Stroke Prediction

Chapter-1

Introduction

1.1. General Introduction

1.1.1. Description of topic under analysis

A stroke, sometimes called a "brain attack," occurs when blood flow to an area in the brain is cut off. The brain cells, deprived of the oxygen and glucose needed to survive, die. If a stroke is not caught early, permanent brain damage or death can result.

According to the World Health Organization (WHO) stroke is the 2nd leading cause of death globally, responsible for approximately 11% of total deaths. This dataset is used to predict whether a patient is likely to get stroke based onthe input parameters like gender, age, various diseases, and smoking status. Each row in the data provides relevant information about the patient.

There can be various factors related to occurrence to stroke. So, using the data we try to list out the potential factors by using various visualization techniques.

1.1.2. Problem Statement

Visualize the relationships between various Healthy and Unhealthy habits to Heart Strokes, and there by predict the stroke probability with best model and hyper tuned parameters.

1.1.3. Intended Operations to be performed

Data Pre-processing: Loading, Missing values and feature extraction

Explanatory data analysis using various types of graphs

Apply different Machine Learning models for prediction

1.1.4. End Users

The analysis can be used by specialist, doctors and healthcare centres to predict heart strokes early.

1.2. Data Collection

Data collection from Kaggle

The reference websites that are considered for creation of the project are:

https://www.healthline.com/health/stroke-vs-heart-attack

https://www.webmd.com/heart-disease/stroke
https://www.heartandstroke.ca/stroke/what-is-stroke
https://www.health.harvard.edu/heart-health/stroke-after-a-heart-attack-whats-the-risk

1.3. Phases of Analysis

1.3.1. Block Diagram

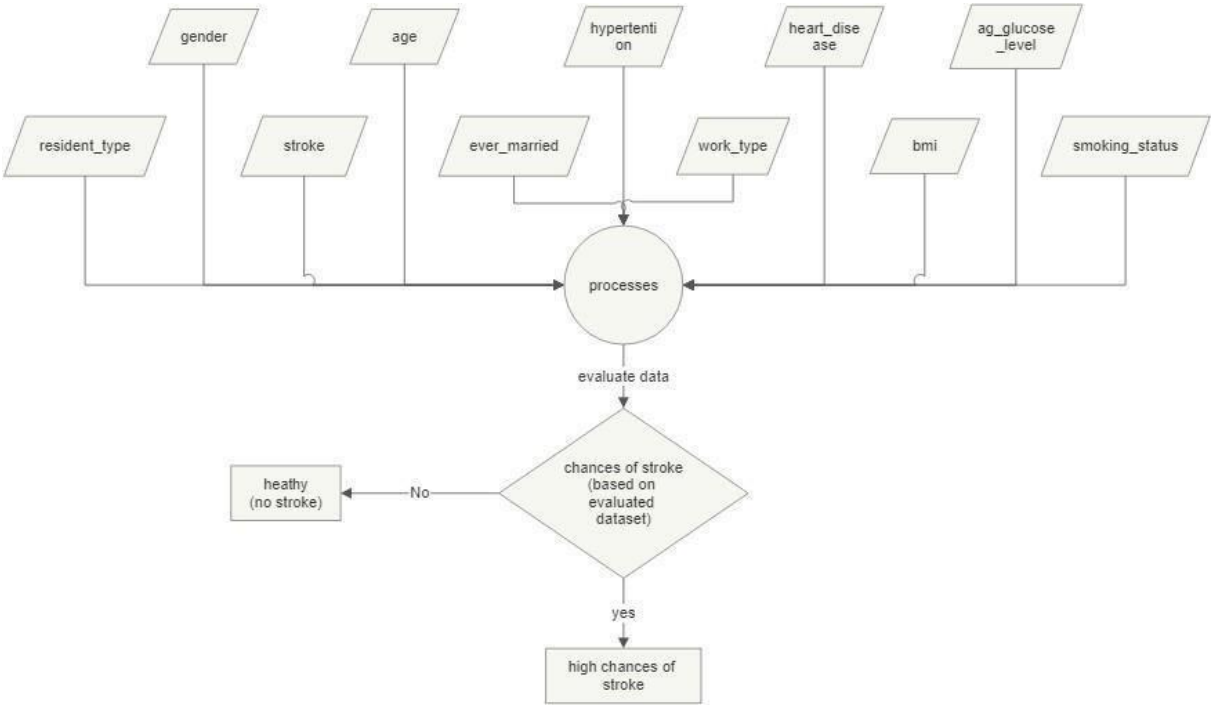


Fig 1.1: Block Diagram

1.3.2. Attributes considered for studying

In the current dataset, there are 11 features and one binary target which are considered as attributes for studying . A brief information about the features are given below:

1. **id**: Identification number of the individual.
2. **gender**: Gender of the individual.
3. **hypertension**: Health related parameter, does person have hypertension.
4. **heart_disease**: Health related parameter, does person have heart disease.
5. **ever_married**: Personal information, is person married or not?
6. **work_type**: Nature of work place.
7. **Residence_type**: Residence type of the individual.

8. **avg_glucose_level**: average glucose level in blood of the individual.
9. **bmi**: body mass index of the individual.
10. **smoking_status**: Current smoking status of individual.
11. **stroke**: Our target, did person suffered stroke?

1.4. Tools/Platforms

1.4.1. Hardware Specifications

Processor	Core i5
RAM	8.00 GB
Memory	512 GB
System type	64-bit operating system, x64-based processor

Table No 1.1: Hardware Specifications

1.4.2. Software Specifications

OS	Windows 11
Language	Python
Software Development Kit	Jupiter Lab

Table No 1.2: Software Specifications

1.4.3. Packages to be imported

Packages that we imported in this project are:

- NumPy
- Pandas
- Matplotlib
- Seaborn
- Sklearn
- XGBoos

1.4.4. Gantt Chart

GANTT CHART stroke prediction

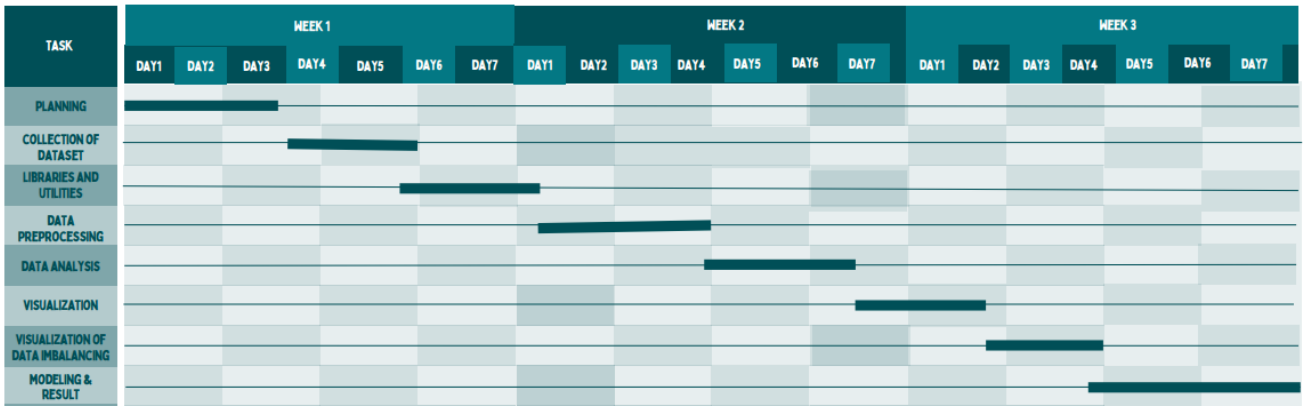


Table No 1.4: Gantt Chart

Chapter-2:
Literature Review

2.1. Summary of Paper Studies

- **Analyzing the Performance of Stroke Prediction using ML Classification Algorithms**

  By: Gangavarapu Sailasya, Gorli L Aruna Kumari Department of Computer Science and Engineering GITAM Institute of Technology, GITAM (Deemed to be University) Visakhapatnam, Andhra Pradesh – 530045

  A Stroke is a health condition that causes damage by tearing the blood vessels in the brain. It can also occur when there is a halt in the blood flow and other nutrients to the brain. According to the World Health Organization (WHO), stroke is the leading cause of death and disability globally. Most of the work has been carried out on the prediction of heart stroke but very few works show the risk of a brain stroke. With this thought, various machine learning models are built to predict the possibility of stroke in the brain. This paper has taken various physiological factors and used machine learning algorithms like Logistic Regression, Decision Tree Classification, Random Forest Classification, K-Nearest Neighbors, Support Vector Machine and Naïve Bayes Classification to train five different models for accurate prediction. The algorithm that best performed this task is Naïve Bayes that gave an accuracy of approximately 82%.

  Keywords—Stroke; machine learning; logistic regression; decision tree classification; random forest classification; k-nearest neighbors; support vector machine; Naïve Bayes classification

- **Stroke Prediction using Distributed Machine Learning Based on Apache Spark**

  By: Hager Ahmed, Sara F. Abd-el ghany, Eman M.G.Youn, Nahla F.Omran, Abdelmgeid A.Ali, Faculty of Computers and Information, Minia University, Egypt , Department of Computer Science and Faculty of Science, South Valley University, Egypt

Stroke is one of death causes and one the primary causes of severe long-term weakness in the world. In this paper, we compare different distributed machine learning algorithms for stroke prediction on the Healthcare Dataset Stroke. This work is implemented by a big data platform that is Apache Spark. Apache Spark is one of the most popular big data platforms that handle big data and includes an MLlib library. MLlib is an API integrated with Spark to provide machine learning algorithms. Four types of machine learning classification algorithms were applied; Decision Tree, Support Vector Machine, Random Forest Classifier, and Logistic Regression were used to build the stroke prediction model. The hyperparameter tuning and cross-validation were applied with machine learning algorithms to enhance results. Accuracy, Precision, Recall, and F1-measure were used to calculate performance measures of machine learning models. The results showed that Random Forest Classifier has achieved the best accuracy at 90 %.

Keywords: Stroke; Stroke Prediction; Machine Learning; Big Bata; Apache Spark

- **Stroke risk prediction using machine learning: a prospective cohort study of 0.5 million Chinese adults**

  By: Matthew Chun, Robert Clarke, Benjamin J Cairns, David Clifton, Derrick Bennett, Yiping Chen, Yu Guo, Pei Pei, Jun Lv, Canqing Yu, Ling Yang, Liming Li, Zhengming Chen, Tingting Zhu, the China Kadoorie Biobank Collaborative Group
  Journal of the American Medical Informatics Association, Volume 28, Issue 8, August 2021

  Novel risk scores for stroke have been developed using data from a contemporary cohort of 0.5 million Chinese adults. Use of ML techniques improved risk prediction over traditional Cox model approaches, with GBT providing the best discrimination and calibration performance. An ensemble approach was also proposed to screen for individuals at high risk of stroke who may benefit from more intensive treatment. The ensemble approach identified high-risk individuals with marginal improvements to accuracy, specificity, and PPV over either Cox or GBT models alone. By identifying a small portion of individuals who would benefit from ML predictions, our ensemble approach provides an incremental benefit beyond current clinical practice that has potential to translate into important benefits for population health and facilitate the adoption of ML-based risk calculators in clinical practice.

- **An Explainable Machine Learning Pipeline for Stroke Prediction on Imbalanced Data**

  By: Christos Kokkotis ,Georgios Giarmatzis ,Erasmia Giannakou ,Serafeim Moustakidis ,Themistoklis Tsatalas ,Dimitrios Tsiptsios ,Konstantinos Vadikolias and Nikolaos Aggelousis

  Stroke is an acute neurological dysfunction attributed to a focal injury of the central nervous system due to reduced blood flow to the brain. Nowadays, stroke is a global threat associated with premature death and huge economic consequences. Hence, there is an urgency to model the effect of several risk factors on stroke occurrence, and artificial intelligence (AI) seems to be the appropriate tool. In the present study, we aimed to (i) develop reliable machine learning (ML) prediction models for stroke disease; (ii) cope with a typical severe class imbalance problem, which is posed due to the stroke patients' class being significantly smaller than the healthy class; and (iii) interpret the model output for understanding the decision-making mechanism. The effectiveness of the proposed ML approach was investigated in a comparative analysis with six well-known classifiers with respect to metrics that are related to both generalization capability and prediction accuracy. The best overall false-negative rate was achieved by the Multi-Layer Perceptron (MLP) classifier (18.60%). Shapley Additive Explanations (SHAP) were employed to investigate the impact of the risk factors on the prediction output. The proposed AI method could lead to the creation of advanced and effective risk stratification strategies for each stroke patient, which would allow for timely diagnosis and the right treatments.

  Keywords: stroke; clinical data; machine learning; prognosis; interpretation

2.2. Integrated summary of the Literature studied

Stroke is a medical condition that causes brain damage by tearing blood vessels. The majority of research has focused on the prediction of heart attacks, but very few studies have focused on the risk of a brain attack. The papers used machine learning algorithms to train models for accurate prediction based on various physiological factors. Naïve Bayes, Random Forest and Multi-Layer Perceptron (MLP) classifier are some of the algorithms that gave the best results. Stroke is an acute neurological dysfunction caused by a central nervous system injury caused by decreased blood flow to the brain.

There is a pressing need to model the effect of various risk factors on stroke occurrence, and artificial intelligence (AI) appears to be the right tool.

Chapter-3

Implementation and Results

3.1. Phase 1: Installing Packages

```
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt
import seaborn as sns
```

3.2. Phase 2: Importing and reading data

```
data = pd.read_csv('healthcare-dataset-stroke-data.csv')
data
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
...	...	...	...	...	...	...	...	...	...	...	...	...
5105	18234	Female	80.0	1	0	Yes	Private	Urban	83.75	NaN	never smoked	0
5106	44873	Female	81.0	0	0	Yes	Self-employed	Urban	125.20	40.0	never smoked	0
5107	19723	Female	35.0	0	0	Yes	Self-employed	Rural	82.99	30.6	never smoked	0
5108	37544	Male	51.0	0	0	Yes	Private	Rural	166.29	25.6	formerly smoked	0
5109	44679	Female	44.0	0	0	Yes	Govt_job	Urban	85.28	26.2	Unknown	0

5110 rows × 12 columns

Check for null values

```
data.isnull().sum()
```

```
id                   0
gender               0
```

```
age                    0
hypertension           0
heart_disease          0
ever_married           0
work_type              0
Residence_type         0
avg_glucose_level      0
bmi                  201
smoking_status         0
stroke                 0
dtype: int64
```

Since the BMI column contains null values we need to fix it

We use the mean value to fix those null values

```
avg = data['bmi'].mean()
avg
```

```
28.893236911794673
```

For adults, the normal BMI range is between 18.5 and 24.9 The average BMI calculation comes out to be more than normal. Hence, we can say that large proportion of the population in the given dataset is overweight.

```
data.bmi=(data.bmi.fillna(28.74))
```

After filling the NA values for BMI let's check again for any null values

```
data.isnull().sum()
```

```
id                   0
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
```

Hence no null values

3.3. Phase 3: Feature Datatypes

What data type variable are provided in the dataset and Initial insights about dataset

```
data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   id                 5110 non-null   int64  
 1   gender             5110 non-null   object 
 2   age                5110 non-null   float64
 3   hypertension       5110 non-null   int64  
 4   heart_disease      5110 non-null   int64  
 5   ever_married       5110 non-null   object 
 6   work_type          5110 non-null   object 
 7   Residence_type     5110 non-null   object 
 8   avg_glucose_level  5110 non-null   float64
 9   bmi                5110 non-null   float64
 10  smoking_status     5110 non-null   object 
 11  stroke             5110 non-null   int64  
dtypes: float64(3), int64(4), object(5)
memory usage: 479.2+ KB
```

```
data.describe()
```

	id	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000
mean	36517.829354	43.226614	0.097456	0.054012	106.147677	28.887209	0.048728
std	21161.721625	22.612647	0.296607	0.226063	45.283560	7.698075	0.215320
min	67.000000	0.080000	0.000000	0.000000	55.120000	10.300000	0.000000
25%	17741.250000	25.000000	0.000000	0.000000	77.245000	23.800000	0.000000
50%	36932.000000	45.000000	0.000000	0.000000	91.885000	28.400000	0.000000
75%	54682.000000	61.000000	0.000000	0.000000	114.090000	32.800000	0.000000
max	72940.000000	82.000000	1.000000	1.000000	271.740000	97.600000	1.000000

From above statistical description of the dataset we can conclude:

1. Mean age of people is around 43

2. Mean BMI is more than normal
3. Both Categorical and numerical features are present.

Categorical Features: gender, ever_married, work_type, Residence_type, smoking_status

Binary Numerical Features: hypertension,heart_disease, stroke

Continous Numerical Features: age, avg_glucose_level, bmi

1. Most of the data is categorical which need a special attention to visulaize those

Chapter-4

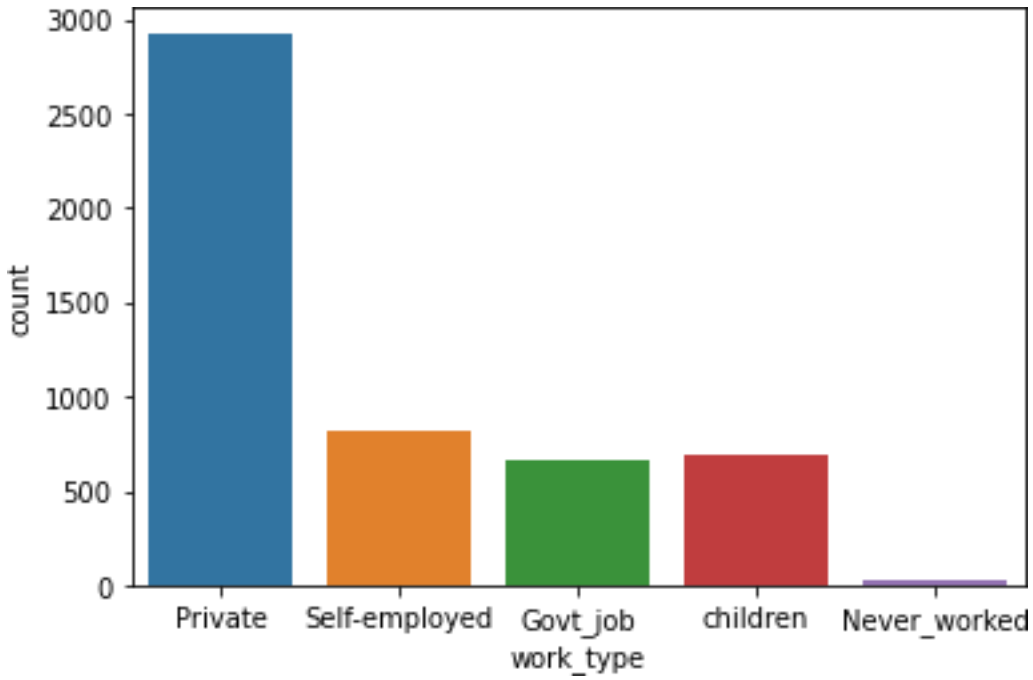
Implementation and Visualisations

4.1. Visualising data through graphs

Countplot to see the distribution of Categorical features:

```
sns.countplot(data['work_type'])
```

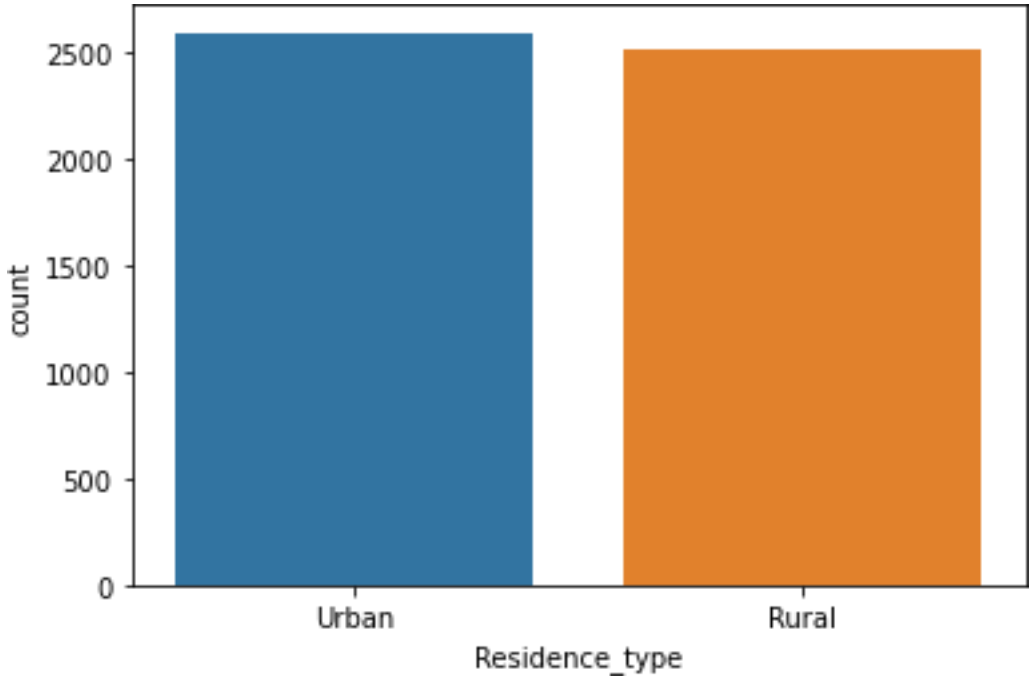
```
<AxesSubplot:xlabel='work_type', ylabel='count'>
```



Most people work in the Private sector

```
sns.countplot(data['Residence_type'])
```

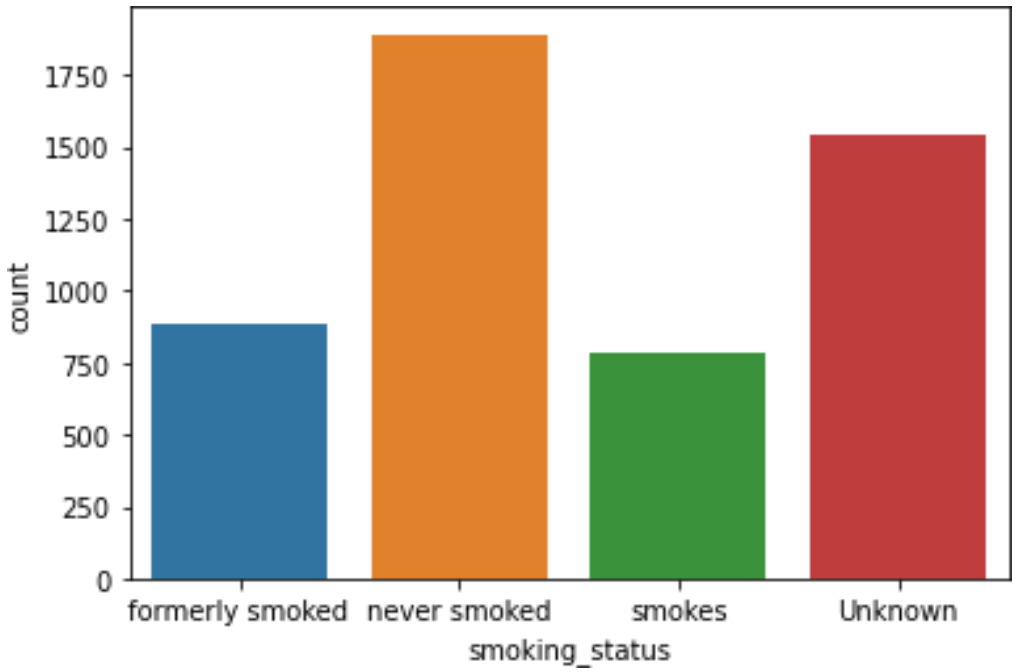
```
<AxesSubplot:xlabel='Residence_type', ylabel='count'>
```



Almost same number of people living in both areas

```
sns.countplot(data['smoking_status'])
```

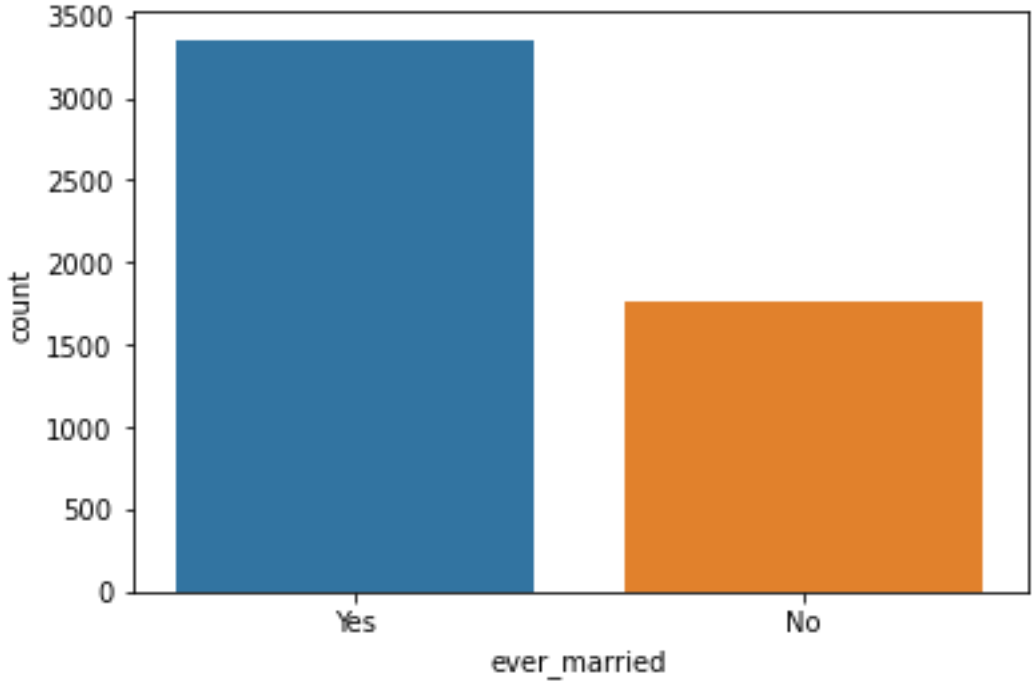
```
<AxesSubplot:xlabel='smoking_status', ylabel='count'>
```



Most people have never smoked

```
sns.countplot(data['ever_married'])
```

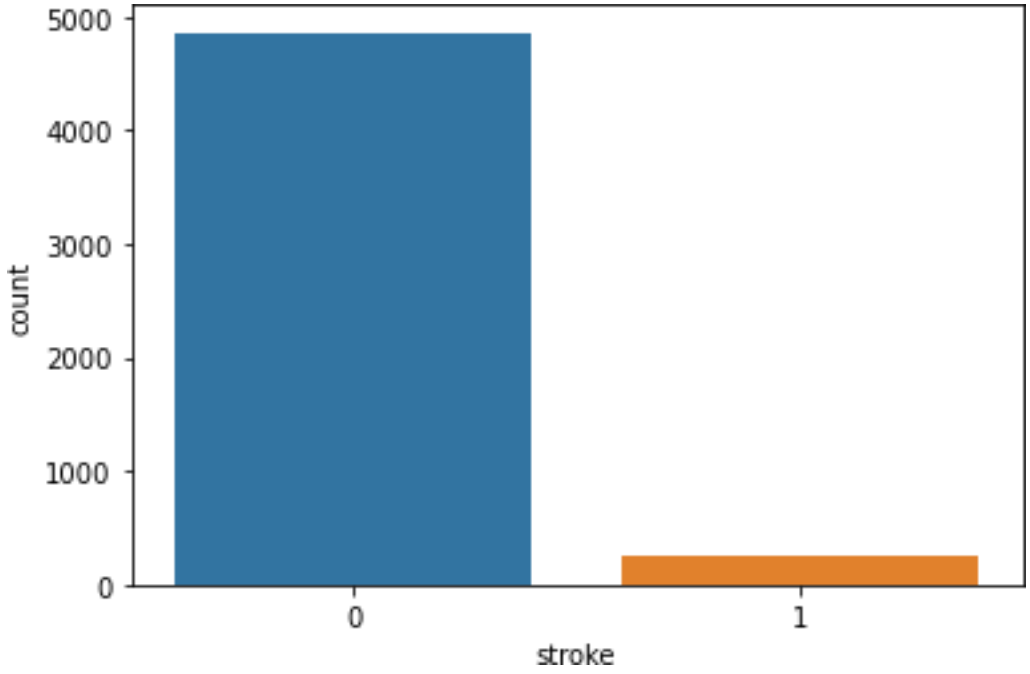
```
<AxesSubplot:xlabel='ever_married', ylabel='count'>
```



More people are married in the dataset population

```
sns.countplot(data['stroke'])
```

```
<AxesSubplot:xlabel='stroke', ylabel='count'>
```



The above graph shows that there is a HIGH IMBALANCE i.e., this is a highly unbalanced data distribution so while applying any ML model and training, either over sampling or under sampling has to be done to obtain best results.

Distplot to see the distribution of Continuous numerical features:

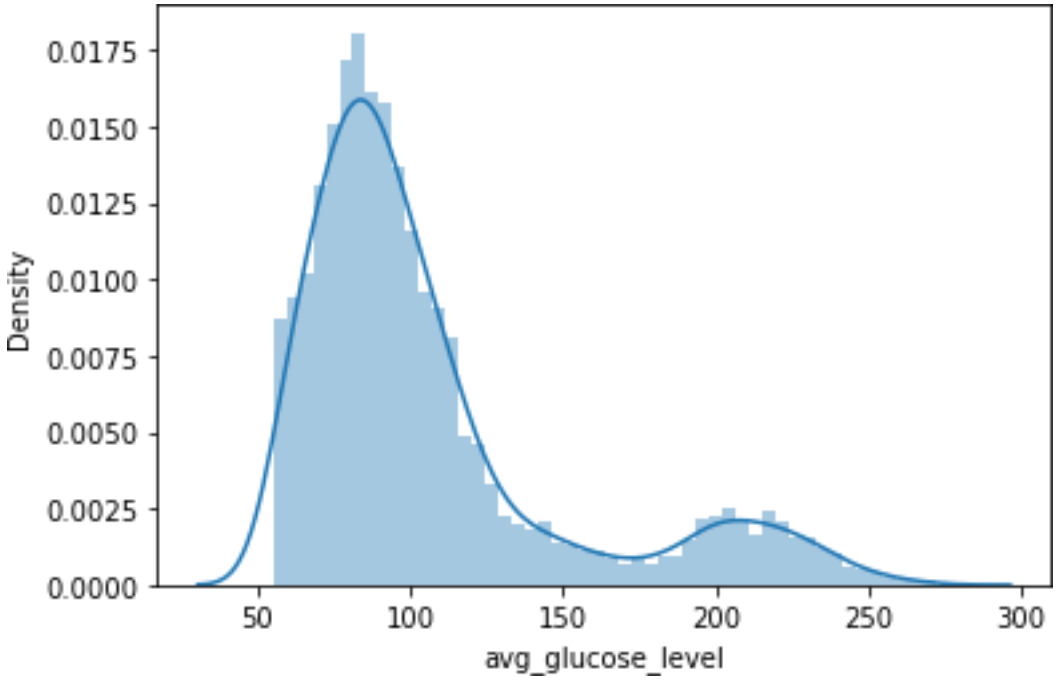
```
min_avg_glucose_level = min(data.avg_glucose_level)
max_avg_glucose_level = max(data.avg_glucose_level)
print(min_avg_glucose_level)
print(max_avg_glucose_level)
```

```
55.12
271.74
```

Since there is a high difference between the minimum and maximum values of average glucose level we need to standardize the column

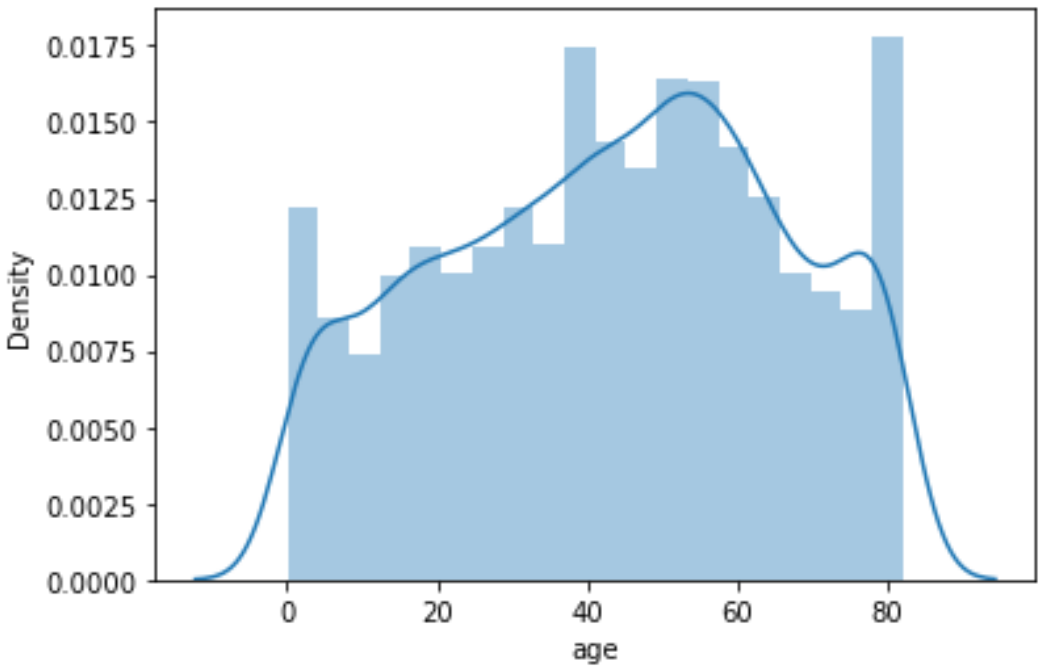
```
sns.distplot(data['avg_glucose_level'])
```

```
<AxesSubplot:xlabel='avg_glucose_level', ylabel='Density'>
```




```
sns.distplot(data['age'])
```

```
<AxesSubplot:xlabel='age', ylabel='Density'>
```



Mapping of categorical variables:

```
data['work_type'] = data['work_type'].map({'Private':0, 'Self-employed': 1,
'Govt_job':2, 'children':3, 'Never_worked':4})
```

```
data['gender'] = data['gender'].map({'Male':0, 'Female':1})
data['Residence_type'] = data['Residence_type'].map({'Urban':0, 'Rural':1})
data['smoking_status'] = data['smoking_status'].map({'formerly smoked':0,
'never smoked':1, 'smokes':2, 'Unknown':3})
data['ever_married'] = data['ever_married'].map({'Yes':0, 'No':1})
```

Checking the dataset after mapping of categorical variables

```
data
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	0.0	67.0	0	1	0	0	0	228.69	36.60	0	1
1	51676	1.0	61.0	0	0	0	1	1	202.21	28.74	1	1
2	31112	0.0	80.0	0	1	0	0	1	105.92	32.50	1	1
3	60182	1.0	49.0	0	0	0	0	0	171.23	34.40	2	1
4	1665	1.0	79.0	1	0	0	1	1	174.12	24.00	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...
5105	18234	1.0	80.0	1	0	0	0	0	83.75	28.74	1	0
5106	44873	1.0	81.0	0	0	0	1	0	125.20	40.00	1	0
5107	19723	1.0	35.0	0	0	0	1	1	82.99	30.60	1	0
5108	37544	0.0	51.0	0	0	0	0	1	166.29	25.60	0	0
5109	44679	1.0	44.0	0	0	0	2	0	85.28	26.20	3	0

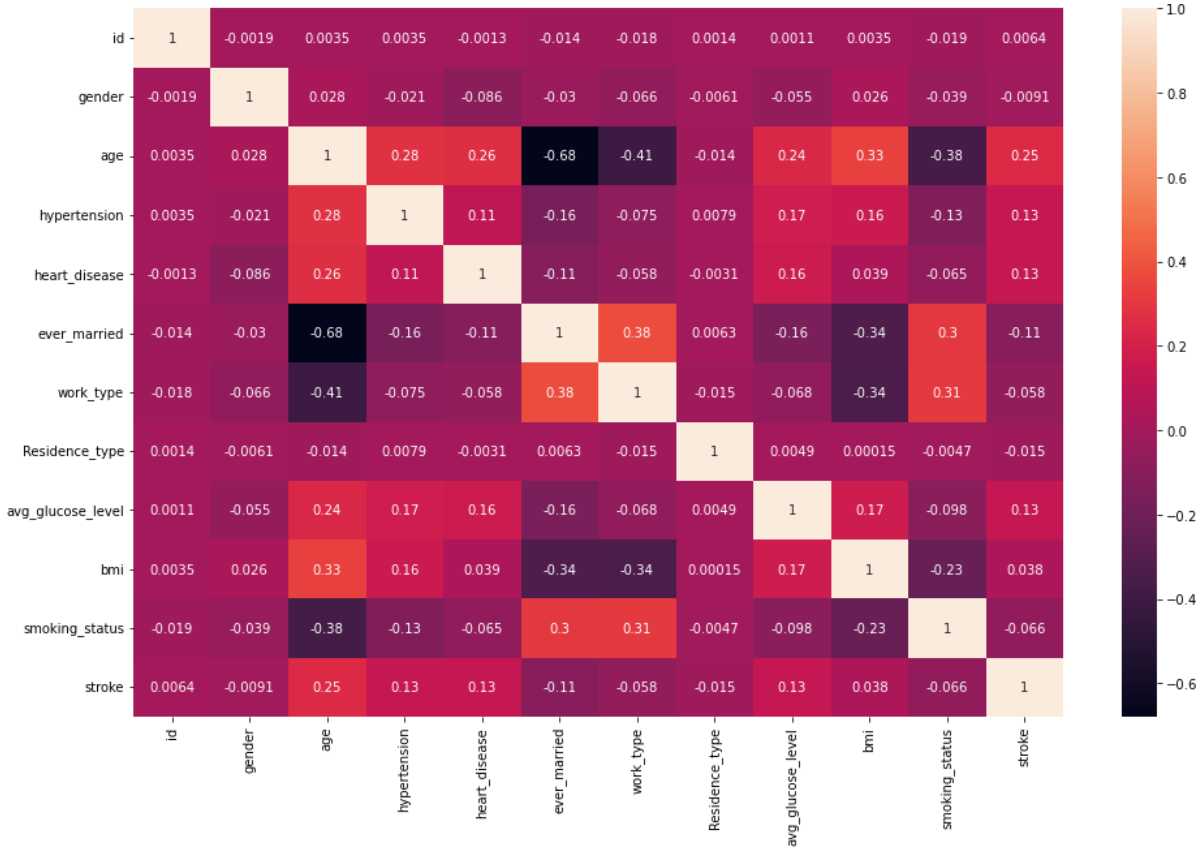
5110 rows × 12 columns

Correlation Heatmap

To check for any correlation between variables

```
plt.figure(figsize=(16,10))
sns.heatmap(data.corr(method='pearson'),annot=True)
```

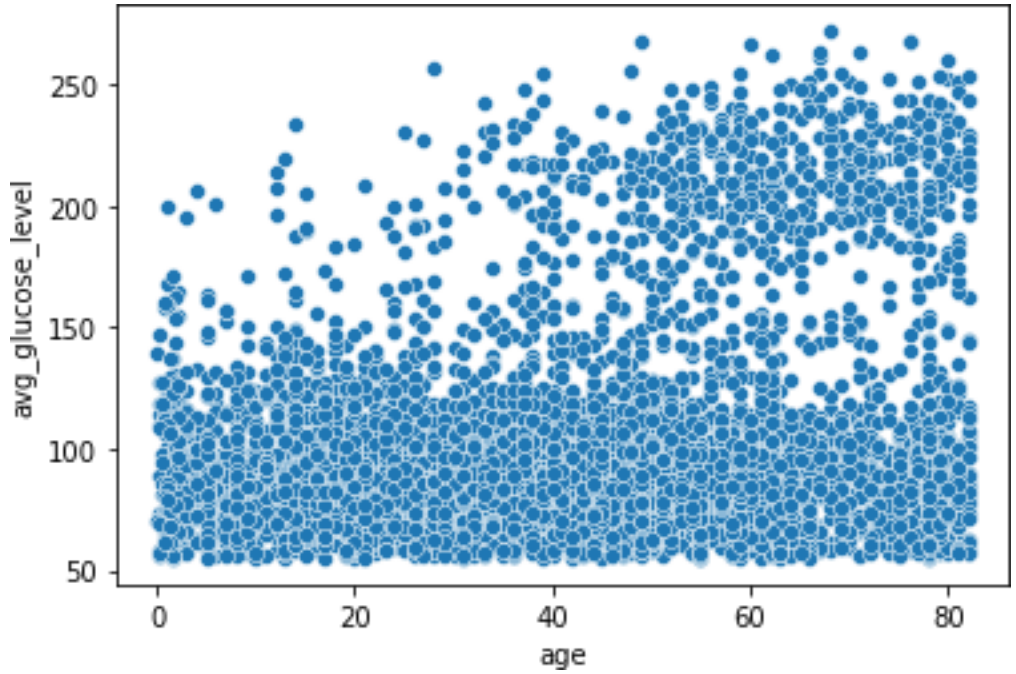
```
<AxesSubplot:>
```



From above figure we can say that marriage and age are negatively correlated in highest order and stroke and age have a postive correlation

```
sns.scatterplot(x=data['age'], y=data['avg_glucose_level'])
```

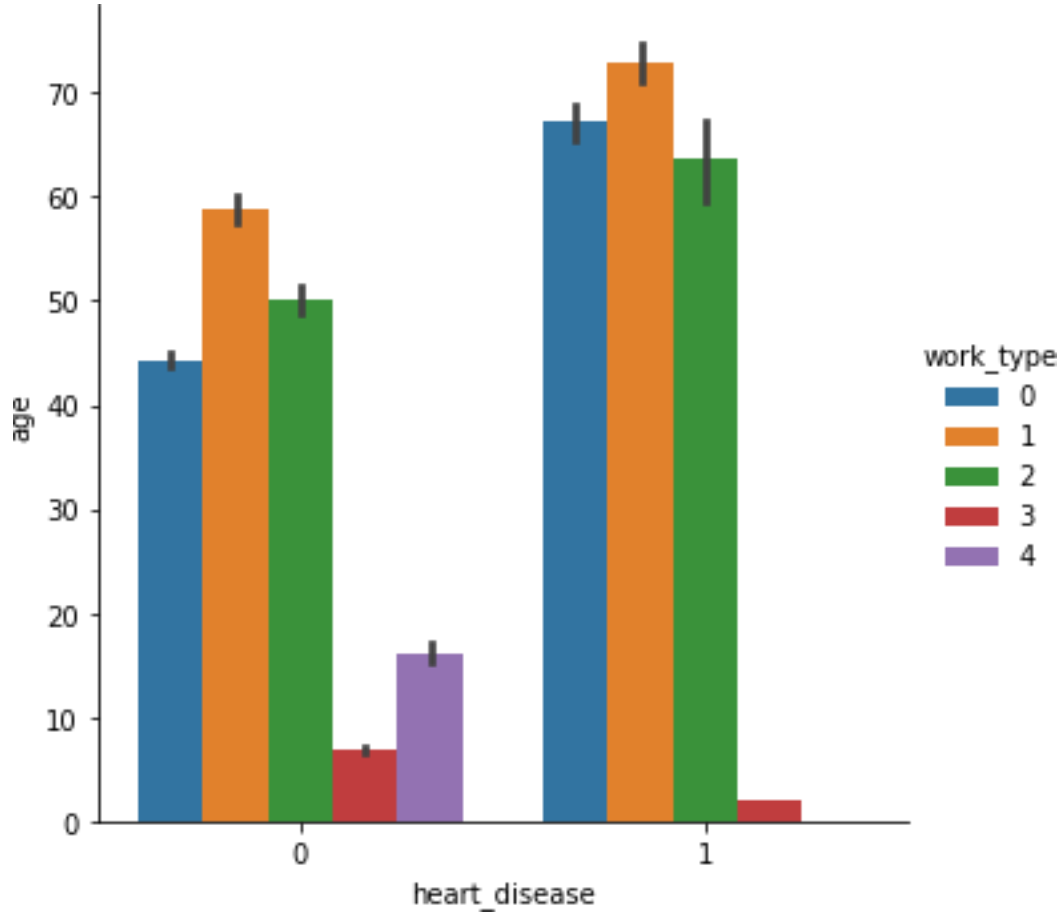
```
<AxesSubplot:xlabel='age', ylabel='avg_glucose_level'>
```



From above scatter plot we can say that as the age increases, the glucose level also increases.

```
sns.catplot(x='heart_disease',y='age', hue="work_type", kind="bar",
data=data)
```

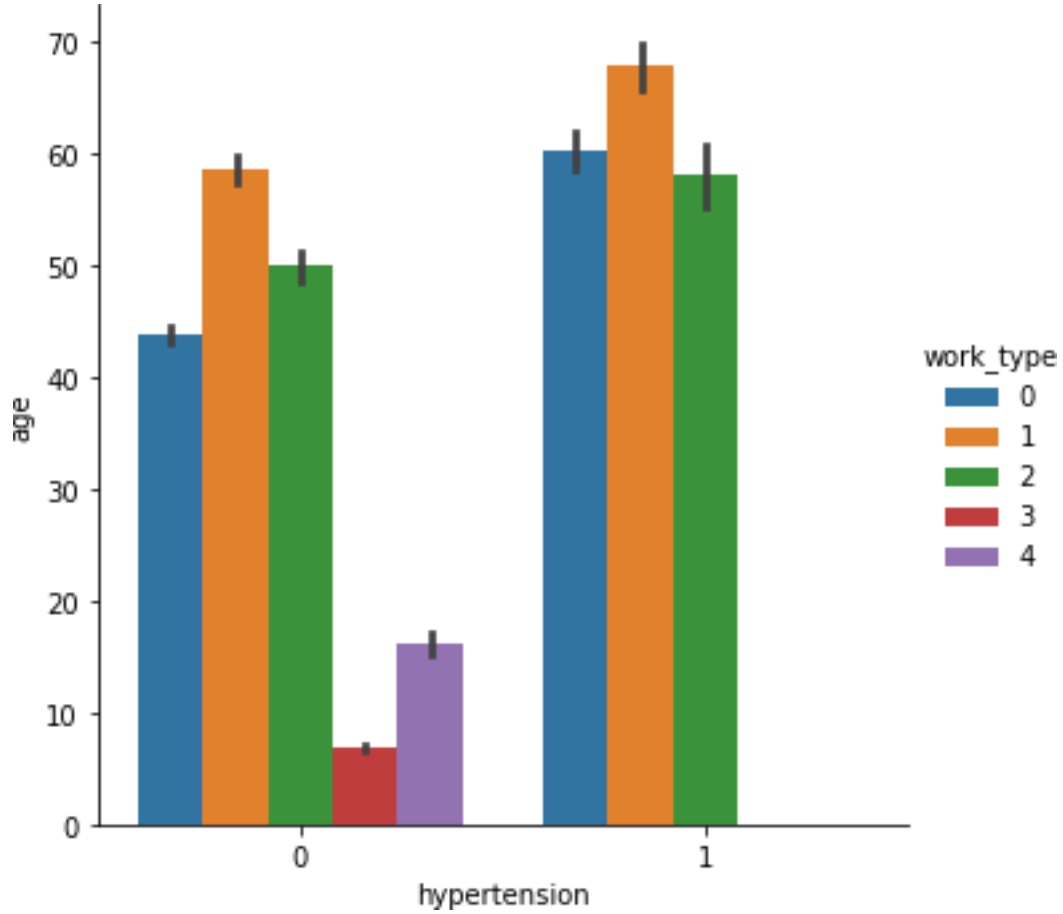
```
<seaborn.axisgrid.FacetGrid at 0x1899c41ad30>
```



People who are self employed have a greater tendency of having a heart disease and hypertension.

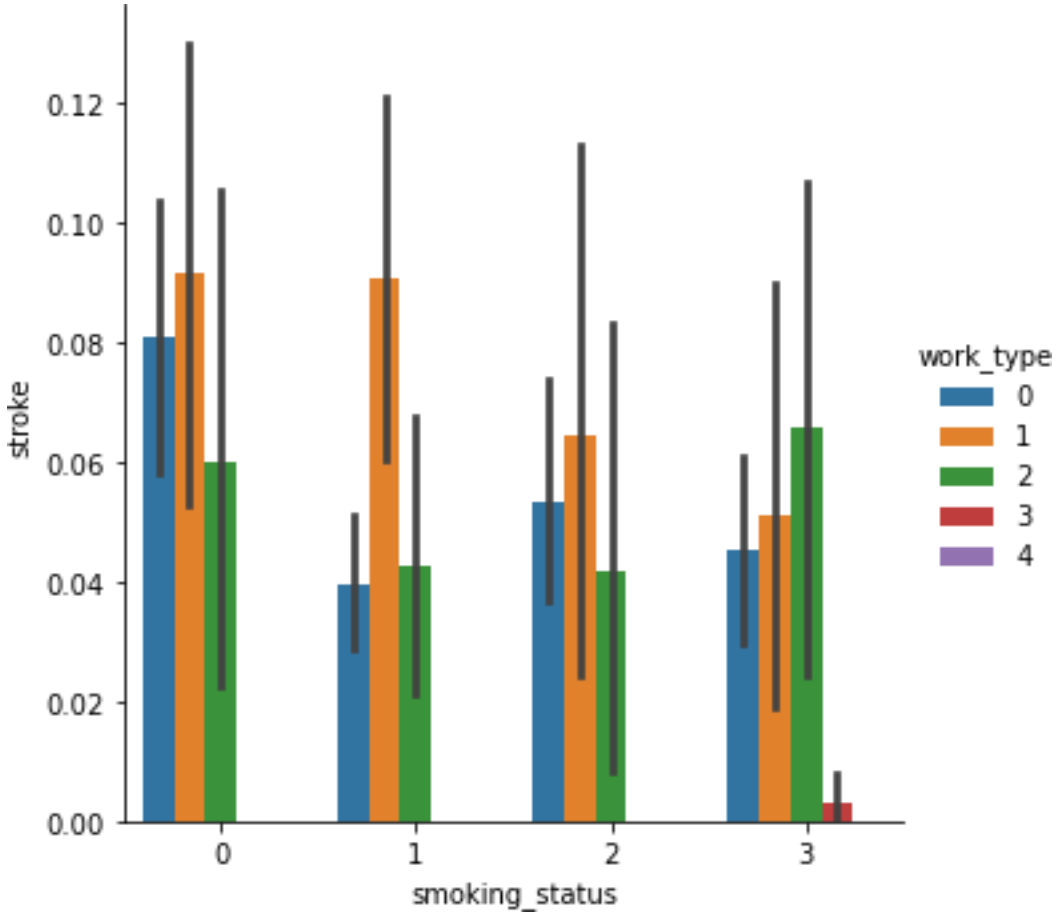
```
sns.catplot(x='hypertension',y='age', hue="work_type", kind="bar",
data=data)
```

```
<seaborn.axisgrid.FacetGrid at 0x1899cae0e20>
```




```
sns.catplot(x="smoking_status", y="stroke", hue="work_type", kind="bar",
data=data)
```

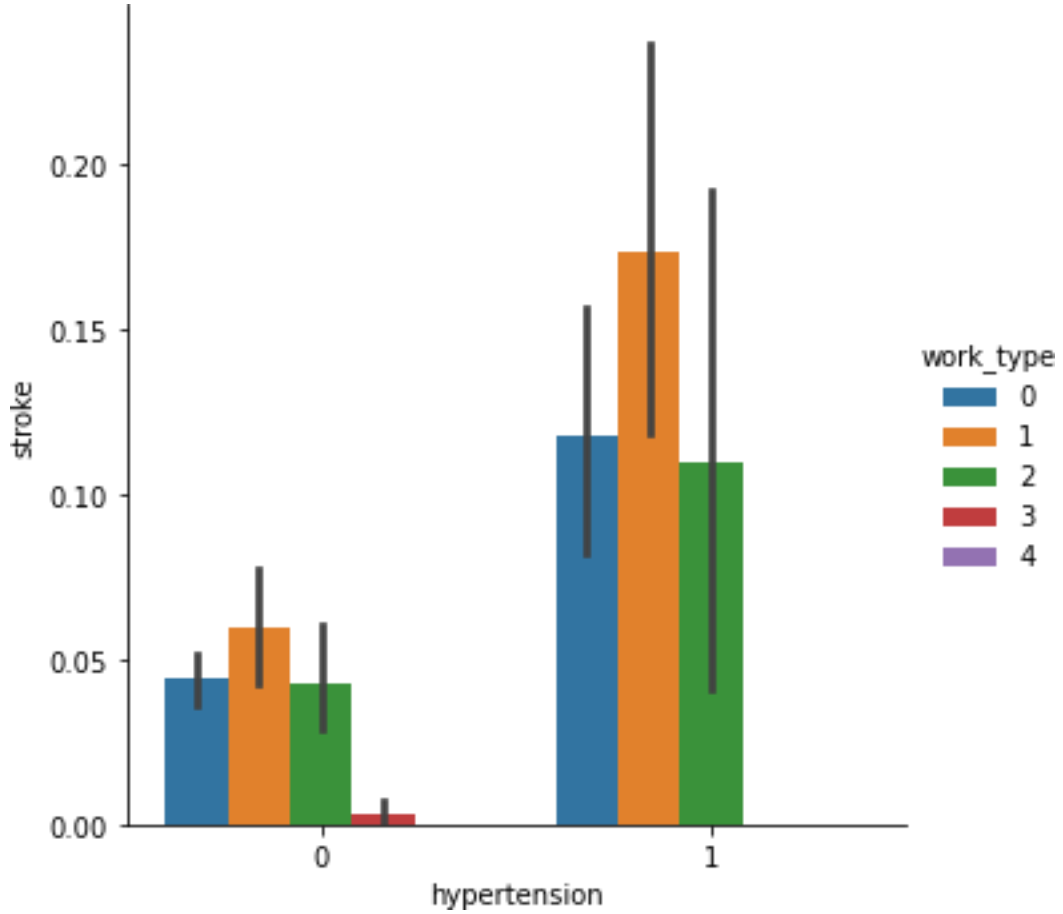
```
<seaborn.axisgrid.FacetGrid at 0x1899cbc2ee0>
```



Self employed people tend to smoke more and hence have a greater chance of stroke. Even if they don't smoke or used to smoke they still have a greater chance of stroke than other people.

```
sns.catplot(x="hypertension", y="stroke", hue="work_type", kind="bar",
data=data)
```

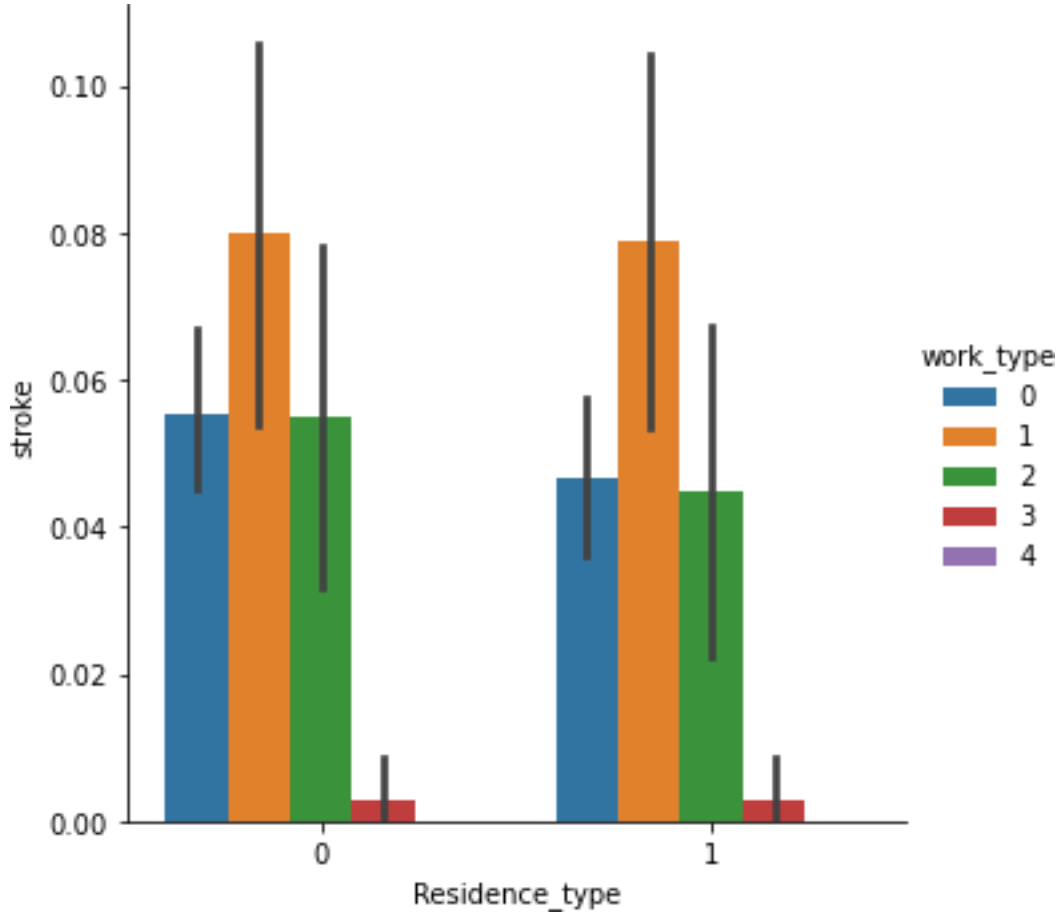
```
<seaborn.axisgrid.FacetGrid at 0x1899cb4a0a0>
```



There are more number of people who have hypertension and again self-employed people are at a greater risk of stroke

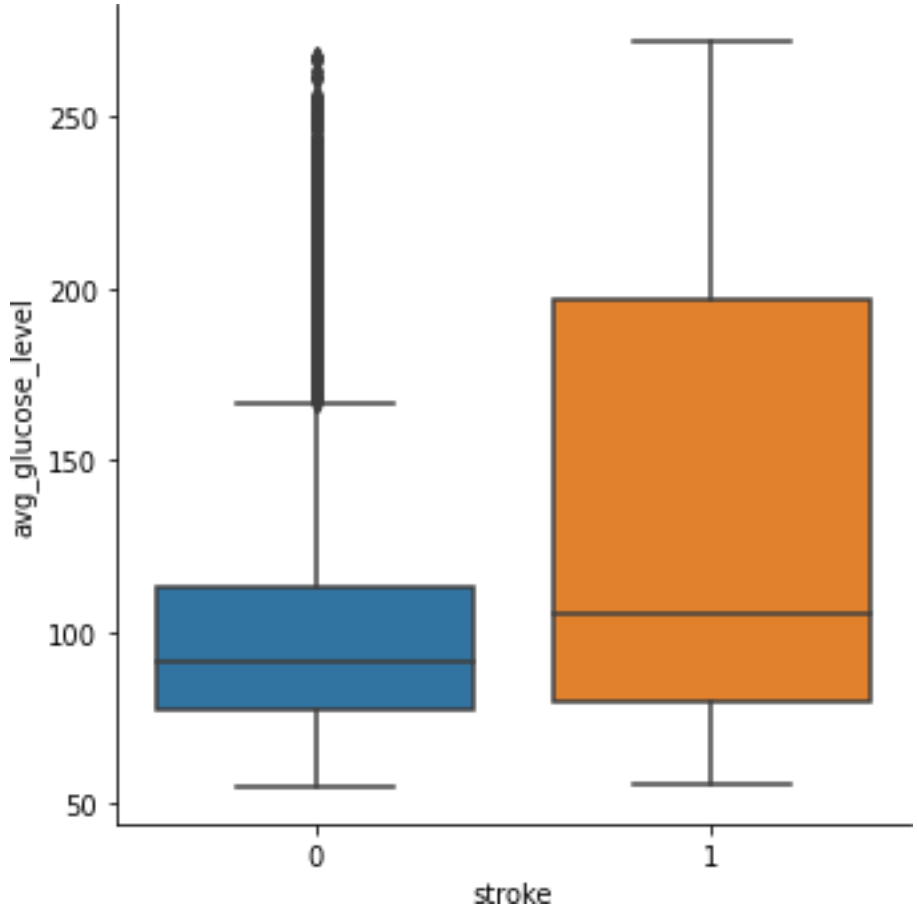
```
sns.catplot(x="Residence_type", y="stroke", hue="work_type", kind="bar",
data=data)
```

```
<seaborn.axisgrid.FacetGrid at 0x1899cbf3d00>
```




```
sns.catplot(x='stroke', y="avg_glucose_level", kind="box", data=data)
```

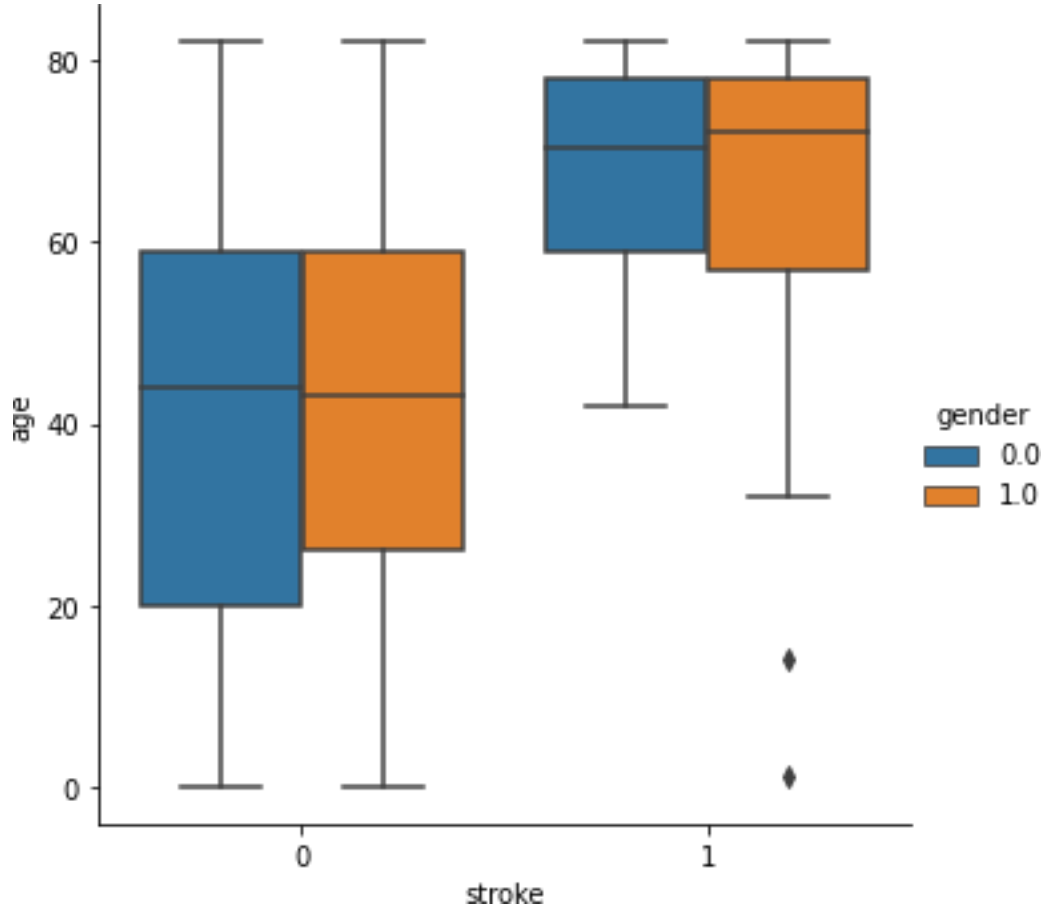
```
<seaborn.axisgrid.FacetGrid at 0x1899cc7af40>
```



People having higher glucose level are at a higher risk of stroke

```
sns.catplot(x='stroke', y="age", hue = 'gender', kind="box", data=data)
```

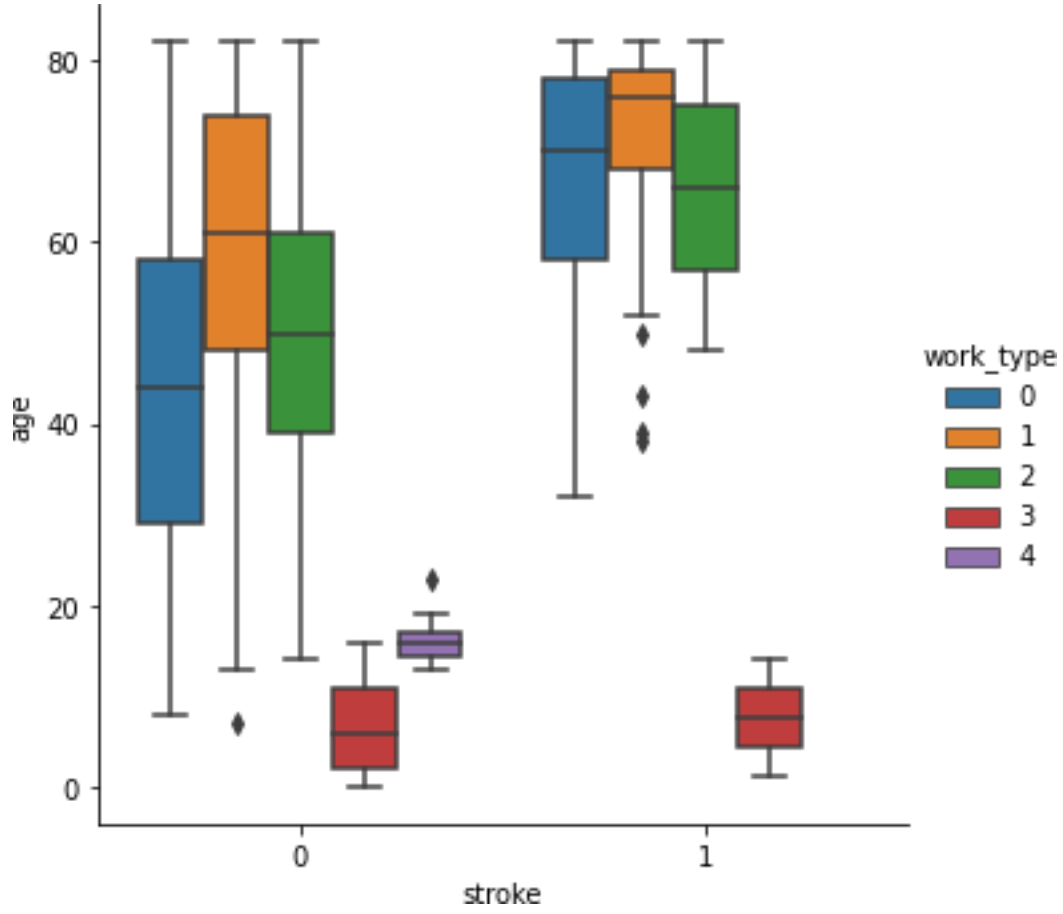
```
<seaborn.axisgrid.FacetGrid at 0x1899e260700>
```



Irrespective of gender, elderly men and women are prone to strokes

```
sns.catplot(x='stroke', y="age", hue = 'work_type', kind="box", data=data)
```

```
<seaborn.axisgrid.FacetGrid at 0x1899cd7ffa0>
```



4.2. Applying Machine Learning models

Applying Machine learning algorithms for prediction

Dividing the dataset into features and labels

```
features = ['id','age',
 'hypertension',
 'heart_disease',
 'ever_married',
 'Residence_type',
 'avg_glucose_level',
 'bmi',
 'gender',
 'work_type',
 'smoking_status']
```

```
label = ['stroke']

X = data[features]
y = data[label]
```

Check for null values in dataset again

```
X.isnull().sum()
```

```
id                   0
age                  0
hypertension         0
heart_disease        0
ever_married         0
Residence_type       0
avg_glucose_level    0
bmi                  0
gender               1
work_type            0
smoking_status       0
dtype: int64
```

There is 1 null value in Gender column

```
X.gender=(X.gender.fillna(1))
```

```
X.isnull().sum()
```

```
id                   0
age                  0
hypertension         0
heart_disease        0
ever_married         0
Residence_type       0
avg_glucose_level    0
bmi                  0
gender               0
work_type            0
smoking_status       0
dtype: int64
```

Treating Imbalance class using SMOTE

Since the target class is highly imbalanced, we need to treat it for better performance of the model. So we used SMOTE technique.

```
from imblearn.over_sampling import SMOTE

smote = SMOTE()
x_smote, y_smote = smote.fit_resample(X, y)
```

Splitting of dataset into train and test

```
from sklearn.model_selection import train_test_split
X_train,X_test,
y_train,y_test=train_test_split(x_smote,y_smote,test_size=0.33,random_state
=42)
X_valid, X_test, y_valid, y_test = train_test_split(X_test, y_test,
test_size=0.5, random_state=42)


testing = X_test['id'] #taking ID column for the purpose of submission
testing
```

```
315     41940
534     71808
1427     5984
9125    55982
6025    40676
        ...
3065    46455
8270    12021
6574    51788
2487    44325
620     34558
Name: id, Length: 1605, dtype: int64
```

Since ID column does not affect the model's performance, we drop it

```
X_train = X_train.drop(columns=['id'])
X_test = X_test.drop(columns=['id'])
```

Standardization of the data:
since data is in different scales

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()

X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Data points after standardization

```
X_train
```

```
array([[-0.04548995, -0.30279733, -0.21509314, ...,  0.9377214 ,
        -0.6292243 ,  0.71763036],
       [ 0.52403515, -0.30279733, -0.21509314, ...,  0.9377214 
         0.45781492, -1.20421443],
       [ 0.85028049, -0.30279733, -0.21509314, ..., -1.2697525 ,
        -0.6292243 , -1.20421443],
       ...,
       [ 1.00242686, -0.30279733, -0.21509314, ...,  0.88000305,
        -0.6292243 , -1.20421443],
       [-0.41759707, -0.30279733, -0.21509314, ...,  0.9377214 ,
        -0.6292243 ,  0.71763036],
       [ 0.21819076,  3.30253905, -0.21509314, ...,  0.09127432,
         0.45781492, -0.24329204]])
```

```
X_test
```

```
array([[ 0.09405022, -0.30279733,  4.64914867, ..., -1.2697525 ,
        -0.6292243 , -1.20421443],
       [-1.6269452 , -0.30279733, -0.21509314, ...,  0.9377214 ,
        -0.6292243 ,  1.67855275],
       [-1.39437825, -0.30279733, -0.21509314, ..., -1.2697525 ,
        -0.6292243 ,  0.71763036],
       ...,
       [ 0.24569974, -0.30279733, -0.21509314, ...,  0.529527  ,
        -0.6292243 , -1.20421443],
       [ 1.07083141, -0.30279733, -0.21509314, ..., -1.2697525 ,
         0.45781492,  0.71763036],
       [-1.02227114, -0.30279733, -0.21509314, ..., -1.2697525 ,
        -0.6292243 , -0.24329204]])
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression()
log_reg.fit(X_train,y_train)
```

```
LogisticRegression()
```

```
y_pred_log_reg = log_reg.predict(X_test)
y_pred_log_reg
```

```
array([0, 0, 0, ..., 1, 1, 0], dtype=int64)
```

Classification report of Logistic Regression

```
from sklearn.metrics import f1_score,
roc_auc_score,accuracy_score,confusion_matrix, precision_recall_curve, auc,
roc_curve, recall_score, classification_report
classification_report = classification_report(y_test, y_pred_log_reg)
print(classification_report)
```

```
              precision    recall  f1-score   support

           0       0.84      0.79      0.81       803
           1       0.80      0.85      0.82       802

    accuracy                           0.82      1605
   macro avg       0.82      0.82      0.82      1605
weighted avg       0.82      0.82      0.82      1605
```

```
auc = roc_auc_score(y_test, y_pred_log_reg)
auc
```

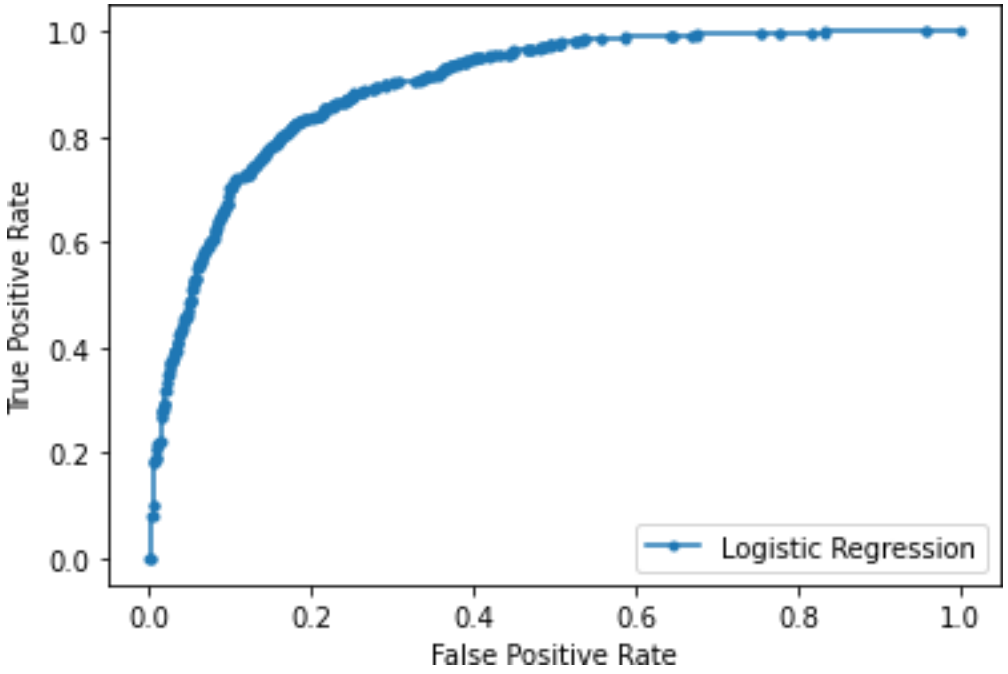
```
0.8187120927444776
```

```
cm = confusion_matrix(y_test, y_pred_log_reg)
cm
```

```
array([[631, 172],
       [119, 683]], dtype=int64)
```

```
predicted_probab_log = log_reg.predict_proba(X_test)
predicted_probab_log = predicted_probab_log[:, 1]
fpr, tpr, _ = roc_curve(y_test, predicted_probab_log)
```

```
from matplotlib import pyplot
pyplot.plot(fpr, tpr, marker='.', label='Logistic Regression')
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')
pyplot.legend()
pyplot.show()
```



Random Forest

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)
```

```
RandomForestClassifier()
```

```
rfc_predict = rfc.predict(X_test)
roc_auc_score(y_test, rfc_predict)
```

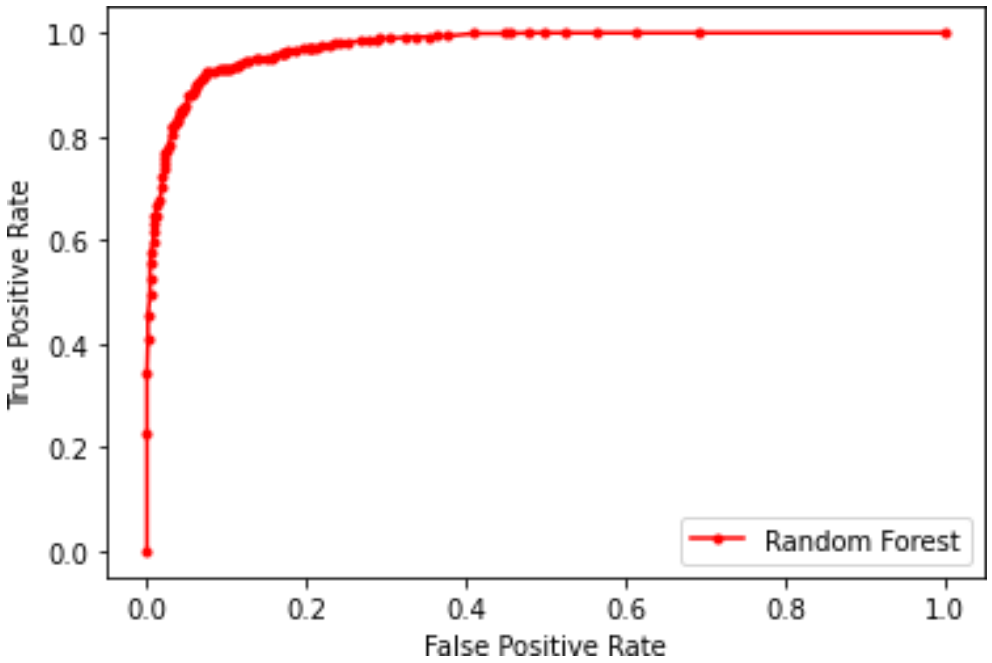
```
0.9208749607922907
```

```
cm = confusion_matrix(y_test, rfc_predict)
cm
```

```
array([[736,  67],
       [ 60, 742]], dtype=int64)
```

```
tn = cm[0,0]
fp = cm[0,1]
```

```
tp = cm[1,1]
fn = cm[1,0]
accuracy = (tp + tn) / (tp + fp + tn + fn)
precision = tp / (tp + fp)
recall    = tp / (tp + fn)
f1score = 2 * precision * recall / (precision + recall)
print(f1score)
0.9211669770328988


predicted_probab = rfc.predict_proba(X_test)


predicted_probab = predicted_probab[:, 1]


fpr, tpr, _ = roc_curve(y_test, predicted_probab)


from matplotlib import pyplot
pyplot.plot(fpr, tpr, marker='.', color='red', label='Random Forest')
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')
pyplot.legend()
pyplot.show()
```



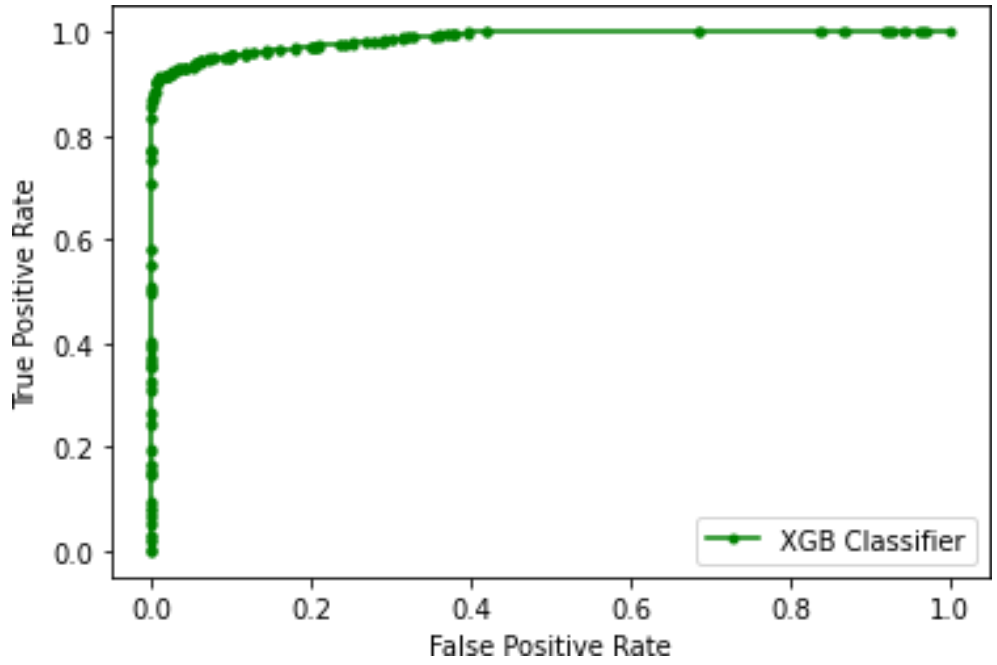
XGBoost Classifier

```
import xgboost as xgb


model = xgb.XGBClassifier()
model.fit(X_train,y_train)


XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise'
,
              importance_type=None, interaction_constraints='',
              learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight
=1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, predictor='auto', random_state
=0,
              reg_alpha=0, reg_lambda=1, ...)


y_pred1 = model.predict(X_test)


roc_auc_score(y_test, y_pred1)


0.9476495560600368


cm = confusion_matrix(y_test, y_pred1)
cm


array([[779,  24],
       [ 60, 742]], dtype=int64)


tn = cm[0,0]
fp = cm[0,1]
tp = cm[1,1]
fn = cm[1,0]
accuracy = (tp + tn) / (tp + fp + tn + fn)
precision = tp / (tp + fp)
recall    = tp / (tp + fn)
f1score = 2 * precision * recall / (precision + recall)
print(f1score)
```

```
0.9464285714285715
```

```
predicted_probab = model.predict_proba(X_test)
predicted_probab = predicted_probab[:, 1]
fpr, tpr, _ = roc_curve(y_test, predicted_probab)
```

```
from matplotlib import pyplot
pyplot.plot(fpr, tpr, marker='.', color='green',label='XGB Classifier')
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')
pyplot.legend()
pyplot.sho
```



Chapter-5

Conclusion and Future Work

5.1. Scope of Improvement

This project helps to predict the stroke risk using prediction model in older people and for people who are addicted to the risk factors as mentioned in the project. In future, the same project can be extended to give the stroke percentage using the output of current project. This project can also be used to find the stroke probabilities in young people and underage people by collecting respective risk factor information's and doctors consulting.

5.2. Summary

1. Loaded few modules and initial cleaning of data
2. Visualization of few relationships between variables and individual variables.
3. Feature correlation visualization
4. Balancing of data and visualization
5. Implemented few models and visualized the results.

S.No	ML Model	F1 Score
1	Logistic Regression	81
2	Random Forest	92
3	XGBoost	94

Table No 5.1: Summary Table

5.3. Conclusion

Stroke is a life-threatening medical illness that should be treated as soon as possible to avoid further complications. The development of an ML model could aid in the early detection of stroke and the subsequent mitigation of its severe consequences. The effectiveness of several ML algorithms in properly predicting stroke based on a number of physiological variables is investigated in this study. Logistic Regression and Random Forest gives an f1 score of 81 and 92 percent respectively whereas XGBoost outperforms the other methods with a classification accuracy of 94 percent. The future scope of this studyis that using a larger dataset and machine learning models, such as AdaBoost, SVM, and Bagging, the framework models may be enhanced. This will enhance the dependability of the framework and the framework's presentation. In exchange for just providing some basicinformation, the machine learning architecture may help the general public in determiningthe likelihood of a stroke occurring in an adult patient. In an ideal world, it would help patients obtain early treatment for strokes and rebuild their lives after the event.