KNN Assignment13

Importing the libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set(color_codes=True)
from sklearn.preprocessing import LabelEncoder
```

Importing the dataset

In [2]:
```
dataset = pd.read_csv("Social_Network_Ads.csv")
```

In [3]:
```
dataset.shape
```

Out[3]:
```
(400, 5)
```

In [4]:
```
dataset.columns
```

Out[4]:
```
Index(['User ID', 'Gender', 'Age', 'EstimatedSalary', 'Purchased'], dtype='object')
```

In [5]:
```
dataset.isna().any()
```

Out[5]:
```
User ID            False
Gender             False
Age                False
EstimatedSalary    False
Purchased          False
dtype: bool
```

In [6]:
```
dataset.isna().sum()
```

Out[6]:
```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

In [7]:
```
dataset["Gender"].unique()
```

Out[7]:
```
array(['Male', 'Female'], dtype=object)
```

In [8]:
```
dataset["Gender"].replace({'Male':'0', 'Female':'1'},inplace=True)
```

In [9]:
```
dataset
```

Out[9]:

| | User ID | Gender | Age | EstimatedSalary | Purchased |
|---|---|---|---|---|---|
| **0** | 15624510 | 0 | 19 | 19000 | 0 |
| **1** | 15810944 | 0 | 35 | 20000 | 0 |
| **2** | 15668575 | 1 | 26 | 43000 | 0 |
| **3** | 15603246 | 1 | 27 | 57000 | 0 |
| **4** | 15804002 | 0 | 19 | 76000 | 0 |
| **...** | ... | ... | ... | ... | ... |
| **395** | 15691863 | 1 | 46 | 41000 | 1 |
| **396** | 15706071 | 0 | 51 | 23000 | 1 |
| **397** | 15654296 | 1 | 50 | 20000 | 1 |
| **398** | 15755018 | 0 | 36 | 33000 | 0 |
| **399** | 15594041 | 1 | 49 | 36000 | 1 |

400 rows × 5 columns

Splitting the dataset into the Training set and Test set

In [10]:
```
from sklearn.model_selection import train_test_split
```

In [11]:
```
X_train, X_test, y_train, y_test = train_test_split(dataset[['Gender','Age','EstimatedSalary']],dataset['Purchased'],test_size=0.2)
```

In [12]:
```
len(X_train)
```

Out[12]:
```
320
```

In [13]:
```
len(X_test)
```

Out[13]:
```
80
```

Feature Scaling

In [14]:
```
from sklearn.preprocessing import StandardScaler
```

In [15]:
```
sc_X = StandardScaler()
```

In [16]:
```
X_train = sc_X.fit_transform(X_train)
```

In [17]:
```
X_test = sc_X.transform(X_test)
```

Fitting K-NN to the Training set

In [20]:
```
import math
def euclideanDistance(instance1, instance2, length):
    distance=0
    for X in range(length):
        distance += pow((instance1[X] - instance2[X]), 2)
    return math.sqrt(distance)
```

In [21]:
```
import operator
def getNeighbors(trainingSet, testInstance, k):
    distances=[]
    length = len(testInstance)-1
    for X in range(len(trainingSet)):
        dist = euclideanDistance(testInstance, trainingSet[X], length)
        distances.append((trainingSet[X], dist))
    distances.sort(key=operator.itemgetter(1))
    neighbors = []
    for X in range(k):
        neighbors.append(distances[X][0])
    return neighbors
```

In [22]:
```
import operator
def getResponse(neighbors):
    classVotes = {}
    for X in range(len(neighbors)):
        response = neighbors[X][-1]
        if response in classVotes:
            classVotes[response] += 1
        else:
            classVotes[response] = 1
    sortedVotes = sorted(classVotes.iteritems(), key=operator.itemgetter(1), reverse=True)
    return sortedVotes[0][0]
```

Predicting the Test set results

In [23]:
```
def getAccuracy(testSet, predictions):
    correct = 0
    for X in range(len(testSet)):
        if testSet[X][-1] is predictions[X]:
            correct += 1
    return (correct/float(len(testSet))) * 100.0
```

Making the Confusion Matrix

In [32]:
```
# confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

# actual values
actual = [0, 0, 0, 1, 0, 0, 0, 1, 0]
# predicted values
predicted = [0, 0, 0, 1, 0, 0, 1, 0, 1]
# confusion matrix
matrix = confusion_matrix(actual,predicted, labels=[1,0])
print('Confusion matrix : \n',matrix)

# outcome values order in sklearn
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)

# classification report for precision, recall f1-score and accuracy
matrix = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n',matrix)
```

```
Confusion matrix :
 [[1 1]
 [2 5]]
Outcome values :
 1 1 2 5
Classification report :
               precision    recall  f1-score   support

           1       0.33      0.50      0.40         2
           0       0.83      0.71      0.77         7

    accuracy                           0.67         9
   macro avg       0.58      0.61      0.58         9
weighted avg       0.72      0.67      0.69         9
```