heart-assignment

May 9, 2023

[27]:
```
import pandas as pd
```

[28]:
```
df=pd.read_csv('heart (2).csv')
```

[29]:
```
df
```

[29]:
```
      age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0      52    1   0       125   212    0        1      168      0      1.0
1      53    1   0       140   203    1        0      155      1      3.1
2      70    1   0       145   174    0        1      125      1      2.6
3      61    1   0       148   203    0        1      161      0      0.0
4      62    0   0       138   294    1        1      106      0      1.9
...   ...  ...  ..       ...   ...  ...      ...      ...    ...      ...
1020   59    1   1       140   221    0        1      164      1      0.0
1021   60    1   0       125   258    0        0      141      1      2.8
1022   47    1   0       110   275    0        0      118      1      1.0
1023   50    0   0       110   254    0        0      159      0      0.0
1024   54    1   0       120   188    0        1      113      0      1.4

      slope  ca  thal  target
0         2   2     3       0
1         0   0     3       0
2         0   0     3       0
3         2   1     3       0
4         1   3     2       0
...     ...  ..   ...     ...
1020      2   0     2       1
1021      1   1     3       0
1022      1   1     2       0
1023      2   0     2       1
1024      1   1     3       0

[1025 rows x 14 columns]
```

[30]:
```
df.isnull().sum()
```

[30]:
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

[31]:
```
df.dtypes
```

[31]:
```
age           int64
sex           int64
cp            int64
trestbps      int64
chol          int64
fbs           int64
restecg       int64
thalach       int64
exang         int64
oldpeak     float64
slope         int64
ca            int64
thal          int64
target        int64
dtype: object
```

[32]:
```
df['oldpeak'] = df['oldpeak'].astype(int)
```

[33]:
```
df
```

[33]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	\
0	52	1	0	125	212	0	1	168	0	1	
1	53	1	0	140	203	1	0	155	1	3	
2	70	1	0	145	174	0	1	125	1	2	
3	61	1	0	148	203	0	1	161	0	0	
4	62	0	0	138	294	1	1	106	0	1	
…	…	…	..	…	…	…	…	…	…	…	
1020	59	1	1	140	221	0	1	164	1	0	
1021	60	1	0	125	258	0	0	141	1	2	

```
1022   47    1   0       110   275    0        0      118      1        1
1023   50    0   0       110   254    0        0      159      0        0
1024   54    1   0       120   188    0        1      113      0        1

      slope  ca  thal  target
0         2   2     3       0
1         0   0     3       0
2         0   0     3       0
3         2   1     3       0
4         1   3     2       0
…     …   ..    …     …
1020      2   0     2       1
1021      1   1     3       0
1022      1   1     2       0
1023      2   0     2       1
1024      1   1     3       0

[1025 rows x 14 columns]
```

[34]:
```
df.dtypes
```

[34]:
```
age         int64
sex         int64
cp          int64
trestbps    int64
chol        int64
fbs         int64
restecg     int64
thalach     int64
exang       int64
oldpeak     int32
slope       int64
ca          int64
thal        int64
target      int64
dtype: object
```

[35]:
```
df.tail()
```

[35]:
```
      age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
1020   59    1   1       140   221    0        1      164      1        0
1021   60    1   0       125   258    0        0      141      1        2
1022   47    1   0       110   275    0        0      118      1        1
1023   50    0   0       110   254    0        0      159      0        0
1024   54    1   0       120   188    0        1      113      0        1

      slope  ca  thal  target
```

```
1020      2   0     2       1
1021      1   1     3       0
1022      1   1     2       0
1023      2   0     2       1
1024      1   1     3       0
```

[46]:
```
countFemale = len(df[df.sex == 0])
countMale = len(df[df.sex == 1])
print("Percentage of Female Patients: {:.2f}%".format((countFemale / (len(df.
 ↪sex))*100)))
print("Percentage of Male Patients: {:.2f}%".format((countMale / (len(df.
 ↪sex))*100)))
```

```
Percentage of Female Patients: 30.44%
Percentage of Male Patients: 69.56%
```

[47]:
```
countNoDisease = len(df[df.target == 0])
countHaveDisease = len(df[df.target == 1])
print("Percentage of Patients Haven't Heart Disease: {:.2f}%".
 ↪format((countNoDisease / (len(df.target))*100)))
print("Percentage of Patients Have Heart Disease: {:.2f}%".
 ↪format((countHaveDisease / (len(df.target))*100)))
```

```
Percentage of Patients Haven't Heart Disease: 48.68%
Percentage of Patients Have Heart Disease: 51.32%
```

[36]:
```
df.groupby('target').mean()
```

[36]:

target	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	56.569138	0.827655	0.482966	134.106212	251.292585	0.164329	0.456914	139.130261	0.549098	1.274549	1.166333	1.158317	2.539078
1	52.408745	0.570342	1.378327	129.245247	240.979087	0.134981	0.598859	158.585551	0.134981	0.342205	1.593156	0.370722	2.119772

[44]:
```
summary_df = df['age'].agg(['mean', 'std', 'min', 'max'])
```

[45]:
```
summary_df
```

[45]:
```
mean    54.434146
std      9.072290
min     29.000000
max     77.000000
Name: age, dtype: float64
```

[37]:
```
a = pd.get_dummies(df['cp'], prefix = "cp")
b = pd.get_dummies(df['thal'], prefix = "thal")
c = pd.get_dummies(df['slope'], prefix = "slope")
```

[38]:
```
frames = [df, a, b, c]
df = pd.concat(frames, axis = 1)
df.head()
```

[38]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	…	\
0	52	1	0	125	212	0	1	168	0	1	…	
1	53	1	0	140	203	1	0	155	1	3	…	
2	70	1	0	145	174	0	1	125	1	2	…	
3	61	1	0	148	203	0	1	161	0	0	…	
4	62	0	0	138	294	1	1	106	0	1	…	

	cp_1	cp_2	cp_3	thal_0	thal_1	thal_2	thal_3	slope_0	slope_1	slope_2
0	0	0	0	0	0	0	1	0	0	1
1	0	0	0	0	0	0	1	1	0	0
2	0	0	0	0	0	0	1	1	0	0
3	0	0	0	0	0	0	1	0	0	1
4	0	0	0	0	0	1	0	0	1	0

[5 rows x 25 columns]

[39]:
```
df = df.drop(columns = ['cp', 'thal', 'slope'])
df.head()
```

[39]:

	age	sex	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	ca	…	\
0	52	1	125	212	0	1	168	0	1	2	…	
1	53	1	140	203	1	0	155	1	3	0	…	
2	70	1	145	174	0	1	125	1	2	0	…	
3	61	1	148	203	0	1	161	0	0	1	…	
4	62	0	138	294	1	1	106	0	1	3	…	

	cp_1	cp_2	cp_3	thal_0	thal_1	thal_2	thal_3	slope_0	slope_1	slope_2
0	0	0	0	0	0	0	1	0	0	1
1	0	0	0	0	0	0	1	1	0	0
2	0	0	0	0	0	0	1	1	0	0
3	0	0	0	0	0	0	1	0	0	1
4	0	0	0	0	0	1	0	0	1	0

[5 rows x 22 columns]

[40]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

y = df.target.values
x_data = df.drop(['target'], axis = 1)
```

[41]:
```
x = (x_data - np.min(x_data)) / (np.max(x_data) - np.min(x_data)).values
```

```
c:\python 39\lib\site-packages\numpy\core\fromnumeric.py:84: FutureWarning: In a
future version, DataFrame.min(axis=None) will return a scalar min over the
entire DataFrame. To retain the old behavior, use 'frame.min(axis=0)' or just
'frame.min()'
  return reduction(axis=axis, out=out, **passkwargs)
c:\python 39\lib\site-packages\numpy\core\fromnumeric.py:84: FutureWarning: In a
future version, DataFrame.max(axis=None) will return a scalar max over the
entire DataFrame. To retain the old behavior, use 'frame.max(axis=0)' or just
'frame.max()'
  return reduction(axis=axis, out=out, **passkwargs)
```

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

[43]:
```
clf = LogisticRegression()

# Train the model on the training set
clf.fit(x_train, y_train)

# Use the trained model to make predictions on the testing set
y_pred = clf.predict(x_test)

# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8634146341463415
```

[ ]:
```
```