Software Network Intrusion Detection

Rutuja Kute, Ruthwik

Overview

Intrusion Detection Systems and Prevention Systems are crucial tools for network users to defend against various online threats. Our project aims to develop a system that can protect SDNs against various types of online threats.
Goal is to develop an accurate and efficient network intrusion detection system for Software Defined Networks (SDNs) using machine learning algorithms.We are using machine learning algorithms to classify four types of online threats in our project: DDoS attacks, Web Attack Brute Force, Web Attack XSS, and Web Attack SQL Injection.

Flow of the Code

01 Importing essentail libaries

02 Exploring the SDN dataset

03 Preprocessing dataset

04 Feature Selection

05 Visualization

06 Algorithms Used

07 Learning Curve

08 Optimising Hyperparameters

09 Confusion Matrix

Importing Libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler

from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import learning_curve
from joblib import parallel_backend
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from xgboost import XGBClassifier
```

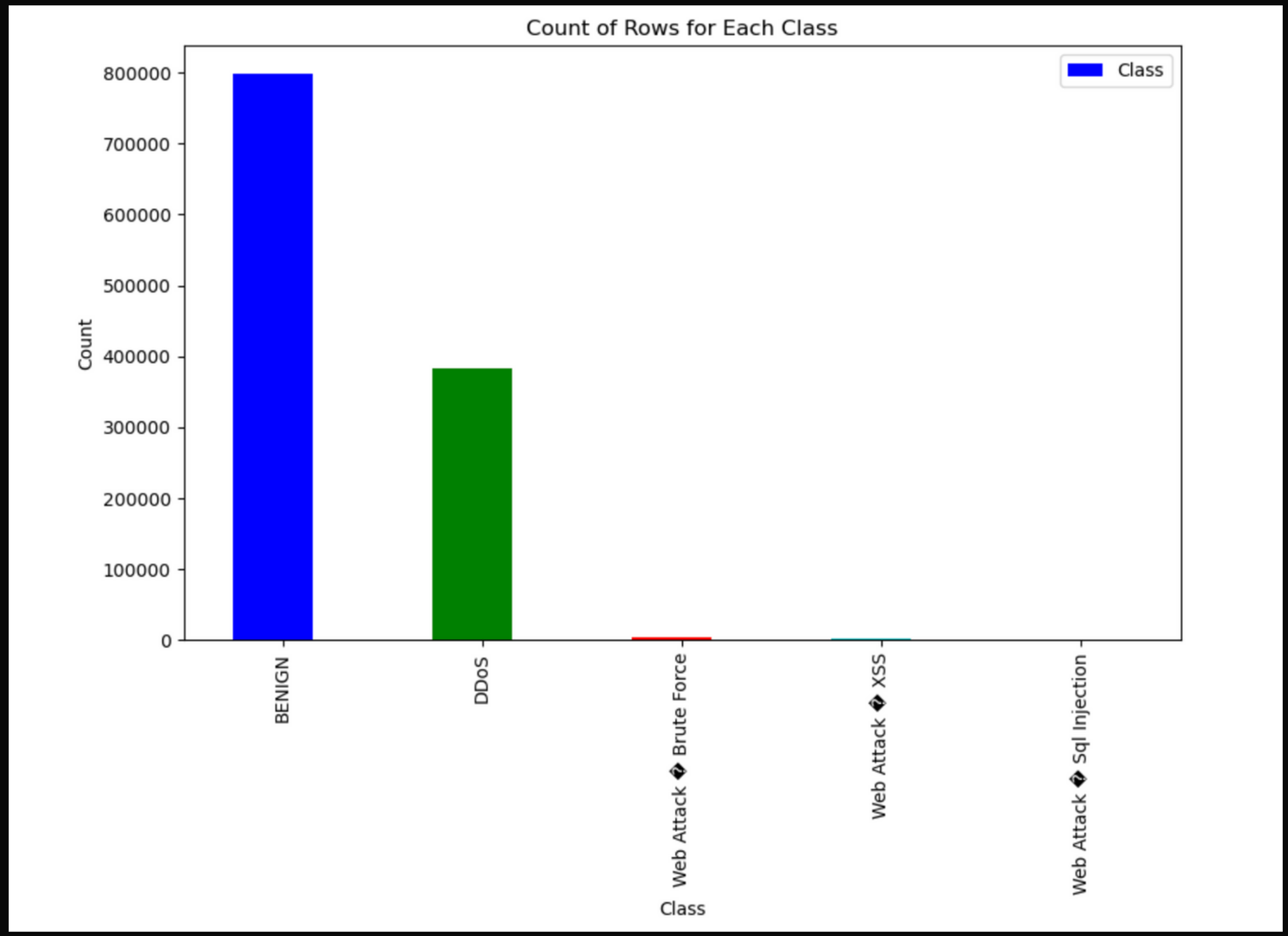
Exploring Dataset

df.info()

df.describe()

```
[4]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1188333 entries, 0 to 1188332
Data columns (total 80 columns):
 #   Column                       Non-Null Count    Dtype
---  ------                       --------------    -----
 0   Unnamed: 0                   1188333 non-null  int64
 1    Destination Port            1188333 non-null  int64
 2    Flow Duration               1188333 non-null  int64
 3    Total Fwd Packets           1188333 non-null  int64
 4    Total Backward Packets      1188333 non-null  int64
 5   Total Length of Fwd Packets  1188333 non-null  int64
 6    Total Length of Bwd Packets 1188333 non-null  int64
 7    Fwd Packet Length Max       1188333 non-null  int64
 8    Fwd Packet Length Min       1188333 non-null  int64
 9    Fwd Packet Length Mean      1188333 non-null  float64
 10   Fwd Packet Length Std       1188333 non-null  float64
 11  Bwd Packet Length Max        1188333 non-null  int64
 12   Bwd Packet Length Min       1188333 non-null  int64
 13   Bwd Packet Length Mean      1188333 non-null  float64
 14   Bwd Packet Length Std       1188333 non-null  float64
 15  Flow Bytes/s                 1188262 non-null  float64
 16   Flow Packets/s              1188333 non-null  float64
 17   Flow IAT Mean               1188333 non-null  float64
 18   Flow IAT Std                1188333 non-null  float64
 19   Flow IAT Max                1188333 non-null  int64
 20   Flow IAT Min                1188333 non-null  int64
 21  Fwd IAT Total                1188333 non-null  int64
 22   Fwd IAT Mean                1188333 non-null  float64
 23   Fwd IAT Std                 1188333 non-null  float64
 24   Fwd IAT Max                 1188333 non-null  int64
 25   Fwd IAT Min                 1188333 non-null  int64
 26  Bwd IAT Total                1188333 non-null  int64
 27   Bwd IAT Mean                1188333 non-null  float64
 28   Bwd IAT Std                 1188333 non-null  float64
 29   Bwd IAT Max                 1188333 non-null  int64
 30   Bwd IAT Min                 1188333 non-null  int64
 31  Fwd PSH Flags                1188333 non-null  int64
 32   Bwd PSH Flags               1188333 non-null  int64
 33   Fwd URG Flags               1188333 non-null  int64
 34   Bwd URG Flags               1188333 non-null  int64
 35   Fwd Header Length           1188333 non-null  int64
 36   Bwd Header Length           1188333 non-null  int64
 37  Fwd Packets/s                1188333 non-null  float64
 38   Bwd Packets/s               1188333 non-null  float64
 39   Min Packet Length           1188333 non-null  int64
 40   Max Packet Length           1188333 non-null  int64
 41   Packet Length Mean          1188333 non-null  float64
 42   Packet Length Std           1188333 non-null  float64
 43   Packet Length Variance      1188333 non-null  float64
```

Visualization



Preprocessing Dataset

Determine columns containing nulls

```
: #Dropping missing rows
df.dropna(inplace = True)
df.drop('Unnamed: 0', axis = 1, inplace = True)
df
```

```
df.shape
```

```
(1188262, 79)
```

Encode the target column i.e Class

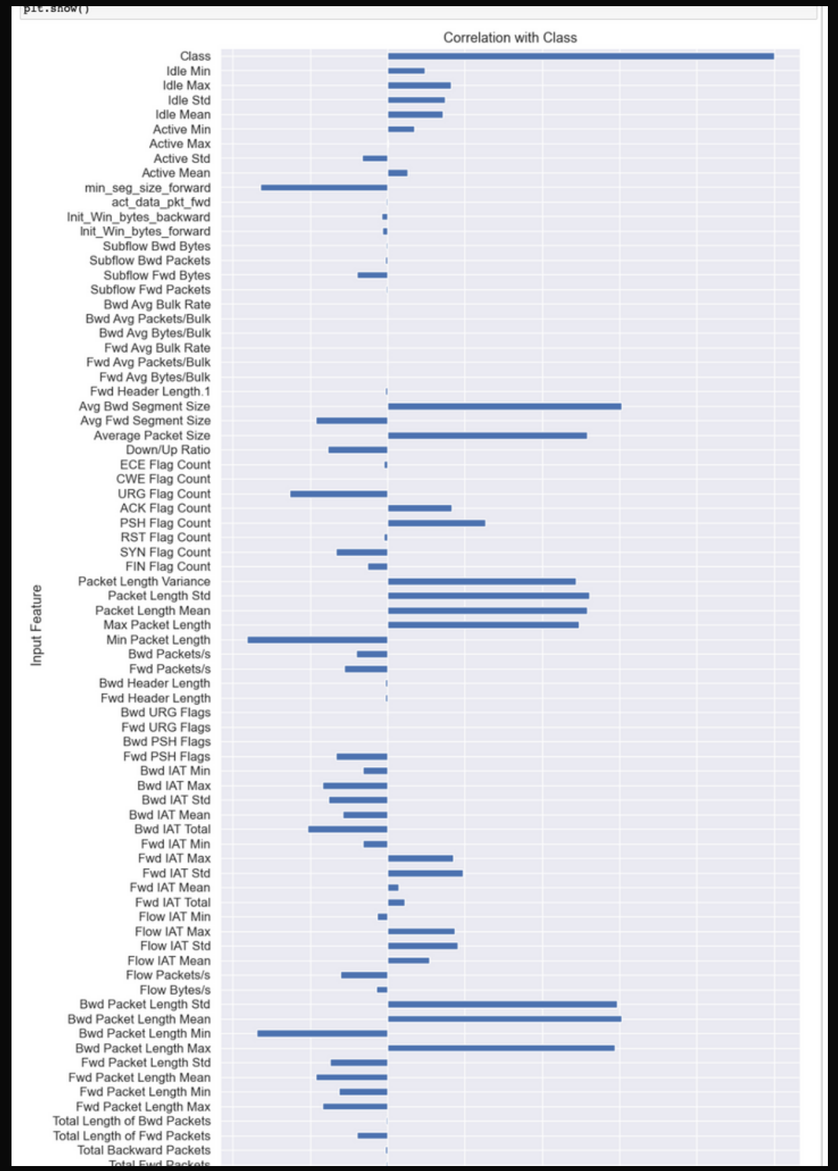
```
7]: #Encoding the Class labels to numerical values to calculate correlation
le = LabelEncoder()
df_1['Class'] = le.fit_transform(df['Class'])
df_1
```

Feature Selection

Encoding the Class column

Examining Correlation of Features

Dropping uncorrelated and higly correlated columns (to avoid overfitting)

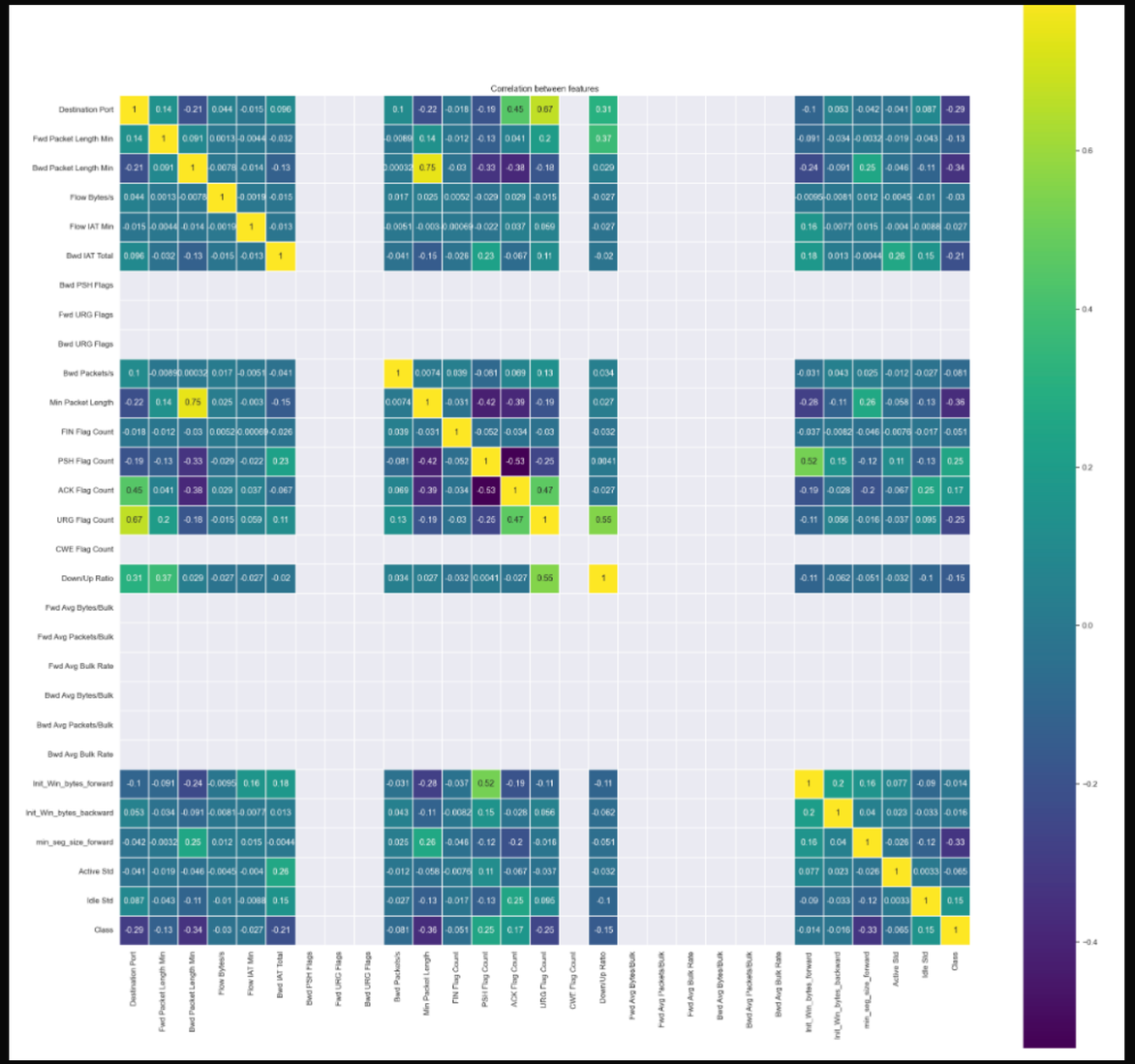


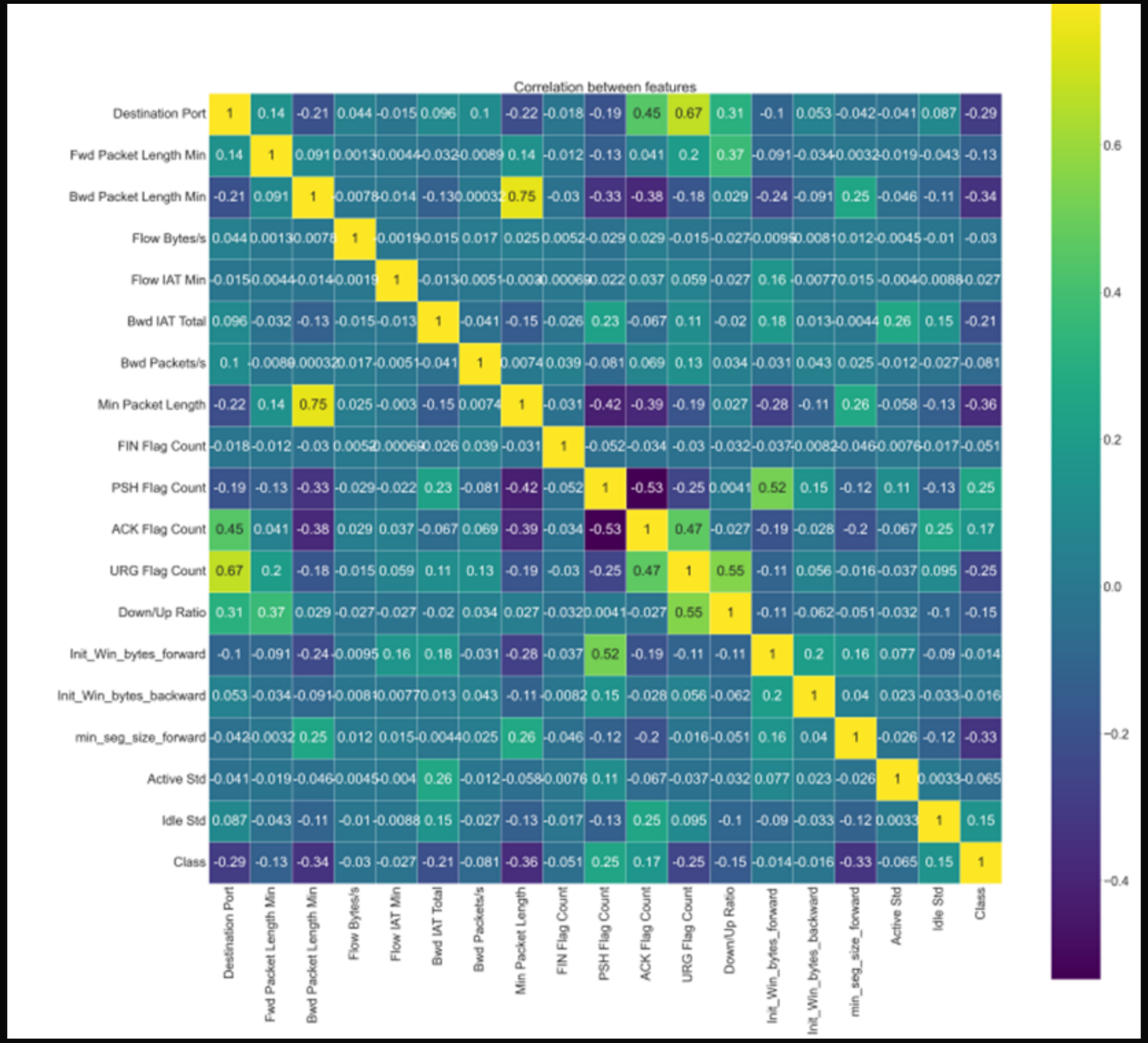
Visualizations

The first step is to compute the correlation matrix using the corr() function from pandas, which returns a square matrix containing the Pearson correlation coefficients between each pair of features.

Seaborn library is used to create a heatmap of the correlation matrix

Dropping uncorrelated and higly correlated columns (to avoid overfitting)




Correlation between features
Destination Port 1 0.14 -0.21 0.044 -0.015 0.096 0.1 -0.22 -0.018 -0.19 0.45 0.67 0.31 -0.1 0.053 -0.042 -0.041 0.087 -0.29
Fwd Packet Length Min 0.14 1 0.091 0.0013 0.0044 -0.032 -0.0089 0.14 -0.012 -0.13 0.041 0.2 0.37 -0.091 -0.034 -0.0032 -0.019 -0.043 -0.13
Bwd Packet Length Min -0.21 0.091 1 -0.0078 -0.014 -0.13 -0.00032 0.75 -0.03 -0.33 -0.38 -0.18 0.029 -0.24 -0.091 0.25 -0.046 -0.11 -0.34
Flow Bytes/s 0.044 0.0013 -0.0078 1 -0.0019 -0.015 0.017 0.025 0.0052 -0.029 0.029 -0.015 -0.027 -0.0095 0.0081 0.012 -0.0045 -0.01 -0.03
Flow IAT Min -0.015 -0.0044 -0.014 -0.0019 1 -0.013 -0.0051 -0.003 0.00069 0.022 0.037 0.059 -0.027 0.16 -0.0077 0.015 -0.004 -0.0088 0.027
Bwd IAT Total 0.096 -0.032 -0.13 -0.015 -0.013 1 -0.041 -0.15 -0.026 0.23 -0.067 0.11 -0.02 0.18 0.013 -0.0044 0.26 0.15 -0.21
Bwd Packets/s 0.1 -0.0089 -0.00032 0.017 -0.0051 -0.041 1 0.0074 0.039 -0.081 0.069 0.13 0.034 -0.031 0.043 0.025 -0.012 -0.027 -0.081
Min Packet Length -0.22 0.14 0.75 0.025 -0.003 -0.15 0.0074 1 -0.031 -0.42 -0.39 -0.19 0.027 -0.28 -0.11 0.26 -0.058 -0.13 -0.36
FIN Flag Count -0.018 -0.012 -0.03 0.0052 0.00069 0.026 0.039 -0.031 1 -0.052 -0.034 -0.03 -0.032 -0.037 -0.0082 -0.046 0.0076 -0.017 -0.051
PSH Flag Count -0.19 -0.13 -0.33 -0.029 -0.022 0.23 -0.081 -0.42 -0.052 1 -0.53 -0.25 0.0041 0.52 0.15 -0.12 0.11 -0.13 0.25
ACK Flag Count 0.45 0.041 -0.38 0.029 0.037 -0.067 0.069 -0.39 -0.034 -0.53 1 0.47 -0.027 -0.19 -0.028 -0.2 -0.067 0.25 0.17
URG Flag Count 0.67 0.2 -0.18 -0.015 0.059 0.11 0.13 -0.19 -0.03 -0.25 0.47 1 0.55 -0.11 0.056 -0.016 -0.037 0.095 -0.25
Down/Up Ratio 0.31 0.37 0.029 -0.027 -0.027 -0.02 0.034 0.027 -0.032 0.0041 -0.027 0.55 1 -0.11 -0.062 -0.051 -0.032 -0.1 -0.15
Init_Win_bytes_forward -0.1 -0.091 -0.24 -0.0095 0.16 0.18 -0.031 -0.28 -0.037 0.52 -0.19 -0.11 -0.11 1 0.2 0.16 0.077 -0.09 -0.014
Init_Win_bytes_backward 0.053 -0.034 -0.091 0.0081 0.0077 0.013 0.043 -0.11 -0.0082 0.15 -0.028 0.056 -0.062 0.2 1 0.04 0.023 -0.033 -0.016
min_seg_size_forward -0.042 0.0032 0.25 0.012 0.015 -0.0044 0.025 0.26 -0.046 -0.12 -0.2 -0.016 -0.051 0.16 0.04 1 -0.026 -0.12 -0.33
Active Std -0.041 -0.019 -0.046 -0.0045 -0.004 0.26 -0.012 -0.058 0.0076 0.11 -0.067 -0.037 -0.032 0.077 0.023 -0.026 1 0.0033 -0.065
Idle Std 0.087 -0.043 -0.11 -0.01 -0.0088 0.15 -0.027 -0.13 -0.017 -0.13 0.25 0.095 -0.1 -0.09 -0.033 -0.12 0.0033 1 0.15
Class -0.29 -0.13 -0.34 -0.03 -0.027 -0.21 -0.081 -0.36 -0.051 0.25 0.17 -0.25 -0.15 -0.014 -0.016 -0.33 -0.065 0.15 1
Destination Port
Fwd Packet Length Min
Bwd Packet Length Min
Flow Bytes/s
Flow IAT Min
Bwd IAT Total
Bwd Packets/s
Min Packet Length
FIN Flag Count
PSH Flag Count
ACK Flag Count
URG Flag Count
Down/Up Ratio
Init_Win_bytes_forward
Init_Win_bytes_backward
min_seg_size_forward
Active Std
Idle Std
Class
0.6
0.4
0.2
0.0
−0.2
−0.4

Algorithms Used

Supervised Classification Algorithms

Decission Tree Classifier

Random Forest Classifier

Logistic Regression Classifier

XGBoost Classifier

Naive Bayes Classifier

Normalization:

Normalization of Data

5]:
```
X = pd.DataFrame(X, columns=X.columns)
threshold = 1000
X = X.clip(lower=-threshold, upper=threshold)



# Scale the data using MinMaxScaler
scaler = MinMaxScaler()
X = scaler.fit_transform(X)
```

6]:
```
#Train Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Random Forest Classifier:

```
# Train a random forest classifier on the training set
clf = RandomForestClassifier(n_estimators=100, random_state=42,max_depth=2)
clf.fit(X_train, y_train)

# Evaluate the model on the testing set
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
Accuracy: 94.53%
```

Optimising Hyperparameters

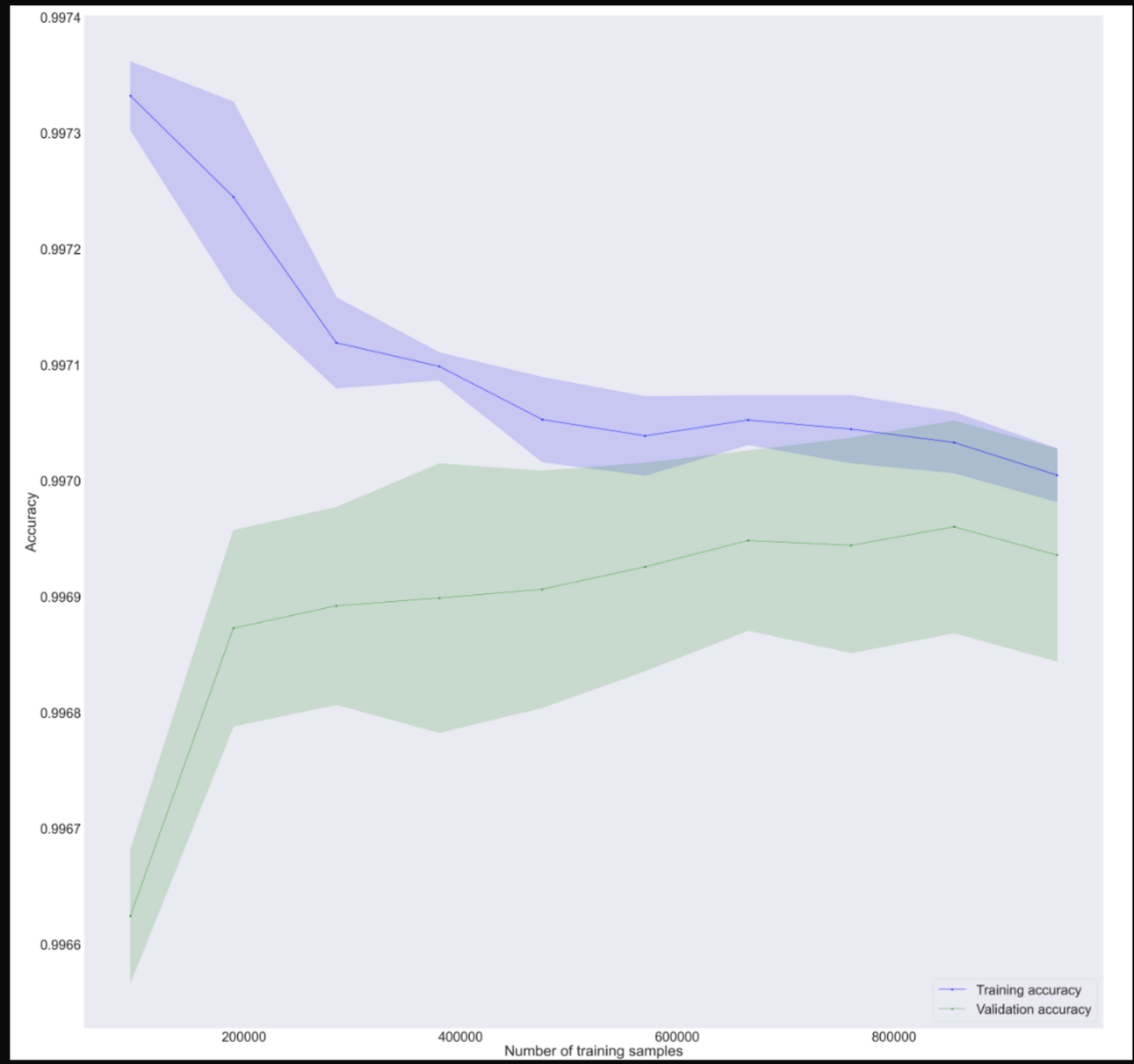
RandomizedSearchCV to find optimal hyperparameters

[55]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint

# Define the decision tree model
model = DecisionTreeClassifier()

# Define the hyperparameters to tune
params = {'max_depth': randint(2, 10),
          'min_samples_split': randint(2, 10),
          'min_samples_leaf': randint(1, 5)}

# Use RandomizedSearchCV to find the best hyperparameters
with parallel_backend('multiprocessing', n_jobs=4):
    random_search = RandomizedSearchCV(model, params, cv=5, n_iter=50, random_state=42)
    random_search.fit(X_train, y_train)



# Train the model with the best hyperparameters
best_model = random_search.best_estimator_
best_model.fit(X_train, y_train)
```

[55]:
```
DecisionTreeClassifier(max_depth=9, min_samples_split=3)
```

Learning Curve

The learning curve shows the relationship between the size of the training set and the performance of the model, such as accuracy. It helps to identify if the model is underfitting (high bias) or overfitting (high variance).
The learning curve is important because it helps to determine the optimal amount of data needed to train a model effectively and to identify any issues with the model's performance.

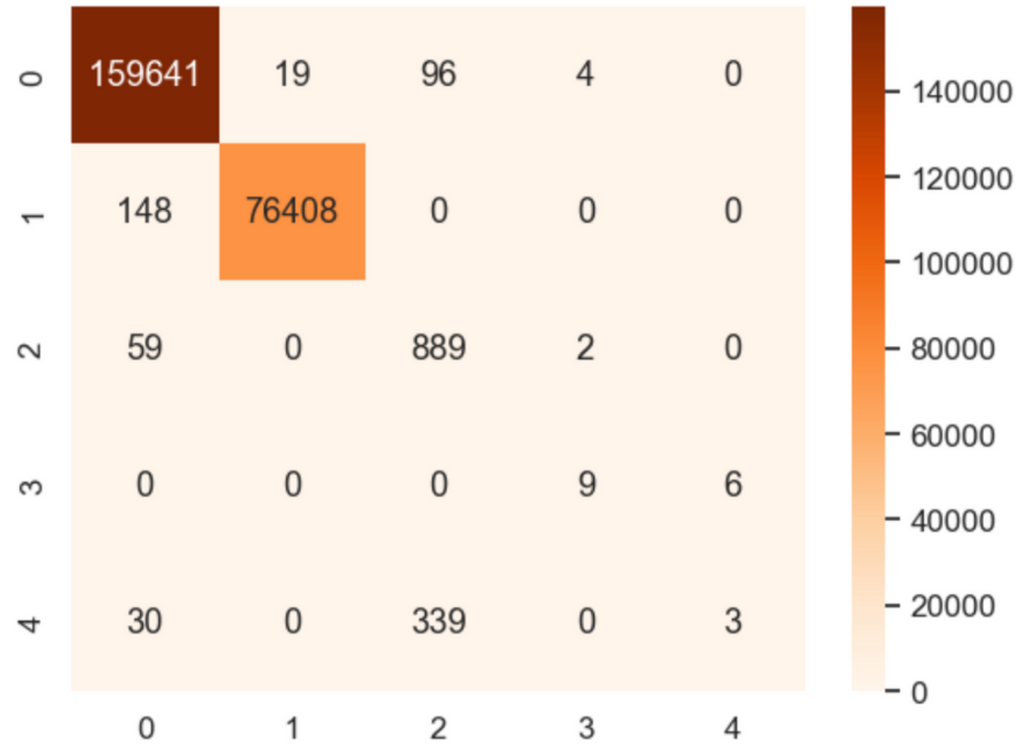


Confusion Matrix

Confusion Matrix

In [63]:
```
import seaborn as sns
sns.set(font_scale=1.10)
sns.heatmap(cm, annot=True, cmap='Oranges', fmt = 'd')
```

Out[63]: <AxesSubplot:>



Cross Validation

Cross Validation

In [33]:
```
from sklearn.model_selection import cross_val_score
cv_scores = cross_val_score(best_model, X_train, y_train, cv=5)

# Calculate the average cross-validation score
avg_cv_score = np.mean(cv_scores)

cv_scores
```

Out[33]:
```
array([0.99684413, 0.9970177 , 0.99694407, 0.99697037, 0.99688619])
```

Conclusion

It is clear that the XGBoost Classifier outperformed the other models in terms of accuracy with an impressive score of 99.74%. However, it was observed that the XGBoost model overfit the data, as it performed significantly better on the training set than on the testing set. The Logistic Regression Classifier also performed well with an accuracy score of 97.78%. The Random Forest Classifier and Naive Bayes Classifier performed slightly lower, but still achieved respectable accuracy scores of 94.53% and 95.58% respectively.

Resources

Data Sources: https://www.kaggle.com/datasets/subhajournal/sdn-intrusion-detection