EXPERIMENT – 9

LOGISTIC REGRESSION

Aim:

To perform model classification using Logistic Regression

Procedure:

- Upload the given dataset
- Import all the necessities
- Read and make it as DataFrame
- Through sklearn and train the model
- Test the model

Program:

```
from google.colab import files
uploaded=files.upload()
import numpy as np
import pandas as pd
file=next(iter(uploaded))
df=pd.read_csv(file)
df
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Social_Network_Ads - Social_Network_Ads.csv to Social_Network_Ads - Social_Network_Ads.csv
```

	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

```
df.head()
```

	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

```
features=df.iloc[:,[2,3]].values
labels=df.iloc[:,4].values
features
```

```
array([[    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],
```

```
[    22,  18000],
[    32, 117000],
[    27,  20000],
[    25,  87000],
[    23,  66000],
[    32, 120000],
[    59,  83000],
[    24,  58000],
[    24,  19000],
[    23,  82000],
[    22,  63000],
[    31,  68000],
[    25,  80000],
[    24,  27000],
[    20,  23000],
[    33, 113000],
[    32,  18000],
[    34, 112000],
[    18,  52000],
[    22,  27000],
[    28,  87000],
[    26,  17000],
[    30,  80000],
[    39,  42000],
[    20,  49000],
[    35,  88000],
[    30,  62000],
[    31, 118000],
[    24,  55000],
[    28,  85000],
[    26,  81000],
```

```
[    60,  46000],
[    60,  83000],
[    39,  73000],
[    59, 130000],
[    37,  80000],
[    46,  32000],
[    46,  74000],
[    42,  53000],
[    41,  87000],
[    58,  23000],
[    42,  64000],
[    48,  33000],
[    44, 139000],
[    49,  28000],
[    57,  33000],
[    56,  60000],
[    49,  39000],
[    39,  71000],
[    47,  34000],
[    48,  35000],
[    48,  33000],
[    47,  23000],
[    45,  45000],
[    60,  42000],
[    39,  59000],
[    46,  41000],
[    51,  23000],
[    50,  20000],
[    36,  33000],
[    49,  36000]])
```

```
labels
```

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0,
       1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0,
       1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1,
       1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0,
       1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 0, 1])
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
for i in range(1,401):
  x_train,x_test,y_train,y_test=train_test_split(features,labels,test_size=0.2,random_state=42)
  model=LogisticRegression()
  model.fit(x_train,y_train)
  train_score=model.score(x_train,y_train)
  test_score=model.score(x_test,y_test)
  if test_score>train_score:
    print("Test {} Train {} RandomState {}".format(test_score,train_score,i))
```

```
Test 0.8875 Train 0.8375 RandomState 1
Test 0.8875 Train 0.8375 RandomState 2
Test 0.8875 Train 0.8375 RandomState 3
Test 0.8875 Train 0.8375 RandomState 4
Test 0.8875 Train 0.8375 RandomState 5
Test 0.8875 Train 0.8375 RandomState 6
Test 0.8875 Train 0.8375 RandomState 7
Test 0.8875 Train 0.8375 RandomState 8
Test 0.8875 Train 0.8375 RandomState 9
Test 0.8875 Train 0.8375 RandomState 10
Test 0.8875 Train 0.8375 RandomState 11
Test 0.8875 Train 0.8375 RandomState 12
Test 0.8875 Train 0.8375 RandomState 13
Test 0.8875 Train 0.8375 RandomState 14
Test 0.8875 Train 0.8375 RandomState 15
Test 0.8875 Train 0.8375 RandomState 16
Test 0.8875 Train 0.8375 RandomState 17
Test 0.8875 Train 0.8375 RandomState 18
Test 0.8875 Train 0.8375 RandomState 19
Test 0.8875 Train 0.8375 RandomState 20
```

```
Test 0.8875 Train 0.8375 RandomState 371
Test 0.8875 Train 0.8375 RandomState 372
Test 0.8875 Train 0.8375 RandomState 373
Test 0.8875 Train 0.8375 RandomState 374
Test 0.8875 Train 0.8375 RandomState 375
Test 0.8875 Train 0.8375 RandomState 376
Test 0.8875 Train 0.8375 RandomState 377
Test 0.8875 Train 0.8375 RandomState 378
Test 0.8875 Train 0.8375 RandomState 379
Test 0.8875 Train 0.8375 RandomState 380
Test 0.8875 Train 0.8375 RandomState 381
Test 0.8875 Train 0.8375 RandomState 382
Test 0.8875 Train 0.8375 RandomState 383
Test 0.8875 Train 0.8375 RandomState 384
Test 0.8875 Train 0.8375 RandomState 385
Test 0.8875 Train 0.8375 RandomState 386
Test 0.8875 Train 0.8375 RandomState 387
Test 0.8875 Train 0.8375 RandomState 388
Test 0.8875 Train 0.8375 RandomState 389
Test 0.8875 Train 0.8375 RandomState 390
Test 0.8875 Train 0.8375 RandomState 391
Test 0.8875 Train 0.8375 RandomState 392
Test 0.8875 Train 0.8375 RandomState 393
Test 0.8875 Train 0.8375 RandomState 394
Test 0.8875 Train 0.8375 RandomState 395
Test 0.8875 Train 0.8375 RandomState 396
Test 0.8875 Train 0.8375 RandomState 397
Test 0.8875 Train 0.8375 RandomState 398
Test 0.8875 Train 0.8375 RandomState 399
Test 0.8875 Train 0.8375 RandomState 400
```

```
x_train,x_test,y_train,y_test=train_test_split(features,labels,test_size=0.2,random_state=42)
finalModel=LogisticRegression()
finalModel.fit(x_train,y_train)
```

```
LogisticRegression()
```

```
print(finalModel.score(x_test,y_test))
print(finalModel.score(x_train,y_train))
```

```
0.8875
0.8375
```

```
from sklearn.metrics import classification_report
print(classification_report(labels,finalModel.predict(features)))
```

```
              precision    recall  f1-score   support

           0       0.85      0.93      0.89       257
           1       0.85      0.70      0.77       143

    accuracy                           0.85       400
   macro avg       0.85      0.81      0.83       400
weighted avg       0.85      0.85      0.84       400
```

Result:

Thus the python program to perform the model classification using Logistic Regression is executed and verified successfully