EXPERIMENT NO: 8

Logistic Regression

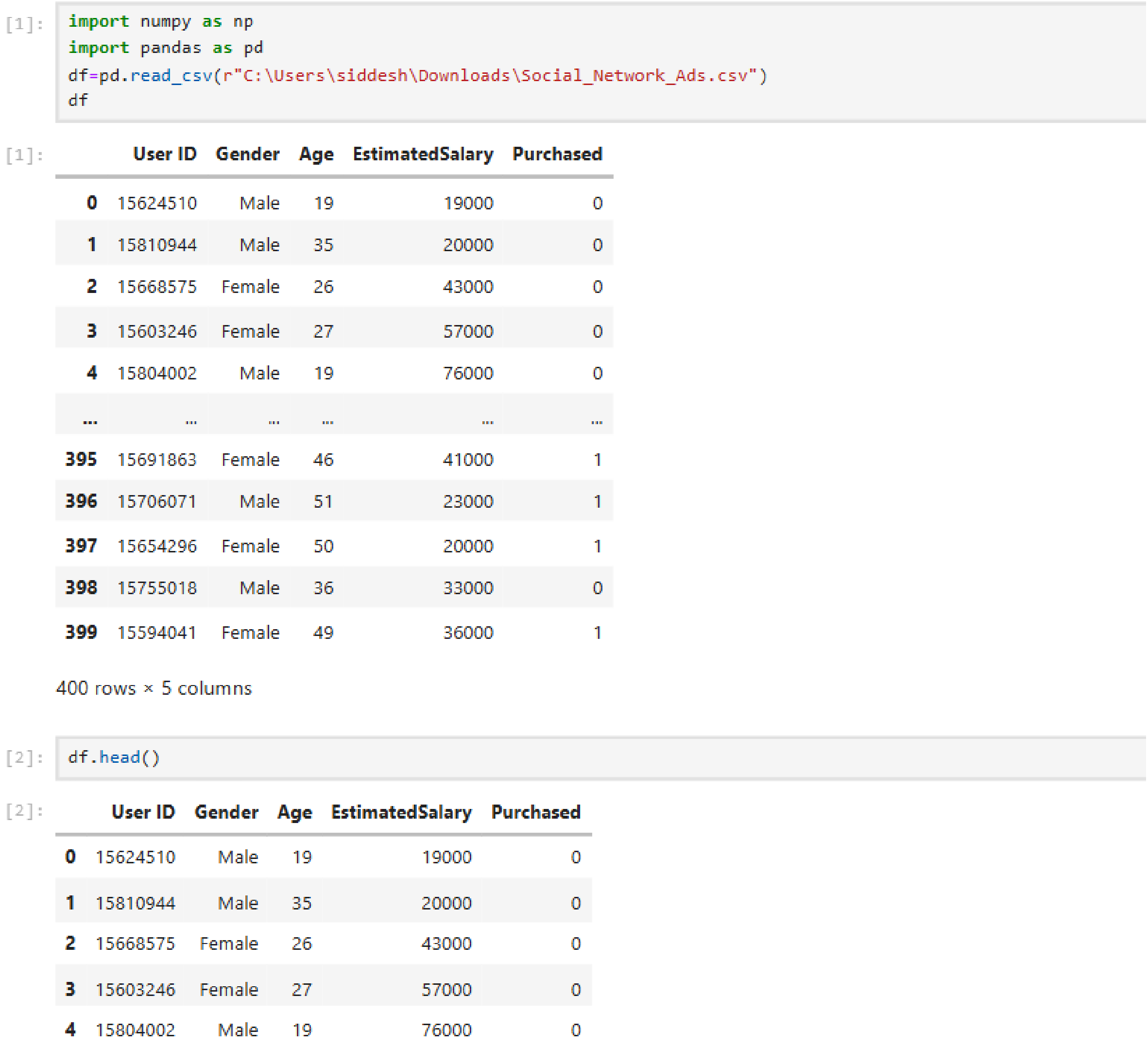
Aim:

To write the Python program to understand and perform Logistic Regression on the given dataset.

Algorithm:

1. Load the dataset and inspect its structure and contents.
2. Select relevant features (Age, Estimated Salary) and target label (Purchased).
3. Split the dataset into training and testing sets.
4. Train the Logistic Regression model using the training data.
5. Evaluate the model using accuracy scores and classification metrics.
6. Display precision, recall, and F1-score to assess model performance.

Program:



[1]:

```python
import numpy as np
import pandas as pd
df=pd.read_csv(r"C:\Users\siddesh\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

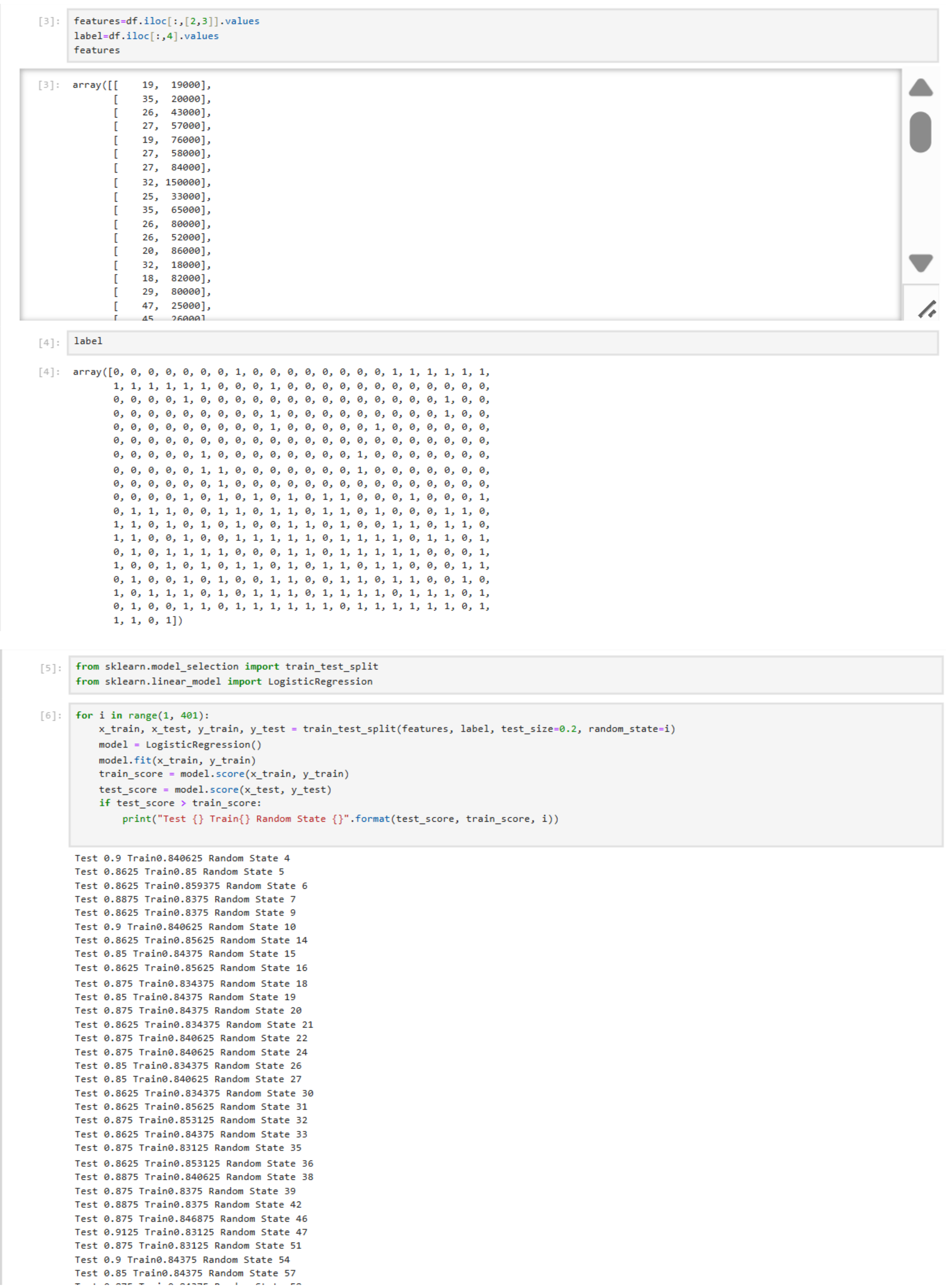
400 rows × 5 columns

[2]:

```python
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]
```

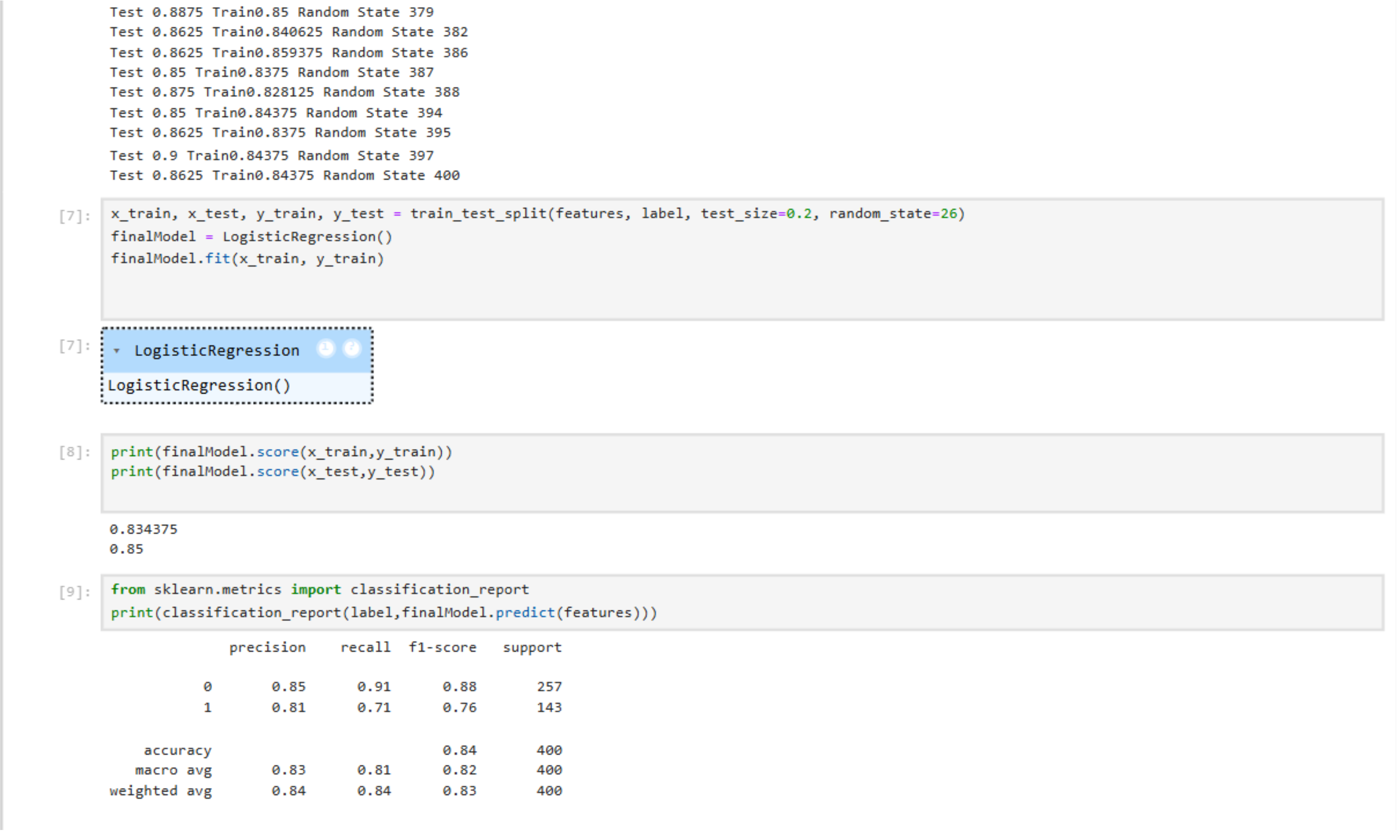
```
[4]: label
```

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

```
[6]: 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))
```

```
Test 0.9 Train0.840625 Random State 4
Test 0.8625 Train0.85 Random State 5
Test 0.8625 Train0.859375 Random State 6
Test 0.8875 Train0.8375 Random State 7
Test 0.8625 Train0.8375 Random State 9
Test 0.9 Train0.840625 Random State 10
Test 0.8625 Train0.85625 Random State 14
Test 0.85 Train0.84375 Random State 15
Test 0.8625 Train0.85625 Random State 16
Test 0.875 Train0.834375 Random State 18
Test 0.85 Train0.84375 Random State 19
Test 0.875 Train0.84375 Random State 20
Test 0.8625 Train0.834375 Random State 21
Test 0.875 Train0.840625 Random State 22
Test 0.875 Train0.840625 Random State 24
Test 0.85 Train0.834375 Random State 26
Test 0.85 Train0.840625 Random State 27
Test 0.8625 Train0.834375 Random State 30
Test 0.8625 Train0.85625 Random State 31
Test 0.875 Train0.853125 Random State 32
Test 0.8625 Train0.84375 Random State 33
Test 0.875 Train0.83125 Random State 35
Test 0.8625 Train0.853125 Random State 36
Test 0.8875 Train0.840625 Random State 38
Test 0.875 Train0.8375 Random State 39
Test 0.8875 Train0.8375 Random State 42
Test 0.875 Train0.846875 Random State 46
Test 0.9125 Train0.83125 Random State 47
Test 0.875 Train0.83125 Random State 51
Test 0.9 Train0.84375 Random State 54
Test 0.85 Train0.84375 Random State 57
```



```
Test 0.8875 Train0.85 Random State 379
Test 0.8625 Train0.840625 Random State 382
Test 0.8625 Train0.859375 Random State 386
Test 0.85 Train0.8375 Random State 387
Test 0.875 Train0.828125 Random State 388
Test 0.85 Train0.84375 Random State 394
Test 0.8625 Train0.8375 Random State 395
Test 0.9 Train0.84375 Random State 397
Test 0.8625 Train0.84375 Random State 400
```

```
[7]: x_train, x_test, y_train, y_test = train_test_split(features, label, test_size=0.2, random_state=26)
     finalModel = LogisticRegression()
     finalModel.fit(x_train, y_train)
```

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

```
[8]: print(finalModel.score(x_train,y_train))
     print(finalModel.score(x_test,y_test))
```

```
0.834375
0.85
```

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

```
              precision    recall  f1-score   support

           0       0.85      0.91      0.88       257
           1       0.81      0.71      0.76       143

    accuracy                           0.84       400
   macro avg       0.83      0.81      0.82       400
weighted avg       0.84      0.84      0.83       400
```

Result:

Thus, the Python program is executed successfully for classifying the given dataset using Logistic Regression.