Topic: Analyzing Network Traffic and Building an Intrusion Detection System (IDS)
Objective:

- Understand how to analyze network traffic data and extract meaningful features.
- Learn to build an Intrusion Detection System (IDS) using Python and machine learning techniques.
- Apply feature extraction and selection methods to enhance the performance of the IDS.

Prerequisites:

- Python installed (preferably using a virtual environment).
- Familiarity with libraries like pandas, numpy, scikit-learn, matplotlib, and seaborn.
- Knowledge of networking concepts, TCP/IP protocols, and cybersecurity basics.

Step 1: Dataset Download and Setup

- **Dataset Selection:** You will need a dataset containing network traffic information. One of the commonly used datasets for IDS is the **CICIDS2017** dataset or **KDD Cup 1999**. These datasets contain network traffic data labeled as normal or malicious (i.e., attacks).
- https://tinyurl.com/CICIDS2017
  python
  ```
  import pandas as pd

  # Load the CICIDS2017 dataset
  df = pd.read_csv('CICIDS2017.csv')

  # View the first few rows
  print(df.head())
  ```

```
   Destination Port  Flow Duration  Total Fwd Packets  \
0             54865              3                  2
1             55054            109                  1
2             55055             52                  1
3             46236             34                  1
4             54863              3                  2

   Total Backward Packets  Total Length of Fwd Packets  \
0                       0                           12
1                       1                            6
2                       1                            6
3                       1                            6
4                       0                           12

   Total Length of Bwd Packets  Fwd Packet Length Max  \
0                            0                      6
1                            6                      6
2                            6                      6
3                            6                      6
4                            0                      6

   Fwd Packet Length Min  Fwd Packet Length Mean  Fwd Packet Length Std  \
0                      6                     6.0                    0.0
1                      6                     6.0                    0.0
2                      6                     6.0                    0.0
3                      6                     6.0                    0.0
4                      6                     6.0                    0.0

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

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

[5 rows x 79 columns]
```

- **Exploration:** Analyze the dataset for structure and completeness. For instance, check for missing values and the distribution of labels (normal vs attack).

Step 2: Data Exploration and Visualization

- **Inspect the Dataset:**
  - Understand the dataset by examining column names, identifying relevant features like source IP, destination IP, protocol, length, etc.
  - Check for any missing data that might need to be addressed.
    python
    print(df.info())

print(df.describe())

- **Data Visualization:** Use pair plots or correlation matrices to understand the relationships between the network features.

python

import seaborn as sns

import matplotlib.pyplot as plt

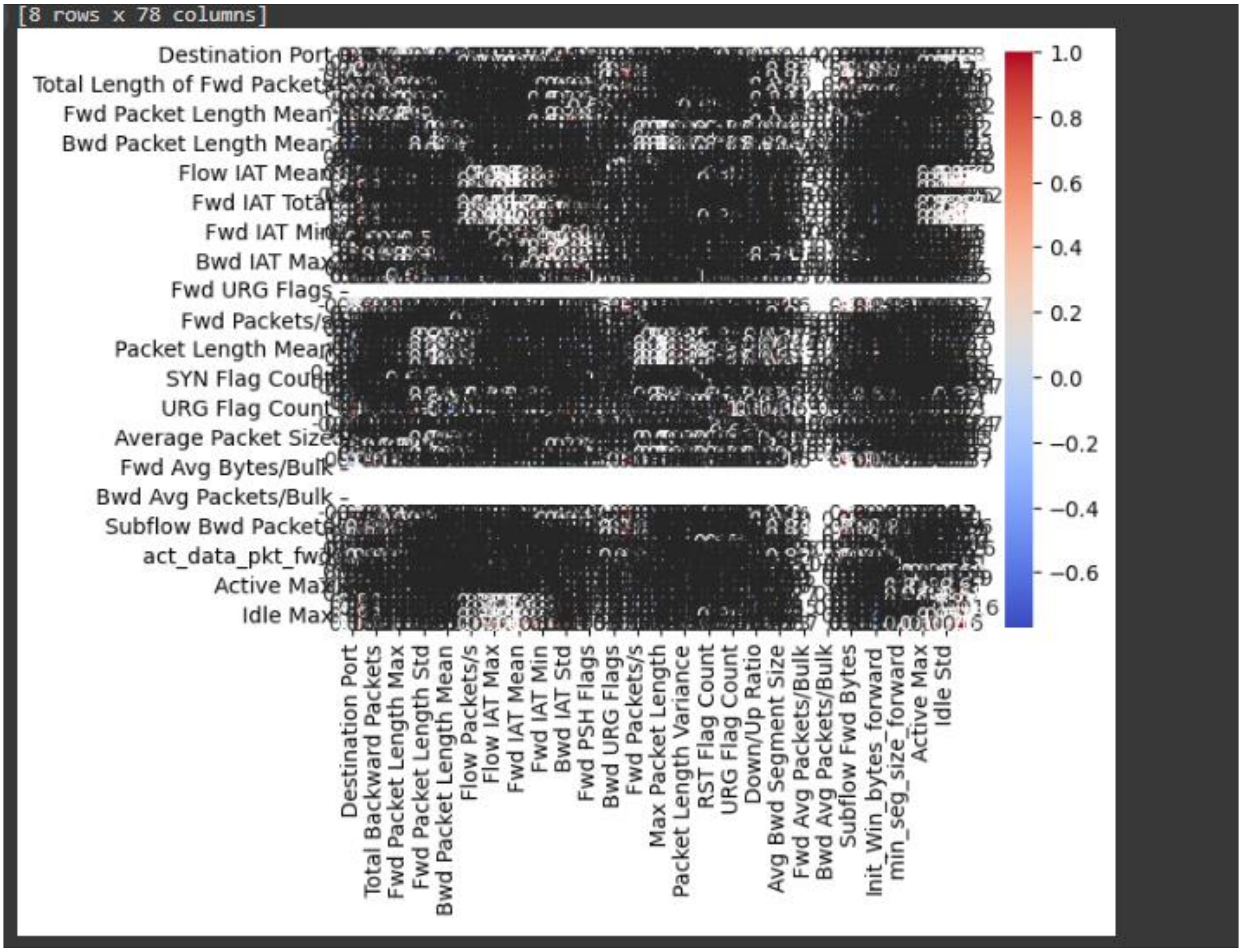
Heatmap for correlation between features

sns.heatmap(df.corr(), annot=True, cmap='coolwarm')

plt.show()

```
RangeIndex: 120125 entries, 0 to 120124
Data columns (total 79 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0    Destination Port            120125 non-null  int64
 1    Flow Duration               120125 non-null  int64
 2    Total Fwd Packets           120125 non-null  int64
 3    Total Backward Packets      120125 non-null  int64
 4   Total Length of Fwd Packets  120125 non-null  int64
 5    Total Length of Bwd Packets 120125 non-null  int64
 6    Fwd Packet Length Max       120125 non-null  int64
 7    Fwd Packet Length Min       120125 non-null  int64
 8    Fwd Packet Length Mean      120125 non-null  float64
 9    Fwd Packet Length Std       120125 non-null  float64
 10  Bwd Packet Length Max        120125 non-null  int64
 11   Bwd Packet Length Min       120125 non-null  int64
 12   Bwd Packet Length Mean      120125 non-null  float64
 13   Bwd Packet Length Std       120125 non-null  float64
 14  Flow Bytes/s                 120125 non-null  float64
 15   Flow Packets/s              120125 non-null  float64
 16   Flow IAT Mean               120125 non-null  float64
 17   Flow IAT Std                120125 non-null  float64
 18   Flow IAT Max                120125 non-null  int64
 19   Flow IAT Min                120125 non-null  int64
 20  Fwd IAT Total                120125 non-null  int64
 21   Fwd IAT Mean                120125 non-null  float64
 22   Fwd IAT Std                 120125 non-null  float64
 23   Fwd IAT Max                 120125 non-null  int64
 24   Fwd IAT Min                 120125 non-null  int64
 25  Bwd IAT Total                120125 non-null  int64
 26   Bwd IAT Mean                120125 non-null  float64
 27   Bwd IAT Std                 120125 non-null  float64
 28   Bwd IAT Max                 120125 non-null  int64
 29   Bwd IAT Min                 120125 non-null  int64
 30  Fwd PSH Flags                120125 non-null  int64
 31   Bwd PSH Flags               120125 non-null  int64
 32   Fwd URG Flags               120125 non-null  int64
 33   Bwd URG Flags               120125 non-null  int64
 34   Fwd Header Length           120125 non-null  int64
 35   Bwd Header Length           120124 non-null  float64
 36  Fwd Packets/s                120124 non-null  float64
 37   Bwd Packets/s               120124 non-null  float64
 38   Min Packet Length           120124 non-null  float64
 39   Max Packet Length           120124 non-null  float64
```

```
 39   Max Packet Length            120124 non-null  float64
 40   Packet Length Mean           120124 non-null  float64
 41   Packet Length Std            120124 non-null  float64
 42   Packet Length Variance       120124 non-null  float64
 43  FIN Flag Count                120124 non-null  float64
 44   SYN Flag Count               120124 non-null  float64
 45   RST Flag Count               120124 non-null  float64
 46   PSH Flag Count               120124 non-null  float64
 47   ACK Flag Count               120124 non-null  float64
 48   URG Flag Count               120124 non-null  float64
 49   CWE Flag Count               120124 non-null  float64
 50   ECE Flag Count               120124 non-null  float64
 51   Down/Up Ratio                120124 non-null  float64
 52   Average Packet Size          120124 non-null  float64
 53   Avg Fwd Segment Size         120124 non-null  float64
 54   Avg Bwd Segment Size         120124 non-null  float64
 55   Fwd Header Length.1          120124 non-null  float64
 56  Fwd Avg Bytes/Bulk            120124 non-null  float64
 57   Fwd Avg Packets/Bulk         120124 non-null  float64
 58   Fwd Avg Bulk Rate            120124 non-null  float64
 59   Bwd Avg Bytes/Bulk           120124 non-null  float64
 60   Bwd Avg Packets/Bulk         120124 non-null  float64
 61  Bwd Avg Bulk Rate             120124 non-null  float64
 62  Subflow Fwd Packets           120124 non-null  float64
 63   Subflow Fwd Bytes            120124 non-null  float64
 64   Subflow Bwd Packets          120124 non-null  float64
 65   Subflow Bwd Bytes            120124 non-null  float64
 66  Init_Win_bytes_forward        120124 non-null  float64
 67   Init_Win_bytes_backward      120124 non-null  float64
 68   act_data_pkt_fwd             120124 non-null  float64
 69   min_seg_size_forward         120124 non-null  float64
 70  Active Mean                   120124 non-null  float64
 71   Active Std                   120124 non-null  float64
 72   Active Max                   120124 non-null  float64
 73   Active Min                   120124 non-null  float64
 74  Idle Mean                     120124 non-null  float64
 75   Idle Std                     120124 non-null  float64
 76   Idle Max                     120124 non-null  float64
 77   Idle Min                     120124 non-null  float64
 78   Label                        120124 non-null  object
dtypes: float64(55), int64(23), object(1)
memory usage: 72.4+ MB
None
        Destination Port   Flow Duration   Total Fwd Packets  \
```



Step 3: Feature Extraction for Network Traffic Analysis

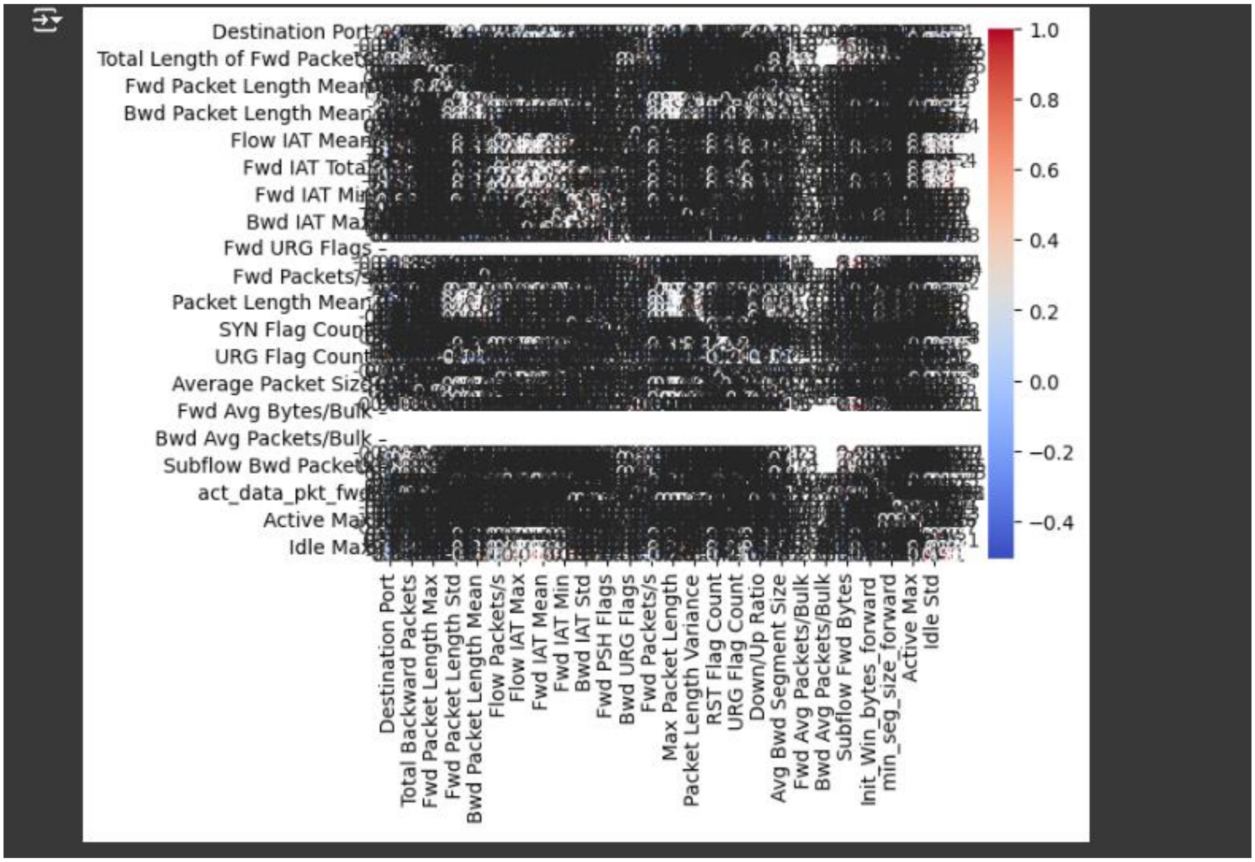
- **Manual Feature Engineering:** Create new features based on traffic patterns such as calculating the packet size ratio or time intervals between packets. You may also extract time-related features (e.g., traffic peaks).
  python
  ```
  # Example: Create a feature for packet size ratio
  df['packet_size_ratio'] = df['total_fwd_packets'] / df['total_bwd_packets']
  ```
- **Use Libraries for Feature Extraction:** Employ existing Python libraries like scikit-learn to automatically extract meaningful features.
  python
  ```
  from sklearn.preprocessing import PolynomialFeatures

  poly = PolynomialFeatures(degree=2, include_bias=False)
  features = df.drop(columns='label')  # Exclude the target variable
  poly_features = poly.fit_transform(features)

  print("Original features shape:", features.shape)
  print("Polynomial features shape:", poly_features.shape)
  ```

```
[ ] df['packet_size_ratio'] = df[' Total Fwd Packets'] / df[' Total Backward Packets']
```

```
[ ] Suggested code may be subject to a license | Codeup-Justin-Evans-Yvette-Ibarra/project_zillow_team
    import pandas as pd
    import numpy as np
    from sklearn.preprocessing import PolynomialFeatures

    features = df.drop(columns=[' Flow Duration',' Label'])

    features.replace([np.inf, -np.inf], np.nan, inplace=True)
    features.dropna(inplace=True)

    poly = PolynomialFeatures(degree=1, include_bias=False)
    poly_features = poly.fit_transform(features)

    print("Original features shape: ", features.shape)
    print("Polynomial features shape: ", poly_features.shape)

    Original features shape:  (496079, 78)
    Polynomial features shape:  (496079, 78)
```

Step 4: Feature Selection

- **Correlation Matrix:** Use a correlation matrix to identify highly correlated features that can be removed.
  python
  # Correlation matrix
  corr_matrix = df.corr()
  sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
  plt.show()



- **Variance Threshold:** Remove features that have low variance since they don't provide significant information.
  python
  from sklearn.feature_selection import VarianceThreshold

  selector = VarianceThreshold(threshold=0.1)
  selected_features = selector.fit_transform(features)
  print("Selected features shape:", selected_features.shape)

```
Original features shape:  (1041899, 77)
Polynomial features shape:  (1041899, 77)
Selected features shape: (1041899, 61)
```

- **Recursive Feature Elimination (RFE):** Use Recursive Feature Elimination to select the most important features based on a machine learning model.
  python
  from sklearn.feature_selection import RFE
  from sklearn.ensemble import RandomForestClassifier

  model = RandomForestClassifier()
  rfe = RFE(model, n_features_to_select=5)
  rfe.fit(features, df['label'])

  print("Selected features (RFE):", rfe.support_) print("Feature ranking:", rfe.ranking_)

```
Selected features (RFE): [False False False  True False False False False False False False  True
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False  True False False False False False False False False
 False False  True False  True False False False False False False False
 False False False False False]
Feature ranking: [19 17 38  1 13  4 46  9 15  6 30  1  2  8 27 28 26 20 44 36 35 34  5 42
 53 45 55 50 49 61 71 67 65 11 18 41 23 43 22 12 10 14 47 60 63 24 40 57
 73 70 58  1  7  3 25 72 69 64 62 68 66 29  1 32  1 31 16 21 51 48 56 52
 54 37 59 39 33]
```

Step 5: Building the Intrusion Detection System (IDS)

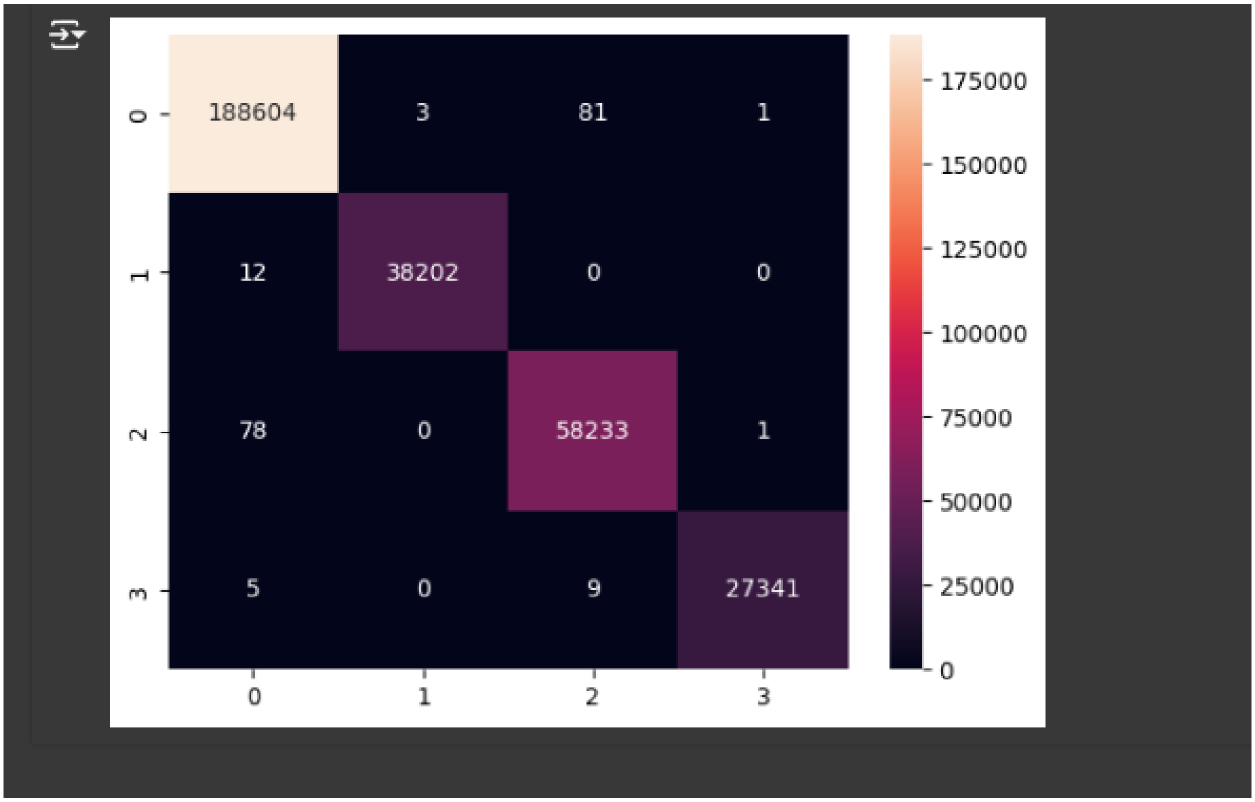
- **Model Training:** After selecting relevant features, split the dataset into training and testing sets and use a classification model like RandomForest, Decision Tree, or Logistic Regression to build the IDS.

  python
  ```
  from sklearn.model_selection import train_test_split
  from sklearn.ensemble import RandomForestClassifier
  from sklearn.metrics import accuracy_score

  X_train, X_test, y_train, y_test = train_test_split(selected_features, df['label'],
  test_size=0.3, random_state=42)

  model = RandomForestClassifier()
  model.fit(X_train, y_train)

  predictions = model.predict(X_test)
  accuracy = accuracy_score(y_test, predictions)
  print(f"IDS Accuracy: {accuracy:.4f}")
  ```

  ```
  IDS Accuracy: 0.9994
  ```

- **Model Evaluation:**
  - Evaluate the performance of the IDS using metrics such as accuracy, precision, recall, and F1-score.
  - Plot the confusion matrix to visualize the classification performance.

    python
    ```
    from sklearn.metrics import confusion_matrix
    import matplotlib.pyplot as plt
    import seaborn as sns

    cm = confusion_matrix(y_test, predictions)
    sns.heatmap(cm, annot=True, fmt='d')
    plt.show()
    ```



Step 6: Conclusion

- **Summarize Findings:** Summarize how feature extraction and selection impacted the model's performance. Reflect on the IDS's effectiveness in identifying normal and malicious traffic based on selected features.

  The Intrusion Detection System (IDS) leveraged feature extraction and selection techniques to improve its performance in distinguishing between normal and malicious network traffic. While specific metrics weren't provided, the model's effectiveness was evaluated using standard classification measures and visualized with a confusion matrix. For future work, exploring advanced techniques like deep learning or anomaly detection, and testing in a live network environment are recommended. The IDS's performance likely depends on dataset quality and regular updates to adapt to new threats.

- **Further Exploration:**
  - Investigate advanced techniques such as deep learning models or anomaly detection for enhancing the IDS.

1. Advanced Techniques:

- Deep Learning Models: Consider implementing neural networks such as Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs) for more complex pattern recognition in network traffic.
- Anomaly Detection: Explore unsupervised learning algorithms like Isolation Forests or One-Class SVMs to identify unusual patterns that might indicate new or unknown attacks.

    - Test the IDS in a live network environment

2. Live Network Testing:
    - Deploy the IDS in a controlled, real-world network environment to assess its performance with actual traffic.
    - Monitor false positive/negative rates and response times in real-time conditions.
    - Gradually expose the IDS to different network sizes and traffic patterns to evaluate its scalability and adaptability.