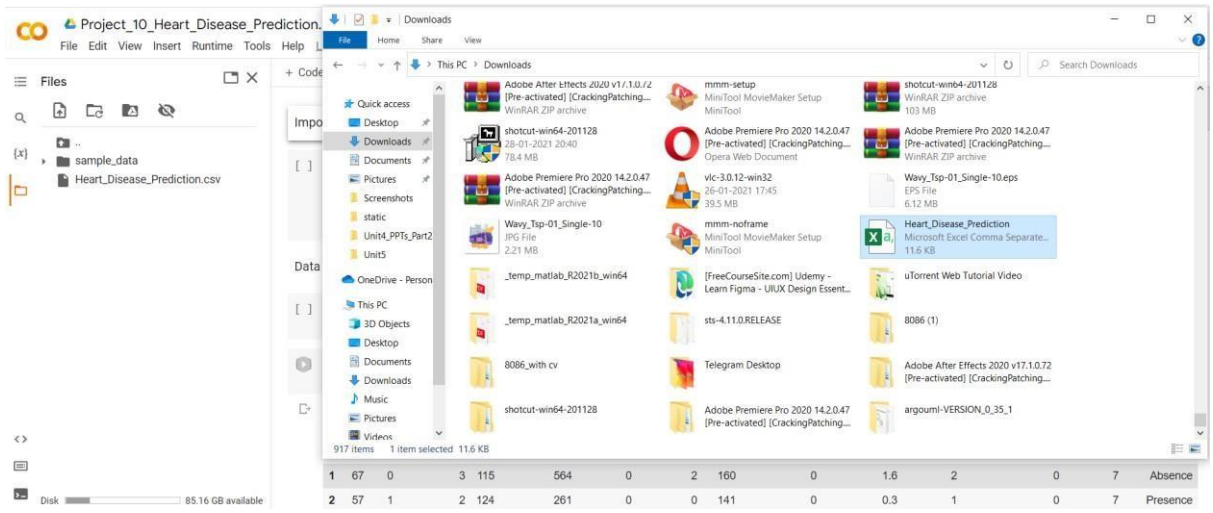
Project Development Phase – Sprint 2

Team ID	PNT2022TMID53202
Project Members	Hari Krishna A S, Pooja Laxmi S, Charulatha S, Amose
Project Name	Visualizing and Predicting Heart Diseases with an Interactive Dash Board
Project mentors	Industry mentor – Mohanavalli Faculty mentor –Dr. Arulkumar Venkatachalam

Prediction of Heart Disease using Logistic Regression in Google colab:

1. Upload the dataset into Google Colab:



Importing the Dependencies

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

Data Importation and Processing loading

the csv data to a Pandas DataFrame

```
[2] # loading the csv data to a Pandas DataFrame
    heart_data = pd.read_csv('/content/Heart_Disease_Prediction.csv')
```

Printing first 5 rows of the dataset

```
# print first 5 rows of the dataset
heart_data.head()
```

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	Slope of ST	Number of vessels fluro	Thallium	Heart Disease
0	70	1	4	130	322	0	2	109	0	2.4	2	3	3	Presence
1	67	0	3	115	564	0	2	160	0	1.6	2	0	7	Absence
2	57	1	2	124	261	0	0	141	0	0.3	1	0	7	Presence
3	64	1	4	128	263	0	0	105	1	0.2	2	1	7	Absence
4	74	0	2	120	269	0	2	121	1	0.2	1	1	3	Absence

Print last 5 rows of the dataset

```
# print last 5 rows of the dataset
heart_data.tail()
```

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	Slope of ST	Number of vessels fluro	Thallium	Heart Disease
265	52	1	3	172	199	1	0	162	0	0.5	1	0	7	Absence
266	44	1	2	120	263	0	0	173	0	0.0	1	0	7	Absence
267	56	0	2	140	294	0	2	153	0	1.3	2	0	3	Absence
268	57	1	4	140	192	0	0	148	0	0.4	2	0	6	Absence
269	67	1	4	160	286	0	2	108	1	1.5	2	3	3	Presence

Number of rows and columns in the dataset

```
# number of rows and columns in the dataset
heart_data.shape
```

```
(270, 14)
```

Getting some info about the data

```
# getting some info about the data
heart_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 270 entries, 0 to 269
Data columns (total 14 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Age                       270 non-null    int64  
 1   Sex                       270 non-null    int64  
 2   Chest pain type           270 non-null    int64  
 3   BP                        270 non-null    int64  
 4   Cholesterol               270 non-null    int64  
 5   FBS over 120              270 non-null    int64  
 6   EKG results               270 non-null    int64  
 7   Max HR                    270 non-null    int64  
 8   Exercise angina           270 non-null    int64  
 9   ST depression             270 non-null    float64
 10  Slope of ST               270 non-null    int64  
 11  Number of vessels fluro   270 non-null    int64  
 12  Thallium                  270 non-null    int64  
 13  Heart Disease             270 non-null    object 
dtypes: float64(1), int64(12), object(1)
memory usage: 29.7+ KB
```

Checking for missing values

```
# checking for missing values
heart_data.isnull().sum()
```

```
Age                       0
Sex                       0
Chest pain type           0
BP                        0
Cholesterol               0
FBS over 120              0
EKG results               0
Max HR                    0
Exercise angina           0
ST depression             0
Slope of ST               0
Number of vessels fluro   0
Thallium                  0
Heart Disease             0
dtype: int64
```

Statistical measures about the data

```
# statistical measures about the data
heart_data.describe()
```

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	Slope of ST	Number of vessels fluro
count	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000	270.00000	270.000000	270.000000
mean	54.433333	0.677778	3.174074	131.344444	249.659259	0.148148	1.022222	149.677778	0.329630	1.05000	1.585185	0.670370
std	9.109067	0.468195	0.950090	17.861608	51.686237	0.355906	0.997891	23.165717	0.470952	1.14521	0.614390	0.943896
min	29.000000	0.000000	1.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.00000	1.000000	0.000000
25%	48.000000	0.000000	3.000000	120.000000	213.000000	0.000000	0.000000	133.000000	0.000000	0.00000	1.000000	0.000000
50%	55.000000	1.000000	3.000000	130.000000	245.000000	0.000000	2.000000	153.500000	0.000000	0.80000	2.000000	0.000000
75%	61.000000	1.000000	4.000000	140.000000	280.000000	0.000000	2.000000	166.000000	1.000000	1.60000	2.000000	1.000000
max	77.000000	1.000000	4.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.20000	3.000000	3.000000

Checking the distribution of Target Variable

```
[9]  # checking the distribution of Target Variable
     heart_data['Heart Disease'].value_counts()
```

```
Absence     150
Presence    120
Name: Heart Disease, dtype: int64
```

Presence --> Defective Heart

Absence --> Healthy Heart

```
# checking the distribution of Target Variable
heart_data['Heart Disease'].value_counts()
```

```
Absence     150
Presence    120
Name: Heart Disease, dtype: int64
```

Splitting the dataset features

Splitting the Features and Target

```
X = heart_data.drop(columns='Heart Disease', axis=1)
Y = heart_data['Heart Disease']
```

```
print(X)
```

```
     age  sex  cp  trestbps  chol  ...  exang  oldpeak  slope  ca  thal
0     63    1   3       145   233  ...      0      2.3      0   0     1
1     37    1   2       130   250  ...      0      3.5      0   0     2
2     41    0   1       130   204  ...      0      1.4      2   0     2
3     56    1   1       120   236  ...      0      0.8      2   0     2
4     57    0   0       120   354  ...      1      0.6      2   0     2
..   ...  ...  ..       ...   ...  ...    ...      ...    ...  ..   ...
298   57    0   0       140   241  ...      1      0.2      1   0     3
299   45    1   3       110   264  ...      0      1.2      1   0     3
300   68    1   0       144   193  ...      0      3.4      1   2     3
301   57    1   0       130   131  ...      1      1.2      1   1     3
302   57    0   1       130   236  ...      0      0.0      1   1     2

[303 rows x 13 columns]
```

```
print(Y)
```

```
0      Presence
1       Absence
2      Presence
3       Absence
4       Absence
         ...
265     Absence
266     Absence
267     Absence
268     Absence
269    Presence
Name: Heart Disease, Length: 270, dtype: object
```

Splitting the Data into Training data & Test Data

Splitting the Data into Training data & Test Data

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)
```

```
print(X.shape, X_train.shape, X_test.shape)
```

```
(270, 13) (216, 13) (54, 13)
```

Model Training using Logistic Regression

Model Training

Logistic Regression

```
[15] model = LogisticRegression()
```

```
# training the LogisticRegression model with Training data
model.fit(X_train, Y_train)
```

```
/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,
LogisticRegression()
```

Model Evaluation

Model Evaluation

Accuracy Score

```
[17] # accuracy on training data
     X_train_prediction = model.predict(X_train)
     training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

```
[18] print('Accuracy on Training data : ', training_data_accuracy)
```

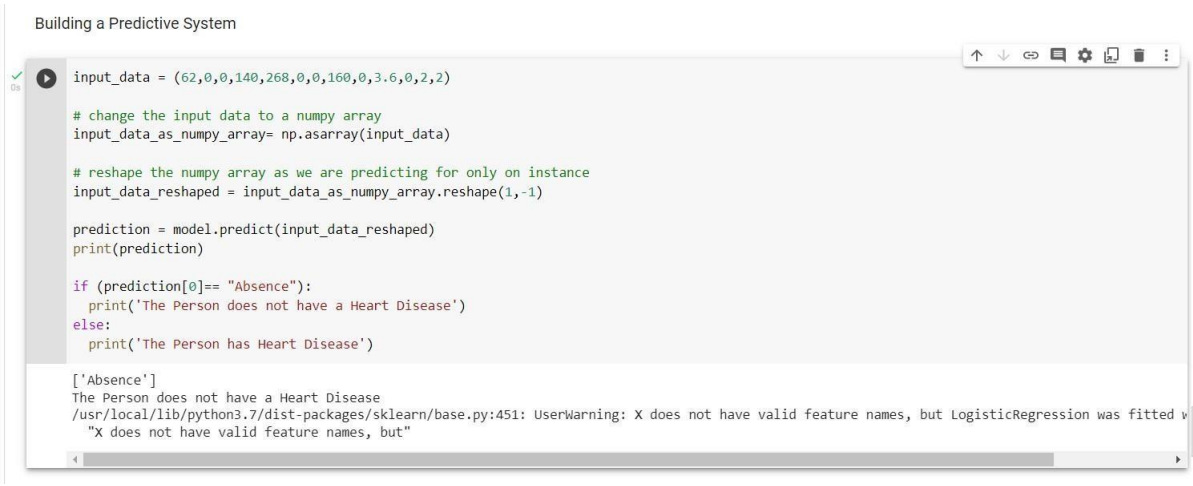
```
Accuracy on Training data :  0.875
```

```
[19] # accuracy on test data
     X_test_prediction = model.predict(X_test)
     test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

```
print('Accuracy on Test data : ', test_data_accuracy)
```

```
Accuracy on Test data :  0.8333333333333334
```

Building a Predictive System



Findings:

Training Accuracy: 87.5%

Testing Accuracy: 83.34%