# CSE 4020 - MACHINE LEARNING

# Lab 29+30

# Lab Assignment-4

# Submitted by: Alokam Nikhitha(19BCE2555)

**MLP**

**Question:**

To train a Multi-Layer Perceptron (MLP) model to classify the network traffic record whether it is a normal or attack…

1. Read and parse the dataset.

2. Create Multi-Layer Perceptron Model (MLP)

3. Train and evaluate a Multi-Layer Perceptron (MLP) model

**Dataset Used:**

NSL KDD – Intrusion Detection Dataset <https://www.unb.ca/cic/datasets/nsl.html>

**Procedure:**

-Using pandas, we first import the dataset into our workspace.

-Assign the column names to our dataset as it doesn’t have one.

- Pick out and encode our specific variable.

- After encoding the specific variable, we want to dummy encode them on the way to keep away from ordinality among nominal information.

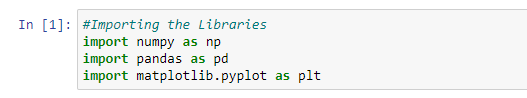
- We then want to re-assign our label information. All labels different than ordinary are assigned as attacks.

- We then want to divide the schooling set and check set information into set of structured attributes and impartial attributes.

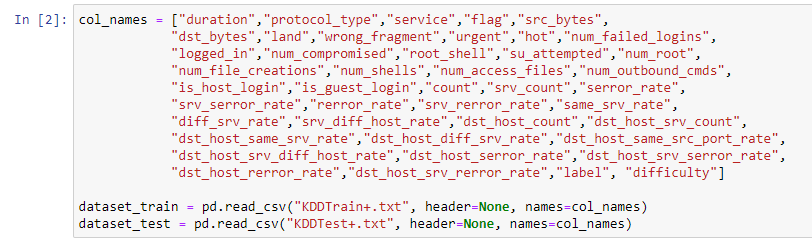
- Next, we lay down the Multi-Layer Perceptron and byskip our enter records into enter layer of our neural network.

- Finally, we generate our check set consequences and evaluation metrices.

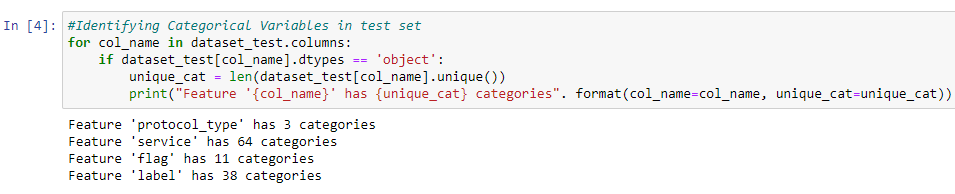
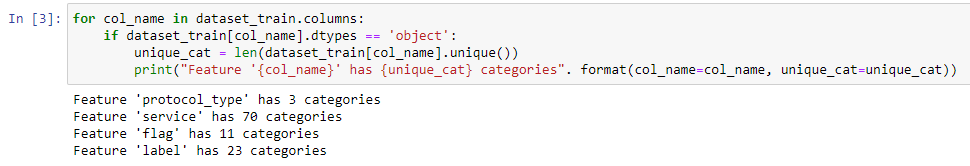
**Code Snippets and Explanation:**

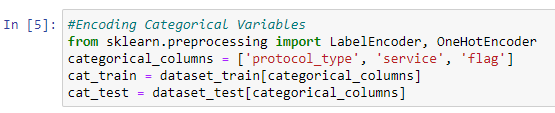


Here we are importing the necessary libraries in our workspace.

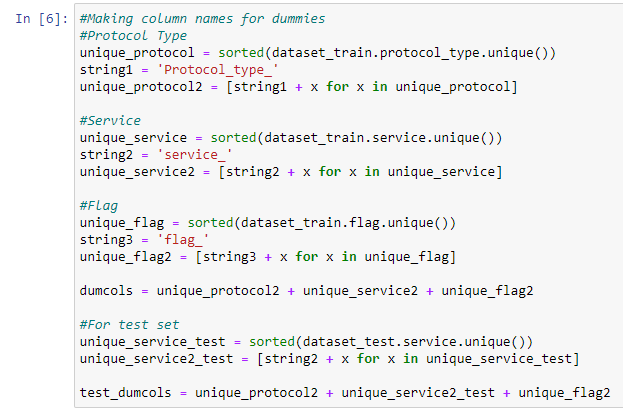


Here we're uploading the dataset into our workspace and are assigning them with the column names because it isn't always pre-blanketed in the given dataset.

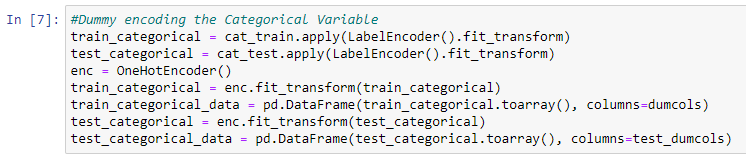


Here we've identified all of the express attributes in our training set and take a look at set. We have additionally identified the range of classes inculcating inside every attribute

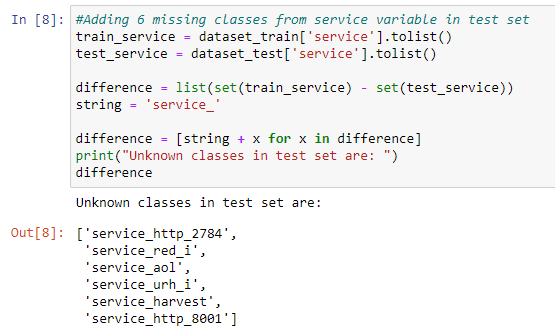
Here we have created 2 dummy data frames to include the categorical attributes in them



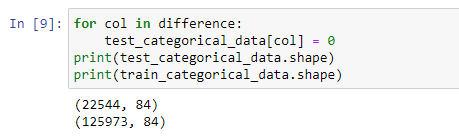
Here we've created the dummy attributes to keep away from the ordinal introduction among those nominal specific attributes.



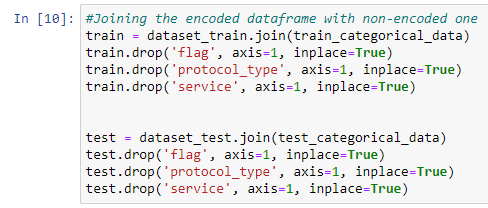
Here we've used the label encoder to fill withinside the dummy attributes in every of the specific attributes.



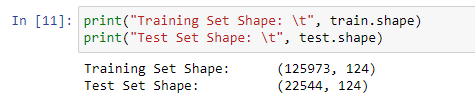
While checking the specific variables we noticed that service characteristic in check set has 70 training whilst schooling set has sixty four training. Hence, we want to encompass the ones 6 dummy attributes with zero fee in every our schooling set. This is what we've got diagnosed and achieved here.



Here we have finalised our data frames with the dummy values of categorical attributes.

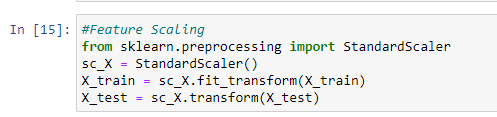
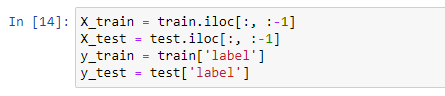
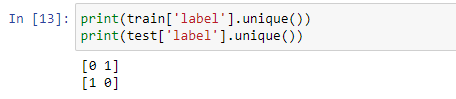


Next, we've got combined our original dataset with dummy attributes that we acquired in our specific assignment. Also right here we've got dropped the original specific attributes for you to inculcate only the non-specific attributes.

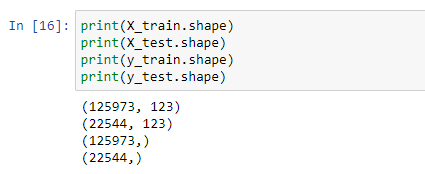


Here we have checked the number of attributes in both the training set and test set to see if they are equal… we can see that they have 124 attributes each and hence are compatible.

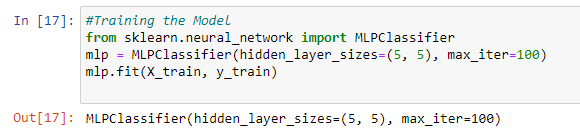
Here we have categorised each of the label attribute either as “normal” or an “attack”. All the labels which are normal are given a label of 0 and all those that indicate an attack are labelled as 1.



Since all the attributes in our dataset don’t follow a common scale, we need to feature scale the dataset in order to avoid any preassumed weight amongst them. We have used standard scalar to do this and it scales down each attribute to a range in -1 to 1.

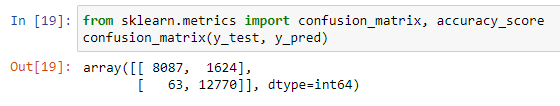


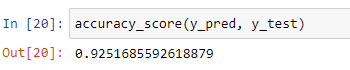
Here we have assigned the set of dependent and independent attributes. Also, we have printed the shape of each category that we have in order to check if they are compatible with each other.



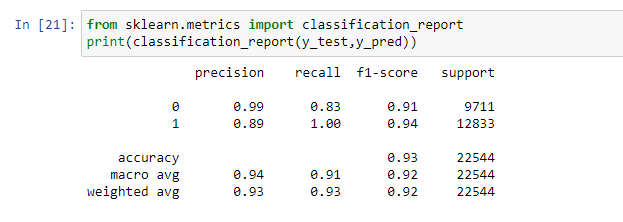
Here we have laid our neural network and then passed our input and output set to it in-order for it to adjust the weight biases.





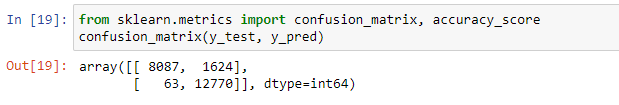
Here we have generated a vector y\_pred that stores the result as predicted by our mlp classifier on test set. We have also generated the confusion matrix to check the performance of our classifier.

we have printed the accuracy of our model and printed the classification reported to finally check the performance of our model. We can see that the accuracy of the model is 92.51%.

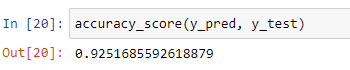


**Result and Conclusion:**

Confusion Matrix:



Accuracy:



Accuracy:92.51%.

* Identified Normal = 9711
* Actual Normal = 9362
* Identified Attack = 12833
* Actual Attack = 13182
* True Normal = 8087
* True Attack = 12770
* False Normal = 1624
* False Attack = 63
* Precision Normal = 0.99
* Precision Attack = 0.89
* Recall Normal = 0.83
* Recall Attack = 1.00

**K Means**

**Question:**

Illustrate the k-means clustering to cluster the data points for at least five epoch properly.

How to Implementing K-Means Clustering?

• Using the elbow method to determine the optimal number of clusters for kmeans clustering

• Visualising the clusters

• Plotting the centroids of the clusters

**Dataset Used:**

• Shopping-data (Uploaded in MS Team)

• <https://archive.ics.uci.edu/ml/machine-learning-databases/>

**Procedure:**

- Import necessary libraries - sklearn, numpy, pandas, etc.

-Using pandas, we first import the dataset into our workspace.

- Select the number of clusters for the dataset ( K )

- Select K number of centroids

- By calculating the Euclidean distance or Manhattan distance assign the points to the nearest centroid, thus creating K groups

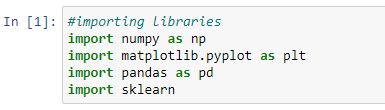
- Now find the original centroid in each group

-  Again reassign the whole data point based on this new centroid, then repeat step 4 until the position of the centroid doesn’t change.

- Using the elbow method to determine the optimal number of clusters for kmeans clustering

- Visualising the clusters and Plotting the centroids of the clusters

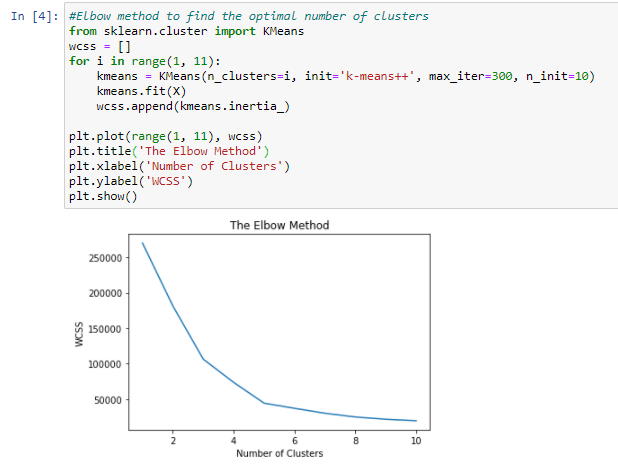
**Code Snippets and Explanation:**

****

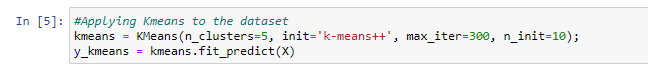
Import necessary libraries - sklearn, numpy, pandas, etc.

****

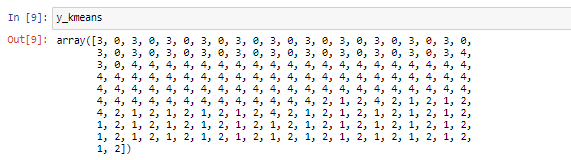
Using pandas, we first import the dataset into our workspace and are assigning the income attribute along with shopping score as independent variables.



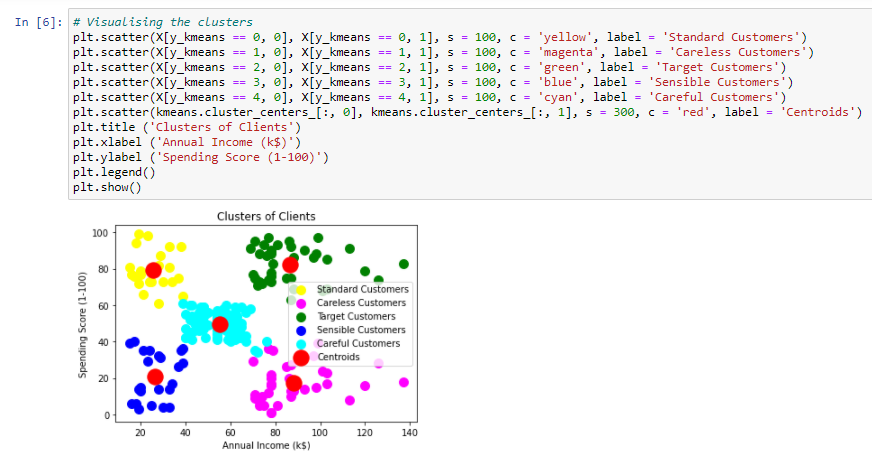
Here we are plotting a graph that marks Within Cluster Sum of Squares (WCSS) with the increase in number of clusters. We can see an elbow formation when the number of clusters is 5 and hence, we assume that optimal number of clusters in our dataset is 5



Here we are training our k-means model with 5 clusters. We are also generating the y\_kmeans array that stores the cluster index of each input attribute from 0 to 4



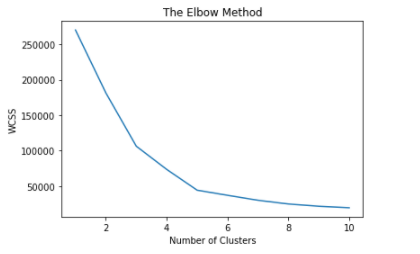
Here we are printing our y\_kmeans array and we can see that each input cell is assigned a value between 0 and 4, both inclusive. This corresponds to the cluster index of each input.



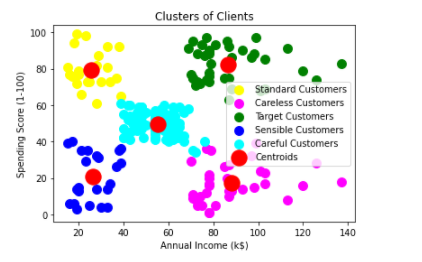
Here we have visualized our results. We have labelled different clusters as blue, green, pink, yellow and cyan. Each cluster correspond to different category of target audience. We have also marked centroids of each cluster which are red in colour.

**Result and Conclusion:**

Elbow Method Graph



Clustering Graph:



Here different clusters are marked as blue, yellow, magenta, cyan and green. The red dot over each cluster represents its centroid.

We can categorise these clusters as: -

* Yellow Cluster corresponds to careless customers as they have low income but high spending.
* Blue Cluster as Sensible customers, becoz they have low income and low spending.
* Cyan Clusters are standard cluster that suggest they have median income and median spending.
* The pink coloured cluster correspond to Target Customers, as they have high income but low spending, the shopping company can give them offers and attractions as they are capable of spending more but they aren’t doing it currently.
* Finally, the Green coloured clusters are Careful customers. They have high income and thus high spending as well.