Single Logistic Regression (Binary classification)

import required package

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

read data from source and describing

In [2]:
```python
df = pd.read_csv('heart_disease.csv')
df.head()
```

Out[2]:

	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

In [3]:
```python
print(df.columns)
```

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

In [4]:
```python
print(df.describe())
```

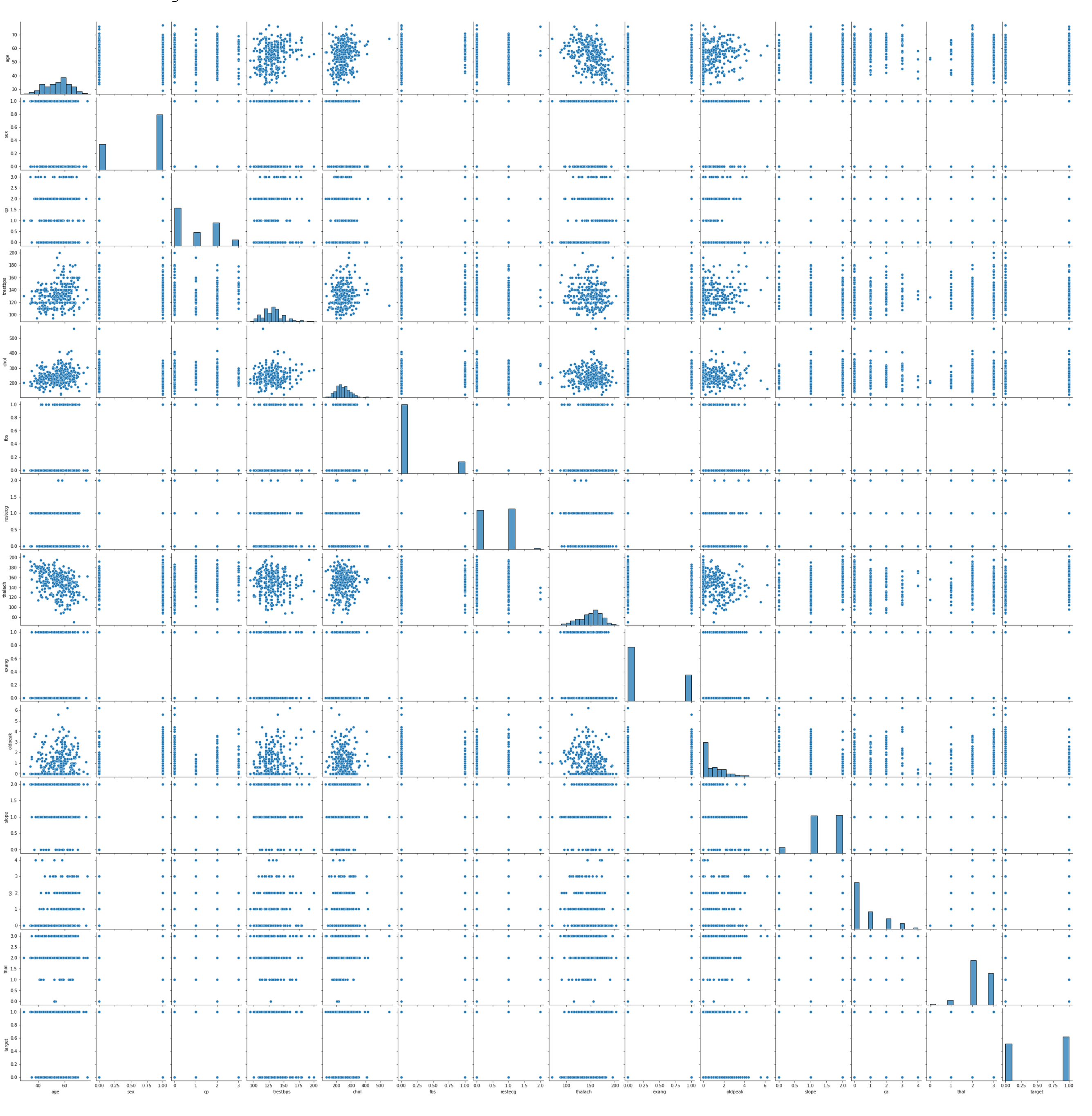
```
              age         sex          cp    trestbps        chol         fbs  \
count  303.000000  303.000000  303.000000  303.000000  303.000000  303.000000
mean    54.366337    0.683168    0.966997  131.623762  246.264026    0.148515
std      9.082101    0.466011    1.032052   17.538143   51.830751    0.356198
min     29.000000    0.000000    0.000000   94.000000  126.000000    0.000000
25%     47.500000    0.000000    0.000000  120.000000  211.000000    0.000000
50%     55.000000    1.000000    1.000000  130.000000  240.000000    0.000000
75%     61.000000    1.000000    2.000000  140.000000  274.500000    0.000000
max     77.000000    1.000000    3.000000  200.000000  564.000000    1.000000

          restecg     thalach       exang     oldpeak       slope          ca  \
count  303.000000  303.000000  303.000000  303.000000  303.000000  303.000000
mean     0.528053  149.646865    0.326733    1.039604    1.399340    0.729373
std      0.525860   22.905161    0.469794    1.161075    0.616226    1.022606
min      0.000000   71.000000    0.000000    0.000000    0.000000    0.000000
25%      0.000000  133.500000    0.000000    0.000000    1.000000    0.000000
50%      1.000000  153.000000    0.000000    0.800000    1.000000    0.000000
75%      1.000000  166.000000    1.000000    1.600000    2.000000    1.000000
max      2.000000  202.000000    1.000000    6.200000    2.000000    4.000000

             thal      target
count  303.000000  303.000000
mean     2.313531    0.544554
std      0.612277    0.498835
min      0.000000    0.000000
25%      2.000000    0.000000
50%      2.000000    1.000000
75%      3.000000    1.000000
max      3.000000    1.000000
```

In [5]:
```python
## chec for data types
print(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
None
```

In [6]:
```python
sns.pairplot(df)
```

Out[6]: <seaborn.axisgrid.PairGrid at 0x1cd080f7850>



check the relation between variables

In [7]:
```python
corr = df.corr()
corr.style.background_gradient(cmap = 'Greens')
```

Out[7]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	old
age	1.000000	-0.098447	-0.068653	0.279351	0.213678	0.121308	-0.116211	-0.398522	0.096801	0.21
sex	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	0.045032	-0.058196	-0.044020	0.141664	0.09
cp	-0.068653	-0.049353	1.000000	0.047608	-0.076904	0.094444	0.044421	0.295762	-0.394280	-0.14
trestbps	0.279351	-0.056769	0.047608	1.000000	0.123174	0.177531	-0.114103	-0.046698	0.067616	0.19
chol	0.213678	-0.197912	-0.076904	0.123174	1.000000	0.013294	-0.151040	-0.009940	0.067023	0.05
fbs	0.121308	0.045032	0.094444	0.177531	0.013294	1.000000	-0.084189	-0.008567	0.025665	0.00
restecg	-0.116211	-0.058196	0.044421	-0.114103	-0.151040	-0.084189	1.000000	0.044123	-0.070733	-0.05
thalach	-0.398522	-0.044020	0.295762	-0.046698	-0.009940	-0.008567	0.044123	1.000000	-0.378812	-0.34
exang	0.096801	0.141664	-0.394280	0.067616	0.067023	0.025665	-0.070733	-0.378812	1.000000	0.28
oldpeak	0.210013	0.096093	-0.149230	0.193216	0.053952	0.005747	-0.058770	-0.344187	0.288223	1.00
slope	-0.168814	-0.030711	0.119717	-0.121475	-0.004038	-0.059894	0.093045	0.386784	-0.257748	-0.57
ca	0.276326	0.118261	-0.181053	0.101389	0.070511	0.137979	-0.072042	-0.213177	0.115739	0.22
thal	0.068001	0.210041	-0.161736	0.062210	0.098803	-0.032019	-0.011981	-0.096439	0.206754	0.21
target	-0.225439	-0.280937	0.433798	-0.144931	-0.085239	-0.028046	0.137230	0.421741	-0.436757	-0.43

In [8]:
```python
### check the relation between variables

corr = df.corr()
corr.style.background_gradient(cmap = 'Greens')
```

Out[8]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	old
age	1.000000	-0.098447	-0.068653	0.279351	0.213678	0.121308	-0.116211	-0.398522	0.096801	0.21
sex	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	0.045032	-0.058196	-0.044020	0.141664	0.09
cp	-0.068653	-0.049353	1.000000	0.047608	-0.076904	0.094444	0.044421	0.295762	-0.394280	-0.14
trestbps	0.279351	-0.056769	0.047608	1.000000	0.123174	0.177531	-0.114103	-0.046698	0.067616	0.19
chol	0.213678	-0.197912	-0.076904	0.123174	1.000000	0.013294	-0.151040	-0.009940	0.067023	0.05
fbs	0.121308	0.045032	0.094444	0.177531	0.013294	1.000000	-0.084189	-0.008567	0.025665	0.00
restecg	-0.116211	-0.058196	0.044421	-0.114103	-0.151040	-0.084189	1.000000	0.044123	-0.070733	-0.05
thalach	-0.398522	-0.044020	0.295762	-0.046698	-0.009940	-0.008567	0.044123	1.000000	-0.378812	-0.34
exang	0.096801	0.141664	-0.394280	0.067616	0.067023	0.025665	-0.070733	-0.378812	1.000000	0.28
oldpeak	0.210013	0.096093	-0.149230	0.193216	0.053952	0.005747	-0.058770	-0.344187	0.288223	1.00
slope	-0.168814	-0.030711	0.119717	-0.121475	-0.004038	-0.059894	0.093045	0.386784	-0.257748	-0.57
ca	0.276326	0.118261	-0.181053	0.101389	0.070511	0.137979	-0.072042	-0.213177	0.115739	0.22
thal	0.068001	0.210041	-0.161736	0.062210	0.098803	-0.032019	-0.011981	-0.096439	0.206754	0.21
target	-0.225439	-0.280937	0.433798	-0.144931	-0.085239	-0.028046	0.137230	0.421741	-0.436757	-0.43

In [9]:
```python
### decide which columns to be used
threshold = 0.28

for ele in df.columns:
    corr_coef = np.corrcoef(df['target'], df[ele])
    if (corr_coef[0,1] >= threshold) or (corr_coef[0,1] <= -threshold):
        print(f'correlation of column {ele} is : {corr_coef[0,1]}')
```

```
correlation of column sex is : -0.2809365755017666
correlation of column cp is : 0.433798261506894
correlation of column thalach is : 0.42174093381067473
correlation of column exang is : -0.43675708335330277
correlation of column oldpeak is : -0.43069600168736843
correlation of column slope is : 0.3458770782417252
correlation of column ca is : -0.3917239923512522
correlation of column thal is : -0.34402926803831047
correlation of column target is : 1.0
```

select input and op variable

In [10]:
```python
x = df.drop(['age','trestbps','chol', 'fbs', 'restecg', 'target'], axis=1)
y = df['target']
```

split the data

In [11]:
```python
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state= 123456, train_
```

creating a model

In [12]:
```python
from sklearn.linear_model import LogisticRegressionCV
model = LogisticRegressionCV(max_iter=1000)
```

In [13]:
```python
## fit the data
model.fit(x_train, y_train)
```

Out[13]: LogisticRegressionCV(max_iter=1000)

parameters to fine tune model

In [14]:
```python
print(model.get_params())
```

```
{'Cs': 10, 'class_weight': None, 'cv': None, 'dual': False, 'fit_intercept': True, 'in
tercept_scaling': 1.0, 'l1_ratios': None, 'max_iter': 1000, 'multi_class': 'auto', 'n_
jobs': None, 'penalty': 'l2', 'random_state': None, 'refit': True, 'scoring': None, 's
olver': 'lbfgs', 'tol': 0.0001, 'verbose': 0}
```

predict the values for unseen data

In [15]:
```python
y_prediction = model.predict(x_test)
#print(y_prediction)
```

evaluation of classification model

In [16]:
```python
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_prediction)
print(cm)
```

```
[[19  5]
 [ 1 36]]
```

In [17]:
```python
correct = cm[0,0] + cm[1,1]
wrong = cm[1,0] + cm[0,1]
total = correct + wrong
accuracy = correct/total
print(accuracy)
```

```
0.9016393442622951
```

In [18]:
```python
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, y_prediction))
```

```
0.9016393442622951
```

precision value

In [19]:
```python
from sklearn.metrics import precision_score
print(precision_score(y_test, y_prediction))
```

```
0.8780487804878049
```

recal value

In [20]:
```python
from sklearn.metrics import recall_score
print(recall_score(y_test, y_prediction))
```

```
0.972972972972973
```

F1 score

In [21]:
```python
from sklearn.metrics import f1_score
print(f1_score(y_test, y_prediction))
```

```
0.923076923076923
```

classification report

In [22]:
```python
from sklearn.metrics import classification_report
print(classification_report(y_test, y_prediction))
```

```
              precision    recall  f1-score   support

           0       0.95      0.79      0.86        24
           1       0.88      0.97      0.92        37

    accuracy                           0.90        61
   macro avg       0.91      0.88      0.89        61
weighted avg       0.91      0.90      0.90        61
```

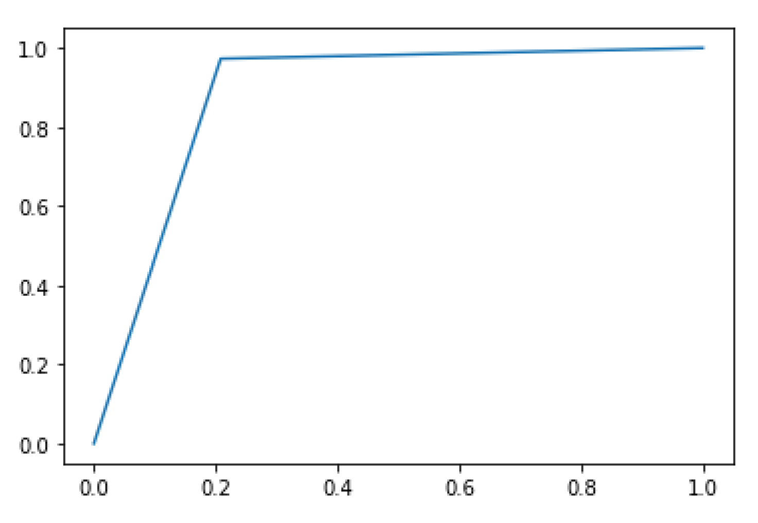
In [23]:
```python
from sklearn.metrics import roc_curve, roc_auc_score, plot_roc_curve

print(roc_auc_score(y_test, y_prediction))
```

```
0.8823198198198198
```

In [24]:
```python
fpr, tpr, threshold = roc_curve(y_test, y_prediction)
print(fpr)
print(tpr)
print(threshold)
plt.plot(fpr, tpr)
```

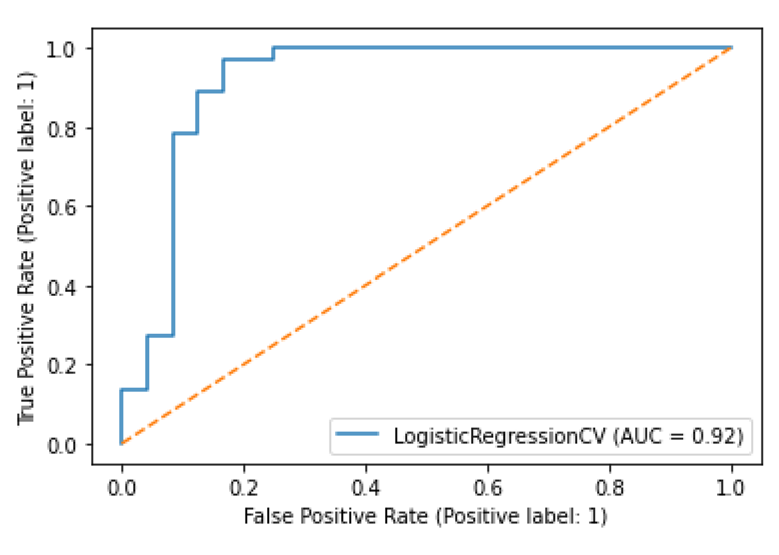
```
[0.         0.20833333 1.        ]
[0.         0.97297297 1.        ]
[2 1 0]
```

Out[24]: [<matplotlib.lines.Line2D at 0x1cd0811a790>]



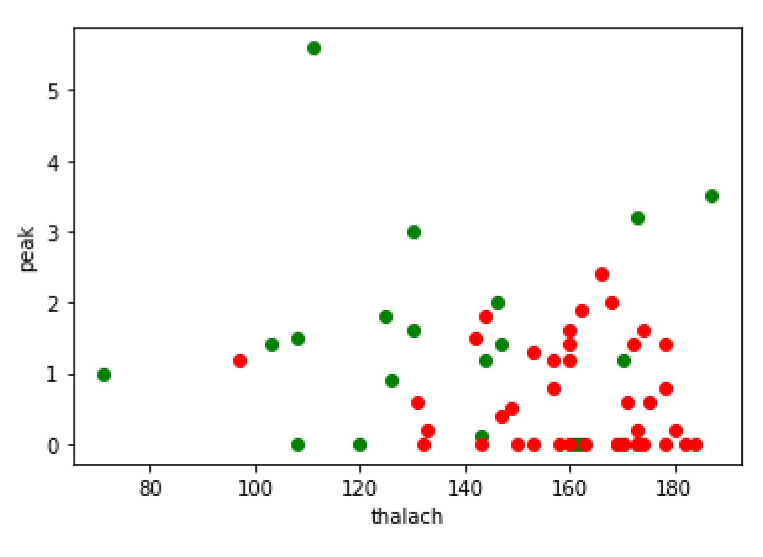
In [25]:
```python
plot = plot_roc_curve(model, x_test, y_test)
plt.plot([0,1], [0,1], linestyle = '--')
```

Out[25]: [<matplotlib.lines.Line2D at 0x1cd1284ca00>]



In [26]:
```python
plt.scatter(x_test['thalach'][y_prediction == 0], x_test['oldpeak'][y_prediction == 0
plt.scatter(x_test['thalach'][y_prediction == 1], x_test['oldpeak'][y_prediction == 1
plt.xlabel('thalach')
plt.ylabel('peak')
```

Out[26]: Text(0, 0.5, 'peak')



In [ ]: