MUSIC GENRE CLASSIFICATION

Team members

AAYUSH SAGAR CB.EN.U4ELC20043
HARI VARSHA CB.EN.U4ELC20021
NAVEEN US CB.EN.U4ELC20043

Dataset Reference:

1. http://marsyas.info/downloads/datasets.html

Other References:

- https://machinelearningmastery.com/confusion-matrix-machine-learning/ - for confusion matrix reference.
- https://scriptverse.academy/tutorials/python-matplotlib-plot-straight-line.html#:~:text=Matplotlib%3A%20Graph%2FPlot%20a%20Straight%20Line&text=The%20equation%20y%3Dmx,gradient%20and%20c%20its%20intercept – for working with Matplotlib in python.
- https://neptune.ai/blog/knn-algorithm-explanation-opportunities-limitations - for KNN classification limitations.

Project Colab Link:

https://colab.research.google.com/drive/1QMXErvFWOWegqa-I4a77f08z_kGr4Ky5?usp=sharing

Problem Formulation (3 marks)

Objective: *Music genre classification*.

Dataset details:

- No. of rows: 9991
- No. of Columns: 60
- No. of Class: 10
- The GTZAN dataset is the most-used public dataset for evaluation in machine listening research for music genre recognition (MGR). The files were collected in 2000-2001 from a variety of sources including personal CDs, radio, microphone recordings, in order to represent a variety of recording conditions.
- The Original GTZAN dataset consists of 1000 audio tracks each 30 seconds long. It contains 10 genres, each represented by 100 tracks.
- CSV file has for each song (30 seconds long divided into 10 - 3 sec audios to increase the data size) a mean and variance computed over multiple features that can be extracted from the audio files from the original dataset.

Assumptions

- The Audio samples are assumed to have minimum to no noise.
- All Data derived from the Audio sample spectral images are accurate enough for the problem.
- The audio sample each are from one particular genre with no overlapping.

Feature Description (2 marks)

- length : length of the audio sample
- chroma_stft_mean : Mean of the chroma short term fourier transform.
- chroma_stft_var : Variance of the chroma short term fourier transform.
- rms_mean : Mean of root mean square.
- rms_var : Variance of root mean square.
- spectral_centroid_mean : Mean of spectral centroid.
- spectral_centroid_var : Variance of spectral centroid.
- spectral_bandwidth_mean : Mean of spectral bandwidth.
- spectral_bandwidth_var : Variance of spectral bandwidth.
- rolloff_mean : Mean of spectral rolloff frequency.
- rolloff_var : Variance spectral rolloff frequency.
- zero_crossing_rate_mean : Mean of the zero crossing rate.
- zero_crossing_rate_var : Variance of the zero crossing rate.

Feature Description

- harmony_mean : Mean of the harmony.
- harmony_var : Variance of the harmony.
- Tempo : Tempo of the audio sample in bpm.
- mfccX_mean : Mean of the Mel frequency cepstral coefficients (20 sets).
- mfccX_var : Variance of the Mel frequency cepstral coefficients (20 sets).

KNN classifier (5 marks)

- K Nearest Neighbour algorithm for classification works on the principle of sorting new data based on it's distance from the data previously used to train the model and the number of nearest neighbours to be taken into consideration is decided by the K-value chosen.
- The dataset was divided into 70% for training and 30% for testing.
- The Minkowski distance was used as the distance metric
- Cross validation was performed using the K-fold approach.
- The CV value was taken as 5.
- The dataset therefore was divided into 5 groups where the corresponding fold value was used for testing while the rest were used for training.

Common to all methods

```
[ ]  import pandas as pd #data analysis toolkit
     import matplotlib.pyplot as plt #for plotting graphs
     import numpy as np #for high level computations
     %matplotlib inline
```

```
[ ]  from sklearn.preprocessing import StandardScaler #standardisation of values
     from sklearn.preprocessing import MinMaxScaler #normalization of values
     from sklearn.model_selection import train_test_split #to split data
     from sklearn.neighbors import KNeighborsClassifier #KNN classifier
     from sklearn.metrics import confusion_matrix,accuracy_score #to get confusion matrix and accuracy
     from sklearn.model_selection import cross_val_score #to perform evaluation and cross_validation
```

```
[ ]  data_set = pd.read_csv("features_3_sec.csv") #data_set input
```

```
[ ]  data_set = data_set.drop('filename', axis = 1) #dropping of columns as mentioned
     data_set = data_set.fillna(data_set.mean()) #mean for missing data
```

```
[ ]  data_set = np.round(data_set, decimals = 2) #rounding all values in dataset to 2 decimal places
     data_set.head() #first 5 values in dataset
```

	length	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean	spectral_centroid_var	spectral_bandwidth_mean	spectral_ba
0	66149	0.34	0.09	0.13	0.0	1773.07	167541.63	1972.74	
1	66149	0.34	0.09	0.11	0.0	1816.69	90525.69	2010.05	
2	66149	0.35	0.09	0.13	0.0	1788.54	111407.44	2084.57	
3	66149	0.36	0.09	0.13	0.0	1655.29	111952.28	1960.04	
4	66149	0.34	0.09	0.14	0.0	1630.66	79667.27	1948.50	

5 rows × 59 columns

```
[ ] dset_modified = data_set.drop('label',axis = 1) #dataset without label feature
```

```
[ ] dset_modified.head() #first 5 values in dataset
```

	length	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean	spectral_centroid_var	spectral_bandwidth_mean	spectr
0	66149	0.34	0.09	0.13	0.0	1773.07	167541.63	1972.74	
1	66149	0.34	0.09	0.11	0.0	1816.69	90525.69	2010.05	
2	66149	0.35	0.09	0.13	0.0	1788.54	111407.44	2084.57	
3	66149	0.36	0.09	0.13	0.0	1655.29	111952.28	1960.04	
4	66149	0.34	0.09	0.14	0.0	1630.66	79667.27	1948.50	

5 rows × 58 columns

```
[ ] data_set_feat = pd.DataFrame(dset_modified,columns = data_set.columns[:-1]) #dataset without label feature
```

```
[ ] data_set_feat.head() #first 5 values in dataset
```

	length	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean	spectral_centroid_var	spectral_bandwidth_mean	spectral_b
0	66149	0.34	0.09	0.13	0.0	1773.07	167541.63	1972.74	
1	66149	0.34	0.09	0.11	0.0	1816.69	90525.69	2010.05	
2	66149	0.35	0.09	0.13	0.0	1788.54	111407.44	2084.57	
3	66149	0.36	0.09	0.13	0.0	1655.29	111952.28	1960.04	
4	66149	0.34	0.09	0.14	0.0	1630.66	79667.27	1948.50	

5 rows × 58 columns

KNN Classifier

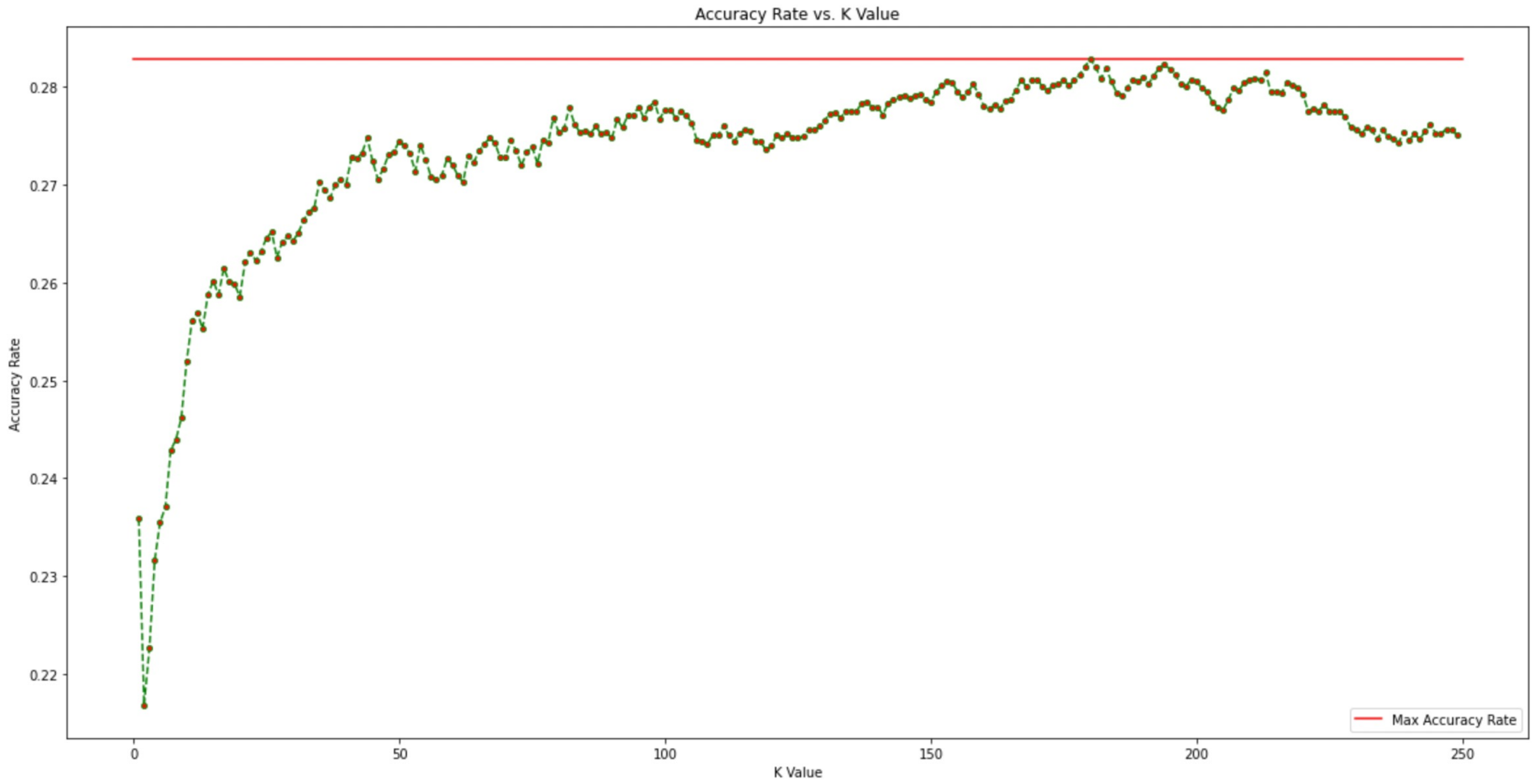
```
data_set_feat = np.round(data_set_feat, decimals=2) #rounding all values to 2 decimal places
```

```
one_train,one_test,two_train,two_test = train_test_split(data_set_feat,data_set['label'], test_size = 0.30) #test_train split
#test size 30% and train size 70%
```

```
 #computation of accuracy rates for various neighbour values
Accuracy_rates = []
a=[]

for i in range(1,250):
  k_nearest_neighbour = KNeighborsClassifier(n_neighbors = i)
  final_score = cross_val_score(k_nearest_neighbour, data_set_feat,data_set['label'], cv = 5)
  Accuracy_rates.append(final_score.mean())
```

```
#plot
plt.figure(figsize = (20,10))

x = np.linspace(0,250,100)
y = [max(Accuracy_rates) for i in range(100)]
plt.plot(x, y, '-r', label='Max Accuracy Rate')
plt.legend(loc='lower right')

plt.plot(range(1,250),Accuracy_rates,color='green', linestyle='dashed', marker='o',
         markerfacecolor='red', markersize=4)
plt.title('Accuracy Rate vs. K Value')
plt.xlabel('K Value')
plt.ylabel('Accuracy Rate')
```

Accuracy rate vs K value



Confusion Matrix

```
max_index = Accuracy_rates.index(max(Accuracy_rates)) #Best case identifier

k_nearest_neighbour = KNeighborsClassifier(n_neighbors=max_index)

k_nearest_neighbour.fit(one_train,two_train)
prediction = k_nearest_neighbour.predict(one_test)

print('For K=', max_index)
print('Confusion matrix: ')
print('\n')
print(confusion_matrix(two_test,prediction)) #confusion matrix
print('\n')
print('Accuracy rate: ',round(accuracy_score(two_test,prediction),2)*100,'%')
#accuracy rate
```

```
For K= 179
Confusion matrix:


[[  5  25  57  17  11  55 100   3  46  10]
 [  0 225   6   1   1  10  51   0   2   1]
 [  6  13  82  20   6  26  46  11  48  12]
 [  1   2  55  68  27  12  50  15  56  17]
 [  2   0  27  60  24   3  37  55  89  12]
 [ 11  53  57   7   5  72  67   3  31  10]
 [  3  49  13  20  15  12 159   0   3  22]
 [  0   2  25  37  24   8   5  88 105  11]
 [  1   3  41  25   8   2  12  35 146   5]
 [  6  24  51  41  13  15  58  14  53  19]]


Accuracy rate:  30.0 %
```

Before normalization of the dataset,
Confusion matrix for the K value = 179
Which corresponds to the maximum accuracy.

Confusion Matrix for random k values

```
For K= 100
Confusion matrix:

[[ 11  24  52  19  18  60  89   4  42  10]
 [  4 219   6   0   1  13  51   0   2   1]
 [  8  12  81  22   5  28  43  11  47  13]
 [  6   2  49  65  32   8  47  16  56  22]
 [  1   0  26  58  34   3  31  56  86  14]
 [ 16  50  57   7   6  78  62   3  24  13]
 [  8  42  21  20  15  12 159   1   3  15]
 [  2   2  20  41  23  12   2  99  94  10]
 [  1   2  37  24   6   2  15  46 138   7]
 [  6  23  53  39  16  15  58  13  51  20]]

Accuracy rate:  20.0 %
```

```
For K= 210
Confusion matrix:

[[  5  27  57  20  11  55  99   3  45   7]
 [  0 225   8   1   0   7  53   0   3   0]
 [  8  14  75  23   3  27  50  12  47  11]
 [  1   2  53  68  23  13  58  15  55  15]
 [  0   0  27  67  19   4  39  53  89  11]
 [ 12  55  56   7   5  71  68   3  30   9]
 [  2  51  17  19  11  12 162   0   3  19]
 [  0   2  27  41  24   6   6  87 103   9]
 [  1   1  39  28   7   2  17  34 145   4]
 [  8  25  52  47   9  14  61  11  54  13]]

Accuracy rate:  28.999999999999996 %
```

Inference:

- The confusion matrix and the accuracy rate for 2 random K values are displayed.
- The best K value can be identified from the graph.
- Minkowski Distance is generalization of the Euclidean and Manhattan distance measures
- A confusion matrix is a tabular summary of the number of correct and incorrect predictions made by a classifier..
- It can be used to evaluate the performance of a classification model through the calculation of performance metrics like accuracy.

Normalization

```
scaled = MinMaxScaler() #function MinMax scaler for normalising values
```

```
scaled.fit(data_set.drop('label',axis=1)) # dropping class feature
```

```
MinMaxScaler()
```

```
dset_modified = scaled.transform(data_set.drop('label',axis=1))
```

```
data_set_feat=pd.DataFrame(dset_modified,columns=data_set.columns[:-1])
```

```
data_set_feat = np.round(data_set_feat, decimals=2)
data_set_feat.head()
```

	length	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean	spectral_centroid_var	spectral_bandwidth_mean	spectral_bandwidth_var
0	0.0	0.36	0.7	0.30	0.0	0.26	0.03	0.46	0.09
1	0.0	0.36	0.7	0.25	0.0	0.27	0.02	0.47	0.05
2	0.0	0.38	0.7	0.30	0.0	0.27	0.02	0.49	0.06
3	0.0	0.39	0.7	0.30	0.0	0.24	0.02	0.46	0.07
4	0.0	0.36	0.7	0.32	0.0	0.23	0.02	0.45	0.05

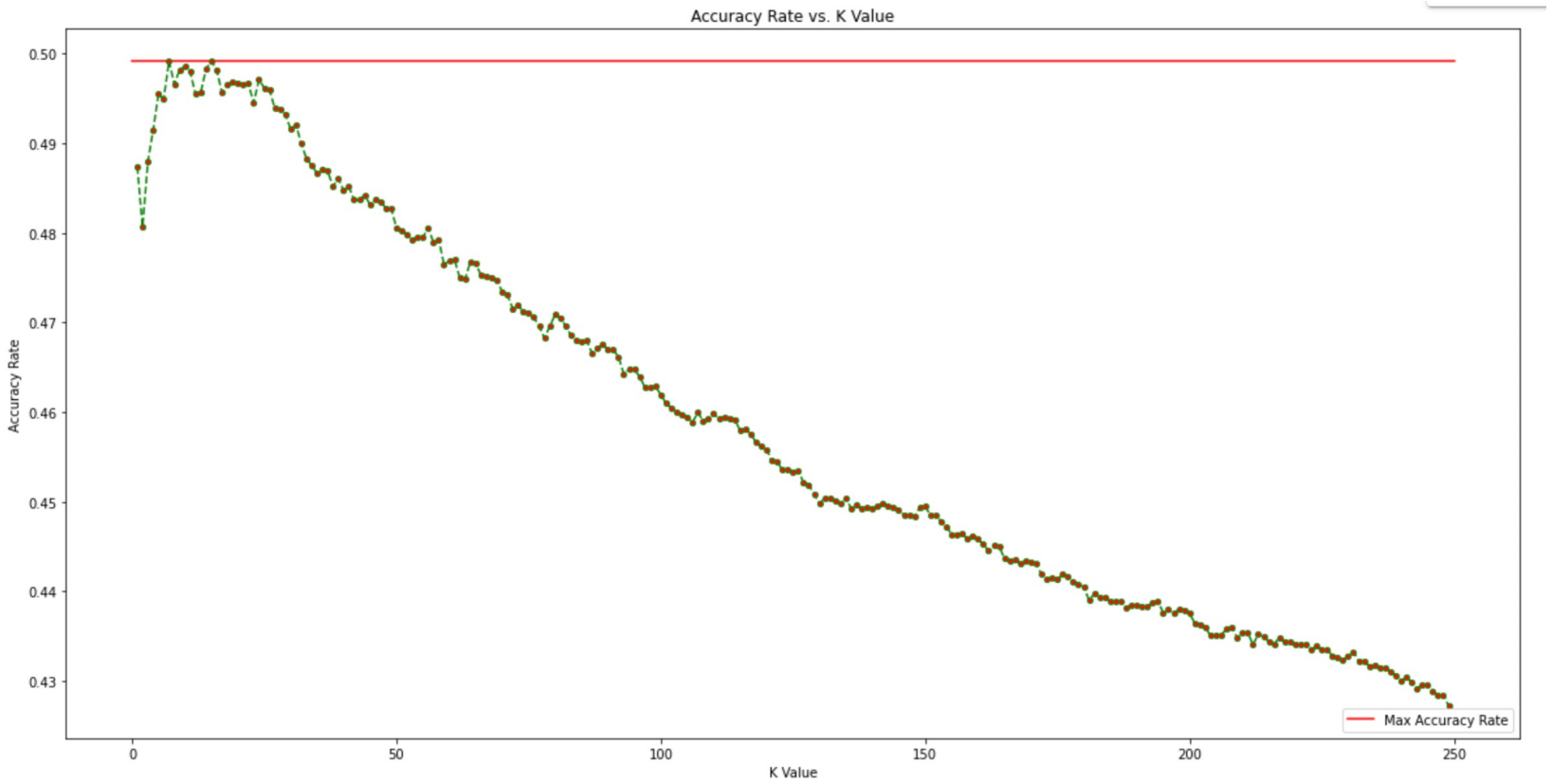
5 rows × 58 columns

Dataset after Normalization

```
one_train,one_test,two_train,two_test = train_test_split(data_set_feat,data_set['label'], test_size = 0.30)
```

```
Accuracy_rates = []
a=[]

for i in range(1,250):
  k_nearest_neighbour = KNeighborsClassifier(n_neighbors = i)
  final_score = cross_val_score(k_nearest_neighbour, data_set_feat,data_set['label'], cv = 5)
  Accuracy_rates.append(final_score.mean())
```

```
plt.figure(figsize = (20,10))

x = np.linspace(0,250,100)
y = [max(Accuracy_rates) for i in range(100)]
plt.plot(x, y, '-r', label='Max Accuracy Rate')
plt.legend(loc='lower right')

plt.plot(range(1,250),Accuracy_rates,color='green', linestyle='dashed', marker='o',
         markerfacecolor='red', markersize=4)
plt.title('Accuracy Rate vs. K Value')
plt.xlabel('K Value')
plt.ylabel('Accuracy Rate')
```


Accuracy Rate vs. K Value
Accuracy Rate
0.50
0.49
0.48
0.47
0.46
0.45
0.44
0.43
0
50
100
150
200
250
K Value
Max Accuracy Rate

Accuracy Rate and confusion matrix after Normalization

```
For K= 14
Confusion matrix:


[[272   0   6   0   0   1   2   0   7   3]
 [  1 275   4   0   0  10   0   0   0   3]
 [  9   3 271   3   1   7   2   2   9  11]
 [  2   1   1 271   1   1   2   2   3   8]
 [  2   1   4   1 257   0   0   4   3   5]
 [  2  14   9   2   1 272   1   0   1   2]
 [  4   0   1   2   0   0 300   0   1   8]
 [  0   0   6   9   1   1   0 275   6   9]
 [  1   0   7   0  10   1   0   4 276   3]
 [  4   2   4  14   4   6   2   4   2 255]]


Accuracy rate:  91.0 %
```

After normalization we get maximum accuracy at k=14

Normalization

- Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

- It is observed that the accuracy has increased comparing to the accuracy achieved using KNN classifier.
- After normalization the accuracy achieved is 91.0% for k=14.

Inference (3 marks)

- In this experiment, K-Nearest Neighbor method of classification and normalization were implemented on the GTZAN dataset for the music genre classification model.
- For this experiment, the K-values were varied from 1 to 250 and an accuracy rate of 30% was obtained. It was observed that for lower values of K, higher values of accuracy rates were obtained.
- After normalization, all values in the dataset were changed to lie within the range of 0 to 1.
- A confusion matrix between the predicted and actual values was also computed.

Miscellaneous (5 marks)

- The basics of google colab cloud based ide was learnt, along with advantages of it such as hassle free collaboration, etc.
- The functionality offered by the various data analysis/ machine learning libraries used in the experiment like sklearn, pandas, numpy, matplotlib was learnt.
- The basics of finding the appropriate dataset and cleaning of dataset was understood.