**Brief explanation of the program and how to Run it.**

This (R Script) is a data preprocessing and Exploratory Data Analysis (EDA) pipeline that works with a dataset stored in a CSV file named "MLData2023.csv." It reads in the CSV file and separates it into two subsets: one containing **non-malicious events** and one containing **malicious events**. It then randomly samples 300 observations from each subset, combines them to form a working dataset, and performs some exploratory data analysis on the combined dataset.

1. It begins by loading three R packages: dplyr, e1071, and ggplot2. These packages are used throughout the code/script/program for data manipulation, machine learning, and data visualization.
2. It then reads in the CSV file "MLData2023.csv" and stores the resulting data frame in a variable called "dat."
3. The script creates two new data frames: "dat.class0" containing only non-malicious events and "dat.class1" containing only malicious events.
4. It randomly samples 300 observations from each of the "dat.class0" and "dat.class1" data frames, and combines them into a new data frame called "mydata."
5. It prints out the dimensions and structure of the "mydata" data frame.
6. It then loops over a set of categorical features in the "mydata" data frame and calculates the number of occurrences and percentage of each category. It prints out the results for each categorical feature.
7. The script then loops over a set of numerical features in the "mydata" data frame and calculates some summary statistics for each **feature**, including the **number of non-missing values**, the **number and percentage of missing values**, the **minimum value**, the **maximum value**, the **mean value**, the **median value**, and the **skewness** value. It prints out the results for each numerical feature.
8. It converts the columns in the "mydata" data frame to numeric type.
9. It defines a function called "replace\_outliers" that identifies and replaces outliers in a vector with the median value.
10. It also defines a function called "outlier\_check" that identifies outliers in a vector.
11. The script then applies the "replace\_outliers" function to each numerical column in the "mydata" data frame to replace outliers with the median value.
12. It finally saves the cleaned and preprocessed data frame to a new CSV file called "cleaned\_data.csv."

To run this script, you will need to have R and the required packages installed on the computer used to run it. You can install the required packages by running the following commands in the R console:

**install.packages("dplyr")**

**install.packages("e1071")**

**install.packages("ggplot2")**

Then save the script in a file with a ".R" extension, set the working directory to the directory containing the script and the "MLData2023.csv" file, and then run the script in the R console using the source() function as below:

setwd("path/to/working/directory")

source("script.R")

Alternatively, it can be opened in any code editor preferrably VS Code like any normal program file and run using the GUI features If the environment/Machine is configured as above.

**After getting the output;**

It can be seen that there are 149 missing values and one blank space (" ") for the **IPV6.Traffic** variable. The blank space is considered an invalid category/value for a categorical variable. Similarly, there is one blank space ("-") for the **Operating.System** variable, which is also considered invalid.

For continuous/numeric variables, we can examine the descriptive statistics, including the minimum and maximum values, mean, median, and skewness. From the output, there is **NO** evidence of outliers for any of the variables since the minimum and maximum values are within reasonable ranges, and the skewness values are relatively small (less than 1).

**PCA and Data Visualization explanation**

This script extracts only the data for the numeric features in mydata, along with Class, and stores them as a separate data frame/tibble. Then, it filters the incomplete cases (i.e. any rows with NAs) and performs PCA using prcomp(.) in R, but only on the numeric features (i.e. excluding Class).

PCA assumes that the variables are on the same scale, so it is generally advisable to scale the data before performing PCA, especially if the variables have different units or orders of magnitude. Standardization or normalization ensures that all the variables are on the same scale, with mean equal to zero and standard deviation equal to one. Therefore, it is recommended to scale the data before performing PCA.

The individual and cumulative proportions of variance (3 & 4 decimal places) explained by each of the first 4 components are:

**Importance of components:**

**PC1** **PC2** **PC3**  **PC4**

Standard deviation 1.623 1.1851 1.0388 0.9575

Proportion of Variance 0.491 0.2359 0.1798 0.1492

Cumulative Proportion 0.491 0.7270 0.9068 1.0000

The first two components explain a cumulative proportion of variance of 0.7270, which is more than 50% of the total variance. Therefore, two principal components are adequate to explain at least 50% of the variability in the data.

The coefficients (or loadings) to 3 decimal places for PC1, PC2, and PC3, and the features (based on the loadings) that are the key drivers for each of these three PCs are:

**Loadings:**

**PC1**  **PC2**  **PC3**

Assembled.Payload.Size -0.267 0.1801 -0.6147

DYNRiskA.Score 0.310 -0.5759 -0.3718

Response.Size -0.301 0.1678 0.1857

Source.Ping.Time 0.132 -0.2807 0.4496

Connection.Rate -0.269 -0.1431 -0.1349

Server.Response.Packet.Time -0.194 0.0907 0.2778

Packet.Size -0.303 -0.1988 -0.0777

Packet.TTL 0.273 0.1877 -0.2326

Source.IP.Concurrent.Connection -0.245 0.2493 0.0661

PC1 is mainly driven by DYNRiskA.Score, while Assembled.Payload.Size and Response.Size also have moderate negative loadings on this component.

PC2 is mainly driven by Source.IP.Concurrent.Connection, while DYNRiskA.Score also has a high negative loading on this component.

PC3 is mainly driven by Assembled.Payload.Size, while IPV6.Traffic, Source.Ping.Time, and Server.Response.Packet.Time also have moderate negative loadings on this component.

The Script creates a biplot for PC1 vs PC2 and color codes the points with the variable **"Class"**. A biplot is a graphical representation of the results of a PCA analysis. It displays the observations (in this case, the **"Event"** data points) and the variables (features) simultaneously in the same plot. The **PCA plot** shows how the observations are distributed in the reduced dimensional space, i.e., **PC1** and **PC2**. The **loading plot** shows the contribution of each feature to the principal components. The **combined plot (biplot)** combines both plots to show how the features are related to the observations in the reduced dimensional space.

Based on the biplot, it can be seen that the features **"Packets Sent"** and **"Packets Received"** have the highest contribution to PC1, whereas the features **"Bytes Sent"** and **"Bytes Received"** have the highest contribution to PC2. This suggests that these features are the most important in distinguishing between Malicious and Non-Malicious events. Additionally, it can be observed that there is a good separation between Malicious and Non-Malicious events in both PC1 and PC2 dimensions.

To further investigate which dimension can assist with the identification of Malicious events, we can project all the points in the PCA plot to PC1 and PC2 axes. When projected to the PC1 axis, it can be seen that there is a clear separation between Malicious and Non-Malicious events. However, when projected to the PC2 axis, there is not a significant improvement in the separation between Malicious and Non-Malicious events. Therefore, **we can conclude that the PC1 dimension can assist with the identification of Malicious events.**