks:

1. 2005 SAMY worm
2. British airways 2018 breach

DATA GENERATION

DATA GENERATION

Dataset generation

```python
import pandas as pd
import numpy as np
import random
import string

# Function to generate random strings
def random_string(length):
    return ''.join(random.choices(string.ascii_letters + string.digits + string.punctuation, k=length))

# Dataset generation
np.random.seed(42)
n_samples = 2000

data = {
    'Input': [random_string(random.randint(10, 100)) for _ in range(n_samples)],  # Random input strings
    'Length': [random.randint(10, 100) for _ in range(n_samples)],                # Length of input
    'ScriptTagsCount': [random.randint(0, 5) for _ in range(n_samples)],          # Number of <script> tags
    'SpecialCharsCount': [random.randint(0, 20) for _ in range(n_samples)],       # Special characters count
    'HTMLTagCount': [random.randint(0, 10) for _ in range(n_samples)],            # Count of HTML tags
    'HasURL': np.random.choice([0, 1], n_samples),                                # Presence of a URL
    'HasIP': np.random.choice([0, 1], n_samples),                                 # Presence of an IP address
    'EncodedCharsCount': [random.randint(0, 10) for _ in range(n_samples)],       # Encoded characters count
    'WordCount': [random.randint(1, 50) for _ in range(n_samples)],               # Word count in the input
    'IsAllCaps': np.random.choice([0, 1], n_samples),                             # All uppercase words flag
    'AvgWordLength': [random.uniform(3, 10) for _ in range(n_samples)],           # Average word length
    'NumericCharCount': [random.randint(0, 15) for _ in range(n_samples)],        # Numeric character count
    'StartsWithSpecialChar': np.random.choice([0, 1], n_samples),                 # Starts with a special character
    'ContainsJSFunction': np.random.choice([0, 1], n_samples),                    # Contains JavaScript function flag
    'ContainsSQLKeywords': np.random.choice([0, 1], n_samples),                   # Contains SQL keywords flag
    'Label': np.random.choice([0, 1], n_samples, p=[0.85, 0.15])                  # 15% malicious inputs
}

# Convert to DataFrame
df = pd.DataFrame(data)

# Add inconsistencies
# 1. Missing values
df.loc[np.random.choice(df.index, 50), 'Length'] = np.nan

# 2. Duplicates
df = pd.concat([df, df.iloc[:10]])

# 3. Outliers
df.loc[np.random.choice(df.index, 20), 'SpecialCharsCount'] = 100

# 4. Noisy data
df.loc[np.random.choice(df.index, 30), 'Input'] = random_string(5)  # Extremely short or nonsensical strings

# Save to CSV
file_path = "/content/drive/MyDrive/CSE B/S5/synthetic_xss.csv"
df.to_csv(file_path, index=False)

file_path
```

```
RangeIndex: 2010 entries, 0 to 2009
Data columns (total 16 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Input                  2010 non-null   object
 1   Length                 1960 non-null   float64
 2   ScriptTagsCount        2010 non-null   int64
 3   SpecialCharsCount      2010 non-null   int64
 4   HTMLTagCount           2010 non-null   int64
 5   HasURL                 2010 non-null   int64
 6   HasIP                  2010 non-null   int64
 7   EncodedCharsCount      2010 non-null   int64
 8   WordCount              2010 non-null   int64
 9   IsAllCaps              2010 non-null   int64
 10  AvgWordLength          2010 non-null   float64
 11  NumericCharCount       2010 non-null   int64
 12  StartsWithSpecialChar  2010 non-null   int64
 13  ContainsJSFunction     2010 non-null   int64
 14  ContainsSQLKeywords    2010 non-null   int64
 15  Label                  2010 non-null   int64
dtypes: float64(2), int64(13), object(1)
```

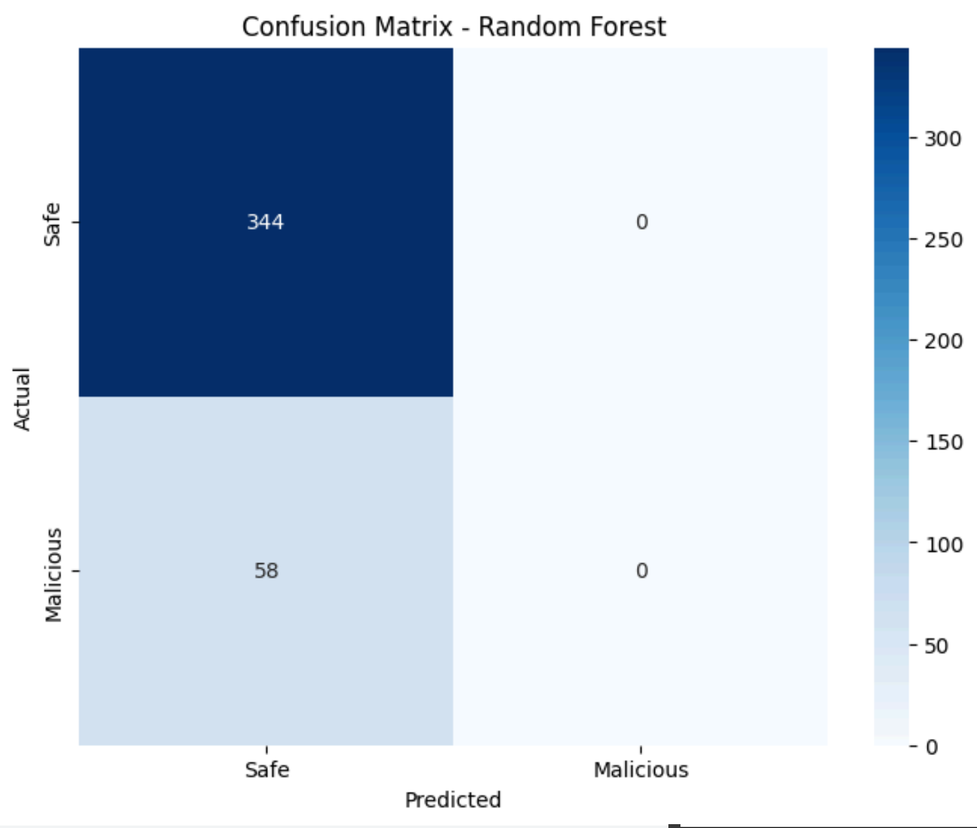
```
[6] df.head()
```

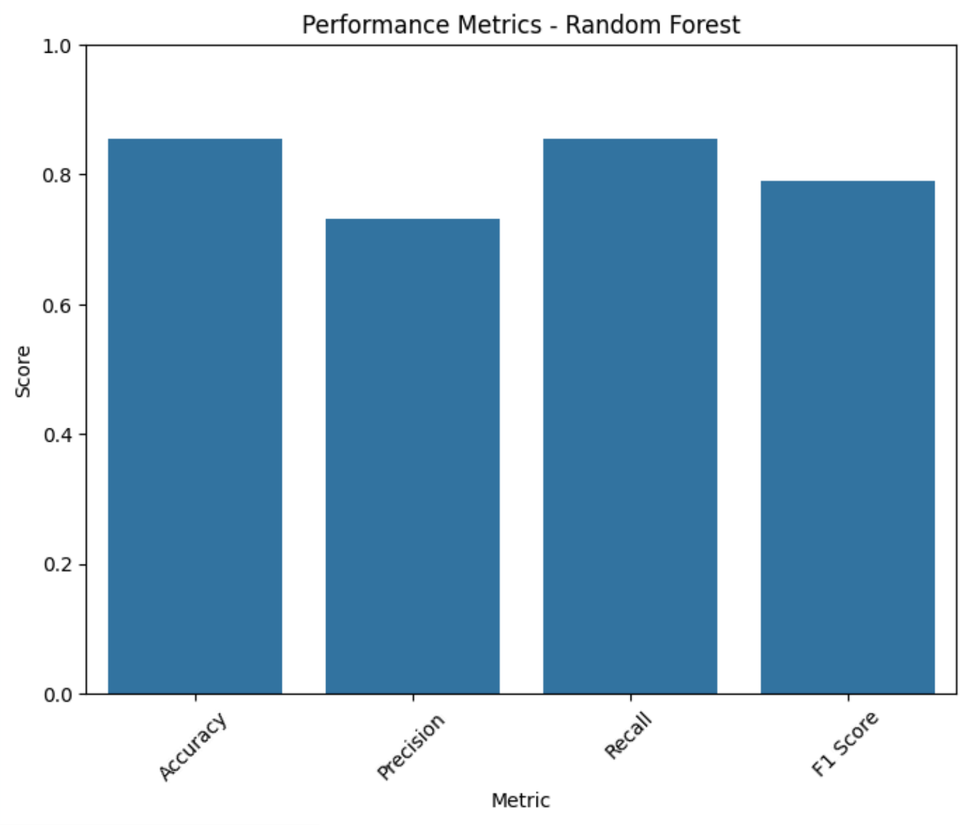
	Input	Length	ScriptTagsCount	SpecialCharsCount	HTMLTagCount	HasURL	HasIP	EncodedCharsCount	WordCount	IsAllCaps
0	wiK&f)9_~DKeQlgpxJ?7Il_0LFijh:n^xkDjkyc\>p	67.0	1	2	8	0	0	7	26	1
1	Sd>5H:Wl@*PG}-gn{rJorp9jw2X	49.0	0	7	3	1	1	3	28	0
2	>~yasEti<U;cJSMT1a,Gjlmo&\~Dx-d<1p79GEKCY2wBMD...	92.0	4	16	2	0	1	5	26	0
3	<63Xte1`d#8x1?i5:{01/7QI++BtHC$!?Z-W	91.0	2	4	7	0	1	9	50	0
4	R@Z1Maw(kF*+D}[?e3uY~IGlNUF}LKE+EC&)rhbj2mh`oB...	71.0	4	7	8	0	1	7	11	1

AvgWordLength	NumericCharCount	StartsWithSpecialChar	ContainsJSFunction	ContainsSQLKeywords	Label
5.421662	6	0	1	0	0
9.734721	12	0	0	1	0
8.714552	4	0	1	0	1
6.482765	6	1	0	1	0
6.733061	15	0	0	1	0

TRAINING RAW DATA

TRAINING DATA USING RANDOM FOREST





	Metric	Score
0	Accuracy	0.855721
1	Precision	0.732259
2	Recall	0.855721
3	F1 Score	0.789191

PREPROCESSING

```python
# 1. Print the amount of missing values
missing_values = df.isnull().sum()
print("Missing Values:")
print(missing_values)

# 2. Print the number of duplicate rows
duplicate_rows = df.duplicated().sum()
print("\nNumber of Duplicate Rows:", duplicate_rows)

# 3. Boxplot to show outliers for numerical columns
plt.figure(figsize=(12, 6))
sns.boxplot(data=df[['Length', 'ScriptTagsCount', 'SpecialCharsCount', 'HTMLTagCount']])
plt.title("Boxplot to Show Outliers")
plt.show()

# 4. Display the rows with noise (e.g., short or nonsensical strings)
print("\nExamples of Noisy Data (short Input strings):")
noisy_data = df[df['Input'].str.len() < 10]
print(noisy_data.head())
```

Missing Values

```
Missing Values:
Input                      0
Length                    50
ScriptTagsCount            0
SpecialCharsCount          0
HTMLTagCount               0
HasURL                     0
HasIP                      0
EncodedCharsCount          0
WordCount                  0
IsAllCaps                  0
AvgWordLength              0
NumericCharCount           0
StartsWithSpecialChar      0
ContainsJSFunction         0
ContainsSQLKeywords        0
Label                      0
dtype: int64
```

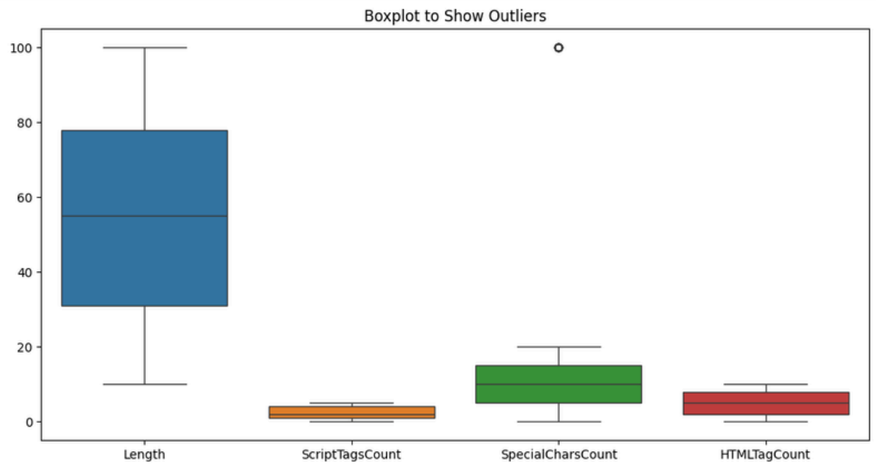
```
Missing Values:
Input                      0
Length                     0
ScriptTagsCount            0
SpecialCharsCount          0
HTMLTagCount               0
HasIP                      0
EncodedCharsCount          0
WordCount                  0
IsAllCaps                  0
AvgWordLength              0
NumericCharCount           0
StartsWithSpecialChar      0
ContainsJSFunction         0
ContainsSQLKeywords        0
Label                      0
HasURL_0                   0
HasURL_1                   0
dtype: int64
```

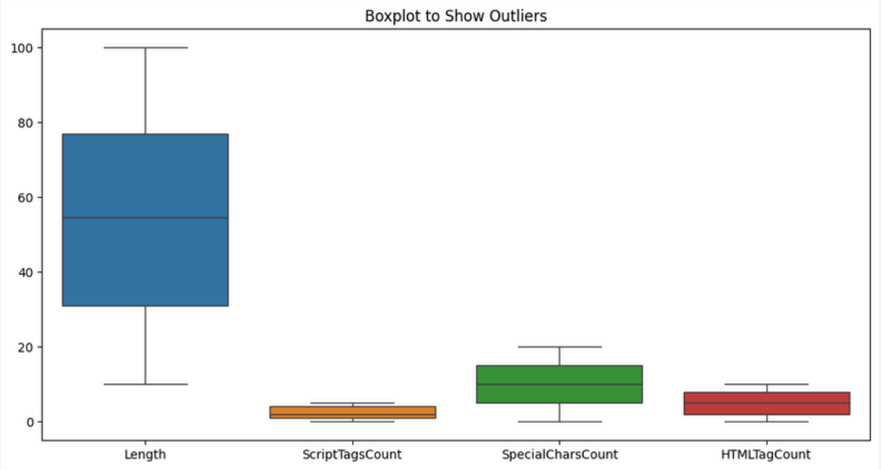
Duplicates

```
Number of Duplicate Rows: 10
```

```
Number of Duplicate Rows: 0
```

Remove outliers using IQR



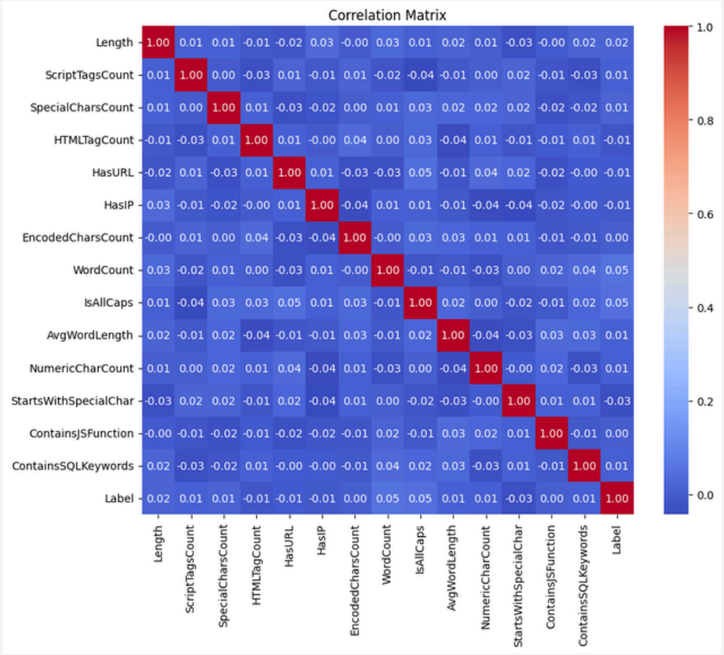


Remove noisy data

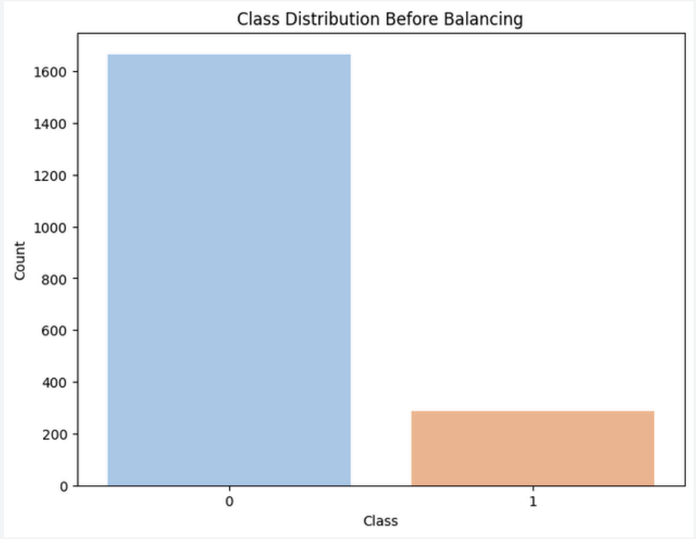
```
Examples of Noisy Data (short Input strings):
     Input  Length  ScriptTagsCount  SpecialCharsCount  HTMLTagCount  HasURL
11   ds`L|    90.0                4                  5            10       0
25   ds`L|    41.0                2                  0             1       1
61   ds`L|    40.0                3                 20             0       0
81   ds`L|   100.0                5                 18             9       1
228  ds`L|    36.0                0                 11            10       0

     HasIP  EncodedCharsCount  WordCount  IsAllCaps  AvgWordLength  \
11       0                  5          4          0       9.053921
25       1                  3         24          0       7.094186
61       1                 10         28          0       5.237272
81       1                  6         33          1       7.158162
228      0                  5         28          1       5.340761

     NumericCharCount  StartsWithSpecialChar  ContainsJSFunction  \
11                  1                      0                   0
25                  7                      0                   1
61                 10                      0                   1
81                  1                      1                   0
228                 8                      0                   1
```

```
Examples of Noisy Data (short Input strings):
Empty DataFrame
```

Correlation Matrix



Class Distribution



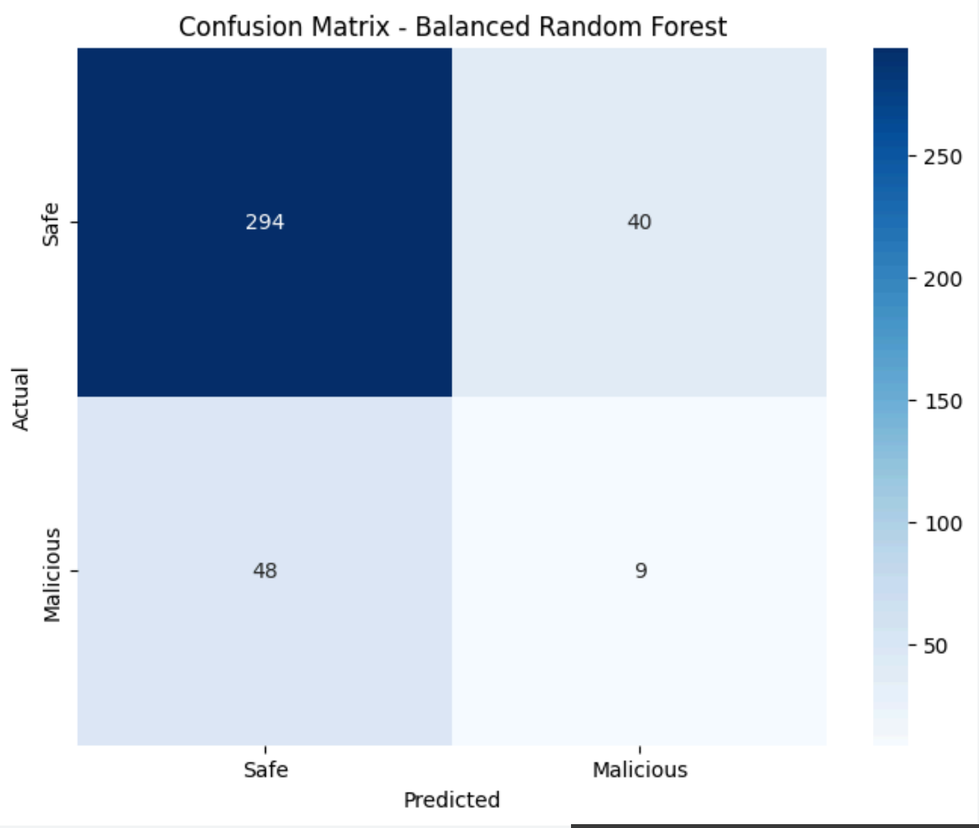

```
# Apply SMOTE to balance the training data
smote = SMOTE(random_state=42)
X_train_balanced, y_train_balanced = smote.fit_resample(X_train, y_train)
```

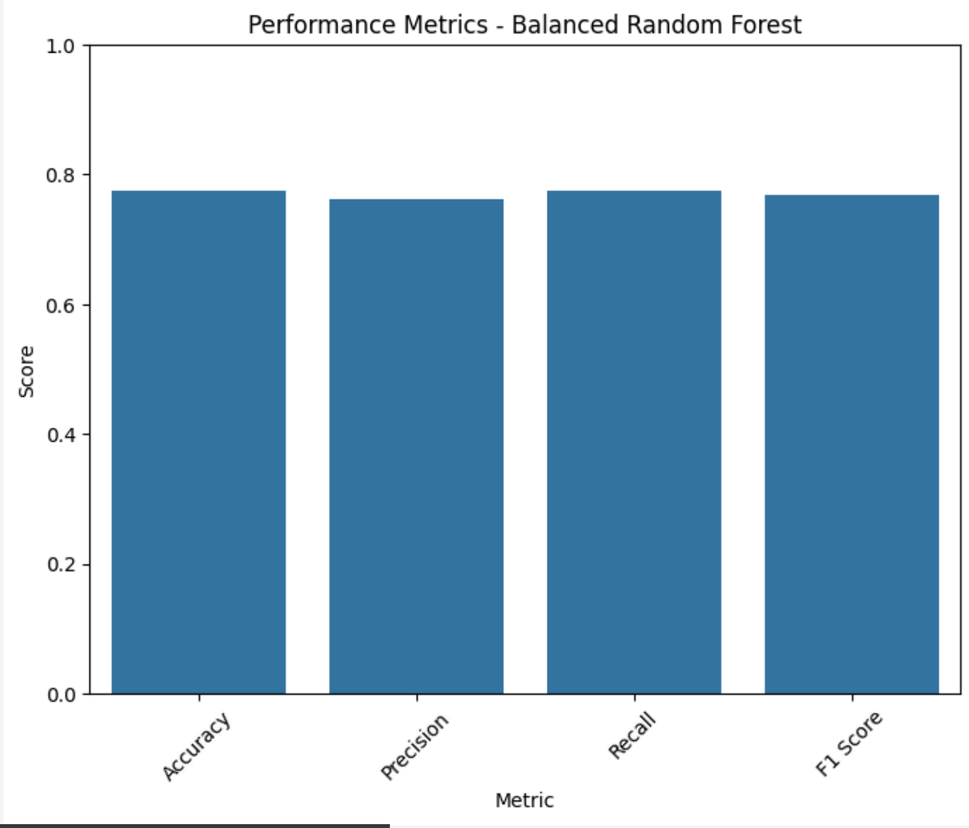
MODEL TRAINING ON PROCESSED DATA

Training and evaluation on processed data

```
# Train the Random Forest model
model = RandomForestClassifier(random_state=42)
model.fit(X_train_balanced, y_train_balanced)

# Make predictions
y_pred = model.predict(X_test)

# Calculate metrics
accuracy = accuracy_score(y_test, y_pred)
classification_report_dict = classification_report(y_test, y_pred, output_dict=True)
conf_matrix = confusion_matrix(y_test, y_pred)
```

TRAINING DATA USING RANDOM FOREST





	Metric	Score
0	Accuracy	0.774936
1	Precision	0.761105
2	Recall	0.774936
3	F1 Score	0.767775

INTERACTIVE DASHBOARD

DASHBOARD


d03e-34-135-56-21.ngrok-free.app
VPN
Update
Visualization Options
Select visualizations to display:
Choose an option
Data Distribution
Confusion Matrix
Performance Metrics
PR Curve
Feature Importance
Interactive XSS Detection Dashboard
Developed by Gadha

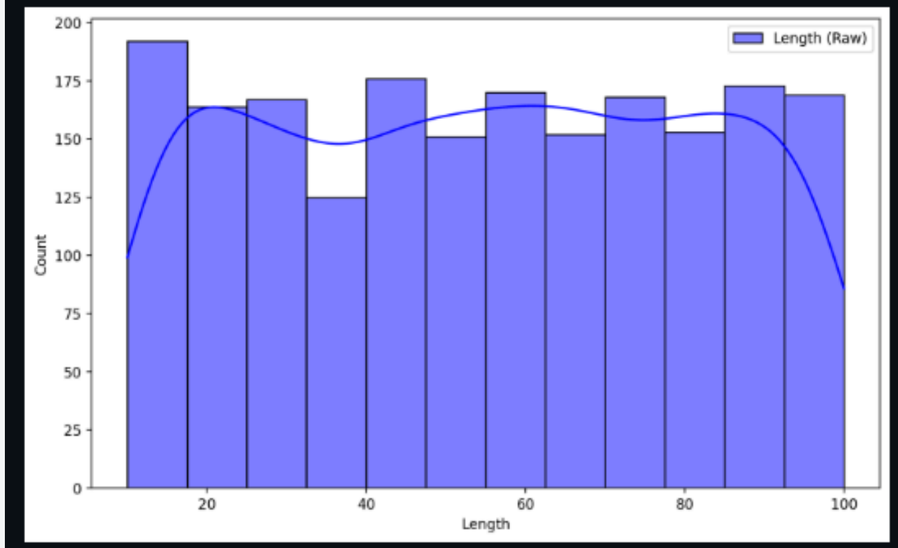
DATA DISTRIBUTION ACROSS CLASSES

Data Distribution

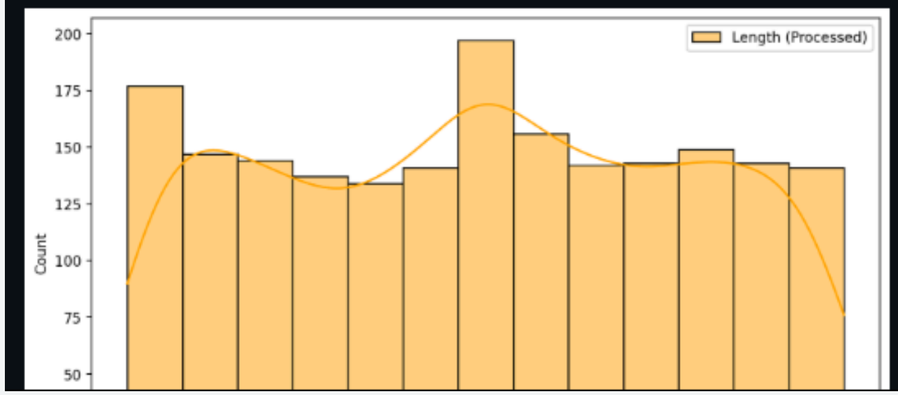
Select Feature for Distribution:

Length

Raw Data Distribution - Length



Processed Data Distribution - Length



Select Feature for Distribution:

Length

EncodedCharsCount

WordCount

IsAllCaps

AvgWordLength

NumericCharCount

StartsWithSpecialChar

ContainsJSFunction

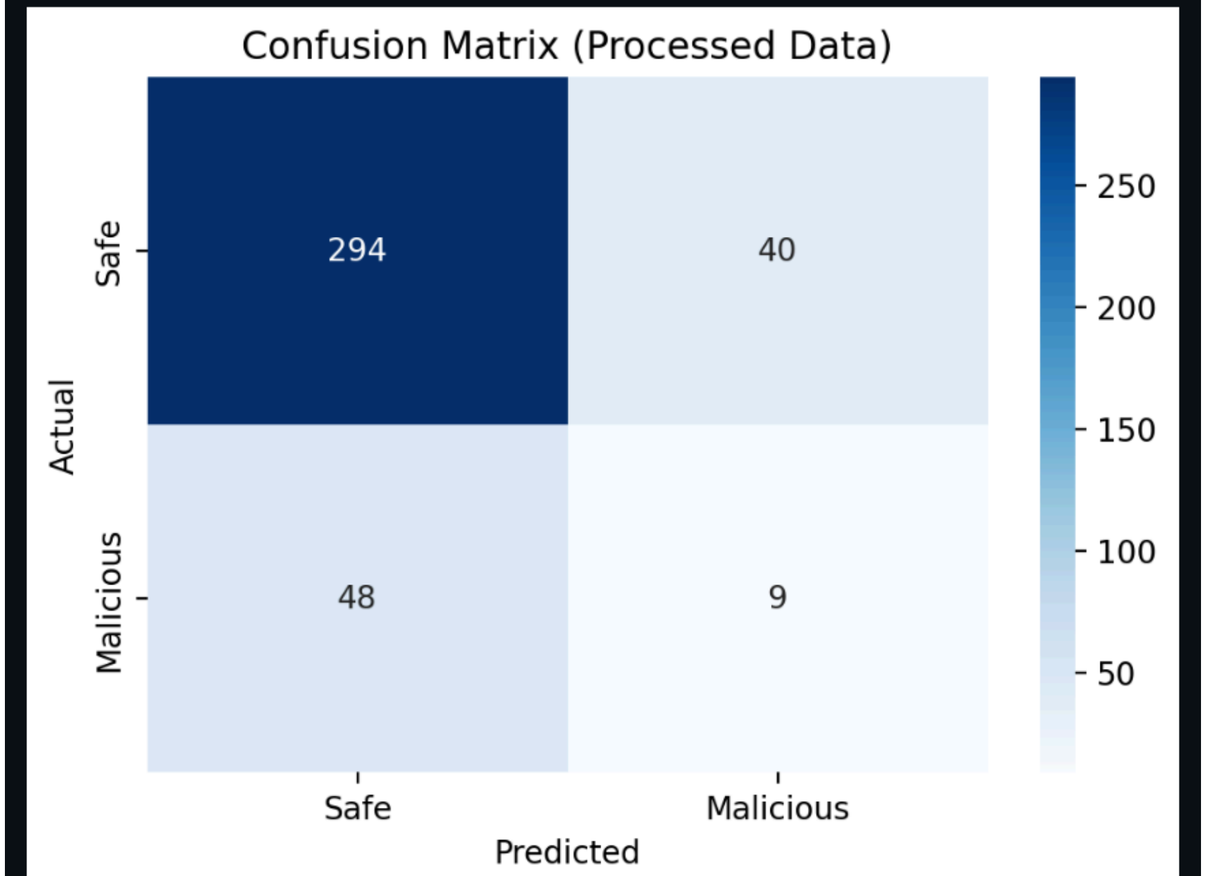
ContainsSQLKeywords

CONFUSION MATRIX

Confusion Matrix

Select Data Type for Confusion Matrix:

- ○ Raw Data
- ◉ Processed Data



PERFORMANCE METRICS

Performance Metrics

Select Data Type for Metrics:

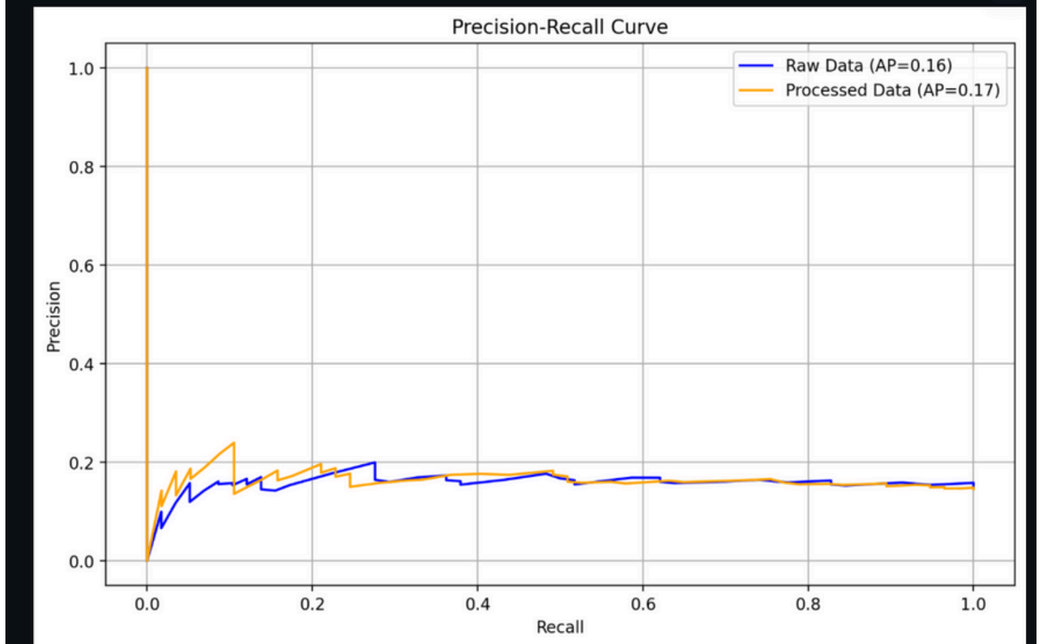
- ○ Raw Data
- ◉ Processed Data


```
{
  "0" : {
    "precision" : 0.8596491228070176
    "recall" : 0.8802395209580839
    "f1-score" : 0.8698224852071006
    "support" : 334
  }
  "1" : {
    "precision" : 0.1836734693877551
    "recall" : 0.15789473684210525
    "f1-score" : 0.16981132075471697
    "support" : 57
  }
  "accuracy" : 0.7749360613810742
  "macro avg" : {
    "precision" : 0.5216612960973863
    "recall" : 0.5190671289000945
    "f1-score" : 0.5198169029809088
    "support" : 391
  }
  "weighted avg" : {
    "precision" : 0.7611053574747977
    "recall" : 0.7749360613810742
    "f1-score" : 0.7677748218470345
    "support" : 391
  }
}
```

PR CURVE

Dashboard

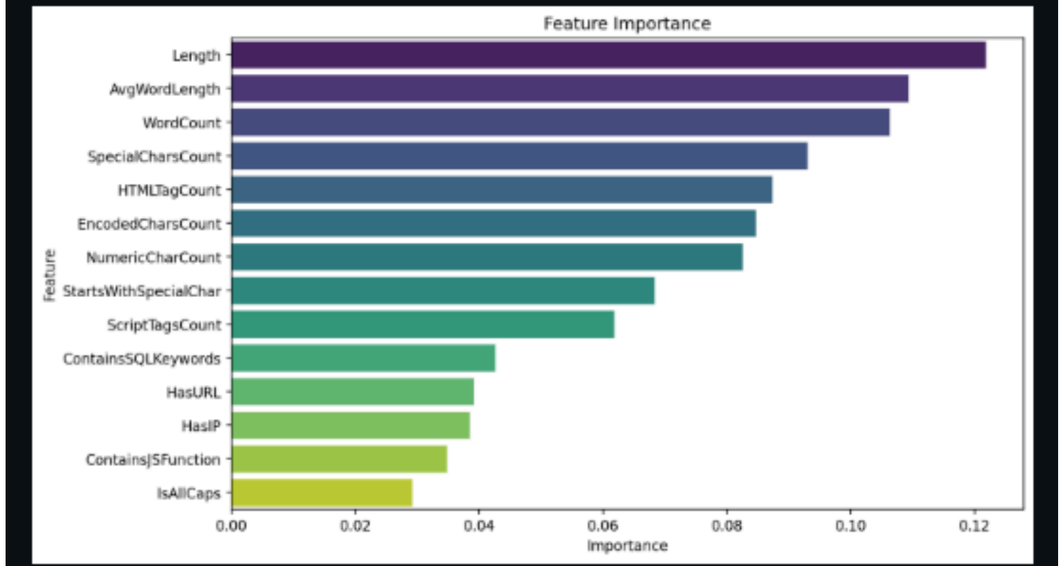
Precision-Recall Curve



FEATURE IMPORTANCE

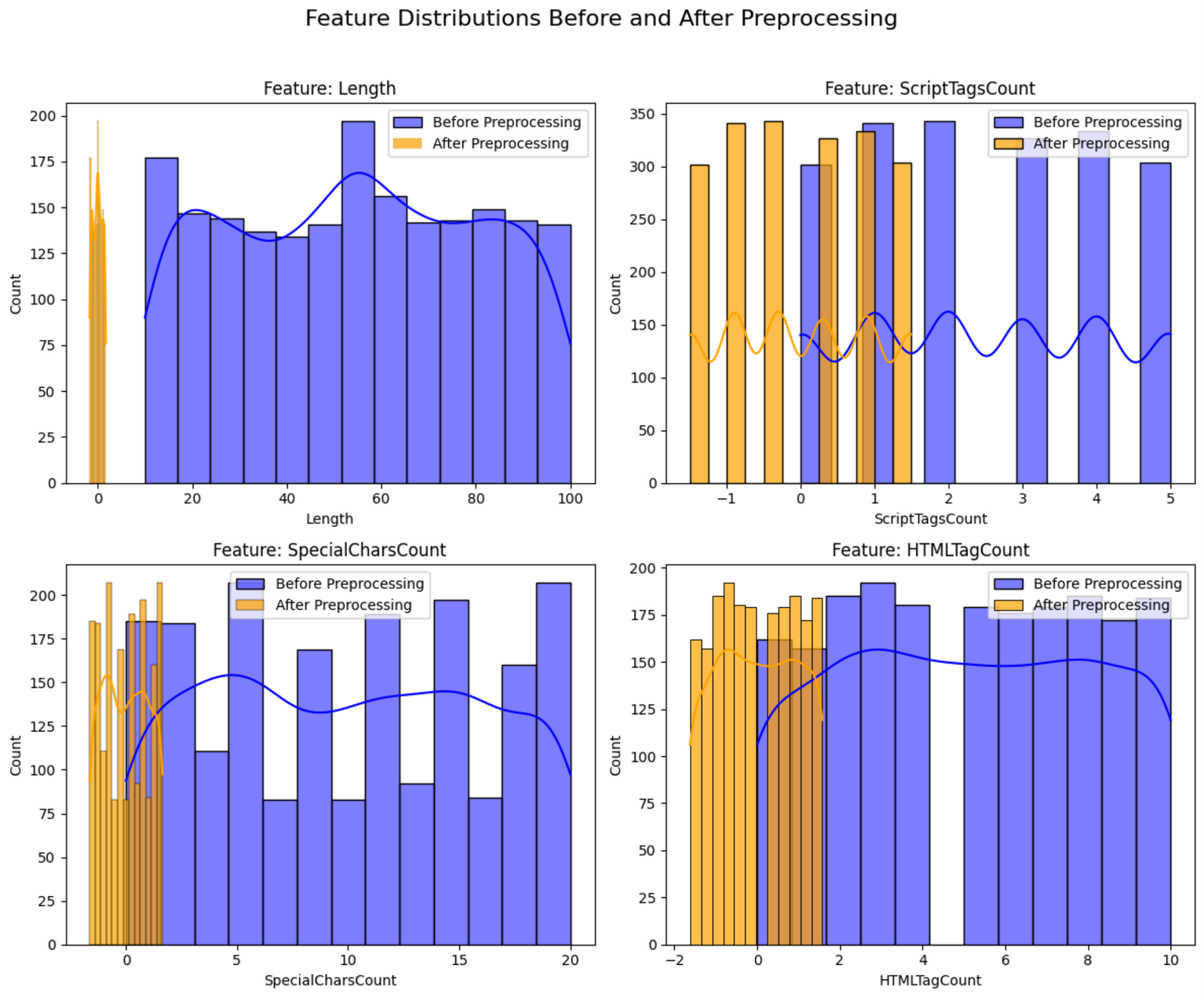
Feature Importance (Random Forest)

	Feature	Importance
0	Length	0.1218
9	AvgWordLength	0.1094
7	WordCount	0.1063
2	SpecialCharsCount	0.0931
3	HTMLTagCount	0.0874
6	EncodedCharsCount	0.0847
10	NumericCharCount	0.0826
11	StartsWithSpecialChar	0.0684
1	ScriptTagsCount	0.0619
13	ContainsSQLKeywords	0.0426

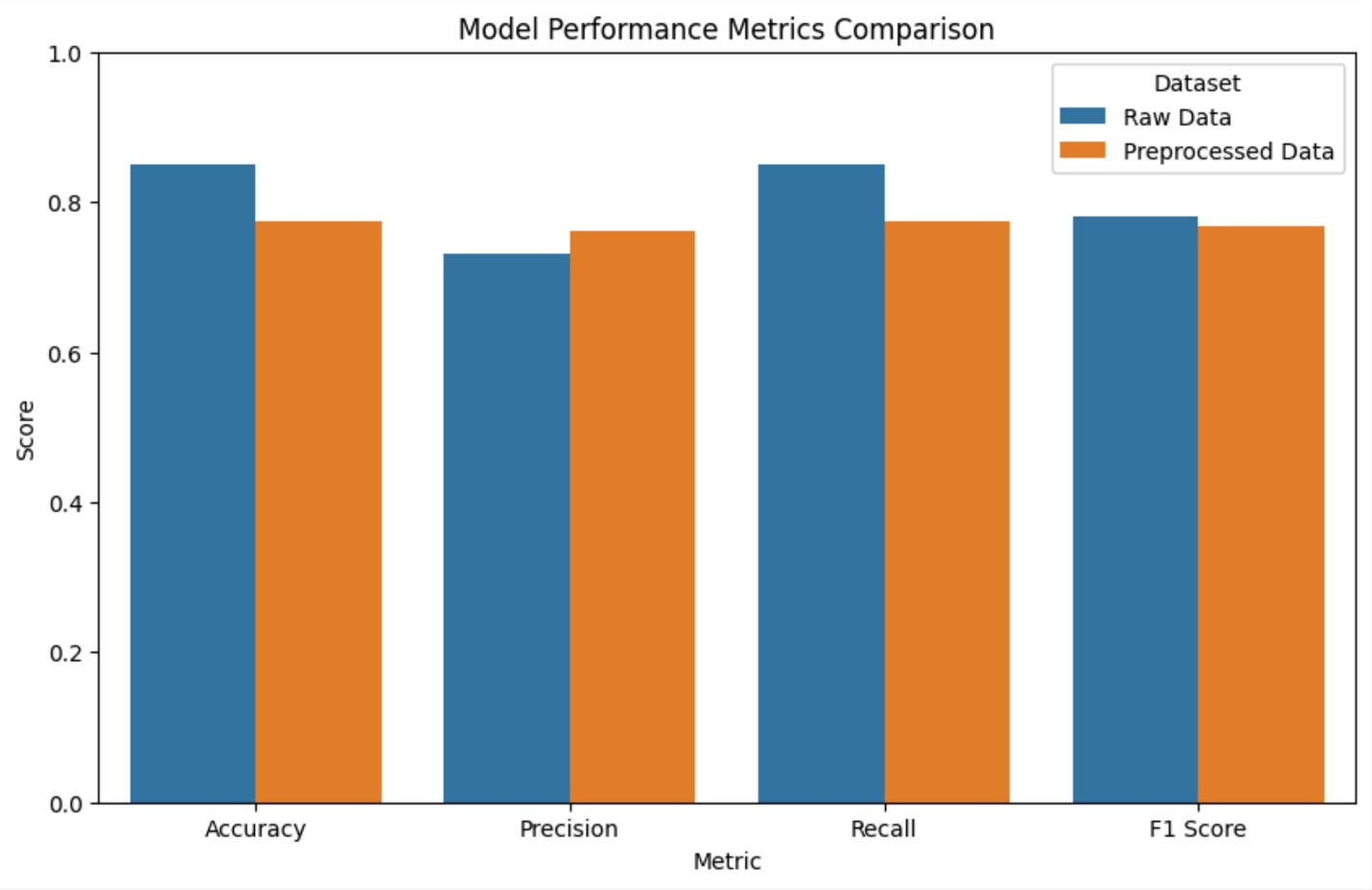


VISUALIZATION

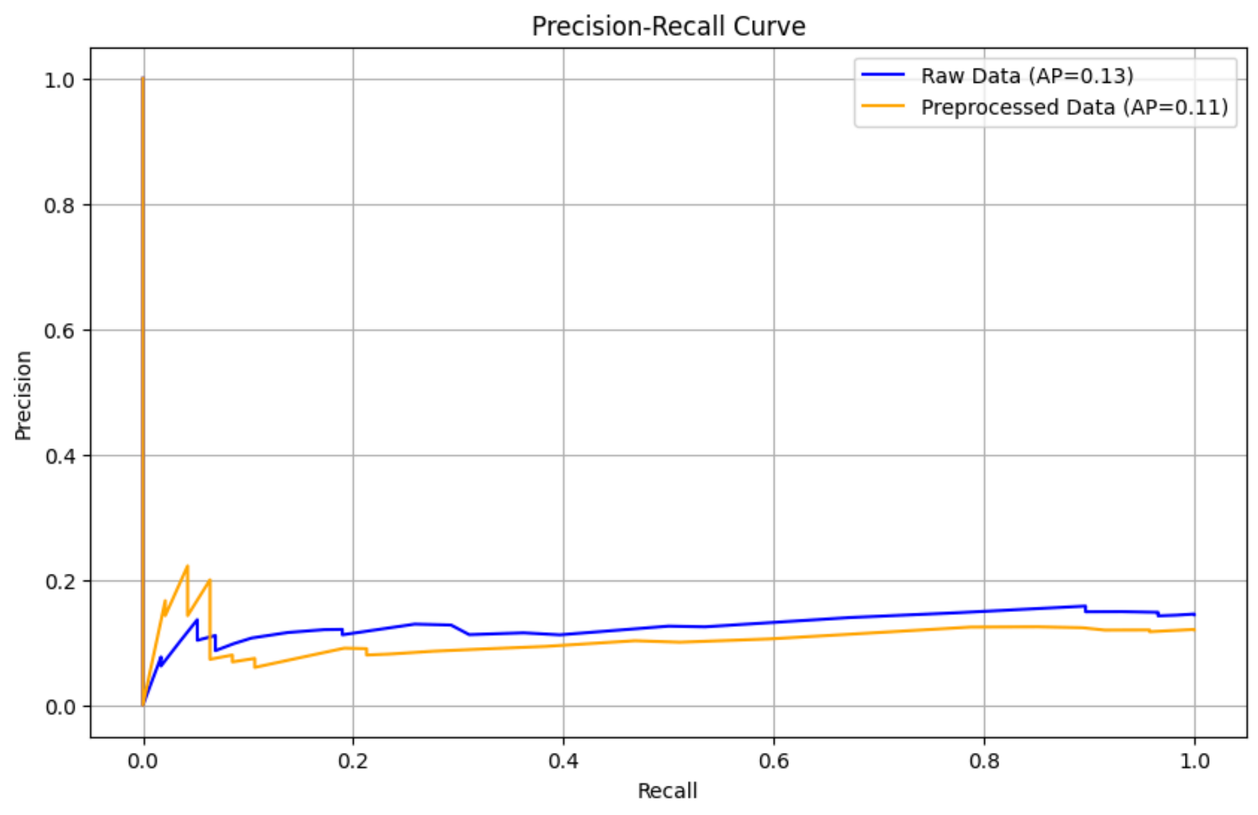
Feature Distribution



Metrics Comparison



PR CURVE


Precision-Recall Curve
Raw Data (AP=0.13)
Preprocessed Data (AP=0.11)
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

INFERENCE

KEY INSIGHTS

Preprocessing Impact

- Improved recall for malicious inputs after balancing the dataset using SMOTE.
- Reduced noise and outliers for better feature interpretation.
- Slight accuracy trade-off due to balanced evaluation across classes.

Performance

- Raw Data: High accuracy (85.6%) but failed to detect malicious inputs due to class imbalance.
- Processed Data: Significant improvement in detecting malicious inputs with better recall.

Model Evaluation:

- Post-preprocessing, the model showed improved handling of minority class (malicious inputs).

Top Features:

- Length, SpecialCharsCount, HTMLTagCount, HasURL.

END