Stroke Prediction

Bhavesh Khanchandani

Problem Statement

- 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 a stroke based on the input parameters like gender, age, various diseases, and smoking status.

Proposed Solution

Data Preprocessing : This step performs all pre-processing steps such as data manipulation, data filling, converting categorical into numeric, and all processes.

The EDA process involves performing

1. Univariate Analysis
2. Bivariate analysis
3. Removing Missing values if any / Outlier treatment
4. Machine Learning : Apply Appropriate machine learning algorithm to Predict the probability of a candidate will work for the company. and also check if the model is to be underfitting or overfitting if it has then solves this by using cross-validation technique, or perform hyper parameters tuning to improve model performance.

Descriptive Analysis

```
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	36.6	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.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
```

```
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
```

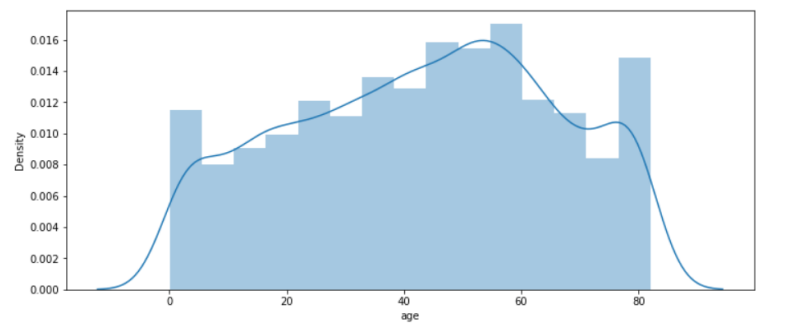
```
df.apply(lambda x: len(x.unique()))
```

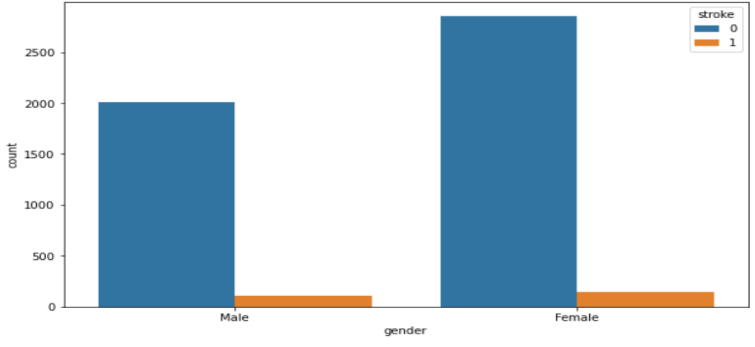
```
id                 5110
gender                3
age                 104
hypertension          2
heart_disease         2
ever_married          2
work_type             5
Residence_type        2
avg_glucose_level  3979
bmi                 419
smoking_status        4
stroke                2
dtype: int64
```

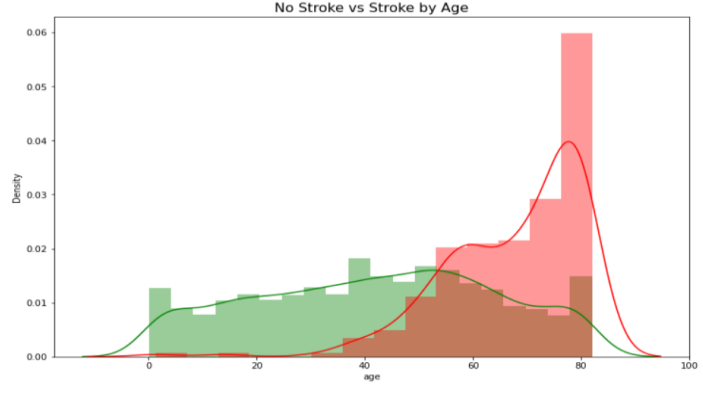
Data Preprocessing

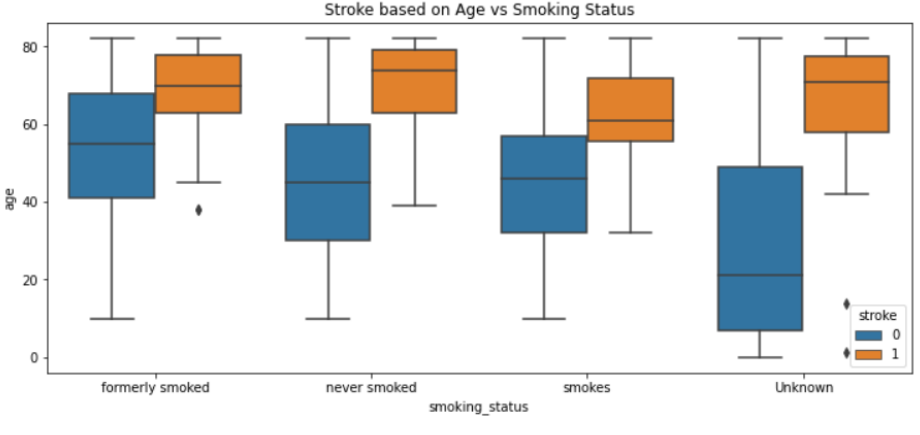
- Drop Id column
- Replace the missing values with mean of bmi attribute
- Remove other gender because it has only 1 value

Exploratory Data Analysis

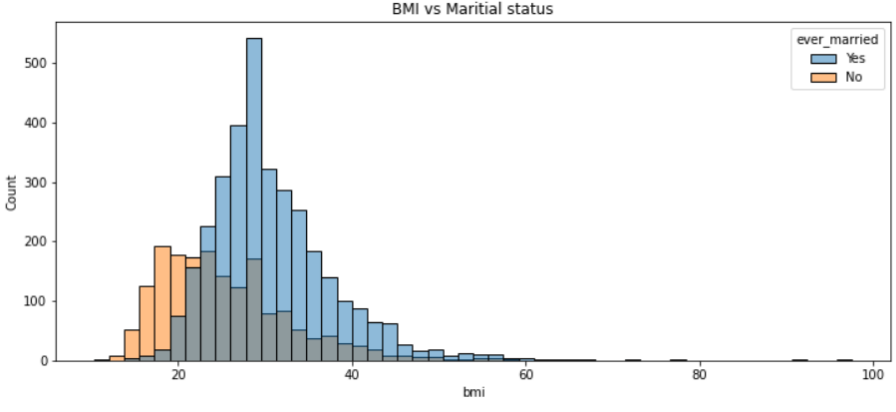


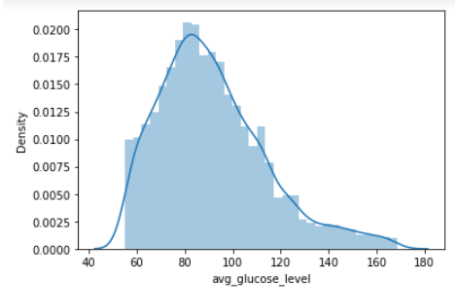


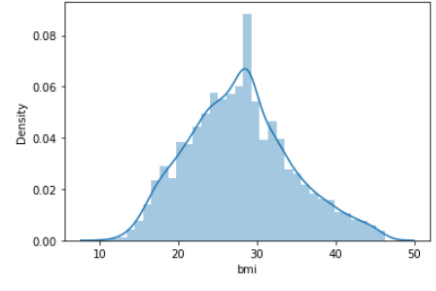


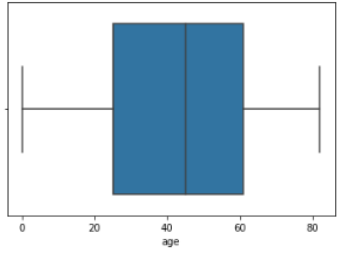


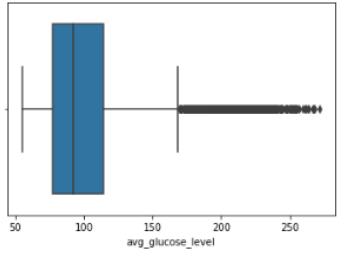
Exploratory Data Analysis

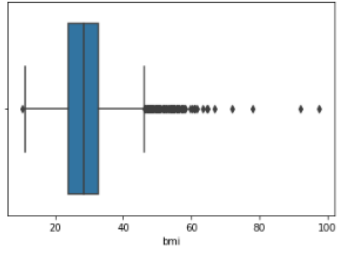












Machine Learning Modelling

- Feature selection for algorithm

  X = df.iloc[:,0:-1]

  y = df.iloc[:, -1]

  Selecting input and output features

- Label encoding

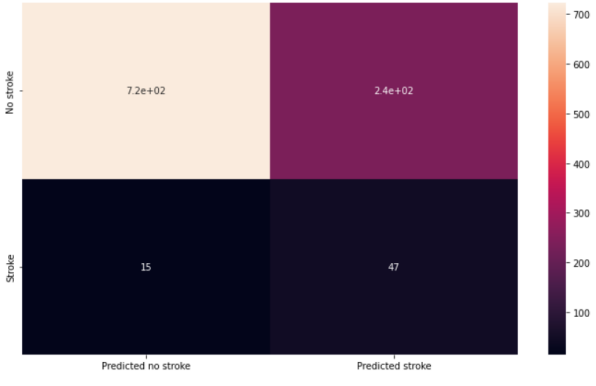
Label encoded gender, ever_married and Residence_type columns

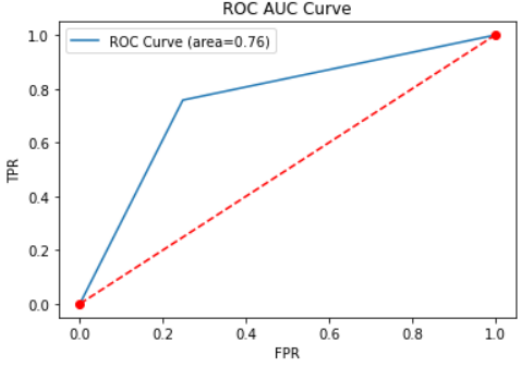
- Split the data using Train test split

Machine Learning Modelling

Model selection , scores and other stats

- Logistic Regression, `model score= 0.7524461839530333  Accuracy: 0.7524461839530333`



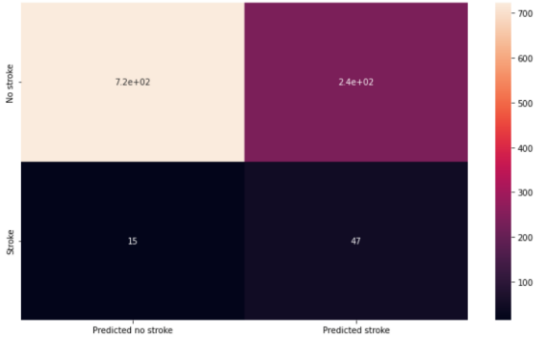


	precision	recall	f1-score	support
0	0.98	0.75	0.85	960
1	0.16	0.76	0.27	62
accuracy			0.75	1022
macro avg	0.57	0.76	0.56	1022
weighted avg	0.93	0.75	0.82	1022

Machine Learning Modelling

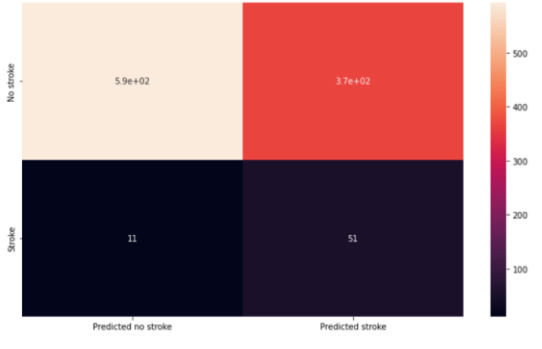
- KNN

	precision	recall	f1-score	support
0	0.98	0.75	0.85	960
1	0.16	0.76	0.27	62
accuracy			0.75	1022
macro avg	0.57	0.76	0.56	1022
weighted avg	0.93	0.75	0.82	1022



- Random Forest Classifier

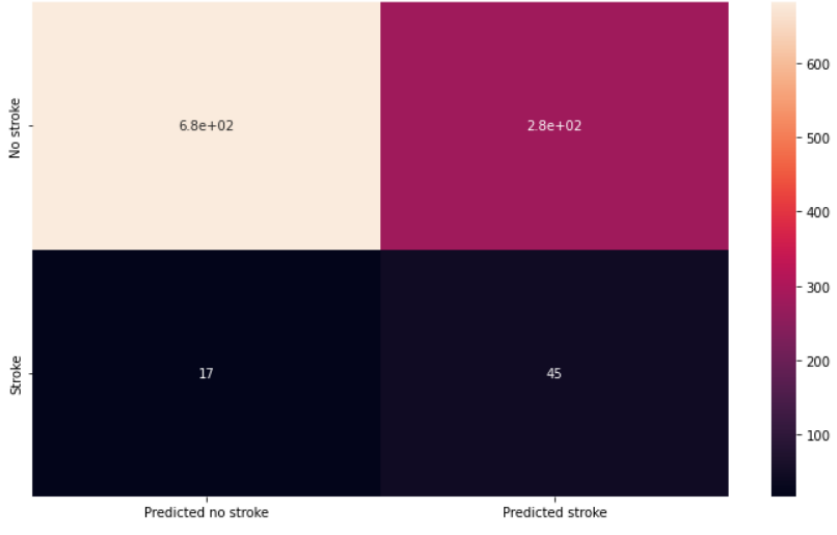
	precision	recall	f1-score	support
0	0.98	0.62	0.76	960
1	0.12	0.82	0.21	62
accuracy			0.63	1022
macro avg	0.55	0.72	0.49	1022
weighted avg	0.93	0.63	0.73	1022



Machine Learning Modelling

- DecisionTreeClassifier

	precision	recall	f1-score	support
0	0.98	0.71	0.82	960
1	0.14	0.73	0.23	62
accuracy			0.71	1022
macro avg	0.56	0.72	0.53	1022
weighted avg	0.92	0.71	0.79	1022



Hyperparameters Tuning

- Logistic regression

```
Best Score is 79.33333333333333
Best Parameters is {'C': 0.01, 'penalty': 'l2'}
```

- KNN

```
Best Score is 92.65384615384616
Best Parameters is {'metric': 'manhattan', 'n_neighbors': 5}
```

- Random Forest

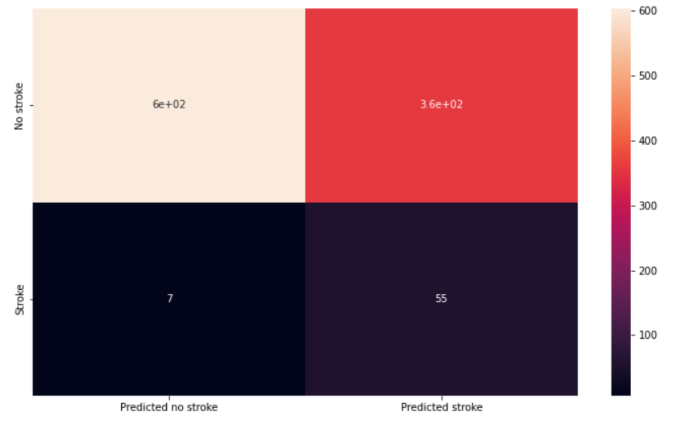
```
Best Score is 96.25641025641026
Best Parameters is {'criterion': 'entropy', 'n_estimators': 100}
```

Conclusion

Now Let's apply the highest accuracy model with best hyperparameters

model = RandomForestClassifier(n_estimators=200, criterion='entropy')

```
model score= 0.6438356164383562    Accuracy: 0.6438356164383562
```



	precision	recall	f1-score	support
0	0.99	0.63	0.77	960
1	0.13	0.89	0.23	62
accuracy			0.64	1022
macro avg	0.56	0.76	0.50	1022
weighted avg	0.94	0.64	0.74	1022