Logistic_Regression_Assignment12

Step 1 | Data Pre-Processing

Importing the Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set(color_codes=True)
from sklearn.preprocessing import LabelEncoder
```

Importing the dataset

In [2]:
```
dataset = pd.read_csv("Social_Network_Ads.csv")
```

In [3]:
```
dataset.head()
```

Out[3]:

	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

In [4]:
```
dataset.shape
```

Out[4]:
```
(400, 5)
```

In [5]:
```
dataset.columns
```

Out[5]:
```
Index(['User ID', 'Gender', 'Age', 'EstimatedSalary', 'Purchased'], dtype='object')
```

In [6]:
```
dataset.isna().any()
```

Out[6]:
```
User ID            False
Gender             False
Age                False
EstimatedSalary    False
Purchased          False
dtype: bool
```

In [7]:
```
dataset.isna().sum()
```

Out[7]:
```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

In [9]:
```
dataset["Gender"].unique()
```

Out[9]:
```
array(['Male', 'Female'], dtype=object)
```

In [10]:
```
dataset["Gender"].replace({'Male':'0', 'Female':'1'},inplace=True)
```

In [11]:
```
dataset
```

Out[11]:

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

400 rows × 5 columns

Splitting the dataset into the Training set and Test set

In [13]:
```
from sklearn.model_selection import train_test_split
```

In [14]:
```
X_train, X_test, y_train, y_test = train_test_split(dataset[['Gender','Age','EstimatedSalary']],dataset['Purchased'],test_size=0.2)
```

In [15]:
```
len(X_train)
```

Out[15]:
```
320
```

In [16]:
```
len(X_test)
```

Out[16]:
```
80
```

Feature Scaling

In [17]:
```
from sklearn.preprocessing import StandardScaler
```

In [18]:
```
sc_X = StandardScaler()
```

In [19]:
```
X_train = sc_X.fit_transform(X_train)
```

In [20]:
```
X_test = sc_X.transform(X_test)
```

Step 2 | Logistic Regression Model

In [21]:
```
from sklearn.linear_model import LogisticRegression
```

In [22]:
```
lr = LogisticRegression()
```

Fitting Logistic Regression to the Training set

In [23]:
```
lr.fit(X_train,y_train)
```

Out[23]:
```
LogisticRegression()
```

Step 3 | Predection

In [24]:
```
lr.predict(X_test)
```

Out[24]:
```
array([0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0], dtype=int64)
```

In [26]:
```
X_test
```

Out[26]:
```
array([[ 9.75304830e-01, -1.12010746e+00, -7.95427841e-01,
         [-1.02532046e+00,  2.30291199e-01, -3.24934479e-01,
         [-1.02532046e+00, -6.44908129e-01, -1.14829786e+00],
         [ 9.75304830e-01, -2.51994035e-01,  2.26277902e+00],
         [-1.02532046e+00,  7.12576434e-01, -1.11889203e+00],
         [ 9.75304830e-01,  3.26748246e-01,  5.73413786e-02],
         [ 9.75304830e-01, -5.90799412e-02,  1.96872066e+00],
         [ 9.75304830e-01, -5.90799412e-02,  2.92588060e-01],
         [ 9.75304830e-01,  1.58068986e+00,  1.11959144e+00],
         [ 9.75304830e-01, -2.51994035e-01,  2.79355434e-02],
         [-1.02532046e+00, -2.51994035e-01, -4.42557819e-01],
         [-1.02532046e+00,  2.15943214e+00, -8.24833677e-01],
         [-1.02532046e+00, -1.12010746e+00,  4.69023071e-01],
         [-1.02532046e+00,  2.30291199e-01, -3.83746149e-01],
         [-1.02532046e+00, -7.34279269e-01,  2.92588060e-01],
         [-1.02532046e+00,  7.12576434e-01, -1.41295038e+00],
         [-1.02532046e+00,  2.79355434e-02],
         [ 9.75304830e-01,  1.96651804e+00,  7.33675587e-01],
         [-1.02532046e+00, -1.21656450e+00,  2.63182225e-01],
         [ 9.75304830e-01,  1.96651804e+00, -6.77804501e-01],
         [-1.02532046e+00, -1.69884974e+00,  4.69023071e-01],
         [-1.02532046e+00, -9.27193363e-01,  2.63182225e-01],
         [ 9.75304830e-01, -5.41365175e-01,  1.90990899e+00],
         [-1.02532046e+00, -1.40947860e+00, -2.07311138e-01],
         [-1.02532046e+00,  4.23205293e-01, -1.77905303e-01],
         [-1.02532046e+00, -2.51994035e-01, -9.13051182e-01],
         [-1.02532046e+00, -5.41365175e-01, -1.53057372e+00],
         [-1.02532046e+00, -1.55536988e-01,  8.51298928e-01],
         [-1.02532046e+00, -7.34279269e-01, -1.55997356e+00],
         [-1.02532046e+00,  4.23205293e-01, -4.71963654e-01],
         [ 9.75304830e-01, -1.55536988e-01, -2.07311138e-01],
         [ 9.75304830e-01,  1.09840462e+00,  2.08634401e+00],
         [ 9.75304830e-01,  1.00194757e+00,  1.99812650e+00],
         [-1.02532046e+00, -6.37822222e-01, +3.54340314e-01],
         [-1.02532046e+00, -2.51994035e-01, -5.01369490e-01],
         [ 9.75304830e-01,  2.30291199e-01,  1.45558884e-01],
         [ 9.75304830e-01,  1.33834153e-01,  1.45558884e-01],
         [-1.02532046e+00,  4.23205293e-01,  9.98328104e-01],
         [ 9.75304830e-01, -2.51994035e-01, -1.38354454e+00],
         [-1.02532046e+00, -9.27193363e-01,  5.57240576e-01],
         [-1.02532046e+00,  1.33834153e-01,  1.88050316e+00],
         [-1.02532046e+00, -2.51994035e-01,  8.67472137e-02],
         [ 9.75304830e-01,  9.05490527e-01,  1.26298062e+00],
         [ 9.75304830e-01,  3.26748246e-01,  2.92588060e-01],
         [-1.02532046e+00, -1.12010746e+00, -1.17770370e+00],
         [-1.02532046e+00, -6.37822222e-01,  1.16153049e-01],
         [-1.02532046e+00, -5.90799412e-02, -5.30775325e-01],
         [ 9.75304830e-01,  2.06297509e+00, -1.20710953e+00],
         [ 9.75304830e-01,  1.96651804e+00, -9.42457017e-01],
         [ 9.75304830e-01,  1.00194757e+00,  1.43941563e+00],
         [ 9.75304830e-01, -7.34279269e-01,  1.35119813e+00],
         [ 9.75304830e-01, -6.37822222e-01,  5.57240576e-01],
         [-1.02532046e+00, -1.79530678e+00, -1.32473287e+00],
         [ 9.75304830e-01,  9.05490527e-01, -5.89586995e-01],
         [-1.02532046e+00, -1.12010746e+00, -1.58938539e+00],
         [ 9.75304830e-01, -1.44908129e-01, -2.95528643e-01],
         [ 9.75304830e-01,  1.96651804e+00, -1.38354454e+00],
         [-1.02532046e+00, -1.55536988e-01, -2.95528643e-01],
         [ 9.75304830e-01, -2.51994035e-01, -3.83746149e-01],
         [-1.02532046e+00, -5.90799412e-02,  2.63182225e-01],
         [-1.02532046e+00,  2.15943214e+00,  3.80805565e-01],
         [ 9.75304830e-01,  2.06297509e+00,  1.74964719e-01],
         [-1.02532046e+00, -1.55536988e-01, -1.08948619e+00],
         [ 9.75304830e-01, -3.48451082e-01,  5.73413786e-02],
         [ 9.75304830e-01, -1.55536988e-01,  1.45558884e-01],
         [ 9.75304830e-01, -1.02365041e+00, -4.71963654e-01],
         [ 9.75304830e-01, -6.37822222e-01, -1.61879123e+00],
         [ 9.75304830e-01,  4.23205293e-01, -1.47029176e-03],
         [-1.02532046e+00,  9.73771057e-02,  2.79355434e-02],
         [ 9.75304830e-01,  1.67714690e+00,  1.61585064e+00],
         [ 9.75304830e-01,  1.38777576e+00, -1.44235622e+00],
         [ 9.75304830e-01, -1.79530678e+00, -1.29532704e+00],
         [-1.02532046e+00, +8.30736316e-01,  2.29218485e+00],
         [ 9.75304830e-01, -1.89176383e+00, -7.66022006e-01],
         [-1.02532046e+00, -1.50593564e+00, -1.53057372e+00],
         [-1.02532046e+00,  8.09033481e-01,  2.63182225e-01],
         [ 9.75304830e-01,  1.00194757e+00, -1.03067452e+00],
         [ 9.75304830e-01, -1.12010746e+00,  2.92588060e-01],
         [-1.02532046e+00,  1.58068986e+00, -1.47029176e-03],
         [ 9.75304830e-01, +1.69884974e+00,  3.51399730e-01]])
```

Step 4 | Evaluating The Predection

In [27]:
```
lr.score(X_test,y_test)
```

Out[27]:
```
0.9
```

Making the Confusion Matrix

In [31]:
```
# confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

# actual values
actual = [0, 0, 0, 1, 0, 0, 0, 1, 0]
# predicted values
predicted = [0, 0, 0, 1, 0, 0, 1, 0, 1]
# confusion matrix
matrix = confusion_matrix(actual,predicted, labels=[1,0])
print('Confusion matrix : \n',matrix)

# outcome values order in sklearn
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)

# classification report for precision, recall f1-score and accuracy
matrix = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n',matrix)
```

```
Confusion matrix :
 [[1 1]
 [2 5]]
Outcome values :
 1 1 2 5
Classification report :
              precision    recall  f1-score   support

           1       0.33      0.50      0.40         2
           0       0.83      0.71      0.77         7

    accuracy                           0.67         9
   macro avg       0.58      0.61      0.58         9
weighted avg       0.72      0.67      0.69         9
```

Visualization

In [32]:
```
sns.pairplot(dataset[['Gender','Age','EstimatedSalary','Purchased']],hue='Purchased')
```

Out[32]:
```
<seaborn.axisgrid.PairGrid at 0x15a8e347d30>
```