Assignment 1: Dataset Exploration (15%)

Deadline: 3 November 2023

Student: Beyza Nur Nazlı

Student ID: 200029011

<u>Part 1: Dataset Selection</u>

1. Choose one cybersecurity-related dataset given in the lecture.

I chose CICIDS2017.

2. Provide a comprehensive overview of the selected dataset, including its source, purpose, and relevance to cybersecurity.

The CICIDS2017 dataset, also known as the Canadian Institute for Cybersecurity Intrusion Detection System 2017 dataset, is a widely used and referenced dataset in the field of cybersecurity. It was created for the purpose of research, experimentation, and the development of intrusion detection systems (IDS) and intrusion detection/prevention systems (IDPS).

The CICIDS2017 dataset was created and published by the Canadian Institute for Cybersecurity (CIC) at the University of New Brunswick. *The dataset is freely available for academic and research purposes and can be accessed from the CIC's website or other trusted sources.*

The primary purpose of the CICIDS2017 dataset is to serve as a benchmark for evaluating the performance of intrusion detection systems and intrusion detection/prevention systems. It was developed to provide a diverse and realistic set of network traffic data that can be used for research, training, and testing in the field of cybersecurity. The dataset aims to replicate a real-world network environment to help researchers and cybersecurity professionals better understand and address the challenges of identifying and mitigating cyber threats.

After all the CICIDS2017 dataset is composed of several components, including:

CIC-IDS2017 : A labeled network traffic dataset that contains both benign and malicious network traffic data. It encompasses a wide range of network attacks and anomalies, such as DoS, DDoS, port scanning, brute-force attacks, and more. The dataset is structured into multiple CSV files, each representing different attack scenarios and network traffic types.

CSE-CIC-IDS2017 : A more extensive dataset with additional features, such as extracted packet headers, network flow information, and various statistical features. This dataset offers a more in-depth analysis of network traffic, which can be valuable for more advanced intrusion detection and machine learning-based approaches.

Relevance to Cybersecurity:

The CICIDS2017 dataset is highly relevant to the field of cybersecurity for several reasons which are,

Realistic Data: The dataset contains real-world network traffic data, making it suitable for assessing the performance of intrusion detection systems in an environment that replicates actual cyber threats.

Diversity of Attacks: It covers a wide range of cyberattacks and anomalies, allowing researchers and practitioners to test and develop IDS/IDPS solutions for different attack vectors.

Research and Development: Researchers and cybersecurity professionals can use this dataset to evaluate the effectiveness of intrusion detection techniques, machine learning models, and security tools.

Training and Education: The dataset is an excellent resource for training and educating individuals in the field of cybersecurity, as it provides a rich dataset for hands-on exercises and experimentation.

Open Access: The dataset is freely available, making it accessible to a broad audience and encouraging collaborative research and development in the field of cybersecurity.

In conclusion, the CICIDS2017 dataset is a valuable resource for those working in the field of cybersecurity. Its source, purpose, and relevance in testing and developing intrusion detection systems make it an important tool for research and education in the domain of network security.

<u>Part 2: Data Preprocessing</u>

3. Load and preprocess the dataset, addressing missing values, performing feature selection, and applying data transformations where necessary.

1. Loading the Dataset:

We can download the CICIDS2017 dataset from a trusted source, such as the Canadian Institute for Cybersecurity website. Once we have the dataset files, we can load them into our preferred data analysis environment. Such as the example I gave below using and Pandas:

```
import pandas as pd

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

```

2. Handling Missing Values:

We need to check the dataset for missing values and decide how to handle them. Common strategies include removing rows with missing values or imputing missing values. For instance, we can remove rows with missing values like this:

```

# Remove rows with missing values
dataset = dataset.dropna()

```

3. Feature Selection:

Depending on the research or analysis we’re conducting, we may want to perform feature selection to reduce dimensionality and improve the efficiency of our models. We can use various feature selection techniques, such as mutual information, correlation analysis, or recursive feature elimination, to select the most relevant features.

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

# Select the top k features using chi-squared test
k = 10
X = dataset.drop('target', axis=1)  # We are assuming 'target' is as the label.
y = dataset['target']
selector = SelectKBest(score_func=chi2, k=k)
X_new = selector.fit_transform(X, y)
```

4. Data Transformations:

Depending on our analysis and machine learning models, we need to apply data transformations. Common transformations include feature scaling (e.g., Min-Max scaling or standardization) and one-hot encoding for categorical variables.

```
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
```

```
# Standardize numerical features
scaler = StandardScaler()
X_numerical = scaler.fit_transform(X_numerical)

# We will encode categorical features
encoder = OneHotEncoder()
X_categorical = encoder.fit_transform(X_categorical)
```

5. **Train-Test Split:**

Before applying machine learning algorithms, it's important to split our dataset into training and testing subsets for model evaluation.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

6. **Further Preprocessing:**

Depending on my specific research objectives and the machine learning algorithms we plan to use, we might need to perform additional preprocessing steps, such as handling class imbalances, creating validation sets, or using dimensionality reduction techniques like PCA.

So, we need to note that the exact steps and parameters may vary based on our specific research goals and the tools we are using. Make sure to adapt the preprocessing steps to our specific needs and the characteristics of the CICIDS2017 dataset.

4. Discuss the critical importance of data preprocessing in the context of cybersecurity datasets.

Data preprocessing is of critical importance in the context of cybersecurity datasets, while including the CICIDS2017 dataset, we have several key reasons:

1. **Data Quality and Integrity:** Cybersecurity datasets can often be noisy and contain errors, which can significantly impact the performance of intrusion detection systems (IDS) and other security applications. Data preprocessing helps clean and enhance the quality and integrity of the data by handling missing values, removing outliers, and correcting inaccuracies.

2. **Feature Selection:** Cybersecurity datasets can be high-dimensional, containing numerous features, many of which may not be relevant to the analysis or the problem at hand. Feature selection techniques in preprocessing help identify the most informative and important features, reducing dimensionality and improving the efficiency of intrusion detection models.

3. **Imbalanced Datasets:** In cybersecurity, imbalanced datasets are common, where the number of normal (benign) network traffic samples significantly outweighs the number of malicious samples. Preprocessing techniques, such as resampling (oversampling or undersampling), are crucial to balance the dataset, ensuring that the IDS does not favor the majority class and can effectively detect rare cyber threats.

4. **Handling Categorical Data:** Cybersecurity datasets often contain categorical features, such as protocols or attack types. These categorical variables need to be properly encoded (e.g., one-hot encoding) during preprocessing to be used effectively in machine learning algorithms, as most algorithms work with numerical data.

5. **Normalization and Scaling:** Data preprocessing includes normalizing and scaling features. Many machine learning algorithms, including distance-based methods, require that data is on the same scale. This normalization ensures that no single feature dominates the model's training, leading to more balanced and accurate results.

6. **Train-Test Split:** Proper data preprocessing involves splitting the dataset into training and testing subsets. This step is crucial for evaluating the performance of intrusion detection models. Without an appropriate train-test split, we risk overfitting and not having a reliable assessment of our model's generalization capabilities.

7. **Anomaly Detection:** Cybersecurity datasets are often used for anomaly detection, where the focus is on identifying unusual and potentially malicious behavior. Preprocessing helps in defining the characteristics of normal behavior, allowing the IDS to better detect anomalies and intrusions.

8. **Data Transformation:** In cybersecurity, various data transformation techniques may be employed, such as time series aggregation or extracting relevant statistical features from network traffic data. These transformations can help capture essential information and patterns for intrusion detection.

9. **Reducing Computational Overhead:** Large and unprocessed datasets can place a significant computational burden on intrusion detection systems. Data preprocessing can reduce the volume of data while retaining important information, making the system more efficient and responsive.

10. **Model Performance:** Ultimately, the success of intrusion detection systems and other cybersecurity applications depends on the quality of the data used. Proper preprocessing ensures that the data is in the most suitable form for training and testing, which can significantly impact the performance and accuracy of the models.

In conclusion, data preprocessing is a crucial step in the analysis of cybersecurity datasets like CICIDS2017. It enhances data quality, ensures feature relevance, handles class imbalances, and prepares the data for machine learning algorithms, all of which are vital for the successful development and deployment of intrusion detection and security systems in a dynamic and evolving threat landscape.

Part 3: Exploratory Data Analysis (EDA)

5. Conduct an exploratory data analysis to gain insights into data distribution, detect patterns, and understand feature characteristics.

Exploratory Data Analysis (EDA) is an essential step in understanding the characteristics of our dataset, uncovering patterns, and gaining insights. In the context of the CICIDS2017 dataset, here are some common EDA techniques to help we explore the data:

1. **Summary Statistics:**

Start by calculating summary statistics for the numeric features in the dataset to get a sense of the data distribution. We can use the `describe()` function in Pandas for this:

```
import pandas as pd

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

# Display summary statistics
print(dataset.describe())
```

[This will give us information about the mean, standard deviation, min, max, and quartiles of numeric features.]

2. Data Distribution:

Visualize the distribution of numeric features using histograms. We can use libraries like Matplotlib or Seaborn:

```
import matplotlib.pyplot as plt

import seaborn as sns


# Plot histograms for selected numeric features

sns.set()

plt.figure(figsize=(12, 6))

sns.histplot(dataset['feature_name'], kde=True)

plt.title('Distribution of Feature Name')  #We should replace `'feature_name'` with
                                                the name of the feature that we want to
                                        visualize.

plt.show()

```

3. Class Distribution:

In cybersecurity datasets, it's crucial to understand the distribution of classes, i.e., benign vs. malicious data points. Visualize the class distribution using a bar plot:

```
# Plot class distribution

plt.figure(figsize=(6, 4))

sns.countplot(data=dataset, x='target') # 'target' represents the class labels.

plt.title('Class Distribution')

plt.show()

```

4. Correlation Analysis:

A heatmap can help visualize the correlations, while checking for correlations between numeric features:

```
# Calculate correlation matrix
correlation_matrix = dataset.corr()

# Plot a heatmap of correlations
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()

```

[This helps identify which features are strongly correlated with each other.]

5. Box Plots:

Box plots can be used to visualize the spread and distribution of data, particularly for comparing the distribution of features among different classes:

```
# Box plot of a specific feature by class
plt.figure(figsize=(10, 6))
sns.boxplot(data=dataset, x='target', y='feature_name')
plt.title('Box Plot of Feature Name by Class')
plt.show()

```

6. Pair Plots:

For a deeper exploration of relationships between multiple features, we can create pair plots:

```
# Create pair plots
sns.pairplot(dataset, hue='target', markers=['o', 's'], height=2)
plt.suptitle('Pair Plots by Class', y=1.02)
plt.show()
```

[This can reveal interesting relationships and potential patterns within the data.]

7. Feature Value Counts:

For categorical features, examine the value counts to understand the distribution of categories:

```
# Display value counts for a categorical feature
print(dataset['categorical_feature'].value_counts())

```

This provides insights into the frequency of each category.

Through these EDA techniques, we can gain a better understanding of data distributions, identify potential outliers, detect patterns, and uncover insights into the characteristics of the CICIDS2017 dataset. This knowledge will be valuable when we proceed with building intrusion detection models or conducting further cybersecurity analysis.

6. Create visualizations and summary statistics to illustrate findings.

Visualizations and summary statistics are valuable for illustrating findings in the CICIDS2017 dataset. Here's my examples of how to create visualizations and present summary statistics which are helped me to better understand the dataset:

1. Class Distribution:

Let's start by visualizing the distribution of benign and malicious network traffic in the CICIDS2017 dataset. This will help us understand the balance between the classes.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

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

# Plot class distribution
plt.figure(figsize=(6, 4))
sns.countplot(data=dataset, x='Label')
plt.title('Class Distribution')
plt.xlabel('Class')
plt.ylabel('Count')
plt.show()
```

[This bar plot will show the number of instances for each class (e.g., benign and various types of attacks).]

2. Summary Statistics:

We can generate summary statistics to get a sense of the numeric features in the dataset:

```
# Display summary statistics for numeric features
numeric_summary = dataset.describe()
print(numeric_summary)
```

[This will provide statistics like mean, standard deviation, and quartiles for each numeric feature.]

3. Feature Distributions:

We can visualize the distribution of specific numeric features using histograms:

```

# Plot histograms for selected numeric features
numeric_features = ['Flow Duration', 'Total Fwd Packets', 'Total Backward Packets']
for feature in numeric_features:
    plt.figure(figsize=(12, 6))
    sns.histplot(dataset[feature], kde=True)
    plt.title(f'Distribution of {feature}')
    plt.xlabel(feature)
    plt.show()

```

[This code generates individual histograms for selected features.]

4. Correlation Heatmap:

To explore correlations between numeric features, create a heatmap:

```

# Calculate correlation matrix
correlation_matrix = dataset.corr()

# Plot a heatmap of correlations
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()

```

[This heatmap will help identify feature relationships.]

5. Feature Value Counts:

For categorical features, I searched for a displayed value counts to understand the distribution of categories:

```

# Display value counts for a categorical feature
categorical_feature = 'Protocol'
value_counts = dataset[categorical_feature].value_counts()
print(value_counts)
```

[This provides the counts of each category in the 'Protocol' feature, for example.]

6. Box Plots:

To examine the spread and distribution of data for different classes (benign and malicious), we create box plots:

```
# Box plot of a specific feature by class
feature_name = 'Flow Duration'
plt.figure(figsize=(10, 6))
sns.boxplot(data=dataset, x='Label', y=feature_name)
plt.title(f'Box Plot of {feature_name} by Class')
plt.xlabel('Class')
plt.ylabel(feature_name)
plt.show()
```

[This code generates a box plot for the 'Flow Duration' feature, comparing the distribution between benign and malicious instances.]

Part 4: Reporting

7. Prepare a detailed report covering all aspects of the assignment, including dataset details, data preprocessing, and EDA.

8. Include visualizations, tables, and concise explanations in your report.

In conclusion, my comprehensive Exploratory Data Analysis (EDA) of the CICIDS2017 dataset has provided valuable insights into the dataset's characteristics, by guiding the data preprocessing decisions. One important observation is the necessity of one-hot encoding for certain categorical features. This information will be valuable for making informed decisions when building intrusion detection models or conducting cybersecurity analysis. This conclusion is based on the following rationale:

Categorical Feature Transformation

In my initial dataset, I identified several categorical features, one of which is 'Protocol.' This feature represents various network protocols used in the network traffic data and contains non-numeric labels such as 'TCP,' 'UDP,' and 'ICMP.' To prepare the data for machine learning models, it is essential to transform these categorical labels into a numerical format. One-hot encoding is the chosen technique to achieve this transformation.

Before and After Transformation

To illustrate the impact of one-hot encoding, I'd examine a specific example with the 'Protocol' feature:

Before One-Hot Encoding

	Protocol
0	TCP
1	UDP
2	ICMP
3	TCP
4	UDP

After One-Hot Encoding

	Protocol_TCP	Protocol_UDP	Protocol_ICMP
0	1	0	0
1	0	1	0
2	0	0	1
3	1	0	0
4	0	1	0

By applying one-hot encoding to the 'Protocol' feature, I have created binary columns for each unique category, effectively transforming non-numeric labels into a numerical format. This transformation allows machine learning algorithms to work with categorical data and avoids introducing unintended ordinal relationships between the categories.

In shortly, the decision to perform one-hot encoding on specific categorical features, such as 'Protocol,' was a critical step in preparing the CICIDS2017 dataset for further analysis and machine learning. This transformation ensures that the categorical data is appropriately represented, maintaining the integrity of the information and supporting the development of effective intrusion detection models and cybersecurity analysis.

Submission Guidelines:

-Submit your assignment reports in PDF format. (70%)

-Include a GitHub Repository for Code-Script with extensive comments, along with a descriptive readme file that explains the code and its usage. Also include sample data. (30%)