qhbmdohku

March 28, 2025

[3]:
```
import pandas as pd
import numpy as np
import matplotlib as plt
```

[4]:
```
df=pd.read_csv("social_network_ads.csv")
```

[5]:
```
df
```

[5]:

	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 x 5 columns]

[6]:
```
df.columns
```

[6]:
```
Index(['User ID', 'Gender', 'Age', 'EstimatedSalary', 'Purchased'],
dtype='object')
```

[7]:
```
df.isnull()
```

[7]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
..	…	…	…	…	…

```
395    False   False  False            False      False
396    False   False  False            False      False
397    False   False  False            False      False
398    False   False  False            False      False
399    False   False  False            False      False

[400 rows x 5 columns]
```

[8]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Gender'] = le.fit_transform(df['Gender'])
newdf=df
```

[9]:
```
df
```

[9]:
```
      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 x 5 columns]
```

[29]:
```
# Splitting the DataFrame into features (X) and target variable (y)
x = df.drop(['Purchased'], axis=1)  # X: Features (all columns except␣
 ↪'Purchased')
y = df['Purchased']                 # y: Target variable (Purchased status)
```

[30]:
```
# Importing the train_test_split function to split the dataset into training␣
 ↪and testing sets
from sklearn.model_selection import train_test_split
```

[31]:
```
# Splitting the dataset into training and testing sets
# X_train, Y_train → Training data (60%)
# X_test, Y_test → Testing data (40%)
X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.4,␣
 ↪random_state=10)
```

[32]:
```
# Importing the LogisticRegression model from sklearn for binary classification
from sklearn.linear_model import LogisticRegression
```

[34]:
```
# Displaying the first 5 rows of the training feature set
print(X_train.head())
```

```
   Gender
0    Male
4    Male
1  Female
```

[36]:
```
# One-hot encoding for categorical variables
# drop_first=True → Avoids multicollinearity by dropping the first category
X_train = pd.get_dummies(X_train, drop_first=True)
X_test = pd.get_dummies(X_test, drop_first=True)

# Initializing the Logistic Regression model
logreg = LogisticRegression()

# Fitting the model to the training data
logreg.fit(X_train, Y_train)
```

[36]:
```
LogisticRegression()
```

[37]:
```
# Making predictions on the testing set using the trained logistic regression␣
 ↪model
Y_pred = logreg.predict(X_test)  # Predicting the target values for X_test

# Printing the predicted values
print("Predictions:", Y_pred)
```

```
Predictions: ['No' 'No']
```

[38]:
```
# Importing the sklearn library and LogisticRegression model
import sklearn
from sklearn.linear_model import LogisticRegression  # Logistic regression model

# Initializing the Logistic Regression model
logreg = LogisticRegression()

# Fitting the model to the training data
model = logreg.fit(X_train, Y_train)  # model contains the trained logistic␣
 ↪regression instance
```

[39]:
```
# Making predictions on the training set
Ytrain_pred = logreg.predict(X_train)  # Predicted labels for the training set

# Making predictions on the testing set
Ytest_pred = logreg.predict(X_test)    # Predicted labels for the testing set
```

[40]:
```
import pandas as pd

# Creating DataFrame for training predictions
df_train = pd.DataFrame({'Actual': Y_train, 'Predicted': Ytrain_pred})

# Creating DataFrame for testing predictions
df_test = pd.DataFrame({'Actual': Y_test, 'Predicted': Ytest_pred})

# Displaying the first few rows
print("Training Predictions:")
print(df_train.head())

print("\nTesting Predictions:")
print(df_test.head())
```

```
Training Predictions:
  Actual Predicted
0    Yes        No
4     No        No
1     No        No

Testing Predictions:
  Actual Predicted
2    Yes        No
3    Yes        No
```

[41]:
```
# Importing evaluation metrics from sklearn
from sklearn.metrics import precision_score, confusion_matrix, accuracy_score,␣
 ↪recall_score

# Creating the confusion matrix for the testing set
cm = confusion_matrix(Y_test, Y_pred)  # Compares actual vs predicted labels
```

[42]:
```
# Creating the confusion matrix for the testing set
cm = confusion_matrix(Y_test, Y_pred)  # Compares actual vs predicted labels
```

[43]:
```
# Generating the confusion matrix for the testing set
cm = confusion_matrix(Y_test, Y_pred)  # Compares actual vs predicted labels

# Printing the confusion matrix
print("Confusion Matrix:\n", cm)
```

```
Confusion Matrix:
 [[0 0]
 [2 0]]
```

[45]:
```
# Printing the model's performance metrics
print("Accuracy:", accuracy_score(Y_test, Y_pred))                    # Overall␣
 ↪accuracy of the model
print("Precision:", precision_score(Y_test, Y_pred, average='weighted'))  #␣
 ↪Precision considering class imbalance
print("Recall:", recall_score(Y_test, Y_pred, average='weighted'))        #␣
 ↪Recall considering class imbalance
```

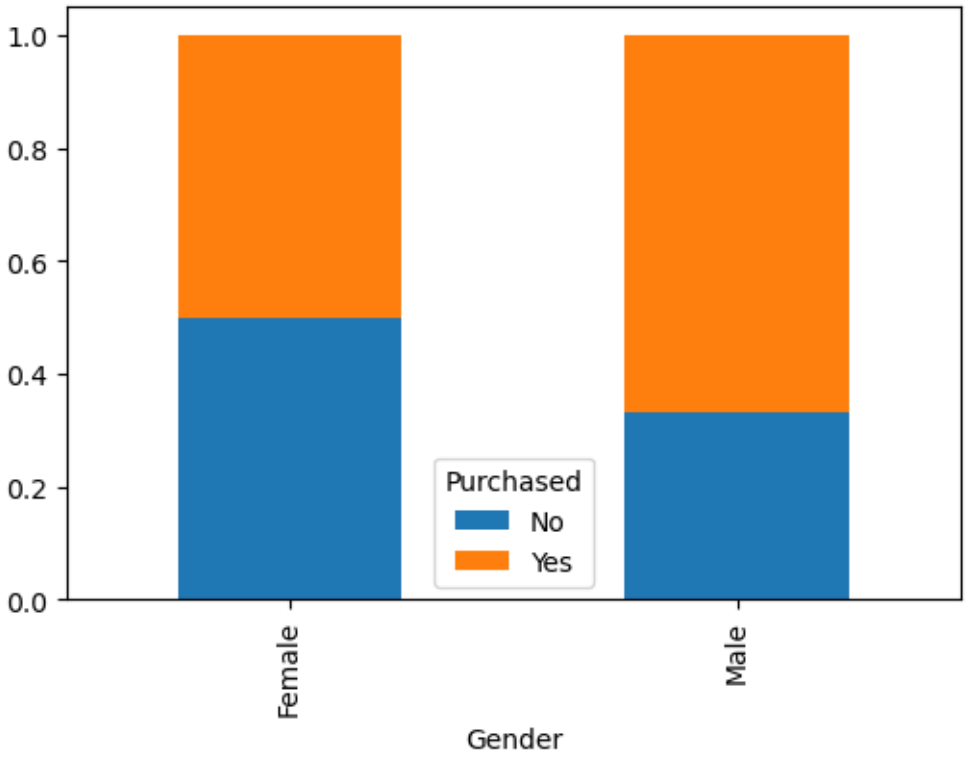
```
Accuracy: 0.0
Precision: 0.0
Recall: 0.0
```

```
C:\Users\Vishwajeet Kulkarni\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Vishwajeet Kulkarni\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 in labels with no true samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[48]:
```
# Importing necessary libraries
import pandas as pd                    # For creating and handling DataFrames
import matplotlib.pyplot as plt        # For plotting visualizations

# Sample DataFrame (replace with your actual data)
data = {
    'Gender': ['Male', 'Female', 'Male', 'Female', 'Male'],    # Gender column
    'Purchased': ['Yes', 'No', 'Yes', 'Yes', 'No']             # Purchased status
}

# Creating a DataFrame from the sample data
df = pd.DataFrame(data)

# Creating a crosstab to show the proportion of purchases by gender
# normalize="index" → Normalizes the values by row (percentage distribution)
cm = pd.crosstab(df['Gender'], df['Purchased'], normalize="index")
print(cm)  # Displaying the normalized crosstab

# Plotting the crosstab as a stacked bar chart
cm.plot.bar(figsize=(6, 4), stacked=True)  # Stacked bar chart with dimensions␣
 ↪6x4
plt.show()  # Displaying the plot
```

```
Purchased        No       Yes
Gender
```

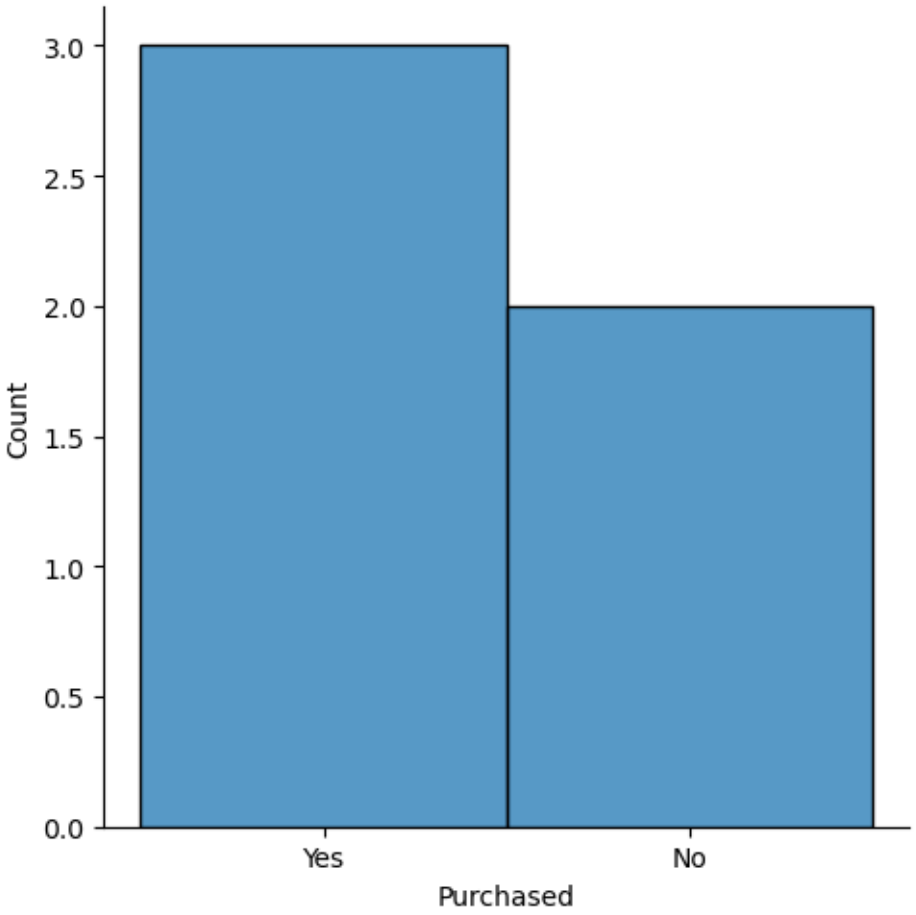
```
Female      0.500000  0.500000
Male        0.333333  0.666667
```



[46]:
```
# Importing Seaborn for data visualization
import seaborn as sns

# Plotting the distribution of the 'Purchased' column
sns.displot(df['Purchased'])  # Distribution plot of the target variable
```

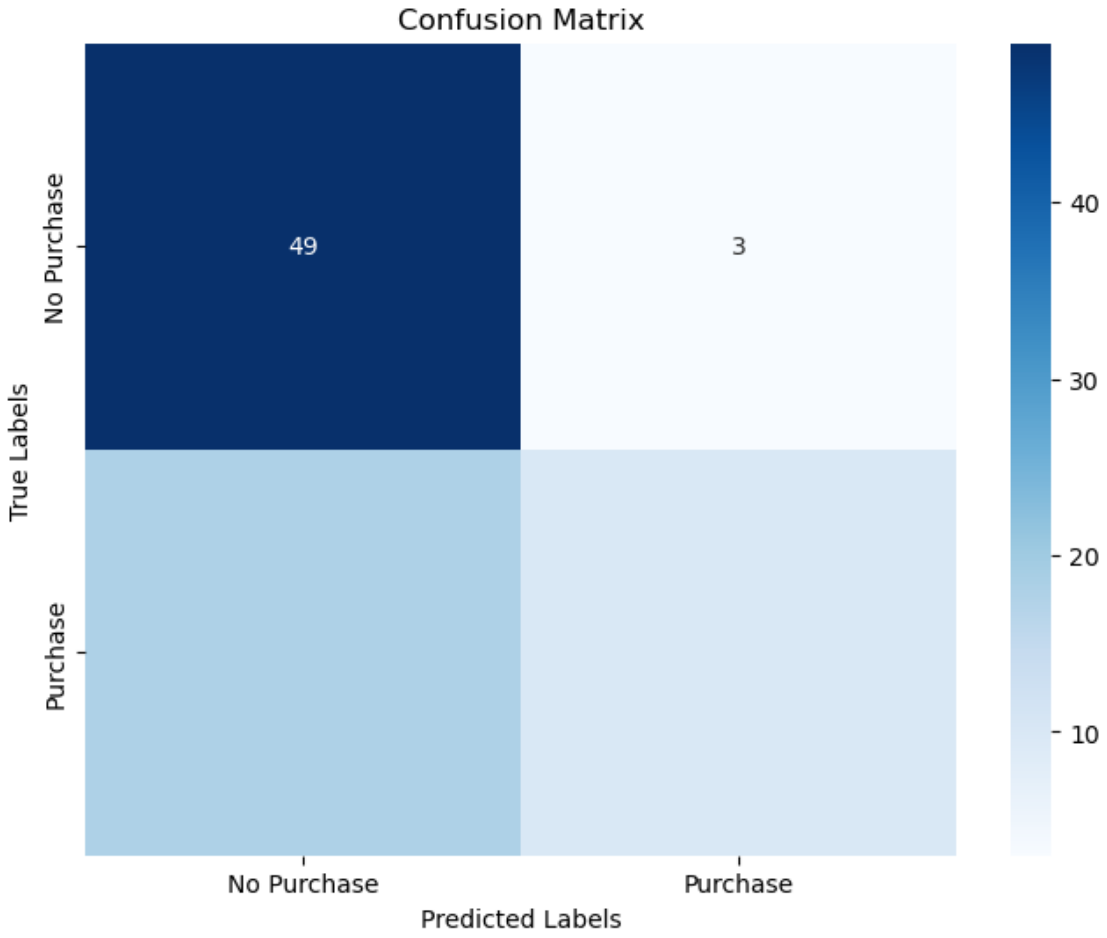
```
C:\Users\Vishwajeet Kulkarni\anaconda3\Lib\site-
packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is
deprecated and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

[46]: `<seaborn.axisgrid.FacetGrid at 0x18038f5e390>`



[47]:
```
# Importing necessary libraries for visualization and evaluation
import matplotlib.pyplot as plt      # For creating visual plots
import seaborn as sns                # For enhanced visualizations
from sklearn.metrics import confusion_matrix  # For generating the confusion␣
 ↪matrix
```

[42]:
```
# Generate confusion matrix
cm = confusion_matrix(Y_test, Y_pred)

# Plot confusion matrix with correct labels
labels = ['No Purchase', 'Purchase']  # Your class labels
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=labels,␣
 ↪yticklabels=labels)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```


Confusion Matrix
No Purchase
Purchase
True Labels
49
3
40
30
20
10
No Purchase
Purchase
Predicted Labels

[ ]: