Code Documentation

IDS-Data-Cleaning:

This script is written in Python and is used for analyzing network traffic data. The script begins by importing several libraries including numpy, pandas, matplotlib, seaborn, glob, os and scipy.stats. The script then sets the base path to the directory containing the csv files of the dataset.

The next section of the code sets the data types for each of the features in the dataset, which are specified in a dictionary called 'types'. The data types are specified as 'uint32', 'uint8', 'object', 'int64', 'float32', 'float64', 'uint16' and 'int64' for different features.

The script then uses the pandas library to read in the csv files from the dataset directory, and applies the specified data types to the features using the 'read\_csv' function and the 'dtype' parameter.

The next step is to use the 'describe' function to get the statistical properties of each feature. This is used to check the distribution of each feature, and to identify any outliers or unusual values.

This script is used for cleaning and preprocessing a dataset containing multiple CSV files. The dataset is the CICIDS 2018 dataset, which is a set of network traffic data that is used for intrusion detection. The script begins by importing the NumPy and Pandas libraries, as well as the os and re libraries.

The script starts by creating a dictionary named "csv\_files" that maps the original CSV file name to the output parquet file name after cleaning and preprocessing.

The variable "processed\_dir" is the name of the directory where the cleaned and preprocessed parquet files will be saved.

The script defines several functions such as remove\_null\_values, remove\_duplicates and clean to perform different cleaning tasks on the dataframe.

The remove\_null\_values function takes a dataframe as input and drops all the rows that have a null value. The remove\_duplicates function takes a dataframe as input and drops all the duplicate rows from the dataframe. The clean function takes a dataframe as input, it loops through the columns in the dataframe and replaces any infinite values with NaN. Then it replaces the NaN with the mean of the column.

The script then checks if the processed\_path directory exists and if it does not, it creates it.

The script then iterates through the csv\_files dictionary and for each file, it reads the contents of the file into a dataframe, drops unnecessary columns, applies the cleaning functions, and then saves the cleaned dataframe as a parquet file in the processed\_path directory.

Finally, the script uses the 'glob' library to list all the files in the dataset directory, and uses the 'os' library to check the file size of each file. This is used to check whether the dataset is complete and to identify any missing files.

IDS-2C&7C&15C:

This script is a machine learning pipeline for intrusion detection using the CICIDS 2018 dataset. It begins by importing various libraries including NumPy, Pandas, Matplotlib, Seaborn, Plotly, scikit-learn and Keras. The script sets the base path for the directory containing the CICIDS 2018 dataset in the form of parquet files and uses the glob library to find all parquet files and concatenates the contents of these files into a single DataFrame, network\_data. It then defines a function change\_label that replaces attack values in the 'label' column of the DataFrame with the value 'Malicious', this function is applied to the network\_data DataFrame. After that the script initializes a LabelEncoder object and uses it to encode the labels in the 'label' column of the DataFrame. The script then splits the dataframe into training and testing data and extracts the label column to use as target variables. It removes the 'timestamp' and 'label' columns from both the training and testing data. The reason for this is to eliminate the possibility of time-based data leakage, and to separate the features and target variables. The script then uses the to\_categorical() function from the Keras library to convert the target training data into a categorical format, with 2 classes. It also performs a similar operation for the target testing data. The script then initializes a StandardScaler object from the scikit-learn library and uses it to scale the training and testing data to a range of 0 to 1. This is done to improve the performance of some machine learning models, such as neural networks. Last but not least, the data is trained using xgboost, random forest, cnn, and dnn. The performance is measured at last using confusion matrix and classification report.