CS353 ML Lab 6

Name: K V Sumanth Reddy

Roll No: 181CO225

Batch: Section 2

Date: 16/03/2021

Q: Build a KNN model to predict whether a person will be a defaulter or not in a credit scoring system.

Dataset Used: Credit Scoring Dataset
(https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients)

Importing Libraries and Dataset

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import plot_confusion_matrix,explained_variance
from sklearn.metrics import mean_squared_error
from sklearn.metrics import accuracy_score, classification_report
```

```
dataset = pd.read_csv('data.csv')
dataset = dataset[1:]
#printing 5 sample tuples
dataset.sample(5)
```

	Unnamed: 0	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	
9213	9213	80000	1	2	1	34	1	2	2	0	0	0	64575	65961	64

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 1 to 30000
Data columns (total 25 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Unnamed: 0  30000 non-null  object
 1   X1          30000 non-null  object
 2   X2          30000 non-null  object
 3   X3          30000 non-null  object
 4   X4          30000 non-null  object
 5   X5          30000 non-null  object
 6   X6          30000 non-null  object
 7   X7          30000 non-null  object
 8   X8          30000 non-null  object
 9   X9          30000 non-null  object
 10  X10         30000 non-null  object
 11  X11         30000 non-null  object
 12  X12         30000 non-null  object
 13  X13         30000 non-null  object
 14  X14         30000 non-null  object
 15  X15         30000 non-null  object
 16  X16         30000 non-null  object
 17  X17         30000 non-null  object
 18  X18         30000 non-null  object
 19  X19         30000 non-null  object
 20  X20         30000 non-null  object
 21  X21         30000 non-null  object
 22  X22         30000 non-null  object
 23  X23         30000 non-null  object
 24  Y           30000 non-null  object
dtypes: object(25)
memory usage: 5.7+ MB
```

Data Preprocessing

```
y = dataset.Y
x = dataset.drop(['Y'], axis = 1)
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size

print('Training dataset size:\nx_train -', len(x_train), '\ny_train
print('Testing dataset size:\nx_test -', len(x_test), '\ny_test -',
```

```
Training dataset size:
x_train - 22500
y_train - 22500
```

```
Testing dataset size:
x_test - 7500
y_test - 7500
```

Training the KNN Model

```
KNN = []
for i in range(1,21):
  KNNModel = KNeighborsClassifier(n_neighbors = i)
  KNN.append(KNNModel)
```

```
for i in range(20):
  KNN[i].fit(x_train, y_train)
```

Finding Accuracies for all models using the test dataset

```
train_accuracyKNN = []    #store training accuracies
test_accuracyKNN = []     #store testing accuracies
maxacc = 0
iofmax = 0

for i in range(20):
  train_accuracyKNN.append(KNN[i].score(x_train, y_train))
  test_accuracyKNN.append(KNN[i].score(x_test, y_test))
  y_pred = KNN[i].predict(x_test)
  print('K = {}'.format(i + 1))
  print("Accuracy: %.2f" %(accuracy_score(y_test, y_pred)*100))
  print("Mean Squared Error: %.2f" %(mean_squared_error(y_test, y_pr
  print("--------------------------------")
  if maxacc<accuracy_score(y_test, y_pred)*100:
    maxacc = accuracy_score(y_test, y_pred)*100
    iofmax = i
```

```
Accuracy: 77.07
Mean Squared Error: 22.93
--------------------------------
K = 7
Accuracy: 76.28
Mean Squared Error: 23.72
--------------------------------
K = 8
Accuracy: 77.35
Mean Squared Error: 22.65
--------------------------------
K = 9
Accuracy: 76.35
Mean Squared Error: 23.65
```

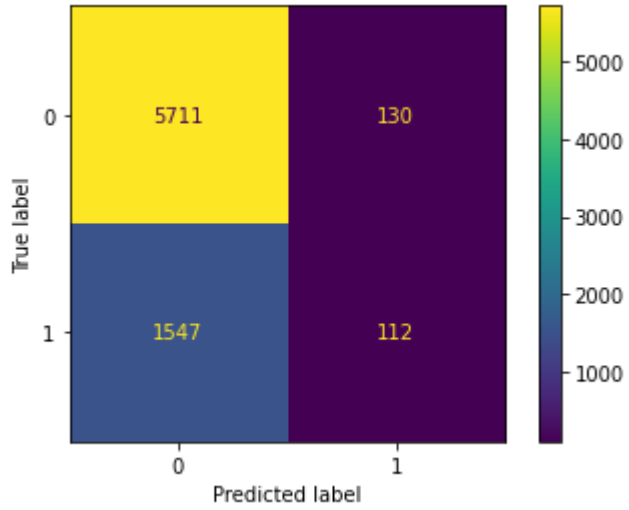
```
--------------------------------
K = 10
Accuracy: 77.52
Mean Squared Error: 22.48
--------------------------------
K = 11
Accuracy: 76.73
Mean Squared Error: 23.27
--------------------------------
K = 12
Accuracy: 77.25
Mean Squared Error: 22.75
--------------------------------
K = 13
Accuracy: 76.87
Mean Squared Error: 23.13
--------------------------------
K = 14
Accuracy: 77.47
Mean Squared Error: 22.53
--------------------------------
K = 15
Accuracy: 77.20
Mean Squared Error: 22.80
--------------------------------
K = 16
Accuracy: 77.57
Mean Squared Error: 22.43
--------------------------------
K = 17
Accuracy: 77.36
Mean Squared Error: 22.64
--------------------------------
K = 18
Accuracy: 77.51
Mean Squared Error: 22.49
--------------------------------
K = 19
Accuracy: 77.31

Mean Squared Error: 22.69
--------------------------------
K = 20
Accuracy: 77.64
Mean Squared Error: 22.36
--------------------------------
```

Results

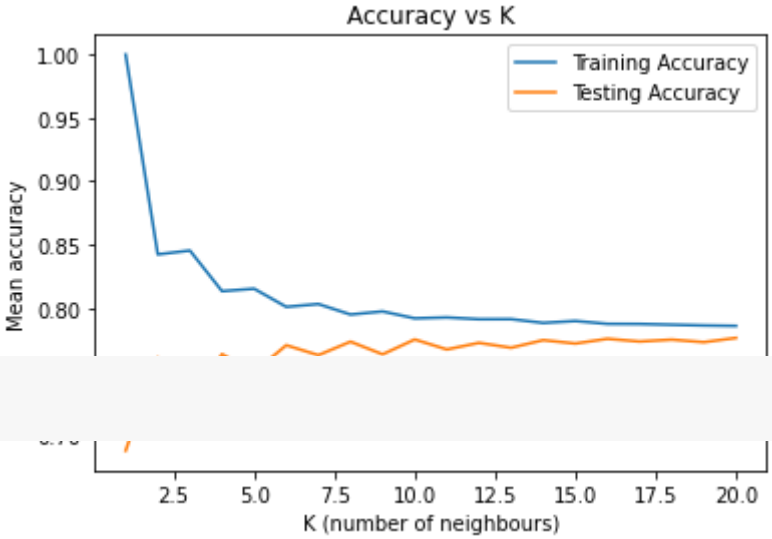
```
y_pred = KNN[iofmax].predict(x_test)

print("--------------------------------")
print('K = {}'.format(iofmax + 1))
print(classification_report(y_test,y_pred))
print("\nAccuracy: %.2f" %(accuracy_score(y_test, y_pred)*100))
print("Mean Squared Error: %.2f" %(mean_squared_error(y_test, y_pred
print("Explained Variance: %.2f" %(explained_variance_score(y_test,
```

```
print(plot_confusion_matrix(KNN[iofmax],x_test, y_test, values_forma
print("--------------------------------")
```

```
--------------------------------
K = 20
              precision    recall  f1-score   support

           0       0.79      0.98      0.87      5841
           1       0.46      0.07      0.12      1659

    accuracy                           0.78      7500
   macro avg       0.62      0.52      0.49      7500
weighted avg       0.72      0.78      0.71      7500


Accuracy: 77.64
Mean Squared Error: 22.36
Explained Variance: -9.08
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7f
--------------------------------
```




```
lst = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]

fig = plt.figure()
ax = plt.axes()
plt.plot(lst, train_accuracyKNN, label = 'Training Accuracy')
plt.plot(lst, test_accuracyKNN, label = 'Testing Accuracy')
plt.xlabel('K (number of neighbours)')
plt.ylabel('Mean accuracy')
plt.title('Accuracy vs K')
plt.legend()
plt.savefig('Graph.png')
plt.show()
```


Accuracy vs K
Training Accuracy
Testing Accuracy
Mean accuracy
1.00
0.95
0.90
0.85
0.80
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
K (number of neighbours)