Lecture 15

June 2, 2023

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

[2]:
```
sns.get_dataset_names()
```

[2]:
```
['anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'dowjones',
 'exercise',
 'flights',
 'fmri',
 'geyser',
 'glue',
 'healthexp',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'seaice',
 'taxis',
 'tips',
 'titanic']
```

[3]:
```
df=sns.load_dataset('tips')
```

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

[4]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3

```
3       23.68  3.31    Male     No  Sun  Dinner     2
4       24.59  3.61  Female     No  Sun  Dinner     4
```

[5]:
```
df.isnull().sum()
```

[5]:
```
total_bill    0
tip           0
sex           0
smoker        0
day           0
time          0
size          0
dtype: int64
```

[6]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

[7]:
```
df['sex']=le.fit_transform(df['sex'])
df['smoker']=le.fit_transform(df['smoker'])
df['day']=le.fit_transform(df['day'])
df['time']=le.fit_transform(df['time'])
```

[8]:
```
df.head()
```

[8]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	0	0	2	0	2
1	10.34	1.66	1	0	2	0	3
2	21.01	3.50	1	0	2	0	3
3	23.68	3.31	1	0	2	0	2
4	24.59	3.61	0	0	2	0	4

1 Splitting dataset into X and Y

[9]:
```
x=df.drop('tip',axis=1)
y=df['tip']
```

[10]:
```
x
```

[10]:

	total_bill	sex	smoker	day	time	size
0	16.99	0	0	2	0	2
1	10.34	1	0	2	0	3
2	21.01	1	0	2	0	3
3	23.68	1	0	2	0	2
4	24.59	0	0	2	0	4
..	…	…	…	…	…	…
239	29.03	1	0	1	0	3
240	27.18	0	1	1	0	2

```
241       22.67    1       1    1     0     2
242       17.82    1       0    1     0     2
243       18.78    0       0    3     0     2

[244 rows x 6 columns]
```

[11]:
```
y
```

[11]:
```
0      1.01
1      1.66
2      3.50
3      3.31
4      3.61
       …
239    5.92
240    2.00
241    2.00
242    1.75
243    3.00
Name: tip, Length: 244, dtype: float64
```

2 Applying Scaling on X

[12]:
```
from sklearn.preprocessing import MinMaxScaler
```

[13]:
```
scale=MinMaxScaler()
```

[14]:
```
x=scale.fit_transform(x)
```

[15]:
```
x
```

[15]:
```
array([[0.29157939, 0.        , 0.        , 0.66666667, 0.        ,
        0.2       ],
       [0.1522832 , 1.        , 0.        , 0.66666667, 0.        ,
        0.4       ],
       [0.3757855 , 1.        , 0.        , 0.66666667, 0.        ,
        0.4       ],
       …,
       [0.41055718, 1.        , 1.        , 0.33333333, 0.        ,
        0.2       ],
       [0.30896523, 1.        , 0.        , 0.33333333, 0.        ,
        0.2       ],
       [0.32907415, 0.        , 0.        , 1.        , 0.        ,
        0.2       ]])
```

3 Splitting X and Y in train and Test

[16]:
```
from sklearn.model_selection import train_test_split
```

[17]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
```

4 Model Building

[18]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
```

[19]:
```
lr.fit(x_train,y_train)
```

[19]:
```
LinearRegression()
```

[20]:
```
y_pred=lr.predict(x_test)
```

5 Evaluation metrics

[21]:
```
from sklearn.metrics import mean_squared_error,r2_score, mean_absolute_error
```

[22]:
```
error=y_test-y_pred
```

[23]:
```
error
```

[23]:
```
64    -0.268407
63     0.694602
55     0.762951
111   -0.476576
225   -0.238789
         …
90    -0.668911
101    0.591909
75    -0.783550
4     -0.036714
109    1.789722
Name: tip, Length: 74, dtype: float64
```

[24]:
```
se=error*error
```

[25]:
```
se
```

[25]:
```
64     0.072042
63     0.482472
55     0.582094
111    0.227124
```

```
225    0.057020
         …
90     0.447442
101    0.350356
75     0.613951
4      0.001348
109    3.203105
Name: tip, Length: 74, dtype: float64
```

[26]:
```
mse=np.mean(se)
```

[27]:
```
mse
```

[27]:
```
0.9166571859645408
```

[28]:
```
mse2=mean_squared_error(y_test,y_pred)
```

[29]:
```
mse2
```

[29]:
```
0.9166571859645408
```

[30]:
```
mae=mean_absolute_error(y_test,y_pred)
```

[31]:
```
mae
```

[31]:
```
0.7166503728723608
```

[32]:
```
rmse=np.sqrt(mse2)
```

[33]:
```
rmse
```

[33]:
```
0.9574221566083275
```

[34]:
```
r2=r2_score(y_test,y_pred)
```

[35]:
```
r2
```

[35]:
```
0.4687997604863058
```

6 Classification Metrics

[37]:
```
data=sns.load_dataset('titanic')
```

[38]:
```
data
```

[38]:
```
     survived  pclass     sex   age  sibsp  parch     fare embarked   class  \
0           0       3    male  22.0      1      0   7.2500        S   Third
1           1       1  female  38.0      1      0  71.2833        C   First
```

```
2           1       3  female  26.0      0      0   7.9250        S   Third
3           1       1  female  35.0      1      0  53.1000        S   First
4           0       3    male  35.0      0      0   8.0500        S   Third
..        ...     ...     ...   ...    ...    ...      ...      ...     ...
886         0       2    male  27.0      0      0  13.0000        S  Second
887         1       1  female  19.0      0      0  30.0000        S   First
888         0       3  female   NaN      1      2  23.4500        S   Third
889         1       1    male  26.0      0      0  30.0000        C   First
890         0       3    male  32.0      0      0   7.7500        Q   Third

       who  adult_male deck  embark_town alive  alone
0      man        True  NaN  Southampton    no  False
1    woman       False    C    Cherbourg   yes  False
2    woman       False  NaN  Southampton   yes   True
3    woman       False    C  Southampton   yes  False
4      man        True  NaN  Southampton    no   True
..     ...         ...  ...          ...   ...    ...
886    man        True  NaN  Southampton    no   True
887  woman       False    B  Southampton   yes   True
888  woman       False  NaN  Southampton    no  False
889    man        True    C    Cherbourg   yes   True
890    man        True  NaN   Queenstown    no   True

[891 rows x 15 columns]
```

[39]:
```
data.isnull().sum()
```

[39]:
```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

[40]:
```
data['age'].fillna(data['age'].median(),inplace=True)
```

[41]:
```
data['deck'].mode()
```

[41]:
```
0    C
Name: deck, dtype: category
Categories (7, object): ['A', 'B', 'C', 'D', 'E', 'F', 'G']
```

[42]:
```
data['deck'].fillna(data['deck'].mode()[0],inplace=True)
```

[43]:
```
data.isnull().sum()
```

[43]:
```
survived       0
pclass         0
sex            0
age            0
sibsp          0
parch          0
fare           0
embarked       2
class          0
who            0
adult_male     0
deck           0
embark_town    2
alive          0
alone          0
dtype: int64
```

[44]:
```
data.dropna(how='any',inplace=True)
```

[45]:
```
data.shape
```

[45]:
```
(889, 15)
```

[46]:
```
data.isnull().sum()
```

[46]:
```
survived       0
pclass         0
sex            0
age            0
sibsp          0
parch          0
fare           0
embarked       0
class          0
who            0
adult_male     0
deck           0
embark_town    0
alive          0
alone          0
```

```
dtype: int64
```

[47]:
```
data.head()
```

[47]:
```
   survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
0         0       3    male  22.0      1      0   7.2500        S  Third
1         1       1  female  38.0      1      0  71.2833        C  First
2         1       3  female  26.0      0      0   7.9250        S  Third
3         1       1  female  35.0      1      0  53.1000        S  First
4         0       3    male  35.0      0      0   8.0500        S  Third

     who  adult_male deck  embark_town alive  alone
0    man        True    C  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False    C  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True    C  Southampton    no   True
```

[48]:
```
data.drop('embark_town',axis=1,inplace=True)
```

[49]:
```
data['sex']=le.fit_transform(data['sex'])
data['embarked']=le.fit_transform(data['embarked'])
data['class']=le.fit_transform(data['class'])
data['who']=le.fit_transform(data['who'])
data['adult_male']=le.fit_transform(data['adult_male'])
data['deck']=le.fit_transform(data['deck'])
data['alive']=le.fit_transform(data['alive'])
data['alone']=le.fit_transform(data['alone'])
```

[50]:
```
data.head()
```

[50]:
```
   survived  pclass  sex   age  sibsp  parch     fare  embarked  class  who  \
0         0       3    1  22.0      1      0   7.2500         2      2    1
1         1       1    0  38.0      1      0  71.2833         0      0    2
2         1       3    0  26.0      0      0   7.9250         2      2    2
3         1       1    0  35.0      1      0  53.1000         2      0    2
4         0       3    1  35.0      0      0   8.0500         2      2    1

   adult_male  deck  alive  alone
0           1     2      0      0
1           0     2      1      0
2           0     2      1      1
3           0     2      1      0
4           1     2      0      1
```

[51]:
```
x=data.drop('survived',axis=1)
y=data['survived']
```

[52]:
```
x
```

[52]:
```
     pclass  sex   age  sibsp  parch     fare  embarked  class  who  \
0         3    1  22.0      1      0   7.2500         2      2    1
1         1    0  38.0      1      0  71.2833         0      0    2
2         3    0  26.0      0      0   7.9250         2      2    2
3         1    0  35.0      1      0  53.1000         2      0    2
4         3    1  35.0      0      0   8.0500         2      2    1
..      ...  ...   ...    ...    ...      ...       ...    ...  ...
886       2    1  27.0      0      0  13.0000         2      1    1
887       1    0  19.0      0      0  30.0000         2      0    2
888       3    0  28.0      1      2  23.4500         2      2    2
889       1    1  26.0      0      0  30.0000         0      0    1
890       3    1  32.0      0      0   7.7500         1      2    1

     adult_male  deck  alive  alone
0             1     2      0      0
1             0     2      1      0
2             0     2      1      1
3             0     2      1      0
4             1     2      0      1
..          ...   ...    ...    ...
886           1     2      0      1
887           0     1      1      1
888           0     2      0      0
889           1     2      1      1
890           1     2      0      1

[889 rows x 13 columns]
```

[53]:
```
y
```

[53]:
```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: survived, Length: 889, dtype: int64
```

[54]:
```
x=scale.fit_transform(x)
```

[55]:
```
x
```

[55]:
```
array([[1.        , 1.        , 0.27117366, …, 0.33333333, 0.        ,
        0.        ],
       [0.        , 0.        , 0.4722292 , …, 0.33333333, 1.        ,
        0.        ],
       [1.        , 0.        , 0.32143755, …, 0.33333333, 1.        ,
        1.        ],
       …,
       [1.        , 0.        , 0.34656949, …, 0.33333333, 0.        ,
        0.        ],
       [0.        , 1.        , 0.32143755, …, 0.33333333, 1.        ,
        1.        ],
       [1.        , 1.        , 0.39683338, …, 0.33333333, 0.        ,
        1.        ]])
```

[56]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
```

[57]:
```
from sklearn.linear_model import LogisticRegression
lor=LogisticRegression()
```

[58]:
```
lor.fit(x_train,y_train)
```

[58]:
```
LogisticRegression()
```

[59]:
```
y_pred=lor.predict(x_test)
```

[60]:
```
from sklearn.metrics import␣
↪accuracy_score,confusion_matrix,roc_auc_score,precision_recall_fscore_support,f1_score,roc_
```

[61]:
```
accuracy_score(y_test,y_pred)
```

[61]:
```
1.0
```

[62]:
```
confusion_matrix(y_test,y_pred)
```

[62]:
```
array([[157,   0],
       [  0, 110]])
```

[63]:
```
pr=precision_recall_fscore_support(y_test,y_pred, average='micro')
```

[64]:
```
pr
```

[64]:
```
(1.0, 1.0, 1.0, None)
```

[65]:
```
f1_score(y_test,y_pred)
```

[65]:
```
1.0
```

[66]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       157
           1       1.00      1.00      1.00       110

    accuracy                           1.00       267
   macro avg       1.00      1.00      1.00       267
weighted avg       1.00      1.00      1.00       267
```

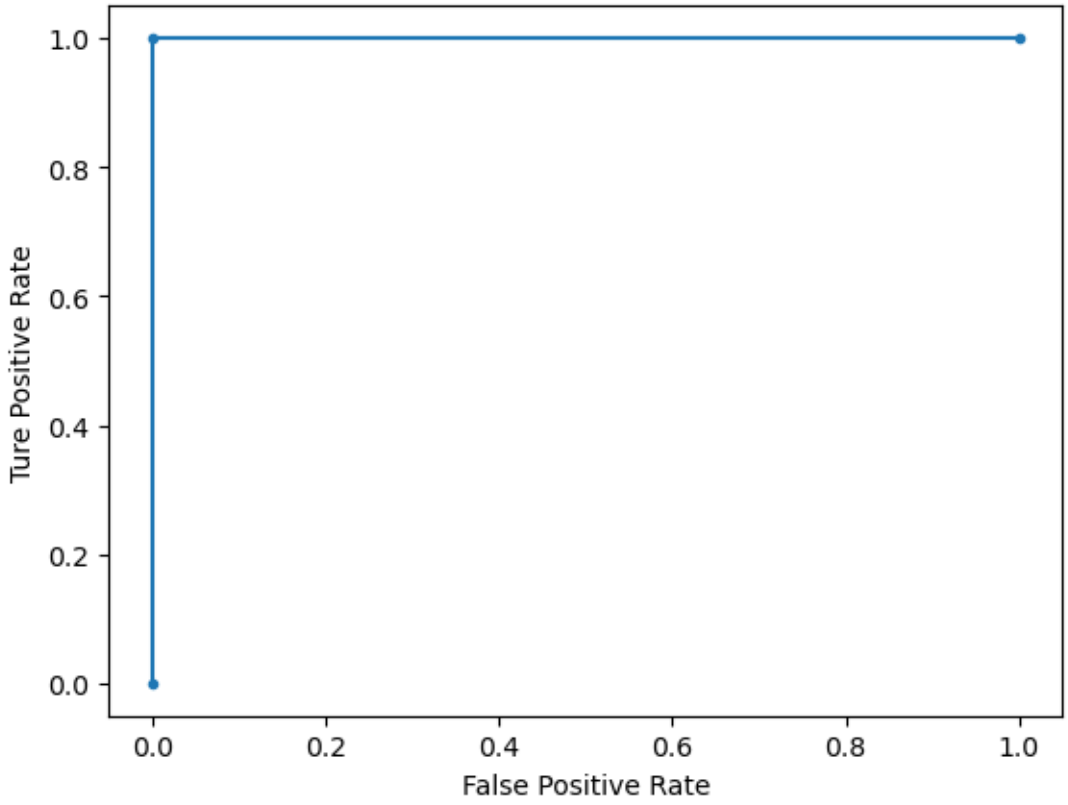
[67]:
```
roc=roc_auc_score(y_test,y_pred)
```

[68]:
```
roc
```

[68]:
```
1.0
```

[69]:
```
fpr,tpr,threshold=roc_curve(y_test,y_pred)
plt.plot(fpr,tpr,marker='.')
plt.xlabel('False Positive Rate')
plt.ylabel('Ture Positive Rate')
```

[69]:
```
Text(0, 0.5, 'Ture Positive Rate')
```



[ ]: