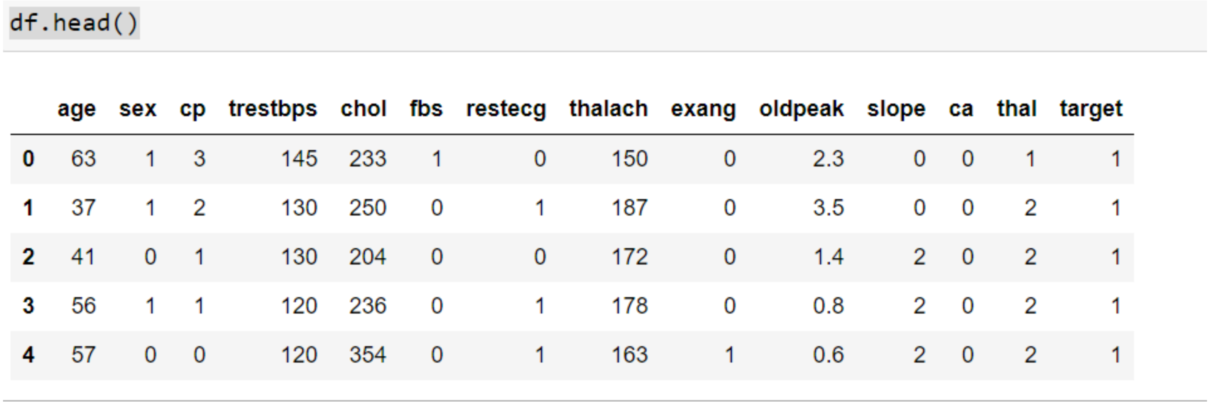
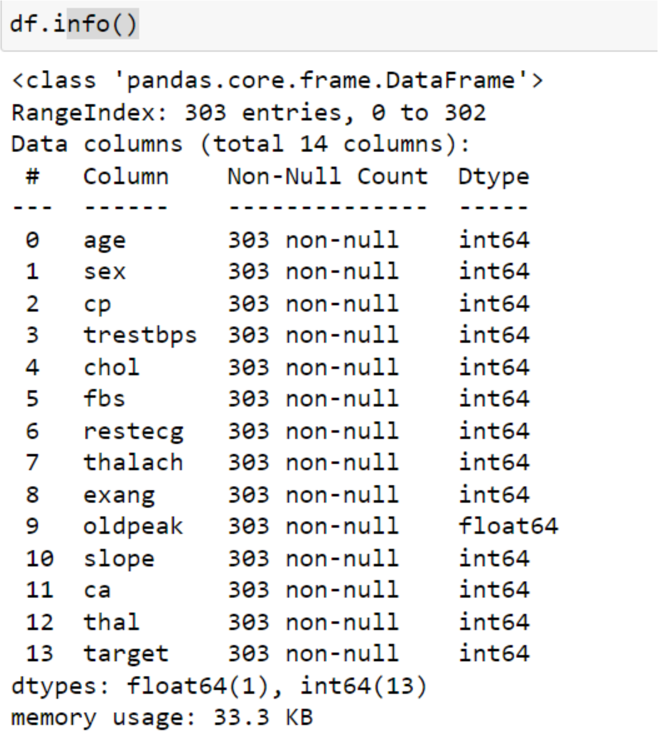
1. Preliminary analysis:

a) Perform preliminary data inspection and report the findings on the structure of the data, missing values, duplicates, etc.



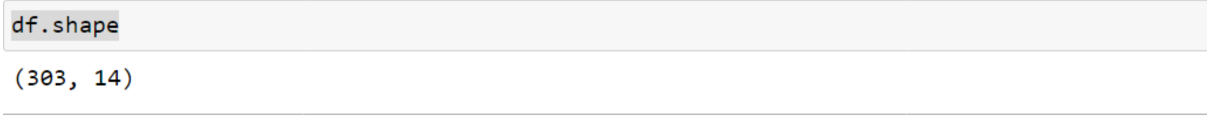
```
df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1



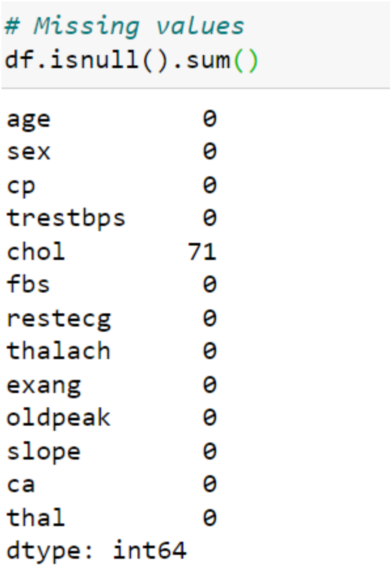
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    int64  
 12  thal      303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```



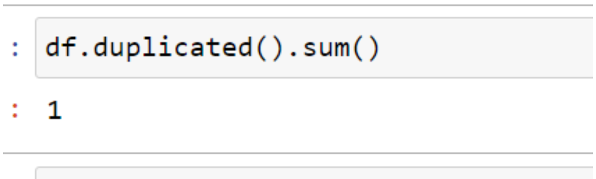
```
df.shape
```

```
(303, 14)
```



```
# Missing values
df.isnull().sum()
```

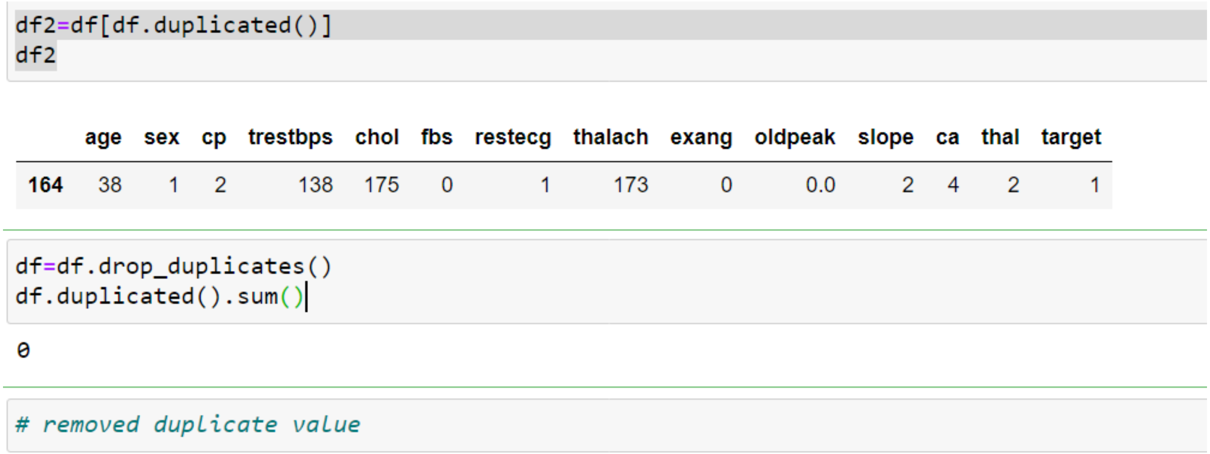
```
age          0
sex          0
cp           0
trestbps     0
chol        71
fbs          0
restecg      0
thalach      0
exang        0
oldpeak      0
slope        0
ca           0
thal         0
dtype: int64
```



```
: df.duplicated().sum()

: 1
```

b) **Based on these findings, remove duplicates (if any) and treat missing values using an appropriate strategy**



```
df2=df[df.duplicated()]
df2
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
164	38	1	2	138	175	0	1	173	0	0.0	2	4	2	1

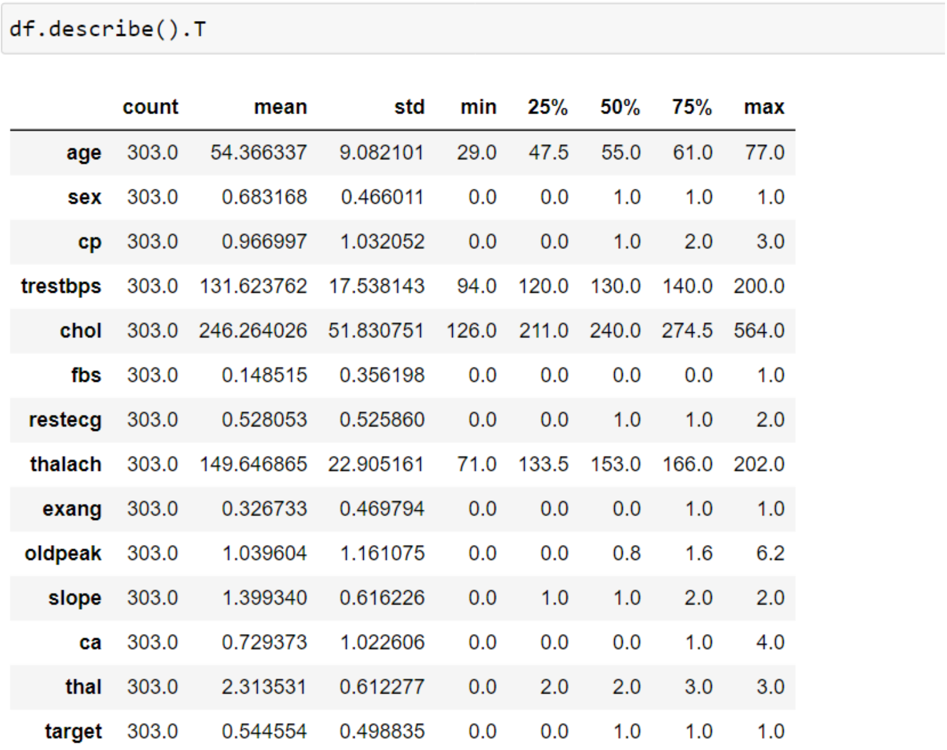
```
df=df.drop_duplicates()
df.duplicated().sum()
```

```
0
```

```
# removed duplicate value
```

2. **Prepare a report about the data explaining the distribution of the disease and the related factors using the steps listed below:**

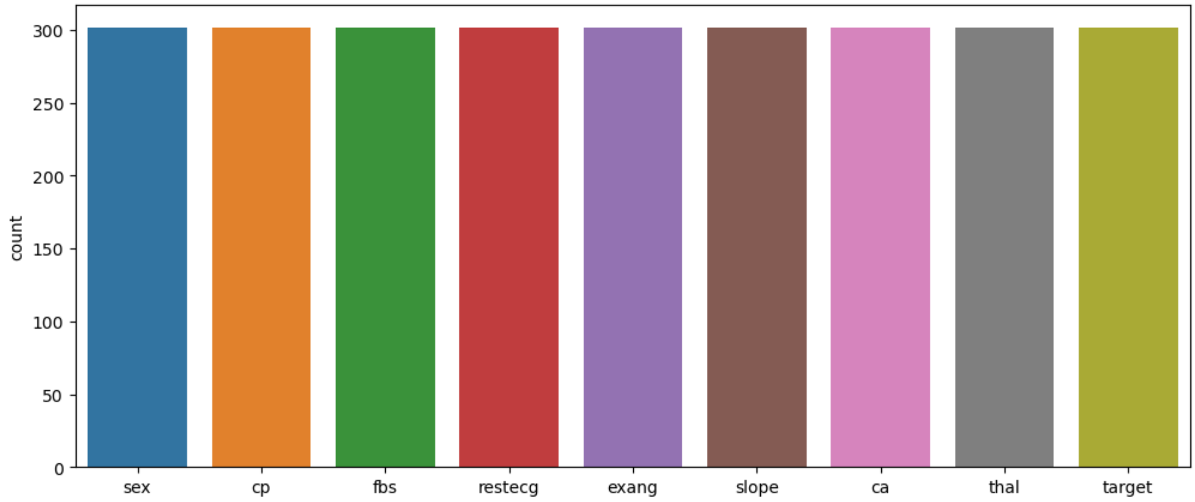
   a. **Get a preliminary statistical summary of the data and explore the measures of central tendencies and spread of the data**



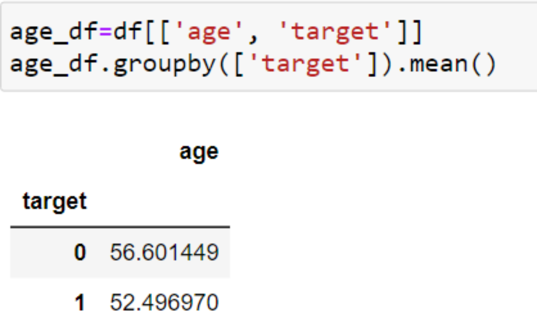
```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
age	303.0	54.366337	9.082101	29.0	47.5	55.0	61.0	77.0
sex	303.0	0.683168	0.466011	0.0	0.0	1.0	1.0	1.0
cp	303.0	0.966997	1.032052	0.0	0.0	1.0	2.0	3.0
trestbps	303.0	131.623762	17.538143	94.0	120.0	130.0	140.0	200.0
chol	303.0	246.264026	51.830751	126.0	211.0	240.0	274.5	564.0
fbs	303.0	0.148515	0.356198	0.0	0.0	0.0	0.0	1.0
restecg	303.0	0.528053	0.525860	0.0	0.0	1.0	1.0	2.0
thalach	303.0	149.646865	22.905161	71.0	133.5	153.0	166.0	202.0
exang	303.0	0.326733	0.469794	0.0	0.0	0.0	1.0	1.0
oldpeak	303.0	1.039604	1.161075	0.0	0.0	0.8	1.6	6.2
slope	303.0	1.399340	0.616226	0.0	1.0	1.0	2.0	2.0
ca	303.0	0.729373	1.022606	0.0	0.0	0.0	1.0	4.0
thal	303.0	2.313531	0.612277	0.0	2.0	2.0	3.0	3.0
target	303.0	0.544554	0.498835	0.0	0.0	1.0	1.0	1.0

b. Identify the data variables which are categorical and describe and explore these variables using the appropriate tools, such as count plot



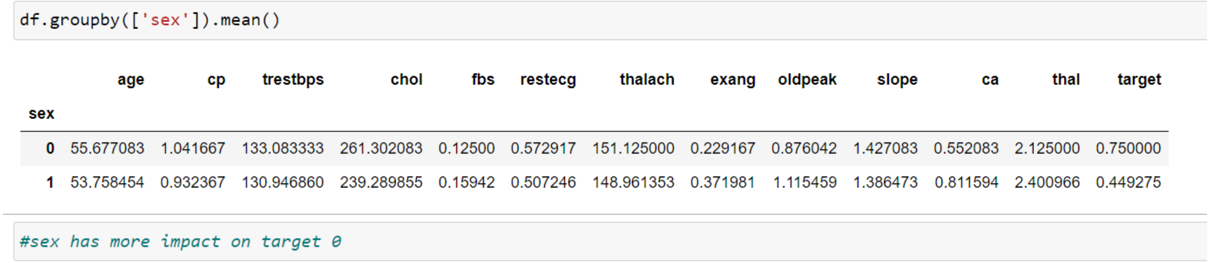
c. Study the occurrence of CVD across the Age category



```
age_df=df[['age', 'target']]
age_df.groupby(['target']).mean()
```

	age
target	
0	56.601449
1	52.496970

d. **Study the composition of all patients with respect to the Sex category**

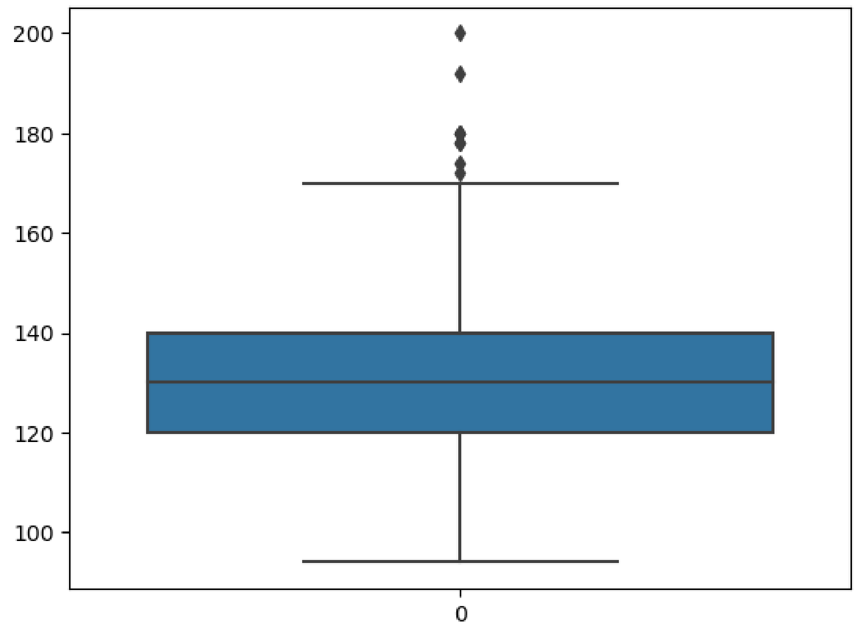


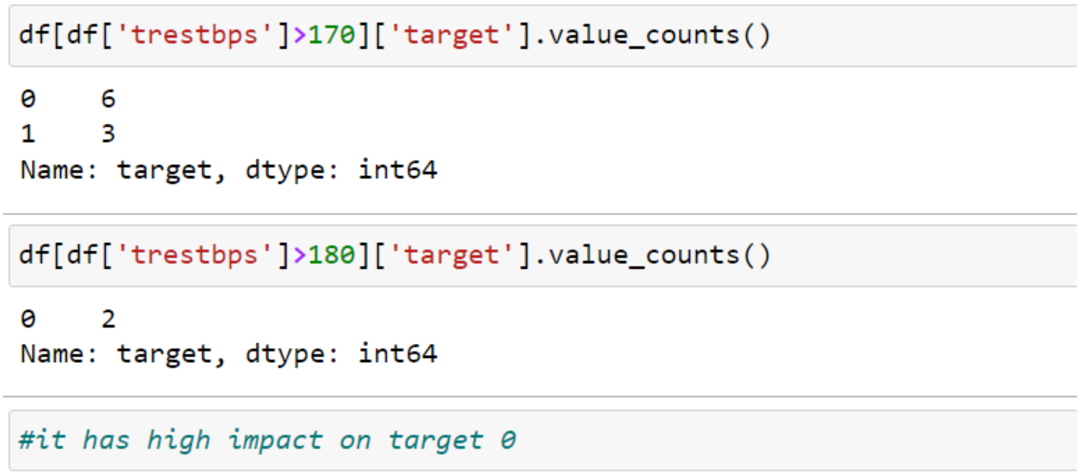
```
df.groupby(['sex']).mean()
```

	age	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
sex													
0	55.677083	1.041667	133.083333	261.302083	0.12500	0.572917	151.125000	0.229167	0.876042	1.427083	0.552083	2.125000	0.750000
1	53.758454	0.932367	130.946860	239.289855	0.15942	0.507246	148.961353	0.371981	1.115459	1.386473	0.811594	2.400966	0.449275

```
#sex has more impact on target 0
```

e. **Study if one can detect heart attacks based on anomalies in the resting blood pressure (trestbps) of a patient.**





```
df[df['trestbps']>170]['target'].value_counts()
```

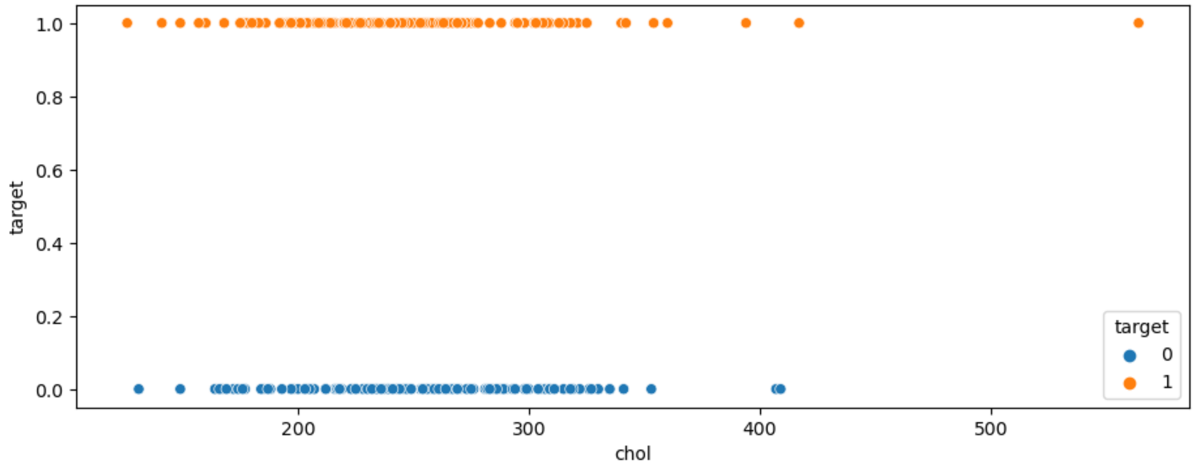
```
0    6
1    3
Name: target, dtype: int64
```

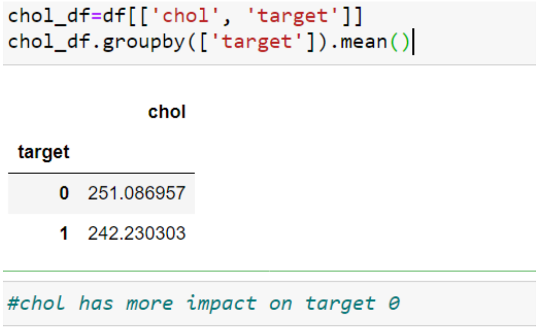
```
df[df['trestbps']>180]['target'].value_counts()
```

```
0    2
Name: target, dtype: int64
```

```
#it has high impact on target 0
```

f. Describe the relationship between cholesterol levels and a target variable



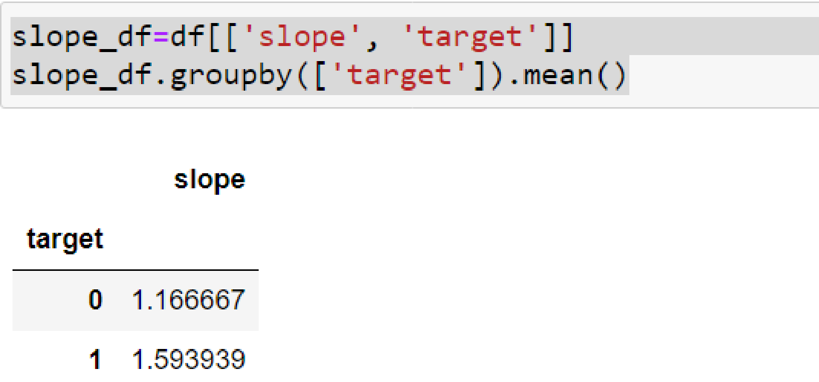


```
chol_df=df[['chol', 'target']]
chol_df.groupby(['target']).mean()
```

	chol
target	
0	251.086957
1	242.230303

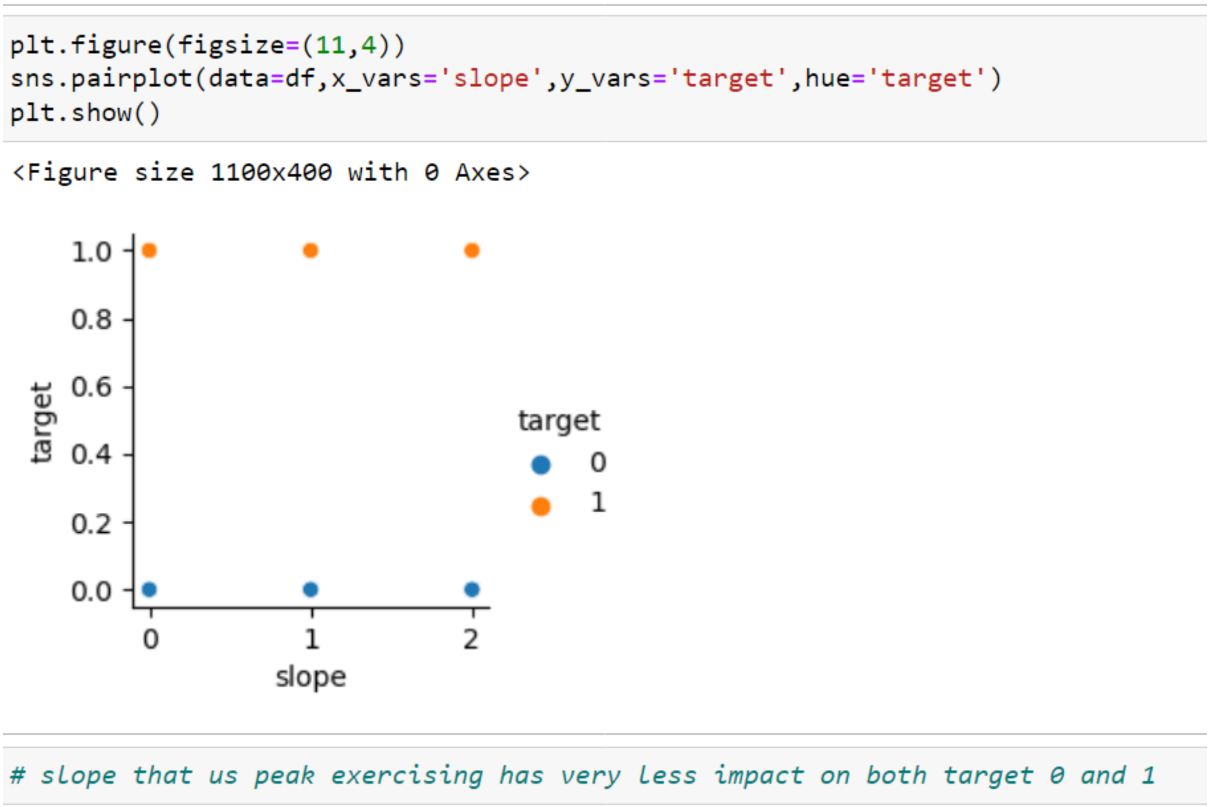
```
#chol has more impact on target 0
```

g. **State what relationship exists between peak exercising and the occurrence of a heart attack**



```
slope_df=df[['slope', 'target']]
slope_df.groupby(['target']).mean()
```

	slope
target	
0	1.166667
1	1.593939

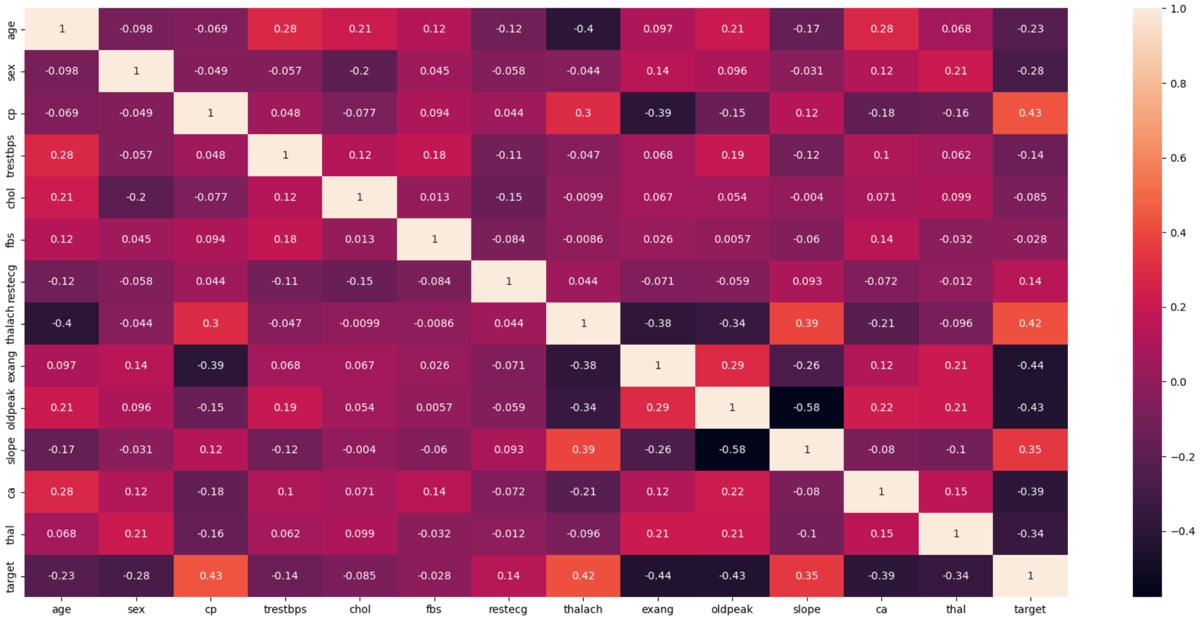


```
plt.figure(figsize=(11,4))
sns.pairplot(data=df,x_vars='slope',y_vars='target',hue='target')
plt.show()
```

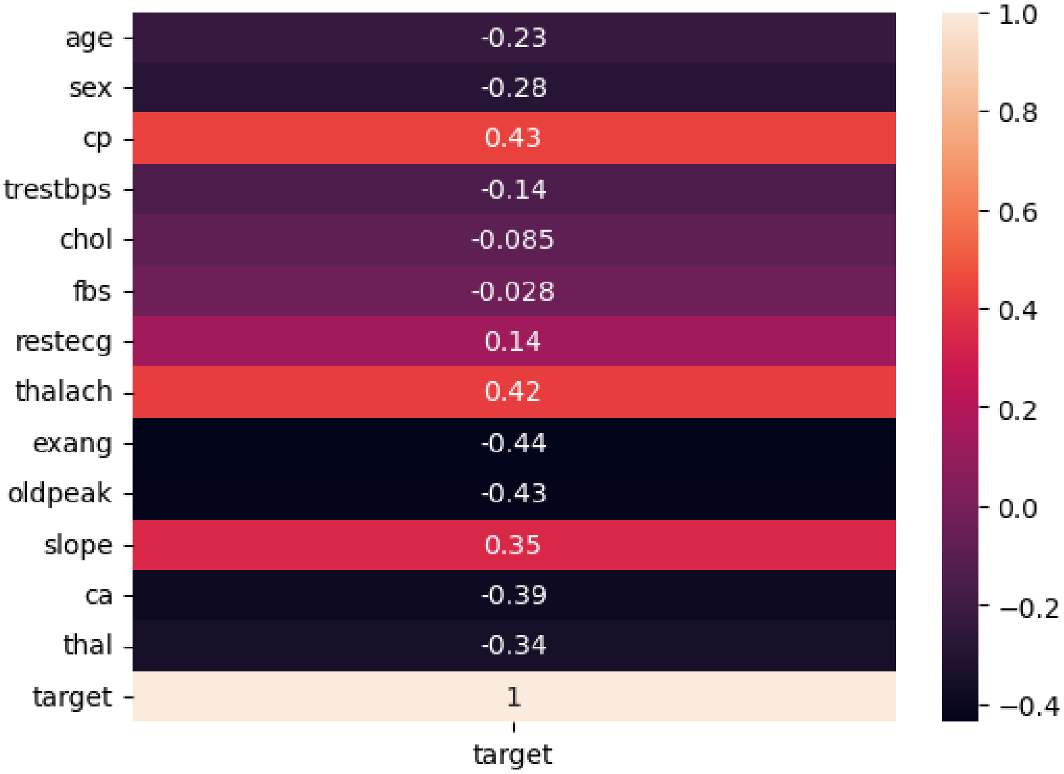
```
<Figure size 1100x400 with 0 Axes>
```

```
# slope that us peak exercising has very less impact on both target 0 and 1
```

h. **Check if thalassemia is a major cause of CVD**

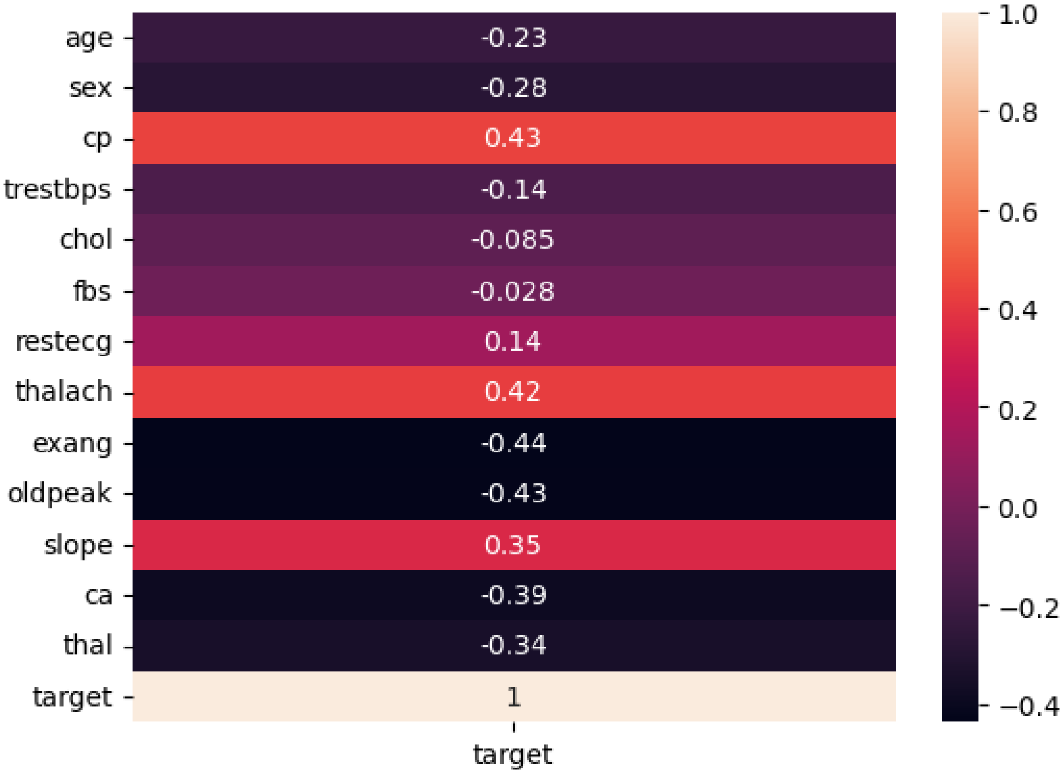


	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
age	1	-0.098	-0.069	0.28	0.21	0.12	-0.12	-0.4	0.097	0.21	-0.17	0.28	0.068	-0.23
sex	-0.098	1	-0.049	-0.057	-0.2	0.045	-0.058	-0.044	0.14	0.096	-0.031	0.12	0.21	-0.28
cp	-0.069	-0.049	1	0.048	-0.077	0.094	0.044	0.3	-0.39	-0.15	0.12	-0.18	-0.16	0.43
trestbps	0.28	-0.057	0.048	1	0.12	0.18	-0.11	-0.047	0.068	0.19	-0.12	0.1	0.062	-0.14
chol	0.21	-0.2	-0.077	0.12	1	0.013	-0.15	-0.0099	0.067	0.054	-0.004	0.071	0.099	-0.085
fbs	0.12	0.045	0.094	0.18	0.013	1	-0.084	-0.0086	0.026	0.0057	-0.06	0.14	-0.032	-0.028
restecg	-0.12	-0.058	0.044	-0.11	-0.15	-0.084	1	0.044	-0.071	-0.059	0.093	-0.072	-0.012	0.14
thalach	-0.4	-0.044	0.3	-0.047	-0.0099	-0.0086	0.044	1	-0.38	-0.34	0.39	-0.21	-0.096	0.42
exang	0.097	0.14	-0.39	0.068	0.067	0.026	-0.071	-0.38	1	0.29	-0.26	0.12	0.21	-0.44
oldpeak	0.21	0.096	-0.15	0.19	0.054	0.0057	-0.059	-0.34	0.29	1	-0.58	0.22	0.21	-0.43
slope	-0.17	-0.031	0.12	-0.12	-0.004	-0.06	0.093	0.39	-0.26	-0.58	1	-0.08	-0.1	0.35
ca	0.28	0.12	-0.18	0.1	0.071	0.14	-0.072	-0.21	0.12	0.22	-0.08	1	0.15	-0.39
thal	0.068	0.21	-0.16	0.062	0.099	-0.032	-0.012	-0.096	0.21	0.21	-0.1	0.15	1	-0.34
target	-0.23	-0.28	0.43	-0.14	-0.085	-0.028	0.14	0.42	-0.44	-0.43	0.35	-0.39	-0.34	1

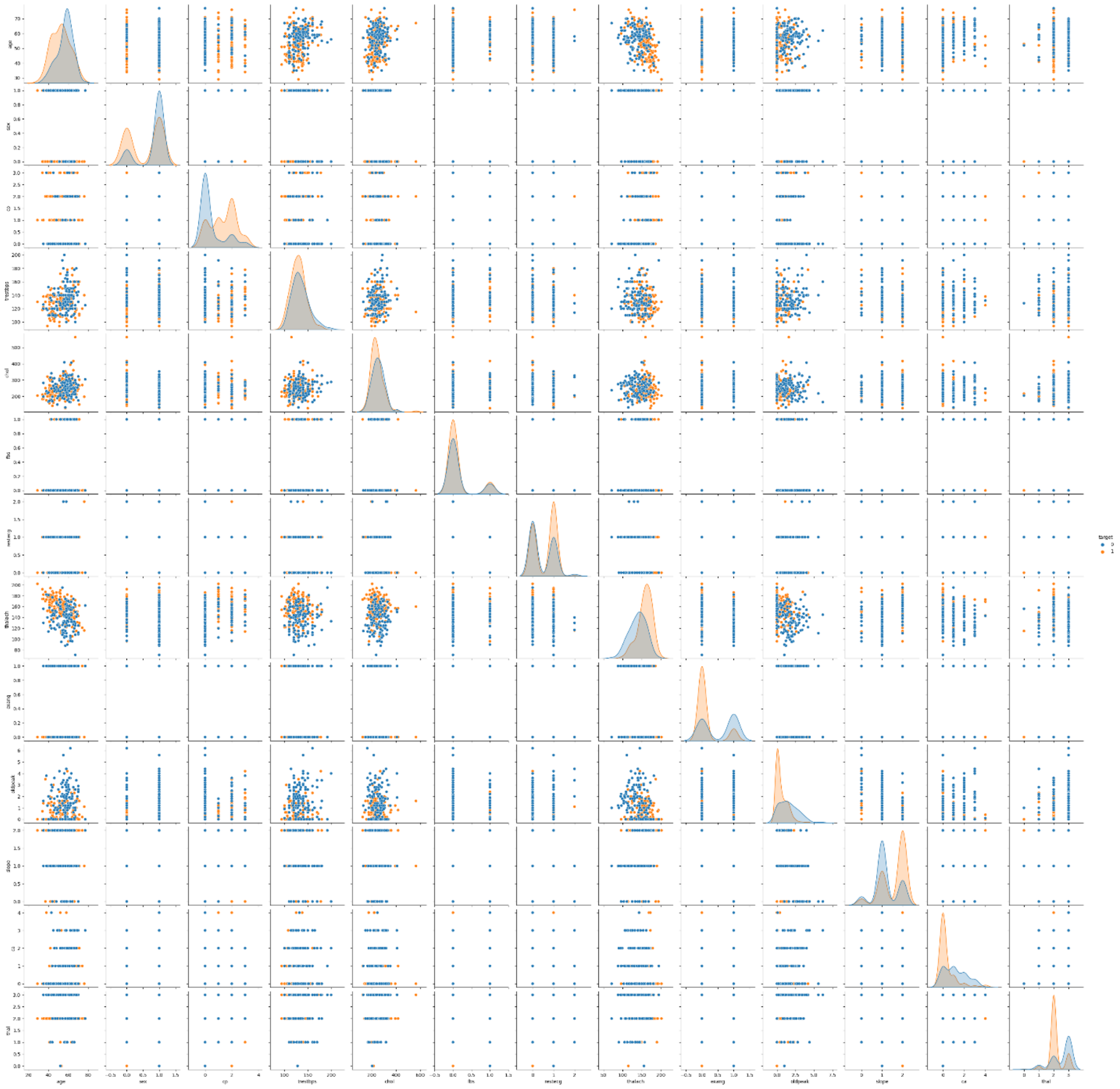


	target
age	-0.23
sex	-0.28
cp	0.43
trestbps	-0.14
chol	-0.085
fbs	-0.028
restecg	0.14
thalach	0.42
exang	-0.44
oldpeak	-0.43
slope	0.35
ca	-0.39
thal	-0.34
target	1

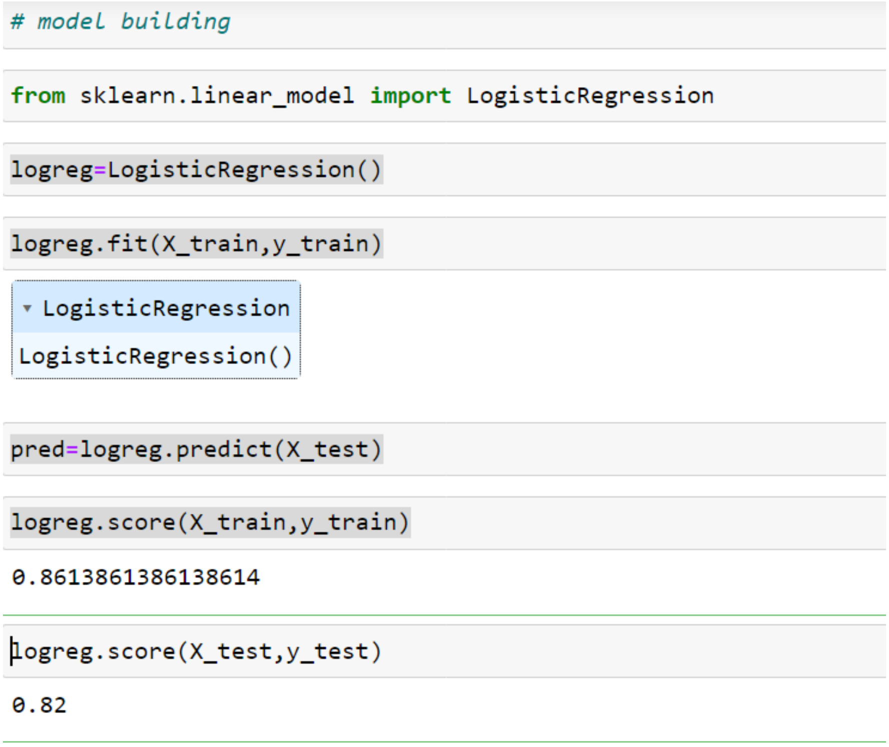
i. List how the other factors determine the occurrence of CVD



j. **Use a pair plot to understand the relationship between all the given variables**



3. Build a baseline model to predict the risk of a heart attack using a logistic regression and random forest and explore the results while using correlation analysis and logistic regression (leveraging standard error and p-values from statsmodels) for feature selection



```
# model building
```

```
from sklearn.linear_model import LogisticRegression
```

```
logreg=LogisticRegression()
```

```
logreg.fit(X_train,y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

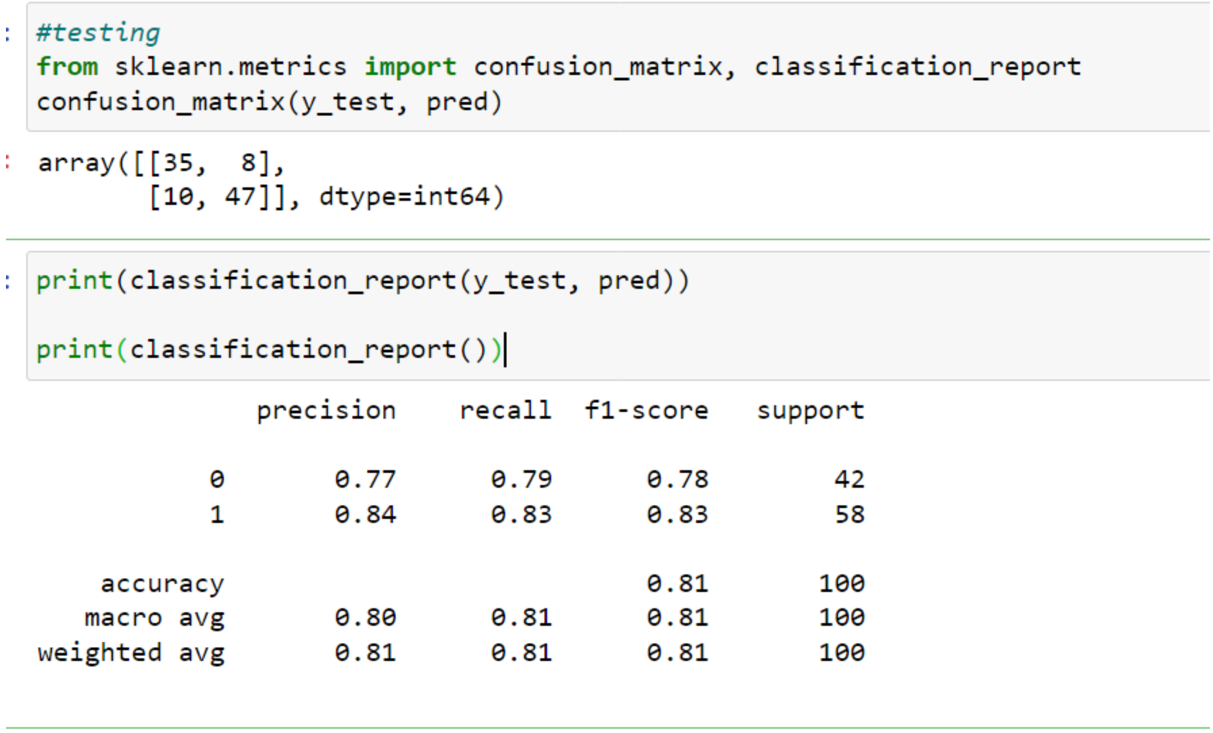
```
pred=logreg.predict(X_test)
```

```
logreg.score(X_train,y_train)
```

```
0.8613861386138614
```

```
logreg.score(X_test,y_test)
```

```
0.82
```



```
#testing
from sklearn.metrics import confusion_matrix, classification_report
confusion_matrix(y_test, pred)
```

```
array([[35,  8],
       [10, 47]], dtype=int64)
```

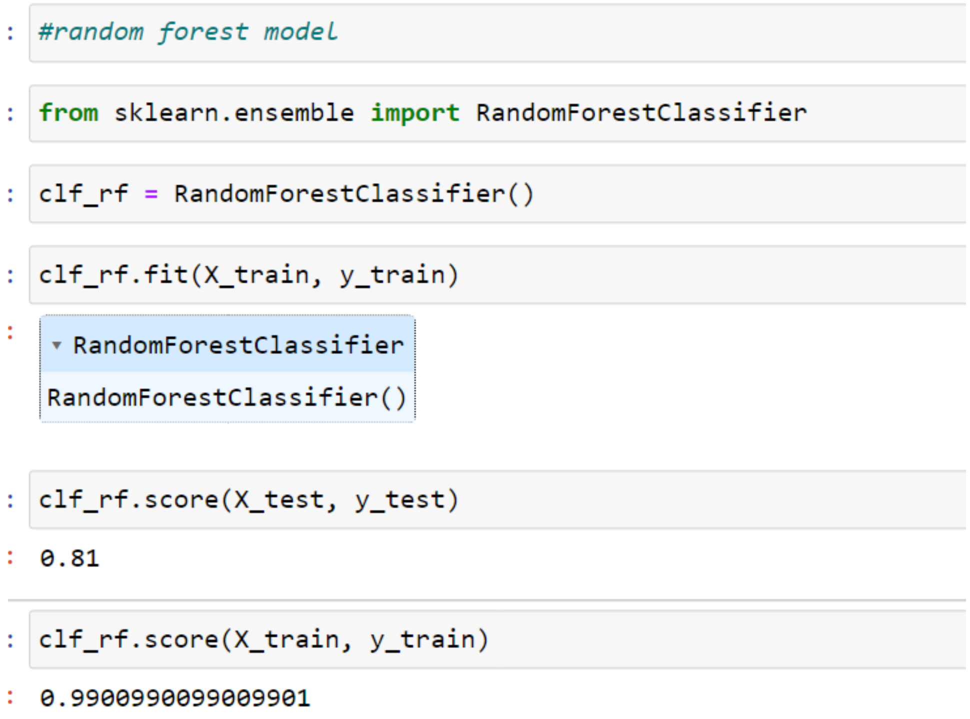
```
print(classification_report(y_test, pred))

print(classification_report())
```

```
              precision    recall  f1-score   support

           0       0.77      0.79      0.78        42
           1       0.84      0.83      0.83        58

    accuracy                           0.81       100
   macro avg       0.80      0.81      0.81       100
weighted avg       0.81      0.81      0.81       100
```



```
#random forest model
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
clf_rf = RandomForestClassifier()
```

```
clf_rf.fit(X_train, y_train)
```

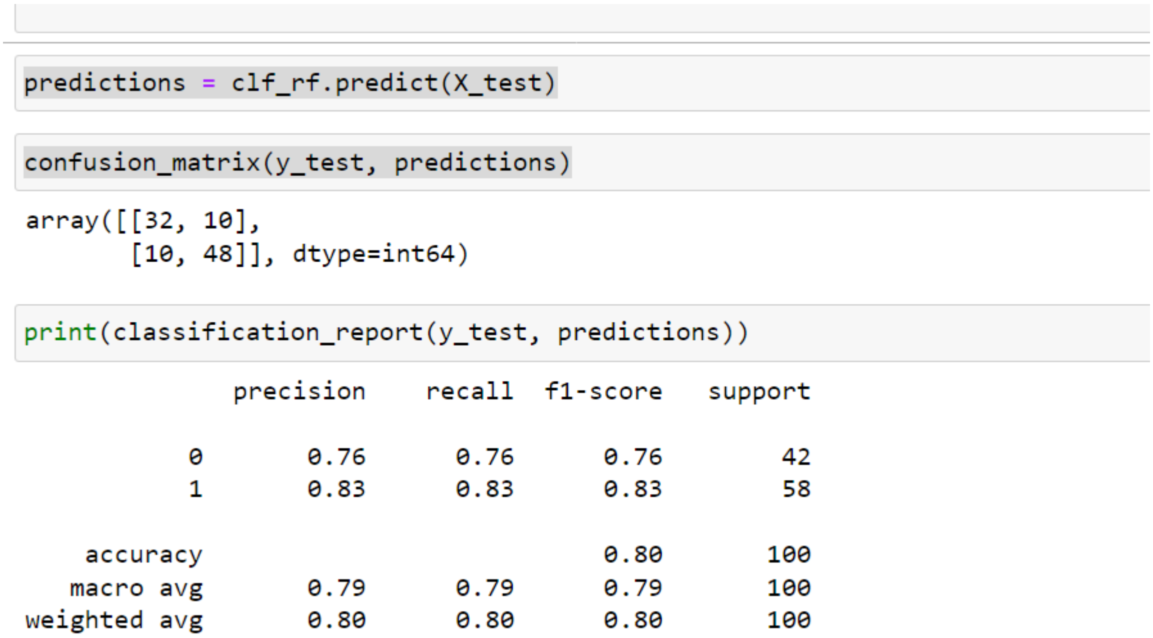
```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
clf_rf.score(X_test, y_test)
```

```
0.81
```

```
clf_rf.score(X_train, y_train)
```

```
0.9900990099009901
```



```
predictions = clf_rf.predict(X_test)
```

```
confusion_matrix(y_test, predictions)
```

```
array([[32, 10],
       [10, 48]], dtype=int64)
```

```
print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.76      0.76      0.76        42
           1       0.83      0.83      0.83        58

    accuracy                           0.80       100
   macro avg       0.79      0.79      0.79       100
weighted avg       0.80      0.80      0.80       100
```