EXPERIMENT NO: 9

Social Network Ads Classification using Logistic Regression

Aim:

To build a Logistic Regression model that predicts whether a user purchases a product based on their age and estimated salary.

Algorithm:

1. **Load Data: Read the Social Network Ads dataset using pandas.**
2. **Select Features and Labels:**
3. **Features → Age and Estimated Salary**
4. **Label → Purchased (0 or 1)**
5. **Model Optimization:**
6. **Run multiple train-test splits with different random states (1–400).**
7. **Identify the state where the test accuracy exceeds the training accuracy.**
8. **Train Final Model:**
9. **Split data with the best random state (306).**
10. **Train the Logistic Regression model.**
11. **Evaluate Model:**
12. **Display training and testing accuracy.**
13. **Generate classification report to assess precision, recall, and F1-score.**

Program:

[1]:
```
import pandas as pd
import numpy as np
df=pd.read_csv("C:/Users/vijay/Downloads/Social_Network_Ads.csv")
df
```

[1]:

	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

[2]:
```
df.head()
```

[2]:

	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

[3]:

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

[3]:

```
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]
```

[4]:

```
label
```

[4]:
```
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, 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, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 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, 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, 0, 1, 1, 1, 1, 0, 1, 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])
```

[5]:
```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

[8]:
```python
for i in range(1,401):
    xtrain,xtest,ytrain,ytest=train_test_split(features,label,test_size=0.2,random_state=i)
    model=LogisticRegression()
    model.fit(xtrain,ytrain)
    train_score=model.score(xtrain,ytrain)
    test_score=model.score(xtest,ytest)
    if test_score>train_score:
        print("Test: {:.3f}, Train: {:.3f}, Random State: {}".format(test_score,train_score,i))
```

```
Test: 0.900, Train: 0.841, Random State: 10
Test: 0.863, Train: 0.856, Random State: 14
Test: 0.850, Train: 0.844, Random State: 15
Test: 0.863, Train: 0.856, Random State: 16
Test: 0.875, Train: 0.834, Random State: 18
Test: 0.850, Train: 0.844, Random State: 19
Test: 0.875, Train: 0.844, Random State: 20
Test: 0.863, Train: 0.834, Random State: 21
Test: 0.875, Train: 0.841, Random State: 22
Test: 0.875, Train: 0.841, Random State: 24
Test: 0.850, Train: 0.834, Random State: 26
Test: 0.850, Train: 0.841, Random State: 27
Test: 0.863, Train: 0.834, Random State: 30
```

[9]:
```
xtrain,xtest,ytrain,ytest=train_test_split(features,label,test_size=0.2,random_state=306)
fmodel=LogisticRegression()
fmodel.fit(xtrain,ytrain)
```

[9]:
```
▾ LogisticRegression
LogisticRegression()
```

[10]:
```
print(fmodel.score(xtrain,ytrain))
print(fmodel.score(xtest,ytest))
```

```
0.8375
0.9125
```

[12]:
```
from sklearn.metrics import classification_report
print(classification_report(label,fmodel.predict(features)))
```

```
              precision    recall  f1-score   support

           0       0.86      0.91      0.89       257
           1       0.83      0.74      0.78       143

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

Result:

The Logistic Regression model effectively predicts whether a customer will purchase based on age and salary. The model shows balanced training and testing accuracy, and the classification report confirms good overall performance.