practical-05

April 26, 2024

[30]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

[9]:
```
data = pd.read_csv("C:/Users/gugal/Desktop/THIRD 2/PRACTICALS/DS/CODES/DATASETS/
↪Social_Network_ads.csv")
data
```

[9]:
```
      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]
```

[10]:
```
data.isnull().sum()
```

[10]:
```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

[11]:
```python
data.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
```

[5]:
```python
data.columns
```

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

[6]:
```python
data.describe()
```

[6]:

	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

[17]:
```python
le = LabelEncoder()
data['Gender'] = le.fit_transform(data['Gender'])
data['Gender'].unique()
```

[17]:
```
array([1, 0], dtype=int64)
```

[18]:
```python
data
```

[18]:

	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]
```

[19]:
```python
X = data[['User ID', 'Gender', 'Age', 'EstimatedSalary']].values
X
```

[19]:
```
array([[15624510,        1,       19,    19000],
       [15810944,        1,       35,    20000],
       [15668575,        0,       26,    43000],
       …,
       [15654296,        0,       50,    20000],
       [15755018,        1,       36,    33000],
       [15594041,        0,       49,    36000]], dtype=int64)
```

[20]:
```python
Y = data['Purchased'].values
Y
```

[20]:
```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0,
       1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0,
       1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1,
       1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0,
       1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 0, 1], dtype=int64)
```

[22]:
```python
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25,␣
 ↪random_state=0)
```

[23]:
```python
print(len(Y_test))
```

100

[25]:
```python
model = LogisticRegression()
model.fit(X_train,Y_train)
```

[25]: LogisticRegression()

[46]:
```python
prediction = model.predict(X_test)
print('Accuracy is', metrics.accuracy_score(Y_test,prediction))
```

Accuracy is 0.89

[28]:
```python
prediction
```

[28]:
```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1], dtype=int64)
```

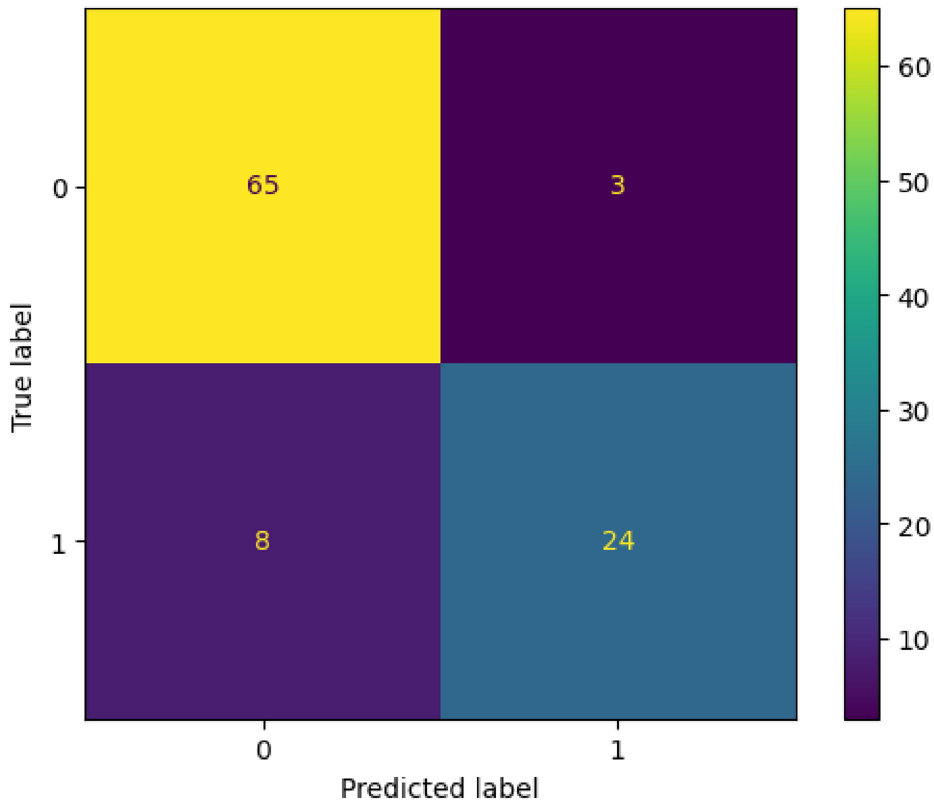
[29]:
```python
print(len(prediction))
```

100

[31]:
```python
CM = confusion_matrix(Y_test,prediction)
CM
```

[31]:
```
array([[65,  3],
       [ 8, 24]], dtype=int64)
```

[34]:
```python
disp = ConfusionMatrixDisplay(confusion_matrix=CM)
disp.plot()
plt.show()
```



[35]:
```
TN = CM[0, 0]
FP = CM[0, 1]
FN = CM[1, 0]
TP = CM[1, 1]

print("True Negative (TN):", TN)
print("False Positive (FP):", FP)
print("False Negative (FN):", FN)
print("True Positive (TP):", TP)
```

```
True Negative (TN): 65
False Positive (FP): 3
False Negative (FN): 8
True Positive (TP): 24
```

[37]:
```
acc= (TP + TN)/(TP+FP+TN+FN)
acc
```

[37]: 0.89

[38]:
```python
# Error Rate
Error_Rate = (FP + FN)/(TP+FP+TN+FN)
Error_Rate
```

[38]: 0.11

[48]:
```python
# Precision
Precision = (TP)/(TP+FP)
Precision
```

[48]: 0.8888888888888888

[49]:
```python
# Recall
Recall = (TP)/(TP+FN)
Recall
```

[49]: 0.75