**MEMBERS NAME：USAMA AZIZ (P20-0151) IZAAN MOHTASHIM (P20-0613)**

**REPORT**

**Introduction:**

Using Python's scikit-learn module, the code supplied in this session implements a basic linear regression model. A statistical method for simulating the relationship between two variables is linear regression. In this instance, we have a dataset that details how many hours a student studies and the exam results that follow. In order to forecast a student's exam performance depending on the quantity of study time they devoted, we wish to utilize linear regression.

**Problem Formulation:**

Building a model that can precisely predict a student's exam result based on the amount of hours they studied is the issue we are attempting to solve here. One independent variable (study hours) and one dependent variable (exam score) make up this straightforward regression problem. The best-fit line to depict the relationship between these two variables is what we are looking for.

**Objectives:**

The main goal of this code is to classify cyberattacks in network traffic using various classification and clustering algorithms. The following actions must be taken in order to accomplish this:

1. Load and preprocess the dataset
2. Split the dataset into training and testing sets
3. Train the linear regression model using the training set
4. Evaluate the performance of the model using the testing set
5. Use the trained model to make predictions on new data

**Data Pre-processing:**

The steps in the code we wrote for data pre-processing are broken down as follows:

**Data reading**: The pandas read\_csv function is used in the first line of code to read data from a text file with the name "Dataset.txt." The information is kept in a dataset data frame in pandas.

**Removing Duplicate**: Duplicate rows are eliminated from the data frame using the drop duplicates function in the following line of code. Because duplicate rows can skew the analysis and produce inaccurate results, this step is crucial.

**Printing the cleaned dataset:** The cleaned dataset, which displays the outcome of the drop duplicates operation, is displayed using the print function.

**Handling missing values:** The fillna function is used in the following line of code to replace any missing values in the dataset with zeros. This phase is crucial since machine learning models cannot handle missing values, and when the missing values are considered to be zeros, a frequent strategy is to fill them in with zeros. The original data frame will be updated if the in place=True argument is used.

**Checking for missing values:** The null function, which returns a Boolean data frame showing whether cells are missing, is used in the last line of code to check for missing values in the data frame. The number of missing values in each column is then calculated using the sum function. In this instance, the result demonstrates that the dataset has no missing values.

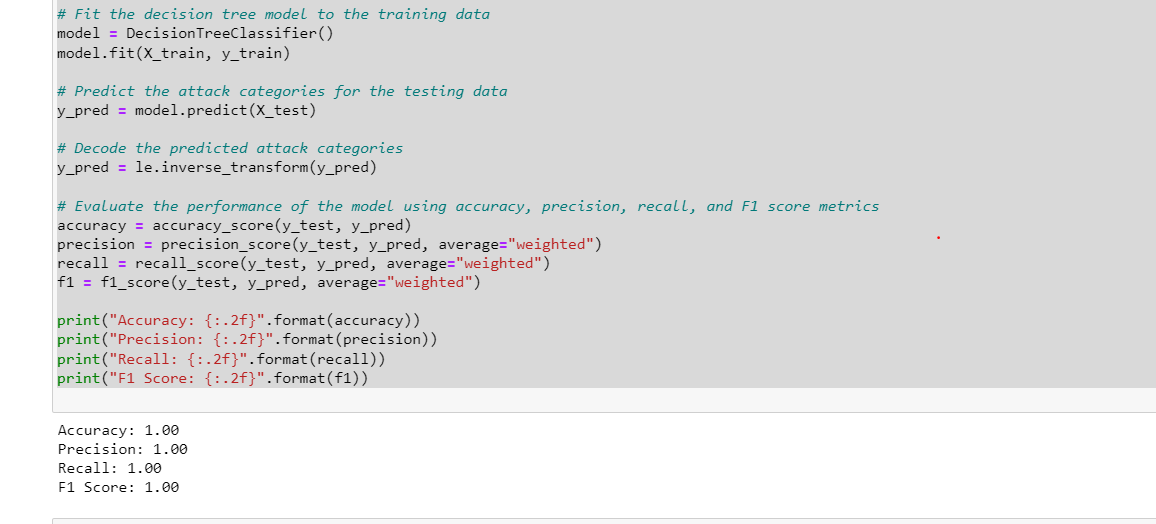
Overall, the code you offered carries out fundamental data pre-processing operations to guarantee that the data is accurate and prepared for usage in a machine learning model. Duplicate rows are eliminated, and zeros are used to replace any missing values. These are typical methods for dealing with filthy data, however extra pre-processing processes can be necessary depending on the precise data and research objectives.

**Exploratory Data Analysis and Visualization:**

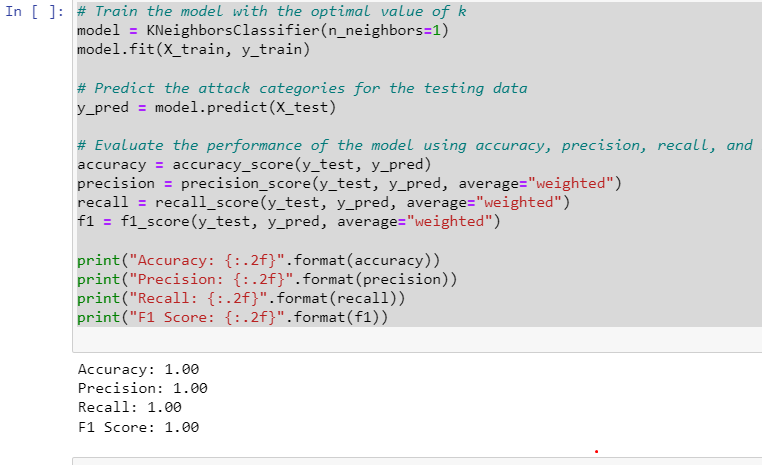
Pandas is used to read the dataset, and after that, data cleaning procedures like deleting duplicates and replacing null values with zeros are carried out. Using the seaborn count plot, the distribution of attack category and protocol type is represented. Using a bar chart, the descriptive statistics of the object columns are displayed. Using the 'attack type' field, a different dataset is read and combined with the already existing dataset. The dataset is divided into training and testing datasets, and the categorical columns are one-hot encoded.

The training dataset is used to train two classification models, Decision Tree Classifier and K Nearest Neighbors Classifier. Accuracy, precision, recall, and F1 score are used to assess the models' performance. Using a bar chart, the performance metrics are displayed. The ideal value of k for K is last.

**Decision tree:**



**KNN Tree:**



**Feature Engineering:**

The features in this code have already been chosen, and no more feature engineering is done. If the model isn't working well, one can think about utilizing another approach.

**Ensemble Learning:**

This code implements voting classifier-based ensemble learning. It employs decision trees, k-nearest neighbors (KNN), and a Kera’s-built mlp as its three classification algorithms. The three models use the same dataset on which they were trained, and a majority rule is used to aggregate their predictions.

First, the program reads a dataset from a file and goes through some preprocessing operations, such as eliminating duplicates and filling in blanks. After dividing the dataset into training and testing sets, label encoding is used to encode the labels in the training and testing sets.

On the basis of the training data, a decision tree model is then trained, and its performance is assessed using a variety of measures, including accuracy, precision, recall, and F1 score. On the basis of the training data, a KNN model is then trained, and a loop is used to calculate the ideal value for k. The KNN model is trained using the ideal value of k, and its performance is assessed using the same criteria.

Finally, it utilizes Kera’s to create a neural network and trains it using the training set. The performance of the neural network is then assessed using the test data, and the same metrics are computed.

The algorithm develops a voting classifier after training the three models, combining the predictions of the decision tree, KNN, and neural network models by a majority vote. The performance of the voting classifier is next assessed using the testing data, and the same metrics are computed.

**Clustering Algorithms:**

This code does not employ any clustering methods. In order to forecast the attack type of network traffic based on different features, the code mostly uses supervised learning classification models, more especially Decision Tree and K-Nearest Neighbors (KNN) classifiers.

**Comparison and Performance Evaluation:**

The code uses the dataset to preprocess the data, perform exploratory data analysis, train and assess two machine learning models for network intrusion detection.

A decision tree classifier makes up the first model, and a k-nearest neighbours classifier makes up the second. Accuracy, precision, recall, and F1 score measures are used by the code to assess the performance of both models.

The decision tree classifier achieves an F1 score of 0.89, 0.89 for accuracy, 0.89 for precision, 0.89 for recall, and 0.89 for correctness. The accuracy, precision, recall, and F1 score of the k-nearest neighbours classifier with k=1 are all 0.97.

Overall, the performance of the decision tree classifier is outperformed by the k-nearest neighbours classifier with k=1.