AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Choose an item.

Assignment Cover Sheet

Project Title:	***Building Disease Detection Classification Models using Python***		
Term :	**Final term**	Date of Submission:	***14 December 2022***
Course Title:	**PROGRAMMING IN PYTHON [B]**		
Course Code:	*CSC4162*	Section:	**B**
Semester:	Fall 22-23	Course Teacher:	**AKINUL ISLAM JONY**

Declaration and Statement of Authorship:

1. I/we hold a copy of this Assignment/Case-Study, which can be produced if the original is lost/damaged.
2. This Assignment/Case-Study is my/our original work and no part of it has been copied from any other student's work or from any other source except where due acknowledgement is made.
3. No part of this Assignment/Case-Study has been written for me/us by any other person except where such collaborationhas been authorized by the concerned teacher and is clearly acknowledged in the assignment.
4. I/we have not previously submitted or currently submitting this work for any other course/unit.
5. This work may be reproduced, communicated, compared and archived for the purpose of detecting plagiarism.
6. I/we give permission for a copy of my/our marked work to be retained by the Faculty for review and comparison, including review by external examiners.
7. I/we understand thatPlagiarism is the presentation of the work, idea or creation of another person as though it is your own. It is a formofcheatingandisaveryseriousacademicoffencethatmayleadtoexpulsionfromtheUniversity. Plagiarized material can be drawn from, and presented in, written, graphic and visual form, including electronic data, and oral presentations. Plagiarism occurs when the origin of them arterial used is not appropriately cited.
8. I/we also understand that enabling plagiarism is the act of assisting or allowing another person to plagiarize or to copy my/our work.

* *Student(s) must complete all details except the faculty use part.*
** Please submit all assignments to your course teacher or the office of the concerned teacher.

Group Name/No.:

No	Name	ID	Program	Signature
1	Rifat Hossain	20-42461-1	Bsc CSE	Rifat

Faculty use only

FACULTYCOMMENTS	**Marks Obtained**	
	Total Marks	

Building Disease Detection Classification Models using Python

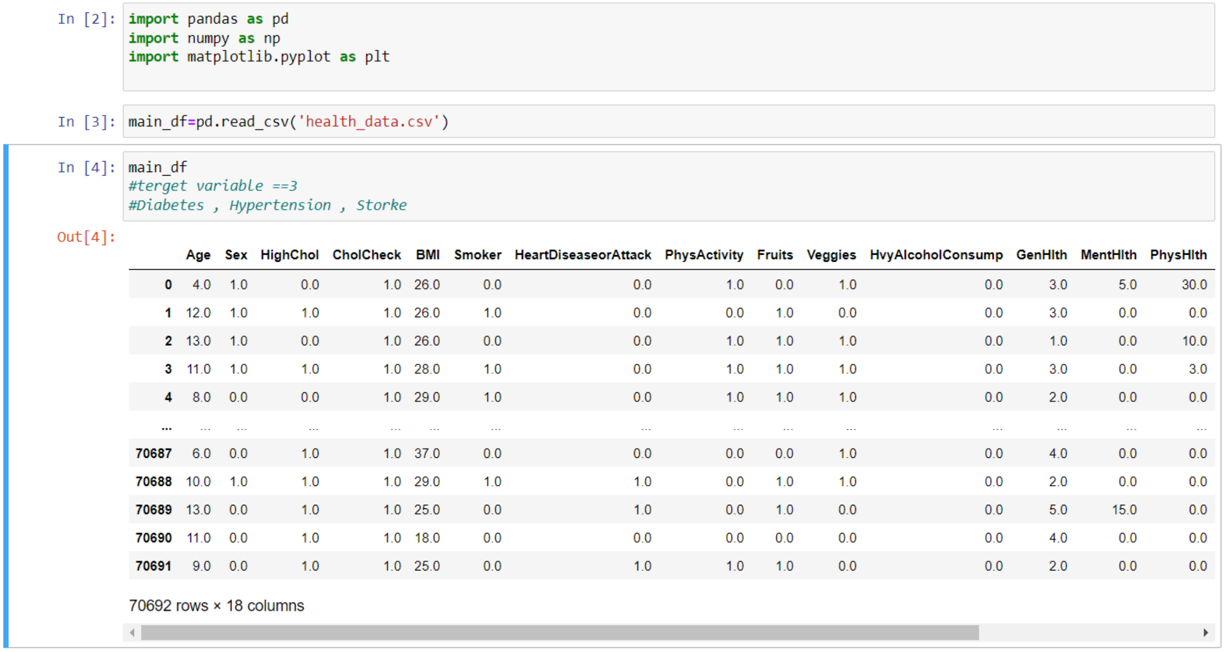
Summary:

Python is well known for its use in data science and machine learning work. In this project various medical data set has been collected from online source such as kiggle where total number of entry was (70692 rows × 18 columns), among them 15 columns were features and 3 target variable. In this work I built classification based machine learning models and finally compair them with each other. These target variable such as diabetes ,hypertension and stroke detection was targeted separately thus every model has 3 detection versions .This work consist of data cleaning ,preprocessing,machine learning based classification model building.

Dataset:

The data set was collected from kiggle named "Diabetes, Hypertension and Stroke Prediction". It is a CSV file formet data consist of 70692 rows and 18 columns, among them 3 target variables such as diabetes , hypertension, stroke. This data set does not contain any null value.

Link of data-set: **https://www.kaggle.com/datasets/prosperchuks/health-dataset**



```
In [2]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
```

```
In [3]: main_df=pd.read_csv('health_data.csv')
```

```
In [4]: main_df
        #terget variable ==3
        #Diabetes , Hypertension , Storke
```

Out[4]:

	Age	Sex	HighChol	CholCheck	BMI	Smoker	HeartDiseaseorAttack	PhysActivity	Fruits	Veggies	HvyAlcoholConsump	GenHlth	MentHlth	PhysHlth
0	4.0	1.0	0.0	1.0	26.0	0.0	0.0	1.0	0.0	1.0	0.0	3.0	5.0	30.0
1	12.0	1.0	1.0	1.0	26.0	1.0	0.0	0.0	1.0	0.0	0.0	3.0	0.0	0.0
2	13.0	1.0	0.0	1.0	26.0	0.0	0.0	1.0	1.0	1.0	0.0	1.0	0.0	10.0
3	11.0	1.0	1.0	1.0	28.0	1.0	0.0	1.0	1.0	1.0	0.0	3.0	0.0	3.0
4	8.0	0.0	0.0	1.0	29.0	1.0	0.0	1.0	1.0	1.0	0.0	2.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
70687	6.0	0.0	1.0	1.0	37.0	0.0	0.0	0.0	0.0	1.0	0.0	4.0	0.0	0.0
70688	10.0	1.0	1.0	1.0	29.0	1.0	1.0	0.0	1.0	1.0	0.0	2.0	0.0	0.0
70689	13.0	0.0	1.0	1.0	25.0	0.0	1.0	0.0	1.0	0.0	0.0	5.0	15.0	0.0
70690	11.0	0.0	1.0	1.0	18.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0	0.0	0.0
70691	9.0	0.0	1.0	1.0	25.0	0.0	1.0	1.0	1.0	0.0	0.0	2.0	0.0	0.0

70692 rows × 18 columns

Preprocessing and Cleaning:

Since there were no error value, or empty values , no row has to be dropped from data frame. A correlation matrix was plotted but all the values were average so no need the change the data frame.

In [5]:
```
main_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70692 entries, 0 to 70691
Data columns (total 18 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Age                   70692 non-null  float64
 1   Sex                   70692 non-null  float64
 2   HighChol              70692 non-null  float64
 3   CholCheck             70692 non-null  float64
 4   BMI                   70692 non-null  float64
 5   Smoker                70692 non-null  float64
 6   HeartDiseaseorAttack  70692 non-null  float64
 7   PhysActivity          70692 non-null  float64
 8   Fruits                70692 non-null  float64
 9   Veggies               70692 non-null  float64
 10  HvyAlcoholConsump     70692 non-null  float64
 11  GenHlth               70692 non-null  float64
 12  MentHlth              70692 non-null  float64
 13  PhysHlth              70692 non-null  float64
 14  DiffWalk              70692 non-null  float64
 15  Diabetes              70692 non-null  float64
 16  Hypertension          70692 non-null  float64
 17  Stroke                70692 non-null  float64
dtypes: float64(18)
memory usage: 9.7 MB
```

In [6]:
```
main_df.describe()
#here we can see Max BMI is 98 ,unfortunatly it is possible to have a bmi of 98.
```

Out[6]:

	Age	Sex	HighChol	CholCheck	BMI	Smoker	HeartDiseaseorAttack	PhysActivity	Fruits	Veggies
count	70692.000000	70692.000000	70692.000000	70692.000000	70692.000000	70692.000000	70692.000000	70692.000000	70692.000000	70692.000000
mean	8.584055	0.456997	0.525703	0.975259	29.856985	0.475273	0.147810	0.703036	0.611795	0.788774
std	2.852153	0.498151	0.499342	0.155336	7.113954	0.499392	0.354914	0.456924	0.487345	0.408181
min	1.000000	0.000000	0.000000	0.000000	12.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	7.000000	0.000000	0.000000	1.000000	25.000000	0.000000	0.000000	0.000000	0.000000	1.000000
50%	9.000000	0.000000	1.000000	1.000000	29.000000	0.000000	0.000000	1.000000	1.000000	1.000000
75%	11.000000	1.000000	1.000000	1.000000	33.000000	1.000000	0.000000	1.000000	1.000000	1.000000
max	13.000000	1.000000	1.000000	1.000000	98.000000	1.000000	1.000000	1.000000	1.000000	1.000000

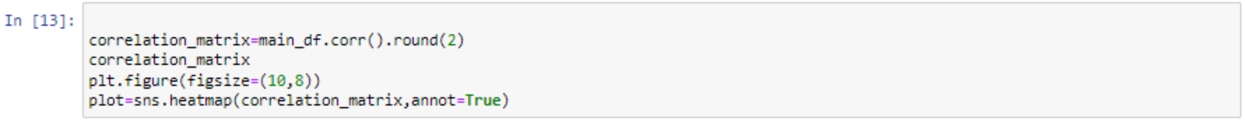
In [7]:
```
#checking null
check1=main_df.isnull().sum()
```

In [8]:
```
check1 #no null values on this dataset
```

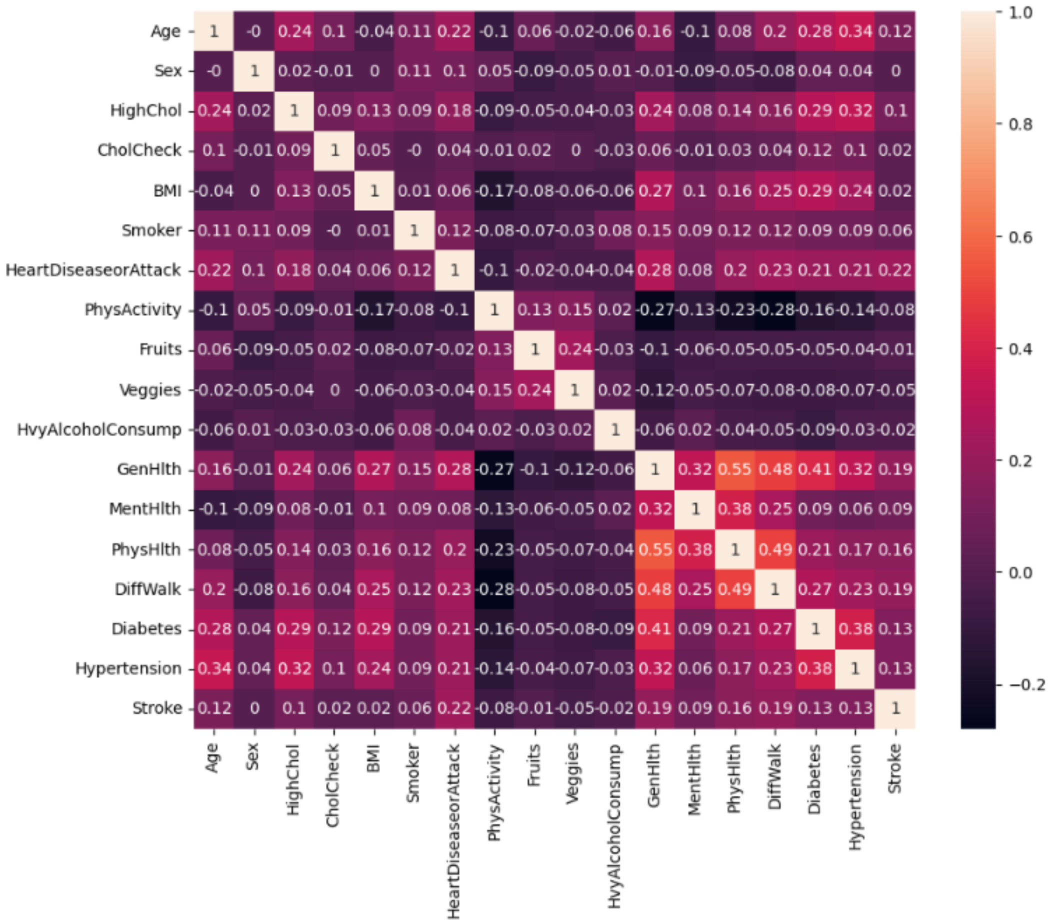
Out[8]:
```
Age                     0
Sex                     0
HighChol                0
CholCheck               0
BMI                     0
Smoker                  0
HeartDiseaseorAttack    0
PhysActivity            0
Fruits                  0
Veggies                 0
HvyAlcoholConsump       0
GenHlth                 0
MentHlth                0
PhysHlth                0
DiffWalk                0
Diabetes                0
Hypertension            0
Stroke                  0
dtype: int64
```

```
main_df.columns
```

```
Index(['Age', 'Sex', 'HighChol', 'CholCheck', 'BMI', 'Smoker',
       'HeartDiseaseorAttack', 'PhysActivity', 'Fruits', 'Veggies',
       'HvyAlcoholConsump', 'GenHlth', 'MentHlth', 'PhysHlth', 'DiffWalk',
       'Diabetes', 'Hypertension', 'Stroke'],
      dtype='object')
```



```
In [13]:
correlation_matrix=main_df.corr().round(2)
correlation_matrix
plt.figure(figsize=(10,8))
plot=sns.heatmap(correlation_matrix,annot=True)
```



Algorithm:

For each target variable various models were fited such as Logistic Regression, KNN, Naive biyas, Decision tree, and SVM were used. These models were imported from sklearn library.Also these data were splitted using train_test_split function from skllearn. After fitting these data a score was collected.

Model and Code:

For every model data selt split into 90% train and 10% test, data set also divided into 2 groups such as features and target group.

Logistic Regression:

```
In [14]: from sklearn.model_selection import train_test_split
```

```
In [15]: target.columns
```

```
Out[15]: Index(['Diabetes', 'Hypertension', 'Stroke'], dtype='object')
```

```
In [16]: features.columns
```

```
Out[16]: Index(['Age', 'Sex', 'HighChol', 'CholCheck', 'BMI', 'Smoker',
                'HeartDiseaseorAttack', 'PhysActivity', 'Fruits', 'Veggies',
                'HvyAlcoholConsump', 'GenHlth', 'MentHlth', 'PhysHlth', 'DiffWalk'],
               dtype='object')
```

```
In [17]: x_train,x_test,y_train,y_test= train_test_split(main_df[['Age', 'Sex', 'HighChol', 'CholCheck', 'BMI', 'Smoker',
                'HeartDiseaseorAttack', 'PhysActivity', 'Fruits', 'Veggies',
                'HvyAlcoholConsump', 'GenHlth', 'MentHlth', 'PhysHlth', 'DiffWalk']],target.Diabetes,test_size=0.1)
```

```
In [18]: x_train.head()
```

Out[18]:

	Age	Sex	HighChol	CholCheck	BMI	Smoker	HeartDiseaseorAttack	PhysActivity	Fruits	Veggies	HvyAlcoholConsump	GenHlth	MentHlth	PhysHlth
18809	3.0	0.0	0.0	0.0	25.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	12.0	0.0
934	8.0	1.0	1.0	1.0	38.0	1.0	0.0	1.0	0.0	1.0	0.0	3.0	0.0	20.0
4961	4.0	0.0	0.0	1.0	28.0	0.0	0.0	1.0	0.0	1.0	0.0	2.0	0.0	0.0
23361	5.0	0.0	0.0	1.0	26.0	1.0	0.0	1.0	1.0	1.0	0.0	2.0	2.0	0.0
3472	8.0	0.0	0.0	1.0	28.0	0.0	0.0	1.0	0.0	1.0	0.0	2.0	0.0	0.0

```
In [19]: from sklearn.linear_model import LogisticRegression
```

```
In [19]: from sklearn.linear_model import LogisticRegression
```

```
In [20]: model_1_logistic=LogisticRegression()
         model_1_logistic.fit(x_train,y_train)
```

```
E:\lib\site-packages\sklearn\linear_model\_logistic.py:444: 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
  n_iter_i = _check_optimize_result(
```

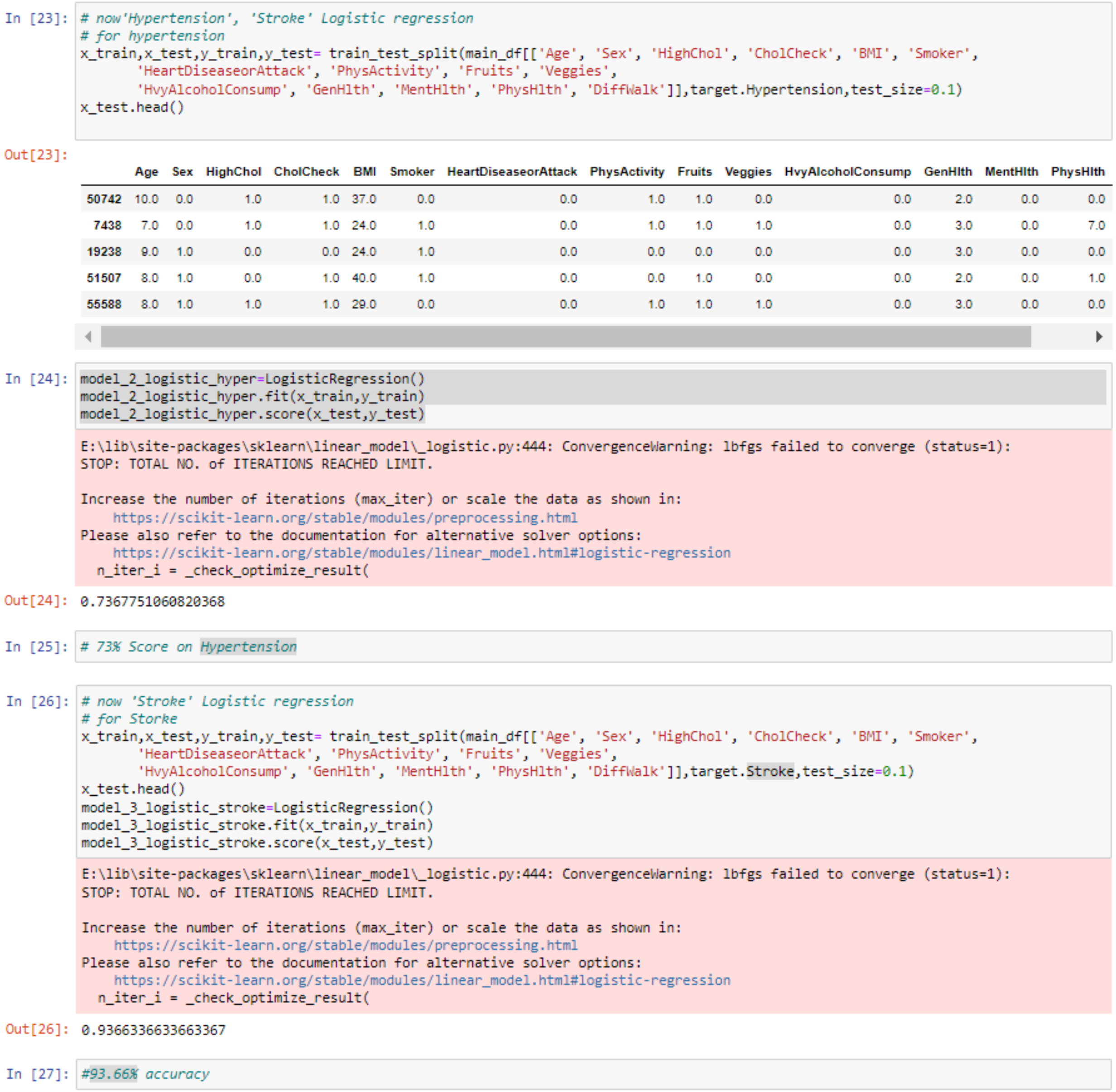
```
Out[20]: ▾ LogisticRegression
         LogisticRegression()
```

```
In [21]: #prediction for Diabetes using Logistic regression.
         model_1_logistic.score(x_test,y_test)
```

```
Out[21]: 0.746958981612447
```

```
In [22]: model_1_logistic.predict(x_test)    # Score= 74.69 %
```

```
Out[22]: array([0., 0., 0., ..., 0., 1., 0.])
```



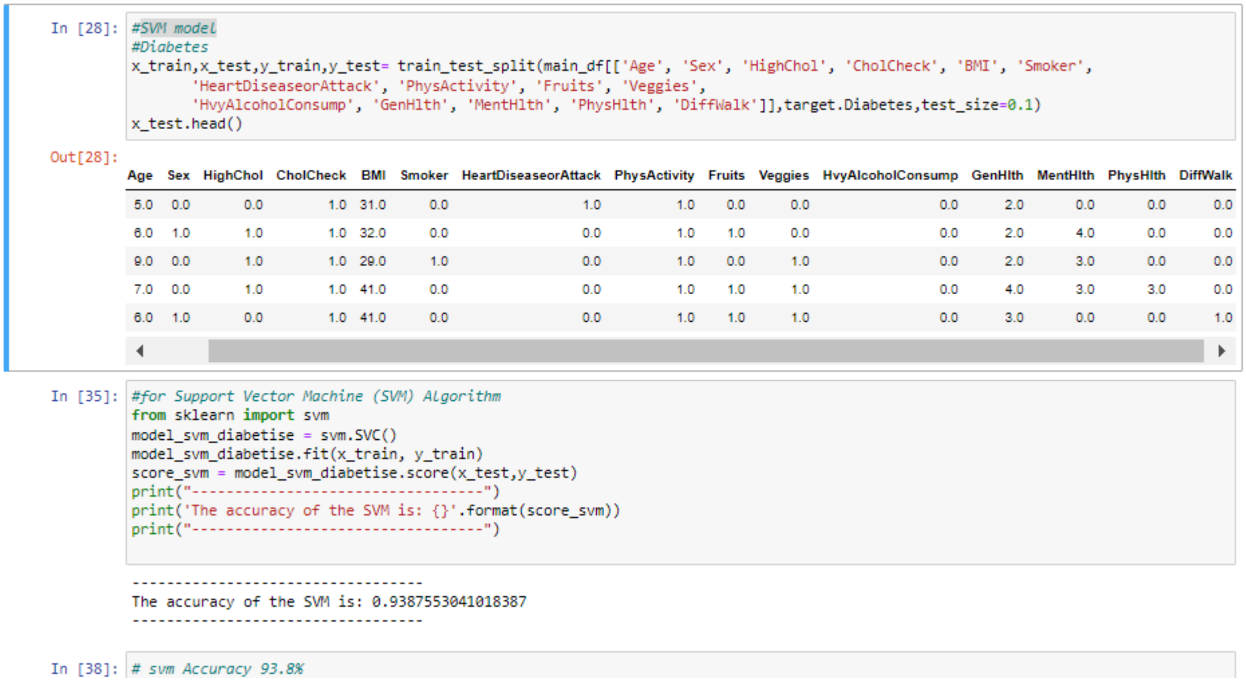
```
In [23]: # now'Hypertension', 'Stroke' Logistic regression
         # for hypertension
         x_train,x_test,y_train,y_test= train_test_split(main_df[['Age', 'Sex', 'HighChol', 'CholCheck', 'BMI', 'Smoker',
                'HeartDiseaseorAttack', 'PhysActivity', 'Fruits', 'Veggies',
                'HvyAlcoholConsump', 'GenHlth', 'MentHlth', 'PhysHlth', 'DiffWalk']],target.Hypertension,test_size=0.1)
         x_test.head()
```

Out[23]:

	Age	Sex	HighChol	CholCheck	BMI	Smoker	HeartDiseaseorAttack	PhysActivity	Fruits	Veggies	HvyAlcoholConsump	GenHlth	MentHlth	PhysHlth
50742	10.0	0.0	1.0	1.0	37.0	0.0	0.0	1.0	1.0	0.0	0.0	2.0	0.0	0.0
7438	7.0	0.0	1.0	1.0	24.0	1.0	0.0	1.0	1.0	1.0	0.0	3.0	0.0	7.0
19238	9.0	1.0	0.0	0.0	24.0	1.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0
51507	8.0	1.0	0.0	1.0	40.0	1.0	0.0	0.0	1.0	0.0	0.0	2.0	0.0	1.0
55588	8.0	1.0	1.0	1.0	29.0	0.0	0.0	1.0	1.0	1.0	0.0	3.0	0.0	0.0

```
In [24]: model_2_logistic_hyper=LogisticRegression()
         model_2_logistic_hyper.fit(x_train,y_train)
         model_2_logistic_hyper.score(x_test,y_test)

         E:\lib\site-packages\sklearn\linear_model\_logistic.py:444: 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
           n_iter_i = _check_optimize_result(

Out[24]: 0.7367751060820368

In [25]: # 73% Score on Hypertension

In [26]: # now 'Stroke' Logistic regression
         # for Storke
         x_train,x_test,y_train,y_test= train_test_split(main_df[['Age', 'Sex', 'HighChol', 'CholCheck', 'BMI', 'Smoker',
                'HeartDiseaseorAttack', 'PhysActivity', 'Fruits', 'Veggies',
                'HvyAlcoholConsump', 'GenHlth', 'MentHlth', 'PhysHlth', 'DiffWalk']],target.Stroke,test_size=0.1)
         x_test.head()
         model_3_logistic_stroke=LogisticRegression()
         model_3_logistic_stroke.fit(x_train,y_train)
         model_3_logistic_stroke.score(x_test,y_test)

         E:\lib\site-packages\sklearn\linear_model\_logistic.py:444: 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
           n_iter_i = _check_optimize_result(

Out[26]: 0.9366336633663367

In [27]: #93.66% accuracy
```

SVM model



In [28]:
```
#SVM model
#Diabetes
x_train,x_test,y_train,y_test= train_test_split(main_df[['Age', 'Sex', 'HighChol', 'CholCheck', 'BMI', 'Smoker',
       'HeartDiseaseorAttack', 'PhysActivity', 'Fruits', 'Veggies',
       'HvyAlcoholConsump', 'GenHlth', 'MentHlth', 'PhysHlth', 'DiffWalk']],target.Diabetes,test_size=0.1)
x_test.head()
```

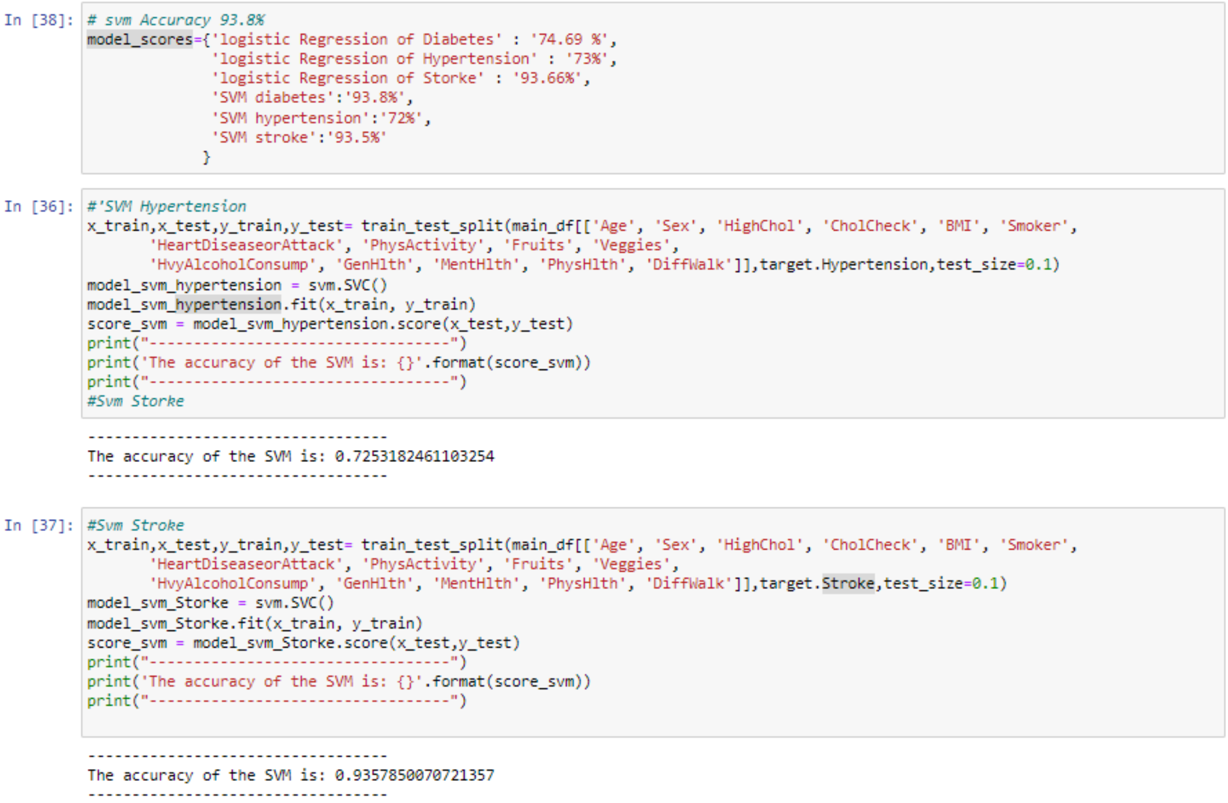
Out[28]:

Age	Sex	HighChol	CholCheck	BMI	Smoker	HeartDiseaseorAttack	PhysActivity	Fruits	Veggies	HvyAlcoholConsump	GenHlth	MentHlth	PhysHlth	DiffWalk
5.0	0.0	0.0	1.0	31.0	0.0	1.0	1.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0
6.0	1.0	1.0	1.0	32.0	0.0	0.0	1.0	1.0	0.0	0.0	2.0	4.0	0.0	0.0
9.0	0.0	1.0	1.0	29.0	1.0	0.0	1.0	0.0	1.0	0.0	2.0	3.0	0.0	0.0
7.0	0.0	1.0	1.0	41.0	0.0	0.0	1.0	1.0	1.0	0.0	4.0	3.0	3.0	0.0
6.0	1.0	0.0	1.0	41.0	0.0	0.0	1.0	1.0	1.0	0.0	3.0	0.0	0.0	1.0

In [35]:
```
#for Support Vector Machine (SVM) Algorithm
from sklearn import svm
model_svm_diabetise = svm.SVC()
model_svm_diabetise.fit(x_train, y_train)
score_svm = model_svm_diabetise.score(x_test,y_test)
print("---------------------------------")
print('The accuracy of the SVM is: {}'.format(score_svm))
print("---------------------------------")
```

```
---------------------------------
The accuracy of the SVM is: 0.9387553041018387
---------------------------------
```

In [38]:
```
# svm Accuracy 93.8%
```



In [38]:
```
# svm Accuracy 93.8%
model_scores={'logistic Regression of Diabetes' : '74.69 %',
              'logistic Regression of Hypertension' : '73%',
              'logistic Regression of Storke' : '93.66%',
              'SVM diabetes':'93.8%',
              'SVM hypertension':'72%',
              'SVM stroke':'93.5%'
             }
```

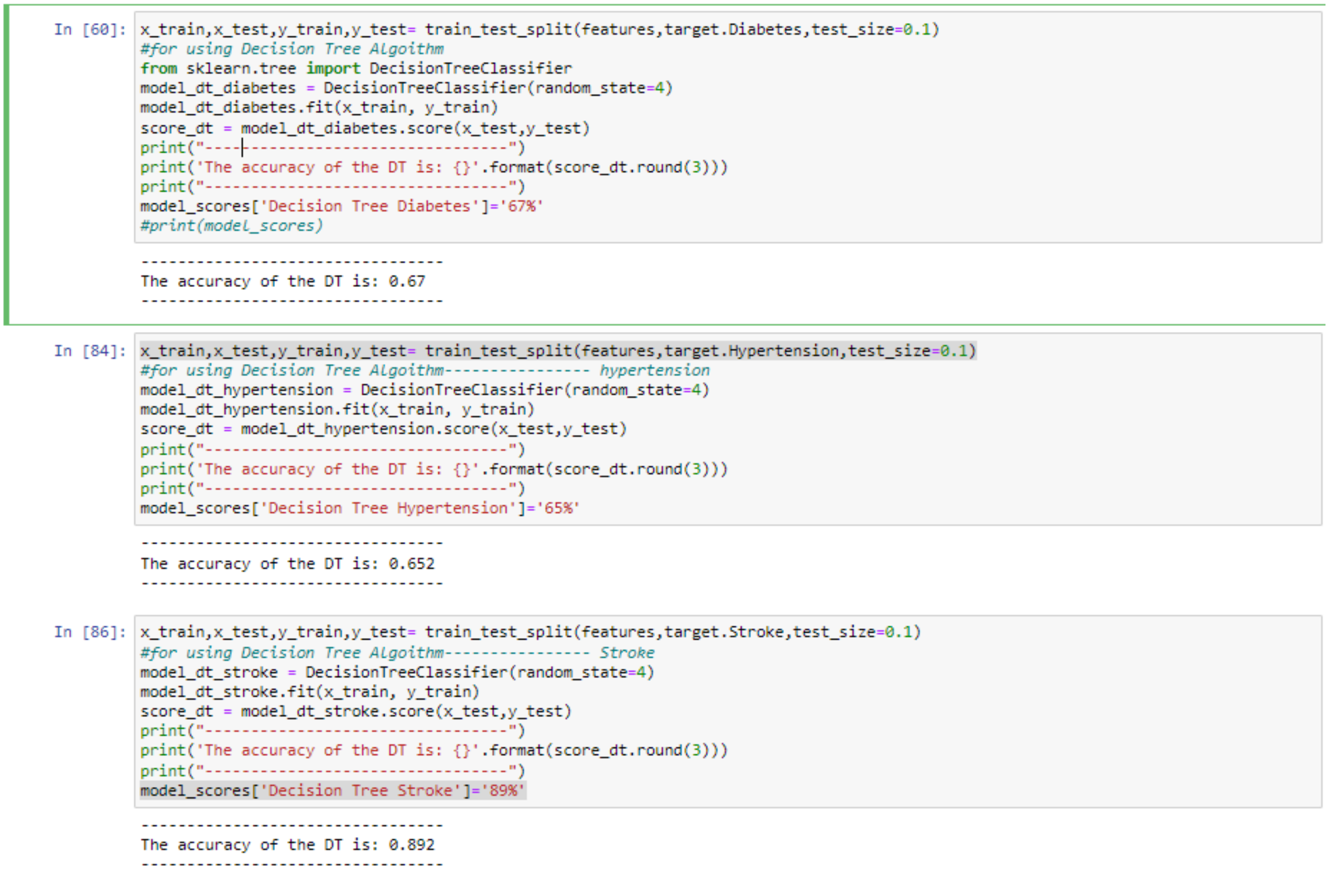
In [36]:
```
#'SVM Hypertension
x_train,x_test,y_train,y_test= train_test_split(main_df[['Age', 'Sex', 'HighChol', 'CholCheck', 'BMI', 'Smoker',
       'HeartDiseaseorAttack', 'PhysActivity', 'Fruits', 'Veggies',
       'HvyAlcoholConsump', 'GenHlth', 'MentHlth', 'PhysHlth', 'DiffWalk']],target.Hypertension,test_size=0.1)
model_svm_hypertension = svm.SVC()
model_svm_hypertension.fit(x_train, y_train)
score_svm = model_svm_hypertension.score(x_test,y_test)
print("---------------------------------")
print('The accuracy of the SVM is: {}'.format(score_svm))
print("---------------------------------")
#Svm Storke
```

```
---------------------------------
The accuracy of the SVM is: 0.7253182461103254
---------------------------------
```

In [37]:
```
#Svm Stroke
x_train,x_test,y_train,y_test= train_test_split(main_df[['Age', 'Sex', 'HighChol', 'CholCheck', 'BMI', 'Smoker',
       'HeartDiseaseorAttack', 'PhysActivity', 'Fruits', 'Veggies',
       'HvyAlcoholConsump', 'GenHlth', 'MentHlth', 'PhysHlth', 'DiffWalk']],target.Stroke,test_size=0.1)
model_svm_Storke = svm.SVC()
model_svm_Storke.fit(x_train, y_train)
score_svm = model_svm_Storke.score(x_test,y_test)
print("---------------------------------")
print('The accuracy of the SVM is: {}'.format(score_svm))
print("---------------------------------")
```

```
---------------------------------
The accuracy of the SVM is: 0.9357850070721357
---------------------------------
```

Decision Tree

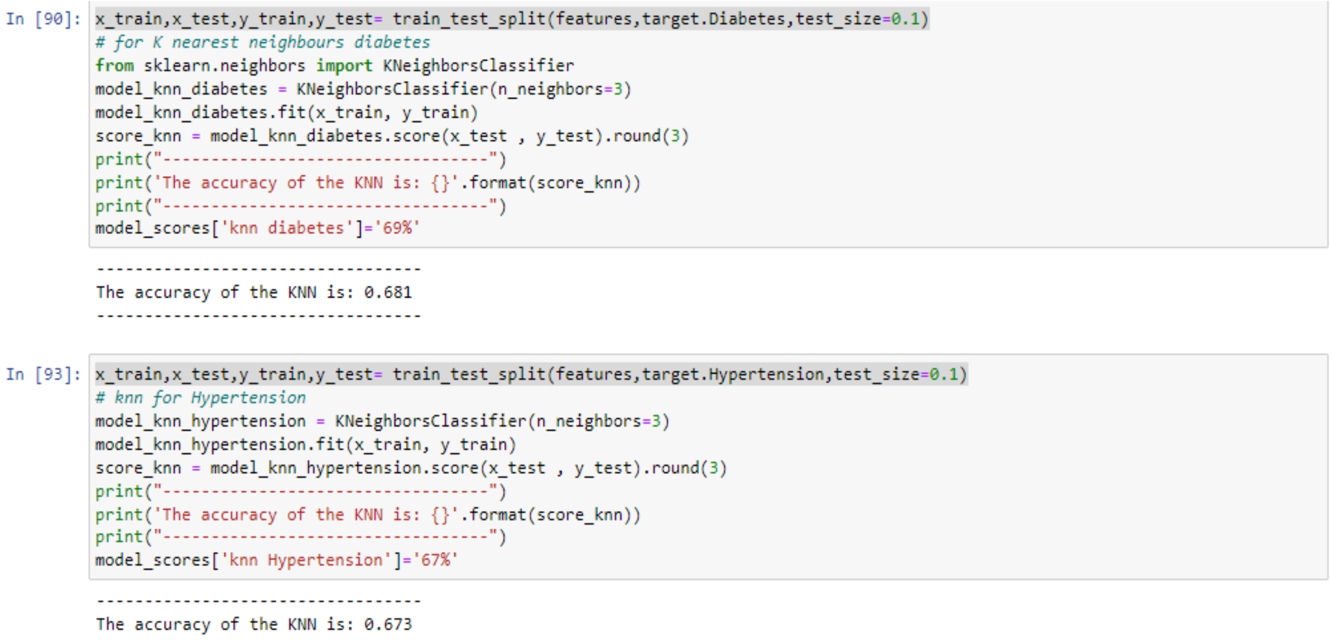


```
In [60]: x_train,x_test,y_train,y_test= train_test_split(features,target.Diabetes,test_size=0.1)
         #for using Decision Tree Algoithm
         from sklearn.tree import DecisionTreeClassifier
         model_dt_diabetes = DecisionTreeClassifier(random_state=4)
         model_dt_diabetes.fit(x_train, y_train)
         score_dt = model_dt_diabetes.score(x_test,y_test)
         print("---------------------------------")
         print('The accuracy of the DT is: {}'.format(score_dt.round(3)))
         print("---------------------------------")
         model_scores['Decision Tree Diabetes']='67%'
         #print(model_scores)

         ---------------------------------
         The accuracy of the DT is: 0.67
         ---------------------------------
```

```
In [84]: x_train,x_test,y_train,y_test= train_test_split(features,target.Hypertension,test_size=0.1)
         #for using Decision Tree Algoithm---------------- hypertension
         model_dt_hypertension = DecisionTreeClassifier(random_state=4)
         model_dt_hypertension.fit(x_train, y_train)
         score_dt = model_dt_hypertension.score(x_test,y_test)
         print("---------------------------------")
         print('The accuracy of the DT is: {}'.format(score_dt.round(3)))
         print("---------------------------------")
         model_scores['Decision Tree Hypertension']='65%'

         ---------------------------------
         The accuracy of the DT is: 0.652
         ---------------------------------
```

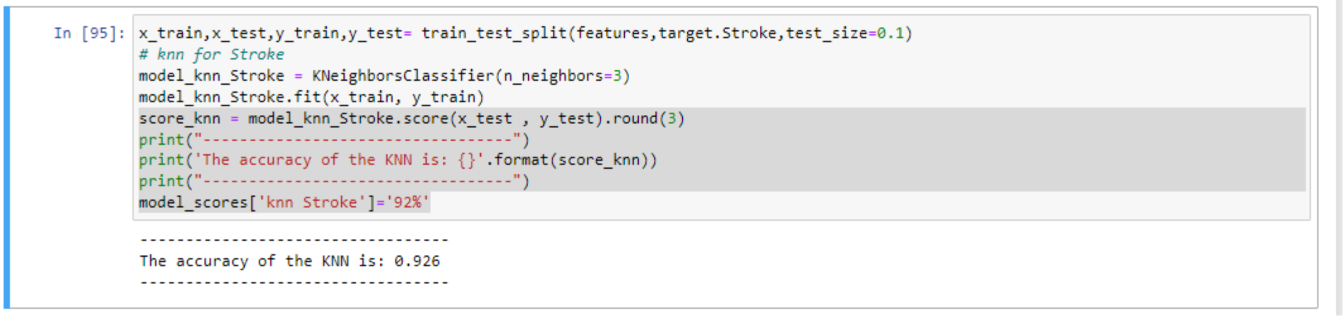
```
In [86]: x_train,x_test,y_train,y_test= train_test_split(features,target.Stroke,test_size=0.1)
         #for using Decision Tree Algoithm---------------- Stroke
         model_dt_stroke = DecisionTreeClassifier(random_state=4)
         model_dt_stroke.fit(x_train, y_train)
         score_dt = model_dt_stroke.score(x_test,y_test)
         print("---------------------------------")
         print('The accuracy of the DT is: {}'.format(score_dt.round(3)))
         print("---------------------------------")
         model_scores['Decision Tree Stroke']='89%'

         ---------------------------------
         The accuracy of the DT is: 0.892
         ---------------------------------
```

KNN Model



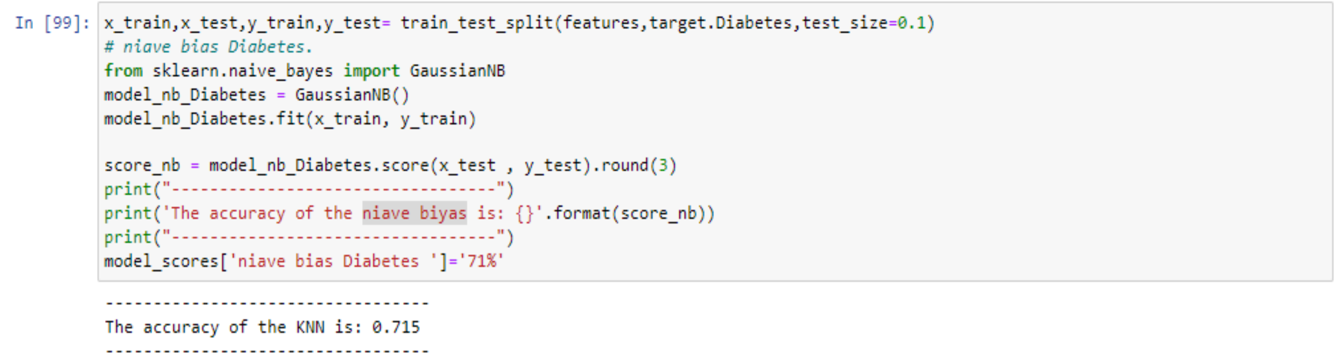
```
In [90]: x_train,x_test,y_train,y_test= train_test_split(features,target.Diabetes,test_size=0.1)
         # for K nearest neighbours diabetes
         from sklearn.neighbors import KNeighborsClassifier
         model_knn_diabetes = KNeighborsClassifier(n_neighbors=3)
         model_knn_diabetes.fit(x_train, y_train)
         score_knn = model_knn_diabetes.score(x_test , y_test).round(3)
         print("---------------------------------")
         print('The accuracy of the KNN is: {}'.format(score_knn))
         print("---------------------------------")
         model_scores['knn diabetes']='69%'

         ---------------------------------
         The accuracy of the KNN is: 0.681
         ---------------------------------
```

```
In [93]: x_train,x_test,y_train,y_test= train_test_split(features,target.Hypertension,test_size=0.1)
         # knn for Hypertension
         model_knn_hypertension = KNeighborsClassifier(n_neighbors=3)
         model_knn_hypertension.fit(x_train, y_train)
         score_knn = model_knn_hypertension.score(x_test , y_test).round(3)
         print("---------------------------------")
         print('The accuracy of the KNN is: {}'.format(score_knn))
         print("---------------------------------")
         model_scores['knn Hypertension']='67%'

         ---------------------------------
         The accuracy of the KNN is: 0.673
```

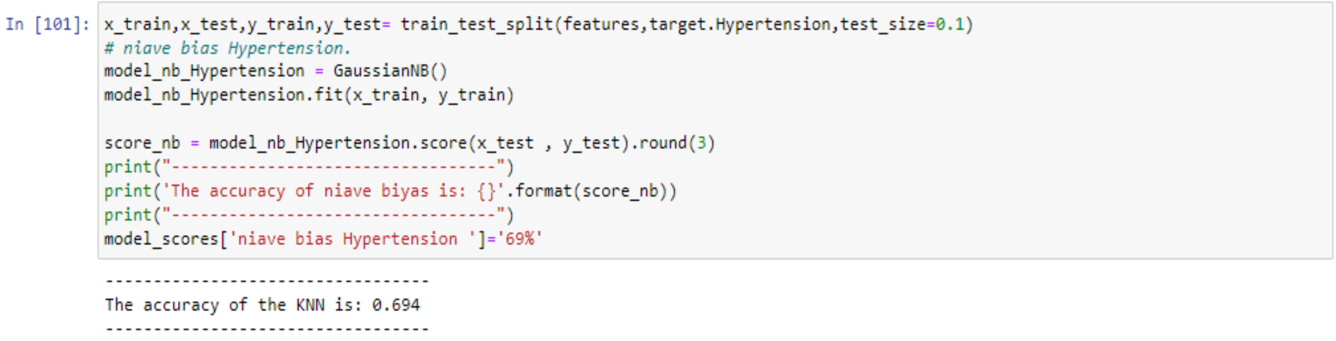


```
In [95]: x_train,x_test,y_train,y_test= train_test_split(features,target.Stroke,test_size=0.1)
         # knn for Stroke
         model_knn_Stroke = KNeighborsClassifier(n_neighbors=3)
         model_knn_Stroke.fit(x_train, y_train)
         score_knn = model_knn_Stroke.score(x_test , y_test).round(3)
         print("---------------------------------")
         print('The accuracy of the KNN is: {}'.format(score_knn))
         print("---------------------------------")
         model_scores['knn Stroke']='92%'

         ---------------------------------
         The accuracy of the KNN is: 0.926
         ---------------------------------
```

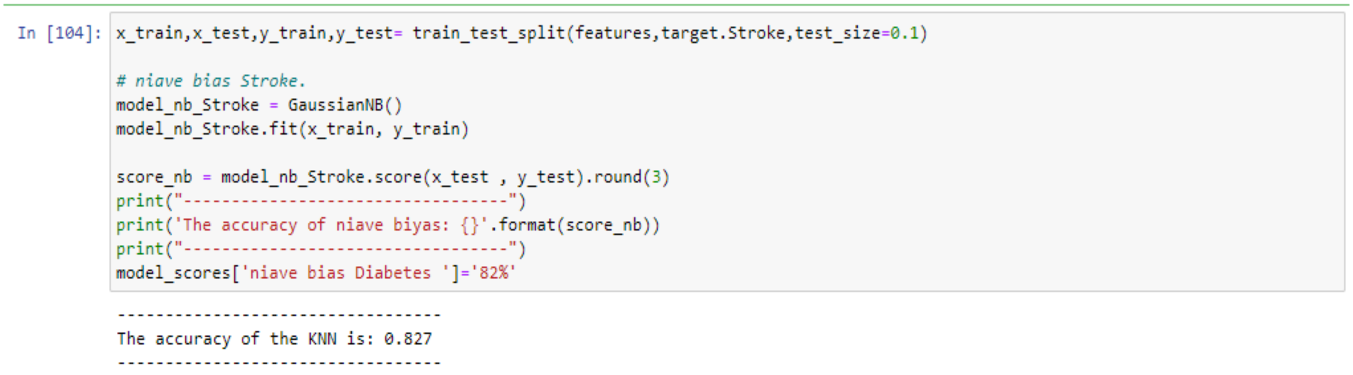
Naive biyas



```
In [99]: x_train,x_test,y_train,y_test= train_test_split(features,target.Diabetes,test_size=0.1)
         # niave bias Diabetes.
         from sklearn.naive_bayes import GaussianNB
         model_nb_Diabetes = GaussianNB()
         model_nb_Diabetes.fit(x_train, y_train)

         score_nb = model_nb_Diabetes.score(x_test , y_test).round(3)
         print("---------------------------------")
         print('The accuracy of the niave biyas is: {}'.format(score_nb))
         print("---------------------------------")
         model_scores['niave bias Diabetes ']='71%'

         ---------------------------------
         The accuracy of the KNN is: 0.715
         ---------------------------------
```



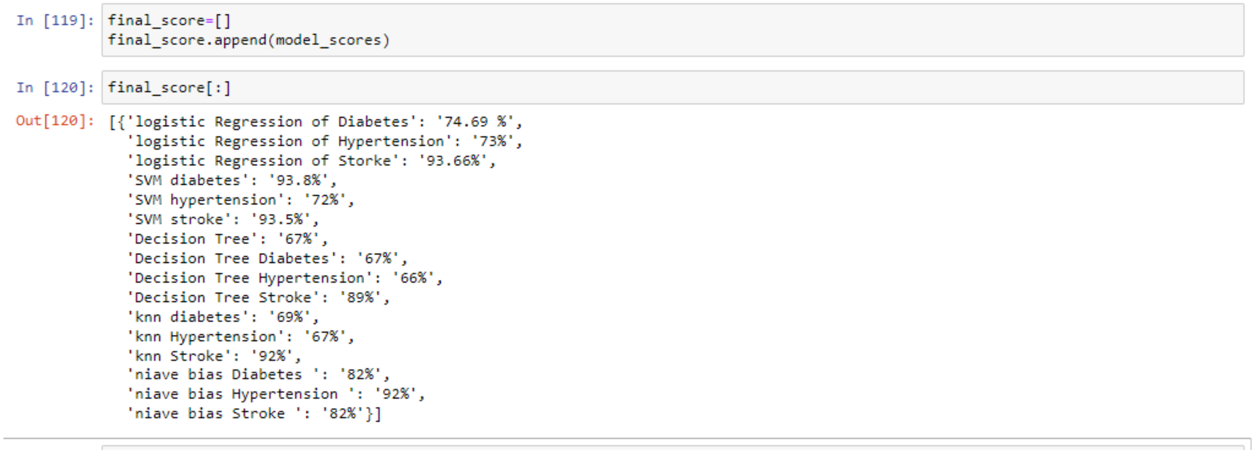
```
In [101]: x_train,x_test,y_train,y_test= train_test_split(features,target.Hypertension,test_size=0.1)
          # niave bias Hypertension.
          model_nb_Hypertension = GaussianNB()
          model_nb_Hypertension.fit(x_train, y_train)

          score_nb = model_nb_Hypertension.score(x_test , y_test).round(3)
          print("---------------------------------")
          print('The accuracy of niave biyas is: {}'.format(score_nb))
          print("---------------------------------")
          model_scores['niave bias Hypertension ']='69%'

          ---------------------------------
          The accuracy of the KNN is: 0.694
          ---------------------------------
```



```
In [104]: x_train,x_test,y_train,y_test= train_test_split(features,target.Stroke,test_size=0.1)

          # niave bias Stroke.
          model_nb_Stroke = GaussianNB()
          model_nb_Stroke.fit(x_train, y_train)

          score_nb = model_nb_Stroke.score(x_test , y_test).round(3)
          print("---------------------------------")
          print('The accuracy of niave biyas: {}'.format(score_nb))
          print("---------------------------------")
          model_scores['niave bias Diabetes ']='82%'

          ---------------------------------
          The accuracy of the KNN is: 0.827
          ---------------------------------
```

Result:

The result shows that Among all the classification models best performing model was naive biyas . for 3 target variable this model gives best performing accuracy.



```
In [119]: final_score=[]
          final_score.append(model_scores)

In [120]: final_score[:]

Out[120]: [{'logistic Regression of Diabetes': '74.69 %',
            'logistic Regression of Hypertension': '73%',
            'logistic Regression of Storke': '93.66%',
            'SVM diabetes': '93.8%',
            'SVM hypertension': '72%',
            'SVM stroke': '93.5%',
            'Decision Tree': '67%',
            'Decision Tree Diabetes': '67%',
            'Decision Tree Hypertension': '66%',
            'Decision Tree Stroke': '89%',
            'knn diabetes': '69%',
            'knn Hypertension': '67%',
            'knn Stroke': '92%',
            'niave bias Diabetes ': '82%',
            'niave bias Hypertension ': '92%',
            'niave bias Stroke ': '82%'}]
```

Future Scope:

This project has many limitations such as there are many feature that dosent affect the target variable , removing these features could give better result. With proper time and effort this project could be improved.

Work Acknowledgement:

To our Honarable Teacher “**AKINUL ISLAM JONY** ” sir.
Department of Computer Science and engineering.