cmpe442-assigment-2-classification

April 29, 2024

importing libraries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

importing my data set which i choose heart diseases data set

[2]:
```
data=pd.read_csv('heartState_document.csv')
data.head()
```

[2]:

	age	sex	chest pain type	resting bp s	cholesterol	fasting blood sugar	resting ecg	max heart rate	exercise angina	oldpeak	ST slope	target
0	40	1	2	140	289	0	0	172	0	0.0	1	0
1	49	0	3	160	180	0	0	156	0	1.0	2	1
2	37	1	2	130	283	0	1	98	0	0.0	1	0
3	48	0	4	138	214	0	0	108	1	1.5	2	1
4	54	1	3	150	195	0	0	122	0	0.0	1	0

[3]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1190 entries, 0 to 1189
Data columns (total 12 columns):
```

```
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   age                  1190 non-null   int64
 1   sex                  1190 non-null   int64
 2   chest pain type      1190 non-null   int64
 3   resting bp s         1190 non-null   int64
 4   cholesterol          1190 non-null   int64
 5   fasting blood sugar  1190 non-null   int64
 6   resting ecg          1190 non-null   int64
 7   max heart rate       1190 non-null   int64
 8   exercise angina      1190 non-null   int64
 9   oldpeak              1190 non-null   float64
 10  ST slope             1190 non-null   int64
 11  target               1190 non-null   int64
dtypes: float64(1), int64(11)
memory usage: 111.7 KB
```

[4]:
```
data.dropna(inplace=True)
data
```

[4]:

	age	sex	chest pain type	resting bp s	cholesterol	\
0	40	1	2	140	289	
1	49	0	3	160	180	
2	37	1	2	130	283	
3	48	0	4	138	214	
4	54	1	3	150	195	
…	…	…	…	…	…	
1185	45	1	1	110	264	
1186	68	1	4	144	193	
1187	57	1	4	130	131	
1188	57	0	2	130	236	
1189	38	1	3	138	175	

	fasting blood sugar	resting ecg	max heart rate	exercise angina	\
0	0	0	172	0	
1	0	0	156	0	
2	0	1	98	0	
3	0	0	108	1	
4	0	0	122	0	
…	…	…	…	…	
1185	0	0	132	0	
1186	1	0	141	0	
1187	0	0	115	1	
1188	0	2	174	0	
1189	0	0	173	0	

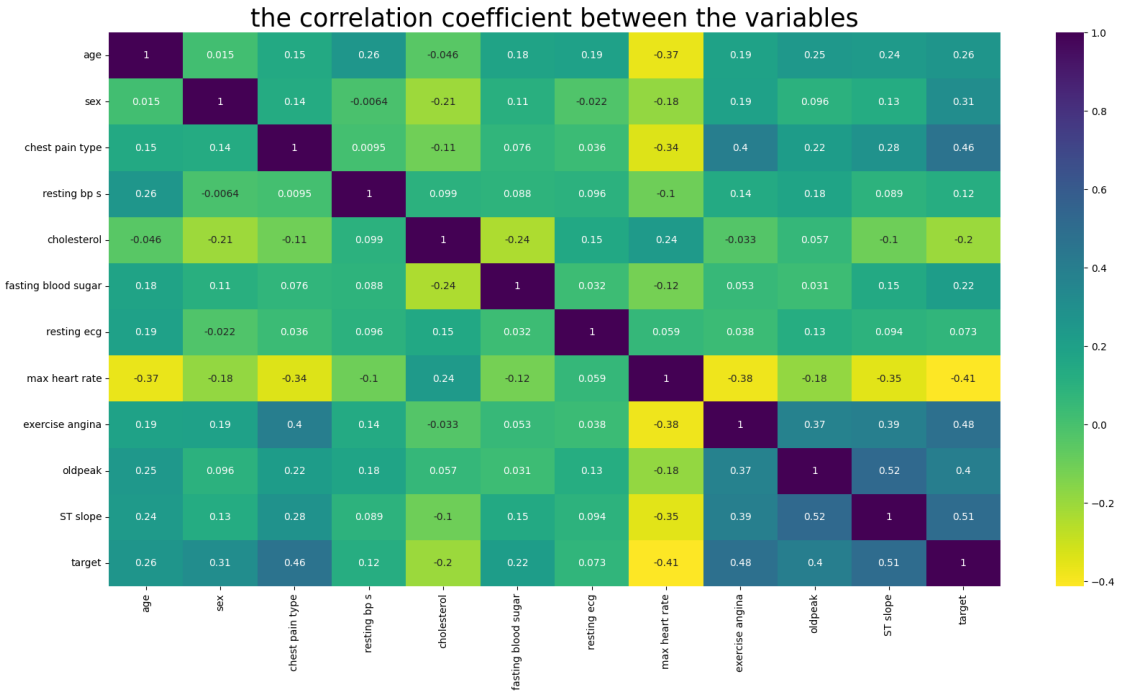
	oldpeak	ST slope	target

```
0          0.0          1       0
1          1.0          2       1
2          0.0          1       0
3          1.5          2       1
4          0.0          1       0
…          …          …       …
1185       1.2          2       1
1186       3.4          2       1
1187       1.2          2       1
1188       0.0          2       1
1189       0.0          1       0

[1190 rows x 12 columns]
```

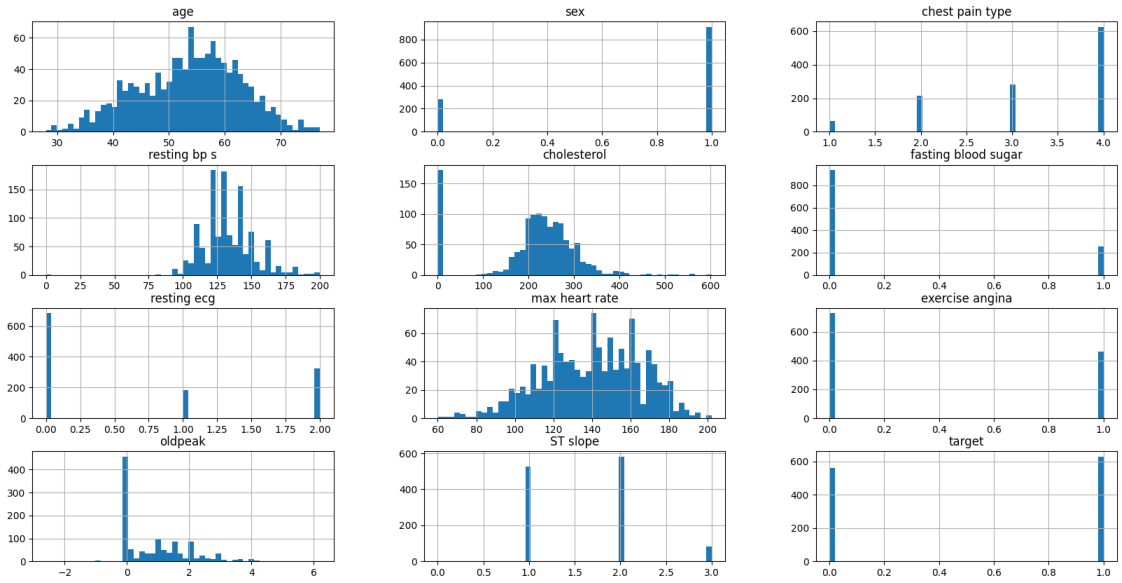
[5]:
```
sns.pairplot(data)
plt.show()
```

[6]:
```
plt.rcParams["figure.figsize"] = 20,10
sns.heatmap(data.corr(),cmap='viridis_r',annot= True)
plt.title("the correlation coefficient between the variables",size = 25);
```



[7]:
```
data.hist(figsize=(20,10),bins = 50)
```

[7]:
```
array([[<Axes: title={'center': 'age'}>, <Axes: title={'center': 'sex'}>,
        <Axes: title={'center': 'chest pain type'}>],
       [<Axes: title={'center': 'resting bp s'}>,
        <Axes: title={'center': 'cholesterol'}>,
        <Axes: title={'center': 'fasting blood sugar'}>],
       [<Axes: title={'center': 'resting ecg'}>,
        <Axes: title={'center': 'max heart rate'}>,
        <Axes: title={'center': 'exercise angina'}>],
       [<Axes: title={'center': 'oldpeak'}>,
        <Axes: title={'center': 'ST slope'}>,
        <Axes: title={'center': 'target'}>]], dtype=object)
```



[8]:
```
X=data.drop(["target"],axis=1)
X
```

[8]:

	age	sex	chest pain type	resting bp s	cholesterol	\
0	40	1	2	140	289	
1	49	0	3	160	180	
2	37	1	2	130	283	
3	48	0	4	138	214	
4	54	1	3	150	195	
…	…	…	…	…	…	
1185	45	1	1	110	264	
1186	68	1	4	144	193	
1187	57	1	4	130	131	
1188	57	0	2	130	236	
1189	38	1	3	138	175	

	fasting blood sugar	resting ecg	max heart rate	exercise angina	\
0	0	0	172	0	
1	0	0	156	0	
2	0	1	98	0	
3	0	0	108	1	
4	0	0	122	0	
…	…	…	…	…	
1185	0	0	132	0	
1186	1	0	141	0	
1187	0	0	115	1	
1188	0	2	174	0	
1189	0	0	173	0	

```
      oldpeak  ST slope
0         0.0         1
1         1.0         2
2         0.0         1
3         1.5         2
4         0.0         1
…         …         …
1185      1.2         2
1186      3.4         2
1187      1.2         2
1188      0.0         2
1189      0.0         1

[1190 rows x 11 columns]
```

[9]:
```
y=data["target"]
y=pd.DataFrame(y)
y
```

[9]:
```
      target
0          0
1          1
2          0
3          1
4          0
…        …
1185       1
1186       1
1187       1
1188       1
1189       0

[1190 rows x 1 columns]
```

Standard Scaler for Data

[10]:
```
scaler = StandardScaler(copy=True, with_mean=True, with_std=True)
X = scaler.fit_transform(X)
X
```

[10]:
```
array([[-1.46672783,  0.55599543, -1.31835093, …, -0.79521891,
        -0.84979236, -1.02321701],
       [-0.50460037, -1.79857595, -0.24893198, …, -0.79521891,
         0.07111913,  0.61558278],
       [-1.78743698,  0.55599543, -1.31835093, …, -0.79521891,
        -0.84979236, -1.02321701],
```

```
       …,
       [ 0.35062404,  0.55599543,  0.82048698, …,  1.25751537,
         0.25530143,  0.61558278],
       [ 0.35062404, -1.79857595, -1.31835093, …, -0.79521891,
        -0.84979236,  0.61558278],
       [-1.68053393,  0.55599543, -0.24893198, …, -0.79521891,
        -0.84979236, -1.02321701]])
```

[11]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15,␣
↪random_state=42, shuffle =True)
```

Applying RandomForestClassifier Model

[12]:
```
from sklearn.ensemble import RandomForestClassifier
RandomForestClassifierModel = RandomForestClassifier(criterion =␣
↪'gini',n_estimators=300,random_state=33) #criterion can be also : entropy
RandomForestClassifierModel.fit(X_train, y_train)
```

```
C:\Users\subas\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5n
2kfra8p0\LocalCache\local-packages\Python312\site-packages\sklearn\base.py:1474:
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().
  return fit_method(estimator, *args, **kwargs)
```

[12]:
```
RandomForestClassifier(n_estimators=300, random_state=33)
```

Calculating Details and Printing

[13]:
```
print('RandomForestClassifierModel Train Score is : ' ,␣
↪RandomForestClassifierModel.score(X_train, y_train))
print('RandomForestClassifierModel Test Score is : ' ,␣
↪RandomForestClassifierModel.score(X_test, y_test))
print('RandomForestClassifierModel features importances are : ' ,␣
↪RandomForestClassifierModel.feature_importances_)
```

```
RandomForestClassifierModel Train Score is :  1.0
RandomForestClassifierModel Test Score is :  0.9608938547486033
RandomForestClassifierModel features importances are :  [0.09782564 0.03856438
0.14416081 0.08278857 0.10795441 0.02139774
 0.02720902 0.12873186 0.06737376 0.11809657 0.16589724]
```

Calculating Prediction

[14]:
```
y_pred = RandomForestClassifierModel.predict(X_test)
y_pred_prob = RandomForestClassifierModel.predict_proba(X_test)
print('Predicted Value for RandomForestClassifierModel is : ' , y_pred)
```

```
print('Prediction Probabilities Value for RandomForestClassifierModel is : ' ,␣
 ↪y_pred)
```

```
Predicted Value for RandomForestClassifierModel is :  [1 0 0 1 1 1 0 0 0 0 0 1 0
0 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 0 1
 1 1 1 0 0 1 0 1 1 1 1 1 1 1 0 1 0 1 1 0 0 0 1 1 0 0 0 1 1 0 1 1 1 0 0 1 1
 1 0 0 1 1 1 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 1 1 0 1 1 1 1 0 0 0 1 1 0
 1 0 1 1 1 1 0 1 1 1 1 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 1 1
 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 0 1 0 1 1 0 1 1 1 0 1]
Prediction Probabilities Value for RandomForestClassifierModel is :  [1 0 0 1 1
1 0 0 0 0 0 1 0 0 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 0 1
 1 1 1 0 0 1 0 1 1 1 1 1 1 1 0 1 0 1 1 0 0 0 1 1 0 0 0 1 1 0 1 1 1 0 0 1 1
 1 0 0 1 1 1 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 1 1 0 1 1 1 1 0 0 0 1 1 0
 1 0 1 1 1 1 0 1 1 1 1 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 1 1
 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 0 1 0 1 1 0 1 1 1 0 1]
```

Scores of My Calculations

[15]:
```
prediction=RandomForestClassifierModel.predict(X_test)
from sklearn import metrics
print(metrics.classification_report(y_test, prediction))
print(metrics.confusion_matrix(y_test, prediction))
```

```
              precision    recall  f1-score   support

           0       0.94      0.97      0.96        80
           1       0.98      0.95      0.96        99

    accuracy                           0.96       179
   macro avg       0.96      0.96      0.96       179
weighted avg       0.96      0.96      0.96       179

[[78  2]
 [ 5 94]]
```

Finally The Accuracy of my work

[16]:
```
print("accuracy:",metrics.accuracy_score(y_test,prediction))
```

```
accuracy: 0.9608938547486033
```

AUTHOR: Melisa SUBAŞI - id:22829169256