In [ ]:
```
import pandas as pd
import numpy as np
import glob
import os
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatri
```

In [ ]:
```
# Load and merge dataset files
data_dir = 'CICIDS2017'

all_files = glob.glob(os.path.join(data_dir, '*.csv'))
df_list = []
for file in all_files:
    try:
        df = pd.read_csv(file, encoding="ISO-8859-1", encoding_errors="replace", lo
        df_list.append(df)
    except Exception as e:
        print(f"Error reading {file}: {e}")  # Debugging info

df = pd.concat(df_list, ignore_index=True)
```

In [ ]:
```
# Handle missing values
df.replace([np.inf, -np.inf], np.nan, inplace=True)
df.dropna(inplace=True)
df.reset_index(drop=True, inplace=True)
```

In [ ]:
```
# Checker
print(df.head())  # Show first few rows
```

```
                                   Flow ID       Source IP   Source Port  \
0  192.168.10.5-104.16.207.165-54865-443-6  104.16.207.165         443.0
1    192.168.10.5-104.16.28.216-55054-80-6   104.16.28.216          80.0
2    192.168.10.5-104.16.28.216-55055-80-6   104.16.28.216          80.0
3  192.168.10.16-104.17.241.25-46236-443-6   104.17.241.25         443.0
4  192.168.10.5-104.19.196.102-54863-443-6  104.19.196.102         443.0

   Destination IP   Destination Port   Protocol      Timestamp  \
0    192.168.10.5            54865.0        6.0  7/7/2017 3:30
1    192.168.10.5            55054.0        6.0  7/7/2017 3:30
2    192.168.10.5            55055.0        6.0  7/7/2017 3:30
3   192.168.10.16            46236.0        6.0  7/7/2017 3:30
4    192.168.10.5            54863.0        6.0  7/7/2017 3:30

    Flow Duration   Total Fwd Packets   Total Backward Packets  ...  \
0             3.0                 2.0                     0.0  ...
1           109.0                 1.0                     1.0  ...
2            52.0                 1.0                     1.0  ...
3            34.0                 1.0                     1.0  ...
4             3.0                 2.0                     0.0  ...

    min_seg_size_forward  Active Mean   Active Std   Active Max   Active Min  \
0                   20.0          0.0          0.0          0.0          0.0
1                   20.0          0.0          0.0          0.0          0.0
2                   20.0          0.0          0.0          0.0          0.0
3                   20.0          0.0          0.0          0.0          0.0
4                   20.0          0.0          0.0          0.0          0.0

   Idle Mean   Idle Std   Idle Max   Idle Min   Label
0        0.0        0.0        0.0        0.0  BENIGN
1        0.0        0.0        0.0        0.0  BENIGN
2        0.0        0.0        0.0        0.0  BENIGN
3        0.0        0.0        0.0        0.0  BENIGN
4        0.0        0.0        0.0        0.0  BENIGN

[5 rows x 85 columns]
```

In [ ]:
```python
# Strip spaces from all column names
df.columns = df.columns.str.strip()
```

In [ ]:
```python
# Encode categorical labels
label_encoder = LabelEncoder()
df['Label'] = label_encoder.fit_transform(df['Label'])
label_mapping = dict(zip(label_encoder.classes_, label_encoder.transform(label_enco
```

In [ ]:
```python
# Convert both keys and values to standard Python types
label_mapping_fixed = {str(key): int(value) for key, value in label_mapping.items()

# Save as JSON
import json
with open("label_mapping.json", "w") as f:
    json.dump(label_mapping_fixed, f)

print("Label encoding saved as label_mapping.json")
```

```
Label encoding saved as label_mapping.json
```

In [ ]:
```
# Feature selection
drop_columns = ['Flow ID', 'Source IP', 'Destination IP', 'Timestamp']
df.drop(drop_columns, axis=1, inplace=True, errors='ignore')
```

In [ ]:
```
# Normalize numerical features
scaler = StandardScaler()
numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns
df[numerical_cols] = scaler.fit_transform(df[numerical_cols])
```

In [ ]:
```
# Split dataset
X = df.drop('Label', axis=1)
y = df['Label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_sta
```

In [ ]:
```
# Train Random Forest Model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42, n_jobs=-1)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)
```

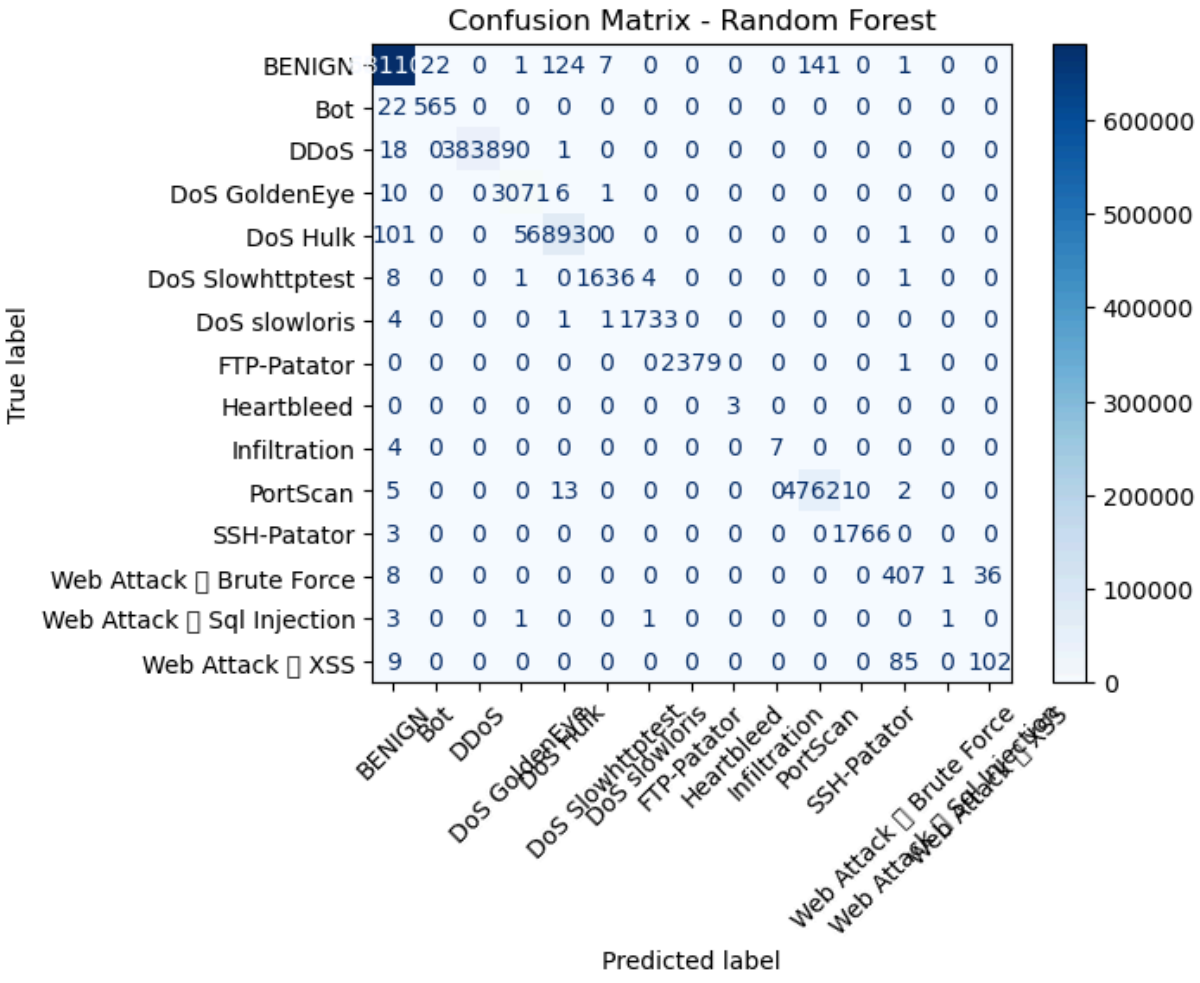
In [ ]:
```
# Train XGBoost Model
xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='mlogloss', n_jobs=-
xgb_model.fit(X_train, y_train)
y_pred_xgb = xgb_model.predict(X_test)
```

```
c:\Users\abhim\anaconda3\Lib\site-packages\xgboost\training.py:183: UserWarning: [2
0:28:47] WARNING: C:\actions-runner\_work\xgboost\xgboost\src\learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
```

In [ ]:
```
# Model Evaluation
def evaluate_model(y_true, y_pred, model_name):
    print(f"{model_name} Classification Report:")
    print(classification_report(y_true, y_pred))

    cm = confusion_matrix(y_true, y_pred)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=label_mapping
    disp.plot(cmap=plt.cm.Blues, xticks_rotation=45)
    plt.title(f"Confusion Matrix - {model_name}")
    plt.show()

evaluate_model(y_test, y_pred_rf, "Random Forest")
evaluate_model(y_test, y_pred_xgb, "XGBoost")

from sklearn.metrics import accuracy_score

rf_accuracy = accuracy_score(y_test, y_pred_rf)
xgb_accuracy = accuracy_score(y_test, y_pred_xgb)

print(f"Random Forest Accuracy: {rf_accuracy:.4f}")
print(f"XGBoost Accuracy: {xgb_accuracy:.4f}")
```

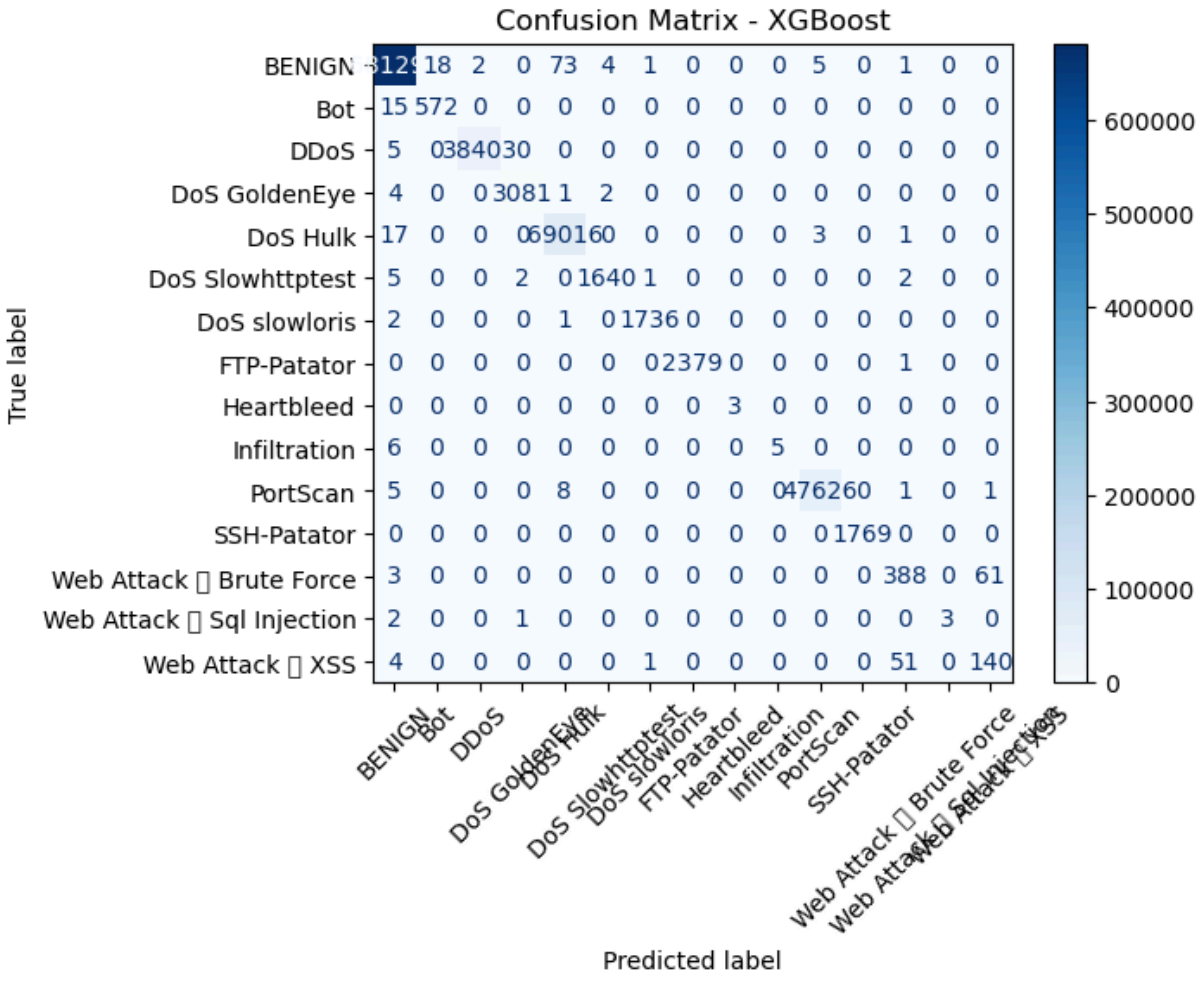
```
Random Forest Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    681396
           1       0.96      0.96      0.96       587
           2       1.00      1.00      1.00     38408
           3       1.00      0.99      1.00      3088
           4       1.00      1.00      1.00     69037
           5       0.99      0.99      0.99      1650
           6       1.00      1.00      1.00      1739
           7       1.00      1.00      1.00      2380
           8       1.00      1.00      1.00         3
           9       1.00      0.64      0.78        11
          10       1.00      1.00      1.00     47641
          11       1.00      1.00      1.00      1769
          12       0.82      0.90      0.86       452
          13       0.50      0.17      0.25         6
          14       0.74      0.52      0.61       196

    accuracy                           1.00    848363
   macro avg       0.93      0.88      0.90    848363
weighted avg       1.00      1.00      1.00    848363
```

```
c:\Users\abhim\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:170: UserWarni
ng: Glyph 150 (\x96) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



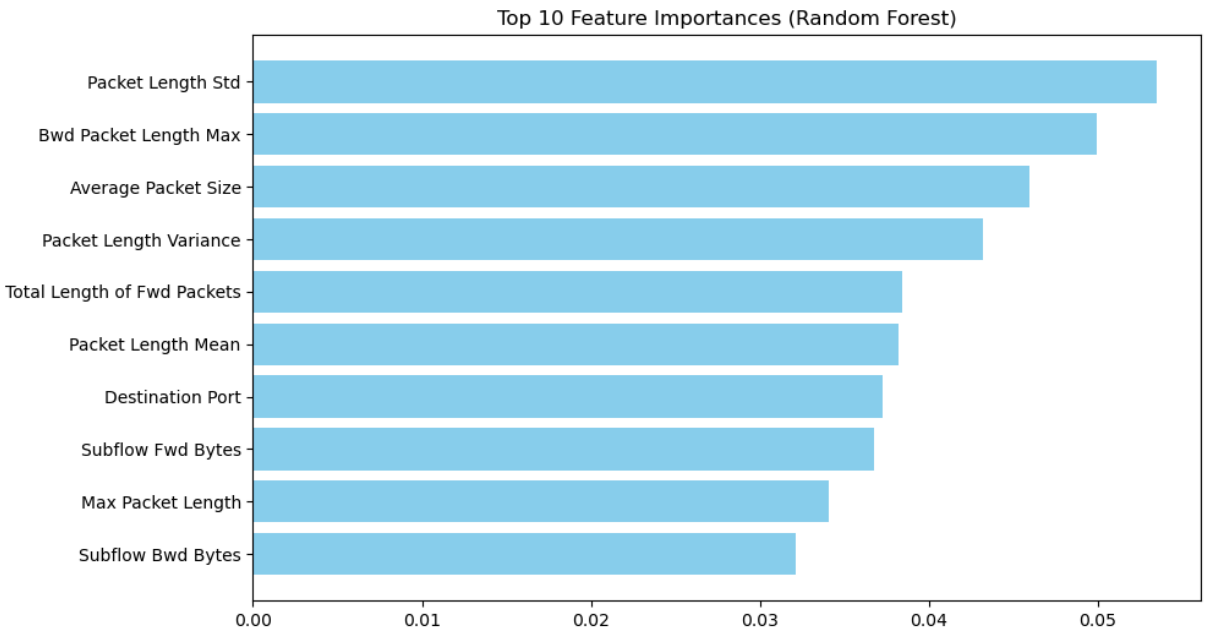
```
XGBoost Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    681396
           1       0.97      0.97      0.97       587
           2       1.00      1.00      1.00     38408
           3       1.00      1.00      1.00      3088
           4       1.00      1.00      1.00     69037
           5       1.00      0.99      1.00      1650
           6       1.00      1.00      1.00      1739
           7       1.00      1.00      1.00      2380
           8       1.00      1.00      1.00         3
           9       1.00      0.45      0.62        11
          10       1.00      1.00      1.00     47641
          11       1.00      1.00      1.00      1769
          12       0.87      0.86      0.87       452
          13       1.00      0.50      0.67         6
          14       0.69      0.71      0.70       196

    accuracy                           1.00    848363
   macro avg       0.97      0.90      0.92    848363
weighted avg       1.00      1.00      1.00    848363
```

```
c:\Users\abhim\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:170: UserWarni
ng: Glyph 150 (\x96) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



```
Random Forest Accuracy: 0.9992
XGBoost Accuracy: 0.9996
```

In [ ]:
```python
# Feature Importance
rf_importances = rf_model.feature_importances_
rf_indices = np.argsort(rf_importances)[-10:]

plt.figure(figsize=(10, 6))
plt.barh(range(len(rf_indices)), rf_importances[rf_indices], color='skyblue')
plt.yticks(range(len(rf_indices)), [X.columns[i] for i in rf_indices])
plt.title('Top 10 Feature Importances (Random Forest)')
plt.show()

from sklearn.metrics import roc_auc_score
from sklearn.preprocessing import label_binarize

# Binarize the labels (convert multiclass to multiple binary columns)
y_test_bin = label_binarize(y_test, classes=np.unique(y_test))
y_pred_rf_bin = label_binarize(y_pred_rf, classes=np.unique(y_test))
y_pred_xgb_bin = label_binarize(y_pred_xgb, classes=np.unique(y_test))

# Compute ROC-AUC Score for each class and take the average
rf_auc = roc_auc_score(y_test_bin, y_pred_rf_bin, average="macro")
xgb_auc = roc_auc_score(y_test_bin, y_pred_xgb_bin, average="macro")

print(f"Random Forest ROC-AUC Score: {rf_auc:.4f}")
print(f"XGBoost ROC-AUC Score: {xgb_auc:.4f}")
```




```
Random Forest ROC-AUC Score: 0.9387
XGBoost ROC-AUC Score: 0.9497
```