```
# Name:Atharva Kangralkar
# SY AIML
# Roll no: 54
# PRN: 12311493
# Colab Link: https://colab.research.google.com/drive/1P_Omhk-BsT6Dhdf0BKC90bleQgeGTxvJ?usp=sharing


# Q1) Import necessary libraries.
import pandas as pd
import numpy as np
import seaborn as sns
import sklearn as sk
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, classification_report
import matplotlib.pyplot as plt


# Q2) Load the dataset.
df = pd.read_csv("/content/Social_Network_Ads.csv")
df
```

	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 df | View recommended plots | New interactive sheet

```
df.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
```

```
df.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

```
df = df.drop(columns=["User ID"])
df
```

	Gender	Age	EstimatedSalary	Purchased
0	Male	19	19000	0
1	Male	35	20000	0
2	Female	26	43000	0
3	Female	27	57000	0
4	Male	19	76000	0
...	...	...	...	...
395	Female	46	41000	1
396	Male	51	23000	1
397	Female	50	20000	1
398	Male	36	33000	0
399	Female	49	36000	1

400 rows × 4 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df = pd.get_dummies(df)
df = df.astype(int)


df
```

	Age	EstimatedSalary	Purchased	Gender_Female	Gender_Male
0	19	19000	0	0	1
1	35	20000	0	0	1
2	26	43000	0	1	0
3	27	57000	0	1	0
4	19	76000	0	0	1
...	...	...	...	...	...
395	46	41000	1	1	0
396	51	23000	1	0	1
397	50	20000	1	1	0
398	36	33000	0	0	1
399	49	36000	1	1	0

400 rows × 5 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# Q3) Select features and target variable.
```

```
X = df.drop(columns=["Purchased"])
# X = df.drop(columns=["Purchased",'Gender_Female','Gender_Male','Age'])
y = df["Purchased"]


# Q4) Splitting the dataset into the Training set and Test set.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)


X_train
```

	Age	EstimatedSalary	Gender_Female	Gender_Male
247	57	122000	1	0
110	39	71000	1	0
16	47	25000	0	1
66	24	19000	0	1
153	36	50000	1	0
...	...	...	...	...
71	24	27000	1	0
106	26	35000	1	0
270	43	133000	1	0
348	39	77000	0	1
102	32	86000	1	0

300 rows × 4 columns

Next steps: Generate code with X_train | View recommended plots | New interactive sheet

```
y_train
```

	Purchased
247	1
110	0
16	1
66	0
153	0
...	...
71	0
106	0
270	0
348	0
102	0

300 rows × 1 columns

dtype: int64

```
model = LogisticRegression()
model.fit(X_train, y_train)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

▾ LogisticRegression

```
LogisticRegression()
```

```
accuracy_score(y_test, model.predict(X_test))
#without scaling
```

```
0.88
```

```
# Q4) Splitting the dataset into the Training set and Test set.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

```
# Q5) Perform Feature Scaling. (Use StandardScaler())
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
# Q6) Fit Logistic Regression to the Training set.
model_2 = LogisticRegression()
model_2.fit(X_train, y_train)
```

```
LogisticRegression()
```

```
accuracy_score(y_test, model_2.predict(X_test))
#after scaling
```

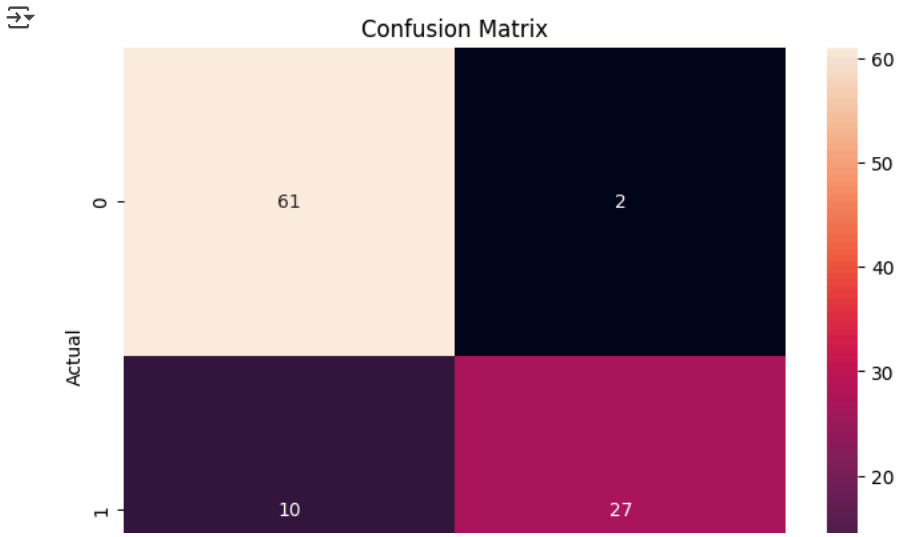
```
0.88
```

```
#   Q7) Predict the Test set result.
y_pred = model_2.predict(X_test)
```

```
# Q8) Evaluating the Model. Find out accuracy, confusion matrix, precision, recall, classification report
cm = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.88
Precision: 0.9310344827586207
Recall: 0.7297297297297297
Classification Report:
               precision    recall  f1-score   support

           0       0.86      0.97      0.91        63
           1       0.93      0.73      0.82        37

    accuracy                           0.88       100
   macro avg       0.90      0.85      0.86       100
weighted avg       0.89      0.88      0.88       100
```

```
# Q9) Visualize Confusion Matrix using Heatmap.
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```




```
# Q10) Plot Scatterplot of the data points.
plt.figure(figsize=(8, 6))
sns.scatterplot(x=df["Age"], y=df["EstimatedSalary"], hue=y)
plt.xlabel("Age")
plt.ylabel("Estimated Salary")
plt.title("Scatter Plot of Data Points")
plt.show()
```