DSBDA practical5

May 12, 2023

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
df= pd.read_csv("D:\College Practicals\DSBDApractical5\Social_Network_Ads.csv")
```

[3]:
```
df
```

[3]:

	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]

[4]:
```
df.isnull()
```

[4]:

	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]

[5]:
```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
df['Gender'] = le.fit_transform(df['Gender'])
```

[6]:
```
df
```

[6]:

	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]

[7]:
```
# Select the columns you want to include in the covariance matrix
columns = ['Age', 'EstimatedSalary', 'Purchased']

# Create a new DataFrame that contains only the selected columns
df_selected = df[columns]

# Build the covariance matrix
covariance_matrix = df_selected.cov()

# Print the covariance matrix
print(covariance_matrix)
```

```
                      Age  EstimatedSalary    Purchased
Age            109.890702     5.548738e+04     3.131165
EstimatedSalary  55487.380952     1.162603e+09  5924.367168
Purchased          3.131165     5.924367e+03     0.230269
```

[8]:
```
# Select the independent variables
X = df.drop('Purchased', axis=1)

# Select the dependent variable
Y = df['Purchased']
```

[9]:
```
from sklearn.model_selection import train_test_split
```

[10]:
```
# Select the independent and dependent variables
X = df.drop('Purchased', axis=1)
Y = df['Purchased']
```

[22]:
```
# Split the dataset into a training set and a testing set
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3,␣
 ↪random_state=42)
```

[23]:
```
#import the class
from sklearn.linear_model import LogisticRegression
```

[24]:
```
#instantiate the model

logreg = LogisticRegression()
```

[25]:
```
logreg.fit(X_train,Y_train)
```

[25]:
```
LogisticRegression()
```

[30]:
```
y_pred=logreg.predict(X_test)
```

[32]:
```
print("Predicted values for the testing data: ", Y_pred)
```

```
Predicted values for the testing data:  [0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0
0 1 0 0 0 0 1 0 0 0 1 1 1 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0 1 0 1 0 0 0 0 0 0
 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 1 1 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 1 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 1 0 1 1 0 0
 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0
 0 1 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 0 0 0 1 0 1 0 1 0 0 0 0]
```

[27]:
```
# Predict the Y values for the training data
Y_pred = logreg.predict(X_train)
```

[29]:
```
print("Predicted values for the training data: ", Y_pred)
```

```
Predicted values for the training data:  [0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 1 0
0 0 1 0 0 0 0 1 0 0 0 1 1 1 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0 1 0 1 0 0 0 0 0 0
 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 1 1 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 1 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 1 0 1 1 0 0
 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0
 0 1 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 0 0 0 1 0 1 0 1 0 0 0 0]
```

[35]:
```
from sklearn.metrics import accuracy_score

# Generate predicted values for the training data
Y_pred = logreg.predict(X_train)
```

[36]:
```
# Calculate the accuracy of the model
train_accuracy = accuracy_score(Y_train, Y_pred)
```

[37]:
```
# Print the accuracy
print("Accuracy on training data: {:.2f}%".format(train_accuracy * 100))
```

```
Accuracy on training data: 77.50%
```

[40]:
```
from sklearn.metrics import precision_score,confusion_matrix,recall_score

# Calculate the precision score of the model
train_precision = precision_score(Y_train, Y_pred)
```

[41]:
```
print("Precision on training data: {:.2f}%".format(train_precision * 100))
```

```
Precision on training data: 84.75%
```

[43]:
```
# Calculate the confusion matrix of the model
train_cm = confusion_matrix(Y_train, Y_pred)
```

[44]:
```
print("Confusion matrix on training data:\n", train_cm)
```

```
Confusion matrix on training data:
 [[167   9]
 [ 54  50]]
```

[46]:
```
# Calculate the recall score of the model
train_recall = recall_score(Y_train, Y_pred)
```

[47]:
```
print("Recall on training data: {:.2f}%".format(train_recall * 100))
```

```
Recall on training data: 48.08%
```

[ ]:
```

```