

# LAB3

# AMIL

Student Name: Joud Alahmari

Student ID: 2008071

# IMPORT THE DATA SET

The screenshot shows the Spyder Python IDE interface. The left pane contains a Python script for importing and preprocessing data. The right pane shows the Variable Explorer with a list of variables and their types. The bottom pane shows the console output.

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Jan 2 17:35:33 2023
4
5 @author: joude
6 """
7
8 # 1-import Libraries
9 import numpy as np
10 import matplotlib.pyplot as plt
11 import pandas as pd
12
13 # 2-import dataset
14 mydataset= pd.read_csv("C:\\Users\\joude\\Desktop\\ML\\Social_Network_Ads.csv")
15 X = mydataset.iloc[:,2:5].values
16 y = mydataset.iloc[:, -1].values
17
18 # 4-splitting the dataset into training set and test set
19 from sklearn.model_selection import train_test_split
20 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0 )
21
22 # 5-feature Scaling
23 from sklearn.preprocessing import StandardScaler
24 sc = StandardScaler()
25 X_train = sc.fit_transform(X_train)
26 X_test = sc.fit_transform(X_test)
27
28 # Fitting the knn classifier to the Training set
29 from sklearn.neighbors import KNeighborsClassifier
30 knn = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
31 knn.fit(X_train, y_train)
32
33 # Predicting the Test set results
34 y_pred = knn.predict(X_test)
35
36 # 8-making the confusion matrix
37 from sklearn.metrics import confusion_matrix
38 cm=confusion_matrix(y_test, y_pred)
```

Variable Explorer:

Name	Type	Size	Value
cm	Array of int64	(2, 2)	[[54 4] [ 1 21]]
i	int	1	1
j	int64	1	1
knn	neighbors_classification.KNeighborsClassifier	1	KNeighborsClassifier object of sklearn.neighbors_classification modul ...
mydataset	DataFrame	(400, 5)	Column names: User ID, Gender, Age, EstimatedSalary, Purchased
sc	preprocessing_data.StandardScaler	1	StandardScaler object of sklearn.preprocessing_data module
X	Array of int64	(400, 2)	[[ 19 19000] [ 35 20000]]
X1	Array of float64	(592, 609)	[[-2.96547978 -2.95547978 -2.9547978 ... -
X2	Array of float64	(592, 609)	[[-2.59138156 -2.59138156 -2.59138156 ... -
X_set	Array of float64	(320, 2)	[[ 1.92295008  2.14601566] [ 2.02916082  0.3787193 ]

Console:

```
C:\Users\joude\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning:
Unlike other reduction functions (e.g. 'skew', 'kurtosis'), the default behavior of 'mode' typically
preserves the axis it acts along. In SciPy 1.11.0, this behavior will change: the default value of
'keepdims' will become False, the 'axis' over which the statistic is taken will be eliminated, and the
value None will no longer be accepted. Set 'keepdims' to True or False to avoid this warning.
mode, _ = stats.mode(y[neigh_ind, k], axis=1)
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-
mapping will have precedence in case its length matches with *x* & *y*. Please use the *color*
keyword-argument or provide a 2D array with a single row if you intend to specify the same RGB or RGBA
value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-
mapping will have precedence in case its length matches with *x* & *y*. Please use the *color*
keyword-argument or provide a 2D array with a single row if you intend to specify the same RGB or RGBA
value for all points.
```

The screenshot shows a DataFrame viewer window titled "mydataset - DataFrame". It displays a table with 11 rows and 6 columns. The columns are Index, User ID, Gender, Age, EstimatedSale, and Purchased. The data is as follows:

Index	User ID	Gender	Age	EstimatedSale	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
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0
10	15570769	Female	26	80000	0
11	15606374	Female	26	52000	0

At the bottom of the window, there are checkboxes for "Format", "Resize", "Background color", and "Column min/max". There are also buttons for "Save and Close" and "Close".

```
16 X = mydataset.iloc[:, [2, 3]].values
17 y = mydataset.iloc[:, -1].values
18
```

X - NumPy object array

	0	1
0	19	19000
1	35	20000
2	26	43000
3	27	57000
4	19	76000
5	27	58000
6	27	84000
7	32	150000
8	25	33000
9	35	65000
10	26	80000
11	26	52000

y - NumPy object array

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0
9	0
10	0
11	0

## SPLIT THE DATA SET

```
# 4-splitting the dataset into 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 = 0.2, random_state = 0 )
```

X_test - NumPy object array			X_train - NumPy object array			y_test - NumPy object array			y_train - NumPy object array		
	0	1		0	1		0			0	
0	-0.496186	0.560214	0	1.92295	2.14602	0	0	0	1		
1	0.238904	-0.591337	1	2.02016	0.378719	1	0	1	0		
2	-0.0367545	0.186738	2	-1.38222	-0.432499	2	0	2	0		
3	-0.496186	0.31123	3	-1.18779	-1.01194	3	0	3	0		
4	-0.0367545	-0.591337	4	1.92295	-0.925024	4	0	4	1		
5	-0.771845	-1.52503	5	0.367578	0.291803	5	0	5	1		
6	-0.4043	-1.68064	6	0.173157	0.146943	6	0	6	0		
7	0.0551318	2.33422	7	2.02016	1.74041	7	1	7	1		
8	-1.59882	-0.031123	8	0.756421	-0.838108	8	0	8	1		
9	1.06588	-0.809198	9	0.270367	-0.287638	9	0	9	0		
10	-0.496186	-0.62246	10	0.367578	-0.17175	10	0	10	0		
11	-0.679959	-0.435722	11	0.118436	2.20206	11	0	11	1		

## FEATURE SCALING

```
# 5-feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

## FITTING THE KNN TO THE TRAINING SET

```
# Fitting the knn classifier to the Training set
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
knn.fit(X_train, y_train)
```

## PREDICT THE TEST SET RESULTS

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

y\_pred - NumPy object array

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0
9	1
10	0
11	0

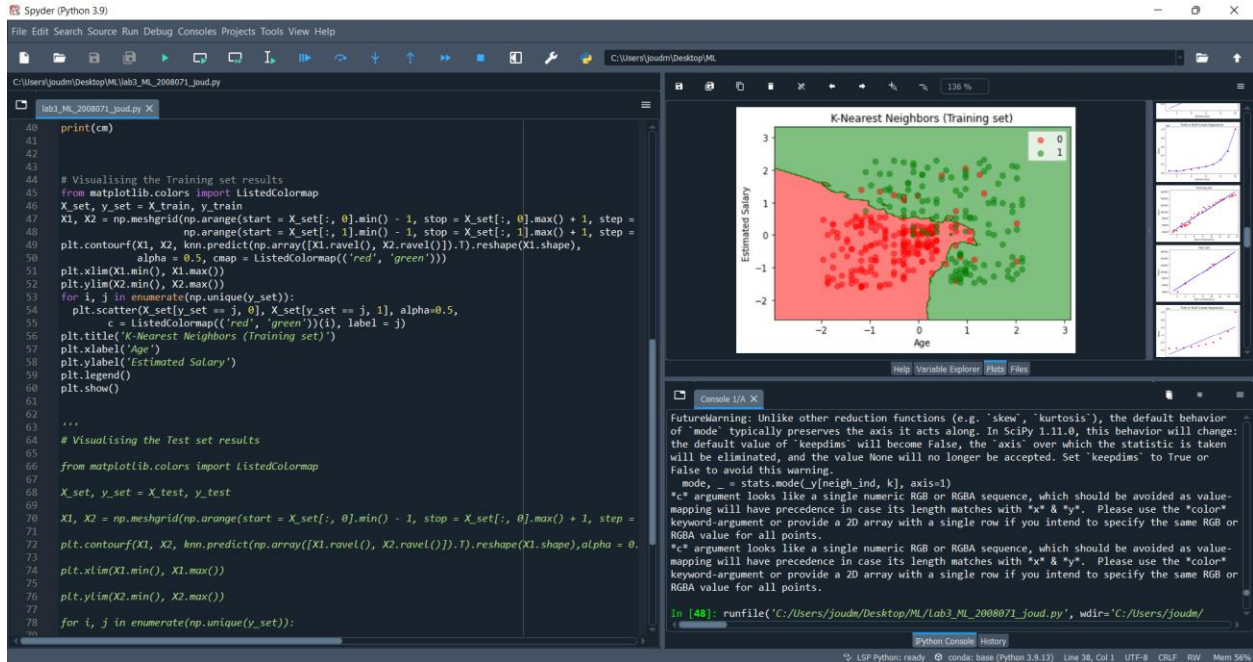
## MAKE THE CONFUSION MATRIX

```
# 8-making the confusion matrix
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test, y_pred)
print(cm)
```

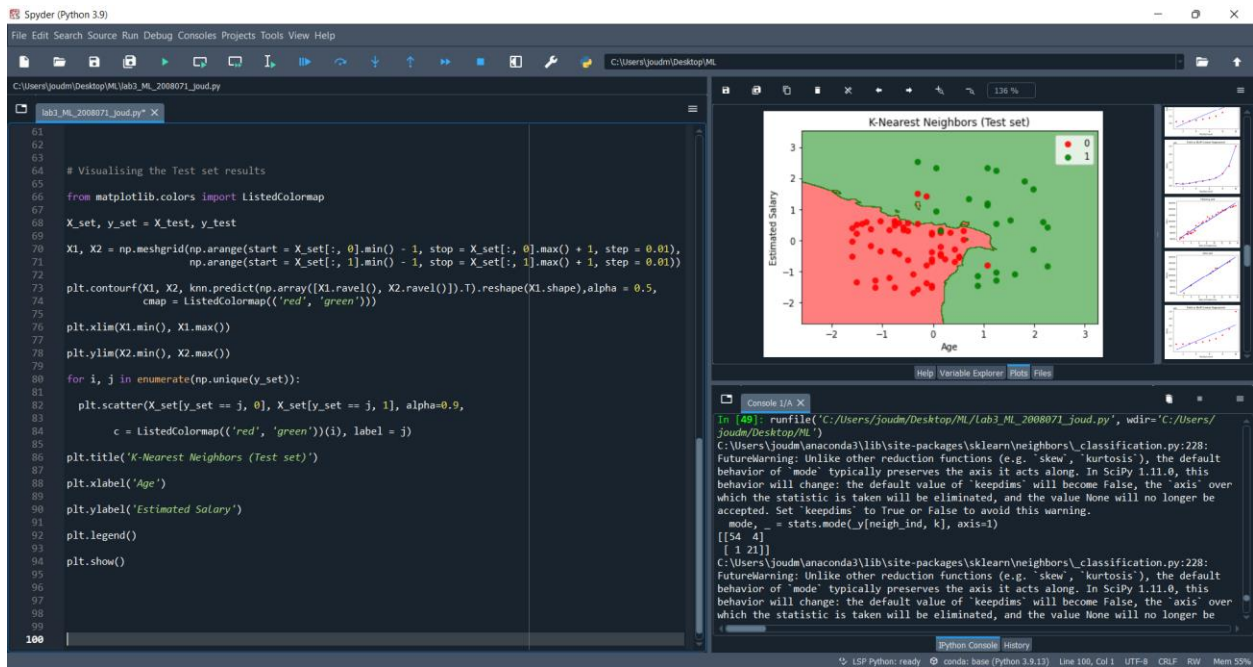
cm - NumPy object array

	0	1
0	54	4
1	1	21

# Visualising the Training set results



# Visualising the Test set results



9. Write your opinion about the two models' performance.

In the test set model the nodes are less and not crowded as the training set and easy to view

and shows the predicted data