A Mini- Project Report

on

"Brain Stroke Prediction by Machine Learning"

Submitted to the

Pune Institute of Computer Technology, Pune

In partial fulfilment for the award of the Degree of

Bachelor of Engineering

in

Information Technology

by

Deepak Abande	**33101**	**T190058502**
Sumit Chavan	**33113**	**T190058534**
Sakshi Dalvi	**33117**	**T190058544**
Divya Dhomase	**33123**	**T190058560**

Under the guidance of

Prof. Deepali Londhe

Department Of Information Technology

Pune Institute of Computer Technology College of Engineering

Sr. No 27, Pune-Satara Road, Dhankawadi, Pune - 411 043.

2021-2022

CERTIFICATE

This is to certify that the project report entitled

BRAIN STROKE PREDICTION

Submitted by

Deepak Abande	**T190058502**
Sumit Chavan	**T190058534**
Sakshi Dalvi	**T190058544**
Divya Dhomase	**T190058560**

is a bonafide work carried out by them under the supervision of Prof. Deepali Londhe Ma'am and it is approved for the partial fulfilment of the requirement of Software Laboratory Course-2015 for the award of the Degree of Bachelor of Engineering (Information Technology)

Deepali Londhe	**Dr. Archana S. Ghotkar**
Internal Guide	Head of Department
Department of Information Technology	Department of Information Technology

Place: Pune
Date:

CONTENTS

INDEX

Abstract

Tearing of the blood vessels present in the brain leads to serious health condition named as brain stroke. It can also occur when there is distraction in blood flow and other important nutrients to the brain parts. According to the report presented by World Health Organisation (WHO), stroke is the main cause of death and disability in the world.
For heart stroke prediction, various work has been carried out but in order to predict brain stroke, very few work has been carried out. Various machine learning techniques and models are designed to detect the probability of stroke occurrence in the brain.
This seminar work has used machine learning algorithms like Logistic Regression, Decision Tree Classification, Support Vector Machine and has taken various physiological factors to train five different models for accurate prediction.

Keywords:
Stroke, machine learning, decision tree classification, support vector machine, Logistic regression.

ACKNOWLEDGEMENT

I would like to express my deep sense of gratitude to respected Prof. Deepali Londhe ma'am as my subject teacher for her constant support and valuable guidance from selection of mini project topic to the completion of the report. She has played a crucial role in completion of the mini project report. I also would like to express my sincere thanks to Head of the Department Dr. Archana Ghotkar and Principal Dr. R. Shreemati for their encouragement and support. I am thankful to all the staff members of our department for their kind help and cooperation. And finally, I would like to extend my thankful gratitude to my colleagues for providing moral support and valuable suggestions.

CHAPTER 1
INTRODUCTION

Introduction:

Stroke is the second leading cause of death in the world and it will remain as an important health burden for the individuals and for national healthcare systems. Potentially most identifiable risk factors for stroke occurrence include hypertension, cardiac disease, diabetes,atrial fibrillation, and also various lifestyle factors, etc.

A dataset used in the project work is referred from Kaggle with various physiological traits asits attributes to proceed with this task. These traits are then analyzed and used for the final detection/prediction. First of all, the dataset is cleaned and made ready for the machine learning model to understand which is the process of data- preprocessing. For this purpose,the dataset is initially checked for null values and fill them with not null values. Then to convert string values into integers Label encoding is performed followed by one-hot encoding.

After the step of Data Preprocessing, the mentioned dataset is used to split into two different parts i.e. train data and test data. Using this new data, a model is then built with the help of various Classification Algorithms. Using methods like confusion matrix, accuracy is calculated for all the algorithms and compared to get the best-trained model for prediction purpose. After proper analysis, the project work concludes which algorithm is most appropriate for the prediction of brain stroke.

Purpose

The purpose of this project is to create good prediction model for detection of stroke so that itcan be beneficial for doctors also and for patients also to decrease the death rate due to stroke. The main purpose of seminar is to apply principles of machine learning over large pre- existing dataset to effectively predict the brain stroke based on potentially modifiable risk factors.

Background and Motivation

According to the CDC (Centre's for Disease Control and Prevention), stroke is the fifth- leading cause of death globally. Stroke is a non-communicable infection that is responsible for around 11% of total deaths in the world. Over 795,000 individuals in the United States have to suffer from the ill effects of a brain stroke. It is the fourth most significant reason for increasing death rate in India. According to WHO, Stroke will continue to enhance mortalityrate in the upcoming years.

More than 70% of strokes are first events, hence making primary stroke prevention is particularly an important aspect. So, if we predict its occurrence then it will be very beneficial for people and machine learning model can be used anywhere for predicting strokes.

CHAPTER 2
LITERATURE SURVEY OF STROKE PREDICTION USINGMACHINE LEARNING

In order to get the required knowledge and information about various concepts and algorithms related to the present analysis of stroke, existing literature were studied. Some of the important conclusions were made from these surveys. Some of them are listed below.

1. **"Prediction of Brain Stroke Severity Using Machine Learning. In: International Information and Engineering Technology Association (2020)**"- Vamsi Bandi, Debnath Bhattacharyya, Divya Midhun Chakravarthy

The authors of this paper have performed the work of brain stroke prediction by using random forest algorithm which helped to analyze the levels of risks obtained from the strokes. Authors suggested that , this method will give better performance when compared to the existing algorithms. This research is limited to very less types of strokes and cannot be used for any new stroke type in the upcoming future.

2. **"Predicting stroke from electronic health records. In: 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society IEEE (2019)"-** Nwosu, C.S., Dev, S., Bhardwaj, P., Veera Valli, B., John, D.

This paper shows that the model was trained using techniques like Decision Tree, Random Forest, and Multi-layer perceptron for prediction of stroke. With the slight differences, the obtained accuracies for the three methods were quite close. For Decision Tree, calculated accuracy was 74.31%, for Random Forest it was 74.53%, and for Multi-layer perceptron was 75.02%. According to this paper, Multi- layer perceptron is more accurate than the other two methods. Accuracy score was the only metric used for calculating the performance which might not always give favorable and accurate results.

3. **" Prediction of Stroke using Data Mining Classification Techniques: International Journal of Advanced Computer Science and Applications (IJACSA) (2018)"-** Ohoud Almadani, Riyad Alshammari

In this paper, the authors have used different data mining classification algorithms and techniques for prediction of the possibility of a stroke. The dataset in the paper was taken from the Ministry of National Guards Health Affairs Hospitals, Kingdom of Saudi Arabia. The three classification algorithms used were JRIP, C.5, and Multi layers perceptron. With the help of these algorithms, the model obtained an accuracy of around 95%. Though the paper claims to obtain an accuracy of 95%, the time taken for training and predicting is much high as the authors have used complex algorithms for implementation purpose.

CHAPTER 3
Proposed system methodology

Attribute Name	Type *(Values)*	Description
1. id	Integer	A unique integer value for patients
2. gender	String literal *(Male, Female, Other)*	Tells the gender of the patient
3. age	Integer	Age of the Patient
4. hypertension	Integer *(1, 0)*	Tells whether the patient has hypertension or not
5. heart_disease	Integer *(1, 0)*	Tells whether the patient has heart disease or not
6. ever_married	String literal *(Yes, No)*	It tells whether the patient is married or not
7. work_type	String literal *(children, Govt_job, Never_worked, Private, Self-employed)*	It gives different categories for work
8. Residence_type	String literal *(Urban, Rural)*	The patient's residence type is stored
9. avg_glucose_level	Floating point number	Gives the value of average glucose level in blood
10. bmi	Floating point number	Gives the value of the patient's Body Mass Index
11. smoking_status	String literal *(formerly smoked, never smoked, smokes, unknown)*	It gives the smoking status of the patient
12. stroke	Integer *(1, 0)*	Output column that gives the stroke status

Table 1: Stroke dataset from kaggle

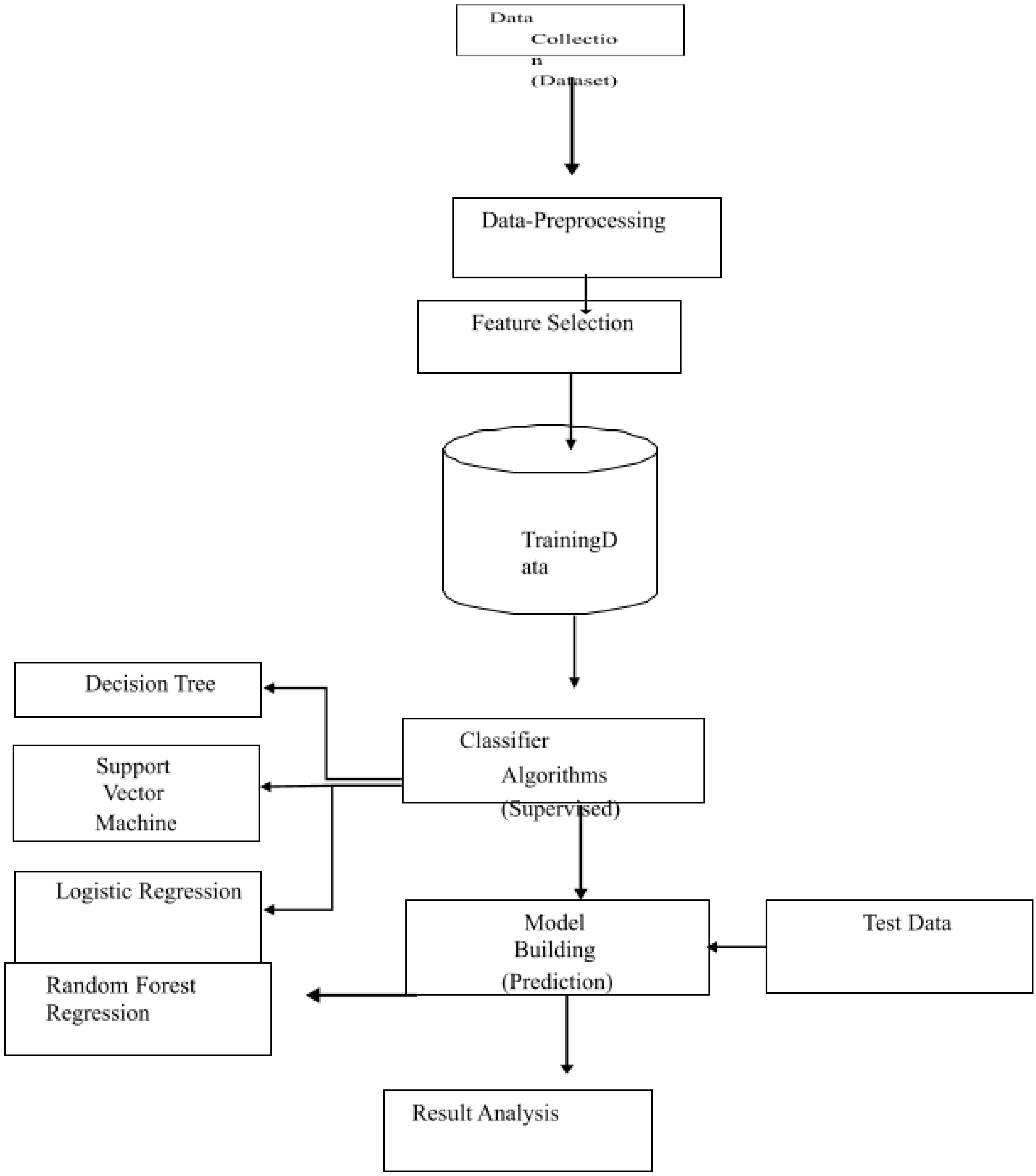


Figure 2: Proposed System flow diagram

ALGORITHMS USED IN THE STROKE PREDICTION

Logistic regression:
It is a supervised learning algorithm which is used for predicting the probability of the output variable. When the output variable has binary values then this algorithm is the best fit. A performing this algorithm on the chosen dataset, 78% accuracy was obtained. Using other metrics like precision score and recall score, we can enhance efficiency of the algorithm. In this case, precision score obtained was about 0.15% and recall score was about 0.72% and The F1 Score obtained by this algorithm is 0.25%.

Support vector machine:
SVM is a technique that can be combined with learning algorithms for analysing the data for regression and classification. It also scales well to high dimensional data. Particularly for this dataset, accuracy obtained is 78.50% by SVM while precision score is 0.13% and recall score is 0.76%. F1 score is 0.22%.

Decision tree classifier:
Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation. Particularly for this dataset, accuracy obtained is 89.72% by SVM while precision score is 0.16% and recall score is 0.22%. F1 score is 0.19%.

Random forest algorithm:
Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

IMPLEMENTATION PROCESS

1) Introduction

• Importing Libraries

```
import seaborn as sns
import matplotlib.pyplot as plt import warnings
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression from
sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,roc
_auc_score,ConfusionMatrixDisplay,precision_score,recall_score,
f1_score,classification_report,roc_curve,plot_roc_curve,auc,pre
cision_recall_curve,plot_precision_recall_curve,average_precisi
on_score
from sklearn.model_selection import cross_val_score from
sklearn.model_selection import train_test_split

from imblearn.over_sampling import SMOTE
from sklearn.compose import ColumnTransformer from
sklearn.preprocessing import OneHotEncoder from
sklearn.preprocessing import LabelEncoder from
sklearn.model_selection import GridSearchCV
```

• Importing Dataset

	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

```
df=pd.read_csv('/content/healthcare.csv') df.head()

df.tail()
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
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

```
[ ]  df.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                4909 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
```

- **Missing Values**

```
[ ]  df.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
```

```
[ ]  #We fill the missing values in the Body Mass Index variable with the average value.
     df.bmi.replace(to_replace=np.nan, value=df.bmi.mean(),inplace=True)
     df.head()
```

	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	36.600000	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	28.893237	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.500000	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.400000	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.000000	never smoked	1

```
[ ] df.describe()
```

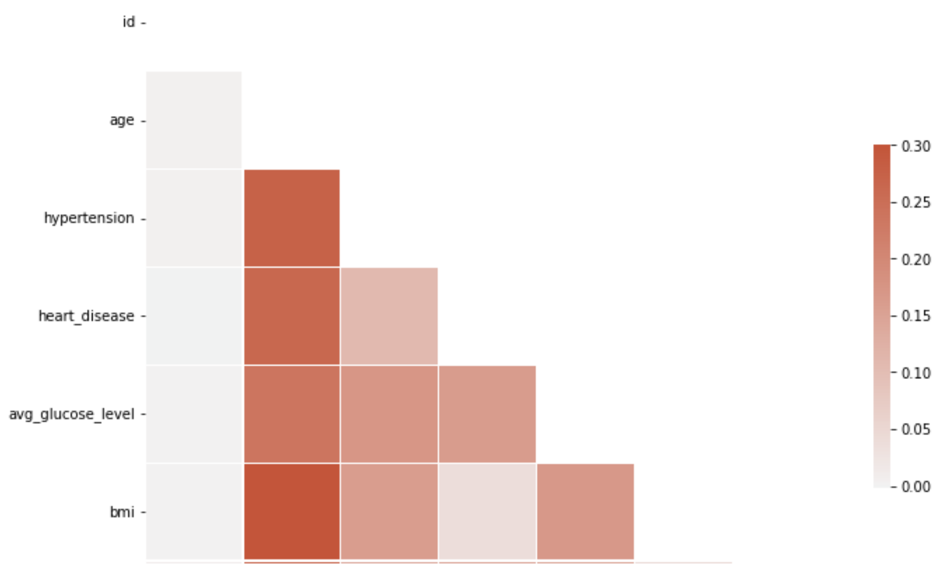
	id	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000
mean	36517.829354	43.226614	0.097456	0.054012	106.147677	28.893237	0.048728
std	21161.721625	22.612647	0.296607	0.226063	45.283560	7.698018	0.215320
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

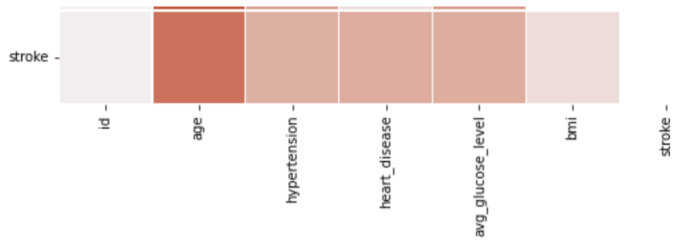
2) Data Visualization

- **Corr Heat Map**

```
# compute the corr matrixcorr = df.corr()
# generate a mask for the upper triangle mask = np.triu(np.ones_like(corr,dtype=bool))

# set up the matplotlib figure
f, ax = plt.subplots(figsize=(11,9))

# generate a custom diverging colormap
cmap = sns.diverging_palette(230,20,as_cmap=True)

#draw the heatpmap with the mask and correct aspect ratio
sns.heatmap(corr,mask=mask,cmap=cmap,vmax=.3,center=0,square=Tr
ue,linewidths=.5,cbar_kws={'shrink':.5})
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd4c9d02390>
```

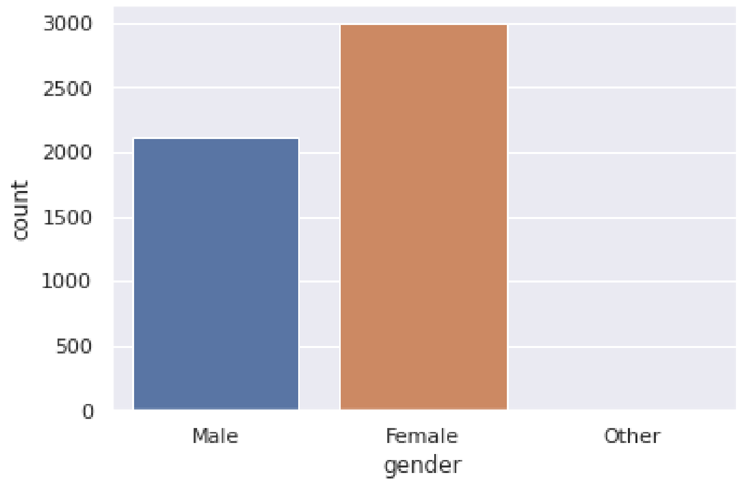




- **Count Plot**

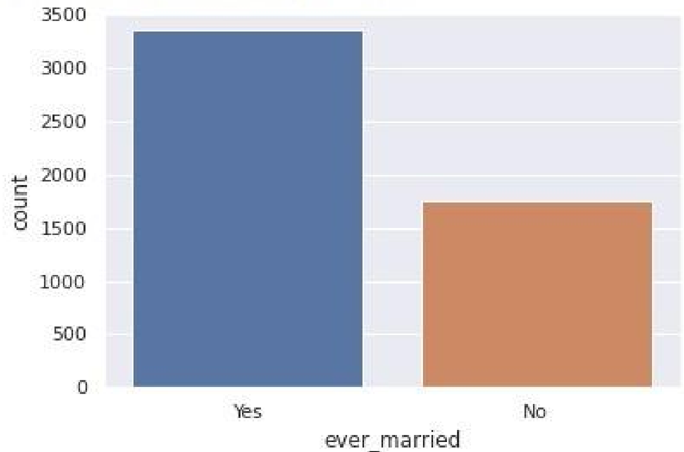
```
[ ]  print(df.gender.value_counts())
     sns.set_theme(style='darkgrid')
     ax = sns.countplot(data=df,x='gender')
```

```
Female    2994
Male      2115
Other        1
Name: gender, dtype: int64
```



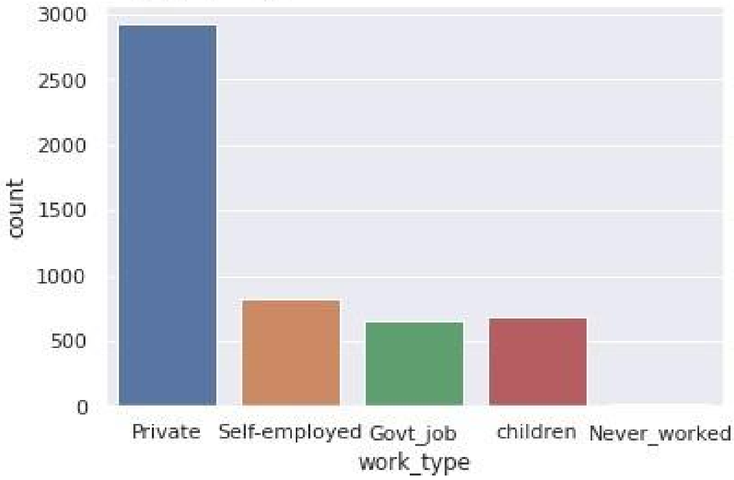

```
[ ]  print(df.ever_married.value_counts())
     sns.set_theme(style='darkgrid')
     ax = sns.countplot(data=df, x='ever_married')
```

```
Yes    3353
No     1757
Name: ever_married, dtype: int64
```



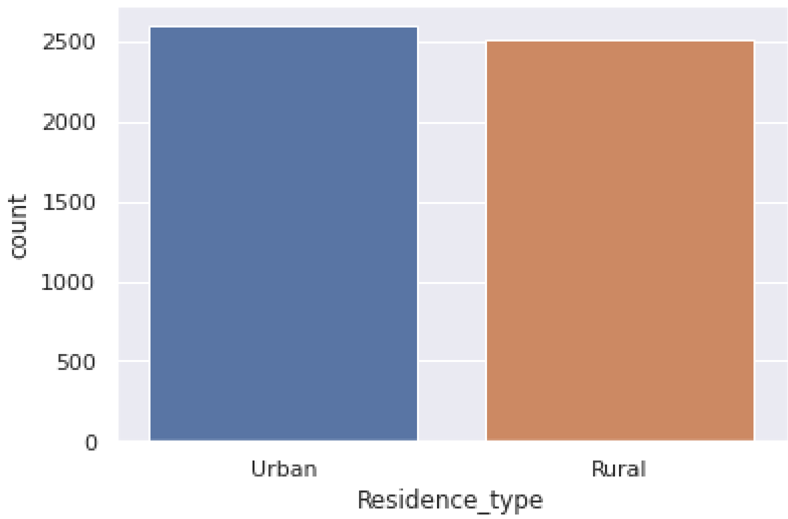
```
[ ] print(df.work_type.value_counts())
    sns.set_theme(style='darkgrid')
    ax = sns.countplot(data=df,x='work_type')
```

```
Private          2925
Self-employed     819
children          687
Govt_job          657
Never_worked       22
Name: work_type, dtype: int64
```



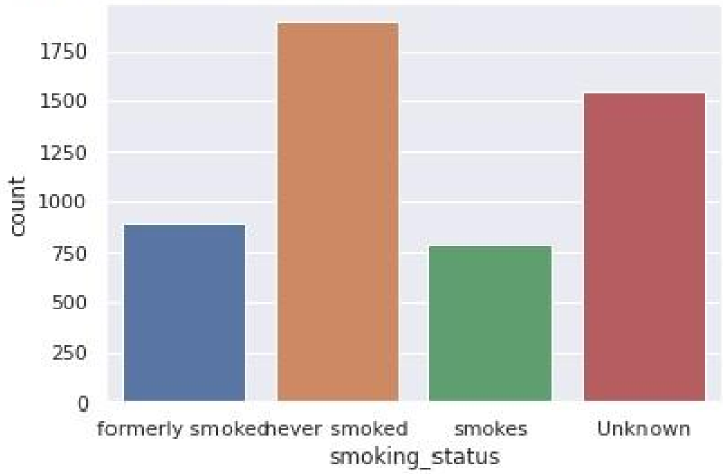

```
[ ] print(df.Residence_type.value_counts())
    sns.set_theme(style='darkgrid')
    ax = sns.countplot(data=df,x='Residence_type')
```

```
Urban    2596
Rural    2514
Name: Residence_type, dtype: int64
```



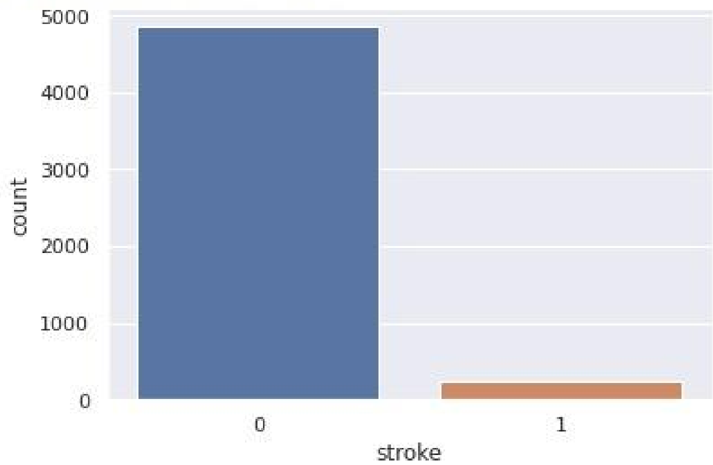
```
[ ]  print(df.smoking_status.value_counts())
     sns.set_theme(style='darkgrid')
     ax = sns.countplot(data=df,x='smoking_status')
```

```
never smoked       1892
Unknown            1544
formerly smoked     885
smokes              789
Name: smoking_status, dtype: int64
```




```
[ ]  print(df.stroke.value_counts())
     sns.set_theme(style='darkgrid')
     ax = sns.countplot(data=df,x='stroke')
```

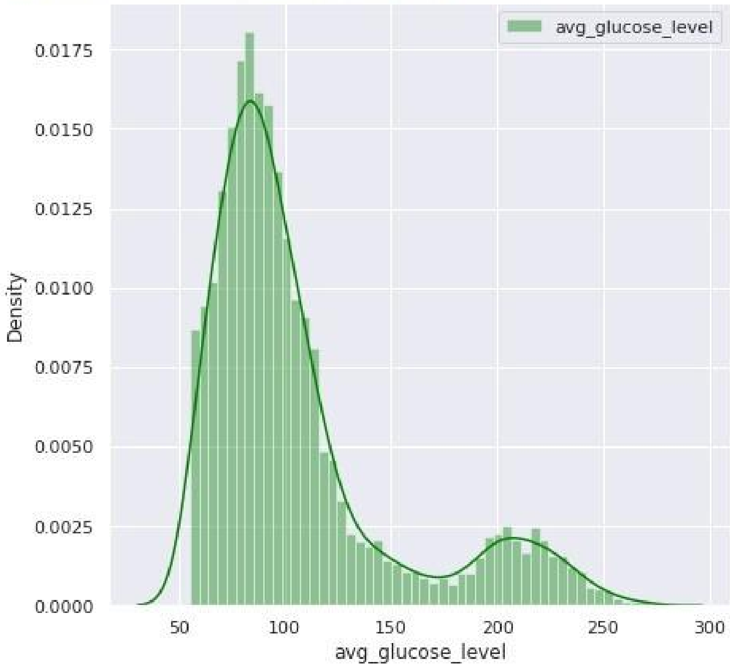
```
0    4861
1     249
Name: stroke, dtype: int64
```



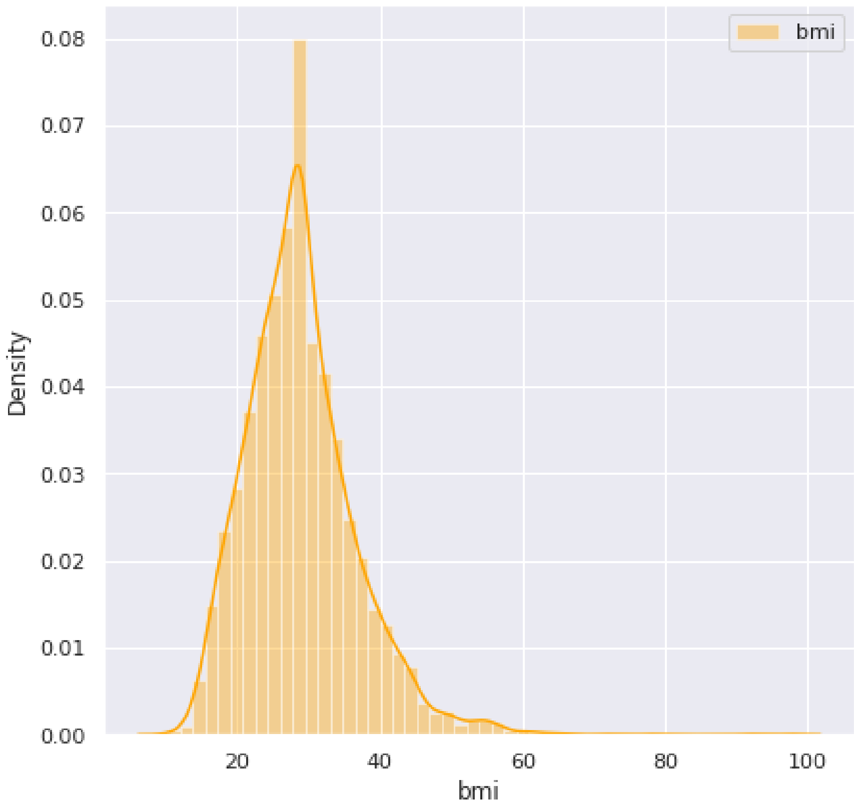
- **Distibution Plot**

```
fig = plt.figure(figsize=(7,7))
sns.distplot(df.avg_glucose_level,color='green',label='avg_glucose_level',kde=True)
plt.legend()
```

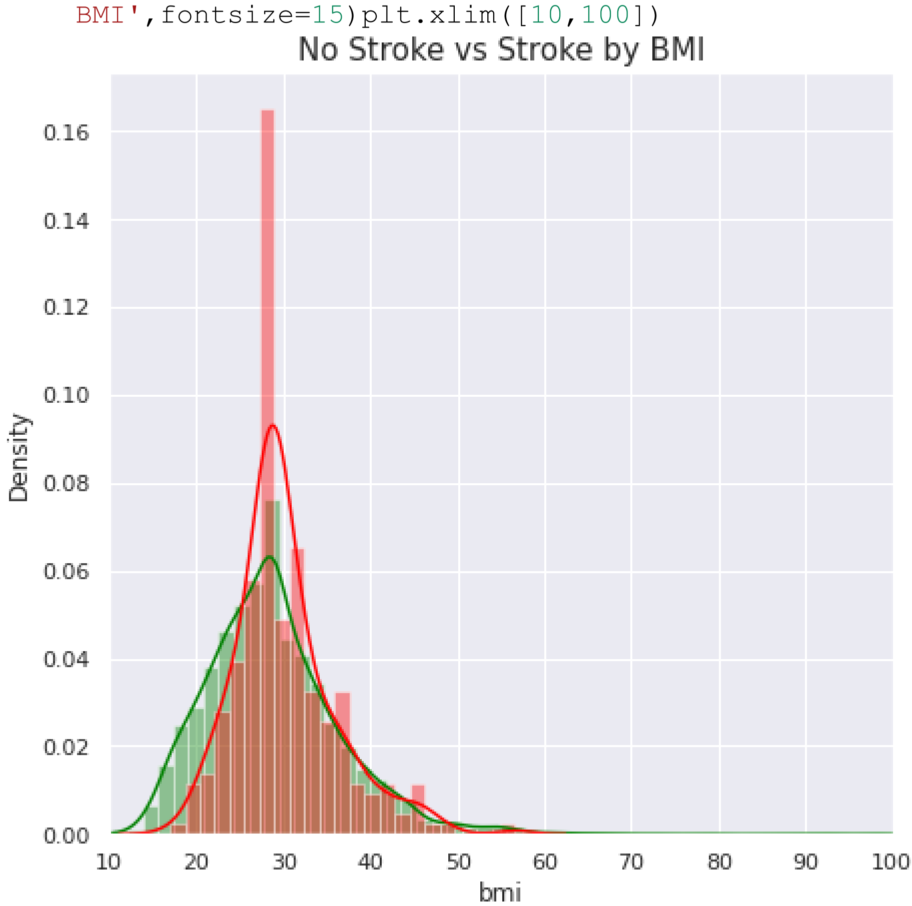
```
<matplotlib.legend.Legend at 0x7fd4bfb01e10>
```



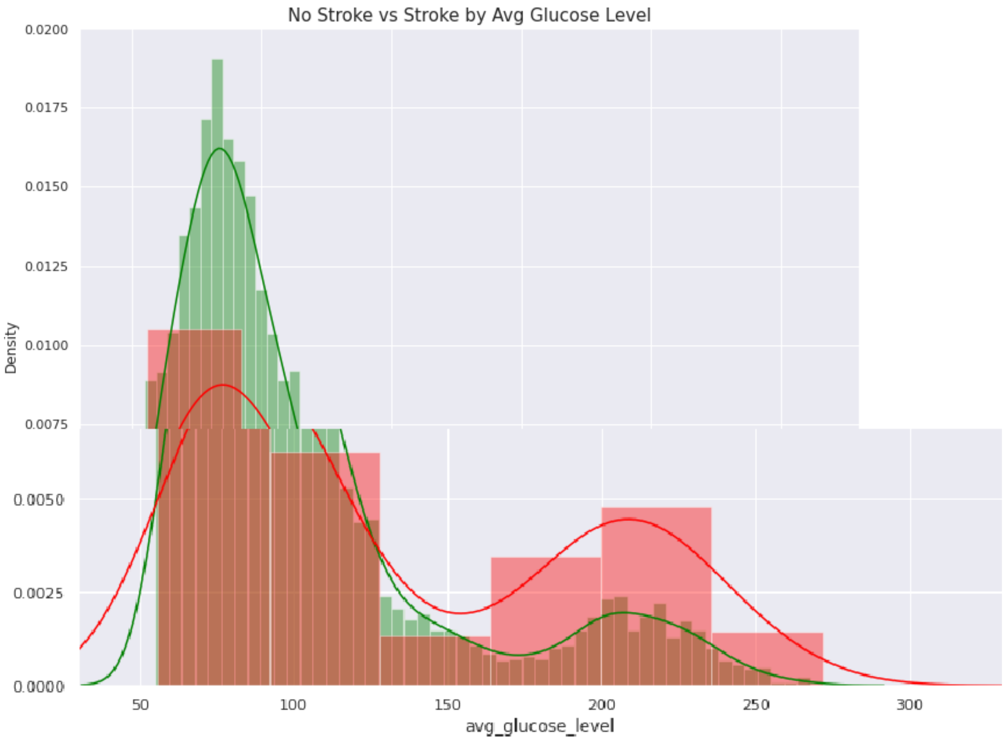

```
fig = plt.figure(figsize=(7,7))
sns.distplot(df.bmi,color='orange',label='bmi',kde=True)
plt.legend()
```



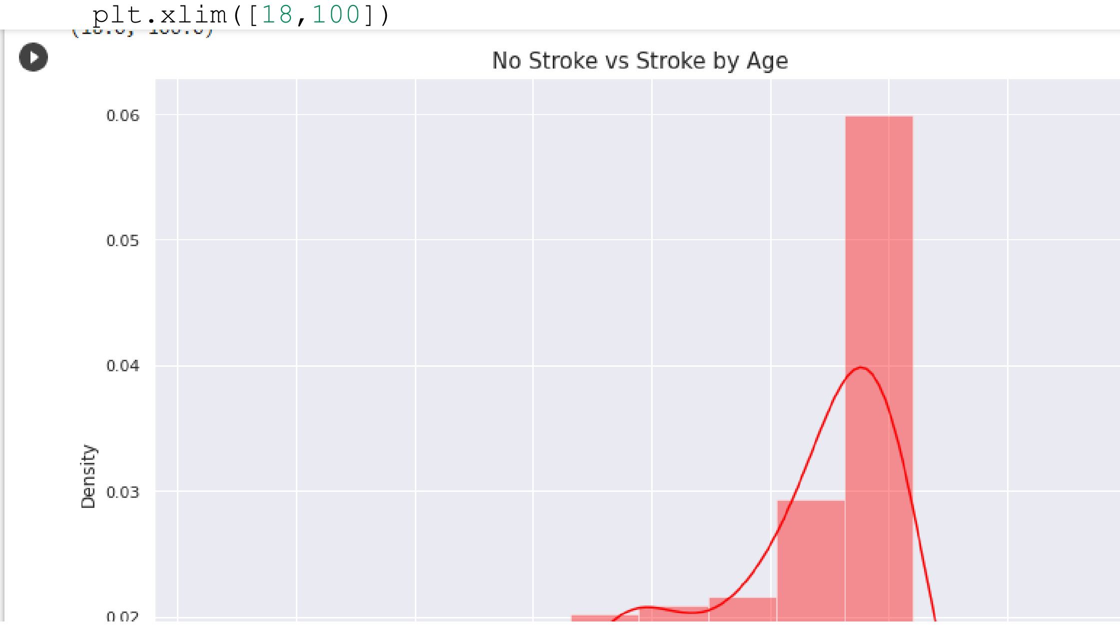
```
plt.figure(figsize=(7,7))
sns.distplot(df[df['stroke'] == 0]['bmi'],color='green')
sns.distplot(df[df['stroke'] == 1]['bmi'],color='red')

plt.title('No Stroke vs Stroke by
BMI',fontsize=15)plt.xlim([10,100])
```

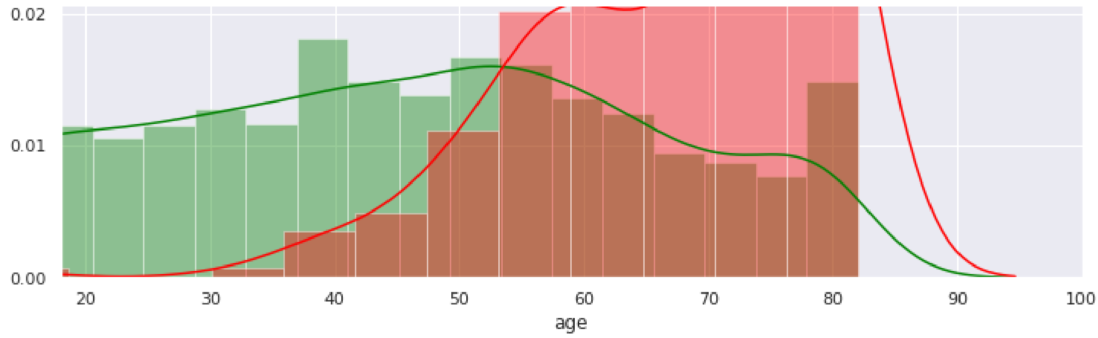



```
plt.figure(figsize=(12,10))
sns.distplot(df[df['stroke'] == 0]
['avg_glucose_level'],color='green')
sns.distplot(df[df['stroke'] == 1]
['avg_glucose_level'],color='red')
plt.title('No Stroke vs Stroke by Avg Glucose Level',fontsize=15)
plt.xlim([30,330])
```




```
plt.figure(figsize=(12,10))
sns.distplot(df[df['stroke'] == 0]['age'],color='green')
sns.distplot(df[df['stroke'] == 1]['age'],color='red')
plt.title('No Stroke vs Stroke by Age',fontsize=15)
plt.xlim([18,100])
```



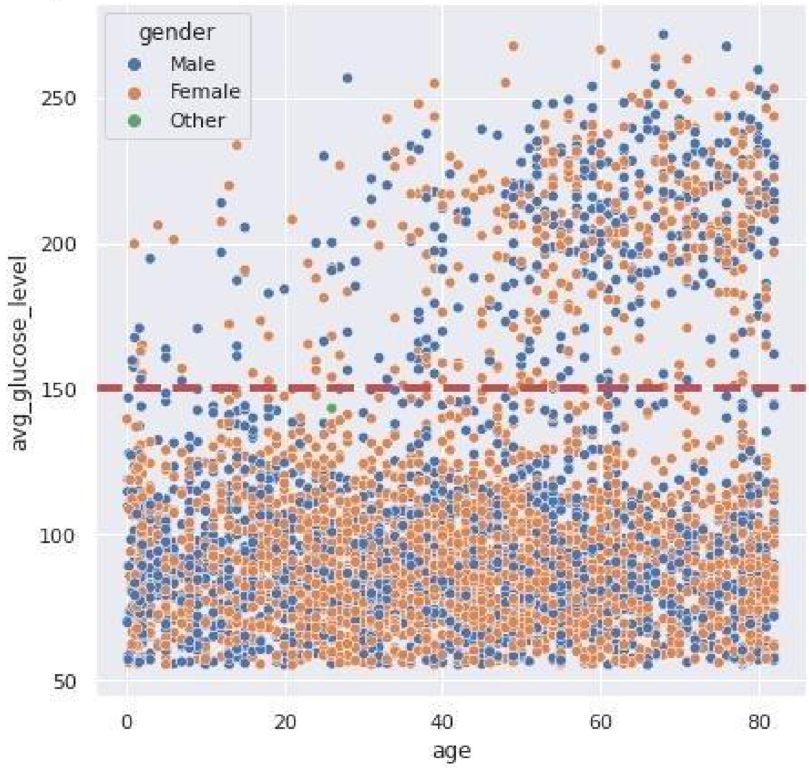


- **Scatter Plot**

```
fig = plt.figure(figsize=(7,7))
graph =
sns.scatterplot(data=df,x='age',y='bmi',hue='gender')
```

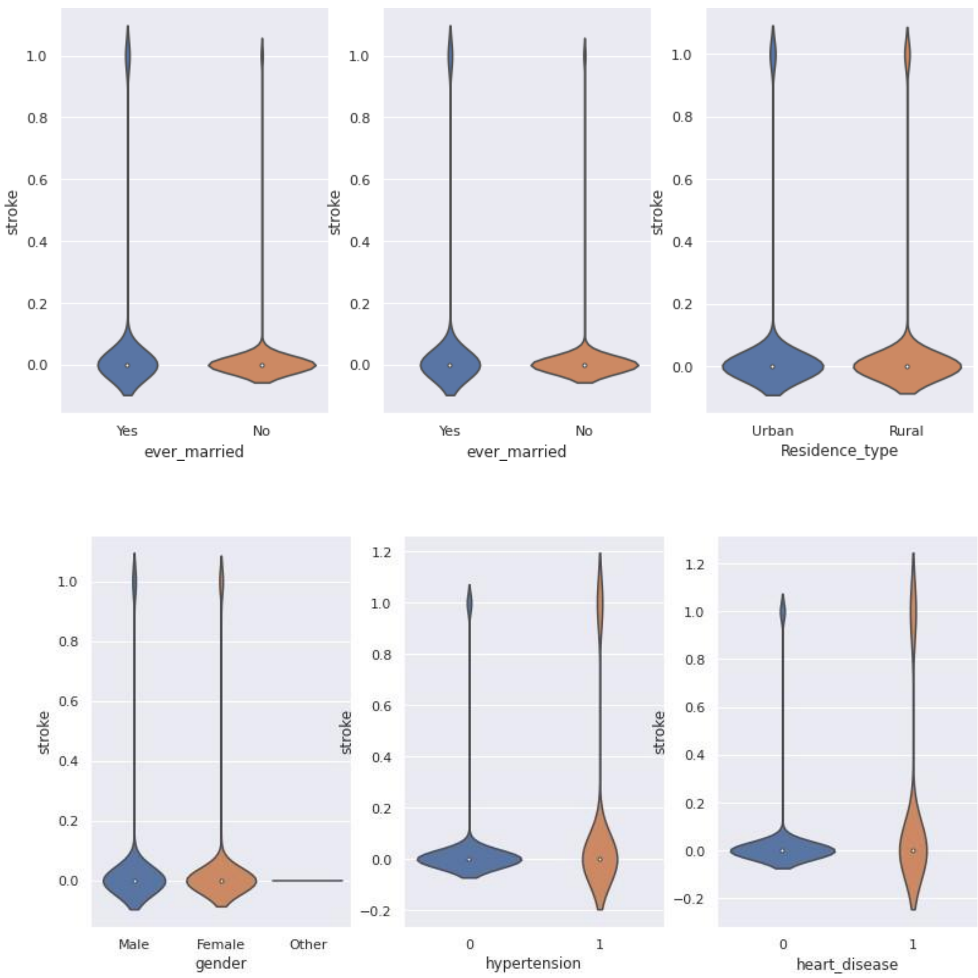
```
<matplotlib.lines.Line2D at 0x7fd4c9df7290>
```


```
fig = plt.figure(figsize=(7,7))
graph =
sns.scatterplot(data=df,x='age',y='avg_glucose_level',hue='g
ender')
graph.axhline(y=150,linewidth=4,color='r',linestyle='--')
```


gender
Male
Female
Other
avg_glucose_level
250
200
150
100
50
0
20
40
60
80
age

- **Violin Plot**

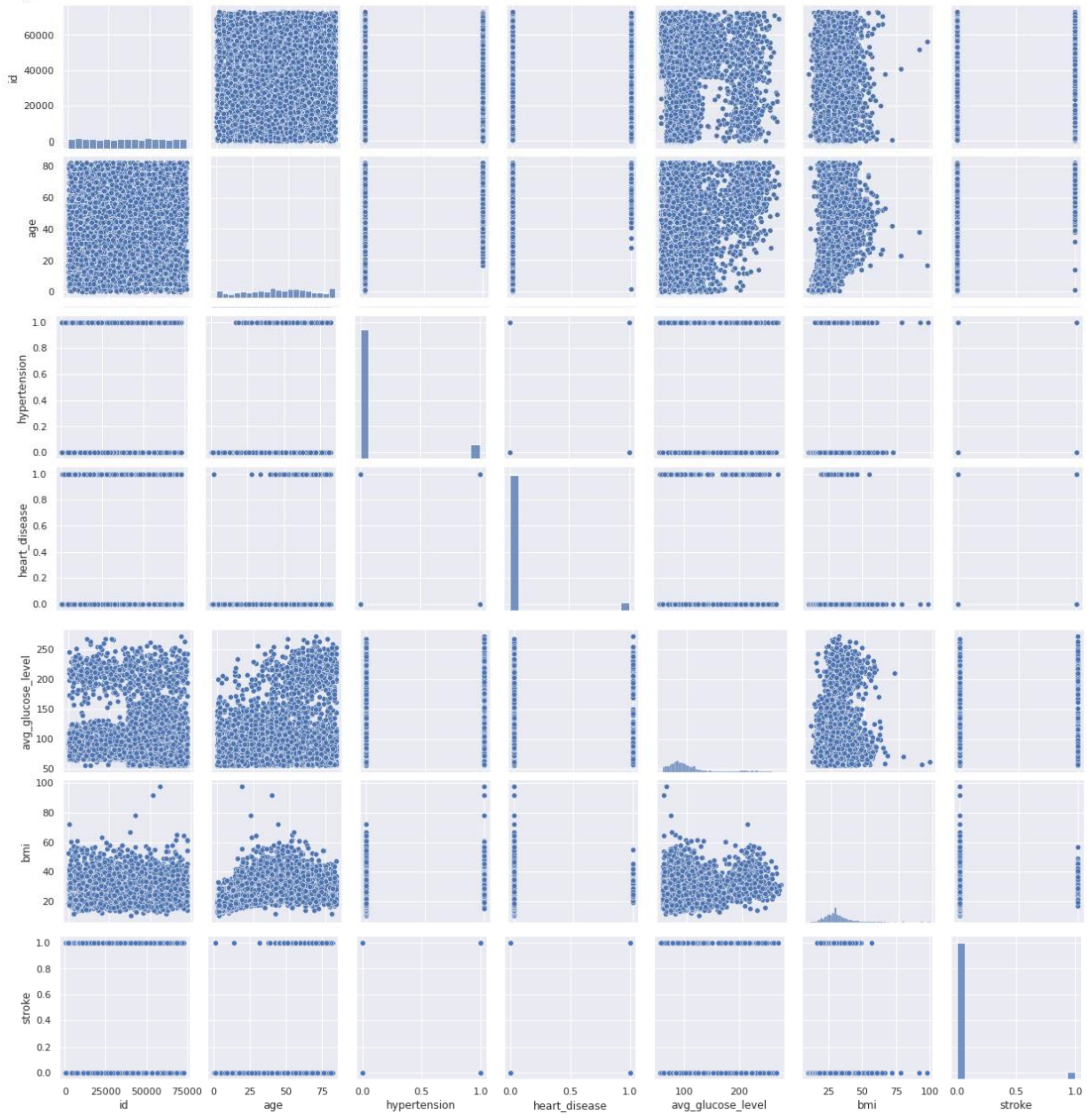
```
plt.figure(figsize=(13,13
))
sns.set_theme(style='dark
grid')plt.subplot(2,3,1)
sns.violinplot(x='gender',y='stroke',data=df)
plt.subplot(2,3,2)
sns.violinplot(x='hypertension',y='stroke',data=
df)plt.subplot(2,3,3)
sns.violinplot(x='heart_disease',y='stroke',dat
a=df) plt.subplot(2,3,4)
sns.violinplot(x='ever_married',y='stroke',data=
df)plt.subplot(2,3,5)
sns.violinplot(x='ever_married',y='stroke',data=
df)plt.subplot(2,3,6)
sns.violinplot(x='Residence_type',y='stroke',da
ta=df)plt.show()
```



- **Pair Plot**

```
fig =
plt.figure(figsize=(10,10)
)sns.pairplot(df)
plt.show()
```

<Figure size 720x720 with 0 Axes>



3) Data Preprocessing

Label Encoder

```
[ ] df.head()
```

	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	36.600000	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	28.893237	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.500000	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.400000	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.000000	never smoked	1

```
[ ] #Label Encoder
    le = LabelEncoder()
    df['gender'] = le.fit_transform(df['gender'])
    df['ever_married'] = le.fit_transform(df['ever_married'])
    df['work_type'] = le.fit_transform(df['work_type'])
    df['Residence_type'] = le.fit_transform(df['Residence_type'])
    df['smoking_status'] = le.fit_transform(df['smoking_status'])
```

X and Y Splitting

```
[ ] #X and Y Splitting
    x = df.iloc[:,1:-1].values
    y = df.iloc[:,-1].values

    print('X Shape', x.shape)
    print('Y Shape',y.shape)
```

```
X Shape (5110, 10)
Y Shape (5110,)
```

Column Transformator and OneHotEncoder

```
#Column Transformator and OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder',OneHotEncoder(),[0,5,9])],remainder='passthrough')
x = np.array(ct.fit_transform(x))
```

Train Test Split

```
[ ] #Train Test Split
    X_train,X_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=0)

    print('Number transations x_train df',X_train.shape)
    print('Number transations x_test df',X_test.shape)
    print('Number transations y_train df',y_train.shape)
    print('Number transations y_test df',y_test.shape)
```

```
Number transations x_train df (4088, 19)
Number transations x_test df (1022, 19)
Number transations y_train df (4088,)
Number transations y_test df (1022,)
```

Smote

```
[ ] # SMOTE
    # pip install imblearn
    # from imblearn.over_sampling import SMOTE

    print('Before OverSampling, counts of label 1: {}'.format(sum(y_train==1)))
    print('Before OverSampling, counts of label 0: {} \n'.format(sum(y_train==0)))

    Before OverSampling, counts of label 1: 195
    Before OverSampling, counts of label 0: 3893
```

```
[ ] sm = SMOTE(random_state=2)
    X_train_res, y_train_res = sm.fit_resample(X_train,y_train.ravel())

    print('After OverSampling, the shape of train_x: {}'.format(X_train_res.shape))
    print('After OverSampling, the shape of train_y: {}'.format(y_train_res.shape))

    print('After OverSampling, counts of label 1: {}'.format(sum(y_train_res == 1)))
    print('After OverSampling, counts of label 0: {}'.format(sum(y_train_res == 0)))

    After OverSampling, the shape of train_x: (7786, 19)
    After OverSampling, the shape of train_y: (7786,)
    After OverSampling, counts of label 1: 3893
    After OverSampling, counts of label 0: 3893
```

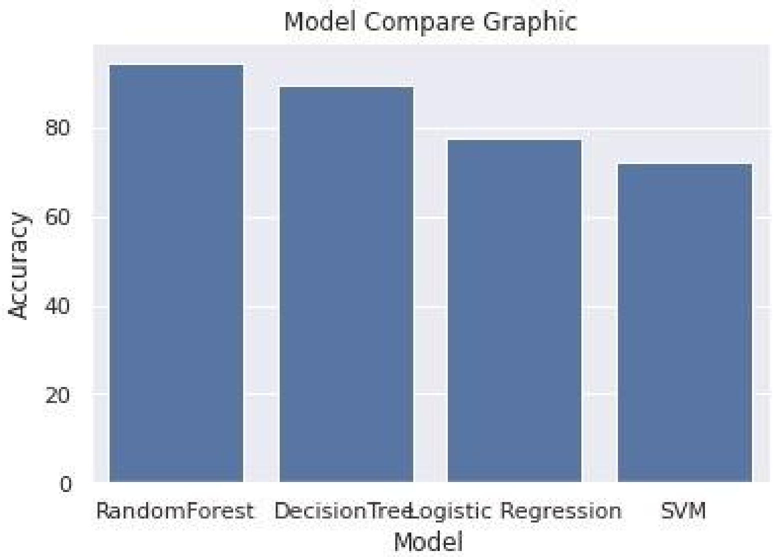
4) Model Selection

```
models = []
models.append(['Logistic
Regression',LogisticRegression(
random_state=0)])
models.append(['SVM',SVC(random
_state=0)])
models.append(['DecisionTree',D
ecisionTreeClassifier(random
_state
=0)])
models.append(['RandomForest',RandomForestClassifier(random_state=0)])
lst_1 =
[] for m
in
    range(len(models))
    :lst_2 = []
    model = models[m][1]
    model.fit(X_train_res,y_trai
    n_res)y_pred =
    model.predict(X_test)
    cm = confusion_matrix(y_test,y_pred)
    accuracies = cross_val_score(estimator= model, X =
X_train_res,y = y_train_res, cv=10)

# k-fOLD Validation
    roc = roc_auc_score(y_test,y_pred)
    precision =
```

```
    precision_score(y_test,y_pred)recall
    = recall_score(y_test,y_pred)
    f1 =f1_score(y_test,y_pr
    )print(models[m] [0],':')
    print(cm) print('Accuracy
    Score:
    ',accuracy_score(y_test,y_pred))
    print('')print('K-
Fold Validation Mean Accuracy: {:.2f} %'.format(accuracies.mean()
*100))
```

```
print('Standard Deviation: {:.2f}
        %'.format(accuracies.std()*100))
            print('')
            print('ROC AUC Score: {:.2f}
            %'.format(roc))print('')
            print('Precision: {:.2f}
            %'.format(precision))print('')
            print('Recall: {:.2f}
            %'.format(recall))print('')
            print('F1 Score: {:.2f}
            %'.format(f1))print('-'*40)
            print('')

            lst_2.append(models[m][0])
            lst_2.append(accuracy_score(y_test,y_pred
            )*100)lst_2.append(accuracies.mean()*100)
            lst_2.append(accuracies.std()*100)
            lst_2.append(roc)
            lst_2.append(prec
            ision)
            lst_2.append(reca
            ll)
            lst_2.append(f1)
            lst_1.append(lst_
            2)
              df2 =
              pd.DataFrame(lst_1,columns=['Model','Accuracy','K-
              Fold Mean
              Accuracy','Std.Deviation','ROC_AUC','Precision','R
              ecall','F1 Score'])

              df2.sort_values(by=['Accuracy','K-
              Fold Mean
              Accuracy'],inplace=True,ascending=False)df2

              # COMPARE
```

	Model	Accuracy	K-Fold Mean Accuracy	Std.Deviation	ROC_AUC	Precision	Recall	F1 Score
2	DecisionTree	89.726027	94.927664	5.821712	0.578570	0.160000	0.222222	0.186047
0	Logistic Regression	77.690802	78.640766	1.652194	0.751090	0.154762	0.722222	0.254902
1	SVM	72.113503	78.499724	1.881722	0.739134	0.130990	0.759259	0.223433



5) Model Tuning

```
[ ] grid_models = [(DecisionTreeClassifier(),[{'criterion':['gini','entropy'],'random_state':[0]}])]
```

```
[ ] for i,j in grid_models:
        grid = GridSearchCV(estimator=i,param_grid = j, scoring = 'accuracy',cv = 10)
        grid.fit(X_train_res,y_train_res)
        best_accuracy = grid.best_score_
        best_param = grid.best_params_
        print(' {}: \n Best Accuracy: {:.2f} %'.format(i,best_accuracy*100))
        print('')
        print('-'*25)
        print('')
```

```
 DecisionTreeClassifier():
 Best Accuracy: 95.20 %

-------------------------
```

```
classifier =
DecisionTreeClassifier(random_state=0)
classifier.fit(X_train_res, y_train_res)
y_pred = classifier.predict(X_test)
y_prob = classifier.predict_proba(X_test)
[:,1]cm = confusion_matrix(y_test,
y_pred)

print(classification_report(y_test, y_pred))
print(f'ROC AUC score: {roc_auc_score(y_test,
y_prob)}')print('Accuracy Score:
',accuracy_score(y_test, y_pred))

# Visualizing Confusion
Matrixplt.figure(figsize
= (8, 5))
sns.heatmap(cm, cmap = 'Blues', annot = True, fmt = 'd', linewidths = 5
, cbar = False, annot_kws = {'fontsize': 15},
            yticklabels = ['No stroke', 'Stroke'], xticklabels =
```

```python
['Predicted no stroke', 'Predicted
stroke']) plt.yticks(rotatio
n = 0)plt.show()

# Roc Curve
false_positive_rate, true_positive_rate, thresholds =
 roc_curve(y_test,y_prob)
roc_auc = auc(false_positive_rate, true_positive_rate)

sns.set_theme(style =
'white')
plt.figure(figsize = (8,
8))
plt.plot(false_positive_rate,true_positive_rate, color =
'#b01717', label = 'AUC = %0.3f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1], linestyle = '--', color = '#174ab0')
plt.axis('tight')
plt.ylabel('True
Positive
Rate')plt.xlabel('False
```

```
              precision    recall  f1-score   support

           0       0.96      0.93      0.95       968
           1       0.16      0.22      0.19        54

    accuracy                           0.90      1022
   macro avg       0.56      0.58      0.57      1022
weighted avg       0.91      0.90      0.91      1022

ROC AUC score: 0.5785697887970616
Accuracy Score:  0.8972602739726028
```

CHAPTER V

Results

	Accuracy	**Precision score**	**Recall score**	**F1 score**
Logistic regression	78%	0.15%	0.72%	0.25%
Support vector machine	72%	0.13%	0.76%	0.22%
Decision tree classifier	89%	0.16%	0.22%	0.19%
Random forest regression	94%	0.29%	0.04%	0.07%

Table 2: Results

CHAPTER VI

CONCLUSION

Stroke is a critical medical condition that should be treated before it becomes critical. Building an effective machine learning model can definitely help in the early prediction of stroke and reduce the severe impact on the future.

In this seminar we showed the performance of various machine learning algorithms for successfully predicting the stroke based on multiple physiological attributes. Out of all the algorithms chosen, performance of Decision tree classification was best with an accuracy of 89%. Among all the precision, recall and F1 scores obtained, Decision tree has performed better.

CHAPTER VII

REFERENCES

List all the material used from various sources for making this seminar.

1. Prediction of Brain Stroke Severity Using Machine Learning- Bandi V, Bhattacharyya D, Midhunchakkravarthy D. In: International Information and Engineering Technology Association, (2020).

2. Predicting Stroke from Electronic Health Records- Nwosu, C.S., Dev S., Bhardwaj P., Veeravalli B., John D. In: 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society IEEE (2019).

3. Prediction of Stroke using Data Mining Classification Techniques- Almadani O, Alshammari R.In: International Journal of Advanced Computer Science and Applications (IJACSA) (2018).

4. Probability of stroke: a risk profile from the Framingham Study- Wolf P, Ralph B, Belanger A,Kannel W. Journal of American Heart Association(2017)

5. Development of an Algorithm for Stroke Prediction: A National Health Insurance Database Study in Korea- Seung Nam Min a, Se Jin Park b,Dong Joon Kim c, Murali Subramaniyam d , Kyung-Sun Lee. Eur Neural (2018).

6. Dataset named ‘Stroke Prediction Dataset’ from Kaggle:
https://www.kaggle.com/ahmtcnbs/stroke-prediction-xgboost-97/data