21MAT204 MATHEMATICS FOR INTELLIGENT SYSTEMS-3

HEART DISEASE PREDICTION USING LOGISTIC REGRESSION

PROJECT REPORT

Submitted to: Dr. NEETHU MOHAN

By ,
NAME : j. sai chandana
ROLL N.O : cb.en.u4aie21118
BATCH : B

ACKNOWLEDGEMENTS : -

We would like to thank Dr. Neethu Mohan, our professor-in-charge, for her support and guidance in classes and for helping us in completing our project on the topic. It was a great learning experience.

TABLE OF CONTENTS:

HEART DISEASE :

- A buildup of fatty plaques in the arteries (atherosclerosis) is the most common cause of coronary artery disease. Risk factors include a poor diet, lack of exercise, obesity and smoking. Healthy lifestyle choices can help lower the risk of atherosclerosis

- Chest pain, chest tightness, chest pressure and chest discomfort (angina) Shortness of breath. Pain in the neck, jaw, throat, upper belly area or back. Pain, numbness, weakness or coldness in the legs or arms if the blood vessels in those body areas are narrowed.

- Heart failure can be ongoing (chronic), or it may start suddenly (acute)

- The number of deaths due to heart attacks in India has remained consistently over 25,000 in the last four years, and over 28,000 in the last three year.

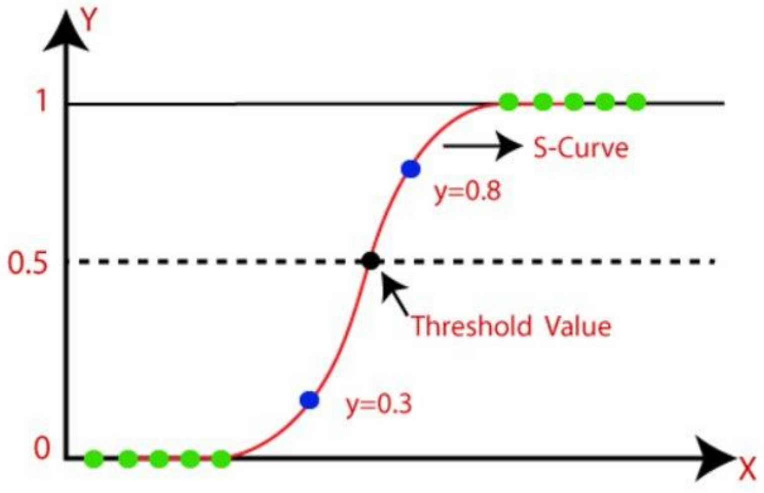
BASED ON THE FEAUTURES WE NEED TO CHECK WETHER THE PERSON IS HAVING HEART DISEASE OR NOT

IT HELPS THE DOCTORS TO TREAT THE PATIENTS AND DIAGNOISE THEM .

FOR THIS WE USE THE MACHINE LEARNING
ALGORITHM CALLED LOGISTIC REGRESSION

LOGISTIC- REGRESSION :-

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique.
It is used for predicting the categorical dependent variable using a given set of independent variables.
In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).



- he sigmoid function is a mathematical function used to map the predicted values to probabilities.
- It maps any real value into another value within a range of 0 and 1.

- The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form.

- The S-form curve is called the Sigmoid function or the logistic function.

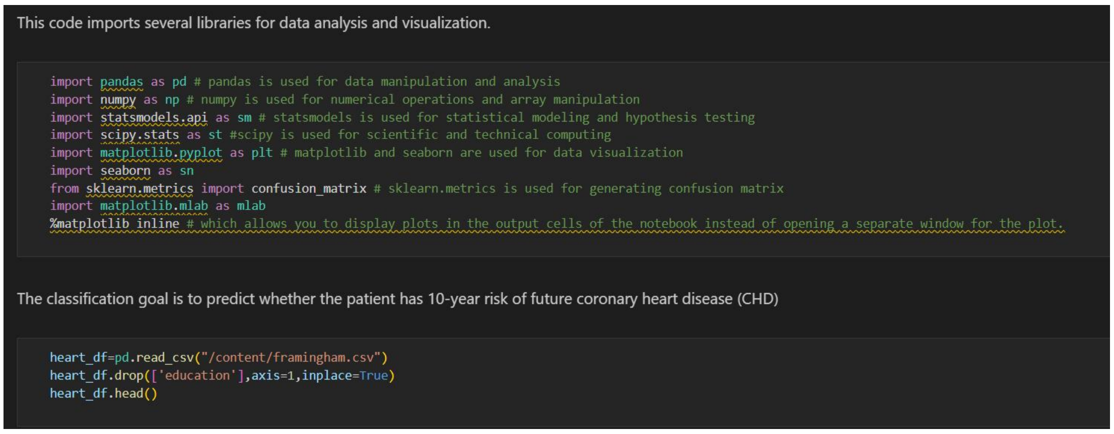
- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

Features used : -

1.age: The person's age in years
2. sex: The person's sex (1 = male, 0 = female
3. cp: The chest pain experienced (Value 1: typical angina, Value 2: atypical angina, Value 3: non-anginal pain, Value 4: asymptomatic)
4. trestbps: The person's resting blood pressure (mm Hg on admission to the hospital)
5. chol: The person's cholesterol measurement in mg/dl
6. fbs: The person's fasting blood sugar (> 120 mg/dl, 1 = true; 0 = false)
7. restecg: Resting electrocardiographic measurement (0 = normal, 1 = having ST-T wave abnormality, 2 = showing probable or definite left ventricular hypertrophy by Estes' criteria)
8. thalach: The person's maximum heart rate achieved
9. exang: Exercise induced angina (1 = yes; 0 = no)
10. oldpeak: ST depression induced by exercise relative to rest ('ST' relates to positions on the ECG plot. See more here)
11. slope: the slope of the peak exercise ST segment (Value 1: upsloping, Value 2: flat, Value 3: downsloping)
12. ca: The number of major vessels (0-3)
13. thal: A blood disorder called thalassemia (3 = normal; 6 = fixed defect; 7 = reversable defect)
14. target: Heart disease (0 = no, 1 = yes)

Heart disease risk factors to the following: high cholesterol, high blood pressure, diabetes, weight, family history and smoking .
According to another source , the major factors that can't be changed are: increasing age, male gender and heredity.
Note that thalassemia, one of the variables in this dataset, is heredity.
Major factors that can be modified are: Smoking, high cholesterol, high blood pressure, physical inactivity, and being overweight and having diabetes.
Other factors include stress, alcohol and poor diet/nutrition.

Code screen shots : -

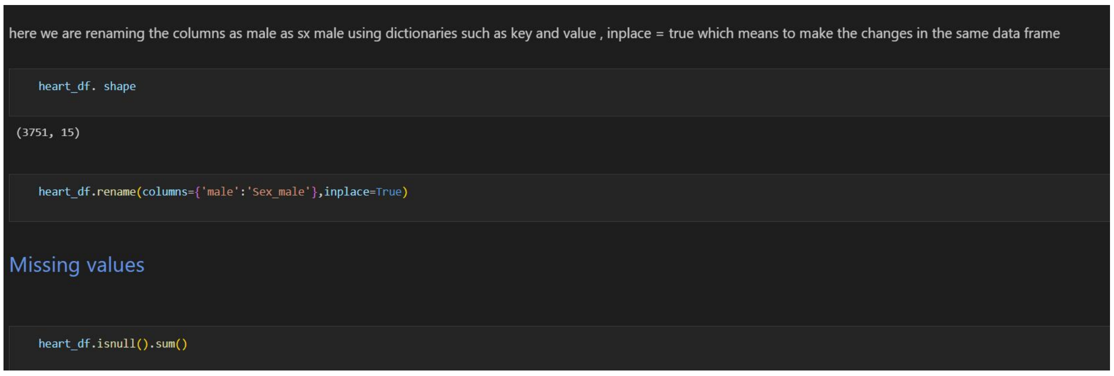


This code imports several libraries for data analysis and visualization.

```
import pandas as pd # pandas is used for data manipulation and analysis
import numpy as np # numpy is used for numerical operations and array manipulation
import statsmodels.api as sm # statsmodels is used for statistical modeling and hypothesis testing
import scipy.stats as st #scipy is used for scientific and technical computing
import matplotlib.pyplot as plt # matplotlib and seaborn are used for data visualization
import seaborn as sn
from sklearn.metrics import confusion_matrix # sklearn.metrics is used for generating confusion matrix
import matplotlib.mlab as mlab
%matplotlib inline # which allows you to display plots in the output cells of the notebook instead of opening a separate window for the plot.
```

The classification goal is to predict whether the patient has 10-year risk of future coronary heart disease (CHD)

```
heart_df=pd.read_csv("/content/framingham.csv")
heart_df.drop(['education'],axis=1,inplace=True)
heart_df.head()
```



here we are renaming the columns as male as sx male using dictionaries such as key and value , inplace = true which means to make the changes in the same data frame

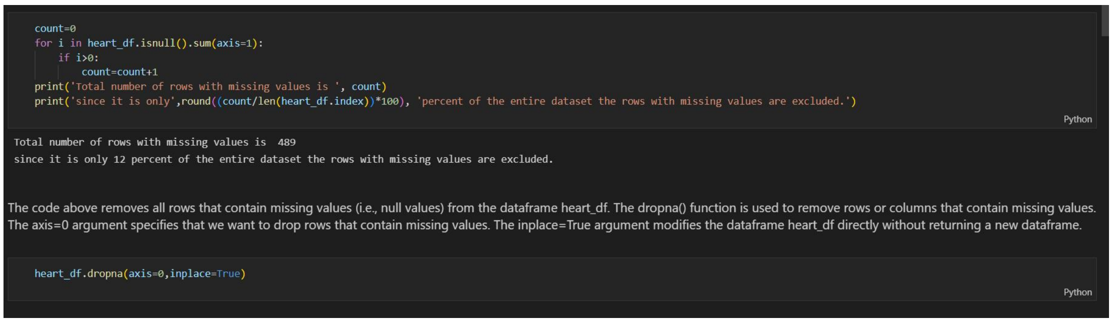
```
heart_df. shape
```

```
(3751, 15)
```

```
heart_df.rename(columns={'male':'Sex_male'},inplace=True)
```

Missing values

```
heart_df.isnull().sum()
```

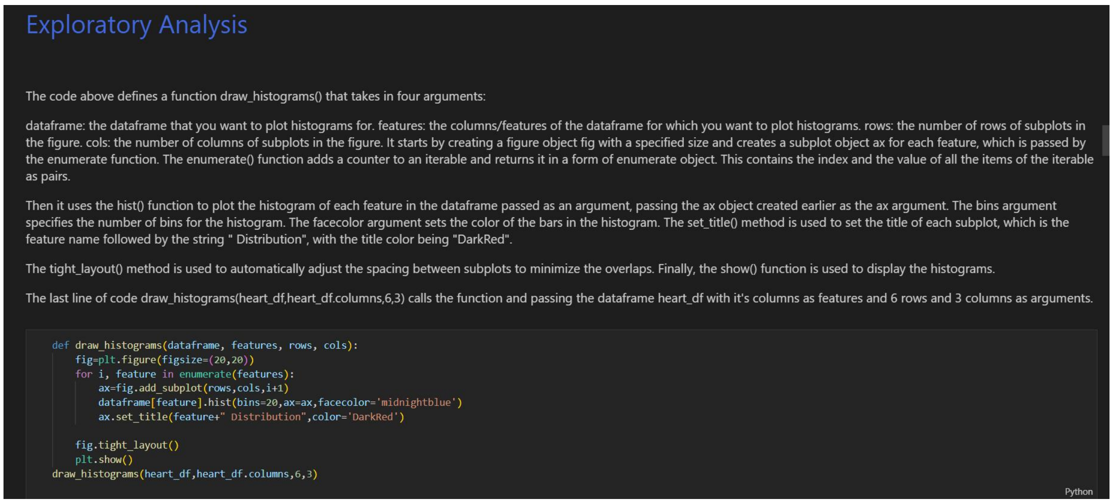


```
count=0
for i in heart_df.isnull().sum(axis=1):
    if i>0:
        count=count+1
print('Total number of rows with missing values is ', count)
print('since it is only',round((count/len(heart_df.index))*100), 'percent of the entire dataset the rows with missing values are excluded.')
```

```
Total number of rows with missing values is  489
since it is only 12 percent of the entire dataset the rows with missing values are excluded.
```

The code above removes all rows that contain missing values (i.e., null values) from the dataframe heart_df. The dropna() function is used to remove rows or columns that contain missing values. The axis=0 argument specifies that we want to drop rows that contain missing values. The inplace=True argument modifies the dataframe heart_df directly without returning a new dataframe.

```
heart_df.dropna(axis=0,inplace=True)
```



Exploratory Analysis

The code above defines a function draw_histograms() that takes in four arguments:

dataframe: the dataframe that you want to plot histograms for. features: the columns/features of the dataframe for which you want to plot histograms. rows: the number of rows of subplots in the figure. cols: the number of columns of subplots in the figure. It starts by creating a figure object fig with a specified size and creates a subplot object ax for each feature, which is passed by the enumerate function. The enumerate() function adds a counter to an iterable and returns it in a form of enumerate object. This contains the index and the value of all the items of the iterable as pairs.

Then it uses the hist() function to plot the histogram of each feature in the dataframe passed as an argument, passing the ax object created earlier as the ax argument. The bins argument specifies the number of bins for the histogram. The facecolor argument sets the color of the bars in the histogram. The set_title() method is used to set the title of each subplot, which is the feature name followed by the string " Distribution", with the title color being "DarkRed".

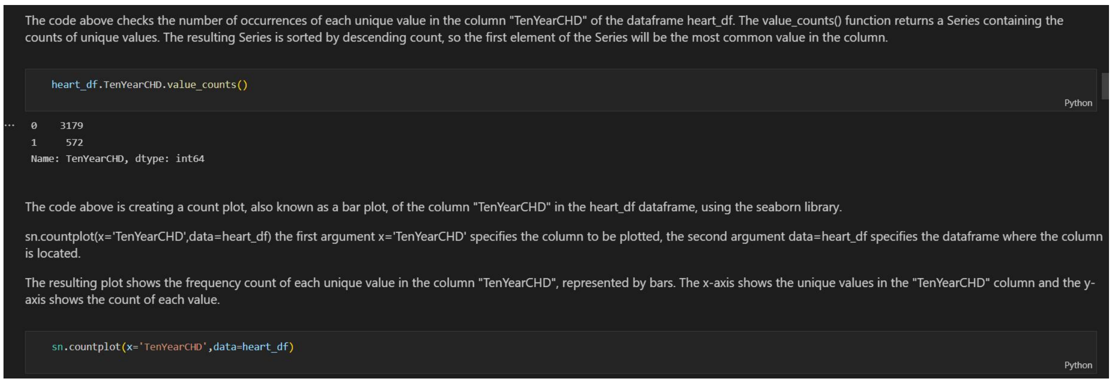
The tight_layout() method is used to automatically adjust the spacing between subplots to minimize the overlaps. Finally, the show() function is used to display the histograms.

The last line of code draw_histograms(heart_df,heart_df.columns,6,3) calls the function and passing the dataframe heart_df with it's columns as features and 6 rows and 3 columns as arguments.

```python
def draw_histograms(dataframe, features, rows, cols):
    fig=plt.figure(figsize=(20,20))
    for i, feature in enumerate(features):
        ax=fig.add_subplot(rows,cols,i+1)
        dataframe[feature].hist(bins=20,ax=ax,facecolor='midnightblue')
        ax.set_title(feature+" Distribution",color='DarkRed')

    fig.tight_layout()
    plt.show()
draw_histograms(heart_df,heart_df.columns,6,3)
```

Python



The code above checks the number of occurrences of each unique value in the column "TenYearCHD" of the dataframe heart_df. The value_counts() function returns a Series containing the counts of unique values. The resulting Series is sorted by descending count, so the first element of the Series will be the most common value in the column.

```python
heart_df.TenYearCHD.value_counts()
```

Python

```
0    3179
1     572
Name: TenYearCHD, dtype: int64
```

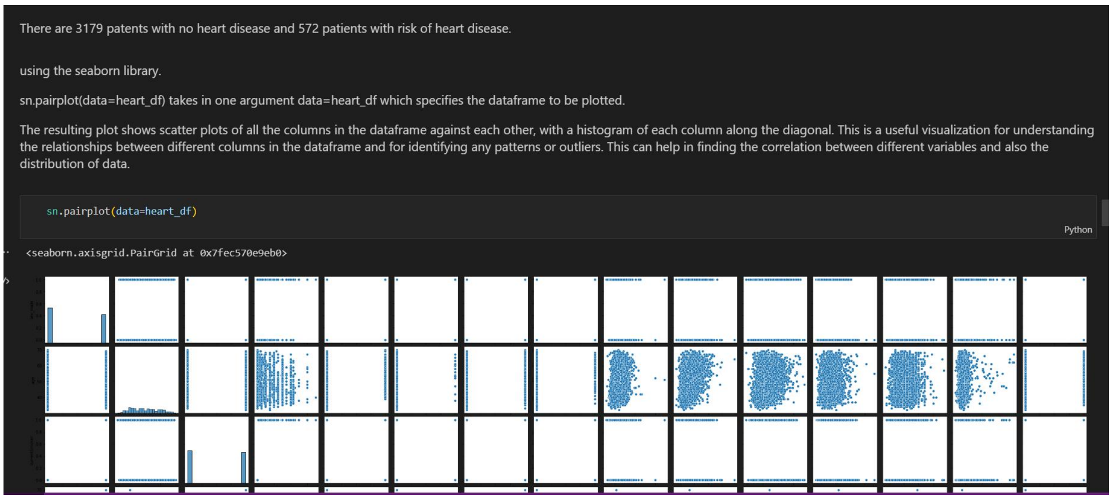
The code above is creating a count plot, also known as a bar plot, of the column "TenYearCHD" in the heart_df dataframe, using the seaborn library.

sn.countplot(x='TenYearCHD',data=heart_df) the first argument x='TenYearCHD' specifies the column to be plotted, the second argument data=heart_df specifies the dataframe where the column is located.

The resulting plot shows the frequency count of each unique value in the column "TenYearCHD", represented by bars. The x-axis shows the unique values in the "TenYearCHD" column and the y-axis shows the count of each value.

```python
sn.countplot(x='TenYearCHD',data=heart_df)
```

Python



There are 3179 patents with no heart disease and 572 patients with risk of heart disease.

using the seaborn library.

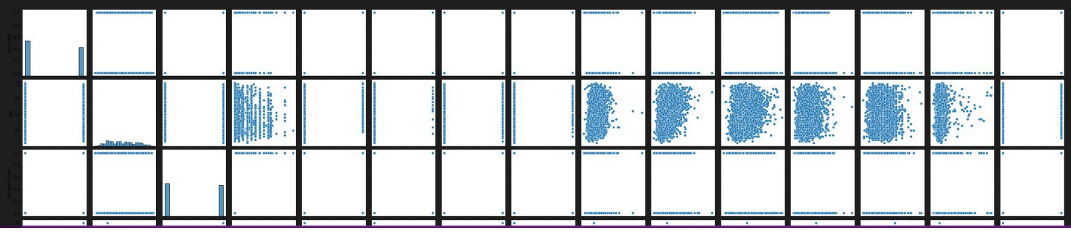
sn.pairplot(data=heart_df) takes in one argument data=heart_df which specifies the dataframe to be plotted.

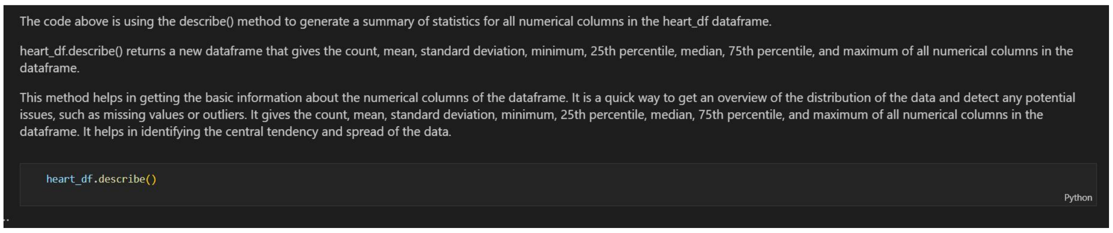
The resulting plot shows scatter plots of all the columns in the dataframe against each other, with a histogram of each column along the diagonal. This is a useful visualization for understanding the relationships between different columns in the dataframe and for identifying any patterns or outliers. This can help in finding the correlation between different variables and also the distribution of data.

```python
sn.pairplot(data=heart_df)
```

Python

```
<seaborn.axisgrid.PairGrid at 0x7fec570e9eb0>
```



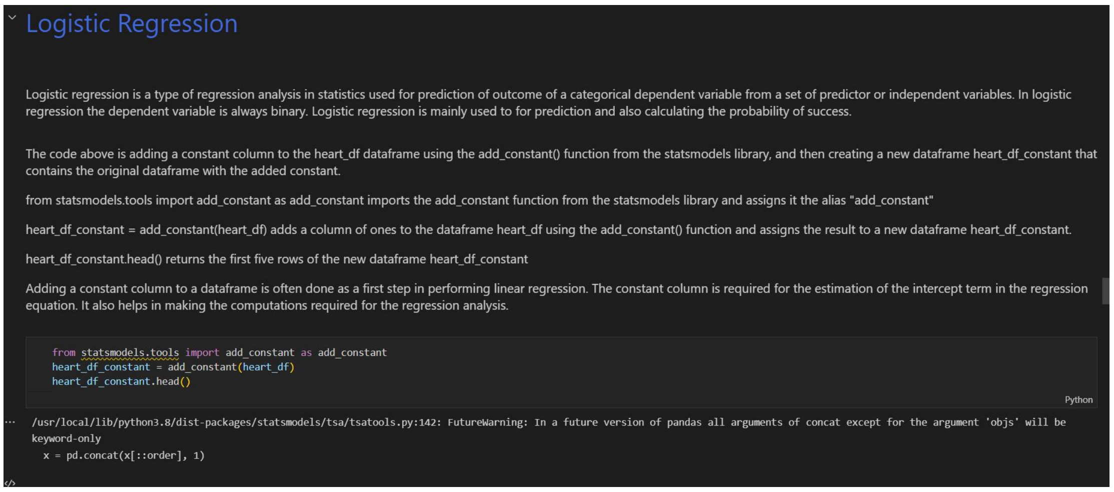


The code above is using the describe() method to generate a summary of statistics for all numerical columns in the heart_df dataframe.

heart_df.describe() returns a new dataframe that gives the count, mean, standard deviation, minimum, 25th percentile, median, 75th percentile, and maximum of all numerical columns in the dataframe.

This method helps in getting the basic information about the numerical columns of the dataframe. It is a quick way to get an overview of the distribution of the data and detect any potential issues, such as missing values or outliers. It gives the count, mean, standard deviation, minimum, 25th percentile, median, 75th percentile, and maximum of all numerical columns in the dataframe. It helps in identifying the central tendency and spread of the data.

```python
heart_df.describe()
```



Logistic Regression

Logistic regression is a type of regression analysis in statistics used for prediction of outcome of a categorical dependent variable from a set of predictor or independent variables. In logistic regression the dependent variable is always binary. Logistic regression is mainly used to for prediction and also calculating the probability of success.

The code above is adding a constant column to the heart_df dataframe using the add_constant() function from the statsmodels library, and then creating a new dataframe heart_df_constant that contains the original dataframe with the added constant.

from statsmodels.tools import add_constant as add_constant imports the add_constant function from the statsmodels library and assigns it the alias "add_constant"

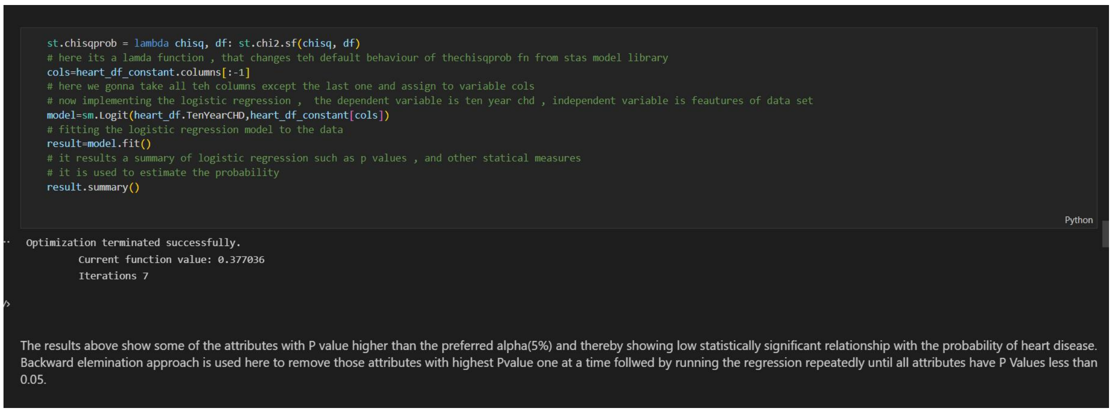
heart_df_constant = add_constant(heart_df) adds a column of ones to the dataframe heart_df using the add_constant() function and assigns the result to a new dataframe heart_df_constant.

heart_df_constant.head() returns the first five rows of the new dataframe heart_df_constant

Adding a constant column to a dataframe is often done as a first step in performing linear regression. The constant column is required for the estimation of the intercept term in the regression equation. It also helps in making the computations required for the regression analysis.

```python
from statsmodels.tools import add_constant as add_constant
heart_df_constant = add_constant(heart_df)
heart_df_constant.head()
```

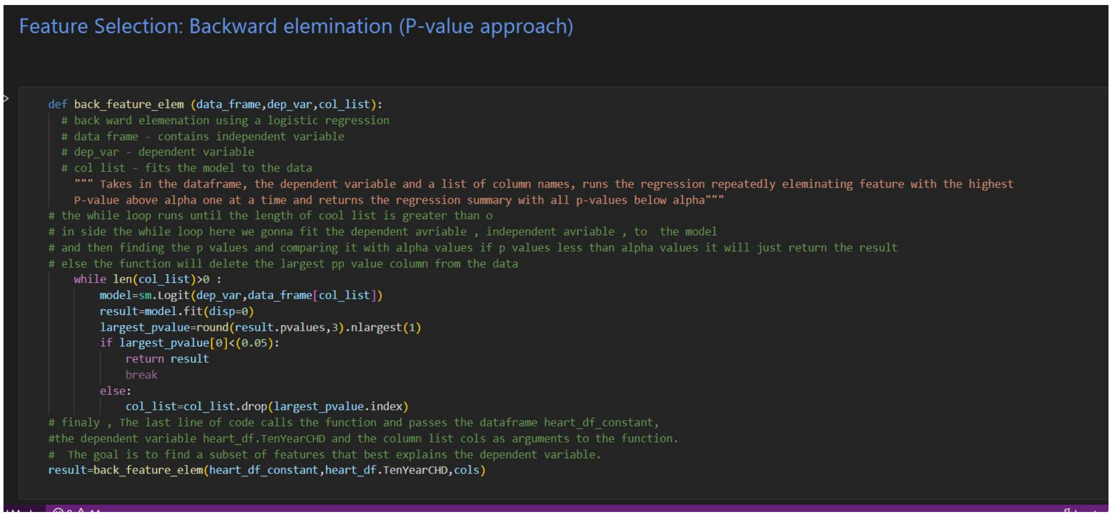
```
/usr/local/lib/python3.8/dist-packages/statsmodels/tsa/tsatools.py:142: FutureWarning: In a future version of pandas all arguments of concat except for the argument 'objs' will be
keyword-only
  x = pd.concat(x[::order], 1)
```



```python
st.chisqprob = lambda chisq, df: st.chi2.sf(chisq, df)
# here its a lamda function , that changes teh default behaviour of thechisqprob fn from stas model library
cols=heart_df_constant.columns[:-1]
# here we gonna take all teh columns except the last one and assign to variable cols
# now implementing the logistic regression ,  the dependent variable is ten year chd , independent variable is feautures of data set
model=sm.Logit(heart_df.TenYearCHD,heart_df_constant[cols])
# fitting the logistic regression model to the data
result=model.fit()
# it results a summary of logistic regression such as p values , and other statical measures
# it is used to estimate the probability
result.summary()
```

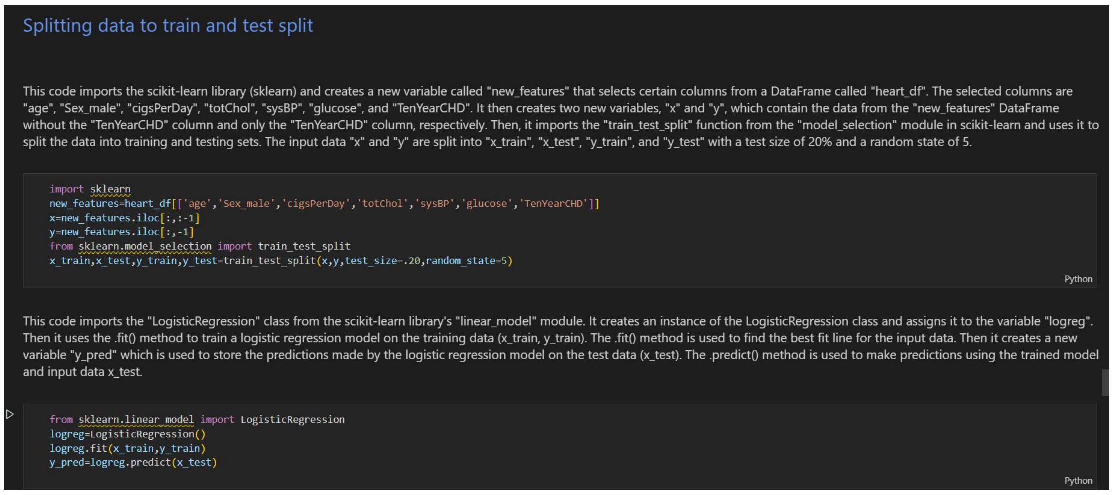
```
Optimization terminated successfully.
         Current function value: 0.377036
         Iterations 7
```

The results above show some of the attributes with P value higher than the preferred alpha(5%) and thereby showing low statistically significant relationship with the probability of heart disease. Backward elemination approach is used here to remove those attributes with highest Pvalue one at a time follwed by running the regression repeatedly until all attributes have P Values less than 0.05.



Feature Selection: Backward elemination (P-value approach)

```python
def back_feature_elem (data_frame,dep_var,col_list):
  # back ward elemenation using a logistic regression
  # data frame - contains independent variable
  # dep_var - dependent variable
  # col list - fits the model to the data
    """ Takes in the dataframe, the dependent variable and a list of column names, runs the regression repeatedly eleminating feature with the highest
    P-value above alpha one at a time and returns the regression summary with all p-values below alpha"""
# the while loop runs until the length of cool list is greater than o
# in side the while loop here we gonna fit the dependent avriable , independent avriable , to  the model
# and then finding the p values and comparing it with alpha values if p values less than alpha values it will just return the result
# else the function will delete the largest pp value column from the data
    while len(col_list)>0 :
        model=sm.Logit(dep_var,data_frame[col_list])
        result=model.fit(disp=0)
        largest_pvalue=round(result.pvalues,3).nlargest(1)
        if largest_pvalue[0]<(0.05):
            return result
            break
        else:
            col_list=col_list.drop(largest_pvalue.index)
# finaly , The last line of code calls the function and passes the dataframe heart_df_constant,
#the dependent variable heart_df.TenYearCHD and the column list cols as arguments to the function.
#  The goal is to find a subset of features that best explains the dependent variable.
result=back_feature_elem(heart_df_constant,heart_df.TenYearCHD,cols)
```



Splitting data to train and test split

This code imports the scikit-learn library (sklearn) and creates a new variable called "new_features" that selects certain columns from a DataFrame called "heart_df". The selected columns are "age", "Sex_male", "cigsPerDay", "totChol", "sysBP", "glucose", and "TenYearCHD". It then creates two new variables, "x" and "y", which contain the data from the "new_features" DataFrame without the "TenYearCHD" column and only the "TenYearCHD" column, respectively. Then, it imports the "train_test_split" function from the "model_selection" module in scikit-learn and uses it to split the data into training and testing sets. The input data "x" and "y" are split into "x_train", "x_test", "y_train", and "y_test" with a test size of 20% and a random state of 5.

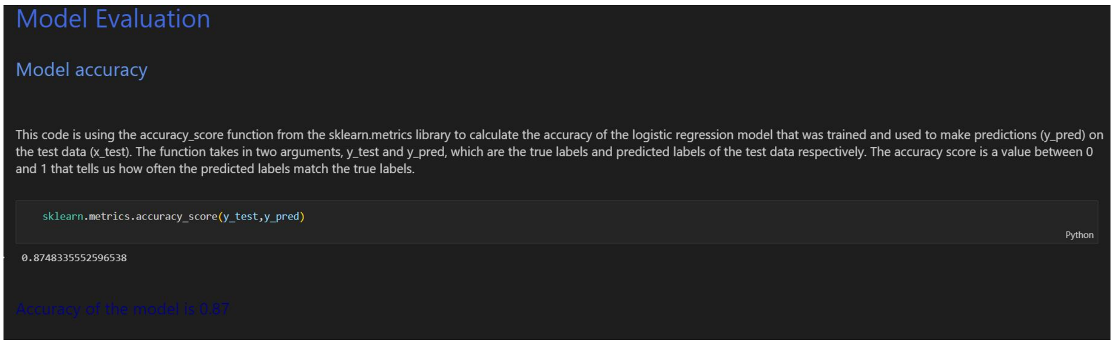
```python
import sklearn
new_features=heart_df[['age','Sex_male','cigsPerDay','totChol','sysBP','glucose','TenYearCHD']]
x=new_features.iloc[:,:-1]
y=new_features.iloc[:,-1]
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20,random_state=5)
```

Python

This code imports the "LogisticRegression" class from the scikit-learn library's "linear_model" module. It creates an instance of the LogisticRegression class and assigns it to the variable "logreg". Then it uses the .fit() method to train a logistic regression model on the training data (x_train, y_train). The .fit() method is used to find the best fit line for the input data. Then it creates a new variable "y_pred" which is used to store the predictions made by the logistic regression model on the test data (x_test). The .predict() method is used to make predictions using the trained model and input data x_test.

```python
from sklearn.linear_model import LogisticRegression
logreg=LogisticRegression()
logreg.fit(x_train,y_train)
y_pred=logreg.predict(x_test)
```

Python



Model Evaluation

Model accuracy

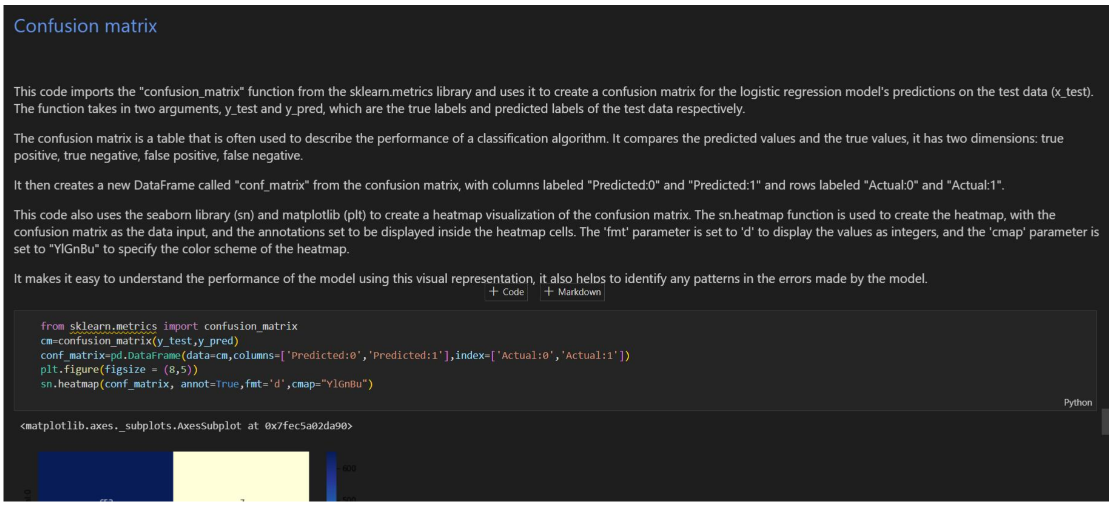
This code is using the accuracy_score function from the sklearn.metrics library to calculate the accuracy of the logistic regression model that was trained and used to make predictions (y_pred) on the test data (x_test). The function takes in two arguments, y_test and y_pred, which are the true labels and predicted labels of the test data respectively. The accuracy score is a value between 0 and 1 that tells us how often the predicted labels match the true labels.

```python
sklearn.metrics.accuracy_score(y_test,y_pred)
```

Python

```
0.8748335552596538
```

Accuracy of the model is 0.87



Confusion matrix

This code imports the "confusion_matrix" function from the sklearn.metrics library and uses it to create a confusion matrix for the logistic regression model's predictions on the test data (x_test). The function takes in two arguments, y_test and y_pred, which are the true labels and predicted labels of the test data respectively.

The confusion matrix is a table that is often used to describe the performance of a classification algorithm. It compares the predicted values and the true values, it has two dimensions: true positive, true negative, false positive, false negative.

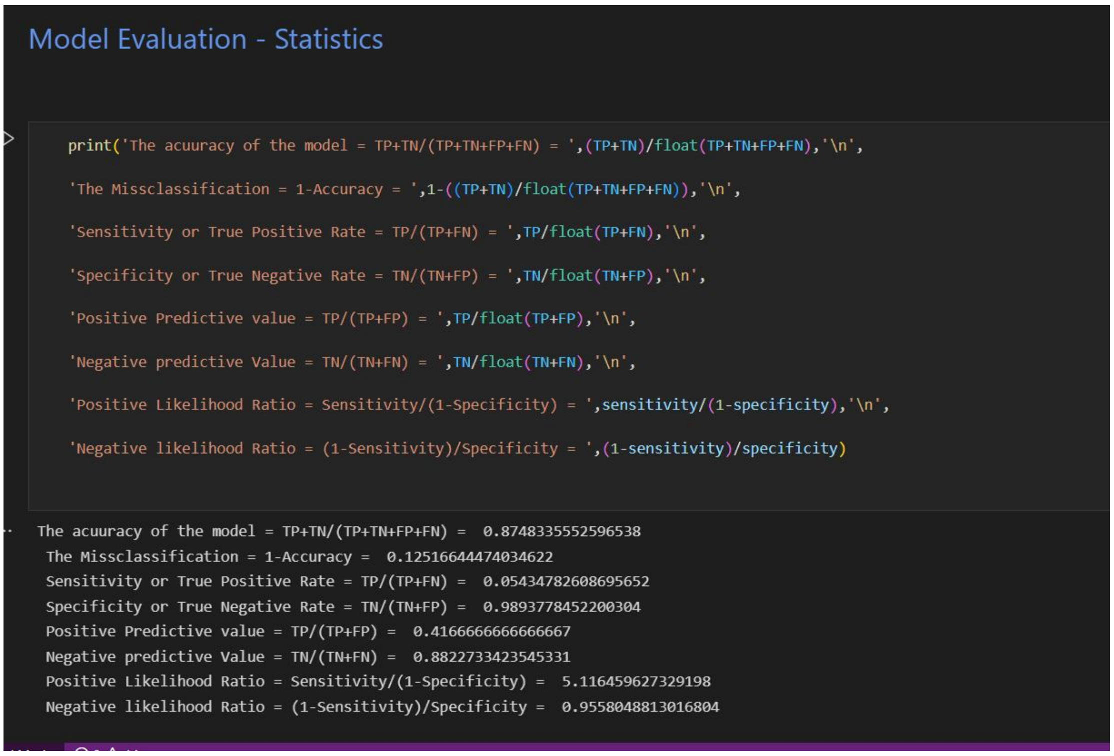
It then creates a new DataFrame called "conf_matrix" from the confusion matrix, with columns labeled "Predicted:0" and "Predicted:1" and rows labeled "Actual:0" and "Actual:1".

This code also uses the seaborn library (sn) and matplotlib (plt) to create a heatmap visualization of the confusion matrix. The sn.heatmap function is used to create the heatmap, with the confusion matrix as the data input, and the annotations set to be displayed inside the heatmap cells. The 'fmt' parameter is set to 'd' to display the values as integers, and the 'cmap' parameter is set to "YlGnBu" to specify the color scheme of the heatmap.

It makes it easy to understand the performance of the model using this visual representation, it also helps to identify any patterns in the errors made by the model.

```python
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
conf_matrix=pd.DataFrame(data=cm,columns=['Predicted:0','Predicted:1'],index=['Actual:0','Actual:1'])
plt.figure(figsize = (8,5))
sn.heatmap(conf_matrix, annot=True,fmt='d',cmap="YlGnBu")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fec5a02da90>
```



Model Evaluation - Statistics

```python
print('The acuuracy of the model = TP+TN/(TP+TN+FP+FN) = ',(TP+TN)/float(TP+TN+FP+FN),'\n',

'The Missclassification = 1-Accuracy = ',1-((TP+TN)/float(TP+TN+FP+FN)),'\n',

'Sensitivity or True Positive Rate = TP/(TP+FN) = ',TP/float(TP+FN),'\n',

'Specificity or True Negative Rate = TN/(TN+FP) = ',TN/float(TN+FP),'\n',

'Positive Predictive value = TP/(TP+FP) = ',TP/float(TP+FP),'\n',

'Negative predictive Value = TN/(TN+FN) = ',TN/float(TN+FN),'\n',

'Positive Likelihood Ratio = Sensitivity/(1-Specificity) = ',sensitivity/(1-specificity),'\n',

'Negative likelihood Ratio = (1-Sensitivity)/Specificity = ',(1-sensitivity)/specificity)
```

```
The acuuracy of the model = TP+TN/(TP+TN+FP+FN) =  0.8748335552596538
 The Missclassification = 1-Accuracy =  0.12516644474034622
 Sensitivity or True Positive Rate = TP/(TP+FN) =  0.05434782608695652
 Specificity or True Negative Rate = TN/(TN+FP) =  0.9893778452200304
 Positive Predictive value = TP/(TP+FP) =  0.4166666666666667
 Negative predictive Value = TN/(TN+FN) =  0.8822733423545331
 Positive Likelihood Ratio = Sensitivity/(1-Specificity) =  5.116459627329198
 Negative likelihood Ratio = (1-Sensitivity)/Specificity =  0.9558048813016804
```

THANK YOU :)