Report on

Machine Learning Approach on Brain Stroke Prediction

Prepared for

Mr. Md. Siam Ansary, Lecturer

Ms. Tamanna Tabassum, Lecturer

Department: Computer Science and Engineering

Prepared by

Ifrat Jahan Chowdhury

ID: 18.02.04.003

Fariya Islam

ID: 18.02.04.007

Md. Fahim Rahman

ID: 18.02.04.028

Date: 04 September, 2022

Ahsanullah University of Science and Technology

TABLE OF CONTENTS

Introduction

A stroke is a medical condition in which poor blood flow to the brain causes cell death. There are two main types of stroke: ischemic, due to lack of blood flow, and hemorrhagic, due to bleeding. Both cause parts of the brain to stop functioning properly. Signs and symptoms of a stroke may include an inability to move or feel on one side of the body, problems understanding or speaking, dizziness, or loss of vision to one side. Signs and symptoms often appear soon after the stroke has occurred. If symptoms last less than one or two hours, the stroke is a transient ischemic attack (TIA), also called a mini-stroke. A hemorrhagic stroke may also be associated with a severe headache. The symptoms of a stroke can be permanent. Long-term complications may include pneumonia and loss of bladder control.

The main risk factor for stroke is high blood pressure. Other risk factors include high blood cholesterol, tobacco smoking, obesity, diabetes mellitus, a previous TIA, end-stage kidney disease, and atrial fibrillation. An ischemic stroke is typically caused by blockage of a blood vessel, though there are also less common causes. A hemorrhagic stroke is caused by either bleeding directly into the brain or into the space between the brain's membranes. Bleeding may occur due to a ruptured brain aneurysm. Diagnosis is typically based on a physical exam and supported by medical imaging such as a CT scan or MRI scan. A CT scan can rule out bleeding, but may not necessarily rule out ischemia, which early on typically does not show up on a CT scan. Other tests such as an electrocardiogram (ECG) and blood tests are done to determine risk factors and rule out other possible causes. Low blood sugar may cause similar symptoms.

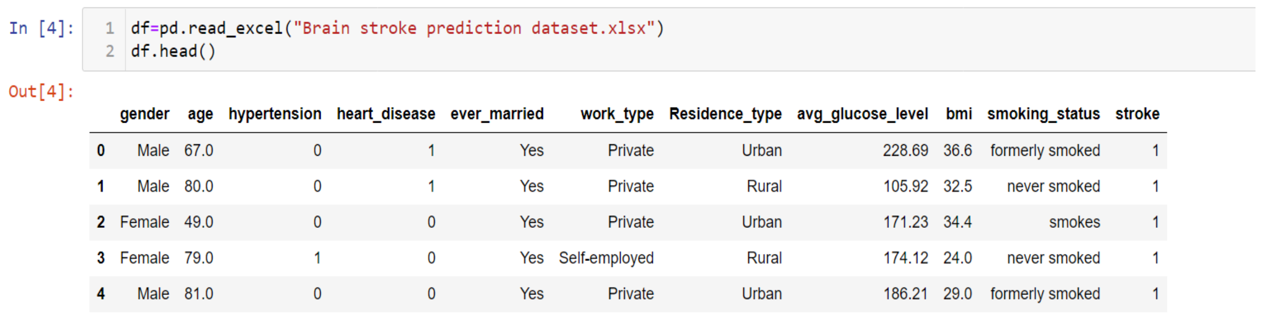
Prevention includes decreasing risk factors, surgery to open up the arteries to the brain in those with problematic carotid narrowing, and warfarin in people with atrial fibrillation. Aspirin or statins may be recommended by physicians for prevention. A stroke or TIA often requires emergency care. An ischemic stroke, if detected within three to four and half hours, may be treatable with a medication that can break down the clot. Some hemorrhagic strokes benefit from surgery. Treatment to attempt recovery of lost function is called stroke rehabilitation, and ideally takes place in a stroke unit; however, these are not available in much of the world.

After analysing the dataset we have come to the point that it is possible to predict a person’s brain stroke condition on the basis of a person’s life style, health condition. If a person is on the condition that he/she has a high probability of having brain stroke he/she can consult with doctor. This is a classification problem. There are several machine learning model is applied to get the balanced fit model where our target is to get low variance and low bias. False negative would be dangerous in this case.

Dataset

There are 4981 rows and 11 columns in the dataset - ‘Brain stroke prediction dataset.xlsx’.
The features are -
1) gender: "Male", "Female" or "Other"
2) age: Age of the patient
3) hypertension: 0 if the patient doesn't have hypertension, 1 if the patient has hypertension
4) heartdisease: 0 if the patient doesn't have any heart diseases, 1 if the patient has a heart disease 5) evermarried: "No" or "Yes"
6) worktype: "children", "Govtjov", "Neverworked", "Private" or "Self-employed" 7) Residencetype: "Rural" or "Urban"
8) avgglucoselevel: average glucose level in blood
9) bmi: body mass index
10) smoking_status: "formerly smoked", "never smoked", "smokes" or "Unknown"*
11) stroke: 1 if the patient had a stroke or 0 if not

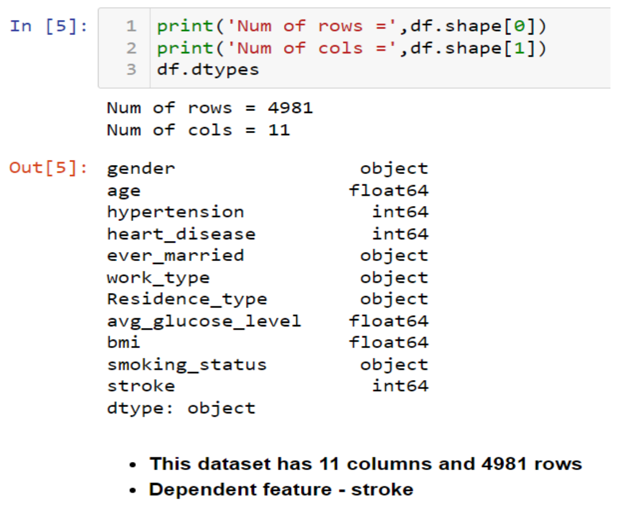
***Note:** "Unknown" in smoking_status means that the information is unavailable for this patient



```
In [4]: df=pd.read_excel("Brain stroke prediction dataset.xlsx")
        df.head()
```

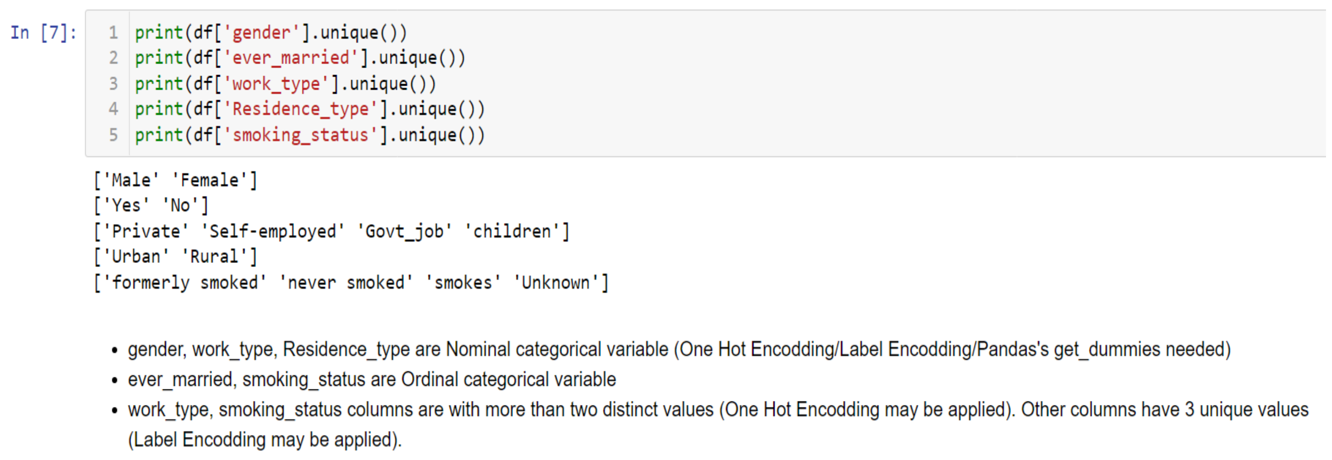
Out[4]:

	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



```
In [5]: print('Num of rows =',df.shape[0])
        print('Num of cols =',df.shape[1])
        df.dtypes

Num of rows = 4981
Num of cols = 11

Out[5]: gender                object
        age                  float64
        hypertension           int64
        heart_disease          int64
        ever_married          object
        work_type             object
        Residence_type        object
        avg_glucose_level    float64
        bmi                  float64
        smoking_status        object
        stroke                 int64
        dtype: object
```

- **This dataset has 11 columns and 4981 rows**
- **Dependent feature - stroke**



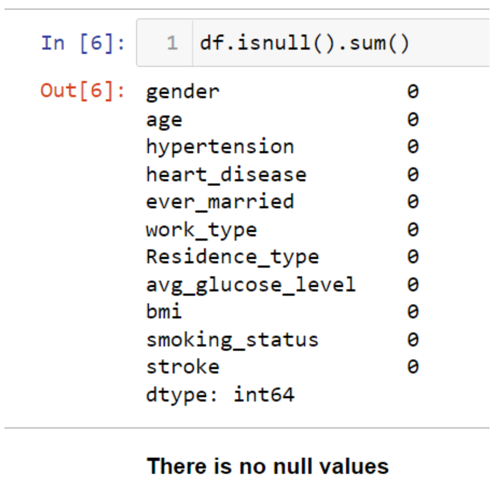
```
In [7]:
print(df['gender'].unique())
print(df['ever_married'].unique())
print(df['work_type'].unique())
print(df['Residence_type'].unique())
print(df['smoking_status'].unique())
```

```
['Male' 'Female']
['Yes' 'No']
['Private' 'Self-employed' 'Govt_job' 'children']
['Urban' 'Rural']
['formerly smoked' 'never smoked' 'smokes' 'Unknown']
```

- gender, work_type, Residence_type are Nominal categorical variable (One Hot Encodding/Label Encodding/Pandas's get_dummies needed)
- ever_married, smoking_status are Ordinal categorical variable
- work_type, smoking_status columns are with more than two distinct values (One Hot Encodding may be applied). Other columns have 3 unique values (Label Encodding may be applied).

Feature Engineering

There is no null values. So we don't need to handle null values.

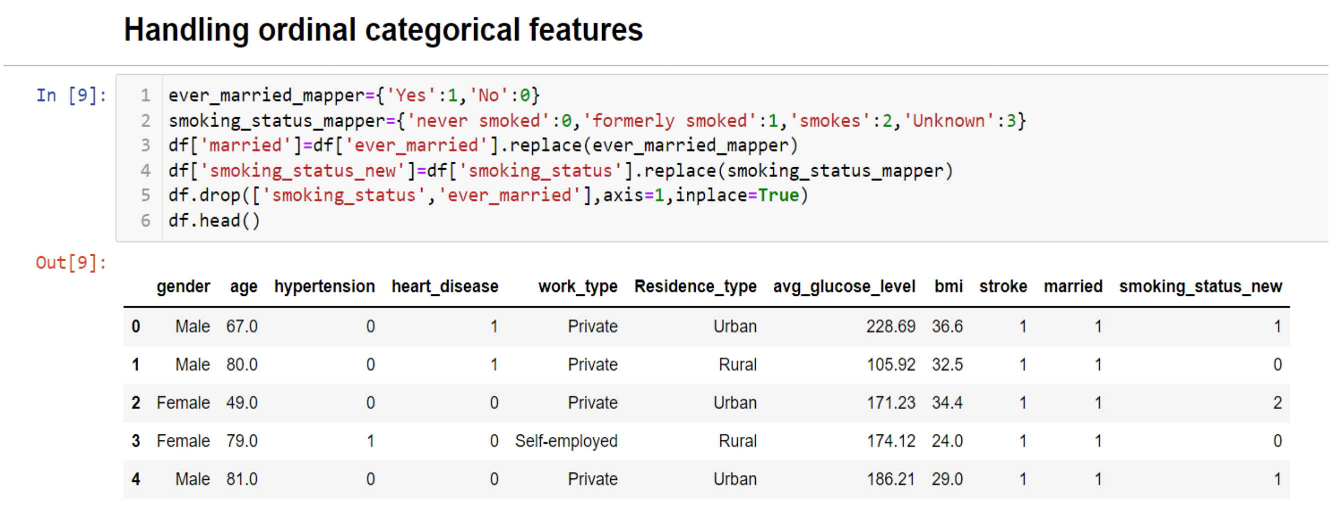


```
In [6]:
df.isnull().sum()
```

```
Out[6]: gender               0
        age                  0
        hypertension         0
        heart_disease        0
        ever_married         0
        work_type            0
        Residence_type       0
        avg_glucose_level    0
        bmi                  0
        smoking_status       0
        stroke               0
        dtype: int64
```

There is no null values

We are assuming that 'ever_married' and 'smoking_status' are ordinal categorical features. We replaced the categorical value with a numerical value by the help of a corresponding mapper in both cases and finally deleted the columns containing categorical values.



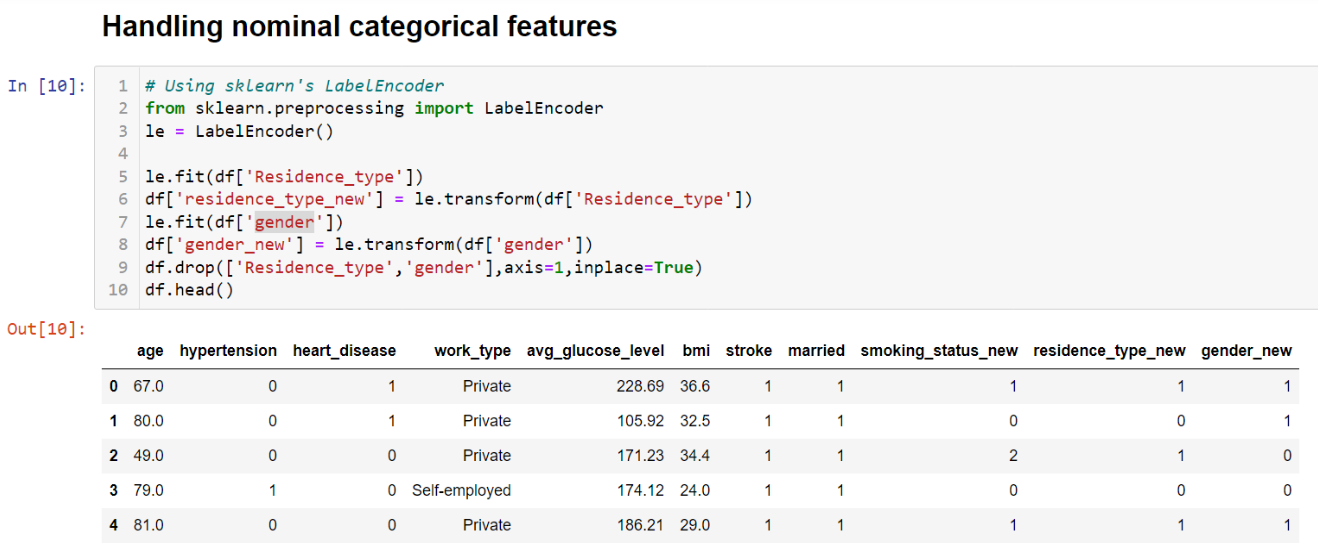
Handling ordinal categorical features

In [9]:
```python
ever_married_mapper={'Yes':1,'No':0}
smoking_status_mapper={'never smoked':0,'formerly smoked':1,'smokes':2,'Unknown':3}
df['married']=df['ever_married'].replace(ever_married_mapper)
df['smoking_status_new']=df['smoking_status'].replace(smoking_status_mapper)
df.drop(['smoking_status','ever_married'],axis=1,inplace=True)
df.head()
```

Out[9]:

	gender	age	hypertension	heart_disease	work_type	Residence_type	avg_glucose_level	bmi	stroke	married	smoking_status_new
0	Male	67.0	0	1	Private	Urban	228.69	36.6	1	1	1
1	Male	80.0	0	1	Private	Rural	105.92	32.5	1	1	0
2	Female	49.0	0	0	Private	Urban	171.23	34.4	1	1	2
3	Female	79.0	1	0	Self-employed	Rural	174.12	24.0	1	1	0
4	Male	81.0	0	0	Private	Urban	186.21	29.0	1	1	1

We applied LabelEncoder on ‘Residence_type’ and ‘gender’ features and assigned the values to new columns.



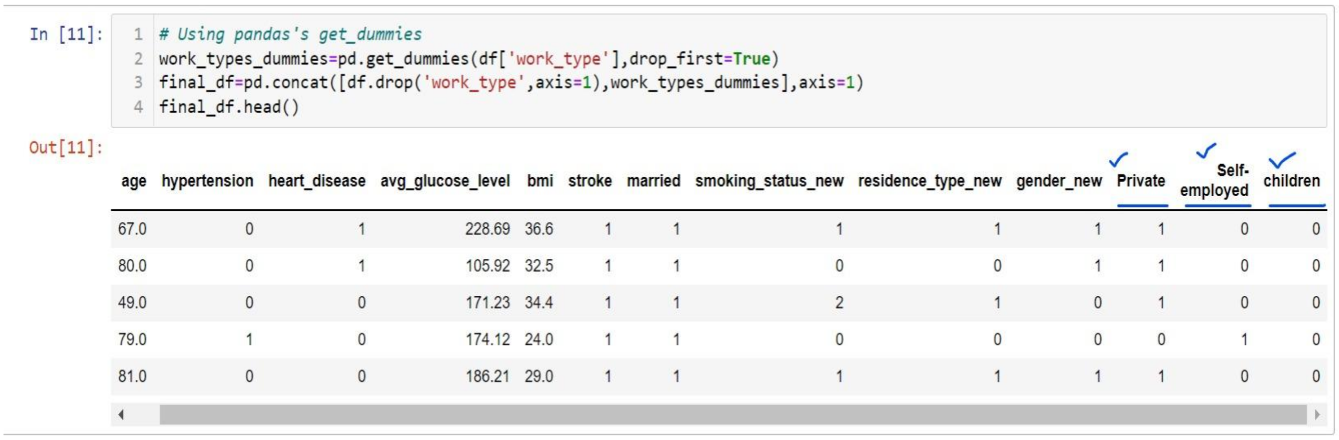
Handling nominal categorical features

In [10]:
```python
# Using sklearn's LabelEncoder
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

le.fit(df['Residence_type'])
df['residence_type_new'] = le.transform(df['Residence_type'])
le.fit(df['gender'])
df['gender_new'] = le.transform(df['gender'])
df.drop(['Residence_type','gender'],axis=1,inplace=True)
df.head()
```

Out[10]:

	age	hypertension	heart_disease	work_type	avg_glucose_level	bmi	stroke	married	smoking_status_new	residence_type_new	gender_new
0	67.0	0	1	Private	228.69	36.6	1	1	1	1	1
1	80.0	0	1	Private	105.92	32.5	1	1	0	0	1
2	49.0	0	0	Private	171.23	34.4	1	1	2	1	0
3	79.0	1	0	Self-employed	174.12	24.0	1	1	0	0	0
4	81.0	0	0	Private	186.21	29.0	1	1	1	1	1

As ‘work_type’ column is with more than two distinct values, we applied panda’s get_dummies dropping first unique value and got 3 new rows. After that, we deleted the ‘work_type’ column.



In [11]:
```python
# Using pandas's get_dummies
work_types_dummies=pd.get_dummies(df['work_type'],drop_first=True)
final_df=pd.concat([df.drop('work_type',axis=1),work_types_dummies],axis=1)
final_df.head()
```

Out[11]:

age	hypertension	heart_disease	avg_glucose_level	bmi	stroke	married	smoking_status_new	residence_type_new	gender_new	✓ Private	✓ Self-employed	✓ children
67.0	0	1	228.69	36.6	1	1	1	1	1	1	0	0
80.0	0	1	105.92	32.5	1	1	0	0	1	1	0	0
49.0	0	0	171.23	34.4	1	1	2	1	0	1	0	0
79.0	1	0	174.12	24.0	1	1	0	0	0	0	1	0
81.0	0	0	186.21	29.0	1	1	1	1	1	1	0	0

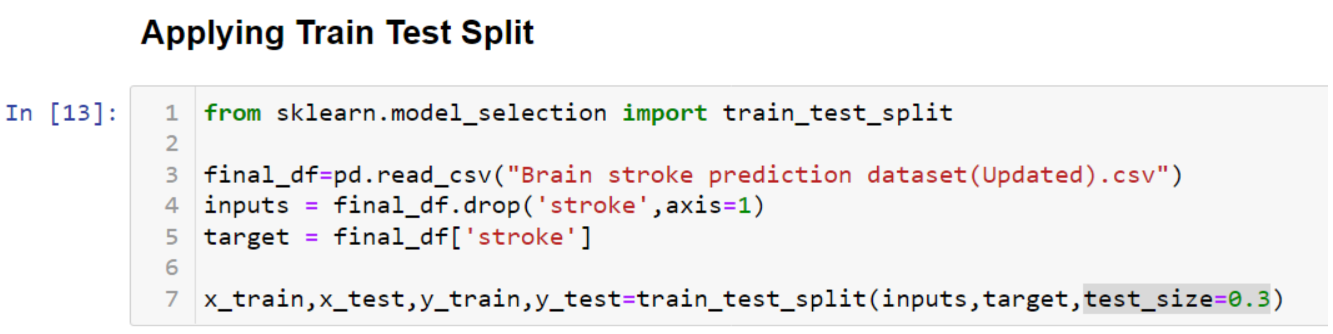
Finally we save the modified dataset to a csv file for future use.

Machine Learning Models

We splitted our dataset into 2 parts(training - 70%, testing - 30%). This is a supervised machine learning problem, we applied the following 6 models -

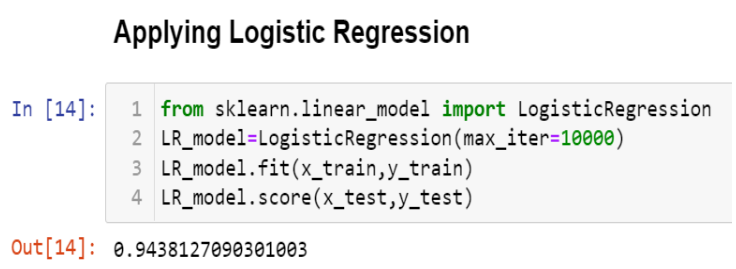
1. Logistic Regression
2. Decision Tree
3. Support Vector Machine(SVM)
4. Random Forest
5. Gaussian Naive Bayes Classifier
6. K nearest neighbor classifier

Though the problem is a classification problem, we tried to treat it as a regression problem at the same time as a classification problem. Therefore we converted the categorical values to numerical values, and were able to apply Logistic Regression.

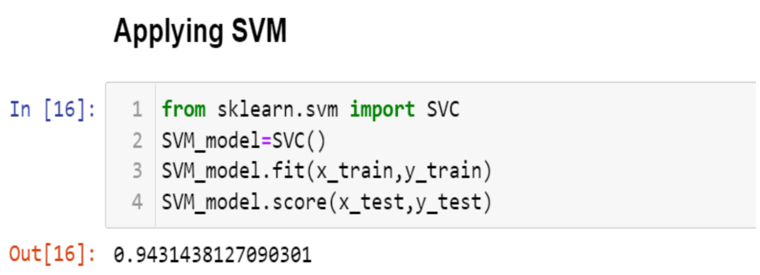


Applying Train Test Split

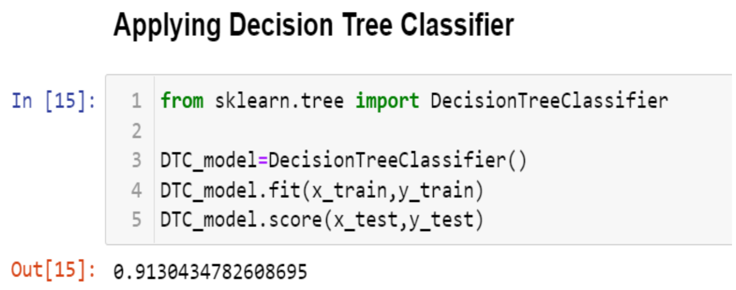
In [13]:
```python
from sklearn.model_selection import train_test_split

final_df=pd.read_csv("Brain stroke prediction dataset(Updated).csv")
inputs = final_df.drop('stroke',axis=1)
target = final_df['stroke']

x_train,x_test,y_train,y_test=train_test_split(inputs,target,test_size=0.3)
```


Applying Logistic Regression

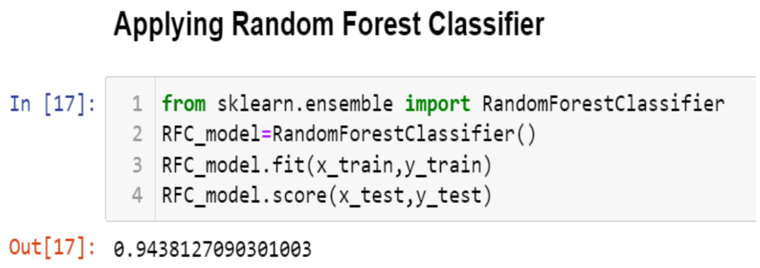
```
In [14]:
from sklearn.linear_model import LogisticRegression
LR_model=LogisticRegression(max_iter=10000)
LR_model.fit(x_train,y_train)
LR_model.score(x_test,y_test)

Out[14]: 0.9438127090301003
```


Applying SVM

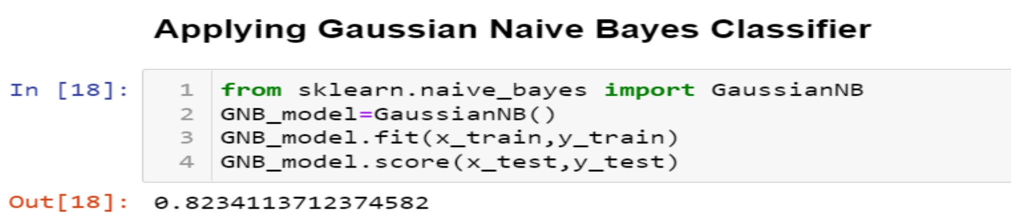
```
In [16]:
from sklearn.svm import SVC
SVM_model=SVC()
SVM_model.fit(x_train,y_train)
SVM_model.score(x_test,y_test)

Out[16]: 0.9431438127090301
```


Applying Decision Tree Classifier

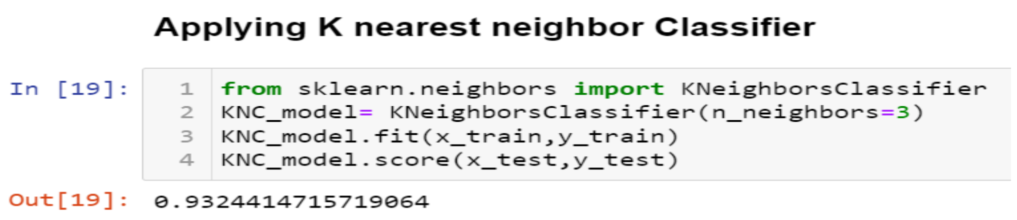
```
In [15]:
from sklearn.tree import DecisionTreeClassifier

DTC_model=DecisionTreeClassifier()
DTC_model.fit(x_train,y_train)
DTC_model.score(x_test,y_test)

Out[15]: 0.9130434782608695
```


Applying Random Forest Classifier

```
In [17]:
from sklearn.ensemble import RandomForestClassifier
RFC_model=RandomForestClassifier()
RFC_model.fit(x_train,y_train)
RFC_model.score(x_test,y_test)

Out[17]: 0.9438127090301003
```


Applying Gaussian Naive Bayes Classifier

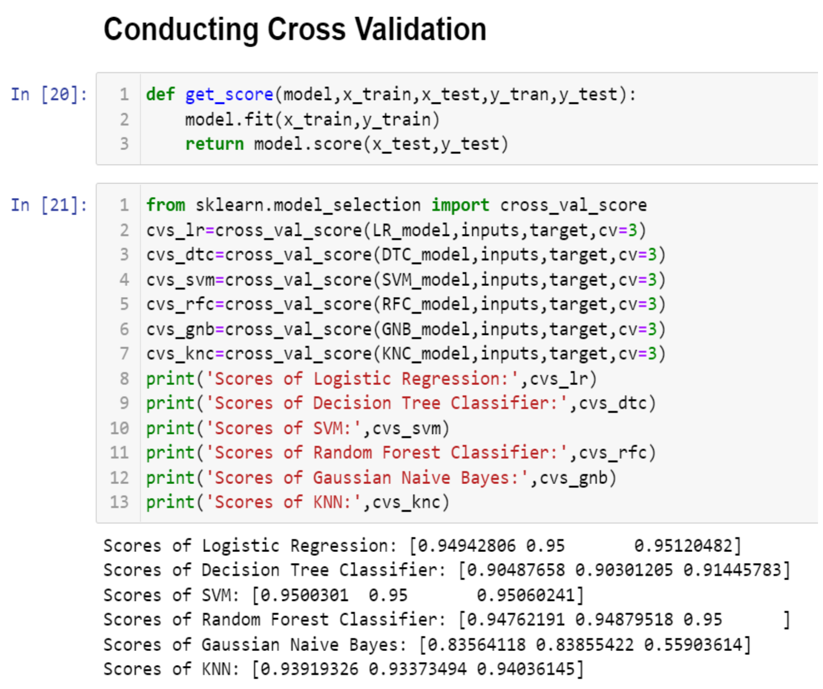
```
In [18]:
from sklearn.naive_bayes import GaussianNB
GNB_model=GaussianNB()
GNB_model.fit(x_train,y_train)
GNB_model.score(x_test,y_test)

Out[18]: 0.8234113712374582
```


Applying K nearest neighbor Classifier

```
In [19]:
from sklearn.neighbors import KNeighborsClassifier
KNC_model= KNeighborsClassifier(n_neighbors=3)
KNC_model.fit(x_train,y_train)
KNC_model.score(x_test,y_test)

Out[19]: 0.9324414715719064
```

Comparison of Performance Scores of ML Models

In order to get the balanced fit model we conduct Stratified KFold cross validation. We got the accuracy scores using 3 folds after applying cross_val_score on 6 models.



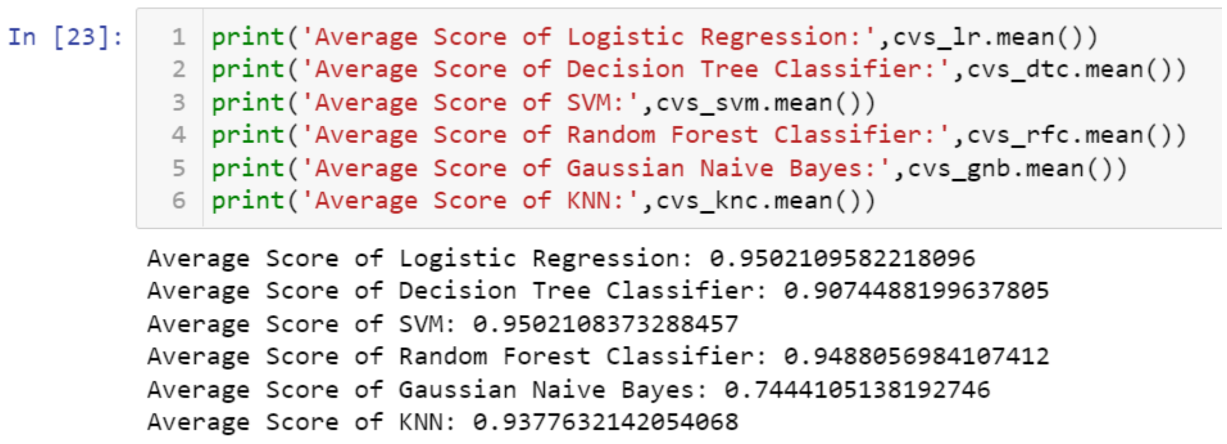
Conducting Cross Validation

```
In [20]:
def get_score(model,x_train,x_test,y_tran,y_test):
    model.fit(x_train,y_train)
    return model.score(x_test,y_test)
```

```
In [21]:
from sklearn.model_selection import cross_val_score
cvs_lr=cross_val_score(LR_model,inputs,target,cv=3)
cvs_dtc=cross_val_score(DTC_model,inputs,target,cv=3)
cvs_svm=cross_val_score(SVM_model,inputs,target,cv=3)
cvs_rfc=cross_val_score(RFC_model,inputs,target,cv=3)
cvs_gnb=cross_val_score(GNB_model,inputs,target,cv=3)
cvs_knc=cross_val_score(KNC_model,inputs,target,cv=3)
print('Scores of Logistic Regression:',cvs_lr)
print('Scores of Decision Tree Classifier:',cvs_dtc)
print('Scores of SVM:',cvs_svm)
print('Scores of Random Forest Classifier:',cvs_rfc)
print('Scores of Gaussian Naive Bayes:',cvs_gnb)
print('Scores of KNN:',cvs_knc)
```

```
Scores of Logistic Regression: [0.94942806 0.95       0.95120482]
Scores of Decision Tree Classifier: [0.90487658 0.90301205 0.91445783]
Scores of SVM: [0.9500301  0.95       0.95060241]
Scores of Random Forest Classifier: [0.94762191 0.94879518 0.95      ]
Scores of Gaussian Naive Bayes: [0.83564118 0.83855422 0.55903614]
Scores of KNN: [0.93919326 0.93373494 0.94036145]
```

We calculated the mean of the score lists and led to the conclusion that Logistic Regression and SVM performs well with the accuracy of around 95%.



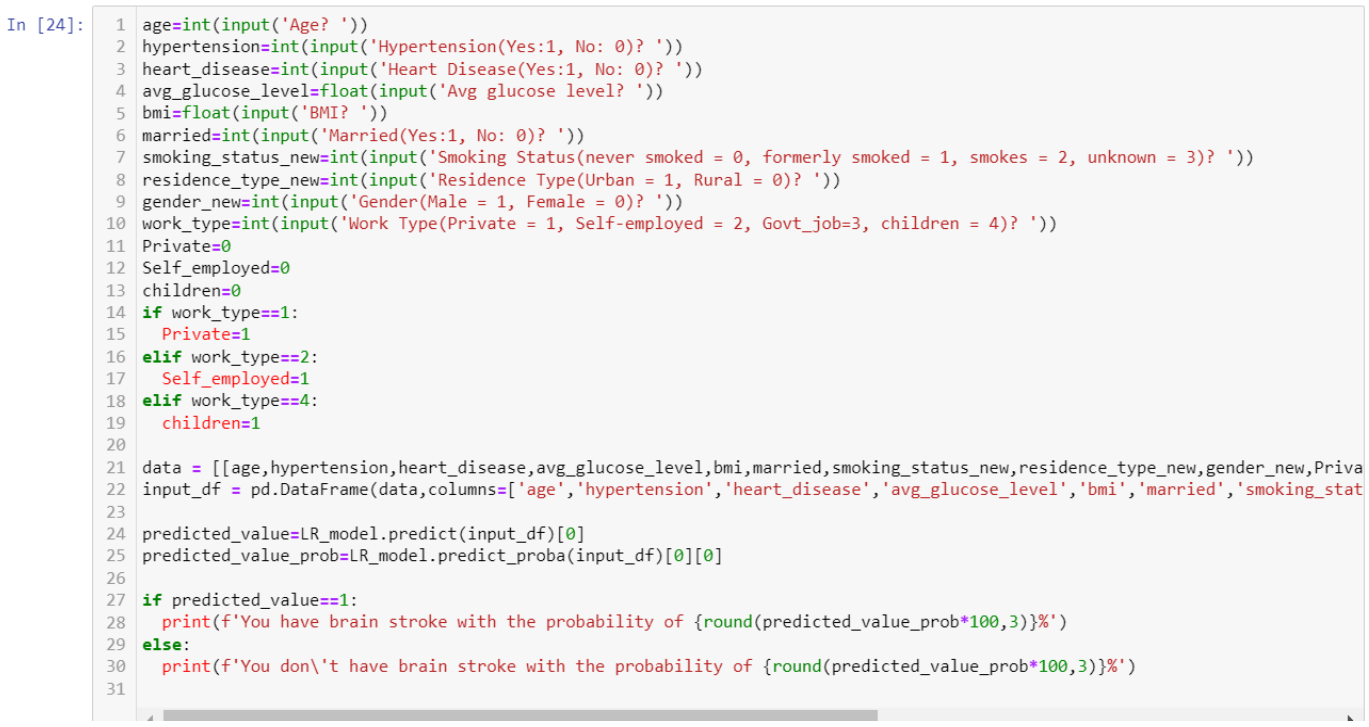
```
In [23]:
print('Average Score of Logistic Regression:',cvs_lr.mean())
print('Average Score of Decision Tree Classifier:',cvs_dtc.mean())
print('Average Score of SVM:',cvs_svm.mean())
print('Average Score of Random Forest Classifier:',cvs_rfc.mean())
print('Average Score of Gaussian Naive Bayes:',cvs_gnb.mean())
print('Average Score of KNN:',cvs_knc.mean())
```

```
Average Score of Logistic Regression: 0.9502109582218096
Average Score of Decision Tree Classifier: 0.9074488199637805
Average Score of SVM: 0.9502108373288457
Average Score of Random Forest Classifier: 0.9488056984107412
Average Score of Gaussian Naive Bayes: 0.7444105138192746
Average Score of KNN: 0.9377632142054068
```

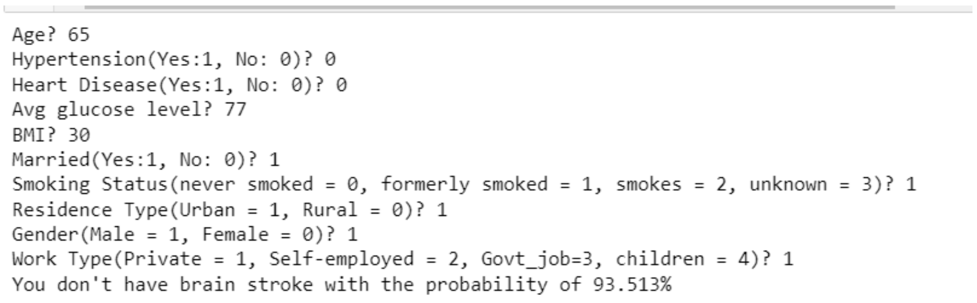
Predicting result using Logistic Regression model

Users will be asked some questions. Depending on the user's answer LR model will predict the chance of having a brain stroke with a probability.

Predicting Brain Stroke depending upon user input



```
In [24]:
age=int(input('Age? '))
hypertension=int(input('Hypertension(Yes:1, No: 0)? '))
heart_disease=int(input('Heart Disease(Yes:1, No: 0)? '))
avg_glucose_level=float(input('Avg glucose level? '))
bmi=float(input('BMI? '))
married=int(input('Married(Yes:1, No: 0)? '))
smoking_status_new=int(input('Smoking Status(never smoked = 0, formerly smoked = 1, smokes = 2, unknown = 3)? '))
residence_type_new=int(input('Residence Type(Urban = 1, Rural = 0)? '))
gender_new=int(input('Gender(Male = 1, Female = 0)? '))
work_type=int(input('Work Type(Private = 1, Self-employed = 2, Govt_job=3, children = 4)? '))
Private=0
Self_employed=0
children=0
if work_type==1:
  Private=1
elif work_type==2:
  Self_employed=1
elif work_type==4:
  children=1

data = [[age,hypertension,heart_disease,avg_glucose_level,bmi,married,smoking_status_new,residence_type_new,gender_new,Priva
input_df = pd.DataFrame(data,columns=['age','hypertension','heart_disease','avg_glucose_level','bmi','married','smoking_stat

predicted_value=LR_model.predict(input_df)[0]
predicted_value_prob=LR_model.predict_proba(input_df)[0][0]

if predicted_value==1:
  print(f'You have brain stroke with the probability of {round(predicted_value_prob*100,3)}%')
else:
  print(f'You don\'t have brain stroke with the probability of {round(predicted_value_prob*100,3)}%')

```



```
Age? 65
Hypertension(Yes:1, No: 0)? 0
Heart Disease(Yes:1, No: 0)? 0
Avg glucose level? 77
BMI? 30
Married(Yes:1, No: 0)? 1
Smoking Status(never smoked = 0, formerly smoked = 1, smokes = 2, unknown = 3)? 1
Residence Type(Urban = 1, Rural = 0)? 1
Gender(Male = 1, Female = 0)? 1
Work Type(Private = 1, Self-employed = 2, Govt_job=3, children = 4)? 1
You don't have brain stroke with the probability of 93.513%
```

Discussion

We have implemented 6 models in this problem.Out of these 6 models , by using K-Fold Cross validation, we have analyzed those models and compared validation values and came to the conclusion that Logistic Regression works better than other models for our data set.

After analyzing the models we have come to learn about feature engineering, how to fit a model, hyper parameter tuning, regression & classification techniques. We also tried to solve a classification problem with logistic regression.