**Logistic Regression**: The target variable has three or more nominal categories such as predicting the type of Wine. Ordinal Logistic Regression: the target variable has three or more ordinal categories such as restaurant or product rating from 1 to 5. Model building in Scikit-learn Let's build the diabetes prediction model.

Here, We are going to predict Coronary Heart Disease using Logistic Regression Classifier.

Let's first load the required Coronary Heart Disease dataset using the pandas' read CSV function.

We will download data from the following link: <https://www.kaggle.com/datasets/billbasener/coronary-heart-disease?resource=download>

from google.colab import drive  
  
drive.mount('/content/gdrive')

Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force\_remount=True).

cd /content/gdrive/MyDrive/Machine Learning/Colab Notebooks/ML\_Practicals/1\_Practical/Logistic regression P2

/content/gdrive/MyDrive/Machine Learning/Colab Notebooks/ML\_Practicals/1\_Practical/Logistic regression P2

ls

CHDdata.csv CHD\_Data.csv CHDdata.gsheet Logistic\_Regression.ipynb

import pandas as pd  
col\_names = ['Systolic BP', 'Tobacco', 'low-density lipoprotein', 'Adiposity', 'Famhist', 'typea', 'Obesity', 'Alcohol', 'Age', 'Chd']  
# load dataset  
CHD = pd.read\_csv("CHD\_Data.csv", header=None, names=col\_names)

**Context**

The data set CHDdata.csv contains cases of coronary heart disease (CHD) and variables associated with the patient's condition: systolic blood pressure, yearly tobacco use (in kg), low density lipoprotein (Idl), adiposity, family history (0 or 1), type A personality score (typea), obesity (body mass index), alcohol use, age, and the diagnosis of CHD (0 or 1).

CHD.head()

Systolic BP Tobacco low-density lipoprotein Adiposity Famhist typea \  
1 160 12.00 5.73 23.11 Present 49   
2 144 0.01 4.41 28.61 Absent 55   
3 118 0.08 3.48 32.28 Present 52   
4 170 7.50 6.41 38.03 Present 51   
5 134 13.60 3.50 27.78 Present 60   
  
 Obesity Alcohol Age Chd   
1 25.30 97.20 52 1   
2 28.87 2.06 63 1   
3 29.14 3.81 46 0   
4 31.99 24.26 58 1   
5 25.99 57.34 49 1

CHD.tail()

Systolic BP Tobacco low-density lipoprotein Adiposity Famhist typea \  
458 214 0.4 5.98 31.72 Absent 64   
459 182 4.2 4.41 32.10 Absent 52   
460 108 3.0 1.59 15.23 Absent 40   
461 118 5.4 11.61 30.79 Absent 64   
462 132 0.0 4.82 33.41 Present 62   
  
 Obesity Alcohol Age Chd   
458 28.45 0.00 58 0   
459 28.61 18.72 52 1   
460 20.09 26.64 55 0   
461 27.35 23.97 40 0   
462 14.70 0.00 46 1

CHD.drop('Famhist', inplace=True, axis=1)

**Selecting Feature** Here, we need to divide the given columns into two types of variables dependent(or target variable) and independent variable(or feature variables).

#split dataset in features and target variable  
feature\_cols = ['Systolic BP', 'Tobacco', 'low-density lipoprotein', 'Adiposity', 'typea', 'Obesity', 'Alcohol', 'Age', ]  
X = CHD[feature\_cols] # Features  
y = CHD.Chd # Target variable

**Splitting Data** To understand model performance, dividing the dataset into a training set and a test set is a good strategy.

Let's split dataset by using function train\_test\_split(). We need to pass 3 parameters features, target, and test\_set size. Additionally, We can use random\_state to select records randomly.

# split X and y into training and testing sets  
from sklearn.model\_selection import train\_test\_split  
X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.65,random\_state=0)

Here, the Dataset is broken into two parts in a ratio of 75:25. It means 75% data will be used for model training and 25% for model testing.

**Model Development and Prediction** First, import the Logistic Regression module and create a Logistic Regression classifier object using LogisticRegression() function.

Then, fit our model on the train set using fit() and perform prediction on the test set using predict().

#import the class  
from sklearn.linear\_model import LogisticRegression  
  
# instantiate the model (using the default parameters)  
logreg = LogisticRegression()  
  
# fit the model with data  
logreg.fit(X\_train,y\_train)  
  
#  
y\_pred=logreg.predict(X\_test)

/usr/local/lib/python3.7/dist-packages/sklearn/linear\_model/\_logistic.py:818: ConvergenceWarning: lbfgs failed to converge (status=1):  
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.  
  
Increase the number of iterations (max\_iter) or scale the data as shown in:  
 https://scikit-learn.org/stable/modules/preprocessing.html  
Please also refer to the documentation for alternative solver options:  
 https://scikit-learn.org/stable/modules/linear\_model.html#logistic-regression  
 extra\_warning\_msg=\_LOGISTIC\_SOLVER\_CONVERGENCE\_MSG,

# **Model Evaluation using Confusion Matrix**

A confusion matrix is a table that is used to evaluate the performance of a classification model. We can also visualize the performance of an algorithm. The fundamental of a confusion matrix is the number of correct and incorrect predictions are summed up class-wise.

# import the metrics class  
from sklearn import metrics  
cnf\_matrix = metrics.confusion\_matrix(y\_test, y\_pred)  
cnf\_matrix

array([[174, 28],  
 [ 56, 43]])

Here, we can see the confusion matrix in the form of the array object. The dimension of this matrix is 2\*2 because this model is binary classification. We have two classes 0 and 1. Diagonal values represent accurate predictions, while non-diagonal elements are inaccurate predictions. In the output, 174 and 28 are actual predictions, and 56 and 43 are incorrect predictions.

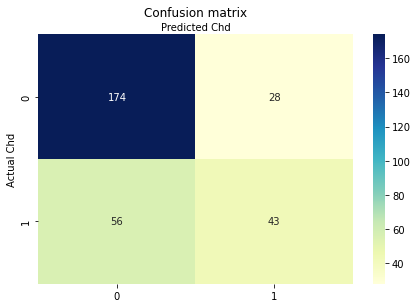
Visualizing Confusion Matrix using Heatmap Let's visualize the results of the model in the form of a confusion matrix using matplotlib and seaborn.

Here, we will visualize the confusion matrix using Heatmap.

# import required modules  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
%matplotlib inline

class\_names=[0,1] # name of classes  
fig, ax = plt.subplots()  
tick\_marks = np.arange(len(class\_names))  
plt.xticks(tick\_marks, class\_names)  
plt.yticks(tick\_marks, class\_names)  
# create heatmap  
sns.heatmap(pd.DataFrame(cnf\_matrix), annot=True, cmap="YlGnBu" ,fmt='g')  
ax.xaxis.set\_label\_position("top")  
plt.tight\_layout()  
plt.title('Confusion matrix', y=1.1)  
plt.ylabel('Actual Chd')  
plt.xlabel('Predicted Chd')

Text(0.5, 257.44, 'Predicted Chd')



**Confusion Matrix Evaluation Metrics** Let's evaluate the model using model evaluation metrics such as accuracy, precision, and recall.

print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))  
print("Precision:",metrics.precision\_score(y\_test, y\_pred))  
print("Recall:",metrics.recall\_score(y\_test, y\_pred))

Accuracy: 0.7209302325581395  
Precision: 0.6056338028169014  
Recall: 0.43434343434343436

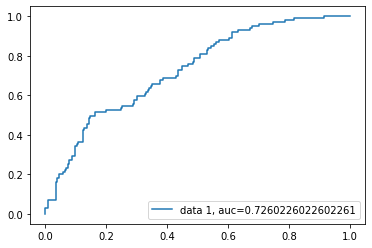
Well, we got a classification rate of 72%, considered as good accuracy.

**Precision:** Precision is about being precise, i.e., how accurate our model is. In other words, we can say, when a model makes a prediction, how often it is correct. In our prediction case, when our Logistic Regression model predicted patients are going to suffer from diabetes, that patients have 60% of the time.

**Recall:** If there are patients who have diabetes in the test set and our Logistic Regression model can identify it 43% of the time.

**ROC Curve** Receiver Operating Characteristic(ROC) curve is a plot of the true positive rate against the false positive rate. It shows the tradeoff between sensitivity and specificity.

y\_pred\_proba = logreg.predict\_proba(X\_test)[::,1]  
fpr, tpr, \_ = metrics.roc\_curve(y\_test, y\_pred\_proba)  
auc = metrics.roc\_auc\_score(y\_test, y\_pred\_proba)  
plt.plot(fpr,tpr,label="data 1, auc="+str(auc))  
plt.legend(loc=4)  
plt.show()



# AUC score for the case is 0.73. AUC score 1 represents perfect classifier, and 0.5 represents a worthless classifier.

**Conclusion** In this Notebook we were able to measure and evaluate the Accuracy, Recall, Precision of the data thoroughly.