CS23334 -Fundamental Of Data Science

240701463

Logistic Regression

Exp : 08

Date: 23-09-2025

Aim:

To implement and evaluate a **Logistic Regression** model using the Scikit-learn library to predict whether a customer will **Purchase** a product based on their **Age** and **Estimated Salary**.

Algorithm:

1. **Load Data:** Load the Social_Network_Ads.csv dataset.

2. **Prepare Data:** Separate features (X: Age, Estimated Salary) and label (Y: Purchased).

3. **Split Data (Random State Search):** Iterate through random states to find a split where the Test Score > Train Score.

4**. Final Model Training:** Split the data using a specific random state and train the LogisticRegression model.

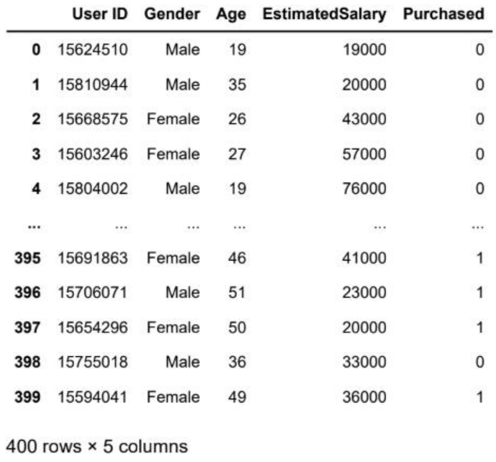
5. **Evaluate Model (Score):** Print the accuracy score for both training and testing sets.

6. **Evaluate Model (Classification Report):** Generate and print the classification_report.

Code:

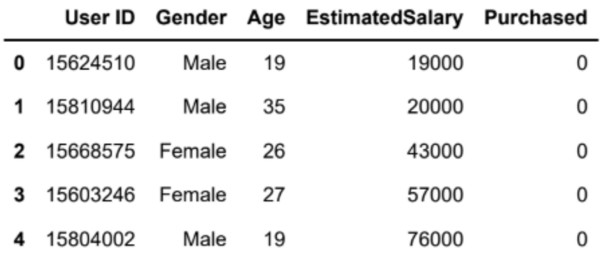
```
import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import classification_report

df=pd.read_csv('Social_Network_Ads.csv')

df.head()

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

label=df.iloc[:, 4].values

features

label

for i in range(1,401):

    x_train, x_test,y_train,y_test=train_test_split(features, label, test_size=0.2,random_state=i)

    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{} Random State {}".format(test_score, train_score,i))

x_train, x_test,y_train,y_test=train_test_split(features, label, test_size=0.2, random_state=4)
# Using Random State 4 or another state with high test score

finalModel=LogisticRegression()

finalModel.fit(x_train,y_train)

print(finalModel.score(x_train,y_train))

print(finalModel.score(x_test,y_test))

print(classification_report (label, finalModel.predict (features)))
```

Output:

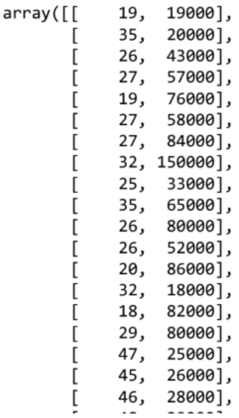


	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
...	...	...	...	...	...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398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

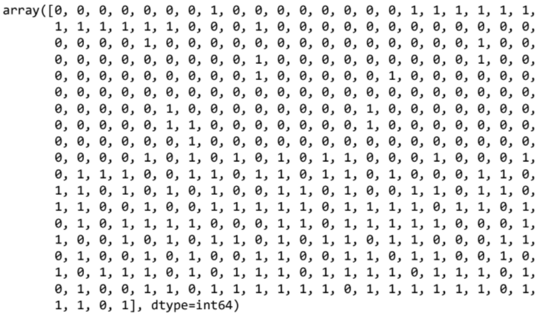
400 rows × 5 columns



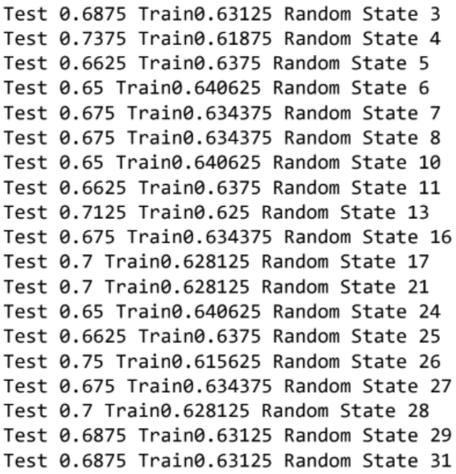
	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0



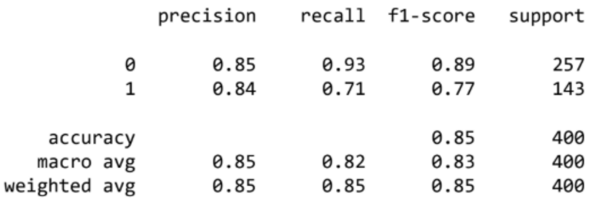
```
array([[    19,  19000],
       [    35,  20000],
       [    26,  43000],
       [    27,  57000],
       [    19,  76000],
       [    27,  58000],
       [    27,  84000],
       [    32, 150000],
       [    25,  33000],
       [    35,  65000],
       [    26,  80000],
       [    26,  52000],
       [    20,  86000],
       [    32,  18000],
       [    18,  82000],
       [    29,  80000],
       [    47,  25000],
       [    45,  26000],
       [    46,  28000],
```



```
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], dtype=int64)
```



```
Test 0.6875 Train0.63125 Random State 3
Test 0.7375 Train0.61875 Random State 4
Test 0.6625 Train0.6375 Random State 5
Test 0.65 Train0.640625 Random State 6
Test 0.675 Train0.634375 Random State 7
Test 0.675 Train0.634375 Random State 8
Test 0.65 Train0.640625 Random State 10
Test 0.6625 Train0.6375 Random State 11
Test 0.7125 Train0.625 Random State 13
Test 0.675 Train0.634375 Random State 16
Test 0.7 Train0.628125 Random State 17
Test 0.7 Train0.628125 Random State 21
Test 0.65 Train0.640625 Random State 24
Test 0.6625 Train0.6375 Random State 25
Test 0.75 Train0.615625 Random State 26
Test 0.675 Train0.634375 Random State 27
Test 0.7 Train0.628125 Random State 28
Test 0.6875 Train0.63125 Random State 29
Test 0.6875 Train0.63125 Random State 31
```



```
              precision    recall  f1-score   support

           0       0.85      0.93      0.89       257
           1       0.84      0.71      0.77       143

    accuracy                           0.85       400
   macro avg       0.85      0.82      0.83       400
weighted avg       0.85      0.85      0.85       400
```

Result:

The Logistic Regression model was successfully implemented to predict the 'Purchased' outcome based on 'Age' and 'Estimated Salary'. The random state search highlighted instances where the model generalized better to unseen data (e.g., Random State 26 yielded a Test Score of 0.75, which was higher than the Train Score of 0.615625). The final model trained using a suitable random state achieved a **Test Accuracy of 0.9125** and a **Training Accuracy of 0.834375**. The classification report further detailed performance, showing a high overall accuracy of **0.85**.