AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Choose an item.

Project Cover Sheet

Assignment Title:	Python Final Term Project		
Assignment No:	01	Date of Submission:	8 August 2022
Course Title:	Programming in Python		
Course Code:	CSC4162	Section:	A
Semester:	Summer 2021-22	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	Mithun Chanda Shuvo	19-39592-1	BSc [CSE]	
2	Salah Ahommod Emu	19-39603-1	BSc [CSE]	
3	Md. Asif Mostofa	19-41266-3	BSc [CSE]	
4	Nusrat Jahan Mou	19-39583-1	BSc [CSE]	
5			Choose an item.	
6			Choose an item.	
7			Choose an item.	
8			Choose an item.	
9			Choose an item.	
10			Choose an item.	

Faculty use only

FACULTYCOMMENTS	**Marks Obtained**	
	Total Marks	

Section 1: Project Overview

The goal of this study is to forecast a person's likelihood of having a stroke based on personal information and figure out if they are related to the case in any manner. To make the prediction, we will use a dataset with over 4000 cases to train our model. There are many symptoms of strokes. All these symptoms resemble different diseases like it occurs in aging persons, so it becomes a challenging task to get a correct diagnosis. But as time passes, a lot of research data and patients records of hospitals are available. There are many open sources for accessing the patient's records and research can be conducted so that various computer technologies could be used for doing the correct diagnosis of the patients and detecting this disease to stop it from becoming fatal. Nowadays it is well known that machine learning and artificial intelligence are playing a huge role in the medical industry. We can use different machine learning models to diagnose the disease and classify or predict the results. A complete data analysis can easily be done using machine learning models. Models can be trained for knowledge predictions and medical records can be transformed and analyzed more deeply for better predictions.

The steps we followed in this project:

1. Data Collection

2. Data Cleaning

3. Exploratory data analysis

4. Feature selection

5. Machine Learning Model develop

6. Result analysis

Section 2: Dataset Overview

The valid URL of the dataset that we made use of for our project is given below:

https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset

The dataset that we have used in our project is called ‘Stroke Prediction Dataset’. This dataset holds columns named as ‘id’,’gender’,’age’,’hypertension’,’heart_disease’,’ever_married’,’work_type’,’residance_type’,’avg_glucose_level’,’bmi’,’smoking_status’,’stroke’.

This dataset will primarily figure out whether a person is susceptible to stroke based on personal information such as id, gender, age, whether the person is married, his working kind, and his residence type. The dataset also includes the individual's medical history, such as hypertension, heart disease, average glucose level, BMI, smoking status, and stroke. Every row in the collection holds information about the patient.

Section 3: Data Preprocessing and Exploratory data analysis

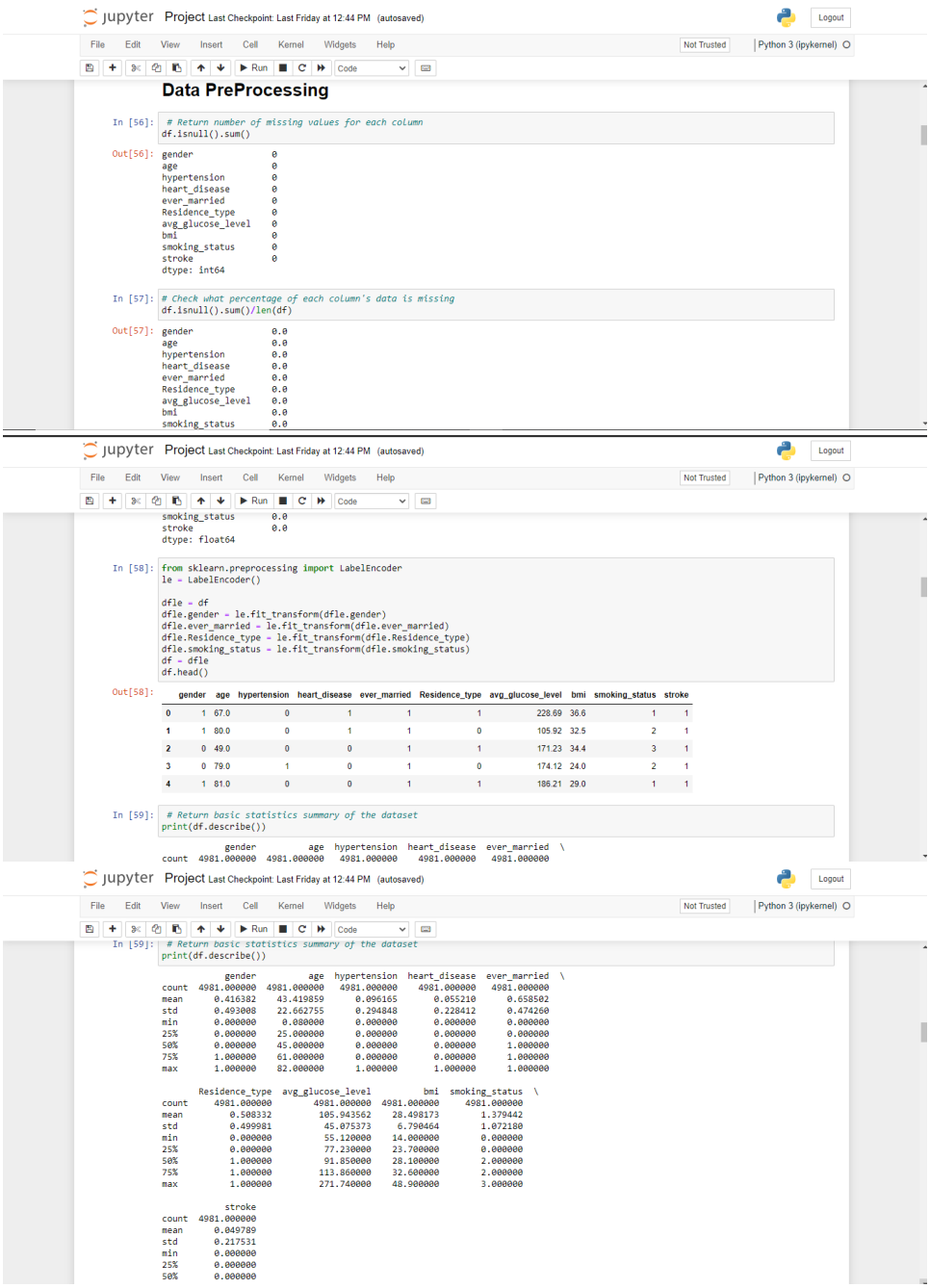
- data preprocessing/cleaning steps

Import all the Library and load dataset

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn import datasets # for using built-in datasets
from sklearn import metrics # for checking the model accuracy
df = pd.read_csv('D:\\Dataset\\archive\\full_data.csv')
df
```

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
2	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
3	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
4	Male	81.0	0	0	Yes	Private	Urban	186.21	29.0	formerly smoked	1
...	...	...	...	...	...	...	...	...	...	...	...
4976	Male	41.0	0	0	No	Private	Rural	70.15	29.8	formerly smoked	0
4977	Male	40.0	0	0	Yes	Private	Urban	191.15	31.1	smokes	0
4978	Female	45.0	1	0	Yes	Govt_job	Rural	95.02	31.8	smokes	0
4979	Male	40.0	0	0	Yes	Private	Rural	83.94	30.0	smokes	0
4980	Female	80.0	1	0	Yes	Private	Urban	83.75	29.1	never smoked	0

4981 rows × 11 columns



jupyter Project Last Checkpoint: Last Friday at 12:44 PM (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

Data PreProcessing

In [56]:
```
# Return number of missing values for each column
df.isnull().sum()
```

Out[56]:
```
gender               0
age                  0
hypertension         0
heart_disease        0
ever_married         0
Residence_type       0
avg_glucose_level    0
bmi                  0
smoking_status       0
stroke               0
dtype: int64
```

In [57]:
```
# Check what percentage of each column's data is missing
df.isnull().sum()/len(df)
```

Out[57]:
```
gender               0.0
age                  0.0
hypertension         0.0
heart_disease        0.0
ever_married         0.0
Residence_type       0.0
avg_glucose_level    0.0
bmi                  0.0
smoking_status       0.0
stroke               0.0
dtype: float64
```

In [58]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

dfle = df
dfle.gender = le.fit_transform(dfle.gender)
dfle.ever_married = le.fit_transform(dfle.ever_married)
dfle.Residence_type = le.fit_transform(dfle.Residence_type)
dfle.smoking_status = le.fit_transform(dfle.smoking_status)
df = dfle
df.head()
```

Out[58]:

	gender	age	hypertension	heart_disease	ever_married	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	1	67.0	0	1	1	1	228.69	36.6	1	1
1	1	80.0	0	1	1	0	105.92	32.5	2	1
2	0	49.0	0	0	1	1	171.23	34.4	3	1
3	0	79.0	1	0	1	0	174.12	24.0	2	1
4	1	81.0	0	0	1	1	186.21	29.0	1	1

In [59]:
```
# Return basic statistics summary of the dataset
print(df.describe())
```

```
            gender          age  hypertension  heart_disease  ever_married  \
count  4981.000000  4981.000000   4981.000000    4981.000000   4981.000000
mean      0.416382    43.419859      0.096165       0.055210      0.658502
std       0.493008    22.662755      0.294848       0.228412      0.474260
min       0.000000     0.080000      0.000000       0.000000      0.000000
25%       0.000000    25.000000      0.000000       0.000000      0.000000
50%       0.000000    45.000000      0.000000       0.000000      1.000000
75%       1.000000    61.000000      0.000000       0.000000      1.000000
max       1.000000    82.000000      1.000000       1.000000      1.000000

       Residence_type  avg_glucose_level          bmi  smoking_status  \
count     4981.000000        4981.000000  4981.000000     4981.000000
mean         0.508332         105.943562    28.498173        1.379442
std          0.499981          45.075373     6.790464        1.071180
min          0.000000          55.120000    14.000000        0.000000
25%          0.000000          77.230000    23.700000        0.000000
50%          1.000000          91.850000    28.100000        2.000000
75%          1.000000         113.860000    32.600000        2.000000
max          1.000000         271.740000    48.900000        3.000000

            stroke
count  4981.000000
mean      0.049789
std       0.217531
min       0.000000
25%       0.000000
50%       0.000000
```

- Exploratory data analysis

jupyter Project Last Checkpoint: Last Friday at 12:44 PM (autosaved)

Logout

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

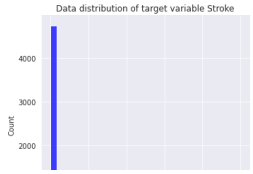
Run Code

Exploratory Data

In [60]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [61]:
```
# To display plots inside notebook
%matplotlib inline
```

In [96]:
```
with sns.axes_style('darkgrid'):
 sns.displot(df['stroke'], bins=0, color='blue')
 plt.title("Data distribution of target variable Stroke");
```



jupyter Project Last Checkpoint: Last Friday at 12:44 PM (autosaved)

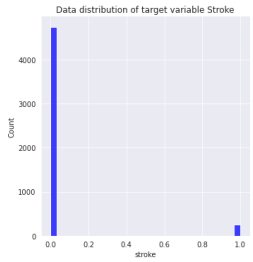
Logout

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

Run Code

%matplotlib inline

In [96]:
```
with sns.axes_style('darkgrid'):
 sns.displot(df['stroke'], bins=0, color='blue')
 plt.title("Data distribution of target variable Stroke");
```



In [64]:
```
# corr() to calculate the correlation between variables
```

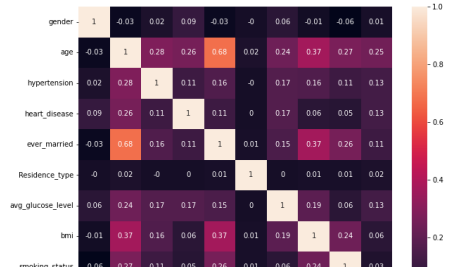
jupyter Project Last Checkpoint: Last Friday at 12:44 PM (autosaved)

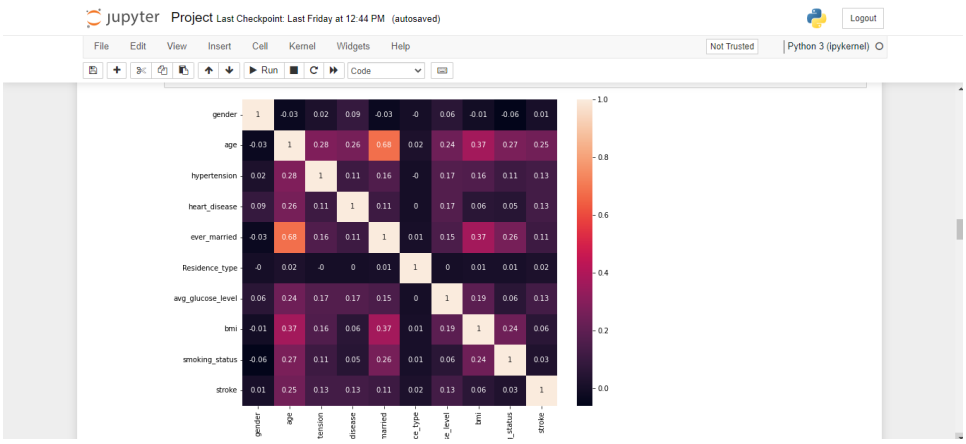
Logout

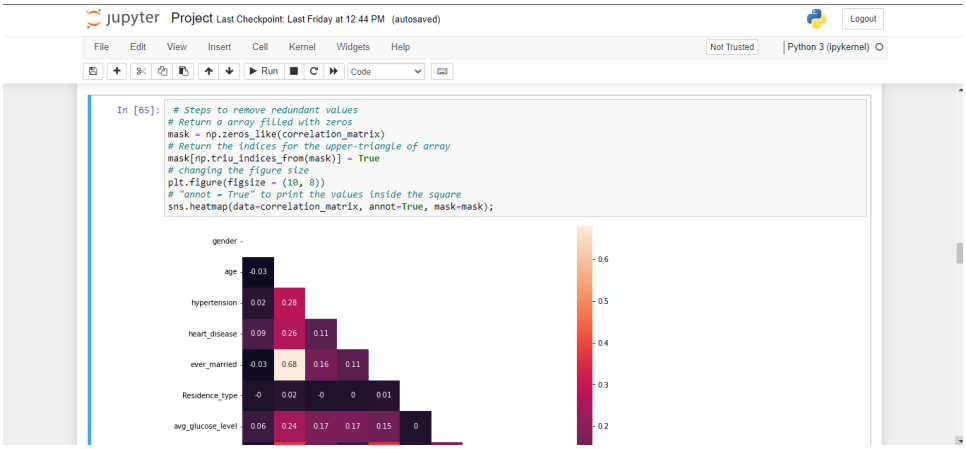
File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

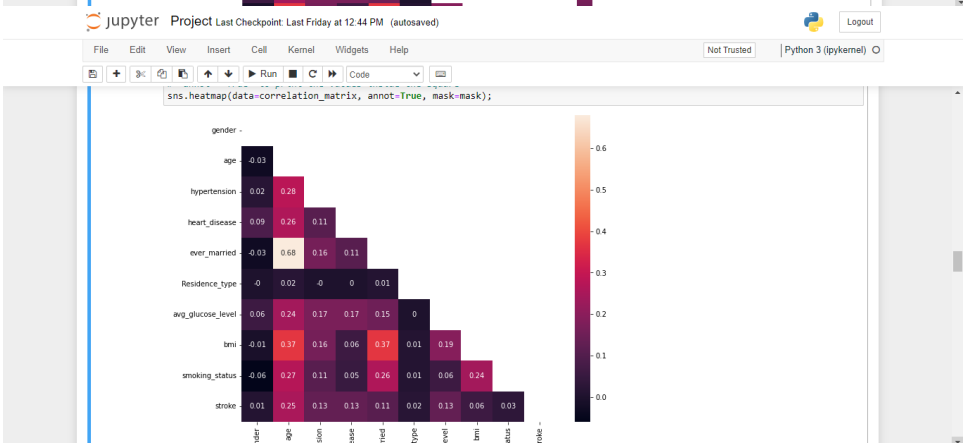
Run Code

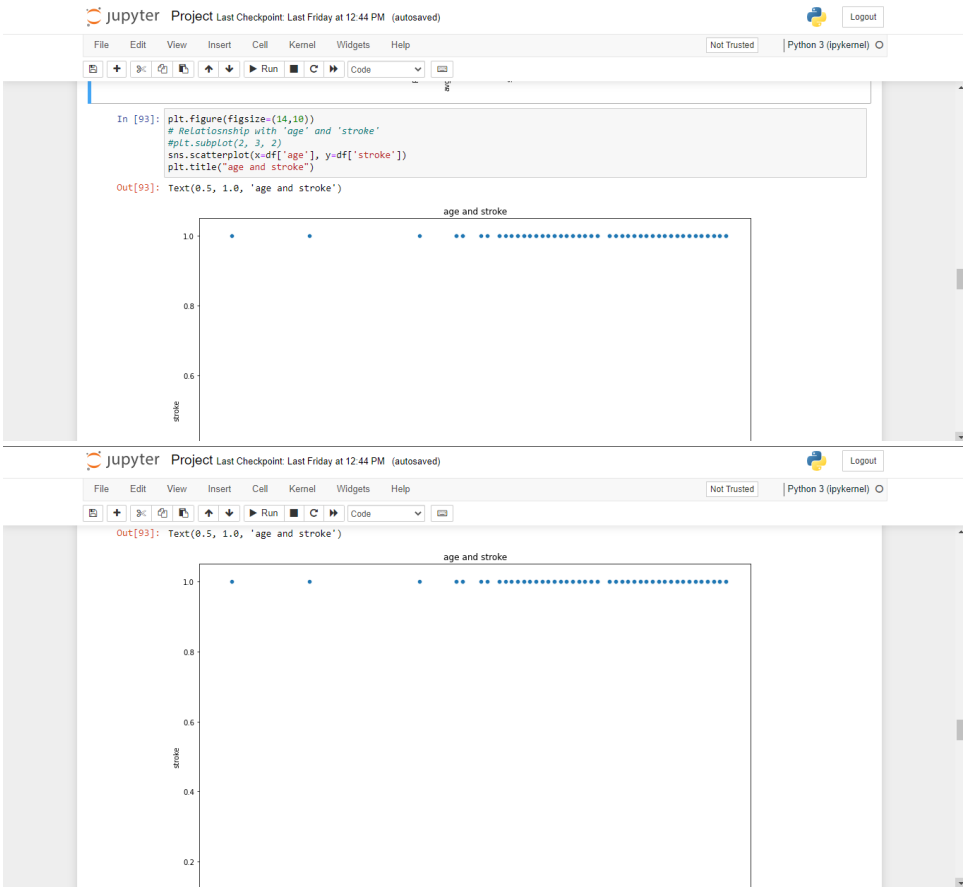
In [64]:
```
# corr() to calculate the correlation between variables
correlation_matrix = df.corr().round(2)
# changing the figure size
plt.figure(figsize = (10, 8))
# "annot = True" to print the values inside the square
sns.heatmap(data=correlation_matrix, annot=True);
```



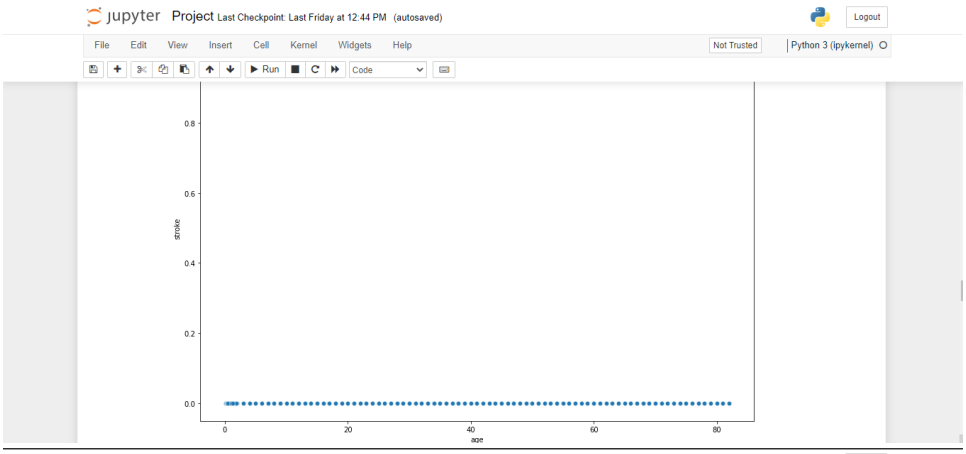

jupyter Project Last Checkpoint: Last Friday at 12:44 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
gender 1 -0.03 0.02 0.09 -0.03 -0 0.06 -0.01 -0.06 0.01
age -0.03 1 0.28 0.26 0.68 0.02 0.24 0.37 0.27 0.25
hypertension 0.02 0.28 1 0.11 0.16 -0 0.17 0.16 0.11 0.13
heart_disease 0.09 0.26 0.11 1 0.11 0 0.17 0.06 0.05 0.13
ever_married -0.03 0.68 0.16 0.11 1 0.01 0.15 0.37 0.26 0.11
Residence_type -0 0.02 -0 0 0.01 1 0 0.01 0.01 0.02
avg_glucose_level 0.06 0.24 0.17 0.17 0.15 0 1 0.19 0.06 0.13
bmi -0.01 0.37 0.16 0.06 0.37 0.01 0.19 1 0.24 0.06
smoking_status -0.06 0.27 0.11 0.05 0.26 0.01 0.06 0.24 1 0.03
stroke 0.01 0.25 0.13 0.13 0.11 0.02 0.13 0.06 0.03 1
1.0 0.8 0.6 0.4 0.2 0.0

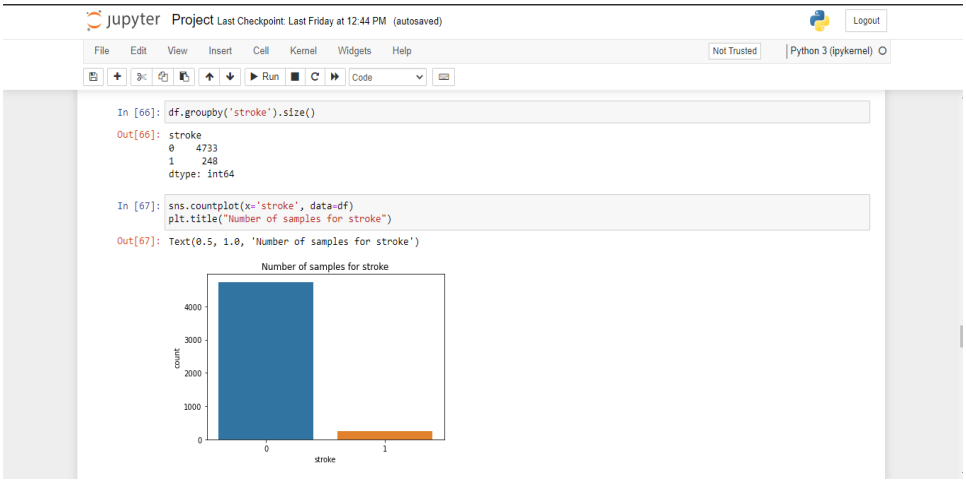

jupyter Project Last Checkpoint: Last Friday at 12:44 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [65]:
# Steps to remove redundant values
# Return a array filled with zeros
mask = np.zeros_like(correlation_matrix)
# Return the indices for the upper-triangle of array
mask[np.triu_indices_from(mask)] = True
# changing the figure size
plt.figure(figsize = (10, 8))
# "annot = True" to print the values inside the square
sns.heatmap(data=correlation_matrix, annot=True, mask=mask);
gender -
age -0.03
hypertension 0.02 0.28
heart_disease 0.09 0.26 0.11
ever_married -0.03 0.68 0.16 0.11
Residence_type -0 0.02 -0 0 0.01
avg_glucose_level 0.06 0.24 0.17 0.17 0.15 0
0.6 0.5 0.4 0.3 0.2


jupyter Project Last Checkpoint: Last Friday at 12:44 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
sns.heatmap(data=correlation_matrix, annot=True, mask=mask);
gender -
age -0.03
hypertension 0.02 0.28
heart_disease 0.09 0.26 0.11
ever_married -0.03 0.68 0.16 0.11
Residence_type -0 0.02 -0 0 0.01
avg_glucose_level 0.06 0.24 0.17 0.17 0.15 0
bmi -0.01 0.37 0.16 0.06 0.37 0.01 0.19
smoking_status -0.06 0.27 0.11 0.05 0.26 0.01 0.06 0.24
stroke 0.01 0.25 0.13 0.13 0.11 0.02 0.13 0.06 0.03
0.6 0.5 0.4 0.3 0.2 0.1 0.0


jupyter Project Last Checkpoint: Last Friday at 12:44 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [93]:
plt.figure(figsize=(14,10))
# Relatiosnship with 'age' and 'stroke'
#plt.subplot(2, 3, 2)
sns.scatterplot(x=df['age'], y=df['stroke'])
plt.title("age and stroke")
Out[93]: Text(0.5, 1.0, 'age and stroke')
age and stroke
stroke
1.0
0.8
0.6

jupyter Project Last Checkpoint: Last Friday at 12:44 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Out[93]: Text(0.5, 1.0, 'age and stroke')
age and stroke
stroke
1.0
0.8
0.6
0.4
0.2





section 4: Model Development

jupyter Project Last Checkpoint: Last Friday at 12:44 PM (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

In [68]:
```
X = df.drop('stroke',axis='columns')
y = df.stroke
```

Split the dataset

In [69]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 16)
print("X_train shape: ", X_train.shape)
print("X_test shape: ", X_test.shape)
print("y_train shape: ", y_train.shape)
print("y_test shape: ", y_test.shape)
```

```
X_train shape:  (3486, 9)
X_test shape:  (1495, 9)
y_train shape:  (3486,)
y_test shape:  (1495,)
```

jupyter Project Last Checkpoint: Last Friday at 12:44 PM (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

Create Model: Support Vector Machine (SVM)

In [72]:
```
# importing the necessary package to use the classification algorithm
from sklearn import svm #for Support Vector Machine (SVM) Algorithm
model_svm = svm.SVC() #select the algorithm
model_svm.fit(X_train, y_train) #train the model with the training dataset
y_prediction_svm = model_svm.predict(X_test) # pass the testing data to the trained model
# checking the accuracy of the algorithm.
# by comparing predicted output by the model and the actual output
score_svm = metrics.accuracy_score(y_prediction_svm, y_test).round(4)
print("-----------------------------------")
print('The accuracy of the SVM is: {}'.format(score_svm))
print("-----------------------------------")
# save the accuracy score
score = set()
score.add(('SVM', score_svm))
```

```
-----------------------------------
The accuracy of the SVM is: 0.9525
-----------------------------------
```

Decision Tree

In [73]:
```
# importing the necessary package to use the classification algorithm
from sklearn.tree import DecisionTreeClassifier #for using Decision Tree Algoithm
```

jupyter Project Last Checkpoint: Last Friday at 12:44 PM (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

Decision Tree

In [73]:
```
# importing the necessary package to use the classification algorithm
from sklearn.tree import DecisionTreeClassifier #for using Decision Tree Algoithm
model_dt = DecisionTreeClassifier(random_state=4)
model_dt.fit(X_train, y_train) #train the model with the training dataset
y_prediction_dt = model_dt.predict(X_test) #pass the testing data to the trained model
# checking the accuracy of the algorithm.
# by comparing predicted output by the model and the actual output
score_dt = metrics.accuracy_score(y_prediction_dt, y_test).round(4)
print("-----------------------------------")
print('The accuracy of the DT is: {}'.format(score_dt))
print("-----------------------------------")
# save the accuracy score
score.add(('DT', score_dt))
```

```
-----------------------------------
The accuracy of the DT is: 0.905
-----------------------------------
```

K Nearest Neighbours (KNN)

In [74]:
```
# importing the necessary package to use the classification algorithm
from sklearn.neighbors import KNeighborsClassifier # for K nearest neighbours
#from sklearn.linear_model import LogisticRegression # for Logistic Regression algorithm
model_knn = KNeighborsClassifier(n_neighbors=3) # 3 neighbours for putting the new data into a class
```

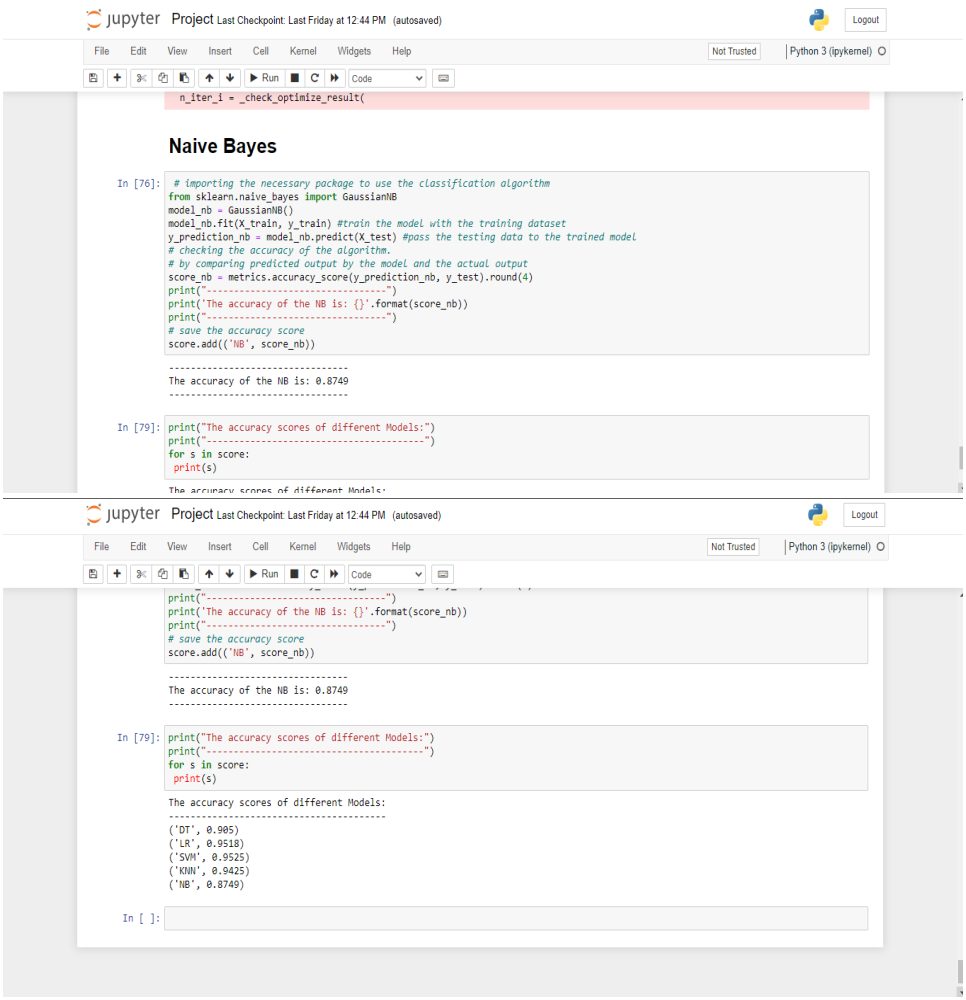

jupyter Project Last Checkpoint: Last Friday at 12:44 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
K Nearest Neighbours (KNN)
In [74]:
# importing the necessary package to use the classification algorithm
from sklearn.neighbors import KNeighborsClassifier # for K nearest neighbours
#from sklearn.linear_model import LogisticRegression # for Logistic Regression algorithm
model_knn = KNeighborsClassifier(n_neighbors=3) # 3 neighbours for putting the new data into a class
model_knn.fit(X_train, y_train) #train the model with the training dataset
y_prediction_knn = model_knn.predict(X_test) #pass the testing data to the trained model
# checking the accuracy of the algorithm.
# by comparing predicted output by the model and the actual output
score_knn = metrics.accuracy_score(y_prediction_knn, y_test).round(4)
print("----------------------------------")
print('The accuracy of the KNN is: {}'.format(score_knn))
print("----------------------------------")
# save the accuracy score
score.add(('KNN', score_knn))

The accuracy of the KNN is: 0.9425

Logistic Regression
In [75]:
importing the necessary package to use the classification algorithm


jupyter Project Last Checkpoint: Last Friday at 12:44 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Logistic Regression
In [75]:
# importing the necessary package to use the classification algorithm
from sklearn.linear_model import LogisticRegression # for Logistic Regression algorithm
model_lr = LogisticRegression()
model_lr.fit(X_train, y_train) #train the model with the training dataset
y_prediction_lr = model_lr.predict(X_test) #pass the testing data to the trained model
# checking the accuracy of the algorithm.
# by comparing predicted output by the model and the actual output
score_lr = metrics.accuracy_score(y_prediction_lr, y_test).round(4)
print("----------------------------------")
print('The accuracy of the LR is: {}'.format(score_lr))
print("----------------------------------")
# save the accuracy score
score.add(('LR', score_lr))

The accuracy of the LR is: 0.9518



Section 5: Discussion & Conclusion

Here in our project, we have used 5 models.

- Naive Bayes
- KNN
- Decision Tree
- Logistic Regression
- SVM

We can see from the comparison among the models that

By training the model with training datasets,

- The accuracy score of Naive Bayes (NB) is 0.8749.

- The accuracy score of KNN is 0.9425
- The accuracy score of Decision tree (DT) is 0.905
- The accuracy score of Logistic Regression (LR) is 0.9518
- The accuracy score of SVM is 0.9525

Here we compared predicted output of all the models with their actual output. So, the accuracy score of SVM model was the highest which is 0.9525 and it is most efficient. The least accuracy score is found from the Naive Bayes which is 0.8749.