Group Report

Course Title:	**MSc in Data Analytics**
Lecturer Name:	**Courtney Ford**
Module Title:	**Machine Learning and Pattern Recognition**
Assignment Title:	**Supervised Machine Learning – Regression**

Submitted by:

10634205 – Sandeep Gopu

10634610 - Anuradha Anuradha

10634920 – Anjali Mary Philip

Table of Contents

INTRODUCTION:

We have observed that in the recent global pandemic people's health got impacted and there has been a sharp increase in the cases of heart attack, which is becoming an area of concern as human beings are feeling isolated in this covid situation and they are unable to share their emotions with others. Due to this, the health sectors are facing difficulties in predicting the chances of attack, and as there are many components and subcomponents due to which it is not an easy job to create a plan manually for identifying which gender, the age bracket and the physical conditions/mental state of the individual are vulnerable to a heart attack. Machine learning approaches can help us to detect the chances of an attack. This technique produces predictive models which learn the behavior of the heart attack and utilize this to detect the most basic cause of a heart attack.

In our assignment, the capability of machine learning methods to detect heart attacks is scrutinized. So, we are utilizing the Logistic Regression supervised learning and optimization techniques to find out the accurate result of the heart dataset.

PROBLEM STATEMENT:

"*Build a regression model which can recognize the chances of an individual to a get Heart attack*"

ANALYSIS :

Following are the action items that we performed which included both descriptive and predictive analysis -

1. **Choice of Dependent and Independent Variables**
2. **Examining dataset**
3. **Data Pre-processing**
4. **Feature Scaling**
5. **Model Development and Model Evaluation**
6. **Model Comparison using Optimization Technique**

1. Choice of Dependent and Independent Variables:

We have a total of 14 variables in our dataset. So, while analyzing the dataset we found that we have a variable “output” which has the value of 0 and 1. The output variable indicates more chances of a person getting a heart attack and fewer chances of a person getting a heart attack. Hence, we decided to take the output variable as the target variable.

So, we have 13 Independent variables and 1 dependent variable. As the nature of our target variable is categorical, therefore, we decided to develop a Logistic Regression model.

2. Examining Dataset:

While examining the Heart Attack Analysis and Prediction dataset, we found that there was a total of 14 columns and 303 rows, which we felt was enough data to train the machine and predict the approximate chances of a heart attack in an individual.

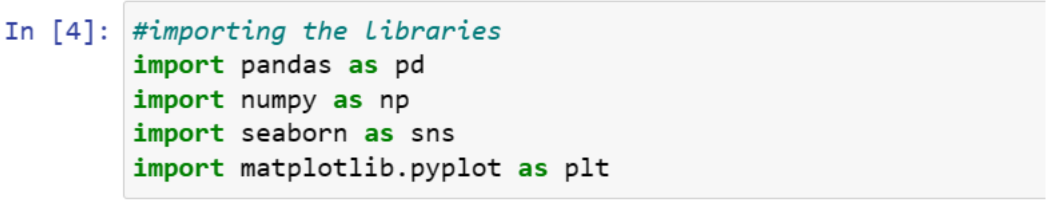
Data field	Field description
1. Age	Age of the Patient
2. Sex	Sex of the Patient
3. Cp	Types of Chests Pain
4. Trtbps	Resting Blood Pressure
5. Chol	Cholesterol (mg/dl)
6. Fbs	Fasting blood sugar > 120 mg/dl (1=true, 0=false)
7. Restecg	Resting electrocardiographic results
8. Thalachh	Maximum heart rate Achieved
9. Exang	Exercise-induced angina (1=yes, 0 = no)
10. Oldpeak	Previously achieved peak
11. Slp	Slope

12. Caa	Number of vessels ranging from 0-3
13. thall	Thallium stress test results (0-3)
14. Output	0- fewer chances of getting a heart attack 1- More chances of getting a heart attack

3. Data Pre-Processing:

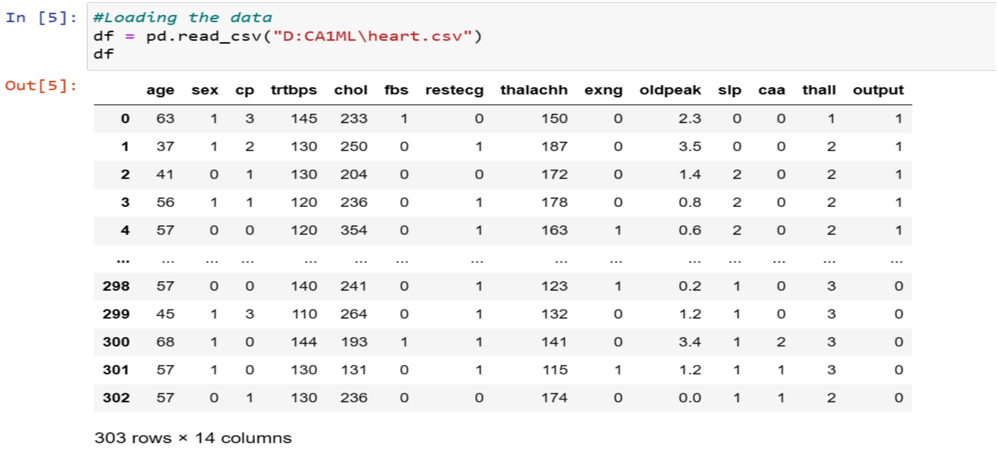
Following are the different steps included in the data preparation:

Step 1: In this step, we are importing the different libraries from the python package for performing some specific operations and assigning them as a variable.



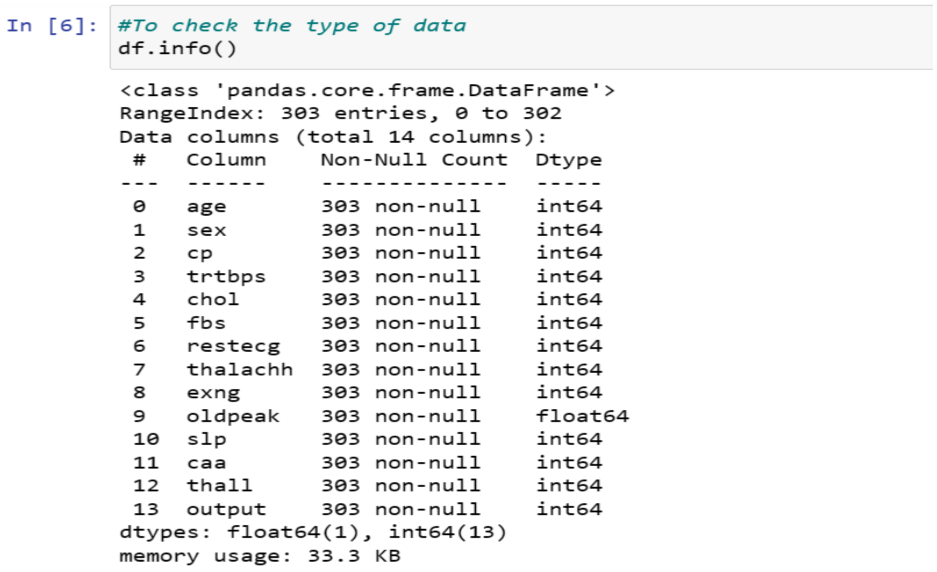
```
In [4]: #importing the libraries
        import pandas as pd
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
```

Step 2: In this step we are reading the csv file which includes the dataset into a variable and when we print the variable we can see the result which shows our dataset which includes 14 columns and 303 rows.



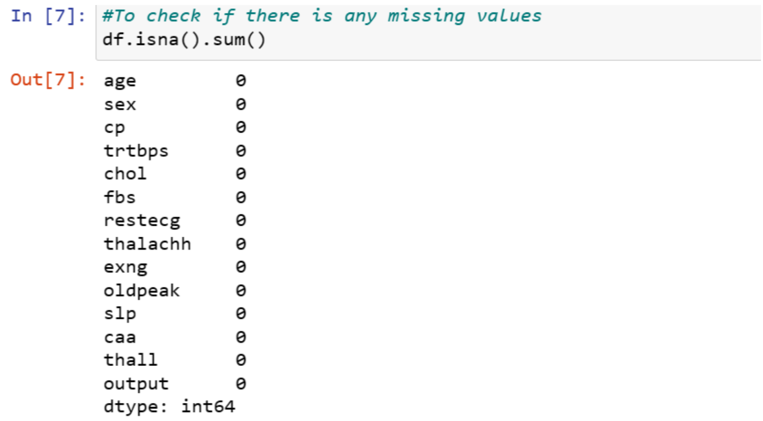
```
In [5]: #Loading the data
        df = pd.read_csv("D:CA1ML\heart.csv")
        df

Out[5]:
            age  sex  cp  trtbps  chol  fbs  restecg  thalachh  exng  oldpeak  slp  caa  thall  output
        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
      ...   ...  ...  ...    ...   ...  ...      ...       ...   ...      ...  ...  ...    ...     ...
      298    57    0   0     140   241    0        1       123     1      0.2    1    0      3       0
      299    45    1   3     110   264    0        1       132     0      1.2    1    0      3       0
      300    68    1   0     144   193    1        1       141     0      3.4    1    2      3       0
      301    57    1   0     130   131    0        1       115     1      1.2    1    1      3       0
      302    57    0   1     130   236    0        0       174     0      0.0    1    1      2       0

        303 rows × 14 columns
```

Step 3: In this step, we execute the info() method which helps in identifying the datatype of all the columns in the dataset and we found out that 13 attributes have an integer datatype and 1 attribute has a float datatype. As we don't have any categorical values in our dataset we don’t need to do the convert it into numerical values.

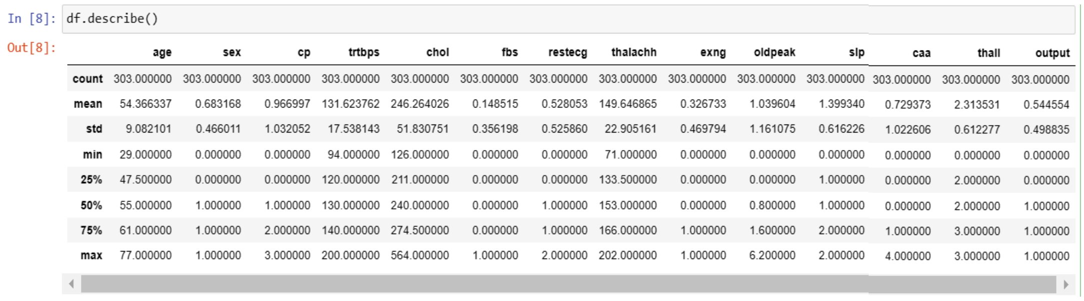


```
In [6]: #To check the type of data
        df.info()

<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   trtbps    303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalachh  303 non-null    int64  
 8   exng      303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slp       303 non-null    int64  
 11  caa       303 non-null    int64  
 12  thall     303 non-null    int64  
 13  output    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

Step 4: After checking the datatype, we are checking for the null values. If we have any null values, we need to handle them. As per our result, our dataset is perfect as we have no null values.



```
In [7]: #To check if there is any missing values
        df.isna().sum()

Out[7]: age         0
        sex         0
        cp          0
        trtbps      0
        chol        0
        fbs         0
        restecg     0
        thalachh    0
        exng        0
        oldpeak     0
        slp         0
        caa         0
        thall       0
        output      0
        dtype: int64
```

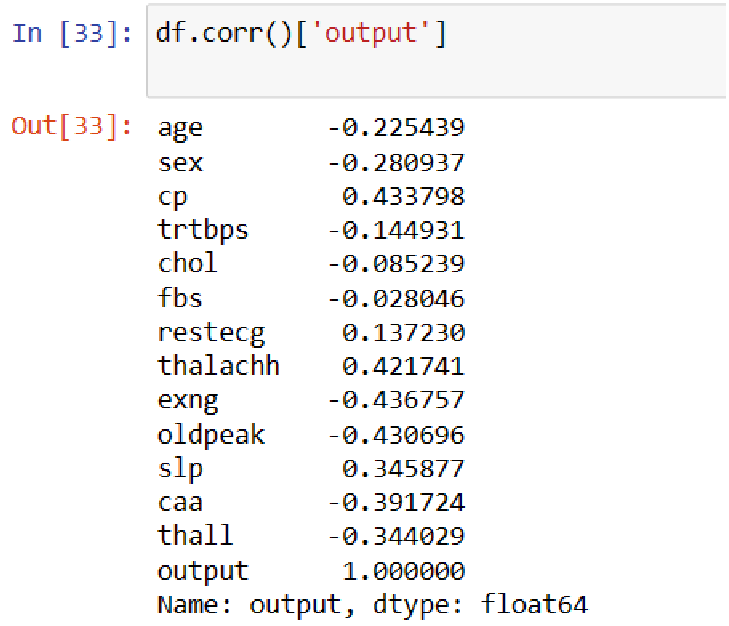


```
In [8]: df.describe()
```

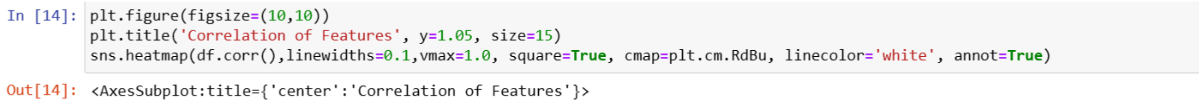
Out[8]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	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	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.313531	0.544554
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.612277	0.498835
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.000000

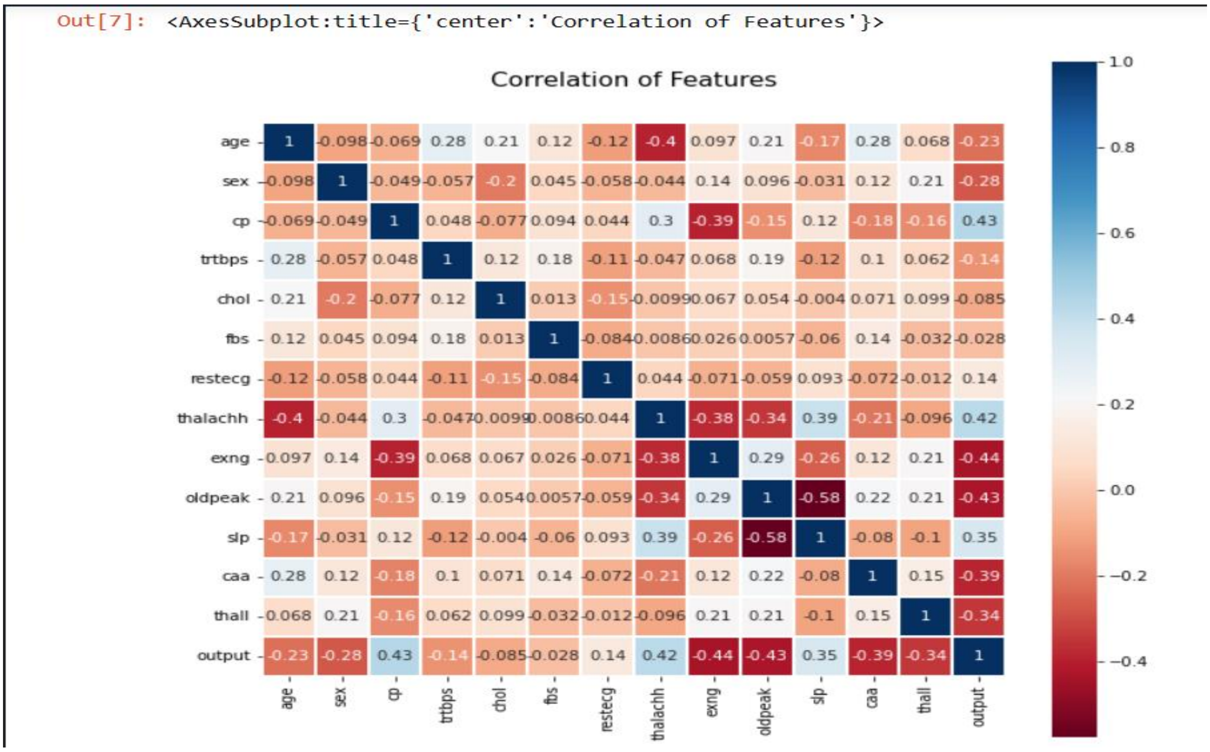
Step 5: In this step, we generated the correlation matrix between the independent and the dependent variables. We also plotted it in form of a heat map to visualize the impact of the features on heart attack.



```
In [33]: df.corr()['output']

Out[33]: age         -0.225439
         sex         -0.280937
         cp           0.433798
         trtbps      -0.144931
         chol        -0.085239
         fbs         -0.028046
         restecg      0.137230
         thalachh     0.421741
         exng        -0.436757
         oldpeak     -0.430696
         slp          0.345877
         caa         -0.391724
         thall       -0.344029
         output       1.000000
         Name: output, dtype: float64
```



```
In [14]: plt.figure(figsize=(10,10))
         plt.title('Correlation of Features', y=1.05, size=15)
         sns.heatmap(df.corr(),linewidths=0.1,vmax=1.0, square=True, cmap=plt.cm.RdBu, linecolor='white', annot=True)

Out[14]: <AxesSubplot:title={'center':'Correlation of Features'}>
```

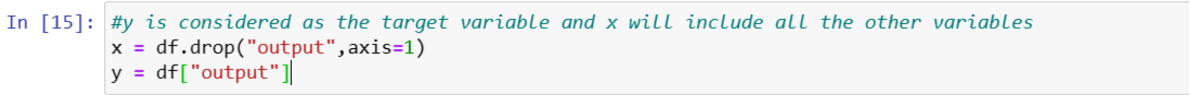


```
Out[7]: <AxesSubplot:title={'center':'Correlation of Features'}>
```

Correlation of Features

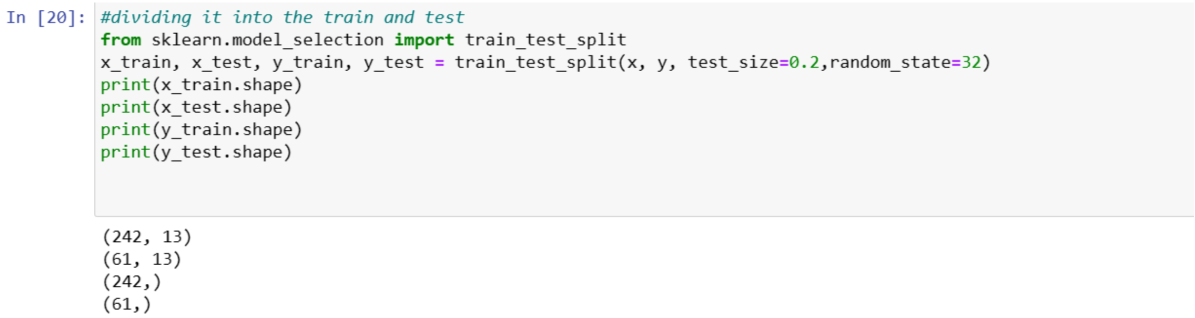
	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
age	1	-0.098	-0.069	0.28	0.21	0.12	-0.12	-0.4	0.097	0.21	-0.17	0.28	0.068	-0.23
sex	-0.098	1	-0.049	-0.057	-0.2	0.045	-0.058	-0.044	0.14	0.096	-0.031	0.12	0.21	-0.28
cp	-0.069	-0.049	1	0.048	-0.077	0.094	0.044	0.3	-0.39	-0.15	0.12	-0.18	-0.16	0.43
trtbps	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
thalachh	-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
exng	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
slp	-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
caa	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
thall	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
output	-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

Step 6: In this step, we created two variables to assign the attributes. We dropped the dependent attribute ‘output’ and assigned the independent variables to ‘x’ and then we assigned our dependent variable to ‘y’.



```
In [15]: #y is considered as the target variable and x will include all the other variables
         x = df.drop("output",axis=1)
         y = df["output"]
```

Step 7: We divided our dataset into train and test set by importing the ‘train_test_split’. We chose the ratio of 0.8 – 0.2 for the train – test set.



```
In [20]: #dividing it into the train and test
         from sklearn.model_selection import train_test_split
         x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,random_state=32)
         print(x_train.shape)
         print(x_test.shape)
         print(y_train.shape)
         print(y_test.shape)

         (242, 13)
         (61, 13)
         (242,)
         (61,)
```

4. Feature Scaling:

“Feature Scaling is a technique to standardize the independent features present in the data in a fixed range.” [1] Feature scaling is performed after splitting the train-test to get more accurate report.

The two main techniques used in Feature scaling are:

Standardization and Normalization [2][3]

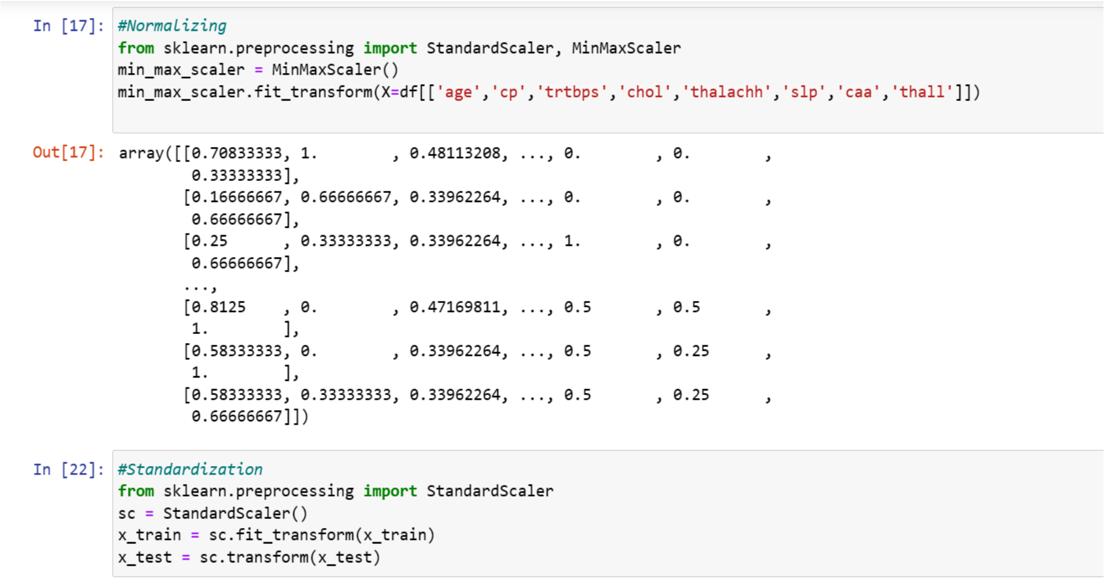
We have performed Normalization using the Min-Max Normalization.

a. **Min-Max Normalization**: It is a technique to scale the data value in the range of 0 to 1 based on the minimum and maximum values.

Formula: $X_{\text{new}} = \frac{X_i - \min(X)}{\max(x) - \min(X)}$

b. **Standardization**: It is a productive technique that re-scales a feature value so that it has a distribution with 0 mean value and variance equal to 1.

Formula: $X_{\text{new}} = \frac{X_i - X_{\text{mean}}}{\text{Standard Deviation}}$



```
In [17]: #Normalizing
         from sklearn.preprocessing import StandardScaler, MinMaxScaler
         min_max_scaler = MinMaxScaler()
         min_max_scaler.fit_transform(X=df[['age','cp','trtbps','chol','thalachh','slp','caa','thall']])

Out[17]: array([[0.70833333, 1.        , 0.48113208, ..., 0.        , 0.        ,
                 0.33333333],
                [0.16666667, 0.66666667, 0.33962264, ..., 0.        , 0.        ,
                 0.66666667],
                [0.25      , 0.33333333, 0.33962264, ..., 1.        , 0.        ,
                 0.66666667],
                ...,
                [0.8125    , 0.        , 0.47169811, ..., 0.5       , 0.5       ,
                 1.        ],
                [0.58333333, 0.        , 0.33962264, ..., 0.5       , 0.25      ,
                 1.        ],
                [0.58333333, 0.33333333, 0.33962264, ..., 0.5       , 0.25      ,
                 0.66666667]])

In [22]: #Standardization
         from sklearn.preprocessing import StandardScaler
         sc = StandardScaler()
         x_train = sc.fit_transform(x_train)
         x_test = sc.transform(x_test)
```

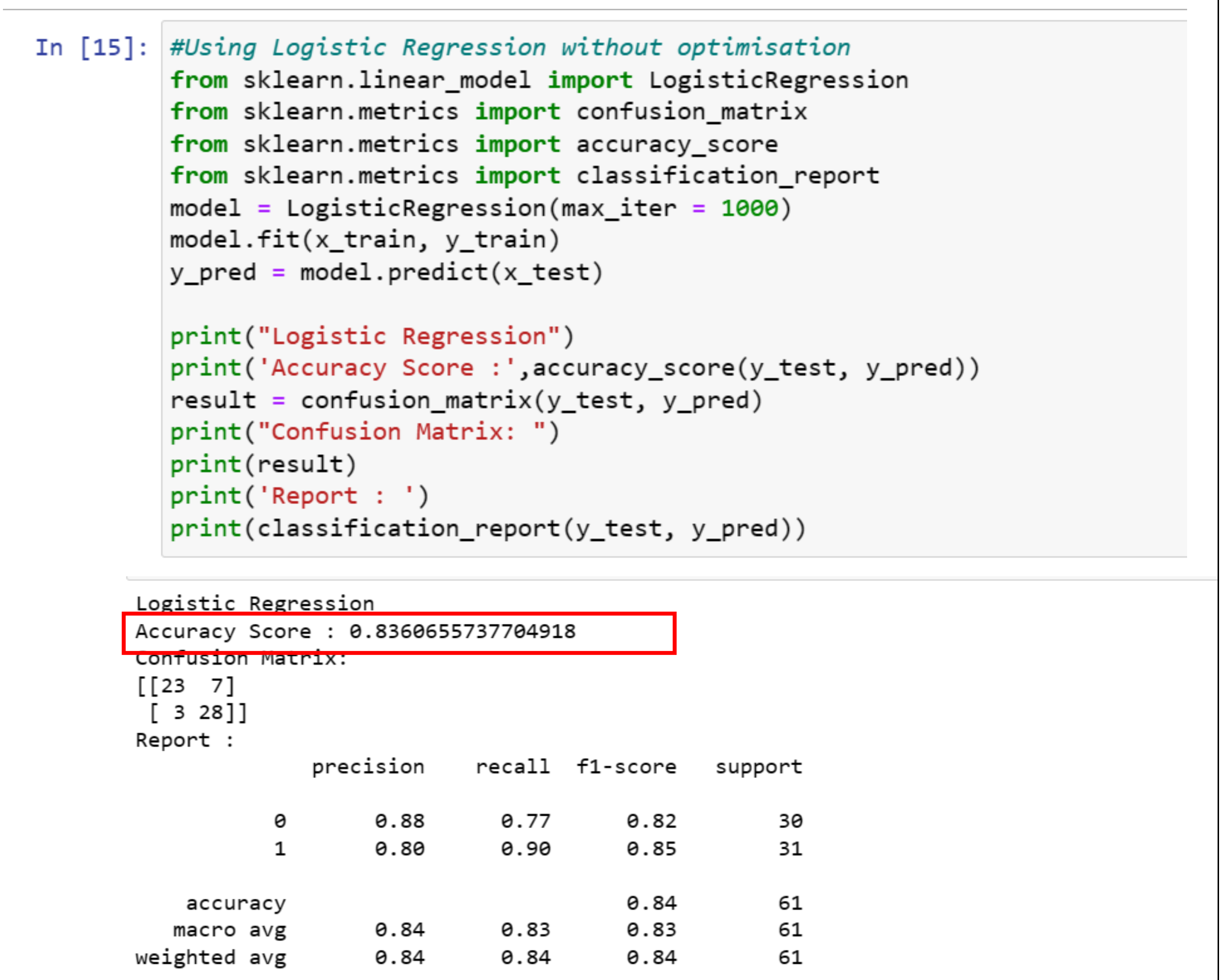
5. Model Development and Model Evaluation:

Logistic Regression is one of the most popular machine learning algorithms which comes under the supervised learning technique. It is used to identify the categorical dependent variable using the independent variables.

It also anticipates the output of a categorical dependent variable. So, the outcome must be a categorical or discrete value, which is either 1 or 0, Yes or No, etc. but instead of giving the exact value as 1 and 0, it gives the probabilistic values which lie between 1 and 0.[12] So, the target column is “output” in the heart dataset, which has the value of either 1 or 0 which is a categorical value. So, that is the reason why we are moving ahead with the logistic regression instead of the linear regression.

Further in this stage, we are importing LogisticRegression from sklearn.linear_model, and along with this, we are importing accuracy_score, confusion_matrix, and classification_report metrics from sklearn.metrics to get the clear report. After importing the required metrics and model, we assigned the LogisticRegression model in a variable called “model” and after that we are fitting it into a training set using fit() method then we are predicting the model using the testing set.

Lastly, we evaluated the performance of the Logistic Regression model using the classification report which includes accuracy, precision, recall, and F1-score, and also printed the confusion matrix.



```
In [15]: #Using Logistic Regression without optimisation
         from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import confusion_matrix
         from sklearn.metrics import accuracy_score
         from sklearn.metrics import classification_report
         model = LogisticRegression(max_iter = 1000)
         model.fit(x_train, y_train)
         y_pred = model.predict(x_test)

         print("Logistic Regression")
         print('Accuracy Score :',accuracy_score(y_test, y_pred))
         result = confusion_matrix(y_test, y_pred)
         print("Confusion Matrix: ")
         print(result)
         print('Report : ')
         print(classification_report(y_test, y_pred))
```

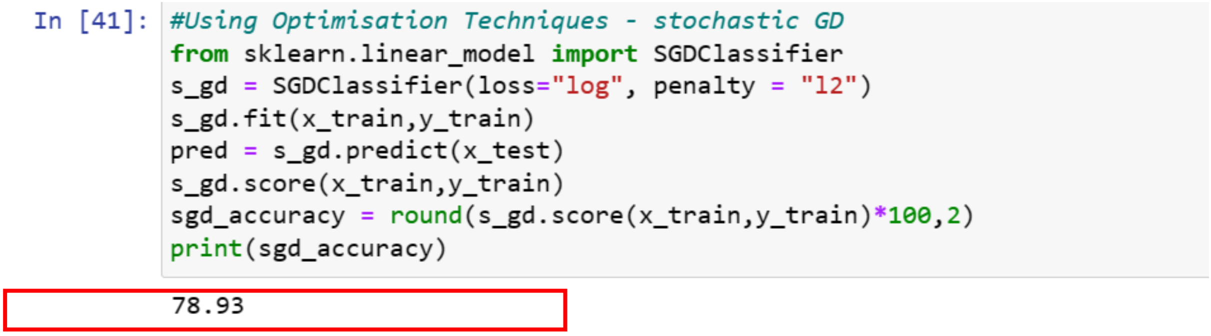
```
Logistic Regression
Accuracy Score : 0.8360655737704918
Confusion Matrix:
[[23  7]
 [ 3 28]]
Report :
              precision    recall  f1-score   support

           0       0.88      0.77      0.82        30
           1       0.80      0.90      0.85        31

    accuracy                           0.84        61
   macro avg       0.84      0.83      0.83        61
weighted avg       0.84      0.84      0.84        61
```

6. Model Comparison:

Gradient descent is an optimization algorithm that is majorly used to train machine learning models and neural networks. This optimization technique is used to make the machine learning model more precise at predicting an outcome or identifying the data.

we choose Stochastic Gradient Descent model as we are taking random samples instead of the entire data set for each iteration.

In the code we imported SGDClassifier from sklearn.linear_model and then assigned the method to a variable called "s_gd". Then, we fitted it into a training set using fit() method and then we predicted the model using the testing set.



```
In [41]: #Using Optimisation Techniques - stochastic GD
         from sklearn.linear_model import SGDClassifier
         s_gd = SGDClassifier(loss="log", penalty = "l2")
         s_gd.fit(x_train,y_train)
         pred = s_gd.predict(x_test)
         s_gd.score(x_train,y_train)
         sgd_accuracy = round(s_gd.score(x_train,y_train)*100,2)
         print(sgd_accuracy)

         78.93
```

The accuracy of the model with the optimization technique was 78.93. Without optimization, we achieved an accuracy score of 0.83.

CONCLUSION:

We understood from the predictive assessment processes that we had developed a regression model that could identify heart attacks based on a number of variables, including age, sex, cp, caa, and other variables. With the approach described above, we created a model that we used to train our machine learning model and give it the capacity to recognize values from a given dataset. Descriptive analysis was also carried out earlier to investigate the dataset, spot faults in the problem description, and prepare the data for model development and validation. The forecast can be made more accurate if we train our machine with more data across more time periods, which may be approximately 1500 individuals. In addition, we can say that this prediction is not 100% accurate due to the limited data, which is just 303 individuals. Additionally, we found that the Heart Attack dataset offers a ton of potential for predictions, and machines can be trained to make a variety of additional predictions based on the same dataset, such as the health of the heart based on age bracket and other categories such as trtbps, chol, caa, oldpeak, caa, and many other predictive questions.

MINUTES OF GROUP MEETING

	DATE & TIME	TOPICS DISCUSSED	MEETING NOTES	PARTICIPANTS
1.	16/02/2023 15:00 PM	Dataset selection	We focused on selecting the dataset from Kaggle and had a word with mam for the confirmation of the dataset and she said we can move forward with the dataset.	Anuradha Anjali Sandeep
2.	17/02/2023 09:00 AM	Allocation of topics	After the confirmation of the dataset. We discussed the topics to be included and divided them among us.	
3.	20/02/2023 12:00 PM	Follow-up meeting about the progress.	Here, we discussed about data preparation and feature scaling.	

4.	27/02/2023 11:00 AM	Model evaluation and model comparison	Completed the coding part.	
5.	28/02/2023 11:00 AM	Preparation of draft	Collected all the data from team members and combined it to prepare the draft report.	
6.	03/03/2022 2:00 PM	Preparation and finalizing of the report.	This was the final meeting where we all reviewed and finalized.	Anuradha Anjali Sandeep

LINK TO COLAB NOTEBOOK:

CA1_ML.ipynb - Colaboratory (google.com)

LINK TO DATASET:

Heart Attack Analysis & Prediction Dataset | Kaggle

REFERENCES:

[1] https://www.geeksforgeeks.org/ml-feature-scaling-part-1/

[2] https://elearning.dbs.ie/pluginfile.php/1789511/mod_resource/content/1/Data%20Processing.pdf (Page - 16)

[3] ML | Feature Scaling – Part 2 - GeeksforGeeks

[4] https://www.youtube.com/watch?v=sxEqtjLC0aM&t=26s

[5] https://www.youtube.com/watch?v=XnOAdxOWXWg

[6] https://www.youtube.com/watch?v=prWyZhcktn4

[7] https://www.youtube.com/watch?v=GwIo3gDZCVQ&t=38s

[8] https://towardsdatascience.com/introduction-to-data-preprocessing-in-machine-learning-a9fa83a5dc9d

[9] https://www.geeksforgeeks.org/ml-stochastic-gradient-descent-sgd/

[10] https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc

[11] https://www.w3schools.com/python/python_ml_logistic_regression.asp#:~:text=Logistic%20regression%20aims%20to%20solve,tumor%20is%20malignant%20or%20benign.

[12] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html

[13] Heart attack death rates took a sharp turn and increased during the pandemic, study shows (news-medical.net)