DSBDAL Assignment 09 - Data Analytics 2

1. Implement logistic regression using Python/R to perform classification on Social_Network_Ads.csv dataset.
2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

```
import numpy as np
import pandas as pd
import seaborn as sns
```

```
ds = pd.read_csv('/content/drive/My Drive/DSBDL/Assignment9/data.csv')
ds
```

	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

Next steps: Generate code with ds | View recommended plots

```
ds.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64
 3   EstimatedSalary  400 non-null    int64
 4   Purchased        400 non-null    int64
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

```
ds.describe()
```

	User ID	Age	EstimatedSalary	Purchased
count	4.000000e+02	400.000000	400.000000	400.000000
mean	1.569154e+07	37.655000	69742.500000	0.357500
std	7.165832e+04	10.482877	34096.960282	0.479864
min	1.556669e+07	18.000000	15000.000000	0.000000
25%	1.562676e+07	29.750000	43000.000000	0.000000
50%	1.569434e+07	37.000000	70000.000000	0.000000
75%	1.575036e+07	46.000000	88000.000000	1.000000
max	1.581524e+07	60.000000	150000.000000	1.000000

```
ds.isna().sum()
```

```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

```
from sklearn.preprocessing import LabelEncoder

ds['Gender'] = LabelEncoder().fit_transform(ds['Gender'])
ds
```

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

400 rows × 5 columns

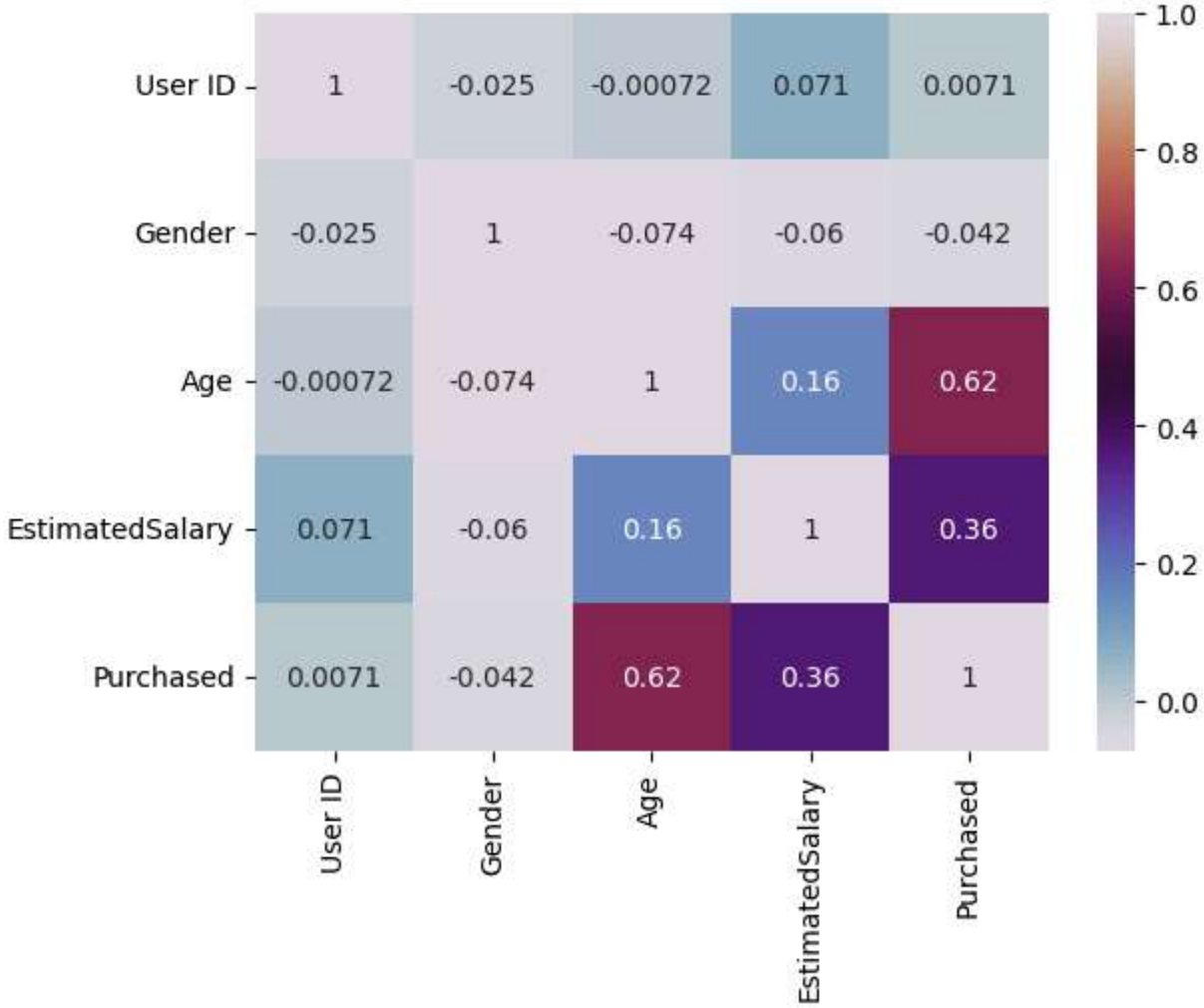
Next steps: Generate code with ds View recommended plots

```
ds.corr()
```

	User ID	Gender	Age	EstimatedSalary	Purchased
User ID	1.000000	-0.025249	-0.000721	0.071097	0.007120
Gender	-0.025249	1.000000	-0.073741	-0.060435	-0.042469
Age	-0.000721	-0.073741	1.000000	0.155238	0.622454
EstimatedSalary	0.071097	-0.060435	0.155238	1.000000	0.362083
Purchased	0.007120	-0.042469	0.622454	0.362083	1.000000

```
sns.heatmap(ds.corr(), cmap = 'twilight', annot = True)
```

```
<Axes: >
```




```
from sklearn.model_selection import train_test_split

ds.drop('User ID', axis = 1, inplace = True)

X = np.asarray(ds.drop('Purchased', axis = 1))
y = np.asarray(ds['Purchased'])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state


from sklearn.preprocessing import StandardScaler

X_train = StandardScaler().fit_transform(X_train)
X_test = StandardScaler().fit_transform(X_test)


from sklearn.linear_model import LogisticRegression

regressor = LogisticRegression()
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
print(y_pred)
```

```
[0 1 0 1 0 0 1 0 0 0 0 1 0 0 0 0 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 0 1 0 1 0 0
 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 1 0 0 1 0 0 0
```

```
 0 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 1 0 0 0 0 0 1 0 1 0]
```

```
from sklearn.metrics import confusion_matrix, classification_report

matrix = confusion_matrix(y_test, y_pred)
print(matrix)

print(classification_report(y_test, y_pred))
```

```
[[61  2]
 [12 25]]
              precision    recall  f1-score   support

           0       0.84      0.97      0.90        63
           1       0.93      0.68      0.78        37

    accuracy                           0.86       100
   macro avg       0.88      0.82      0.84       100
weighted avg       0.87      0.86      0.85       100
```

```
from sklearn.metrics import confusion_matrix, accuracy_score, f1_score, precision_score, re

matrix = confusion_matrix(y_test, y_pred)
tn, fp, fn, tp = matrix.ravel()

print("True Positives: ", tp)
print("True Negatives: ", tn)
print("False Positives: ", fp)
print("False Negatives: ", fn)
print()

print("Accuracy: ", (tp + tn) / (tp + tn + fn + fp))
print("Recall:  ", tp / (tp + fn))
print("Precision: ", tp / (tp + fp))
```

```
True Positives:  25
True Negatives:  61
False Positives:  2
False Negatives:  12

Accuracy:  0.86
Recall:   0.6756756756756757
Precision:  0.9259259259259259
```

```
print("Accuracy: ", accuracy_score(y_test, y_pred))
print("Recall: ", recall_score(y_test, y_pred))
print("Precision: ", precision_score(y_test, y_pred))
```

```
Accuracy:  0.86
Recall:  0.6756756756756757
Precision:  0.9259259259259259
```