practical-05

February 16, 2025

[14]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
```

[15]:
```
data = pd.read_csv("Social_Network_Ads.csv")
```

[16]:
```
data.head()
```

[16]:

	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

[17]:
```
data.tail()
```

[17]:

	User ID	Gender	Age	EstimatedSalary	Purchased
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

[18]:
```
# Separate the features (X) and the target variable (y)
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values
```

[19]:
```
print(X)
```

```
[[15624510 'Male' 19 19000]
 [15810944 'Male' 35 20000]
 [15668575 'Female' 26 43000]
 …
 [15654296 'Female' 50 20000]
```

```
 [15755018 'Male' 36 33000]
 [15594041 'Female' 49 36000]]
```

[20]:
```python
print(y)
```

```
[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]
```

[21]:
```python
# Perform label encoding on the 'Gender' column

"""
In machine learning projects, we usually deal with datasets having different␣
↪categorical columns where some columns have their elements in the ordinal␣
↪variable category for e.g a column income level having elements as low,␣
↪medium, or high in this case we can replace these elements with 1,2,3. where␣
↪1 represents ‘low’  2  ‘medium’  and 3  high’. Through this type of encoding,␣
↪we try to preserve the meaning of the element where higher weights are␣
↪assigned to the elements having higher priority.

Label Encoding :
Label Encoding is a technique that is used to convert categorical columns into␣
↪numerical ones so that they can be fitted by machine learning models which␣
↪only take numerical data. It is an important pre-processing step in a␣
↪machine-learning project.
"""

le = LabelEncoder()
X[:, 1] = le.fit_transform(X[:, 1])
```

[22]:
```python
# Split the dataset into training and testing sets

"""
The train_test_split function of the sklearn.model_selection package in Python␣
↪splits arrays or matrices into random subsets for train and test data,␣
↪respectively.
"""
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

[23]:
```
# Create an instance of the Logistic Regression model
logistic_regression = LogisticRegression()
```

[24]:
```
# Train the model on the training data
logistic_regression.fit(X_train, y_train)
```

[24]: LogisticRegression()

[25]:
```
# Predict the labels for the test set
y_pred = logistic_regression.predict(X_test)
```

[26]:
```
# Compute the confusion matrix

"""
A confusion matrix is a matrix that summarizes the performance of a machine learning model on a set of test data. It is often used to measure the performance of classification models, which aim to predict a categorical label for each input instance.
"""

confusion = confusion_matrix(y_test, y_pred)
```

[27]:
```
# Extract the values from the confusion matrix

"""
True Positive (TP): It is the total counts having both predicted and actual values are Dog.
True Negative (TN): It is the total counts having both predicted and actual values are Not Dog.
False Positive (FP): It is the total counts having prediction is Dog while actually Not Dog.
False Negative (FN): It is the total counts having prediction is Not Dog while actually, it is Dog.
"""

TN = confusion[0, 0]  # True Negative
FP = confusion[0, 1]  # False Positive
FN = confusion[1, 0]  # False Negative
TP = confusion[1, 1]  # True Positive
```

[28]:
```
# Compute the accuracy
accuracy = (TP + TN) / (TP + TN + FP + FN)
```

```python
# Compute the error rate
error_rate = (FP + FN) / (TP + TN + FP + FN)

# Compute the precision
precision = TP / (TP + FP)

# Compute the recall
recall = TP / (TP + FN)
```

[29]:
```python
# display the confusion matrix
print(confusion)
```

```
[[49  3]
 [18 10]]
```

[30]:
```python
# display the accuracy
print(accuracy)
```

```
0.7375
```

[31]:
```python
# display the error rate
print(error_rate)
```

```
0.2625
```

[32]:
```python
# display the precision
print(precision)
```

```
0.7692307692307693
```

[33]:
```python
# display the recall
print(recall)
```

```
0.35714285714285715
```