In [144]:

```
import pandas as pd
```

In [145]:

```
#1 .Load the dataset
df=pd.read_csv('Social_Network_Ads.csv')
```

In [146]:

```
df.head(10)
```

Out[146]:

	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
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0

In [147]:

```
#2. PREPROCESS THE DATA

# A.Check for missing values
print(df.isnull().sum())  # If there are missing values, handle them accordingly
```

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

In [148]:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

In [149]:

```
#Encode Gender column
df['Gender'] = df['Gender'].map({'Male': 0, 'Female': 1})
```

In [150]:

```
# Define features and target variable
X_with_gender = df.drop(columns=['Purchased', 'User ID'])  # Keeping Gender
X_without_gender = df.drop(columns=['Purchased', 'User ID', 'Gender'])  # Removing Gender
y = df['Purchased']
```

In [151]:

```
# Standardize numerical features
scaler = StandardScaler()
X_with_gender[['Age', 'EstimatedSalary']] = scaler.fit_transform(X_with_gender[['Age', 'EstimatedSalary']])
X_without_gender[['Age', 'EstimatedSalary']] = scaler.fit_transform(X_without_gender[['Age', 'EstimatedSalary']]
```

In [152]:

```
# Split data into training and testing sets
X_train1, X_test1, y_train, y_test = train_test_split(X_with_gender, y, test_size=0.2, random_state=42)
X_train2, X_test2, _, _ = train_test_split(X_without_gender, y, test_size=0.2, random_state=42)
```

In [153]:

```
# Train Logistic Regression models
model_with_gender = LogisticRegression()
model_without_gender = LogisticRegression()

model_with_gender.fit(X_train1, y_train)
model_without_gender.fit(X_train2, y_train)

# Make predictions
y_pred1 = model_with_gender.predict(X_test1)
y_pred2 = model_without_gender.predict(X_test2)

# Calculate accuracy
accuracy_with_gender = accuracy_score(y_test, y_pred1)
accuracy_without_gender = accuracy_score(y_test, y_pred2)

print(f"Accuracy with Gender: {accuracy_with_gender:.4f}")
print(f"Accuracy without Gender: {accuracy_without_gender:.4f}")
```

```
Accuracy with Gender: 0.8875
Accuracy without Gender: 0.8625
```

In [154]:

```
df.drop(columns=['User ID', 'Gender'], inplace=True)
```

In [155]:

```
# Split features and target variable
X = df.drop(columns=['Purchased'])  # Features
y = df['Purchased']  # Target variable
```

In [156]:

```
# Feature Scaling (Standardization)
from sklearn.preprocessing import StandardScaler
```

In [157]:

```
scaler = StandardScaler()
X[['Age', 'EstimatedSalary']] = scaler.fit_transform(X[['Age', 'EstimatedSalary']])
```

In [158]:

```
# Display the preprocessed data
df.head()
```

Out[158]:

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0

In [159]:

```
#3. SPLIT THE DATA INTO TRAINING AND TESTING SET

from sklearn.model_selection import train_test_split

X = df.drop(columns=['Purchased'])  # Features (excluding the target)
y = df['Purchased']  # Target variable

# Split into 80% training and 20% testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Display the shape of the splits
print(f"Training set: {X_train.shape}, Testing set: {X_test.shape}")
```

```
Training set: (320, 2), Testing set: (80, 2)
```

In [160]:

```
# 5.FEATURE SCALING
from sklearn.preprocessing import StandardScaler

# Initialize the scaler
scaler = StandardScaler()

# Fit and transform only the numerical columns (Age and EstimatedSalary)
X_train[['Age', 'EstimatedSalary']] = scaler.fit_transform(X_train[['Age', 'EstimatedSalary']])
X_test[['Age', 'EstimatedSalary']] = scaler.transform(X_test[['Age', 'EstimatedSalary']])

# Display scaled data
X_train.head()
```

Out[160]:

	Age	EstimatedSalary
3	-1.066752	-0.386344
18	0.797535	-1.229939
202	0.110692	1.853544
250	0.601294	-0.909955
274	1.876859	-1.288118

In [161]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Initialize the model
model = LogisticRegression()

# Train the model
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)

print(f"Model Accuracy: {accuracy:.4f}")
```

```
Model Accuracy: 0.8625
```

In [164]:

```
# Make predictions on the test set
y_pred = model.predict(X_test)

# Display the predicted values
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 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1 0 0 1 0 0 0
 0 0 1 1 0 0]
```

In [162]:

```
#compare actual data and predicted data

# Create a DataFrame to compare actual and predicted values
comparison_df = pd.DataFrame({'Actual': y_test.values, 'Predicted': y_pred})

# Display the first few rows
print(comparison_df.head())
```

```
   Actual  Predicted
0       0          0
1       1          1
2       0          0
3       1          1
4       0          0
```

In [163]:

```
#  compute and display confusion matrix

from sklearn.metrics import confusion_matrix

# Compute confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Display confusion matrix
print("Confusion Matrix:")
print(cm)
```

```
Confusion Matrix:
[[50  2]
 [ 9 19]]
```

In [ ]: