Machine Learning Lab Assignment

Name: Om Ashish Mishra

Registration Number: 16BCE0789

Slot: F2

Knn

1. Implement k-Nearest Neighbor algorithm for classifying a dataset.

Dataset Used: Iris Dataset
The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimetres. Based on the combination of these four features, Fisher developed a linear discriminant model to distinguish the species from each other.

The Code:

```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.metrics import classification_report,confusion_matrix

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

names = ['sepal-length', 'sepal-width', 'petallength','petal-width', 'Class']

dataset = pd.read_csv(url, names=names)

dataset.head()

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, 4].values

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.20)

from sklearn.preprocessing import StandardScaler

scaler =StandardScaler()

scaler.fit(X_train)

X_train = scaler.transform(X_train)

X_test = scaler.transform(X_test)

from sklearn.neighbors import KNeighborsClassifier

for i in range(1,6):
```

```
    print('for k = ',i)
    classifier = KNeighborsClassifier(n_neighbors=i)
    classifier.fit(X_train, y_train)
    y_pred = classifier.predict(X_test)
    print(confusion_matrix(y_test,y_pred))
    print(classification_report(y_test, y_pred))
```

The Output:

```
py3 )
for k =  1
[[11  0  0]
 [ 0  7  0]
 [ 0  0 12]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      1.00      1.00         7
 Iris-virginica       1.00      1.00      1.00        12

    avg / total       1.00      1.00      1.00        30

for k =  2
[[11  0  0]
 [ 0  7  0]
 [ 0  0 12]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      1.00      1.00         7
 Iris-virginica       1.00      1.00      1.00        12

    avg / total       1.00      1.00      1.00        30

for k =  3
[[11  0  0]
 [ 0  7  0]
 [ 0  0 12]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      1.00      1.00         7
 Iris-virginica       1.00      1.00      1.00        12

    avg / total       1.00      1.00      1.00        30


for k =  4
[[11  0  0]
 [ 0  7  0]
 [ 0  1 11]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       0.88      1.00      0.93         7
 Iris-virginica       1.00      0.92      0.96        12

    avg / total       0.97      0.97      0.97        30

for k =  5
[[11  0  0]
 [ 0  7  0]
 [ 0  1 11]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       0.88      1.00      0.93         7
 Iris-virginica       1.00      0.92      0.96        12

    avg / total       0.97      0.97      0.97        30
```

Implement Multi-Layer Perceptron using a dataset

K-means:

The Dataset:

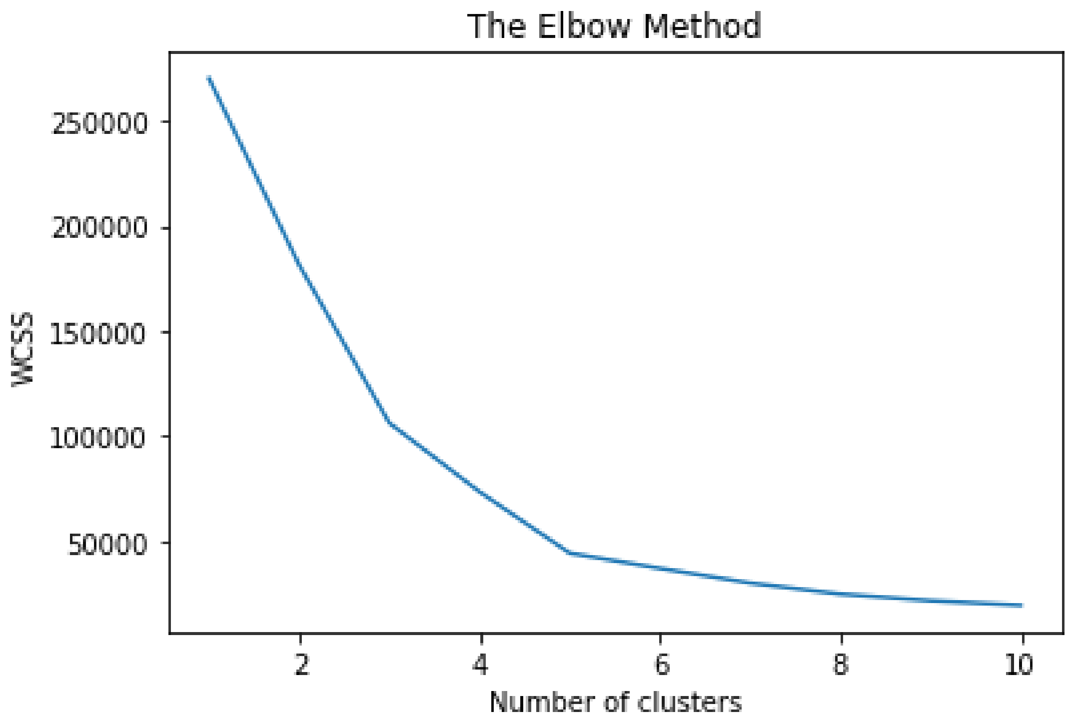
CustomerI	Genre	Age	Annual Inc	Spending Score (1-100)
1	Male	19	15	39
2	Male	21	15	81
3	Female	20	16	6
4	Female	23	16	77
5	Female	31	17	40
6	Female	22	17	76
7	Female	35	18	6
8	Female	23	18	94
9	Male	64	19	3
10	Female	30	19	72
11	Male	67	19	14
12	Female	35	19	99
13	Female	58	20	15
14	Female	24	20	77
15	Male	37	20	13
16	Male	22	20	79
17	Female	35	21	35
18	Male	20	21	66
19	Male	52	23	29
20	Female	35	23	98
21	Male	35	24	35
22	Male	25	24	73
23	Female	46	25	5
24	Male	31	25	73
25	Female	54	28	14
26	Male	29	28	82
27	Female	45	28	32

The Code:

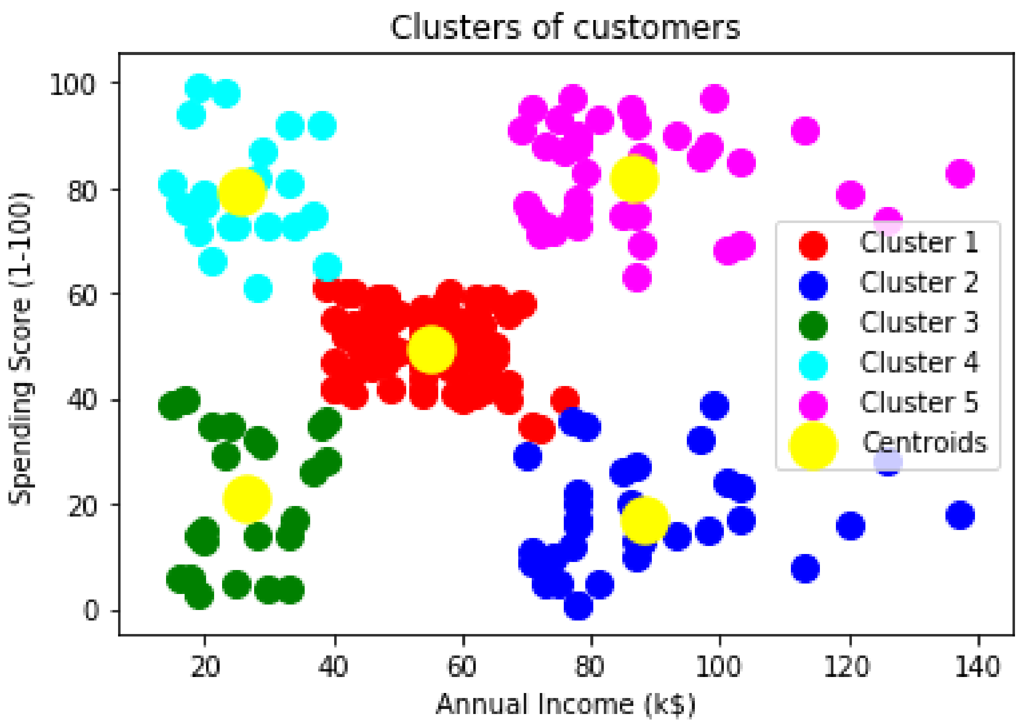
```
# K-Means Clustering

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd


# Importing the dataset

dataset = pd.read_csv('Mall_Customers.csv')

X = dataset.iloc[:, [3, 4]].values

# y = dataset.iloc[:, 3].values


# Using the elbow method to find the optimal number of clusters

from sklearn.cluster import KMeans
```

```
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()

# Fitting K-Means to the dataset
kmeans = KMeans(n_clusters = 5, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(X)

# Visualising the clusters
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Cluster 1')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Cluster 3')
plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')
plt.scatter(X[y_kmeans == 4, 0], X[y_kmeans == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label = 'Centroids')
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```

The Output:



Therefore Optimum Number of Clusters is 5.



SVM:

The Dataset:

User ID	Gender	Age	Estimated!	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	0
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	0
15728773	Male	27	58000	0
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	0
15727311	Female	35	65000	0
15570769	Female	26	80000	0
15606274	Female	26	52000	0
15746139	Male	20	86000	0
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1
15617482	Male	45	26000	1
15704583	Male	46	28000	1
15621083	Female	48	29000	1
15649487	Male	45	22000	1
15736760	Female	47	49000	1
15714658	Male	48	41000	1
15599081	Female	45	22000	1
15705113	Male	46	23000	1
15631159	Male	47	20000	1

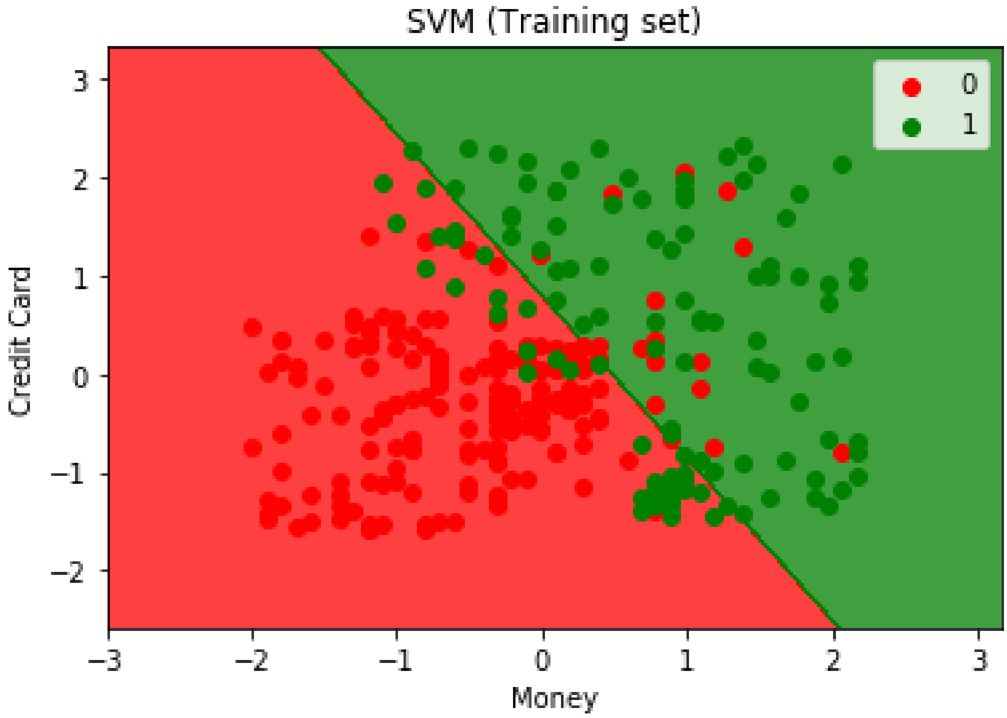
The Code:

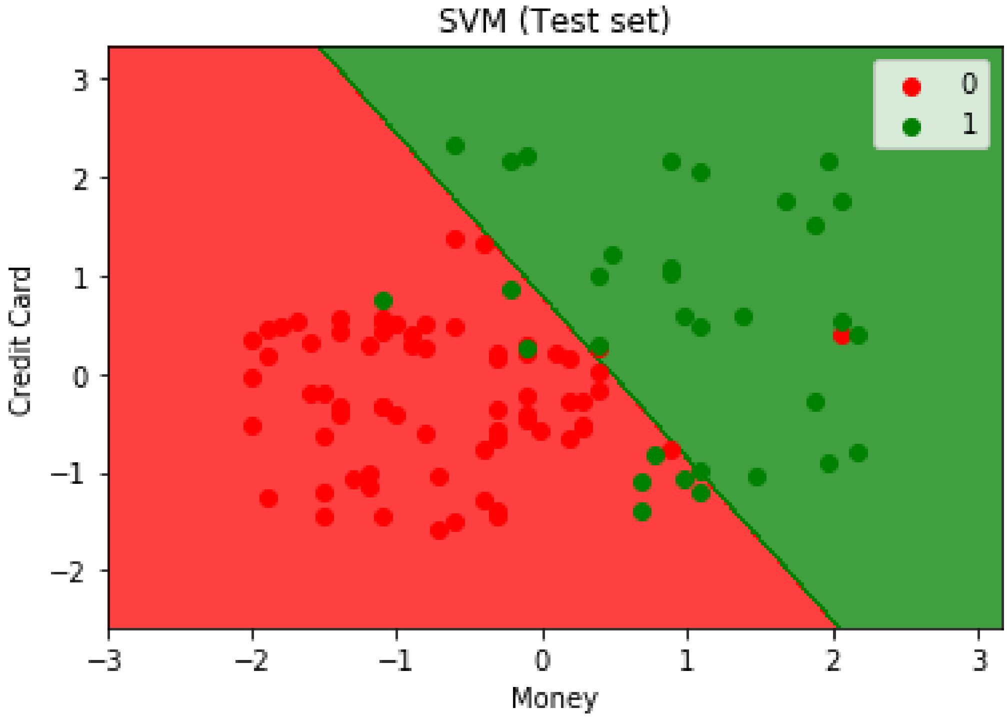
```
# Support Vector Machine (SVM)


# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd


# Importing the dataset

dataset = pd.read_csv('Social_Network_Ads.csv')

X = dataset.iloc[:, [2, 3]].values

y = dataset.iloc[:, 4].values


# Splitting the dataset into the Training set and Test set

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Fitting SVM to the Training set
from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('SVM (Training set)')
```

```
plt.xlabel('Money')
plt.ylabel('Credit Card')
plt.legend()
plt.show()


# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('SVM (Test set)')
plt.xlabel('Money')
plt.ylabel('Credit Card')
plt.legend()
plt.show()
```

The Output:






```
In [2]: cm
Out[2]:
array([[66,  2],
       [ 8, 24]], dtype=int64)
```

Hierarchical clustering:

The Dataset:

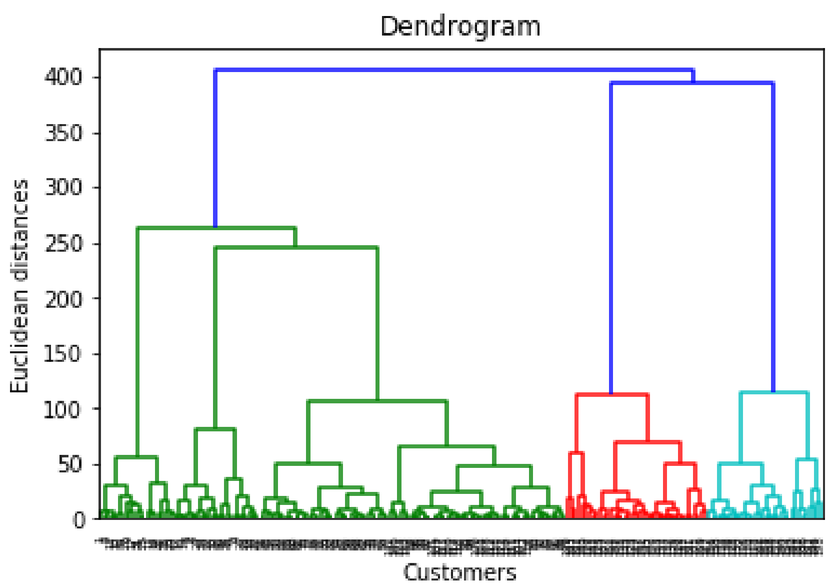
CustomerI	Genre	Age	Annual Inc	Spending Score (1-100)
1	Male	19	15	39
2	Male	21	15	81
3	Female	20	16	6
4	Female	23	16	77
5	Female	31	17	40
6	Female	22	17	76
7	Female	35	18	6
8	Female	23	18	94
9	Male	64	19	3
10	Female	30	19	72
11	Male	67	19	14
12	Female	35	19	99
13	Female	58	20	15
14	Female	24	20	77
15	Male	37	20	13
16	Male	22	20	79
17	Female	35	21	35
18	Male	20	21	66
19	Male	52	23	29
20	Female	35	23	98
21	Male	35	24	35
22	Male	25	24	73
23	Female	46	25	5
24	Male	31	25	73
25	Female	54	28	14
26	Male	29	28	82
27	Female	45	28	32

The Code:

```
# Hierarchical Clustering


# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd


# Importing the dataset

dataset = pd.read_csv('Mall_Customers.csv')

X = dataset.iloc[:, [3, 4]].values

# y = dataset.iloc[:, 3].values


# Using the dendrogram to find the optimal number of clusters

import scipy.cluster.hierarchy as sch

dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward'))

plt.title('Dendrogram')
```

```
plt.xlabel('Customers')
plt.ylabel('Euclidean distances')
plt.show()

# Fitting Hierarchical Clustering to the dataset
from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'ward')
y_hc = hc.fit_predict(X)

# Visualising the clusters
plt.scatter(X[y_hc == 0, 0], X[y_hc == 0, 1], s = 100, c = 'red', label = 'Cluster 1')
plt.scatter(X[y_hc == 1, 0], X[y_hc == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')
plt.scatter(X[y_hc == 2, 0], X[y_hc == 2, 1], s = 100, c = 'green', label = 'Cluster 3')
plt.scatter(X[y_hc == 3, 0], X[y_hc == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')
plt.scatter(X[y_hc == 4, 0], X[y_hc == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```

The Output:



MLP

import numpy as np # linear algebra import pandas as pd # data processing import matplotlib.pyplot as plt import seaborn as sns df = pd.read_csv('Dataset_spine.csv') df = df.drop(['Unnamed: 13'], axis=1) df.head() df.describe()

df = df.drop(['Col7','Col8','Col9','Col10','Col11','Col12'], axis=1) df.head() from sklearn.neural_network import MLPClassifier from sklearn.model_selection import train_test_split from sklearn.metrics import accuracy_score from sklearn.metrics import confusion_matrix

y = df['Class_att'] x = df.drop(['Class_att'], axis=1)

x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=

0.25, random_state=27) clf = MLPClassifier(hidden_layer_sizes=(100,100,100), max_iter=500, alpha=0.0001,solver='sgd',

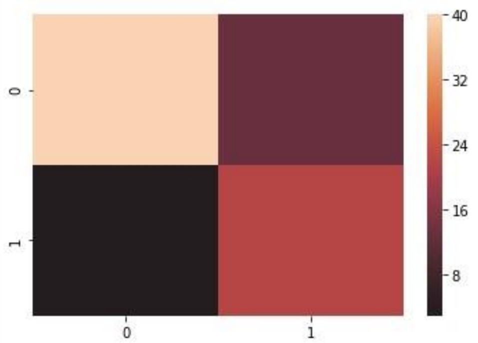
verbose=10,random_state=21,tol=0.000000001) clf.fit(x_train, y_train) y_pred = clf.predict(x_test) print("accuracy : ") accuracy_score(y_test, y_pred) cm = confusion_matrix(y_test, y_pred) print("confusion matrix : ") print(cm)

sns.heatmap(cm, center=True) print("heatmap :")

plt.show()

```
In [10]: runfile('/Users/shivanipriya/Documents/CODES/mlp.py', wdir='/Users/shivanipriya/Documer
CODES')
Iteration 1, loss = 4.26874505
Iteration 2, loss = 7.21713939
Iteration 3, loss = 1.55562179
Iteration 4, loss = 0.94416556
Iteration 5, loss = 1.50851327
Iteration 6, loss = 0.50784081
Iteration 7, loss = 0.39918410
Iteration 8, loss = 0.37867398
Iteration 9, loss = 0.37472479
Iteration 10, loss = 0.34567928
Iteration 11, loss = 0.34293793
Iteration 12, loss = 0.33180054
Iteration 13, loss = 0.33138624
Iteration 14, loss = 0.32280653
Iteration 15, loss = 0.32238711
Iteration 16, loss = 0.33164385
Iteration 17, loss = 0.31229600
Iteration 18, loss = 0.30823387
Iteration 19, loss = 0.32317560
Iteration 20, loss = 0.30697614
Iteration 21, loss = 0.30901736
Iteration 22, loss = 0.35327698
Iteration 23, loss = 0.31835192
Iteration 24, loss = 0.30599160
Iteration 25, loss = 0.30315921
Iteration 26, loss = 0.30774646
Iteration 27, loss = 0.30939928
Iteration 28, loss = 0.30205629
Iteration 29, loss = 0.30492366
Iteration 30, loss = 0.30627954
Iteration 31, loss = 0.32245209
Iteration 32, loss = 0.30092367
Iteration 33, loss = 0.32289947
Iteration 34, loss = 0.30300054
```

```
Iteration 102, loss = 0.28888423
Iteration 103, loss = 0.40146253
Iteration 104, loss = 0.28481547
Iteration 105, loss = 0.28301174
Iteration 106, loss = 0.28824478
Iteration 107, loss = 0.28220960
Iteration 108, loss = 0.29126322
Iteration 109, loss = 0.28446019
Training loss did not improve more than tol=0.000000 for 10 consecutive epochs. Stopping.
accuracy :
confusion matrix :
[[40 13]
 [ 3 22]]
heatmap :
```




```
In [11]:
```

KNN

CODE import numpy as np import matplotlib.pyplot as plt import pandas as pd from sklearn.model_selection import train_test_split from sklearn.preprocessing import StandardScaler

dataset=pd.read_csv('iris.csv') dataset.head()

X=dataset.iloc[:,:-1].values y=dataset.iloc[:,4].values

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20

)

scalar=StandardScaler() scalar.fit(X_train)

X_train=scalar.transform(X_train) X_test=scalar.transform(X_test) from sklearn.neighbors import KNeighbAorsClassifier classifier=KNeighborsClassifier(n_neighbors=5)

classifier.fit(X_train,y_train) y_pred=classifier.predict(X_test) from sklearn.metrics import classification_report,confusion_matrix print (confusion_matrix(y_test,y_pred)) print (classification_report(y_test,y_pred)) error=[] for i in range(1,40):

knn=KNeighborsClassifier(n_neighbors=i)

knn.fit(X_train,y_train) pred_i=knn.predict(X_test) error.append(np.mean(pred_i!=y_test))

plt.figure(figsize=(12,6))

plt.plot(range(1,40),error,color='red',linestyle='dashed',marker='

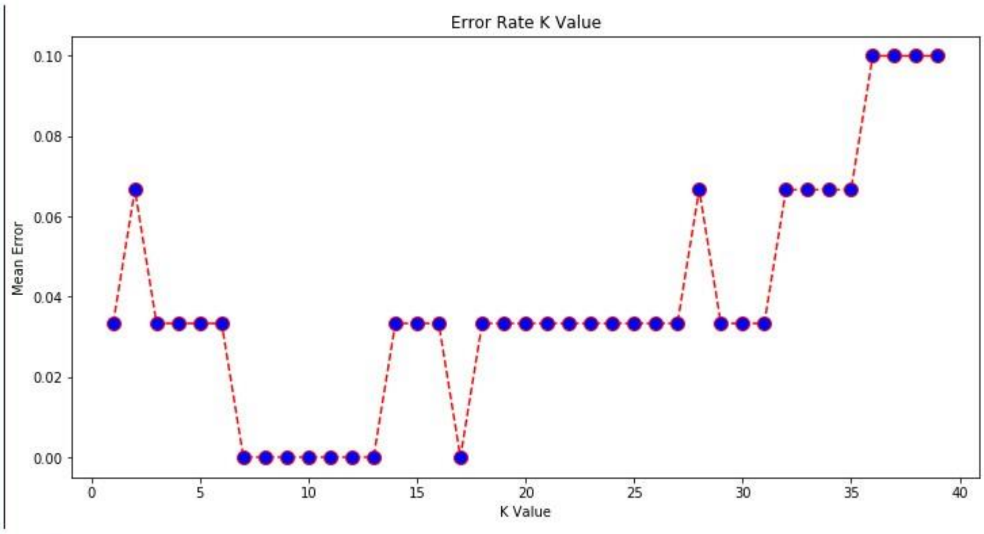
o',markerfacecolor='blue',markersize=10)

plt.title('Error Rate K Value') plt.xlabel('K Value')

plt.ylabel('Mean Error')

OUTPUT

```
In [7]: runfile('C:/Users/16BCE0828/.spyder-py3/temp.py', wdir='C:/Users/16BCE0828/.spyder-py3')
[[ 8  0  0]
 [ 0 11  0]
 [ 0  1 10]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         8
Iris-versicolor       0.92      1.00      0.96        11
 Iris-virginica       1.00      0.91      0.95        11

    avg / total       0.97      0.97      0.97        30
```




```
In [8]:
```

CSE4020 Machine Learning

Lab - 3

Name: S. Mohan Sai

Reg.No: 16BCE0486

Slot. No: L3 + L4

Submitted to: Prof. Vijaysherly .V

1. Implement k-Nearest Neighbour algorithm for classifying a dataset.

Import the dependencies

import numpy as np   import

matplotlib.pyplot as plt

import pandas as pd

Extract the dataset from the UCI repository.

url = "https://archive.ics.uci.edu/ml/machine-learning-

databases/iris/iris.data" names = ['sepal-length', 'sepal-width', 'petal-length',

'petal-width', 'Class'] dataset = pd.read_csv(url, names=names)

Display the items of the first 5 rows ad split the features and labels

dataset.head()

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, 4].values

Split the train and test data in a ratio of 80-20% respectively.

from sklearn.model_selection import
train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)

Preprocess the Data Using standard scalar.

from sklearn.preprocessing import

StandardScaler scaler = StandardScaler()

scaler.fit(X_train)

X_train = scaler.transform(X_train)

X_test = scaler.transform(X_test)

Import KNN model from sklearn and fit the train dataset

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n_neighbors=5)

classifier.fit(X_train, y_train)

Now predict the dataset with the testing data and find the metrics.

y_pred = classifier.predict(X_test)

from sklearn.metrics import classification_report,

confusion_matrix print(confusion_matrix(y_test, y_pred))

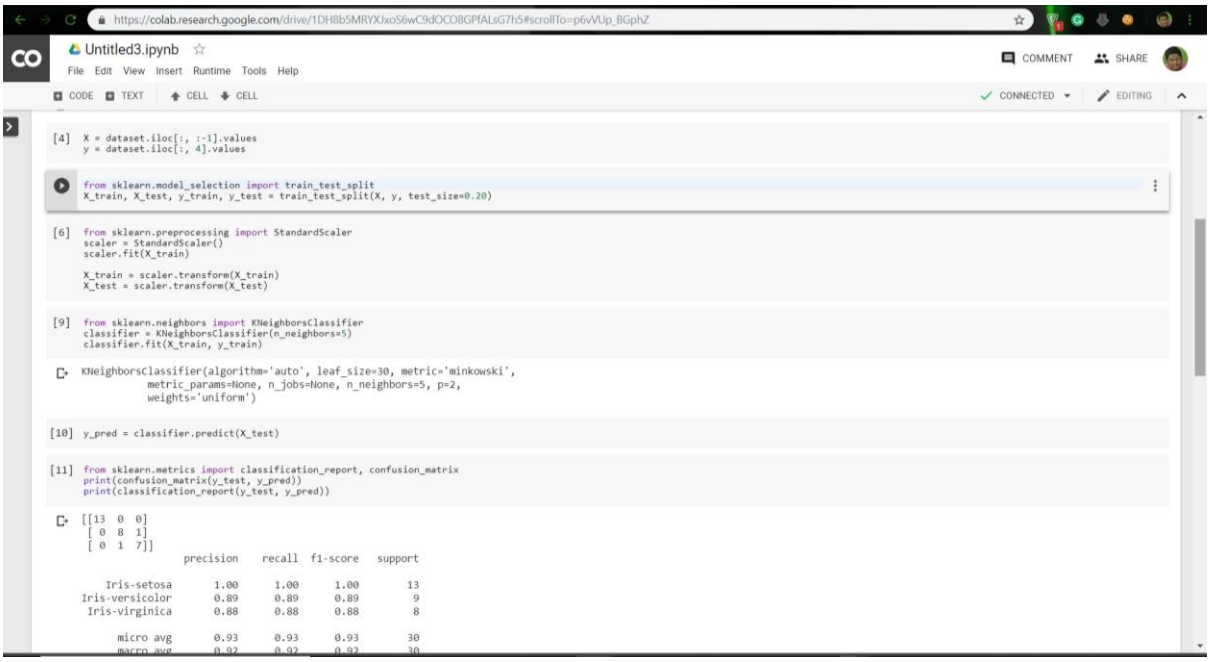
print(classification_report(y_test, y_pred)) error = []

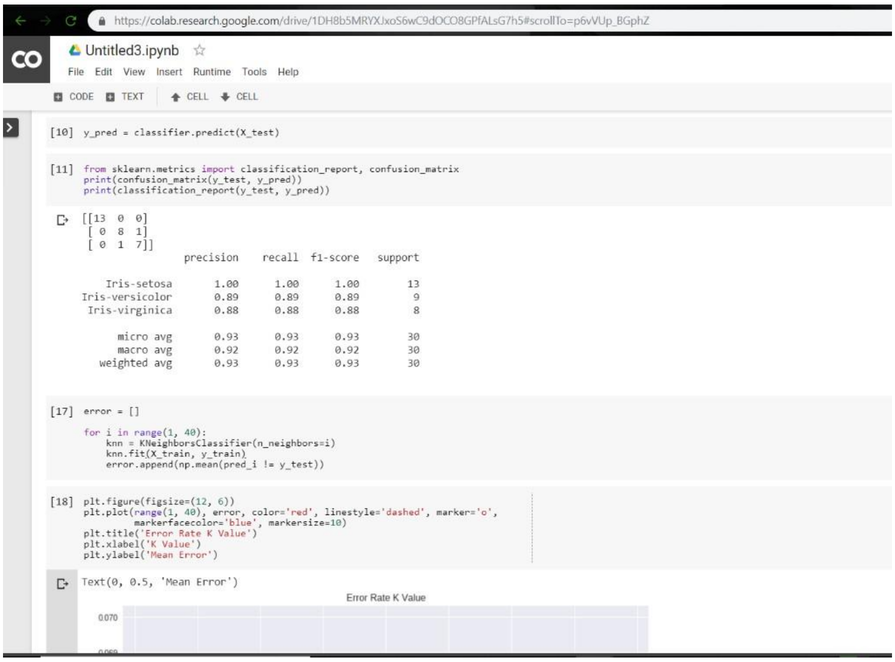
Now we are interested in finding the K value of 1 to 40 to which it has the highest accuracy.

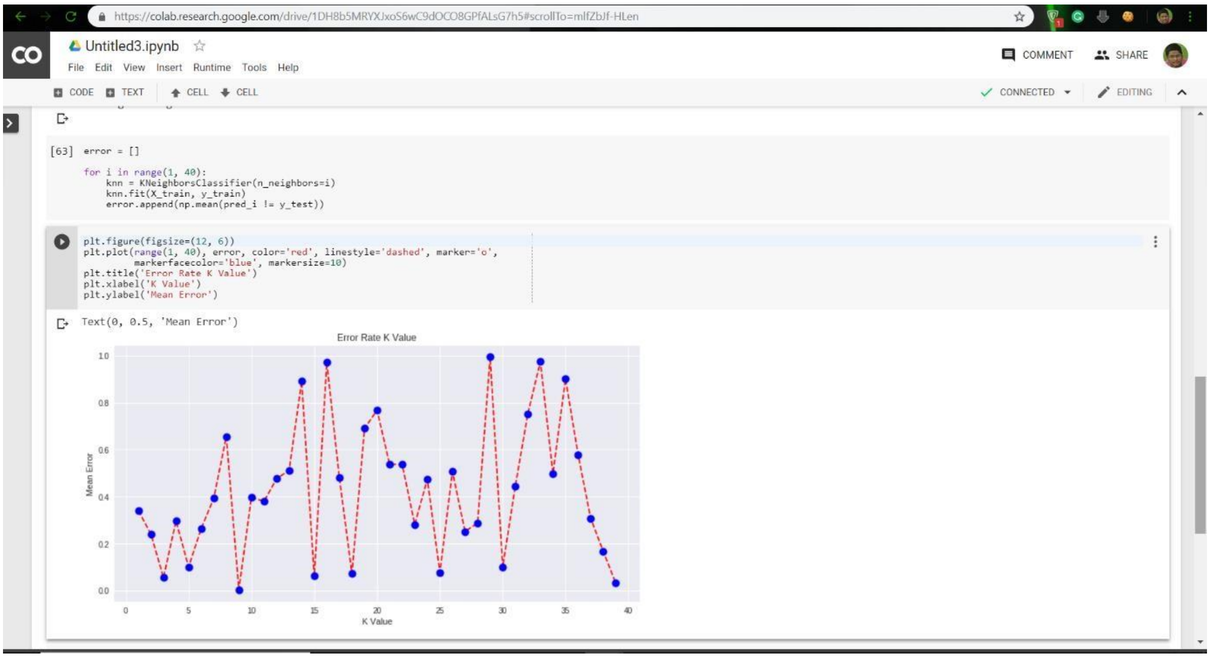
```
for i in range(1, 40):
    knn = KNeighborsClassifier(n_neighbors=i)
knn.fit(X_train, y_train)
    error.append(np.mean(pred_i != y_test))
plt.figure(figsize=(12, 6))
plt.plot(range(1, 40), error, color='red', linestyle='dashed',
marker='o',        markerfacecolor='blue', markersize=10)
plt.title('Error Rate K Value')   plt.xlabel('K Value')
plt.ylabel('Mean Error')
```

Output ScreenShots


https://colab.research.google.com/drive/1DH8b5MRYXJxoS6wC9dOCO8GPfALsG7h5#scrollTo=FGSPHq3gG6G2
Untitled3.ipynb
File Edit View Insert Runtime Tools Help
COMMENT SHARE
CODE TEXT CELL CELL
CONNECTED EDITING
[1] import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
[2] url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']
dataset = pd.read_csv(url, names=names)
[3] dataset.head()
sepal-length sepal-width petal-length petal-width Class
0 5.1 3.5 1.4 0.2 Iris-setosa
1 4.9 3.0 1.4 0.2 Iris-setosa
2 4.7 3.2 1.3 0.2 Iris-setosa
3 4.6 3.1 1.5 0.2 Iris-setosa
4 5.0 3.6 1.4 0.2 Iris-setosa
[4] X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 4].values
[5] from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
[6] from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)


https://colab.research.google.com/drive/1DH8b5MRYXJxoS6wC9dOCO8GPfALsG7h5#scrollTo=p6vVUp_BGphZ
Untitled3.ipynb
File Edit View Insert Runtime Tools Help
COMMENT SHARE
CODE TEXT CELL CELL
CONNECTED EDITING
[4] X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 4].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
[6] from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
[9] from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train, y_train)
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
metric_params=None, n_jobs=None, n_neighbors=5, p=2,
weights='uniform')
[10] y_pred = classifier.predict(X_test)
[11] from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
[[13 0 0]
[0 8 1]
[0 1 7]]
precision recall f1-score support
Iris-setosa 1.00 1.00 1.00 13
Iris-versicolor 0.89 0.89 0.89 9
Iris-virginica 0.88 0.88 0.88 8
micro avg 0.93 0.93 0.93 30





2. Implement Multi-Layer Perceptron using a dataset from UCI repository.

Import all the libraries

import numpy as np import

pandas as pd import

matplotlib.pyplot as plt

import seaborn as sns import

itertools import warnings

from sklearn.model_selection import

train_test_split from sklearn.preprocessing import

StandardScaler from sklearn.neural_network

import MLPClassifier from sklearn import metrics

from keras.models import Sequential import

tensorflow as tf

Read the file which is downloaded from the UCI repository

dataset = pd.read_csv("E:/projects/fraud detection/creditcard.csv") print("Few Entries: ") print(dataset.head())

print("Dataset Shape: ", dataset.shape)

print("Maximum Transaction Value: ", np.max(dataset.Amount)) print("Minimum Transaction Value: ", np.min(dataset.Amount)) **Analysis of the data which we have** color = {1:'blue',0:'yellow'} fraudlist = dataset[dataset.Class == 1] notfraudlist = dataset[dataset.Class == 0] print("The no of Fraud Samples are :", fraudlist.size)

print("The no of Non-Fraud Samples are :", notfraudlist.size)

fig,axes = plt.subplots(1,2)

axes[0].scatter(list(range(1,fraudlist.shape[0]+1)),fraudlist.Amount,color='blue')

axes[1].scatter(list(range(1,notfraudlist.shape[0]+1)),notfraudlist.Amount,color='yellow')

plt.show()

Now split the daset into features and labels with 30 featues and binary label

x = dataset.loc[:,dataset.columns.tolist()[1:30]] x = x.as_matrix() y = dataset.loc[:,'Class'] y = y.as_matrix()

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.33,random_state=0) print("Elements in the training set:" , np.bincount(y_train)) print("Elements in the testing set:" , np.bincount(y_test)) **Function to Train the Model** def trainmodel(model): model.fit(x_train,y_train)

Function To Make Predictions for The Neural Network

def predictmodeln(model):

```
y_pred = model.predict_classes(x_test)
f,t,thresholds = metrics.roc_curve(y_test,y_pred)
cm = metrics.confusion_matrix(y_test,y_pred)
print("Score:", metrics.auc(f,t))
print("Classification report:")

print(metrics.classification_report(y_test,y_pred))
print("Confusion Matrix:")     print(cm)
```

Defining the neural network with 3 layers deep and 1 hidden layer.

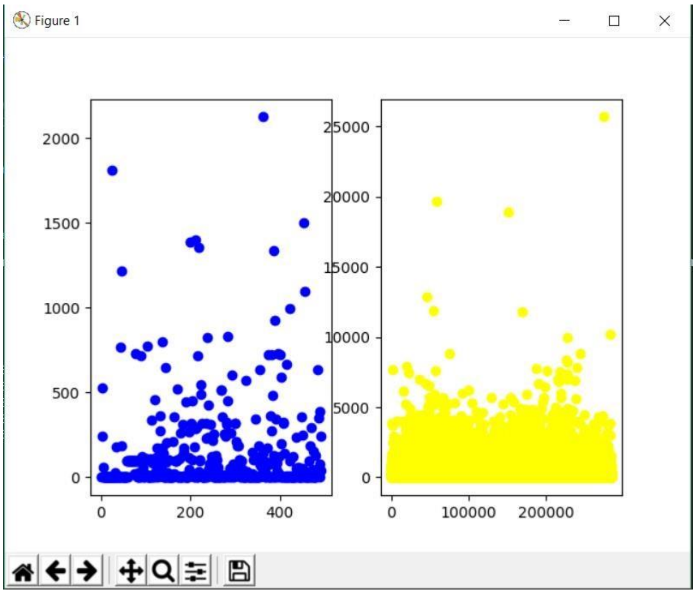
```
model = Sequential()
model.add(Dense(256,activation='sigmoid',input_dim=29))
model.add(Dense(128,activation='sigmoid'))
model.add(Dense(64,activation='sigmoid'))
model.add(Dense(1,activation='sigmoid'))
model.compile(optimizer='rmsprop',loss='binary_crossentropy',metrics=['accuracy'])
```

Set to the number of epochs the whole network should run with the training data

```
model.fit(x_train,y_train,epochs=5) print(predictmodeln(model))
```

Output Images :

Analysing the Dataset



Using TensorFlow backend.

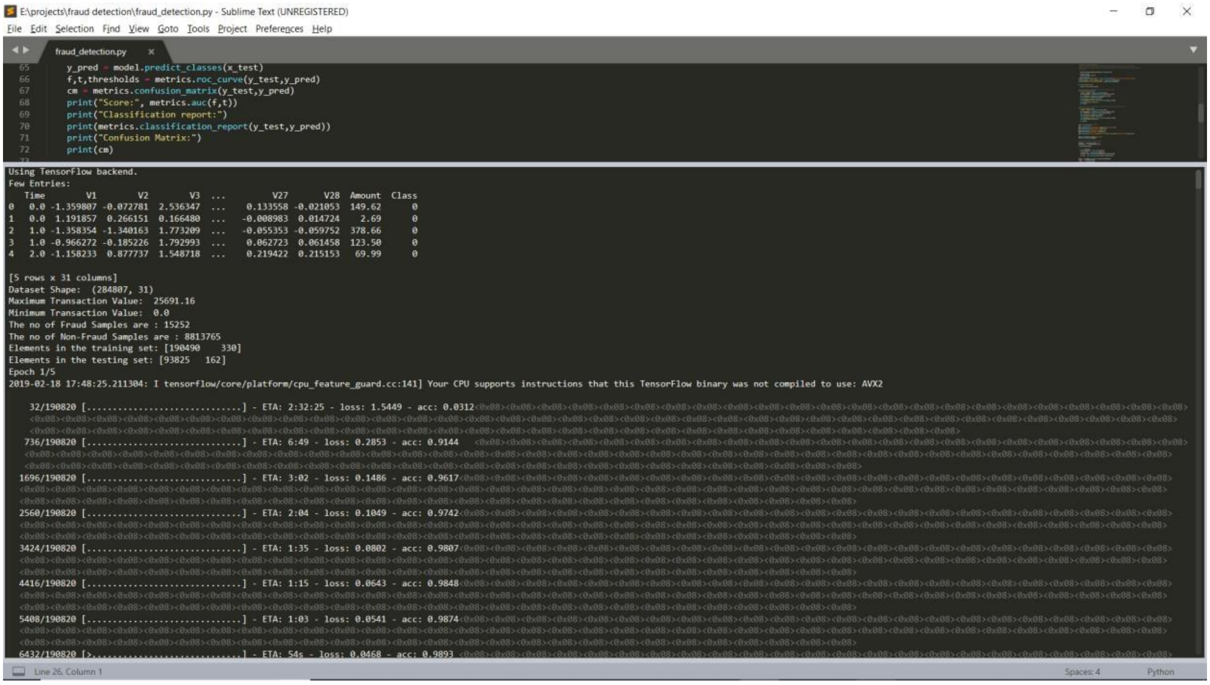
Few Entries:

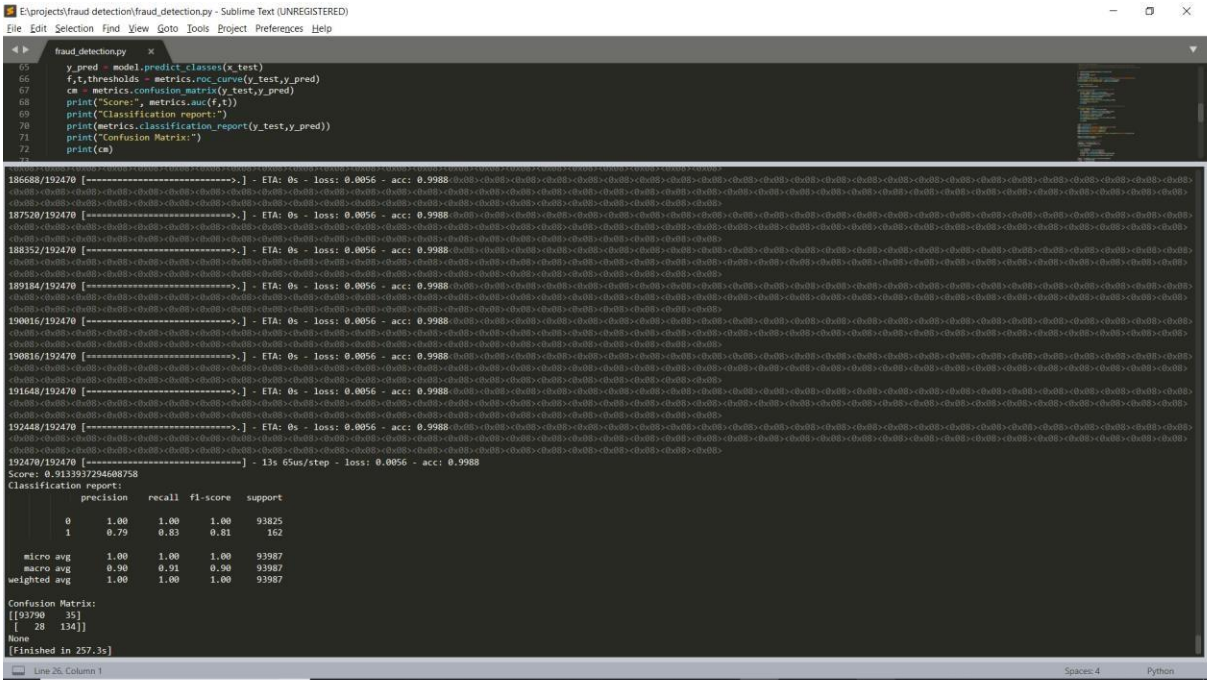
	Time	V1	V2	V3	...	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	...	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	...	-0.008983	0.014724	2.69	0

```
2   1.0 -1.358354 -1.340163  1.773209  ...   -0.055353 -0.059752  378.66      0
3   1.0 -0.966272 -0.185226  1.792993  ...    0.062723  0.061458  123.50      0
4   2.0 -1.158233  0.877737  1.548718  ...    0.219422  0.215153   69.99      0


[5 rows x 31 columns]
Dataset Shape:  (284807, 31)
Maximum Transaction Value:  25691.16
Minimum Transaction Value:  0.0
The no of Fraud Samples are : 15252
The no of Non-Fraud Samples are : 8813765
Elements in the training set: [190490    330]
Elements in the testing set: [93825   162]
poch 1/5
2019-02-18 17:48:25.211304: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU
supports instructions that this TensorFlow binary was not compiled to use: AVX2


    32/190820 [..............................] - ETA: 2:32:25 - loss: 1.5449 - acc: 0.0312
192470/192470 [==============================] - 13s 65us/step - loss: 0.0056 - acc:
0.9988
Score: 0.9133937294608758
Classification report:
              precision    recall  f1-score   support


0       1.00      1.00      1.00     93825
1       0.79      0.83      0.81       162


   micro avg      1.00      1.00      1.00     93987    macro
avg       0.90      0.91      0.90     93987 weighted avg
1.00       1.00       1.00     93987


Confusion Matrix:
[[93790    35]
```

[28 134]]

Images :





……………………………………………….THE END ………………………………………………..