Name :- Shounak Lohokare

Roll num :- 21611

K - nearest neighbour algorithm

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

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
df=pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Data Mining Data/Social_Network_Ads
df
```

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

400 rows × 5 columns

```
a = pd.DataFrame(df,columns = ['Age','EstimatedSalary'])
a
```

	Age	EstimatedSalary
0	19	19000
1	35	20000
2	26	43000
3	27	57000
4	19	76000
...	...	...
395	46	41000
396	51	23000
397	50	20000
398	36	33000
399	49	36000

```
x = a.to_numpy()
y = df.Purchased
print(x)
print(y)
```

```
[[    19  19000]
 [    35  20000]
 [    26  43000]
 [    27  57000]
 [    19  76000]
 [    27  58000]
 [    27  84000]
 [    32 150000]
 [    25  33000]
 [    35  65000]
 [    26  80000]
 [    26  52000]
 [    20  86000]
 [    32  18000]
 [    18  82000]
 [    29  80000]
 [    47  25000]
 [    45  26000]
 [    46  28000]
 [    48  29000]
 [    45  22000]
 [    47  49000]
 [    48  41000]
 [    45  22000]
 [    46  23000]
 [    47  20000]
 [    49  28000]
 [    47  30000]
 [    29  43000]
 [    31  18000]
 [    31  74000]
 [    27 137000]
```

```
[    21  16000]
[    28  44000]
[    27  90000]
[    35  27000]
[    33  28000]
[    30  49000]
[    26  72000]
[    27  31000]
[    27  17000]
[    33  51000]
[    35 108000]
[    30  15000]
[    28  84000]
[    23  20000]
[    25  79000]
[    27  54000]
[    30 135000]
[    31  89000]
[    24  32000]
[    18  44000]
[    29  83000]
[    35  23000]
[    27  58000]
[    24  55000]
[    23  48000]
[    28  79000]
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.15)
print(x_train)
```

```
[[    38  50000]
 [    52 114000]
 [    20  74000]
 [    37  53000]
 [    45 131000]
 [    47  49000]
 [    39  73000]
 [    27  57000]
 [    41  30000]
 [    26  81000]
 [    52  21000]
 [    36  75000]
 [    47 113000]
 [    26  35000]
 [    35  55000]
 [    41  63000]
 [    35  20000]
 [    51 146000]
 [    30  62000]
 [    35 147000]
 [    26  32000]
 [    26  86000]
 [    35  57000]
 [    42 104000]
 [    60  42000]
 [    39  75000]
 [    35  58000]
 [    28  89000]
 [    37  33000]
```

```
[    35  72000]
[    45  26000]
[    27  96000]
[    42  70000]
[    30 107000]
[    33  43000]
[    25  87000]
[    37  77000]
[    60 108000]
[    46  74000]
[    39  96000]
[    40  72000]
[    20  82000]
[    37  78000]
[    29  61000]
[    48 131000]
[    37  79000]
[    45  22000]
[    24  23000]
[    33  69000]
[    26  15000]
[    41  72000]
[    19  21000]
[    54 108000]
[    37  71000]
[    50  88000]
[    28  59000]
[    19  85000]
[    46  23000]
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(x_train)
x_train = scaler.transform(x_train)
x_test = scaler.transform(x_test)
print(x_train)
```

```
[[ 0.03902949 -0.60714766]
 [ 1.38526409  1.25862457]
 [-1.69184357  0.09251692]
 [-0.05713012 -0.51968959]
 [ 0.71214679  1.75422032]
 [ 0.90446602 -0.63630035]
 [ 0.1351891   0.06336423]
 [-1.01872627 -0.40307883]
 [ 0.32750833 -1.19020149]
 [-1.11488588  0.29658576]
 [ 1.38526409 -1.45257571]
 [-0.15328974  0.12166961]
 [ 0.90446602  1.22947188]
 [-1.11488588 -1.04443803]
 [-0.24944935 -0.46138421]
 [ 0.32750833 -0.22816268]
 [-0.24944935 -1.4817284 ]
 [ 1.28910448  2.19151068]
 [-0.73024742 -0.25731537]
 [-0.24944935  2.22066337]
 [-1.11488588 -1.1318961 ]
 [-1.11488588  0.44234922]
```

```
 [-0.24944935 -0.40307883]
 [ 0.42366795  0.96709766]
 [ 2.15454101 -0.84036919]
 [ 0.1351891   0.12166961]
 [-0.24944935 -0.37392613]
 [-0.92256665  0.52980729]
 [-0.05713012 -1.10274341]
 [-0.24944935  0.03421154]
 [ 0.71214679 -1.30681225]
 [-1.01872627  0.73387613]
 [ 0.42366795 -0.02409384]
 [-0.73024742  1.05455573]
 [-0.44176858 -0.8112165 ]
 [-1.2110455   0.47150191]
 [-0.05713012  0.179975  ]
 [ 2.15454101  1.08370842]
 [ 0.80830641  0.09251692]
 [ 0.1351891   0.73387613]
 [ 0.23134872  0.03421154]
 [-1.69184357  0.32573845]
 [-0.05713012  0.20912769]
 [-0.82640704 -0.28646806]
 [ 1.00062563  1.75422032]
 [-0.05713012  0.23828038]
 [ 0.71214679 -1.42342301]
 [-1.30720511 -1.39427032]
 [-0.44176858 -0.05324653]
 [-1.11488588 -1.62749185]
 [ 0.32750833  0.03421154]
 [-1.78800318 -1.45257571]
 [ 1.57758332  1.08370842]
 [-0.05713012  0.00505885]
 [ 1.19294486  0.5006546 ]
 [-0.92256665 -0.34477344]
 [-1.78800318  0.41319652]
 [ 0.80830641 -1.39427032]
```

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=10,metric='manhattan')
classifier.fit(x_train, y_train)
```

```
KNeighborsClassifier(metric='manhattan', n_neighbors=10)
```

```
y_pred = classifier.predict(x_test)
print(y_pred,"  ",y_test)
```

```
[0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 1 0 1 1 1 0 0 0 0 1 0 0
 0 0 1 0 0 1 1 0 1 1 0 0 1 0 1 0 0 0 0 0 0 0 0]    374    0
116    0
316    1
8      0
72     0
187    0
175    0
369    1
256    0
396    1
264    1
```

```
78     0
33     0
237    0
39     0
274    1
343    1
276    0
233    1
22     1
68     0
125    0
393    1
398    0
193    0
31     1
58     0
290    1
7      1
391    1
44     0
110    0
118    0
301    1
361    1
251    0
177    0
11     0
151    0
375    1
267    0
76     0
26     1
219    1
330    0
329    1
291    1
353    0
83     0
371    1
244    0
91     0
325    0
124    0
134    0
49     0
322    0
```

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[38  1]
 [ 2 19]]
              precision    recall  f1-score   support

           0       0.95      0.97      0.96        39
           1       0.95      0.90      0.93        21

    accuracy                           0.95        60
   macro avg       0.95      0.94      0.94        60
```

```
weighted avg       0.95      0.95      0.95        60
```

```python
error = []

# Calculating error for K values between 1 and 40
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))
    print(error)
```

```
[0.1]
[0.1, 0.13333333333333333]
[0.1, 0.13333333333333333, 0.08333333333333333]
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667]
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
[0.1, 0.13333333333333333, 0.08333333333333333, 0.06666666666666667, 0.06666666666666
```

```python
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')
```

```
Text(0, 0.5, 'Mean Error')
```