**Artificial Intelligence Project Report**

**Group Members**

Farhan Abbasi (20P-0044)

Usama Yazdani (20P-0598)

**Section**

BCS-6C

**Introduction**

This AI project is about running the data set on different classification algorithms. The data set deals with the types of cyber attacks having 43 feature columns and 125973 rows in them. The name and types of attacks are provided in a separate data set. The data set will be pre-processed which includes data cleaning and handling unnecessary details. After the pre-processing, the data in the form of vectors will be passed to the classifiers to be trained and produce predictions. The metrics score will be calculated based on predictions produced by the classifiers through which the performance of the classifiers can be evaluated.

**Executive Summary**

The cyber attack data set consists of two files one is the actual data set which needs to be trained and the other file contains the target label. The data visualization and cleaning are performed using Python’s pandas library.

Correlation analysis of features is done against the target label to find out which features are more contributing to the target label.

After the data is preprocessed and cleaned, the data is split into training and testing data with 70 training and 30 percent testing data.

After splitting, 4 different classifying algorithms are being used which are listed below

**Decision Tree**: it is a type of classification algorithm used for the supervised data set. It produces a tree-like structure in the processing of the data set to reach a specific label.

**K-Nearest Neighbour**: This classification algorithm is a nonparametric supervised learning algorithm that classifies the data points based on the closest K number of data points.

**Artificial Neural Network (MultiLayer Perceptron)**: it is a supervised machine learning algorithm that has multiple layers of interconnected perceptrons which compute the weight of its inputs and apply an activation function to it (sigmoid, ReLu, etc)

**K-Means Algorithm**: It is a clustering-based machine learning algorithm used for unsupervised data sets. It clusters similar data points together based on the features given to it.

The Metric Scores (Accuracy, F1 Score, Recall Score, Precision Score) is being calculated for the first three algorithms i.e Decision Tree, KNN, and ANN, and results are visualized in the line graph

**Libraries Used**:

Numpy

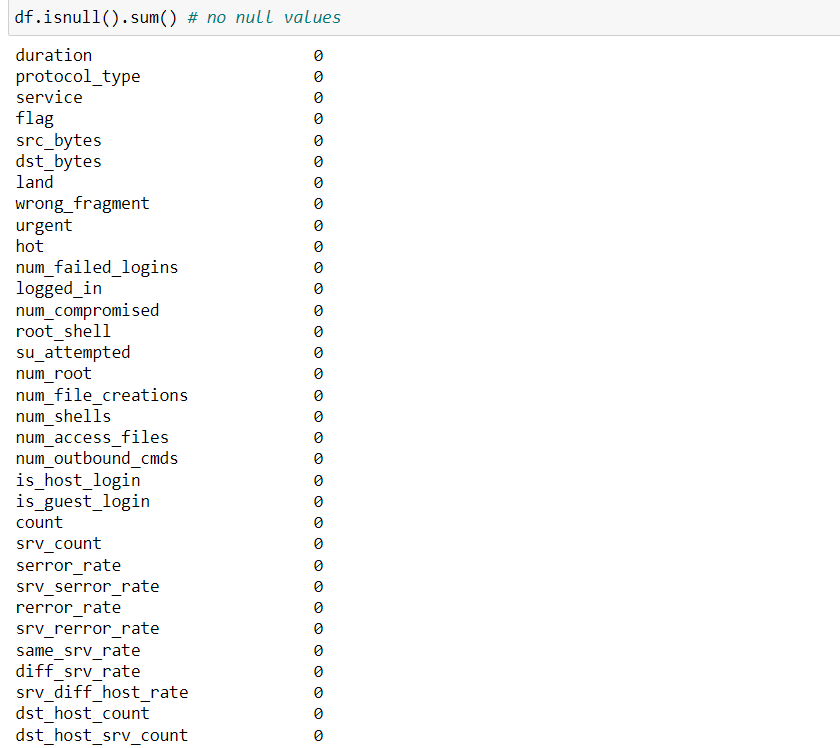
Pandas

Matplotlib

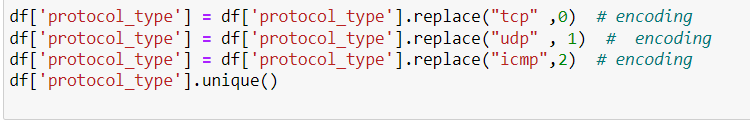
Sklearn

**Data Preprocessing (Extraction and Cleaning)**

In data preprocessing, we used the pandas library for the visualization of the data set. The data set contained no null values.



Since there were no null values, we encoded the categorical values and string values. The column ‘protocol\_type’ contained 3 categorical values, so they were being encoded to [0,1,2]



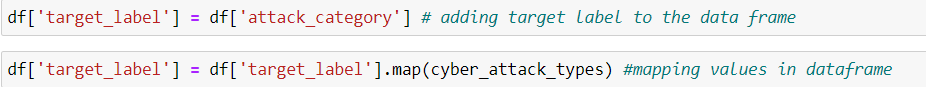
And other columns i.e ‘flag’ and ‘service’ containing string values were replaced by numerical values



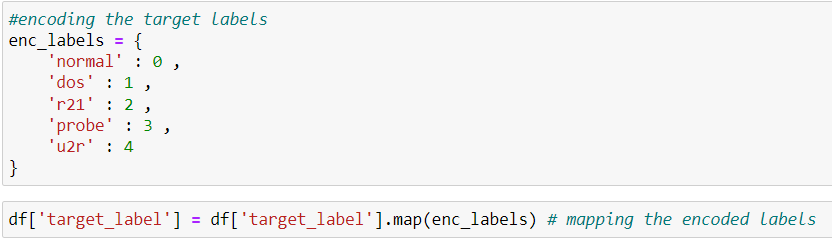


Since target labels were in different text files, a separate dictionary was made containing target labels of the data set which were mapped against the column ‘attack\_category’ which contained the names of the attacks.





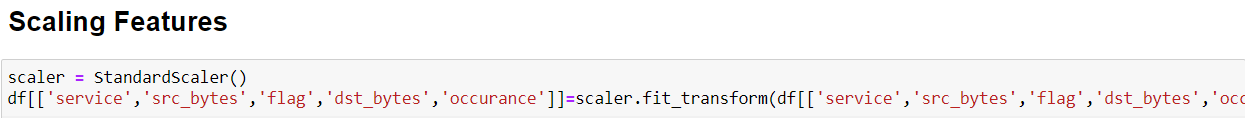
After mapping, the encoding of labels was performed into 5 categories and mapped to the target label



**Scaling:**

The feature having large values are downscaled for better accuracy.

Columns 'service', 'src\_bytes' 'flag', 'dst\_bytes', and ‘occurrence' are standard scaled using a built-in scalar function from the sklearn preprocessing library.



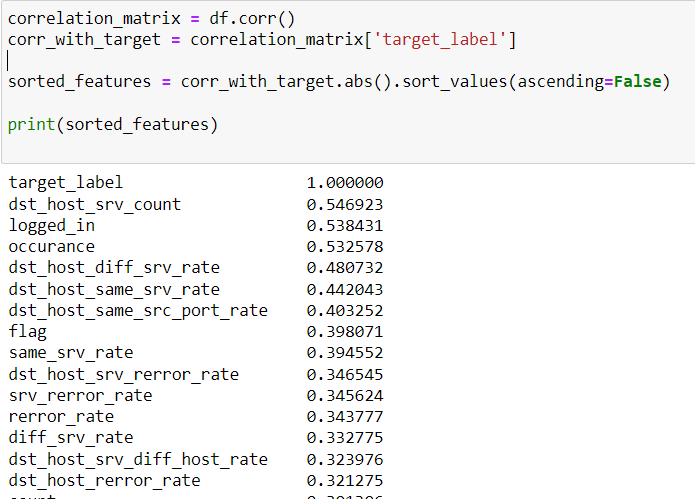
**Feature Engineering**

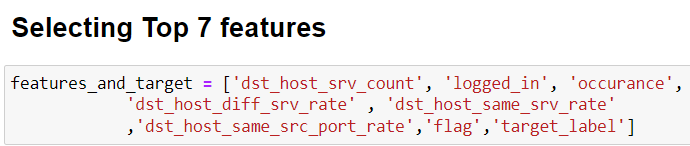
The feature engineering Is done using correlation analysis.

**Correlation Analysis:**

Correlation analysis was performed to find out the importance of the features helping to determine the target label. Feature correlation values are sorted in descending order. We picked the top 7 features for the model training.

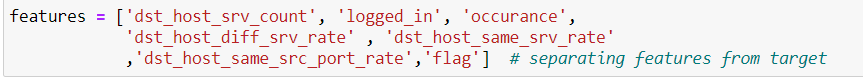
All other features are dropped from the data set to reduce the data set.







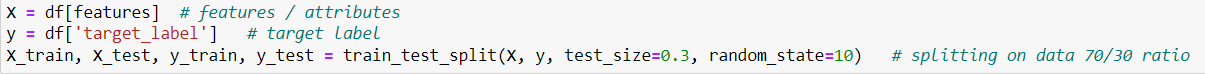
These are the selected features



The data is now cleaned!

**Use of Classification And Clustering Algorithms**

Before passing the data into the model we first need to split the data into training and testing data. Our split ratio was 70/30 i.e. 70 percent training and 30 percent testing data with a random state of 10 for shuffling data to prevent biases.

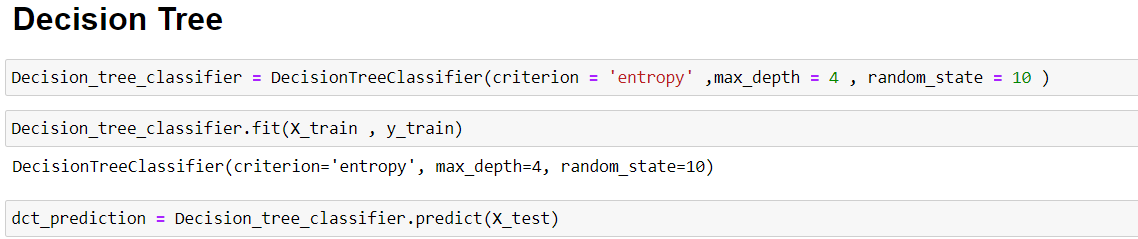


**Decision Tree Algorithm**

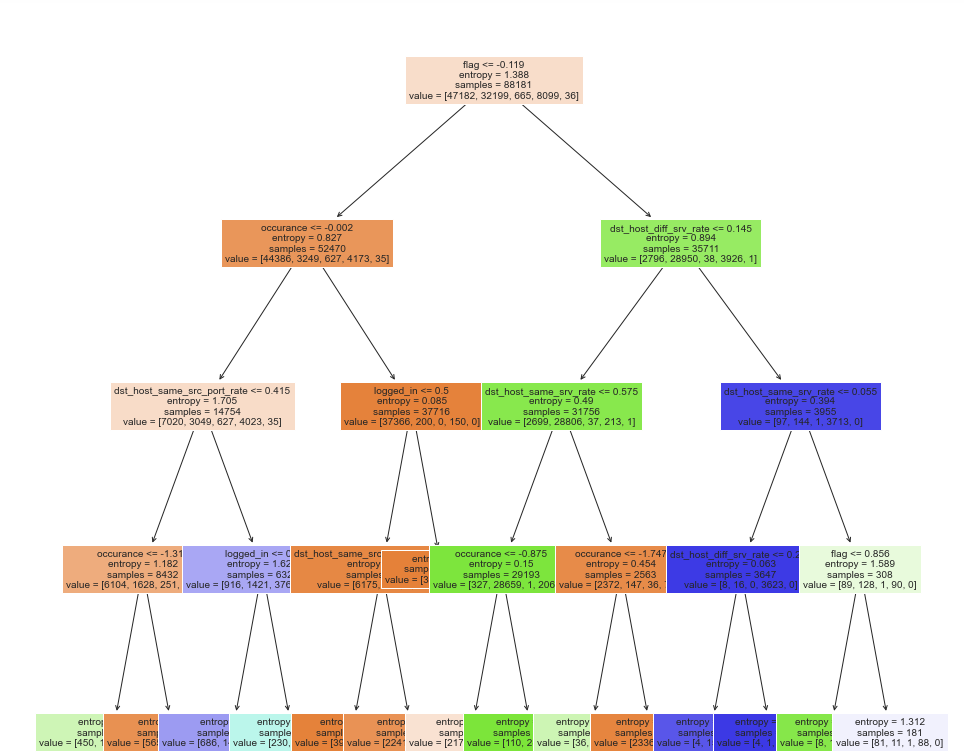
The decision tree algorithm is implemented on the data using the sklearn decision tree classifier. it is a supervised learning model, that makes a tree based on information gain and selects the nodes in a tree accordingly.

An object for the decision tree is made with parameter values passed to it. The criterion was set to ‘entropy’ which measures the impurity of a node, max depth = 4 which means a decision tree can have the maximum level of 4, and random seed = 10 for shuffling.

Now X\_train and y\_train are passed to the fit function of the object which trains the model and predictions are generated.



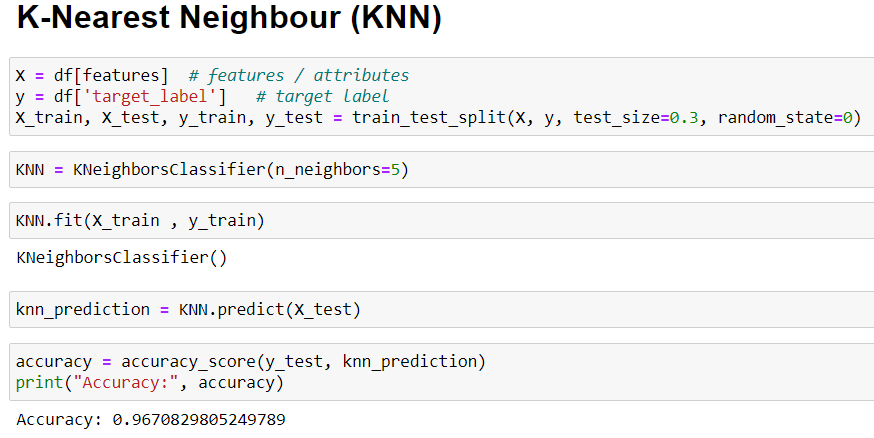
A visual representation of a tree is generated using matplotlib where the root is ‘flag’ having the highest information gain.



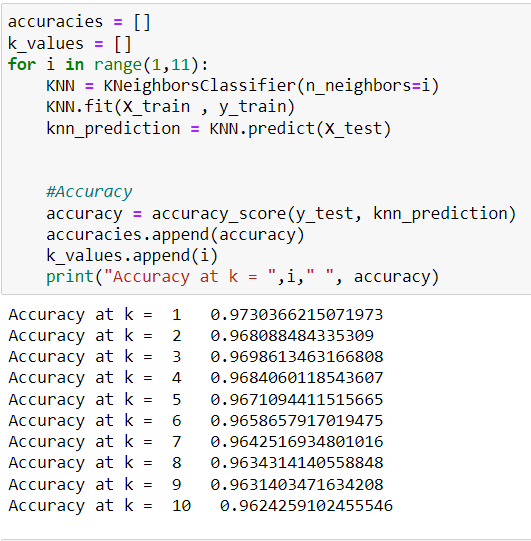
**K-Nearest Neighbour**:

This classification algorithm is a non-parametric supervised learning algorithm that classifies the data points based on the closest K number of data points.

Repeating the same process as above, we create an object of the KNN model and initially, we take k = 5, meaning 5 nearest neighbors of a data point. Now we trained the data on the fit function. The accuracy was 0.967



Now we searched for the optimal value of K, in a loop and trained the data on different values of K ranging from 1 to 10, and accuracy was calculated on each iteration



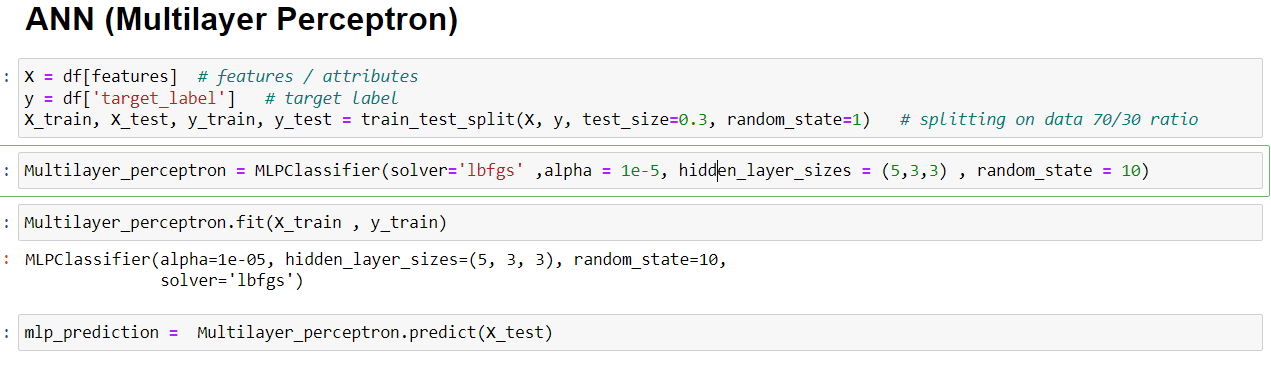
We cant take k= 1 because setting k to 1 can result in overfitting to the training data and make the model more susceptible to noise and outliers in the data. Increasing the value of k can improve the generalization ability of the model, but it may also result in a higher bias and lower model complexity

So we took k = 3, the second highest accuracy result.

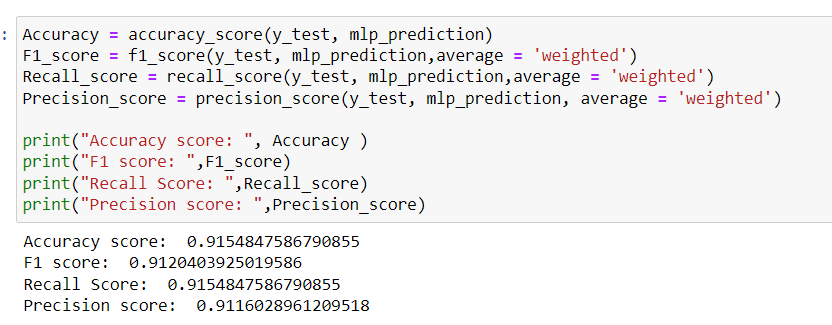
**ANN (Multi-Layer Perceptron):**

After splitting data, we made an object of the Multilayer Perceptron and adjusted the values of the hyperparameters as follows: solver = ‘lbfgs’ which is an optimizer of the quasi-newton family for the weights that are calculated, and alpha = 1e-5 which is a very small value, it is a regularization term which prevents the data from overfitting.

hidden\_layer\_sizes = (5,3,3) which means we have 3 layers the first one will have 5 perceptrons and the other two have 3 perceptrons each. After that data is trained and predictions are made.

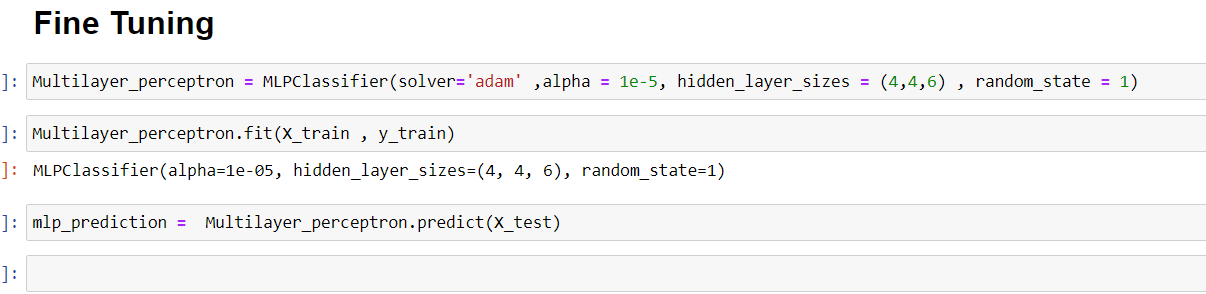


The following metrics scores were calculated after the above training:

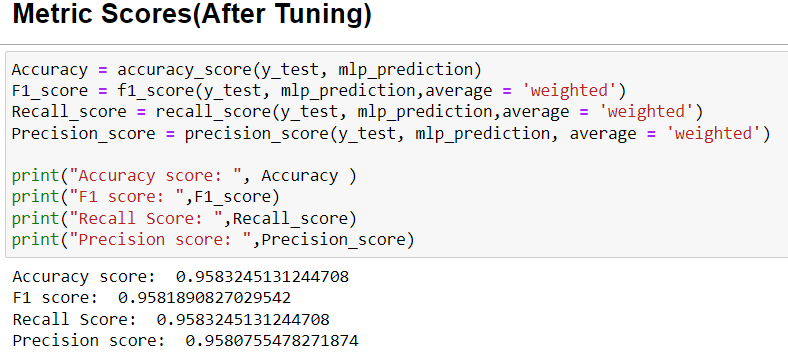


**Fine Tuning:**

We changed the hyperparameters to fine-tune the model. Solver = ‘adam’ which is a more accurate optimizer and hidden layer sizes were changed to (4,4,6).



The metrics score changed from 0.90 to 0.95

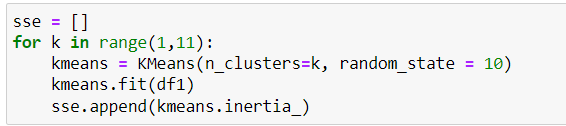


**K-Means Clustering Algorithm:**

We first dropped the target column from the data set.



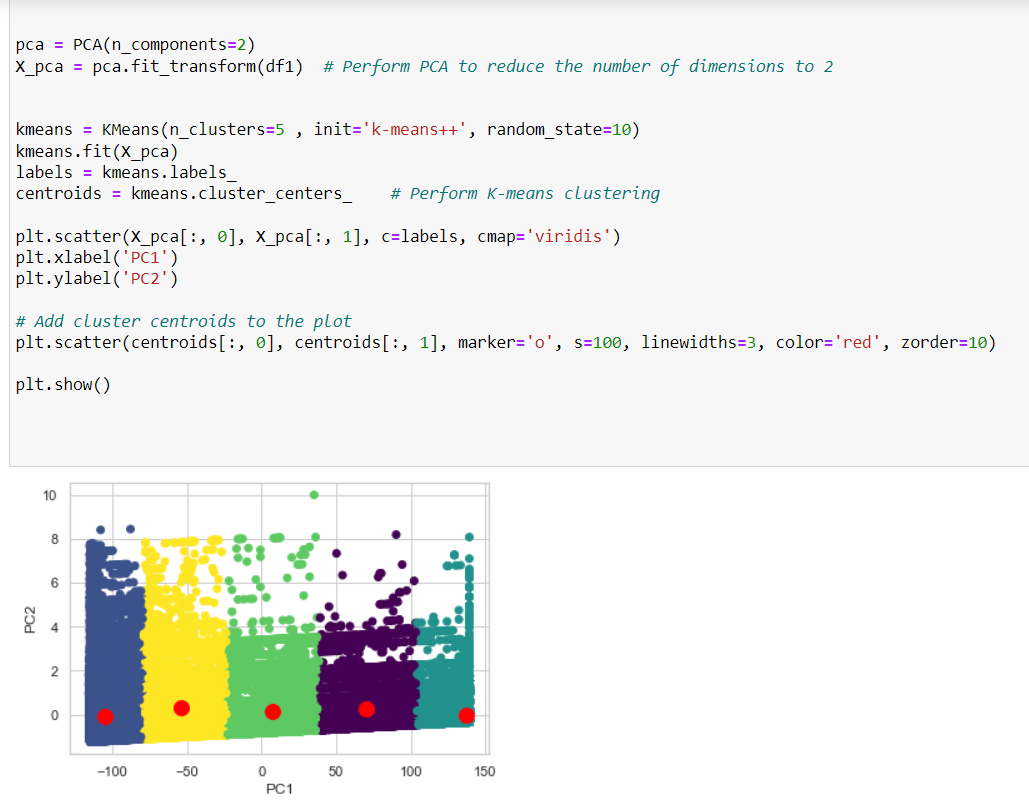
We made a list of sse (Sum Of Squared Error) and appended means inertia which is a metric used to evaluate the quality of clusters produced by the k-means algorithm. It is a measure of how internally coherent the clusters are. We ran a loop a from 1 to 10 which was the value of k(clusters)



Now using the elbow method we plot a graph for the optimum value of K where the error is less and selected the value of K.



Since we had 7 features we performed PCA (Principal Component Analysis) to reduce the dimension of the features to 2 so they can be plotted in a scatter graph. Where red dots are centroids which represent the center of the cluster

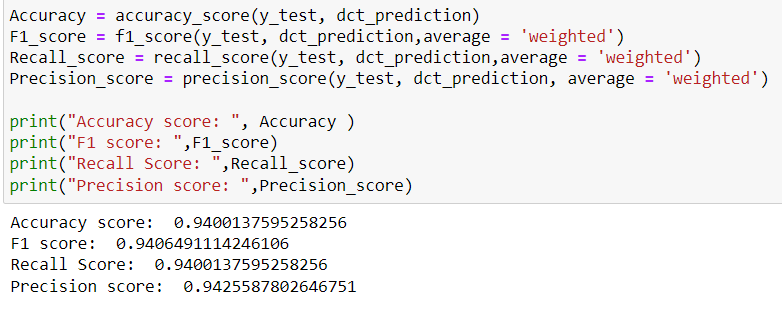


**Comparison and Performance Evaluation**

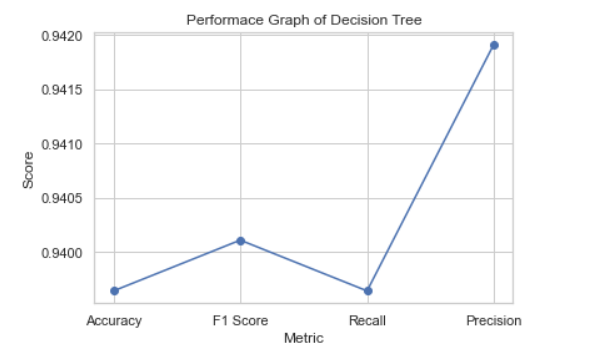
**Performance Evaluation of Algorithms**

Accuracy, F1 Score, Recall Score, and Precision Score are calculated based on predictions produced by the models.

**Decision Tree:**

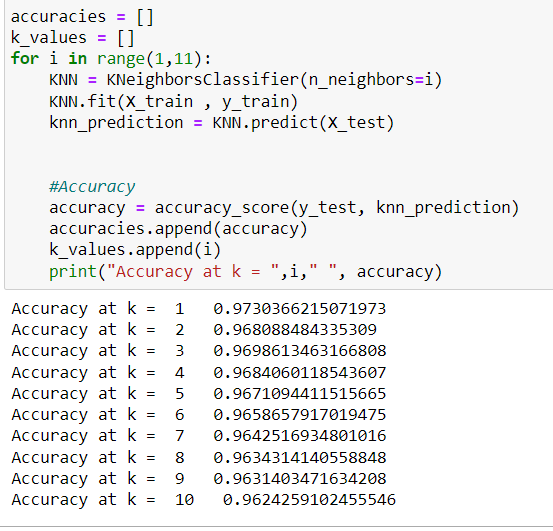


**Line Graph:**

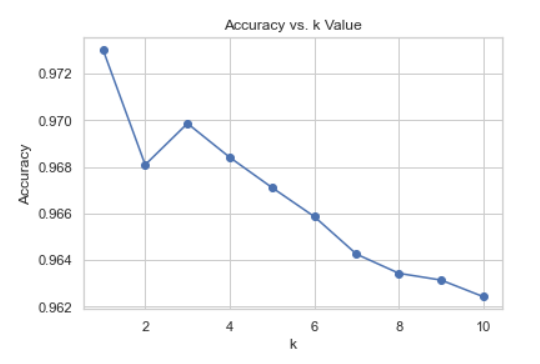


**KNN:**

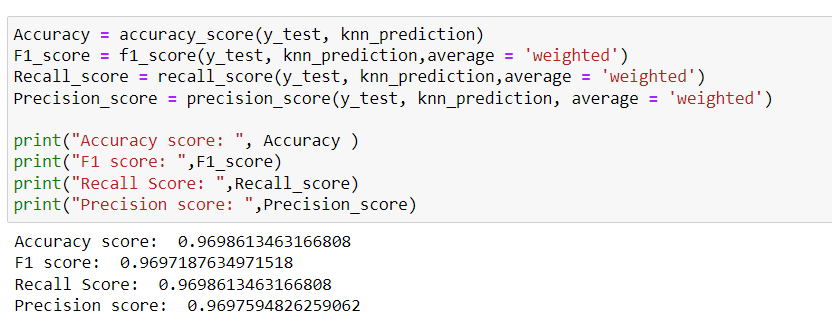
Accuracies of different values of K



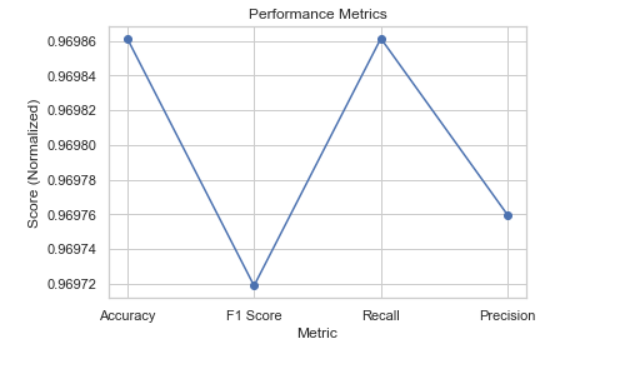
**Line Graph:**



**Performance on K = 3:**

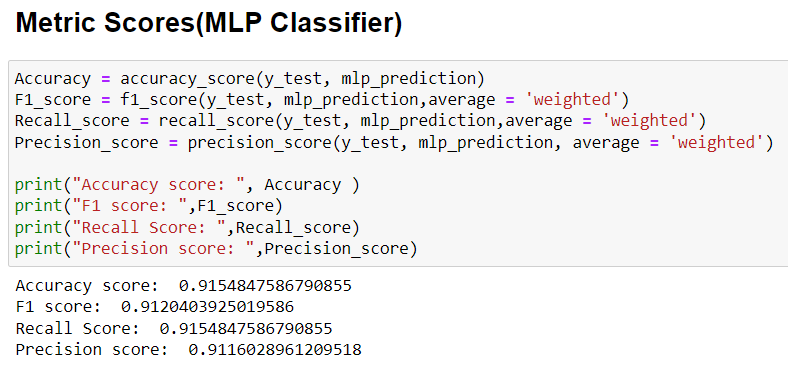


**Line Graph**:

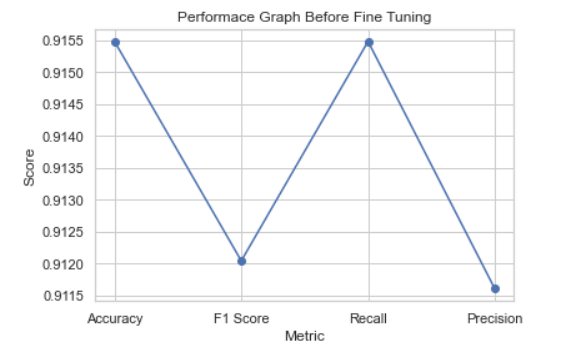


**Multi-Layer Perceptron:**

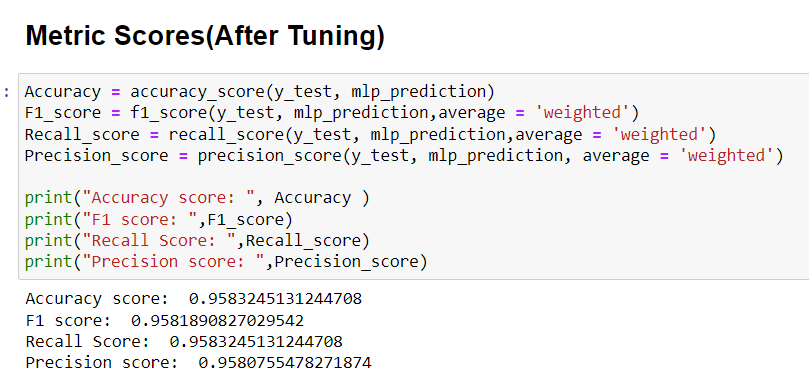
Before fine Tuning



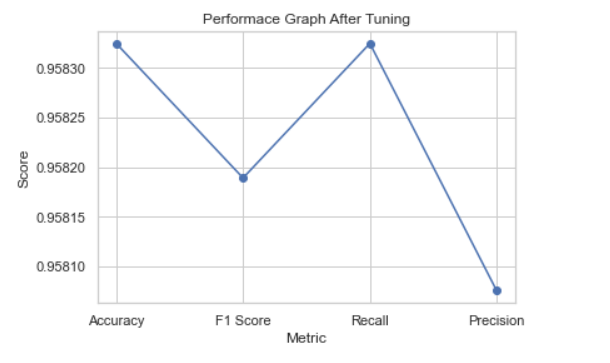
**Line Graph:**



**After fine-tuning:**

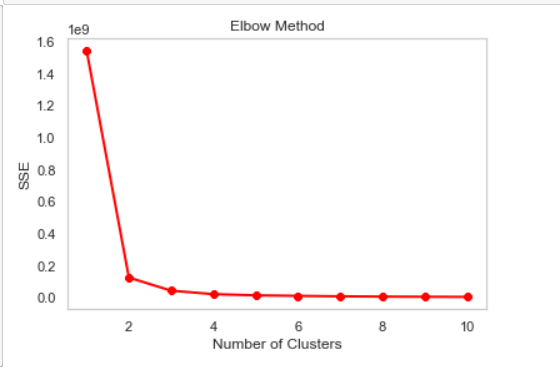


**Line Graph:**

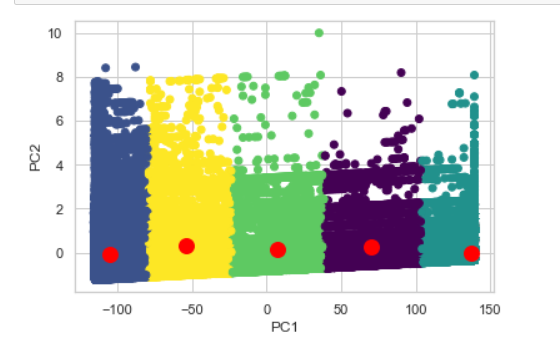


**KMeans :**

At an optimum value of K, the error started reducing



So at k= 5, there is no error, and clusters are made from the data points.



**Comparison Table for Algorithms:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithm | Accuracy | F1 Score | Recall Score | Precision Score |
| Decision Tree | 0.9396433107535986 | 0.9401115580304753 | 0.939643310753598 | 0.9419162525026081 |
| KNN at K=3 | 0.9698613463166808 | 0.9697187634971518 | 0.9698613463166808 | 0.9697594826259062 |
| MLP(Before Tuning) | 0.9154847586790855 | 0.9120403925019586 | 0.9154847586790855 | 0.9116028961209518 |
| MLP(After Tuning) | 0.9583245131244708 | 0.9581890827029542 | 0.9583245131244708 | 0.9580755478271874 |

**Conclusion**

we can conclude that the KNN algorithm with k=3 achieved the highest accuracy score of 0.9699, as well as the highest F1 score, recall score, and precision score, indicating that it performed the best out of the four algorithms tested in this experiment.

The Decision Tree algorithm also performed well with an accuracy score of 0.9396, but its scores were lower than those of KNN in all other metrics.

The MLP algorithm before tuning had the lowest accuracy score of 0.9155 and the lowest scores in all other metrics, indicating that it performed the worst out of the four algorithms. However, after tuning, its accuracy score increased significantly to 0.9583, and its F1 score, recall score, and precision score also improved, indicating that it could potentially perform better with further optimization.

In summary, KNN with k=3 performed the best out of the four algorithms tested in this experiment, while the MLP algorithm showed promise after tuning, but still did not perform as well as KNN. Decision Tree performed relatively well but was outperformed by KNN.