Social Network ads

Shreyas D

1832048

PROBLEM STATEMENT AND ANALYSIS:

Here we have a problem to classify the customers or buyers who have brought the specific products. The data set contains information about the users like Gender, Age, and Salary. The **Purchased column** contains the **labels** for the users. This is a **binary classification** (we have two classes). If the **label is 1** it means that the user **has bought product X** and **0** means the the users **hasn't bought** that specific product.

SAMPLE DATASET:

	A	B	C	D	E
1	User ID	Gender	Age	Estimated	Purchased
2	15624510	Male	19	19000	0
3	15810944	Male	35	20000	0
4	15668575	Female	26	43000	0
5	15603246	Female	27	57000	0
6	15804002	Male	19	76000	0
7	15728773	Male	27	58000	0
8	15598044	Female	27	84000	0
9	15694829	Female	32	150000	1
10	15600575	Male	25	33000	0
11	15727311	Female	35	65000	0
12	15570769	Female	26	80000	0
13	15606274	Female	26	52000	0
14	15746139	Male	20	86000	0
15	15704987	Male	32	18000	0
16	15628972	Male	18	82000	0
17	15697686	Male	29	80000	0
18	15733883	Male	47	25000	1
19	15617482	Male	45	26000	1
20	15704583	Male	46	28000	1
21	15621083	Female	48	29000	1
22	15649487	Male	45	22000	1
23	15736760	Female	47	49000	1
24	15714658	Male	48	41000	1
25	15599081	Female	45	22000	1
26	15705113	Male	46	23000	1
27	15631159	Male	47	20000	1
28	15792818	Male	49	28000	1
29	15633531	Female	47	30000	1
30	15744529	Male	29	43000	0
31	15669656	Male	31	18000	0

ALGORITHM:

- Load the data.
- Initialize K to your chosen number of neighbours.
- For each example in the data,
    1. Calculate the distance between the query example and the current example from the data.
    2. Add the distance and the index of the example to an ordered collection.
- Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances.
- Pick the first K entries from the sorted collection.
- Get the labels of the selected K entries.
- If regression, return the mean of the K labels.
- If classification, return the mode of the K labels.

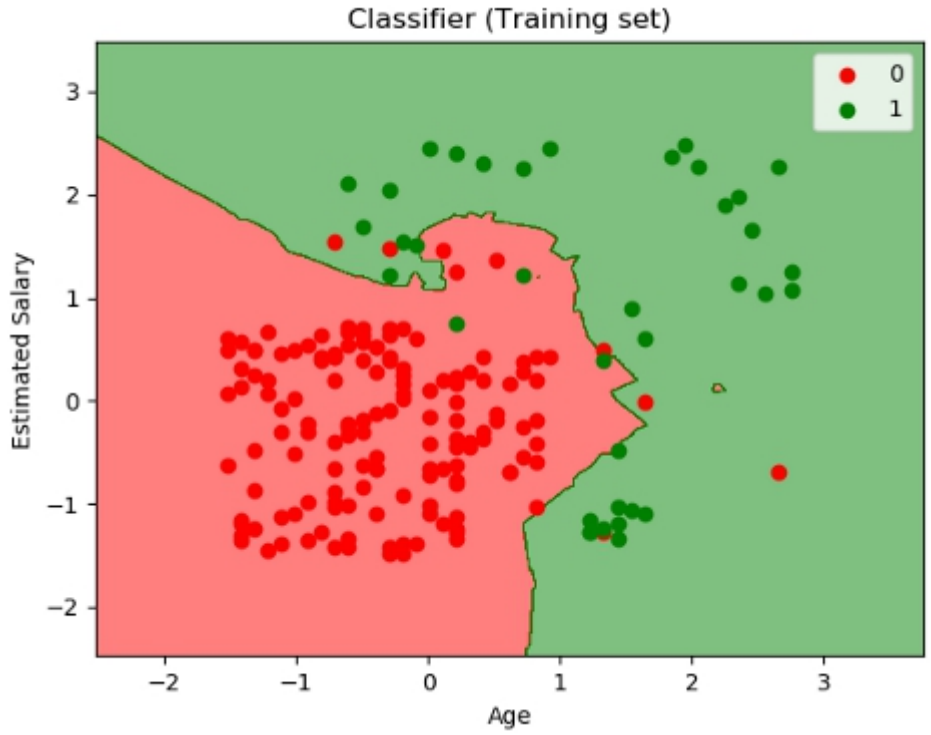
INBUILT CODE:

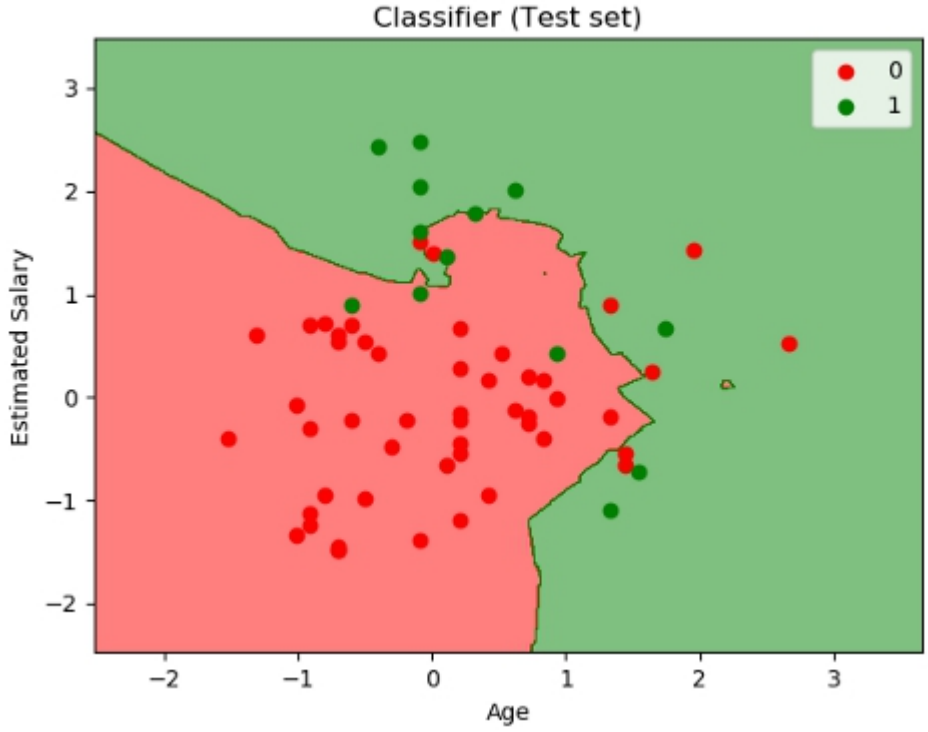
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('ggplot')
#Read the dataset
df = pd.read_csv(r'D:/Users/Downloads/ Social_Network_Ads.csv')
')
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=42,
stratify=y)
from sklearn.neighbors import KNeighborsClassifier
#Setup arrays to store training and test accuracies
neighbors = np.arange(1,9)
train_accuracy =np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))
for i,k in enumerate(neighbors):
    #Setup a knn classifier with k neighbors
    knn = KNeighborsClassifier(n_neighbors=k)

    #Fit the model
    knn.fit(X_train, y_train)

    #Compute accuracy on the training set
    train_accuracy[i] = knn.score(X_train, y_train)
```

```
    #Compute accuracy on the test set
    test_accuracy[i] = knn.score(X_test, y_test)

plt.title('KNN Varying number of neighbors')
plt.plot(neighbors, test_accuracy, label='Testing Accuracy')
plt.plot(neighbors, train_accuracy, label='Training accuracy')
plt.legend()
plt.xlabel('Number of neighbors')
plt.ylabel('Accuracy')
plt.show()
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train,y_train)
print("Accuracy", knn.score(X_test,y_test)*100)
from sklearn.metrics import confusion_matrix
y_pred = knn.predict(X_test)
confusion_matrix(y_test,y_pred)
pd.crosstab(y_test, y_pred, rownames=['True'], colnames=['Predicted'], margins=True)
```

OUTPUT:

```
Accuracy 84%
              [43,  7]
              [ 5,  8]
```





OUTPUT:

```
Accuracy 80.23%
```

CONCLUSION:

Comparing both the in-built and scratch models, we infer that the inbuilt model has performed well with an accuracy of 84%. So the In-built model suits well for KNN classification. This model can be implemented with large data and many features for real time applications. Happy ML!