Fundamentals of Data mining- IT3051

Mini Group Project

Table of Content

1.Background Introduction

This Final Project document is based on the mini project for the fundamentals of data mining module at Sri Lanka Institute of Information Technology. Mainly this project focuses on solving a real world problem that affects the human beings using data mining or machine learning technologies. As per the instruction we as a team found a problem that affected people in their lives.

Heart diseases are frequent these days. There are so many people that are suffering from heart diseases. As the prevention is better than curing, we as a team decided to train a model that classifies the people who are in risk to have a heart disease in the next 10 years. From that model people can check their health and take necessary precautions to prevent it from happening.

So, This project can save lives and minimize the risks of having heart diseases in human beings. We as a team think that it increases the quality of life of people.

2.Problem Definition

Framingham Heart Study, long-term research project developed to identify risk factors of cardiovascular disease, the findings of which had far-reaching impacts on medicine. Indeed, much common knowledge about heart disease—including the effects of smoking, diet, and exercise—can be traced to the Framingham study. The study's findings further emphasized the need for preventing, detecting, and treating risk factors of cardiovascular disease in their earliest stages.

So we plan to create an app for detecting patients who can be future cardiovascular patients.for detecting the ones , we get 10 years of cardiovascular patients details and analyze the dataset and build our prediction model.in here , we take few information of patient to analyze whether is he or she in critical stage or not. In other words we analyze if he or she has a chance to victimize the cardiovascular disease or not.

As final output, we expect to show the particular patient has the chance to have cardiovascular disease or not in future.

3.Variable List

Code List			
Var.#	**Variable Name**	**Description**	**Data type**
1	age	Patient Age	numeric
2	sysBP	Systolic Blood Pressure(mmHg)	numeric
3	diaBP	Diastolic Blood Pressure(mmHg)	numeric
4	glucose	Glucose Level	numeric
5	Option->male	Patient's Gender is male	binary
6	Option->female	Patient's Gender is female	binary
7	Option2->yes	Taking Blood Pressure Medications	binary
8	Option2->no	Not Taking Blood Pressure Medications	binary
9	totChol	Total Cholesterol Level(mg/dL)	numeric
10	BMI	BMI (Body Mass Index)	numeric
11	Option3->yes	Had Prevalent Stroke	binary
12	Option3->no	Had not Prevalent Stroke	binary
13	Option4->yes	Had Prevalent Hypertension	binary
14	Option4->no	Had not Prevalent Hypertension	binary
15	PregnantNo	Number of times been pregnant	numeric
16	plasmaGluecoseConc	Plasma Glucose Concentration	numeric
17	tricepsThickness	Triceps Thickness	numeric
18	SeriumInsulin	Serum Insulin	numeric
19	diabPedigreeFunc	Diabetic Pedigree Function	numeric

4.Data Preparation

- The data set is partitioned into,

    1. Training set - 75% of the records
    2. Testing set - 25% of the records

5.Data Preprocessing

01.Importing Dataset, Feature Selection & Split Dataset

Downloads/ | FDM - Assignment Model Trainir | IRWA-LAB -KNN.py - Jupyter Te | Assignment3 - Jupyter Notebooi

localhost:8888/notebooks/Downloads/FDM%20-%20Assignment%20Model%20Training_All%20(1).ipynb

jupyter FDM - Assignment Model Training_All (1) (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3

Run | Code

In [9]:
```
import os
import pandas as pd
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
```

In [10]:
```
os.chdir("D:/SLIIT Documents/Year 3/Semester 2/FDM/Assignment")

rawDataset=pd.read_csv("framingham.csv")
rawDataset.head()
```

Out[10]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	Te
0	1	39	4.0	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	80.0	77.0	
1	0	46	2.0	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	95.0	76.0	
2	1	48	1.0	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.34	75.0	70.0	
3	0	61	3.0	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.58	65.0	103.0	
4	0	46	3.0	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.10	85.0	85.0	

In [13]:
```
#dropping education column as it is not helpful for the analysis
rawDataset.drop(['education'], axis=1, inplace=True)
```

In [14]:
```
rawDataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4240 entries, 0 to 4239
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   male             4240 non-null   int64
 1   age              4240 non-null   int64
 2   currentSmoker    4240 non-null   int64
 3   cigsPerDay       4211 non-null   float64
 4   BPMeds           4187 non-null   float64
 5   prevalentStroke  4240 non-null   int64
 6   prevalentHyp     4240 non-null   int64
 7   diabetes         4240 non-null   int64
 8   totChol          4190 non-null   float64
 9   sysBP            4240 non-null   float64
 10  diaBP            4240 non-null   float64
 11  BMI              4221 non-null   float64
 12  heartRate        4239 non-null   float64
 13  glucose          3852 non-null   float64
 14  TenYearCHD       4240 non-null   int64
dtypes: float64(8), int64(7)
```

In [15]:
```
rawDataset1=rawDataset.dropna(subset=['cigsPerDay', 'BPMeds'])
```

In [16]:
```
rawDataset1.isnull().sum()
```

Out[16]:
```
male                0
age                 0
currentSmoker       0
cigsPerDay          0
BPMeds              0
prevalentStroke     0
prevalentHyp        0
diabetes            0
totChol            49
sysBP               0
diaBP               0
BMI                19
heartRate           1
glucose           382
TenYearCHD          0
dtype: int64
```

In [17]:
```
#replacing total cholestorol column missing values with mean
pd.options.mode.chained_assignment = None
mean_value_totChol=rawDataset1['totChol'].mean()
rawDataset1['totChol']=rawDataset1['totChol'].fillna(mean_value_totChol)
rawDataset1.isnull().sum()
```

Out[17]:
```
male                0
age                 0
currentSmoker       0
cigsPerDay          0
BPMeds              0
prevalentStroke     0
prevalentHyp        0
diabetes            0
totChol             0
sysBP               0
diaBP               0
BMI                19
heartRate           1
glucose           382
TenYearCHD          0
dtype: int64
```

In [18]:
```
#replacing BMI column missing values with mean
mean_value_bmi=rawDataset1['BMI'].mean()
rawDataset1['BMI']=rawDataset1['BMI'].fillna(mean_value_bmi)
rawDataset1.isnull().sum()
```

Out[18]:
```
male                0
age                 0
currentSmoker       0
cigsPerDay          0
BPMeds              0
prevalentStroke     0
prevalentHyp        0
diabetes            0
totChol             0
sysBP               0
diaBP               0
BMI                 0
heartRate           1
glucose           382
TenYearCHD          0
dtype: int64
```

In [19]:
```
#replacing Heart Rate column missing values with mean
mean_value_heartRate=rawDataset1['heartRate'].mean()
rawDataset1['heartRate']=rawDataset1['heartRate'].fillna(mean_value_heartRate)
rawDataset1.isnull().sum()
```

Out[19]:
```
male                 0
age                  0
currentSmoker        0
cigsPerDay           0
BPMeds               0
prevalentStroke      0
prevalentHyp         0
diabetes             0
totChol              0
sysBP                0
diaBP                0
BMI                  0
heartRate            0
glucose            382
TenYearCHD           0
dtype: int64
```

In [20]:
```
#replacing glucose column missing values with mean
mean_value_glucose=rawDataset1['glucose'].mean()
rawDataset1['glucose']=rawDataset1['glucose'].fillna(mean_value_glucose)
rawDataset1.isnull().sum()
```

Out[20]:
```
male               0
age                0
currentSmoker      0
cigsPerDay         0
BPMeds             0
prevalentStroke    0
prevalentHyp       0
diabetes           0
totChol            0
sysBP              0
diaBP              0
BMI                0
heartRate          0
glucose            0
TenYearCHD         0
dtype: int64
```

In [21]:
```
rawDataset1.drop_duplicates()
```

Out[21]:

	male	age	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	TenYear
0	1	39	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	80.0	77.000000	
1	0	46	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	95.0	76.000000	
2	1	48	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.34	75.0	70.000000	
3	0	61	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.58	65.0	103.000000	
4	0	46	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.10	85.0	85.000000	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4234	1	51	1	43.0	0.0	0	0	0	207.0	126.5	80.0	19.71	65.0	68.000000	
4236	0	44	1	15.0	0.0	0	0	0	210.0	126.5	87.0	19.16	86.0	81.935117	
4237	0	52	0	0.0	0.0	0	0	0	269.0	133.5	83.0	21.47	80.0	107.000000	
4238	1	40	0	0.0	0.0	0	1	0	185.0	141.0	98.0	25.60	67.0	72.000000	
4239	0	39	1	30.0	0.0	0	0	0	196.0	133.0	86.0	20.91	85.0	80.000000	

4158 rows × 15 columns

In [22]:
```
finalDataset=rawDataset1.reset_index(drop=True)
```

Feature selection

In [28]:
```
#Feature selection is to done using AZURE ML
#therefore preprocessed dataset is exported into a csv
finalDataset.to_csv(r'Preprocessed_framingham.csv', index = False)
```

alt text

Click here to see the Azure machine learning Feature selection experiment

Feature selection amounts according to pearson correlation

1. age = 0.225019
2. sysBP = 0.216287
3. prevalentHYP = 0.176191
4. diaBP = 0.14683
5. glucose = 0.120305
6. diabetes = 0.096362
7. male(gender) = 0.092329
8. BPMeds = 0.087417
9. totChol = 0.077092
10. BMI = 0.074763
11. cigsPerDay = 0.05886
12. prevalentStroke = 0.056258
13. heartRate = 0.023553
14. currentSmoker = 0.022745

In [29]:
```
finalFeaturedDataset = finalDataset[['age', 'sysBP','prevalentHyp','diaBP','glucose','diabetes','male','BPMeds','totChol','BMI','
```

In [30]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0,1))

#assign scaler to column:
finalFeaturedDataset_scaled = pd.DataFrame(scaler.fit_transform(finalFeaturedDataset), columns=finalFeaturedDataset.columns)
```

In [45]:
```
#Spliiting data into testing and training
Y = finalFeaturedDataset_scaled['TenYearCHD']
X = finalFeaturedDataset_scaled.drop(['TenYearCHD'], axis = 1)

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.25, random_state=42)
```

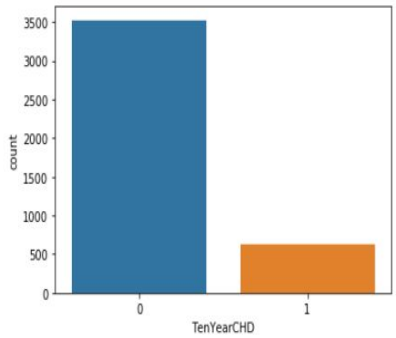
6.Data Visualization

01.Explore dataset using Jupyter Notebook

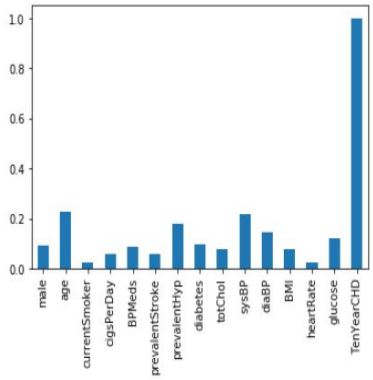
Exploratory Data Analysis

In [23]:
```
sns.countplot(x='TenYearCHD',data=finalDataset,)
```

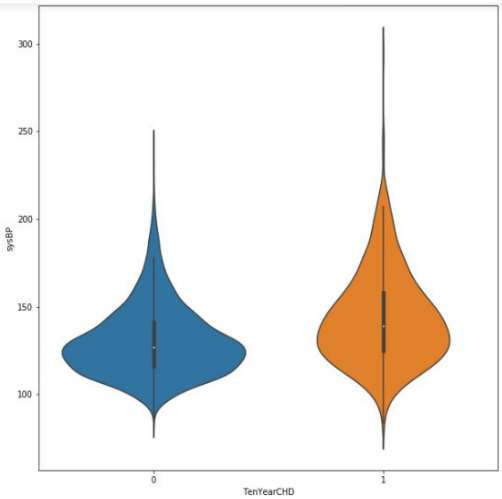
Out[23]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2656baaa688>
```



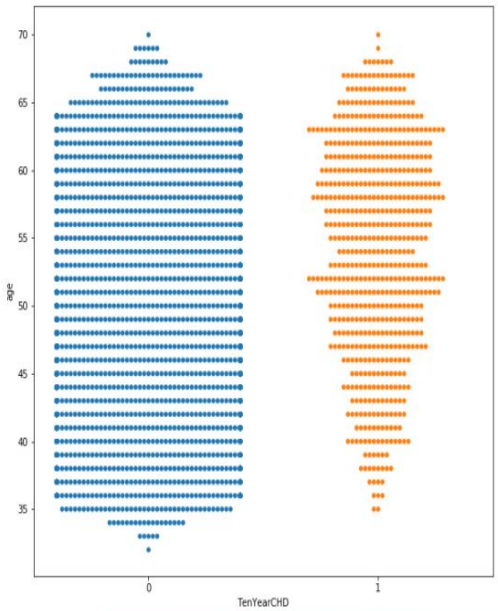
In [24]:
```
#plotting the correlation between target variable and other columns
Correlation=finalDataset.corrwith(finalDataset["TenYearCHD"])
Correlation.plot(kind='bar')
plt.show()
```



```
In [25]: #since the correlation between age and sysBP is high another plot is needed for visualization
         #violin plot for Target variable and systolic blood pressure
         plt.figure(figsize=(10,10))
         sns.violinplot(x='TenYearCHD', y='sysBP', data=finalDataset)

Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x2656da71f48>
```



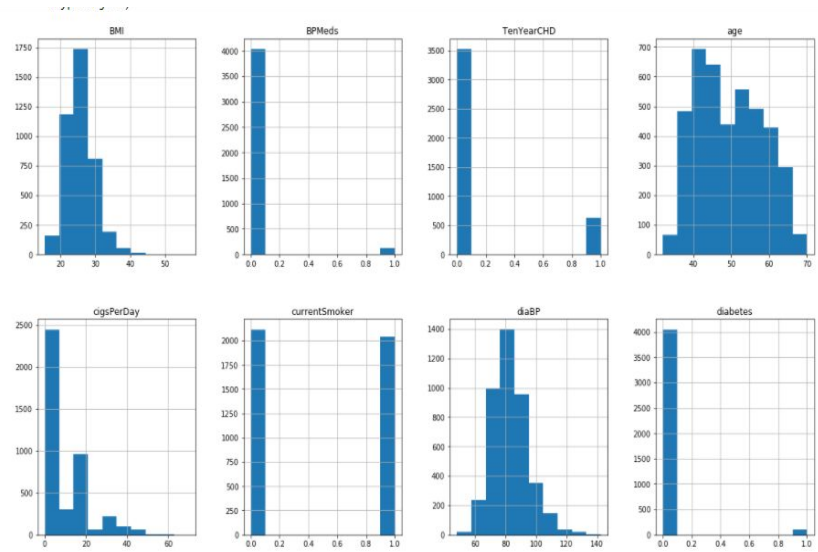

```
In [26]: plt.figure(figsize=(10,10))
         sns.swarmplot(x='TenYearCHD', y='age', data=finalDataset)

Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x2656daef848>
```

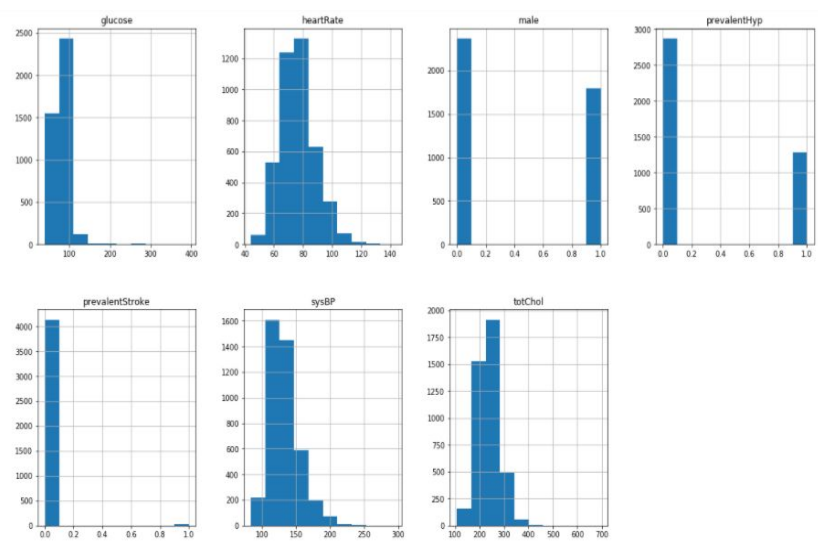


In [27]:
```
#Checking the distributions of all columns
figures = plt.figure(figsize = (20,25))
axis = figures.gca()
finalDataset.hist(ax = axis)
```

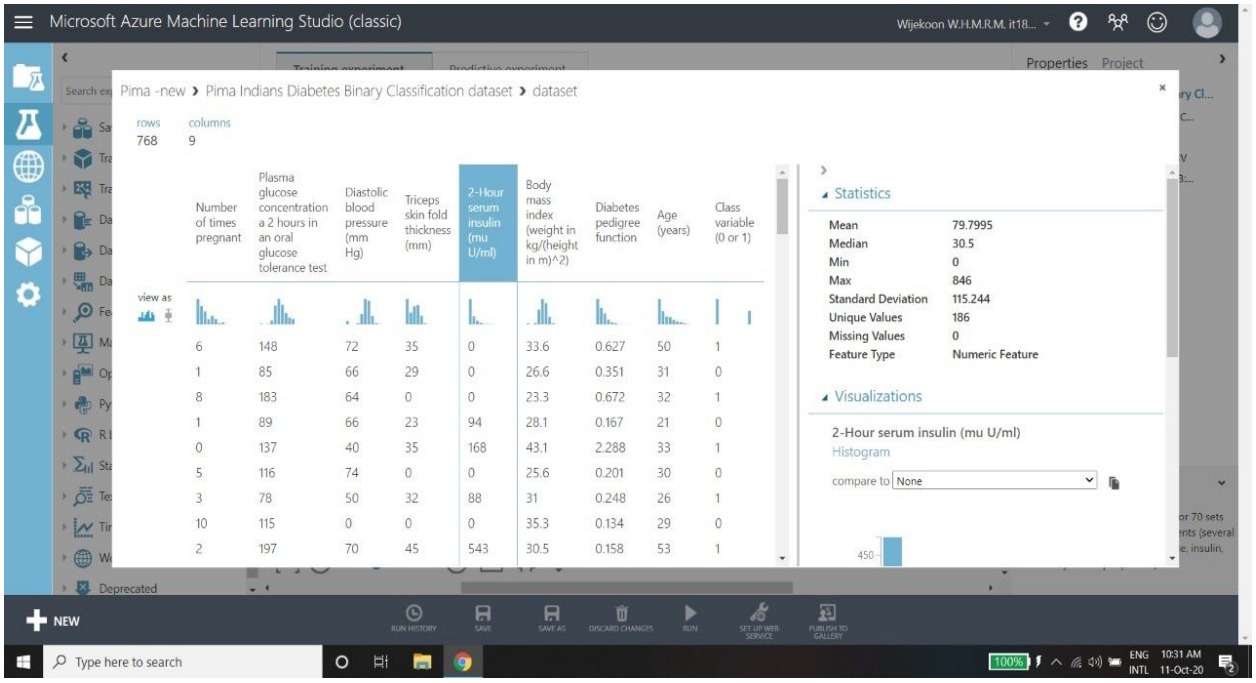
```
C:\Users\User\anaconda3\lib\site-packages\ipykernel_launcher.py:4: UserWarning: To output multiple subplots, the figure contain
ing the passed axes is being cleared
  after removing the cwd from sys.path.
```

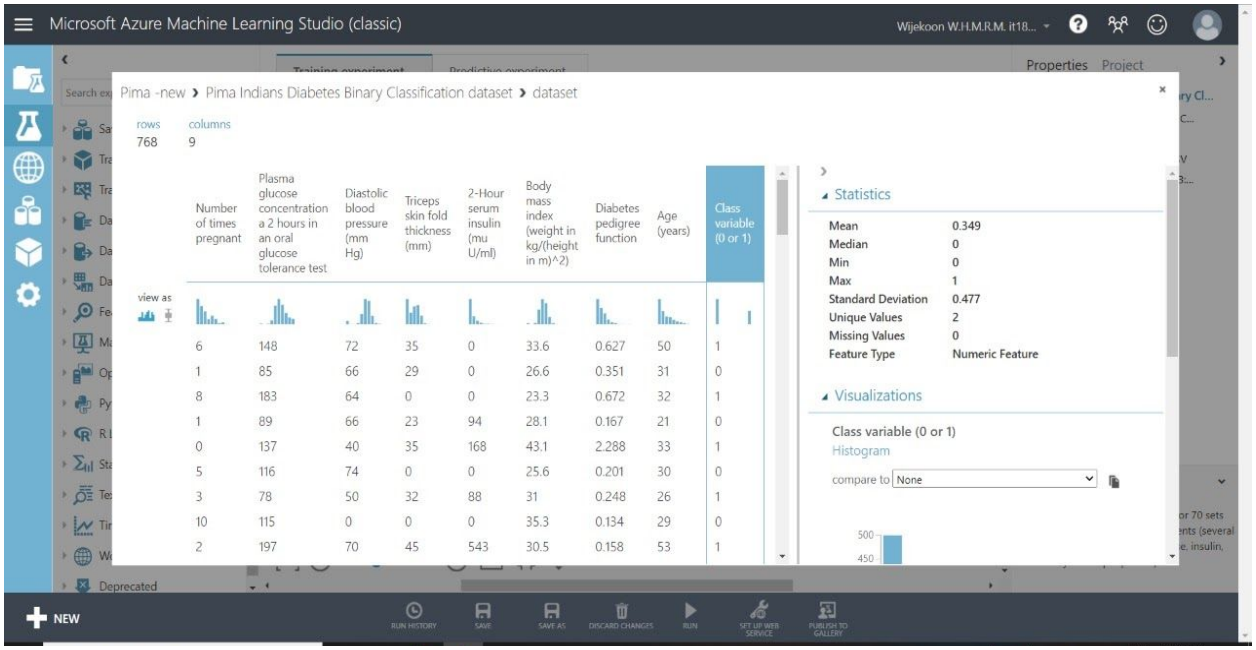
Out[27]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000002656BDB0748>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002656D9FBDC8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002656DAFDCC8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002656DB36DC8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000002656DB6EF08>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002656DBA7208>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002656DBE0308>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002656DC1EF08>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000002656DC1EFC8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002656DC5E208>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002656DCC36C8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002656E02B808>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000002656E062908>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002656E09CA08>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002656E0D6B48>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002656E10FBC8>]],
      dtype=object)
```





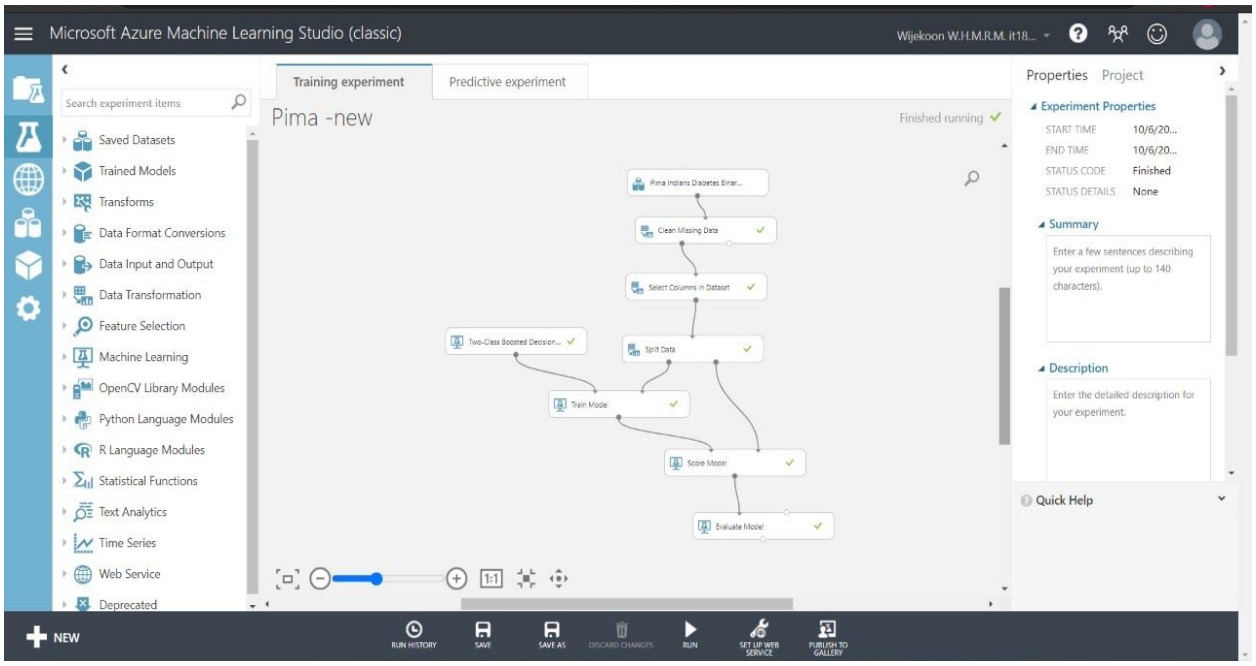
02.Explore dataset using Azure machine Learning

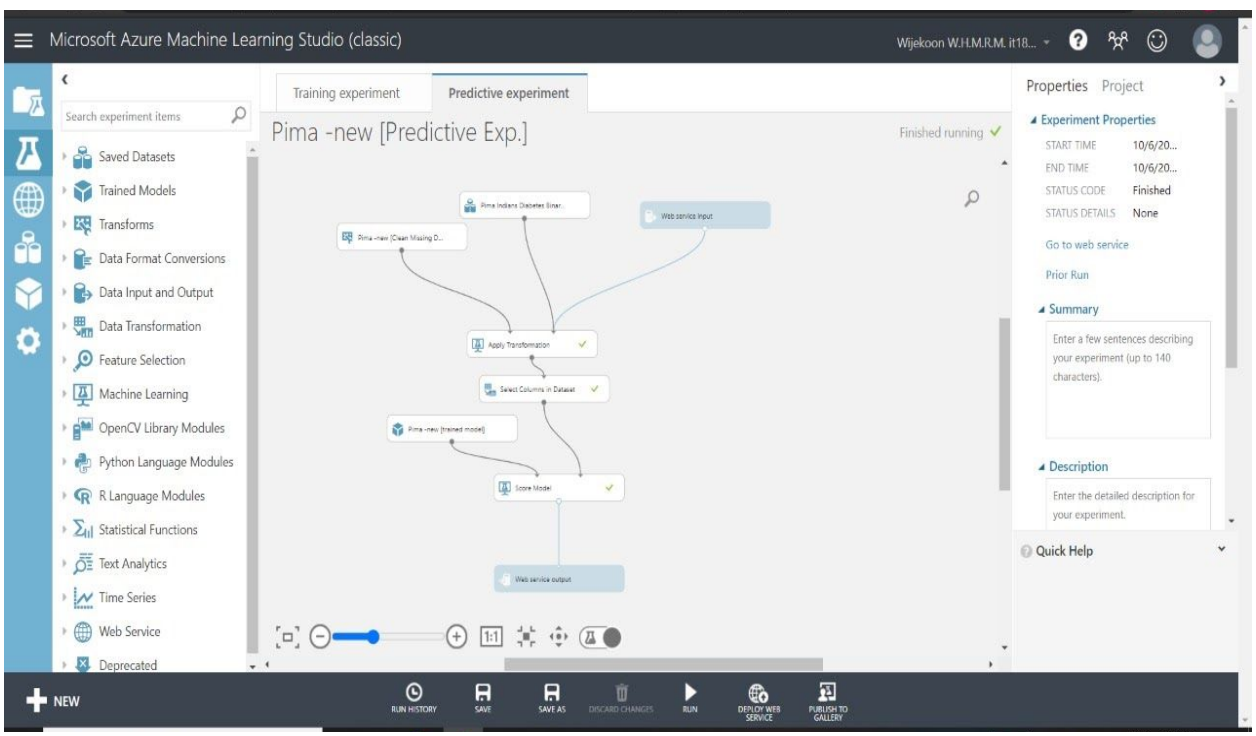




7.Model Building

01.Build a classifier using dataset to train the models





01.Decision Tree

Decision tree

In [46]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [47]:
```
clf = DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=0)

clf.fit(X_train, y_train)
```

Out[47]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=3, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=0, splitter='best')
```

In [48]:
```
y_pred_gini = clf.predict(X_test)
```

In [49]:
```
from sklearn.metrics import accuracy_score

print('Model accuracy score with decision tree classifier: ', accuracy_score(y_test, y_pred_gini))
```

```
Model accuracy score with decision tree classifier:  0.8394230769230769
```

02.Logistic Regression

Logistic Regression

In [50]:
```
from sklearn.linear_model import LogisticRegression
logreg=LogisticRegression()
logreg.fit(X_train,y_train)
y_pred1 =logreg.predict(X_test)


from sklearn import metrics

print('Model accuracy score with logistic regression classifier: ', metrics.accuracy_score(y_test,y_pred1))
#print(metrics.classification_report(y_test,y_pred)),metrics.confusion_matrix(y_test,y_pred)
```

```
Model accuracy score with logistic regression classifier:  0.8432692307692308
```

03.Naive Bayes

Naive Bayes

In [51]:
```
#Import Gaussian Naive Bayes model
from sklearn.naive_bayes import GaussianNB

#Create a Gaussian Classifier
gnb = GaussianNB()

#Train the model using the training sets
gnb.fit(X_train, y_train)

#Predict the response for test dataset
y_pred2 = gnb.predict(X_test)

#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred2))
```

```
Accuracy: 0.8105769230769231
```

04.KNeighborsClassifier

KNeighborsClassifier

In [52]:
```
from sklearn.neighbors import KNeighborsClassifier

from sklearn import metrics

Range_k = range(1,15)
scores = {}
scores_list = []
for k in Range_k:
   classifier = KNeighborsClassifier(n_neighbors=k)
   classifier.fit(X_train, y_train)
   y_pred3 = classifier.predict(X_test)
   scores[k] = metrics.accuracy_score(y_test,y_pred3)
   scores_list.append(metrics.accuracy_score(y_test,y_pred3))



print("Accuracy:",metrics.accuracy_score(y_test, y_pred3))
```

```
Accuracy: 0.8365384615384616
```

05.Random Forest Classifier

RandomForestClassifier

In [53]:
```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 50)
classifier.fit(X_train, y_train)

y_pred4 = classifier.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score


print("Accuracy:",accuracy_score(y_test,y_pred4))
```

```
Accuracy: 0.8355769230769231
```

06.Support Vector Classifier

Support vector classifier

In [54]:
```
from sklearn.svm import SVC

model = SVC(kernel='rbf', C=1, gamma=0.001, random_state=1)
model.fit(X_train, y_train)

from sklearn.metrics import accuracy_score

y_predict = model.predict(X_test)
print("Accuracy:" ,accuracy_score(y_test, y_predict))
```

```
Accuracy: 0.8384615384615385
```

02.Conclusion

- After calculating accuracy of each model, the best model for predicting the most accurate result is the logistic regression algorithm.

- Therefore we use this logistic regression algorithm for further activities.

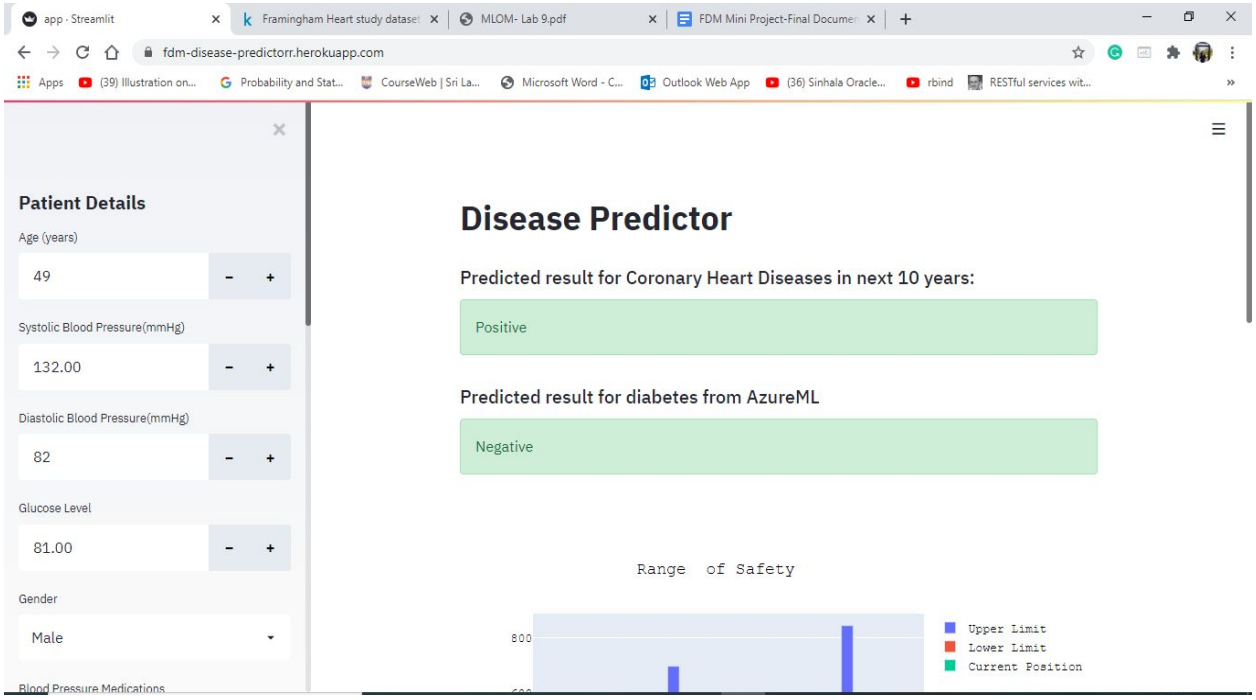
Logistic regression is the best algorithm to train

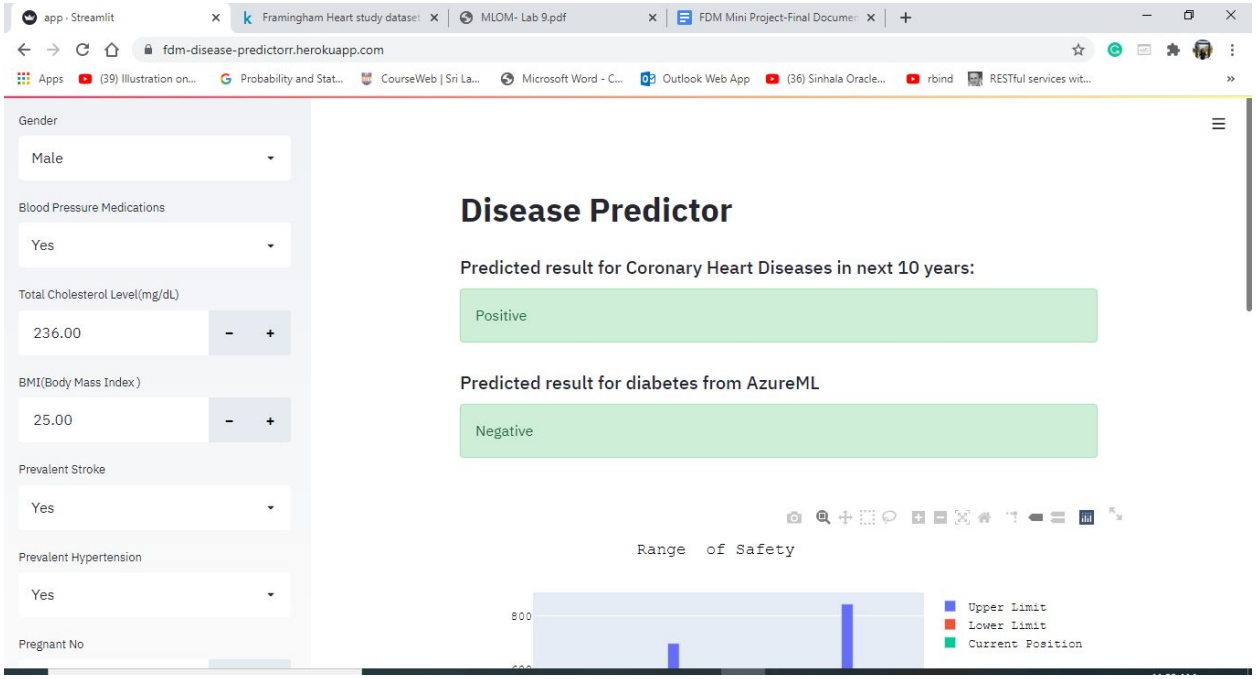
Therefore logistic regeression is used for further activities

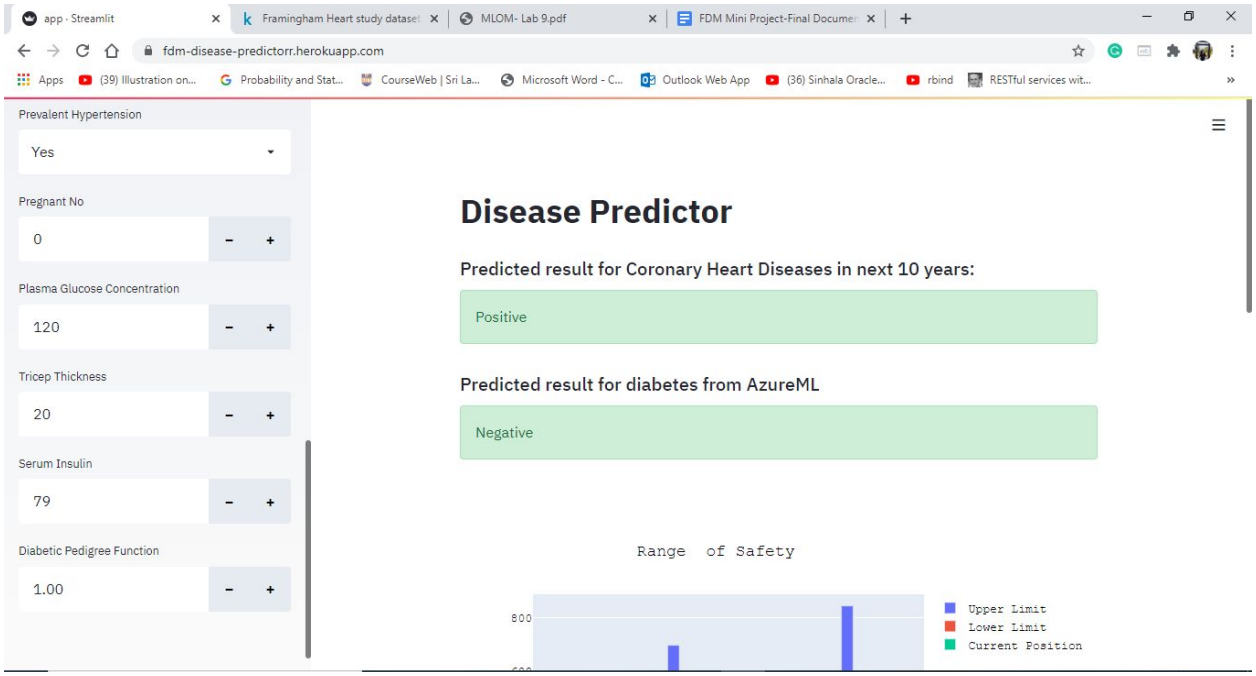
In [64]:
```
#Save the logistic regression strained modelusing pickle
import pickle
with open('framingham_classifier_logistic_regression', 'wb') as picklefile:
    pickle.dump(logreg,picklefile)
```

08.Front End Development

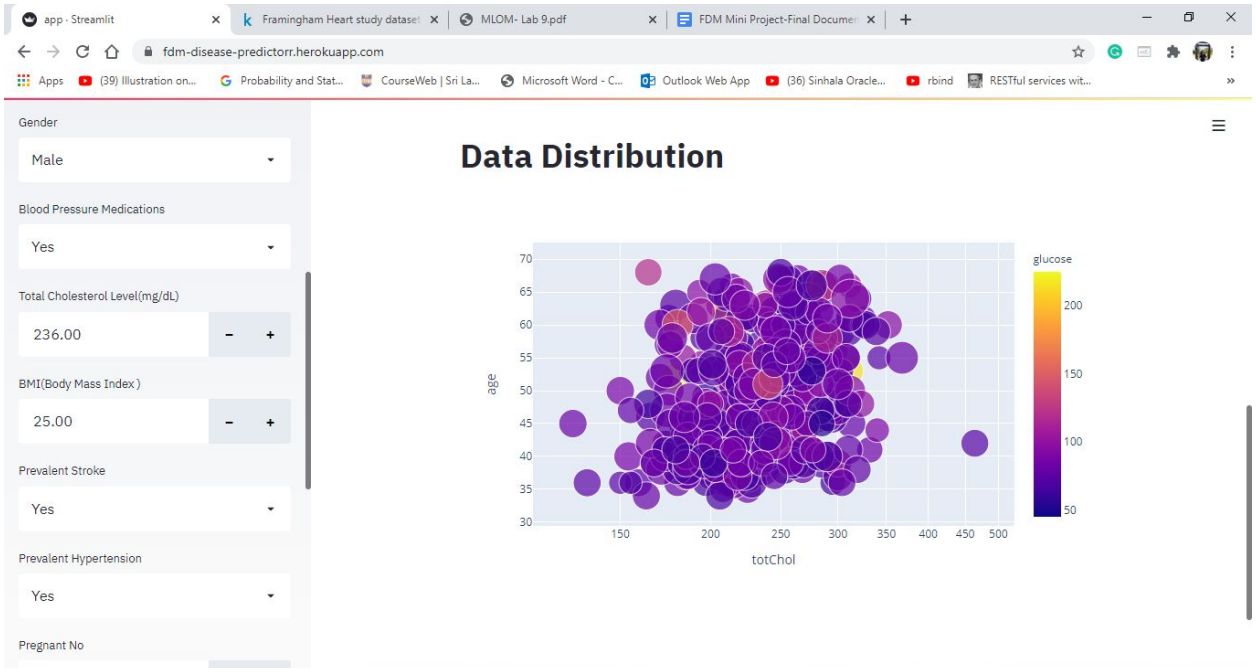
01.Interface of the System






app · Streamlit
Framingham Heart study dataset
MLOM- Lab 9.pdf
FDM Mini Project-Final Docume
fdm-disease-predictorr.herokuapp.com
Prevalent Hypertension
Yes
Pregnant No
0
Plasma Glucose Concentration
120
Tricep Thickness
20
Serum Insulin
79
Diabetic Pedigree Function
1.00
Disease Predictor
Predicted result for Coronary Heart Diseases in next 10 years:
Positive
Predicted result for diabetes from AzureML
Negative
Range of Safety
800
Upper Limit
Lower Limit
Current Position


app · Streamlit
Framingham Heart study dataset
MLOM- Lab 9.pdf
FDM Mini Project-Final Docume
fdm-disease-predictorr.herokuapp.com
Patient Details
Age (years)
49
Systolic Blood Pressure(mmHg)
132.00
Diastolic Blood Pressure(mmHg)
82
Glucose Level
81.00
Range of Safety
800
600
400
200
0
Upper Limit
Lower Limit
Current Position
sysBP
diaBP
glucose
totChol
BMI
plasmaGlucoseConc
tricepsThickness
SerumInsulin
diabPedigreeFunc


app · Streamlit
Framingham Heart study dataset
MLOM- Lab 9.pdf
FDM Mini Project-Final Docume
fdm-disease-predictorr.herokuapp.com
Gender
Male
Blood Pressure Medications
Yes
Total Cholesterol Level(mg/dL)
236.00
BMI(Body Mass Index)
25.00
Prevalent Stroke
Yes
Prevalent Hypertension
Yes
Pregnant No
Data Distribution
glucose
age
totChol

09.Deliverables

In our final output , we present our prediction model as software.so it has a smooth interface and users can easily enter the relevant attributes to get the final prediction. Our App link is mentioned below.

https://fdm-disease-predictorr.herokuapp.com/

10.Group Contribution

Index Number	**Name**
IT18154054	Wijekoon W.H.M.R.M.
IT18058024	K.S.Koralage
IT18024814	B.G.K.Madushani
IT18116366	Wijekoon R.T.P.W.W.M.P.M.B.
IT18006230	Perera T.R.D.
IT18003574	Ekanayaka W.E.M.D.S.
IT18157888	Chamodya Lakshani A.K.Y