untitled5

May 7, 2024

[37]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import classification_report
```

[14]:
```
df = pd.read_csv('Social_Network_Ads.csv')
df
```

[14]:
```
      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]
```

[15]:
```
df.isnull().sum()
```

[15]:
```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

[20]:
```
df.drop(['User ID'],axis = 1, inplace = True)
```

[21]:
```
df
```

[21]:
```
     Gender  Age  EstimatedSalary  Purchased
0         1   19            19000          0
1         1   35            20000          0
2         0   26            43000          0
3         0   27            57000          0
4         1   19            76000          0
..      …   …               …        …
395       0   46            41000          1
396       1   51            23000          1
397       0   50            20000          1
398       1   36            33000          0
399       0   49            36000          1

[400 rows x 4 columns]
```

[22]:
```
df['Gender'].replace(['Male','Female'],[1,0],inplace=True)
```

```
C:\Users\PUSHKAR\AppData\Local\Temp\ipykernel_7376\411176285.py:1:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['Gender'].replace(['Male','Female'],[1,0],inplace=True)
```

[23]:
```
df
```

[23]:
```
     Gender  Age  EstimatedSalary  Purchased
0         1   19            19000          0
1         1   35            20000          0
2         0   26            43000          0
3         0   27            57000          0
4         1   19            76000          0
..      …   …               …        …
395       0   46            41000          1
396       1   51            23000          1
397       0   50            20000          1
```

```
398        1    36             33000          0
399        0    49             36000          1

[400 rows x 4 columns]
```

[10]:
```
df['Age'].max()
```

[10]:
```
60
```

[25]:
```
x = df[['Gender','Age','EstimatedSalary']]
y = df[['Purchased']]
```

[27]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.
↪80,random_state = 42)
```

[35]:
```
model = LogisticRegression()
model.fit(x_train,y_train)
```

```
C:\Users\PUSHKAR\AppData\Local\Programs\Python\Python312\Lib\site-
packages\sklearn\utils\validation.py:1300: DataConversionWarning: A column-
vector y was passed when a 1d array was expected. Please change the shape of y
to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

[35]:
```
LogisticRegression()
```

[38]:
```
pred = model.predict(x_test)
pred
```

[38]:
```
array([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, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0,
       1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 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, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0,
       1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1], dtype=int64)
```

[39]:
```
accuracy_score(y_test,pred)
```

[39]: 0.828125

[40]:
```
cm = confusion_matrix(y_test,pred)
cm
```

[40]:
```
array([[191,  11],
       [ 44,  74]], dtype=int64)
```

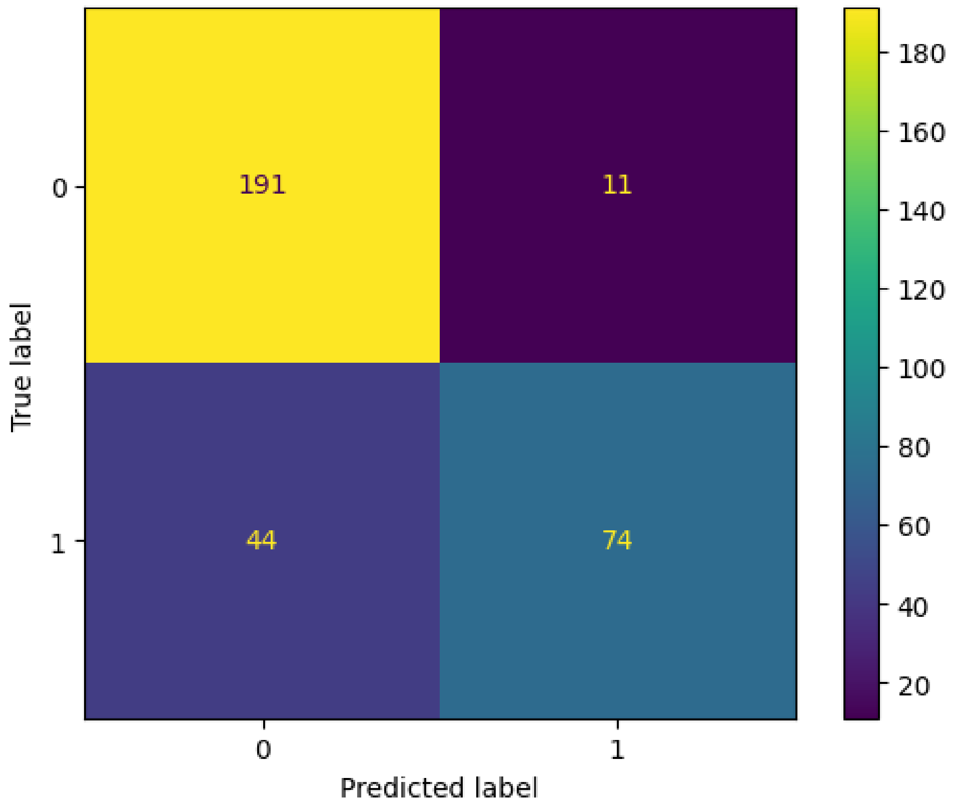
[41]:
```
TP = cm[0][0]
FN = cm[0][1]
TN = cm[1][0]
FP = cm[1][1]
```

[42]:
```
TP,FN,TN,FP
```

[42]:
```
(191, 11, 44, 74)
```

[44]:
```
disp = ConfusionMatrixDisplay(confusion_matrix = cm)
disp.plot()
```

[44]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1cbecb5c680>
```



[45]:
```
print(classification_report(y_test,pred))
```

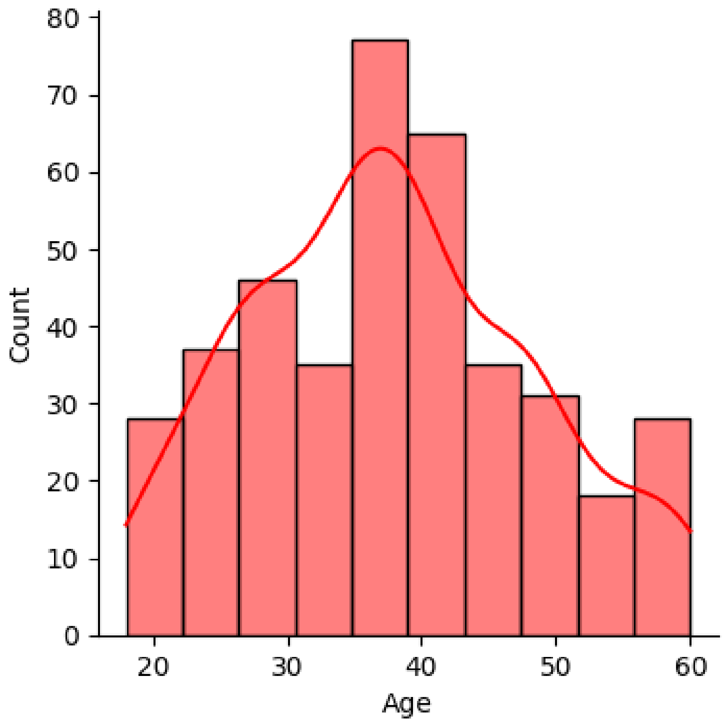
```
              precision    recall  f1-score   support

           0       0.81      0.95      0.87       202
           1       0.87      0.63      0.73       118

    accuracy                           0.83       320
   macro avg       0.84      0.79      0.80       320
weighted avg       0.83      0.83      0.82       320
```

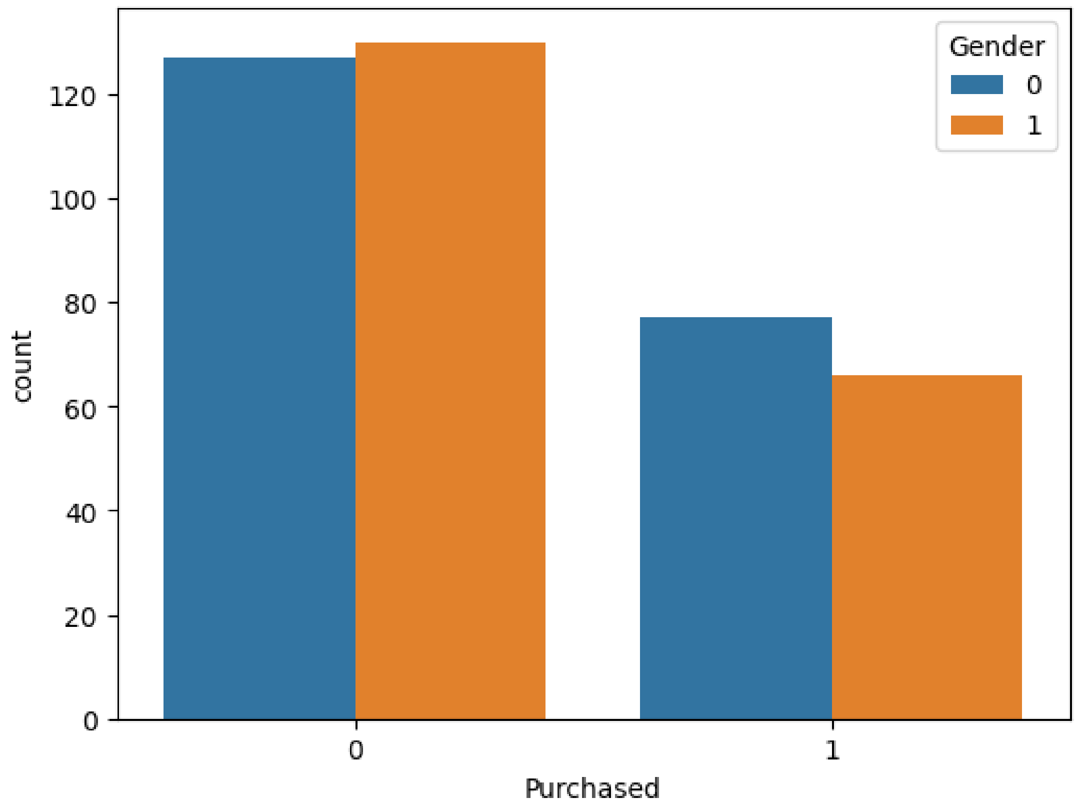
[50]:
```
sns.displot(x='Age',color='red',data=df,height=4,kde=True)
```

[50]:
```
<seaborn.axisgrid.FacetGrid at 0x1cbefc39c40>
```



[51]:
```
sns.countplot(x='Purchased',hue='Gender',data=df)
```

[51]:
```
<Axes: xlabel='Purchased', ylabel='count'>
```

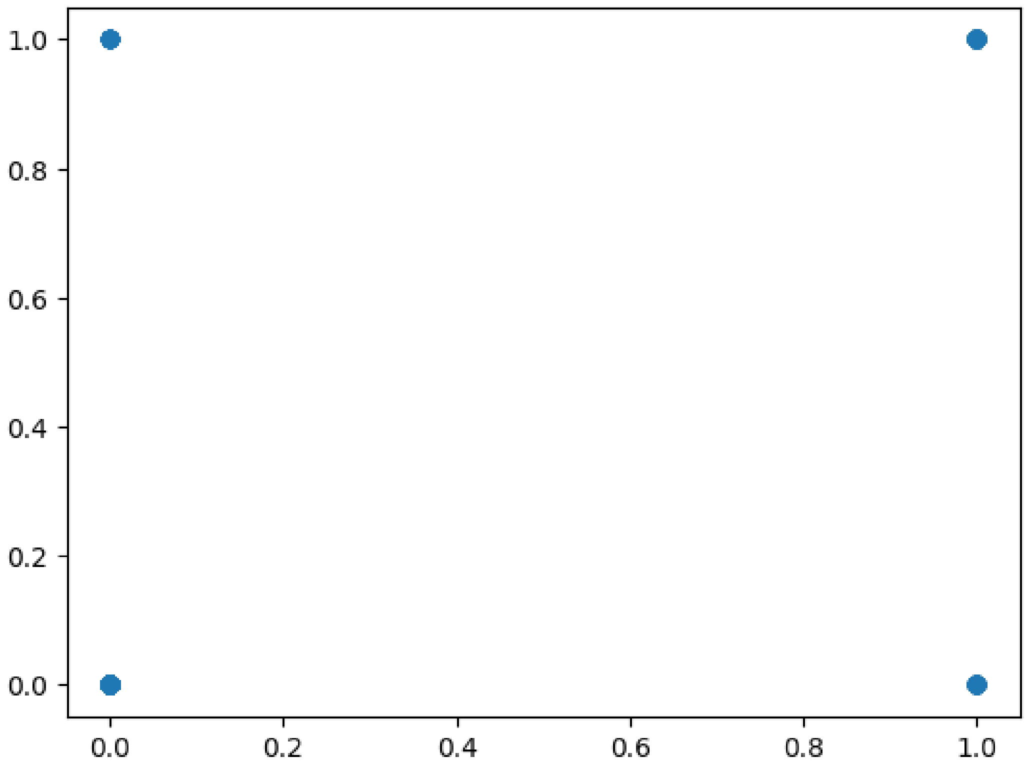


[53]:
```
from sklearn import metrics
print('MSE', metrics.mean_squared_error(y_test,pred))
```

```
MSE 0.171875
```

[61]:
```
pred2 = model.predict(x_test)
plt.scatter(y_test,pred2)
```

[61]:
```
<matplotlib.collections.PathCollection at 0x1cbf43bc170>
```


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

```
[ ]:
```