## Logical Regression – Case study.

In this case study we will try to solve a logical regression analysis (LRA) problem. In (LRA) very often the objective is to predict one of the two possible outcomes. True or false. Have it or do not have it. Exist or is absent. Sometimes we may have to categorize or classify, for example what kind of flower is. What kind of animal species is the predicted outcome.

This case will focus on logical regression. The outcome can be only zero or one.

## Problem Introduction.

The real procedure to determine if a person has a heart disease is an invasive process. It is costly and painfully for individuals. The objective is to find a model that can help predict existence of heart problems using clinical available test results.
A hospital has conducted a test to determine if an individual has a heart disease. The data set contains 14 physical attributes based on physical testing of a patient (13 tested features and one feature for the label). Blood samples are taken and the patient conducts a brief exercise test. The label field is the presence of heart disease in the patient. 0 is marked as no presence, 1 means the patient has a heart disease. More info about this data set and study can be also found in the following link.

**df = pd.read_csv('heart.csv')**

The first five rows of the data are shown in the table below.

**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 |

Age – how old is the person.
Sex – gender of the person (1 = male; 0 = female) .
cp- chest pain type (4 values)
trestbps – resting blood pressure
chol – serum cholestoral in mg/dl
fbs – fasting blood sugar > 120 mg/dl
restecg – resting electrocardiographic results (values 0,1,2)
thalach – maximum heart rate achieved
exang – exercise induced angina
oldpeak – ST depression induced by exercise relative to rest
slope – the slope of the peak exercise ST segment
ca- number of major vessels (0-3) colored by flourosopy
thal – 3 = normal; 6 = fixed defect; 7 = reversable defect
target – 0 is for no presence of heart disease; 1 is for presence of heart disease.

## Data exploration

### 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
```

The data has 13 features. All of them are numeric. The data set is clean and ready for analysis.

### df.describe().transpose()

| | 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 |

It is important to check if the target has some typos.

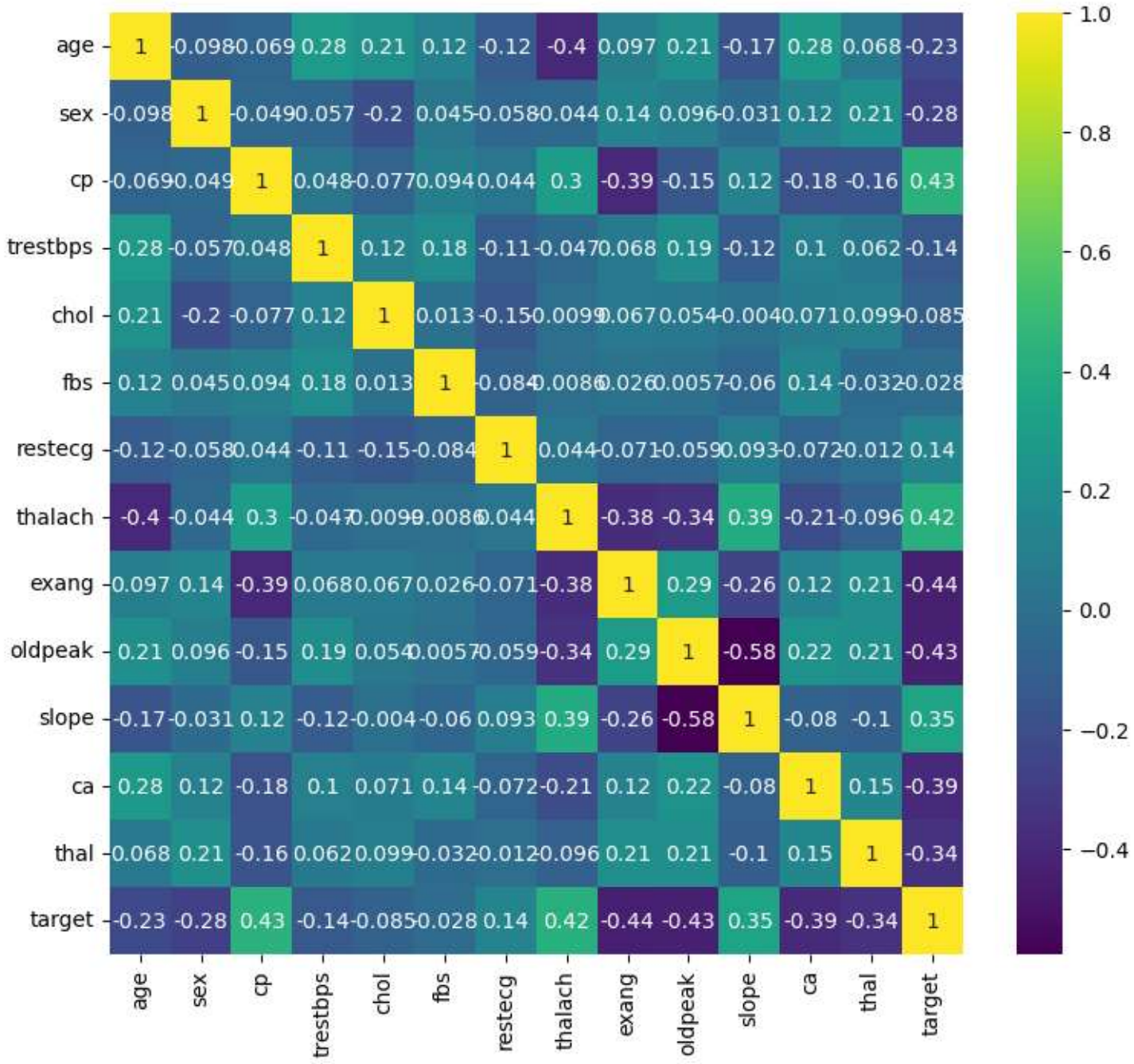
```
df['target'].unique()
```

```
array([1, 0], dtype=int64
```

## Data visualization.

All the values are numeric. It is good practice to check for existence of correlation between feature and target. To know which columns to analyze further in details.

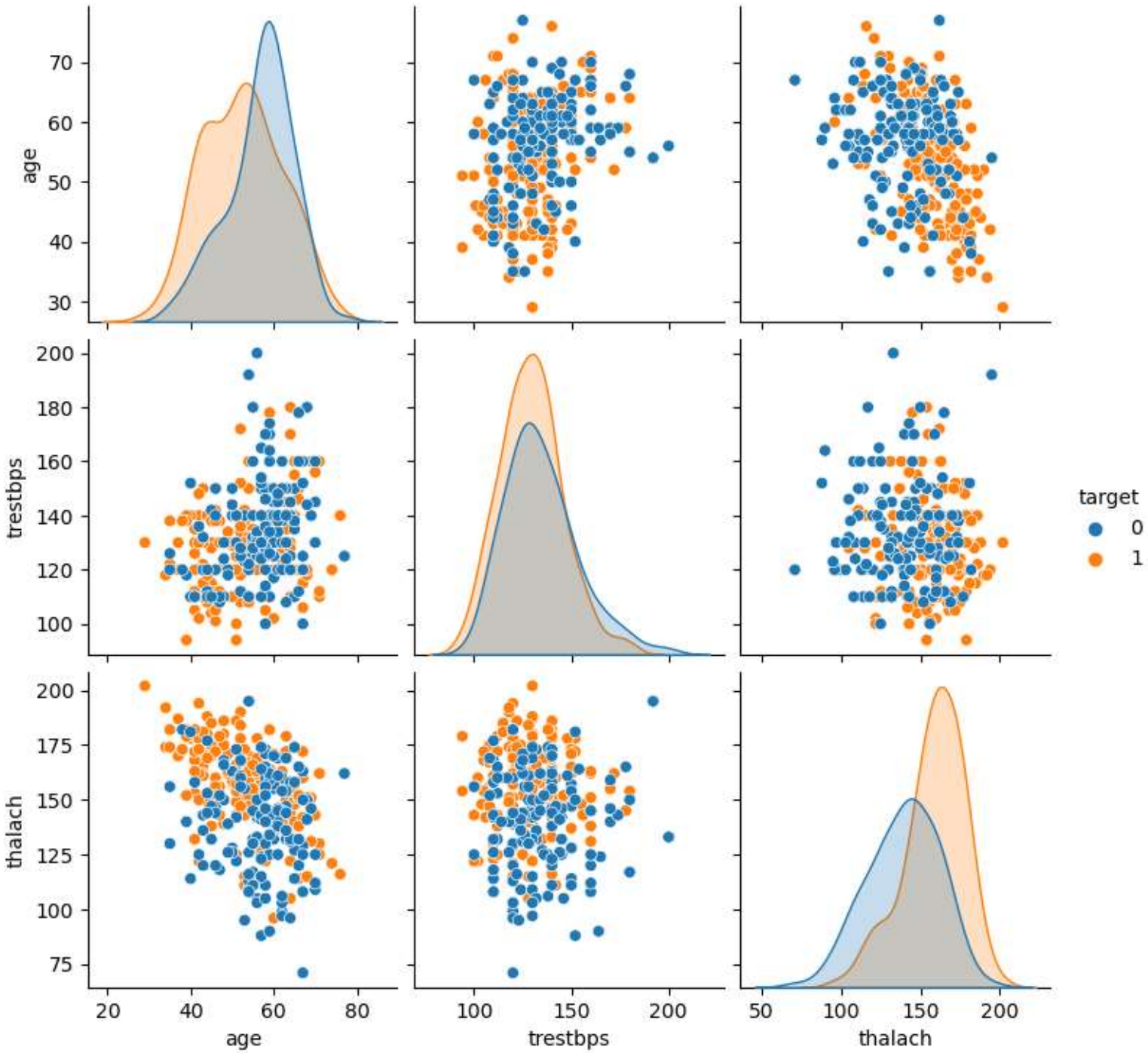
```
plt.figure(figsize=(9,8))

sns.heatmap(df.corr(),cmap='viridis',annot=True)
```



Clearly cp, thalach and slope are positively correlated with the label. Exang, oldpeak, ca and thal negatively. The rest are not correlated that well.

**sns.pairplot(df[['age','trestbps','thalach','target']],hue='target')**

**<seaborn.axisgrid.PairGrid at 0x256703e7d90>**



## Model Building

First, separate the data into two parts. The label (predicted value) is one part (y). Everything else is the second part (X).

### X = df.drop('target',axis=1)
**y = df['target']**

Second, split the data into training set and testing set. (Some explanation about train test split can be found in the Polynomial Regression case study,)

### from sklearn.model_selection import train_test_split
**from sklearn.preprocessing import StandardScaler**

### X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)

Third, normalize the training and testing X. When different columns have very different ranges, min and max values, it is highly recommended all of them to be scaled. For example, one feature might be measured in thousands, while another is measured in ones. Scaling refers to the process of transforming input features to a similar scale or range. Scaling is important because it can help improve the performance of many machine learning algorithms. The objective of this case study is to build a polynomial regression model. More information about the theory in scaling can be found on line.

**scaler = StandardScaler()**

**scaled_X_train = scaler.fit_transform(X_train)**
**scaled_X_test = scaler.transform(X_test)**

### Create a logistic Regression model and use cross-validation to find best hyper paremeter.

**from sklearn.linear_model import LogisticRegressionCV**

**log_model = LogisticRegressionCV()**

**log_model.fit(scaled_X_train,y_train)**

This are the parameters that the model found to give the best results.

**log_model.get_params()**

{'Cs': 10,
'class_weight': None,
'cv': None,
'dual': False,

'fit_intercept': True,
'intercept_scaling': 1.0,
'l1_ratios': None,
'max_iter': 100,
'multi_class': 'auto',
'n_jobs': None,
'penalty': 'l2',
'random_state': None,
'refit': True,
'scoring': None,
'solver': 'lbfgs',
'tol': 0.0001,
'verbose': 0}

It is beyond the scope of this paper to discuss all the parameters in this model. More information can be found in the sklearn documentation.

Coefficients of the model are as follows.

**log_model.coef_**

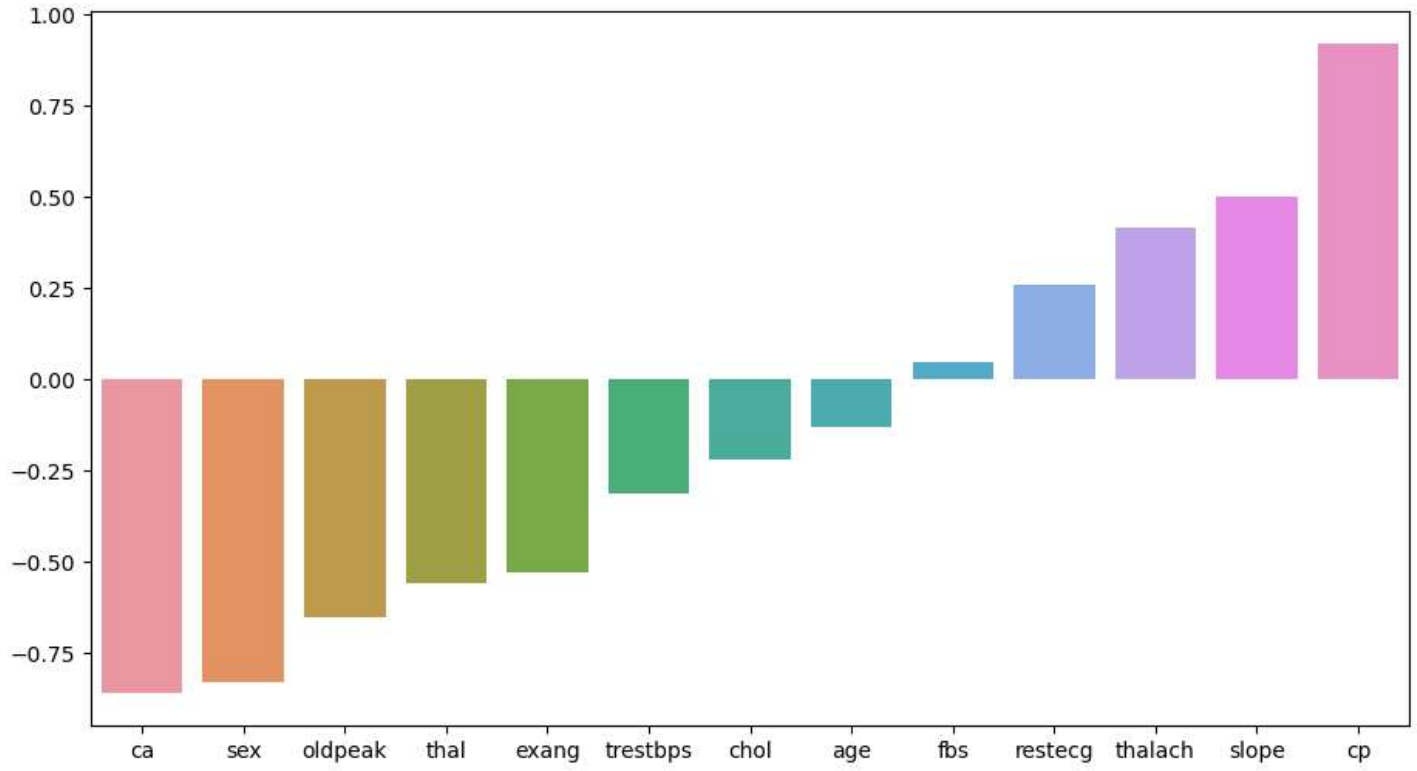
array([[-0.13302166, -0.82983661,  0.91833004, -0.31568785, -0.22018504,
        0.04466928,  0.25689156,  0.41657407, -0.53020481, -0.65248971,
        0.500581  , -0.86278117, -0.56212767]])

If we sort the value of the coefficients. It is very easy to spot which are very small, and eventually can be ignored. Also, it is possible to figure out which ones have positive which one have negative impact.

**coefs = pd.Series(index=X.columns,data=log_model.coef_[0])**

**coefs = coefs.sort_values()plt.figure(figsize=(9,6))**

**sns.barplot(x=coefs.index,y=coefs.values)**




## How accurate is our model?

Logical regression models are evaluated using the confusion matrix. A confusion matrix is a table that is used to evaluate the performance of a classification model. It summarizes the predictions made by a classifier on a dataset in terms of the actual classes of the data. The confusion matrix is particularly useful for understanding the types of errors made by the classifier.

Structure of the matrix:

**True Positive (TP)**: The classifier correctly predicted instances of the positive class.

**False Positive (FP)**: The classifier incorrectly predicted instances as belonging to the positive class when they actually belong to the negative class.

**True Negative (TN)**: The classifier correctly predicted instances of the negative class.

**False Negative (FN)**: The classifier incorrectly predicted instances as belonging to the negative class when they actually belong to the positive class.

**from sklearn.metrics import confusion_matrix,classification_report**

The data set that was set apart will be used to perform the testing. New prediction will be made using the X_test data. Then the predicted results will be compared to the actual y_test.

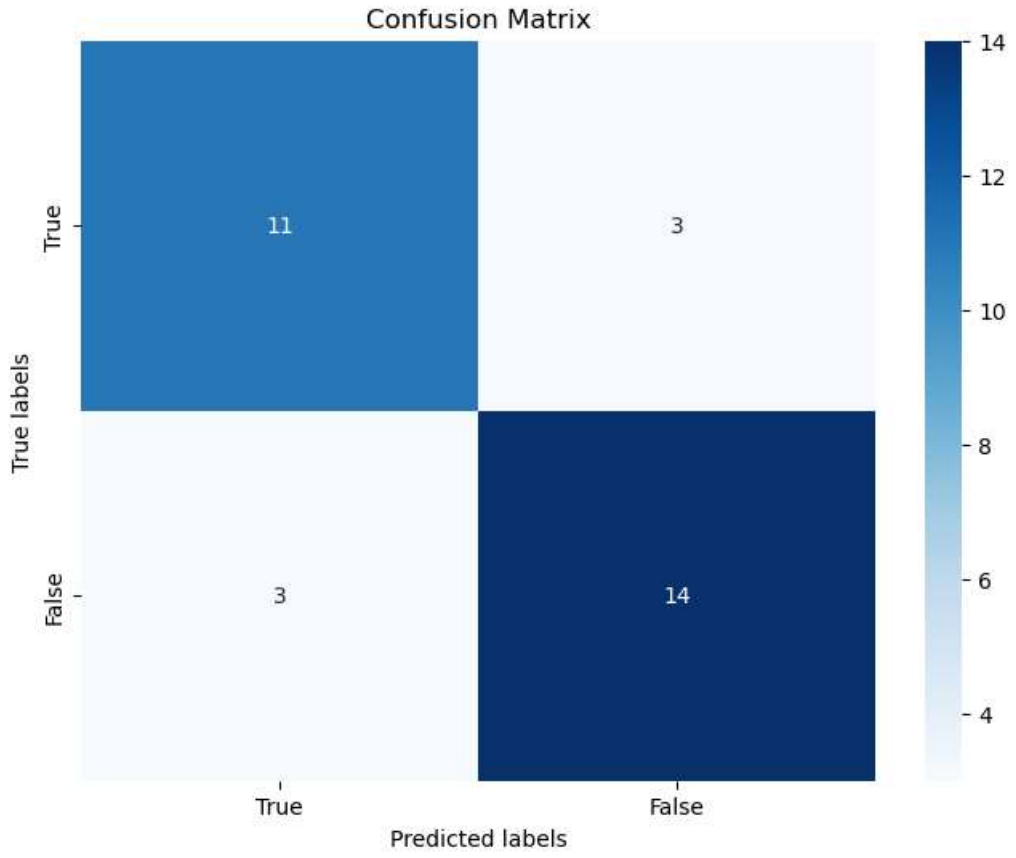
**y_pred = log_model.predict(scaled_X_test)**

**confusion_matrix(y_test,y_pred)**

array([[11,  3],
       [ 3, 14]],

Plotting the matrix

```
def plot_confusion_matrix(y_test,y_pred, labels):
    cm = confusion_matrix(y_test,y_pred)
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, cmap='Blues', xticklabels=labels, yticklabels=labels)
    plt.xlabel('Predicted labels')
    plt.ylabel('True labels')
    plt.title('Confusion Matrix')
    plt.show()
labels=['True', 'False']
plot_confusion_matrix(y_test,y_pred,labels)
```



The model predicted 11 as true out of 14 trues, 3 of the true labels are classified as false. Out of 17 false labels it identified 14 as false, and 3 as true.

There are several measures of accuracy for logical regression. We will print all values.

**print(classification_report(y_test,y_pred))**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.79 | 0.79 | 0.79 | 14 |
| 1 | 0.82 | 0.82 | 0.82 | 17 |
| accuracy | | | 0.81 | 31 |
| macro avg | 0.80 | 0.80 | 0.80 | 31 |
| weighted avg | 0.81 | 0.81 | 0.81 | 31 |

Accuracy: The proportion of correctly classified instances out of the total number of instances.
Precision: The proportion of true positive predictions out of all positive predictions made by the classifier.
Recall (Sensitivity): The proportion of true positive predictions out of all actual positive instances.
F1 Score: The harmonic mean of precision and recall, providing a balance between the two metrics.

It is very important to mention that sometimes the target label might be very imbalanced. For example, the number of attempts to carry gun or knife through are security gate might be only two attempts for 2000 people that went through the gate. In this case measures like **precision and recall** are much more important than accuracy. The objective is to catch these two individuals even thought we might misclassified 50 others and additionally inspected them.

## Plotting the receiver operating characteristics curve.

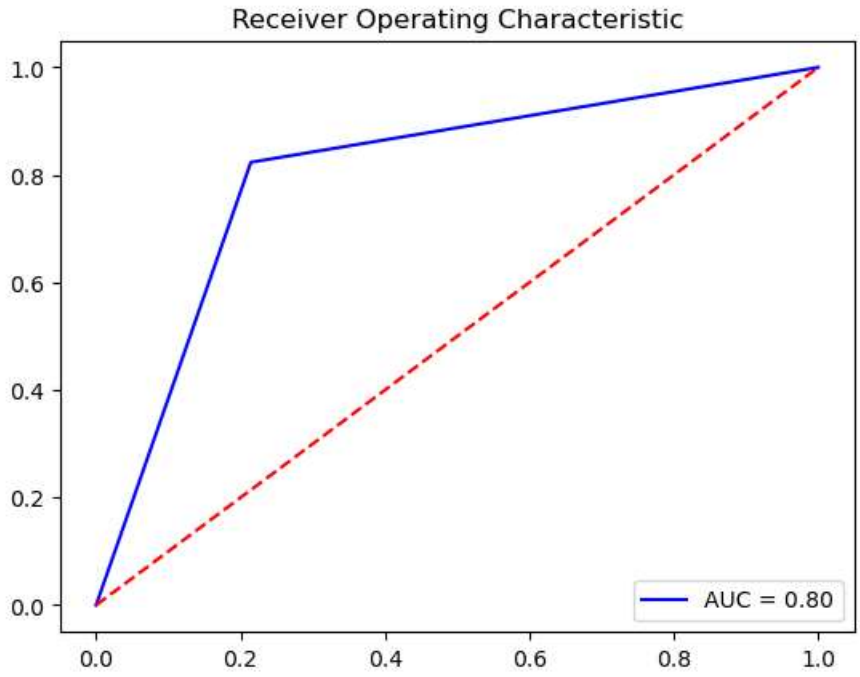
The ROC curve is a graphical representation that illustrates the performance of a binary classifier at various classification thresholds. It plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at different threshold settings.

Information about the theory and history of the receiver operated characteristics curve can be found in this Wikipedia article.

**import sklearn.metrics as metrics**
**# calculate the fpr and tpr for all thresholds of the classification**

**fpr, tpr, threshold = metrics.roc_curve(y_test, y_pred)**
**roc_auc = metrics.auc(fpr, tpr)**

```
import matplotlib.pyplot as plt
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
```



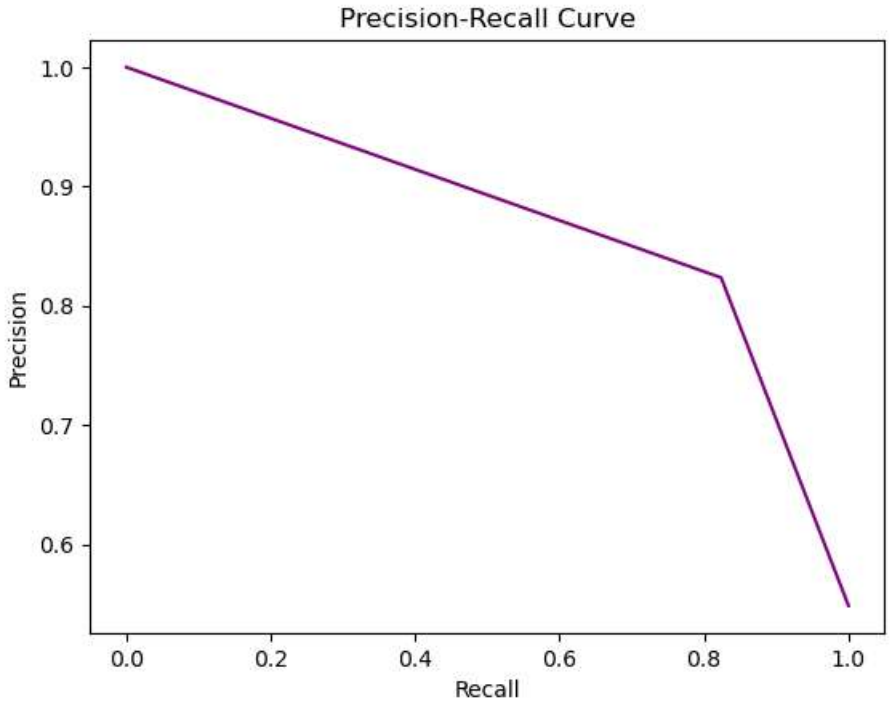

The closer the ROC curve is to the top-left corner of the plot, the better the classifier's performance. The area under the curve represents how accurate is our model. The higher (closer to 1) is the number the more accurate is the model. In this case AUC=.80.

## Plotting the precision recall curve.

The Precision-Recall (PR) curve is another common tool used to evaluate the performance of binary classifiers, particularly in situations where class imbalance exists or when the positive class is of greater interest. It plots the precision against the recall at various classification thresholds. It provides a more informative evaluation in scenarios where the focus is on correctly identifying positive instances (e.g., disease diagnosis, fraud detection).

```
from sklearn import datasets
from sklearn.metrics import precision_recall_curve

precision, recall, thresholds = precision_recall_curve(y_test, y_pred)

fig, ax = plt.subplots()
ax.plot(recall, precision, color='purple')
ax.set_title('Precision-Recall Curve')
```

**ax.set_ylabel('Precision')**
**ax.set_xlabel('Recall')**
**plt.show()**




The closer the PR curve is to the top-right corner of the plot, the better the classifier's performance.

## Model prediction.

Suppose a patient comes to cardiologist office with blood and physical test results. How we can find out if the patient has a heart disease.

For simplicity lets assume the patient has the data in the first raw of our data set.

**test1=[[63,1,3,145,233,1,0,150,0,2.3,0,0,1]]**

The input data needs to be scaled with the same scaler with have in the model preparation.

**scaled_test1 = scaler.transform(test1)**

**log_model.predict(scaled_test1)[0]**

## Result = 1 ***The Patient has a heart disease.***
Correct!!

Bellow I have provided more data from the original data set.

**print(df)**

```
      age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0      63    1   3       145   233    1        0      150      0      2.3
1      37    1   2       130   250    0        1      187      0      3.5
2      41    0   1       130   204    0        0      172      0      1.4
3      56    1   1       120   236    0        1      178      0      0.8
4      57    0   0       120   354    0        1      163      1      0.6
..    ...  ...  ..       ...   ...  ...      ...      ...    ...      ...
298    57    0   0       140   241    0        1      123      1      0.2
299    45    1   3       110   264    0        1      132      0      1.2
300    68    1   0       144   193    1        1      141      0      3.4
301    57    1   0       130   131    0        1      115      1      1.2
302    57    0   1       130   236    0        0      174      0      0.0

     slope  ca  thal  target
0        0   0     1       1
1        0   0     2       1
2        2   0     2       1
3        2   0     2       1
4        2   0     2       1
..     ...  ..   ...     ...
298      1   0     3       0
299      1   0     3       0
300      1   2     3       0
301      1   1     3       0
302      1   1     2       0

[303 rows x 14 columns]
```

The table bellow describes the input parameters. More info about this data set and study can be also found in the following link.

| Variable Name | Role | Type | Demographic | Description | Units | Missing Values |
|---|---|---|---|---|---|---|
| age | Feature | Integer | Age | | years | no |
| sex | Feature | Categorical | Sex | | | no |
| cp | Feature | Categorical | | | | no |
| trestbps | Feature | Integer | | resting blood pressure (on admission to the hospital) | mm Hg | no |
| chol | Feature | Integer | | serum cholestoral | mg/dl | no |
| fbs | Feature | Categorical | | fasting blood sugar > 120 mg/dl | | no |
| restecg | Feature | Categorical | | | | no |
| thalach | Feature | Integer | | maximum heart rate achieved | | no |
| exang | Feature | Categorical | | exercise induced angina | | no |
| oldpeak | Feature | Integer | | ST depression induced by exercise relative to rest | | no |
| slope | Feature | Categorical | | | | no |
| ca | Feature | Integer | | number of major vessels (0-3) colored by flourosopy | | yes |

| Variable Name | Role | Type | Demographic | Description | Units | Missing Values |
|---|---|---|---|---|---|---|
| thal | Feature | Categorical | | | | yes |
| num | Target | Integer | | diagnosis of heart disease<br>0- no heart disease, 1 – existence of heart disease. | | |