
STROKE PREDICTION

Project Report

SUPERVISOR: Ms. Vandita Grover

SUBMITTED BY

Sambit Basu 19001570034

Sarvesh Sharma 19001570036



2022

Department of Computer Science
ACHARYA NARENDRA DEV
COLLEGE

ACKNOWLEDGEMENT

This Project was jointly undertaken by **Sarvesh Sharma** and **Sambit Basu** of B.Sc. (Hons.) Computer Science as part of Project Work of **Data Mining Project 2022**, under the supervision of **Ms. Vandita Grover**. Our primary thanks go to her, who poured over every inch of our project with painstaking attention and helped us throughout the working of the project. It is our privilege to acknowledge our deepest gratitude to her for inspiring us, which has helped us immensely. We are extremely grateful to her for her unstilted support and encouragement in the preparation of this project.

Sarvesh Sharma
(AB-1271)

Sambit Basu
(AB-1264)

ACHARYA NARENDRA DEV COLLEGE

(University of Delhi)

CERTIFICATE

This is to certify that the project entitled “**STROKE PREDICTION**” has been completed by **Sarvesh Sharma** and **Sambit Basu** of Bachelor of Sciences in Computer Science (Hons.) as a part of Project Work of **Data Mining Project 2022** from Acharya Narendra Dev College, University of Delhi under the supervision of **Ms. Vandita Grover**.

Prof. RAVI TOTEJA
Officiating Principal
Acharya Narendra Dev College
University of Delhi

△

Ms.VANDITA GROVER
(Supervisor)
Assistant Professor
Department of Computer Science
Acharya Narendra Dev College
University of Delhi

Table of Contents

Chapter 1	PROBLEM STATEMENT	4
Chapter 2	DATA MINING TECHNIQUES	5
	DM Techniques	5
	Classification	5
Chapter 3	DATASET DESCRIPTION	8
	Dataset	8
Chapter 4	DATA PREPROCESSING	22
	Handling Null Values	23
	Feature Scaling	24
	Feature Selection and Conversion	25
	Data Sampling and Subsetting	27
Chapter 5	BUILDING MODELS	29
	K-NN Model	29
	Decision Tree Model	30
	Naïve-Bayes Model	30
Chapter 6	MODEL EVALUATION AND RESULTS	31
		31
Chapter 7	INFERENCES AND CONCLUSION	36
	References	37
	Appendix	38

Chapter 1

PROBLEM STATEMENT

The aim of the project is to predict stroke in patients.

A stroke occurs when a blood vessel in the brain ruptures and bleeds, or when there's a blockage in the blood supply to the brain. The rupture or blockage prevents blood and oxygen from reaching the brain's tissues.

According to the World Health Organisation (WHO) stroke is the 2nd leading cause of death globally, responsible for approximately 11% of total deaths.[9]

A stroke happens without many further symptoms. So the time to respond when a stroke happens is very less, and it can be fatal too.
So the goal is to predict if an individual will suffer a stroke based on his/her medical background.

Chapter 2

DATA MINING TECHNIQUES

The act of automatically searching for large stores of information to find trends and patterns that go beyond simple analysis procedures. Data mining utilises complex mathematical algorithms for data segments and evaluates the probability of future events.[2]

2.1. DM Techniques

Data mining techniques are deployed to scour large data sets in order to find novel and useful patterns that might otherwise remain unknown. They also provide the capability to predict the outcome of a future observation.[1]

2.1.1. Classification

A classification model is an abstract representation of the relationship between the attribute set and the class label.

It can be expressed mathematically as a target function f that takes as input the attribute set x and produces an output corresponding to the predicted class label. The model is said to classify an instance (x, y) correctly if $f(x) = y$. [1]

2.1.2. Association

Association analysis is useful for discovering interesting relationships hidden in large data sets. The uncovered relationships can be represented in the form of sets of items present in many transactions, which are known as frequent itemsets, or association rules, that represent relationships between two itemsets.[1]

2.1.3. Clustering

Cluster analysis groups data objects based on information found only in the data that describes the objects and their relationships. The goal is that the objects within a group be similar (or related) to one another and different from (or unrelated to) the objects in other groups. The greater the similarity (or homogeneity) within a group and the greater the difference between groups, the better or more distinct the clustering.[1]

2.2. CLASSIFICATION

In classification, the most common techniques used are KNN, Naive Bayes and Decision Tree.

2.2.1. K-NN

The K-NN (k - Nearest Neighbours) algorithm assumes the similarity between the new case/data and available cases and puts the new case into the category that is most similar to the available categories.[2]

The K-NN algorithm stores all the available data and classifies a new data point based on the similarity. So when any new data appears then it can be easily classified into a well-suited category by using K- NN algorithm.[2]

2.2.2. Naïve-Bayes

The Naïve Bayes algorithm is a supervised learning algorithm, which is based on the **Bayes theorem** and is used for solving problems dealing with classification.[2]

It predicts on the basis of the probability of an object, so it is a probabilistic classifier.[2]

2.2.3. Decision Tree

Decision Tree is a Supervised learning technique that can be used for solving

classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.[2]

In a Decision tree, there are two nodes: the Decision Node and the Leaf Node. Decision nodes are used to make any decision and have multiple branches; Leaf nodes are the output of those decisions and do not contain any further branches.[2]

The decisions or the test are performed on the basis of features of the given dataset.[2]

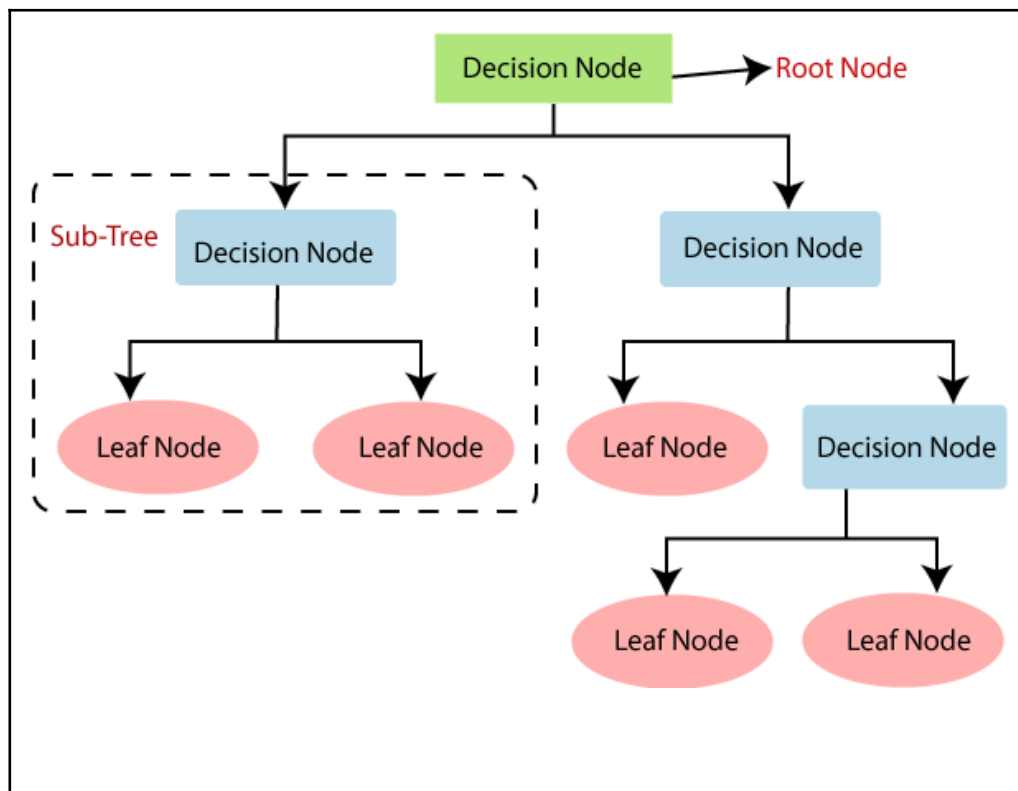


Fig 2.1 Decision Tree

[2]

Chapter 3

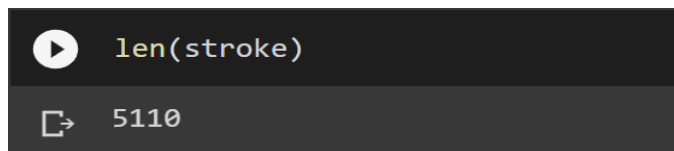
DATASET DESCRIPTION

3.1 Dataset

The Stroke prediction dataset is taken from kaggle.[9]
The name of the file is 'healthcare-dataset-stroke-data.csv'.

3.1.1. Number of Records

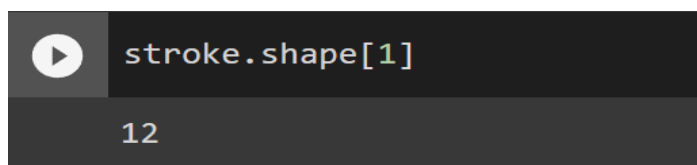
The dataset contains 5110 records.



A screenshot of a Jupyter Notebook cell. The top bar is dark grey with a play button icon and the code `len(stroke)`. The bottom bar is a lighter grey and displays the output `5110`.

3.1.2 Number of Attributes

The dataset contains 12 attributes.

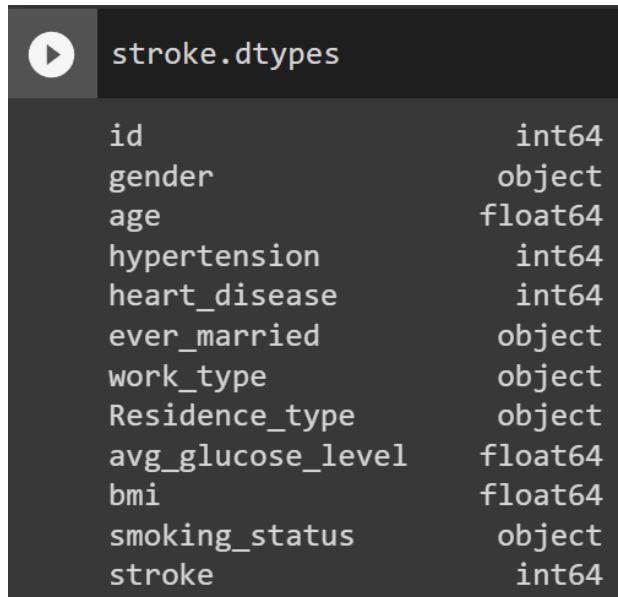


A screenshot of a Jupyter Notebook cell. The top bar is dark grey with a play button icon and the code `stroke.shape[1]`. The bottom bar is a lighter grey and displays the output `12`.

3.1.3. Types of Attributes

- 1) id: Nominal (Categorical)
- 2) gender: Nominal (Categorical)
- 3) age: Ratio (Numeric)
- 4) hypertension: Nominal (Categorical) (1:Yes, 0:No)

- 5) heart_disease: Nominal (Categorical) (1:Yes, 0:No)
- 6) ever_married: Nominal (Categorical)
- 7) work_type: Nominal (Categorical)
- 8) Residence_type: Nominal (Categorical)
- 9) avg_glucose_level: Ratio (Numeric)
- 10) bmi: Interval(Numeric)
- 11) smoking_status: Nominal (Categorical)
- 12) stroke: Nominal (Categorical) (1:Yes, 0:No)

A screenshot of a Jupyter Notebook cell. The cell has a play button icon and the text 'stroke.dtypes'. Below this, a table displays the data types for each column in the 'stroke' dataset. The columns are listed on the left, and their corresponding data types are on the right.

id	int64
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

3.1.4. Missing Values or Nulls

The 'bmi' attribute has 201 Null values and the 'smoking_status' attribute has 1544 Null values.

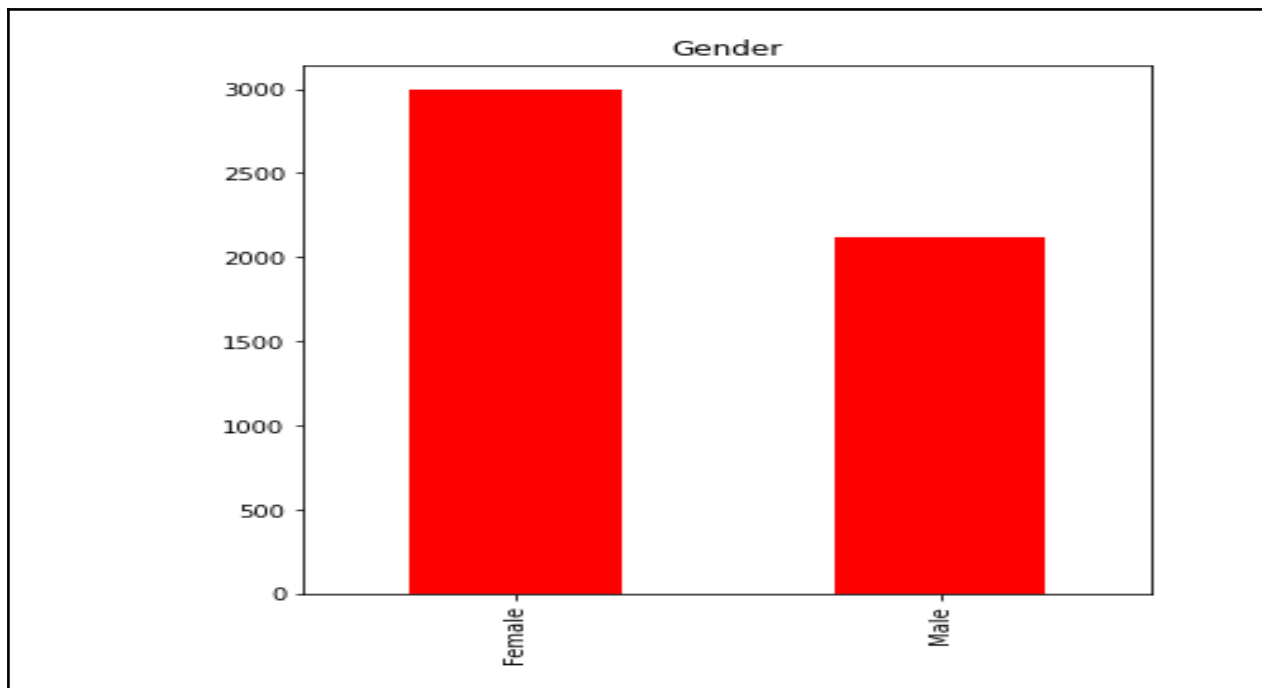
```
stroke.isnull().sum()

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 1544
stroke      0
dtype: int64
```

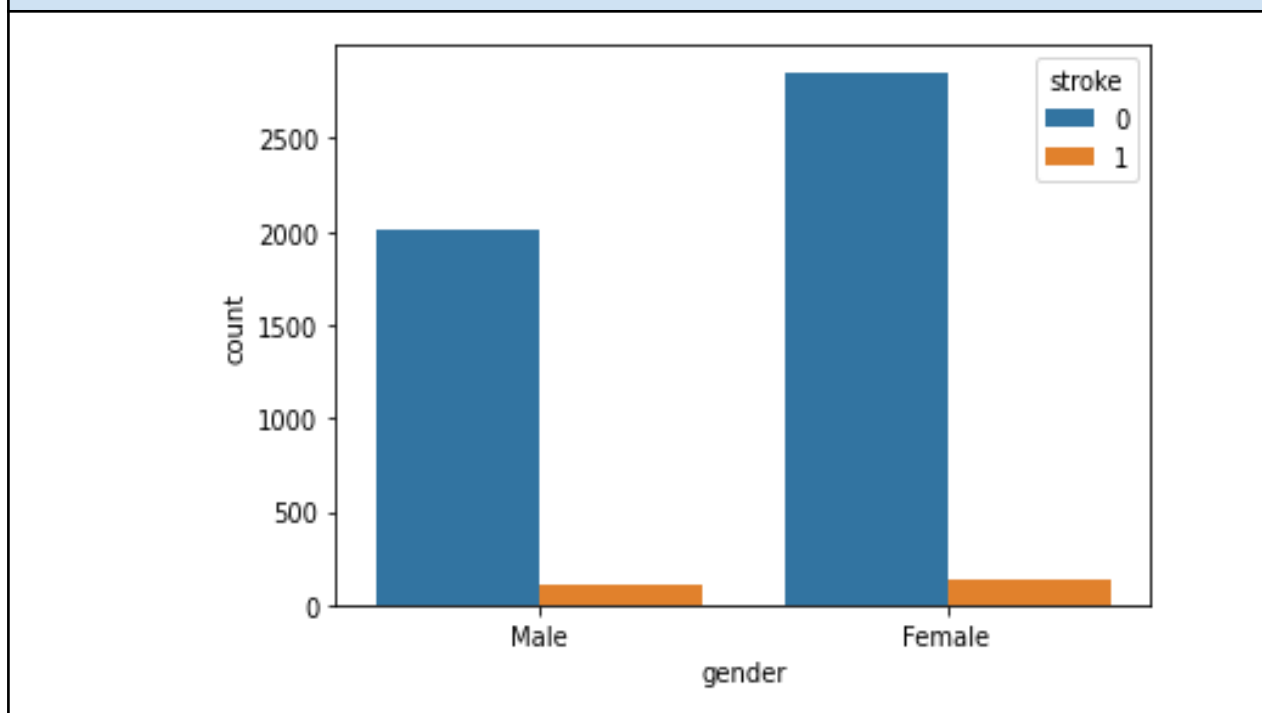
3.1.5. Attributes Description

- 1) id: unique identifier
- 2) gender: "Male" or "Female"
- 3) age: age of the patient
- 4) hypertension: 0 if the patient doesn't have hypertension, 1 if the patient has hypertension
- 5) heart_disease: 0 if the patient doesn't have any heart diseases, 1 if the patient has a heart disease
- 6) ever_married: "No" or "Yes"
- 7) work_type: "children", "Govt_jov", "Never_worked", "Private" or "Self-employed"
- 8) Residence_type: "Rural" or "Urban"
- 9) avg_glucose_level: average glucose level in blood
- 10) bmi: body mass index
- 11) smoking_status: "formerly smoked", "never smoked", "smokes" or "Unknown"
- 12) stroke: 1 if the patient had a stroke or 0 if not

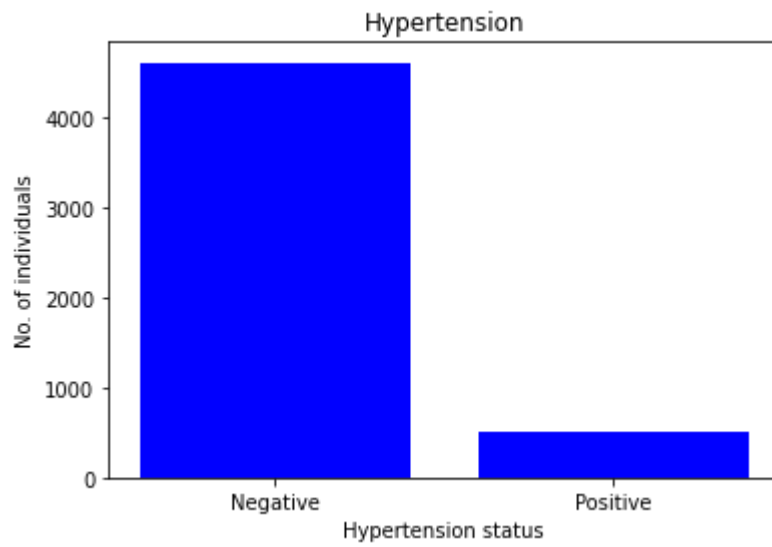
3.1.6. Distribution/Histograms



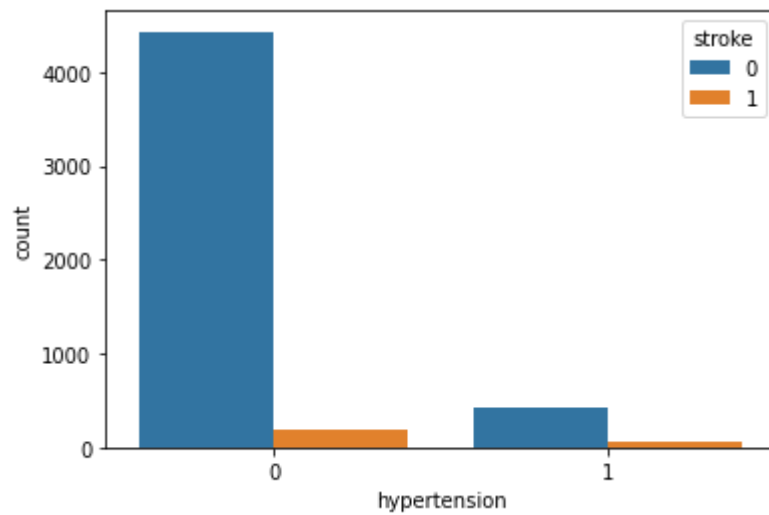
This plot shows the number of male and female patients



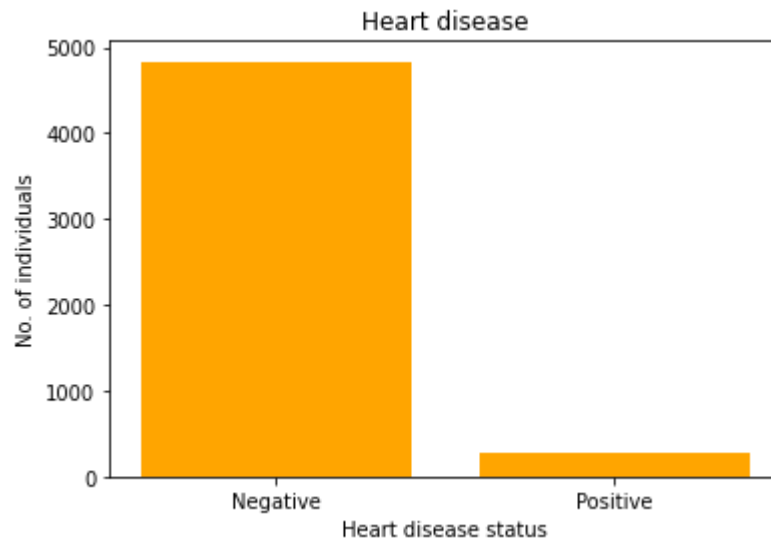
This plot shows the distribution of stroke in different categories of gender



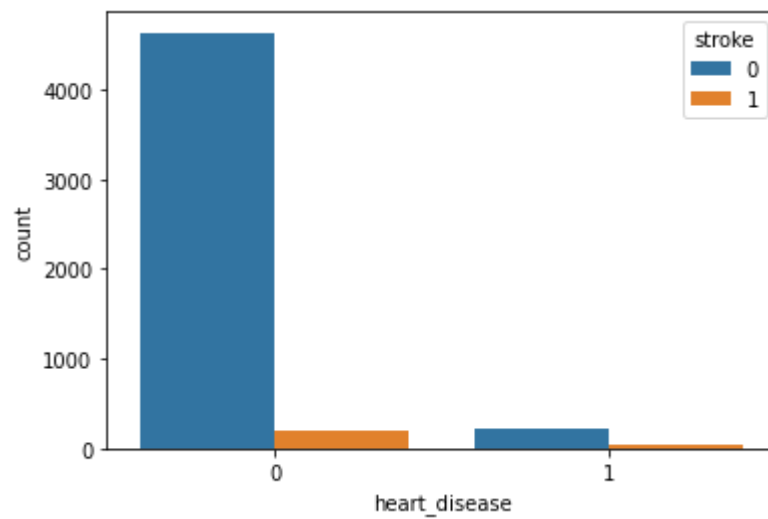
This plot shows the number of patients with hypertension



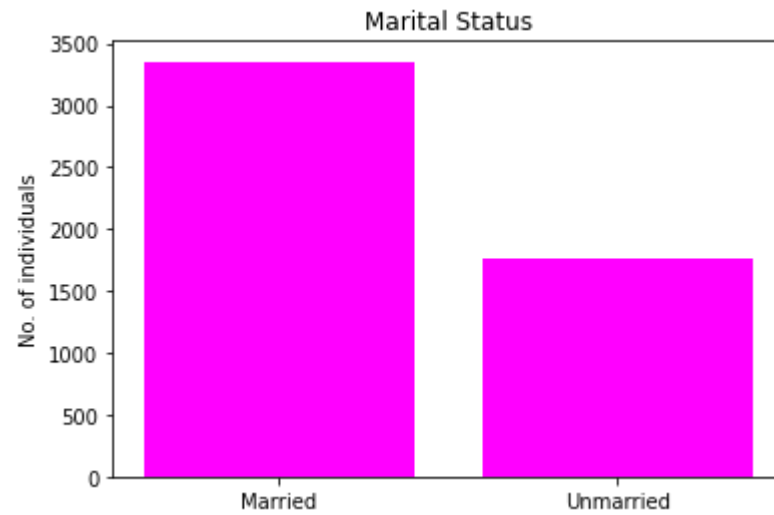
This plot shows the distribution of stroke in different categories of hypertension



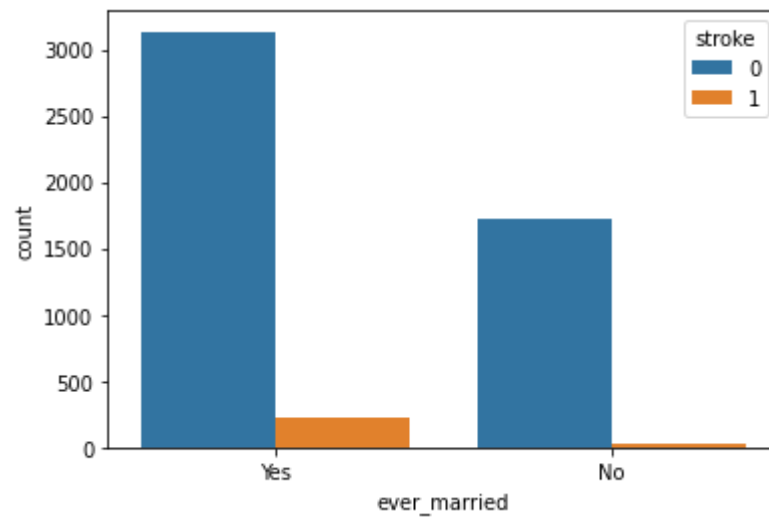
This plot shows the number of patients having heart disease



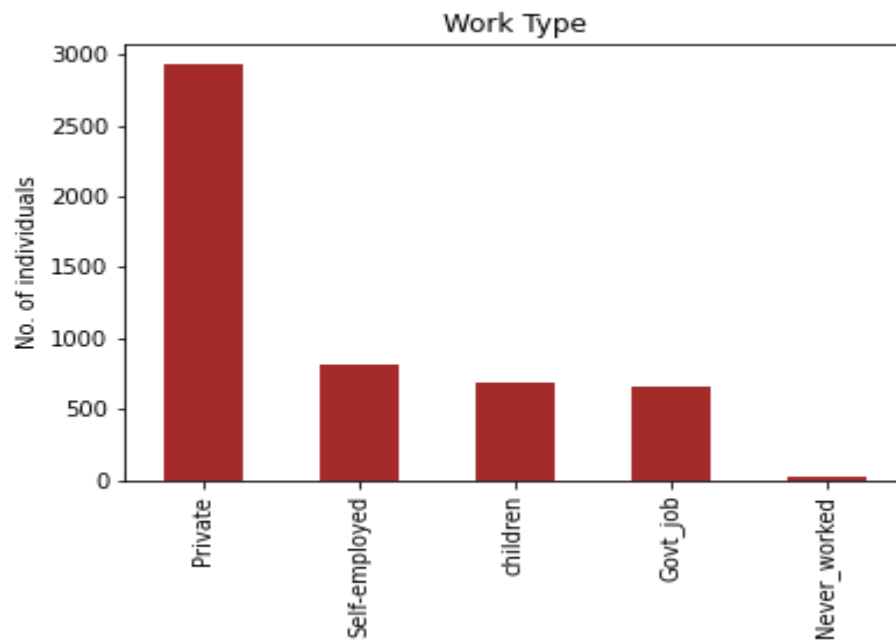
This plot shows the distribution of stroke in different categories of heart disease



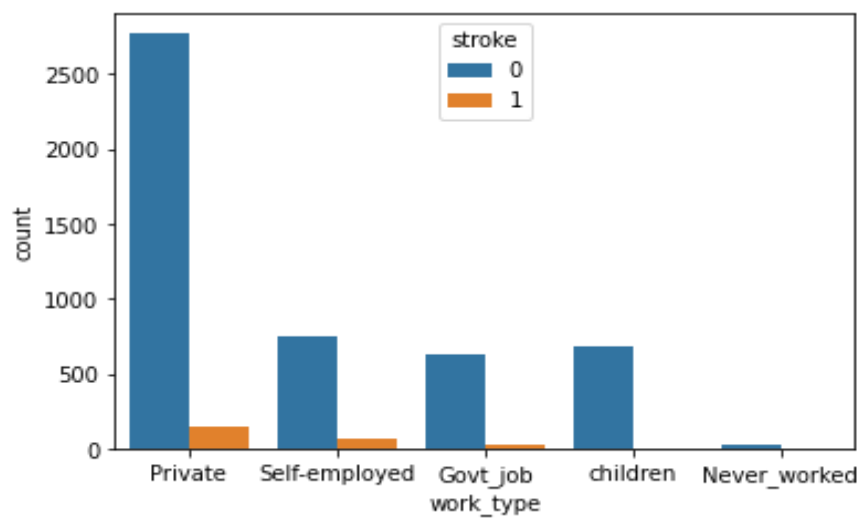
This plot shows the number of Married and Unmarried patients



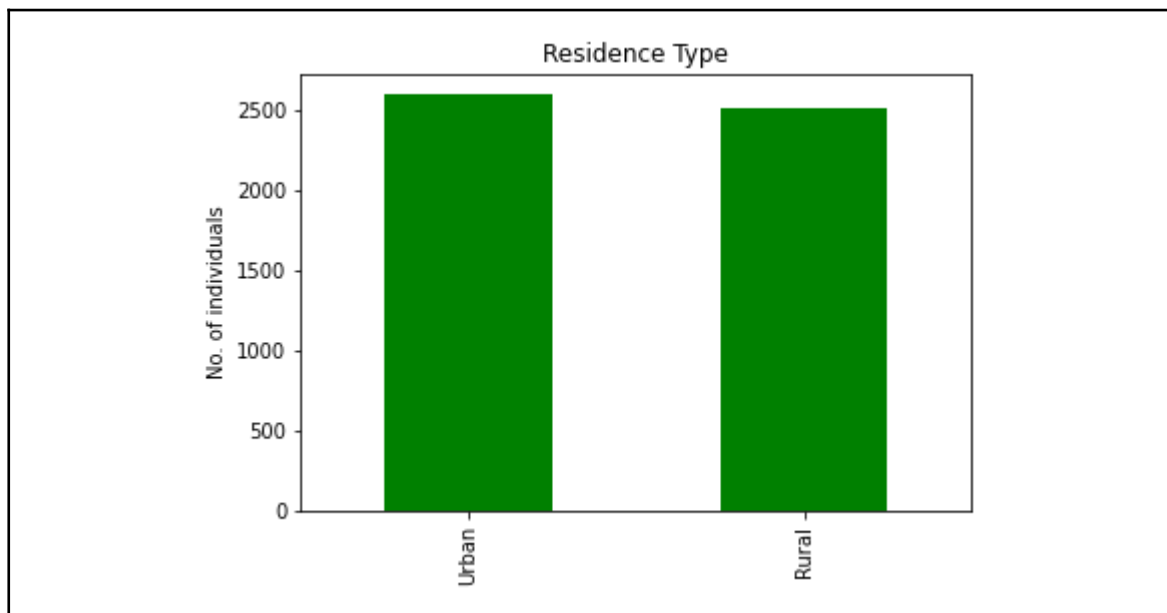
This plot shows the distribution of stroke in different categories of marital status



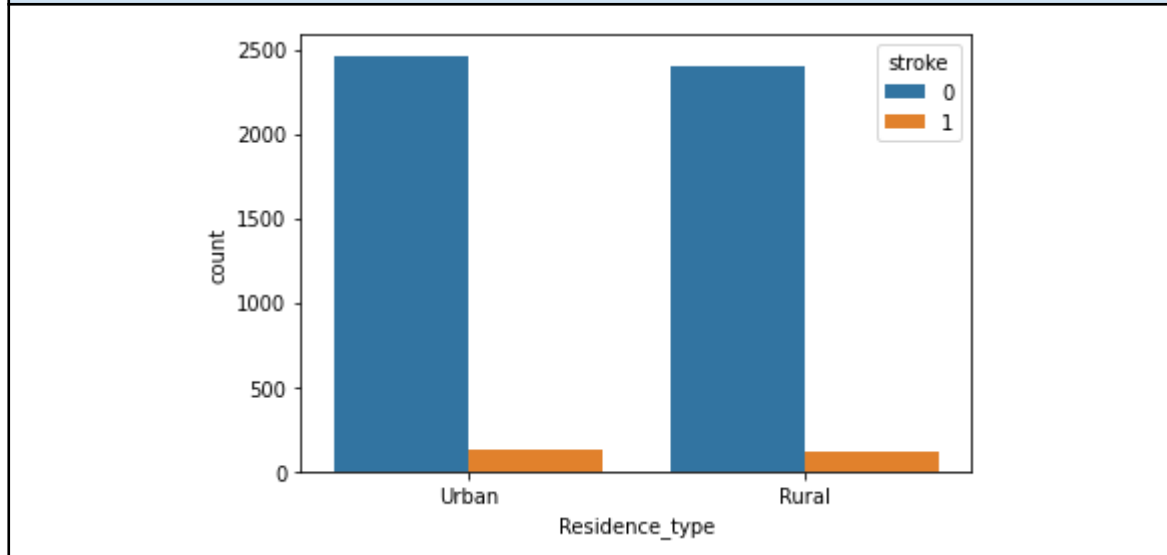
This plot shows the number of patients in different work types



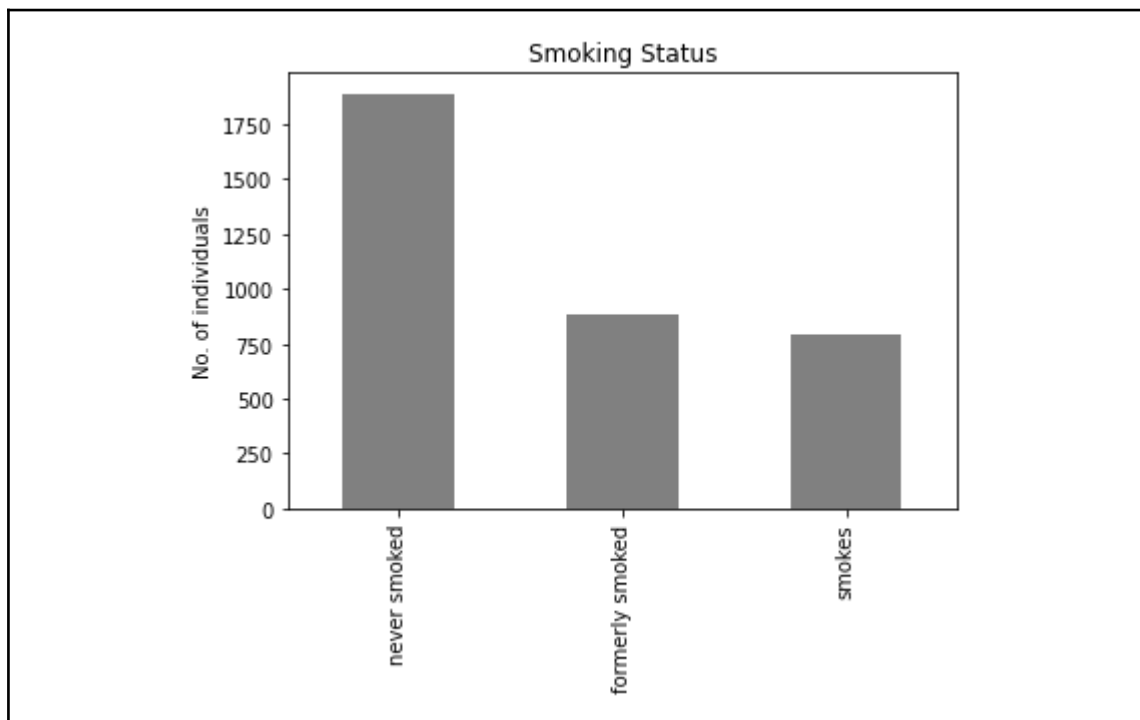
This plot shows the distribution of stroke in different categories of work type



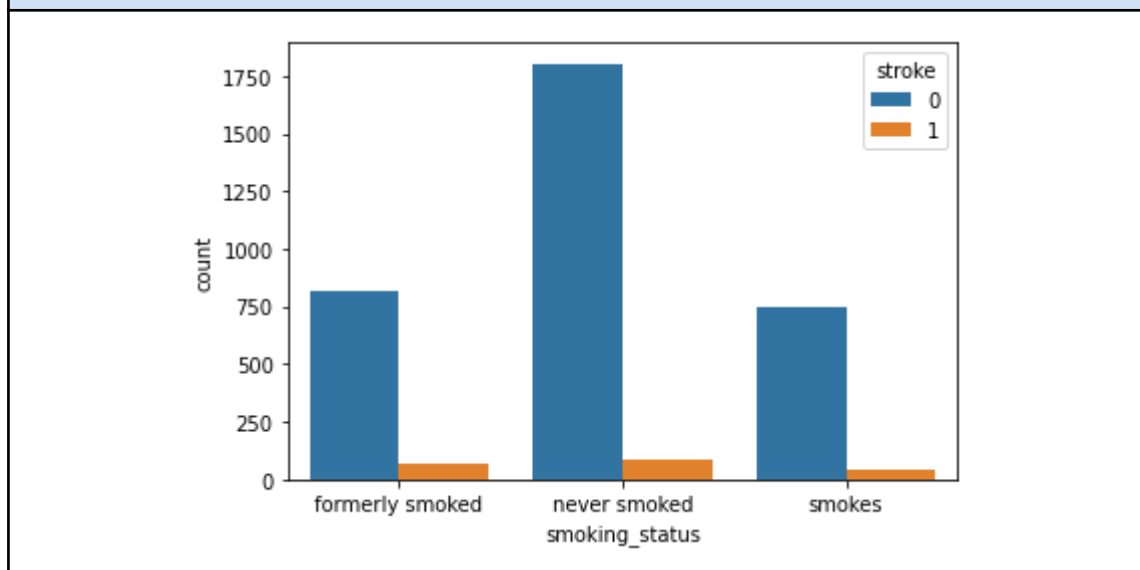
This plot shows the number of patients living in urban or rural areas



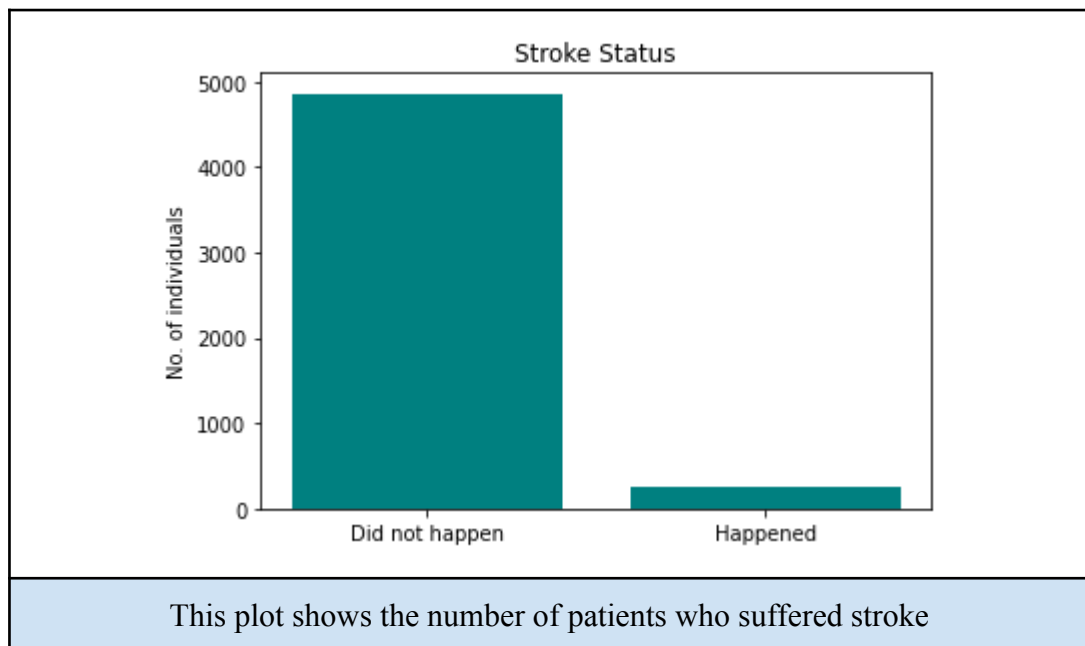
This plot shows the distribution of stroke in different categories of Residence type



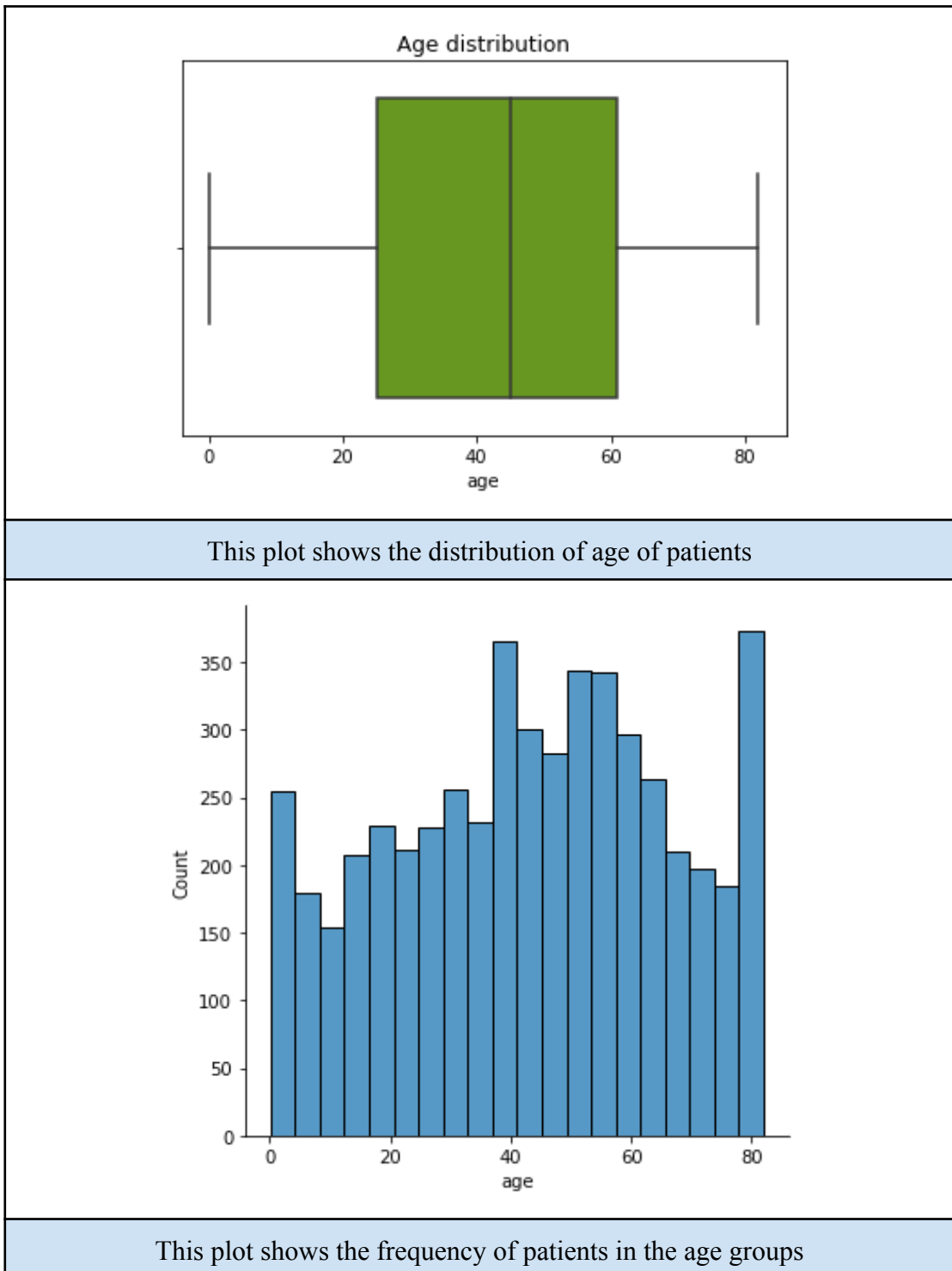
This plot shows the number of patients who have experienced smoking

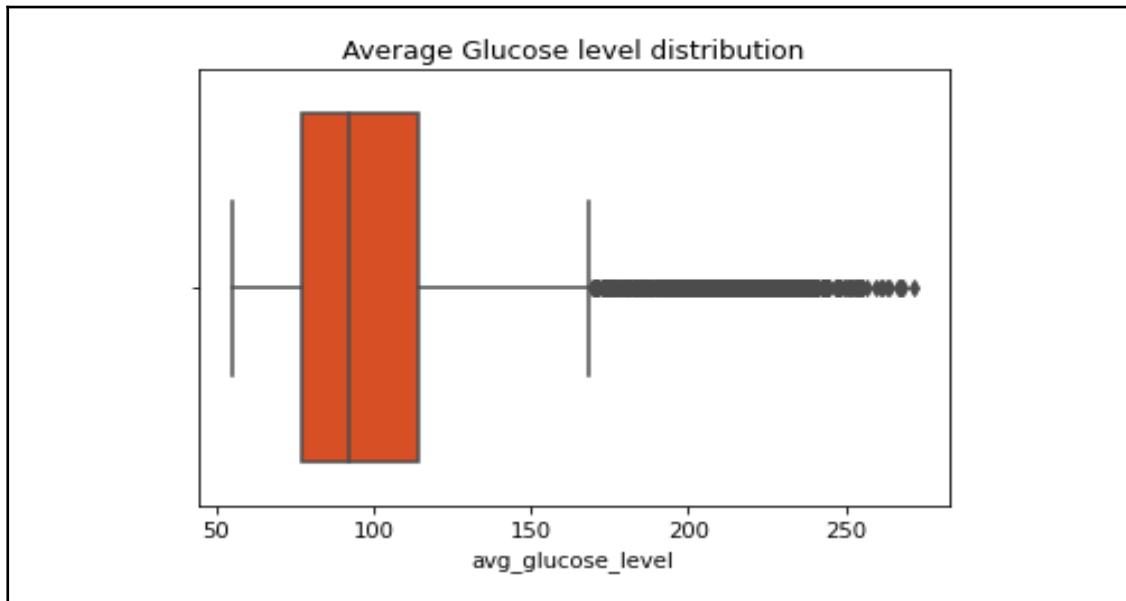


This plot shows the distribution of stroke in different categories of Smoking status

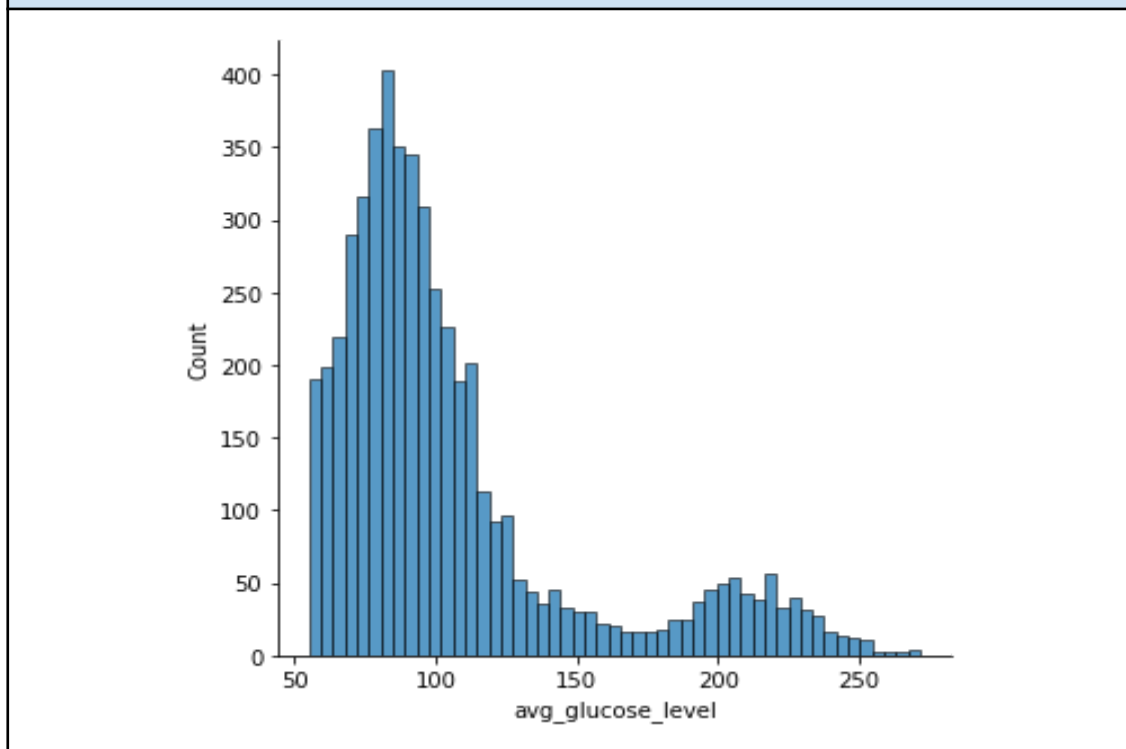


3.1.7. Detecting Outliers

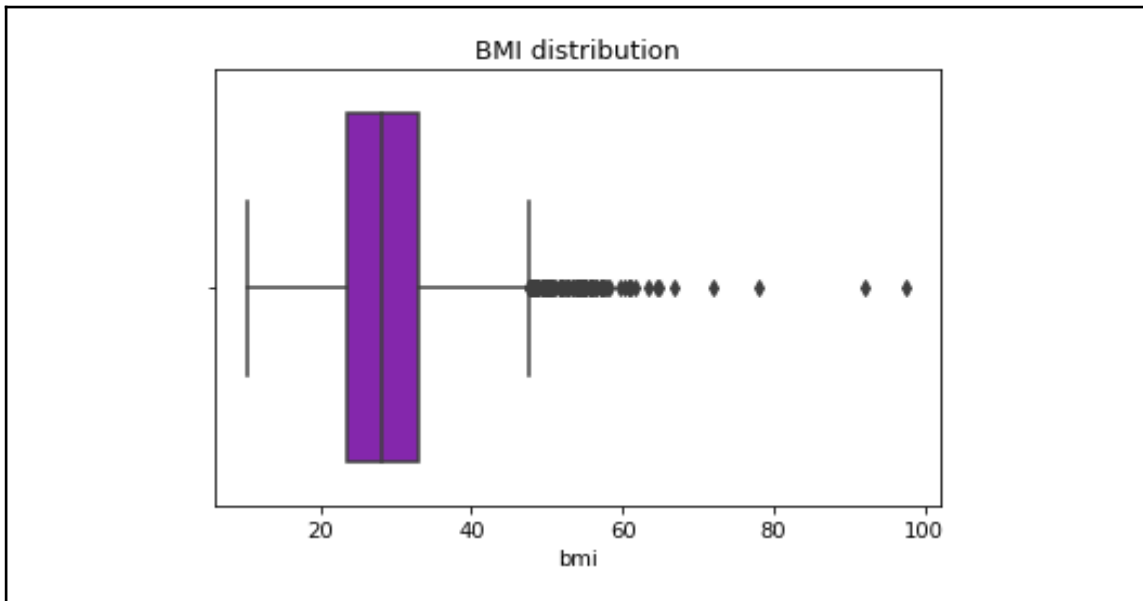




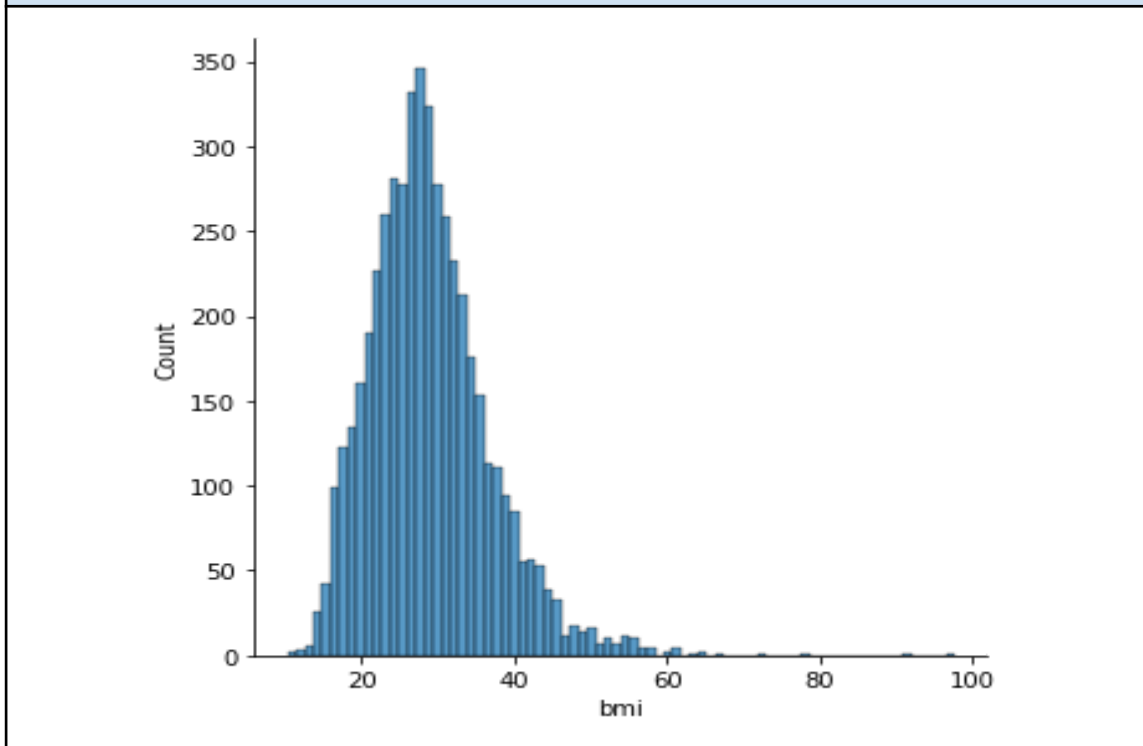
This plot shows the distribution of average glucose level in the patients



This plot shows the frequency of patients having different average glucose level



This plot shows the distribution of BMI of the patients



This plot shows the frequency of patients having different BMI

Chapter 4

DATA PREPROCESSING

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.[2]

When creating a machine learning project, we can not always come across clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put it in a formatted way. So for this, we use data preprocessing tasks.[2]

Some general steps involved in data preprocessing are:[2]

- 1) Finding missing data and taking care of them.
- 2) Making sure that the data are in the right data types as needed for the work ahead.
- 3) Encoding categorical data.
- 4) Splitting the dataset into training and testing data.
- 5) Feature scaling.
- 6) Feature selection and conversion.
- 7) Data sampling and subsetting.

4.1 Handling Null Values

Null values are placed in a record when that attribute is missing or unknown. They can sometimes affect our computation because they represent a lack of data and our aggregate functions get a smaller amount of data to work with.

Null values can be there in the dataset for various reasons, like the person whose data is being collected is unwilling to share some information, or the data is just not available or discoverable for some reasons. Null values can also come if the data is corrupted.

There are quite a few ways to deal with Null values:[4]

- 1) Deleting the records having null values. This is not always a good option because other attributes of the record may have important data, and in a dataset, many records can have null values, so removing the records would take away a big chunk of the original dataset and we will be left with a thin data collection.
- 2) Imputing missing values with mean, median or mode. Attributes that are numeric in general, that have missing values, can be imputed by taking the mean or median of the other values in the attribute. This method, though, does not factor in the covariance between variables.
Attributes that are categorical can be imputed by taking the mode of the remaining values. Although this adds some new features to the model while encoding, which might affect the performance in the end.
- 3) Using methods like forward fill or backward fill. We can fill the missing values by the values which precede or succeed the missing values. Although this method might have its drawbacks as we might miss out on some important data factors.
- 4) Predicting the missing values. We can try to predict the values which are missing using the existing data points, But if our dataset has a number of missing values, then there is not much data that the prediction algorithm can work on. Also, some datasets might be such that predicting the missing data is not an option.

```
stroke['bmi'].fillna(stroke.bmi.median(),inplace=True)
```

```
stroke['smoking_status'].fillna(stroke['smoking_status'].mode()[0],inplace=True)
```



```
stroke.isnull().sum()
```

```
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
```

4.2 Feature Scaling

Feature Scaling is used to standardise the independent features in the data in a fixed range. It is performed during the data pre-processing to handle highly varying magnitudes or values or units. Without feature scaling, a machine learning algorithm tends to weigh greater values, higher and consider smaller values as the lower values, regardless of the unit of the values.[3]

4.2.1 Normalisation

Normalisation is a scaling technique in Machine Learning. It is applied during data preparation to change the values of numeric columns in the dataset to use a common scale. It is required only when features of machine learning models have different ranges, it is not necessary for all datasets.[2]

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

4.2.2 Standardisation

Standardisation scaling is also known as Z-score normalisation, in which values are centred around the mean with a unit standard deviation. This means the attribute becomes zero and the resultant distribution has a unit standard deviation. We can calculate the standardisation by subtracting the feature value from the mean and dividing it by standard deviation.[2]

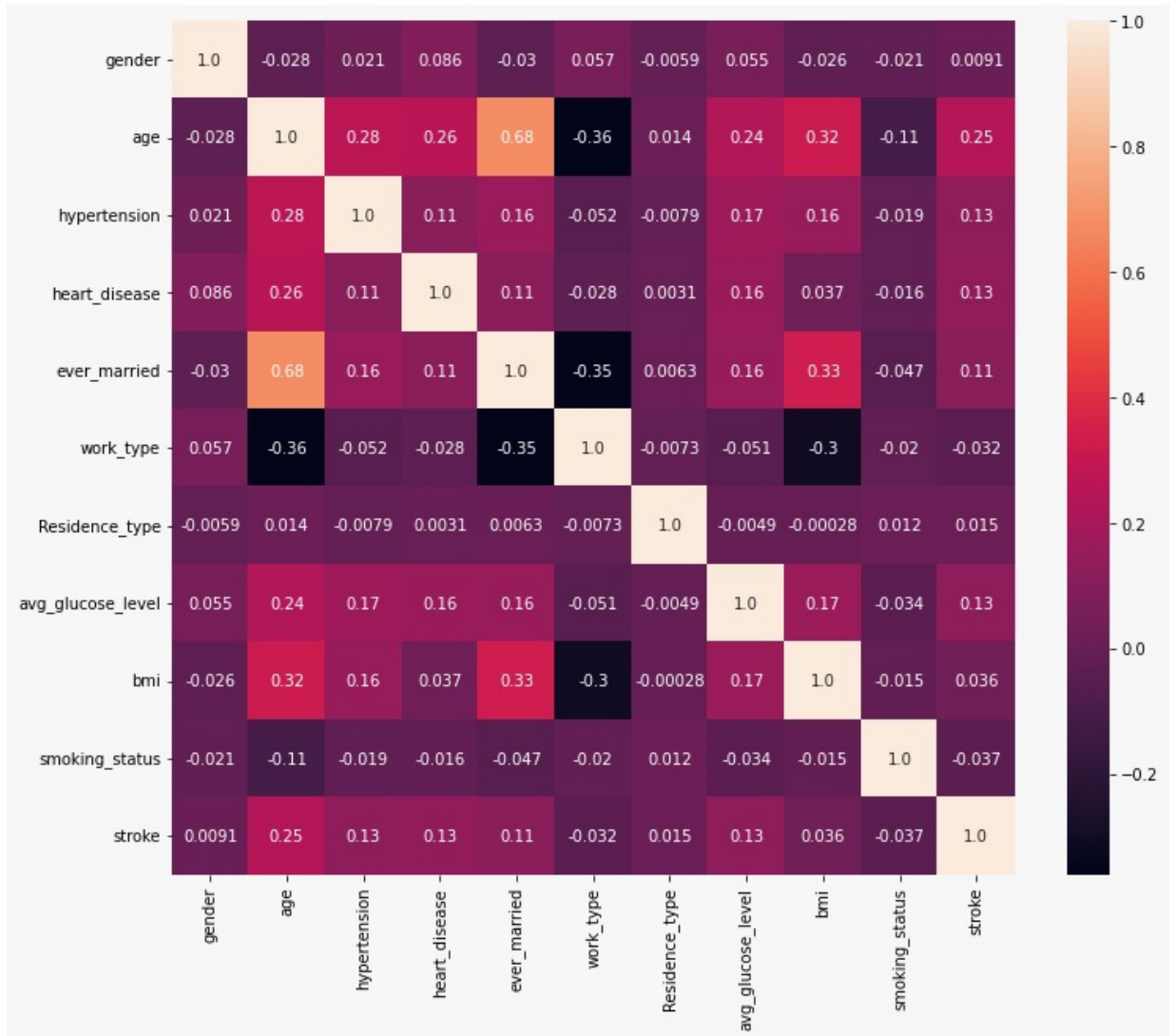
$$Z = \frac{x - \mu}{\sigma}$$

Symbol	Description
Z	z-score
x	Data point in question
μ	Mean
σ	Standard deviation

4.3 Feature Selection and Conversion

It is a way to reduce dimensionality by using only a set of attributes. Any data is not lost in this way because only the redundant attributes are selected and dropped and the relevant attributes which help in the predictive modelling are kept.[1]

The attribute 'id' is not at all important in the predictive model, because it is a unique attribute. The attribute 'age' has the most influence on the target attribute 'stroke', then comes 'heart_disease', 'avg_glucose_level', 'hypertension' and 'bmi' in the decreasing order of influence.



The attributes 'gender', 'ever_married', 'work_type', 'Residence_type', and 'smoking_status' are categorical attributes, so they have to be converted into their equivalent numerical values.

4.3.1 Categorical to Numerical

```
cols=stroke.select_dtypes(include=['object']).columns
ohe=LabelEncoder()
stroke[cols]=stroke[cols].apply(ohe.fit_transform)
stroke.head()
```

gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status
1	67.0	0	1	1	2	1	228.69	36.6	0
0	61.0	0	0	1	3	0	202.21	NaN	1
1	80.0	0	1	1	2	0	105.92	32.5	1
0	49.0	0	0	1	2	1	171.23	34.4	2
0	79.0	1	0	1	3	0	174.12	24.0	1

4.4 Data Sampling and Subsetting

Splitting the dataset into training and testing data is necessary because the model to be created will learn from the training data and then use that knowledge to predict the outcome of the test data. Usually, the training data is much larger than the test data.

4.4.1 Sampling

Sampling is a commonly used approach for selecting a subset of the data objects to be analysed.[1]

Types of sampling are:[8]

1. Simple random sampling: There is an equal probability of selecting any particular item.
2. Stratified sampling: Split the data into several partitions; draw random samples from each partition.
3. Sampling without replacement: As each item is selected, it is removed from the population.
4. Sampling with replacement: Objects are not removed from the population as they are selected for the sample (the same object can be picked up more than once).

4.4.2 TRAIN-TEST SPLIT

```
train_x,test_x,train_y,test_y=train_test_split(stroke[colus],
                                                stroke['stroke'],
                                                random_state=1266,
                                                test_size=0.25)
train_x.shape,test_x.shape,train_y.shape,test_y.shape
((3832, 5), (1278, 5), (3832,), (1278,))
```

Chapter 5

BUILDING MODELS

Training and testing different models on the dataset is very important because different models will handle the data in different ways. The models should also be tinkered with by putting different parameters and thus finding out which gives the optimal result.

5.1 K-NN Model

K-Nearest Neighbour is a simple Machine Learning algorithm based on Supervised Learning technique. The K-NN algorithm assumes the similarity between the new case/data and available cases and puts the new case into the category that is most similar to the available categories.[2]

```
knn=KNeighborsClassifier(n_neighbors=1)
knn.fit(train_x,train_y)
predi=knn.predict(test_x)
accuracy_score(predi,test_y)
```

```
0.6319043693322342
```

5.2 Decision Tree Model

Decision Tree is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. It is based on the Supervised Learning technique.[2]

```
dtr=DecisionTreeClassifier()  
dtr.fit(train_x,train_y)  
ohyeah=dtr.predict(test_x)  
accuracy_score(ohyeah,test_y)  
  
0.7370156636438582
```

5.3 Naïve-Bayes Model

Naive-Bayes algorithm is based on the Bayes theorem. It is a probabilistic classifier, it predicts on the basis of the probability of the object. It is based on the Supervised Learning technique.[2]

```
gnb=GaussianNB()  
gnb.fit(train_x,train_y)  
pred=gnb.predict(test_x)  
accuracy_score(pred,test_y)  
  
0.7790601813685079
```

Chapter 6

MODEL EVALUATION AND RESULTS

6.1 Metrics

Data mining metrics are defined as a set of measurements that helps in determining the efficacy of a Data mining Method / Technique or Algorithm. They are important to help make the right decision, like choosing the right data mining technique or algorithm.[7]

6.1.1 Confusion Matrix

The confusion matrix is a matrix used to determine the performance of the classification models for a given set of test data. It can only be determined if the true values for test data are known.[2]

n = Total Predictions	Actual: No	Actual; Yes
Predicted: No	True Negative	False Positive
Predicted: Yes	False Negative	True Positive

[2]

6.1.2 Accuracy

It defines how often the model predicts the correct output. It can be calculated as the ratio of the number of correct predictions made by the classifier to all number of predictions made by the classifiers.[2]

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN}$$

6.1.3 Precision

It can be defined as the number of correct outputs provided by the model or out of all positive classes that have predicted correctly by the model, how many of them were actually true.[2]

$$Precision = \frac{TP}{TP+FP}$$

6.1.4 Recall

It is defined as the out of total positive classes, how our model predicted correctly. The recall must be as high as possible.[2]

$$Recall = \frac{TP}{TP + FN}$$

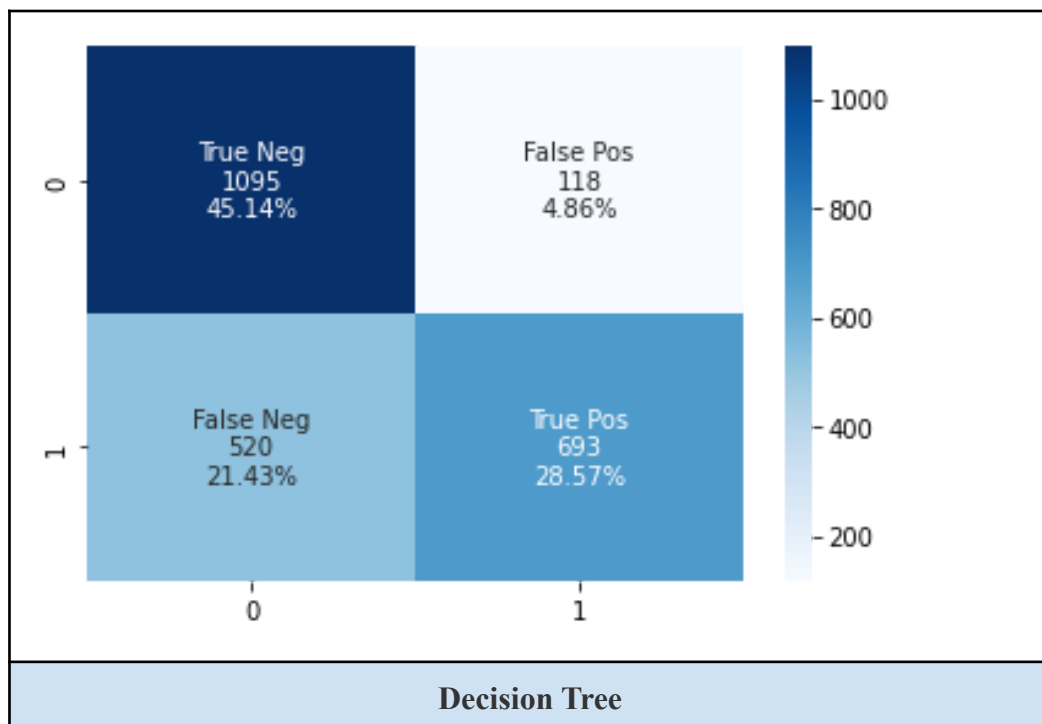
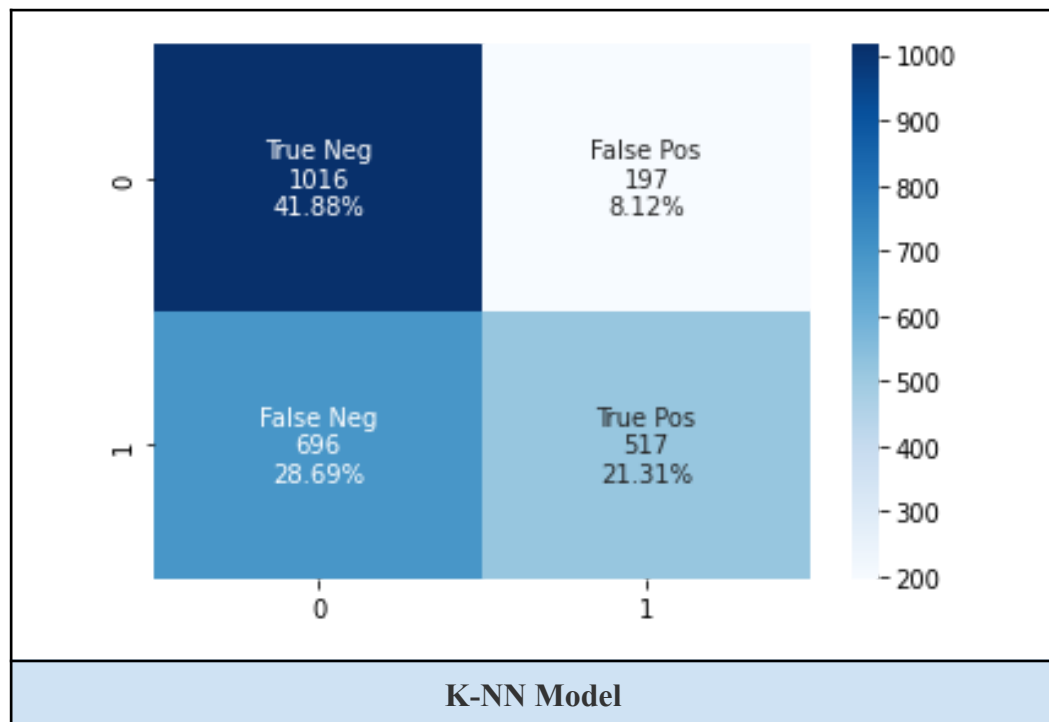
6.1.5 F1-Score

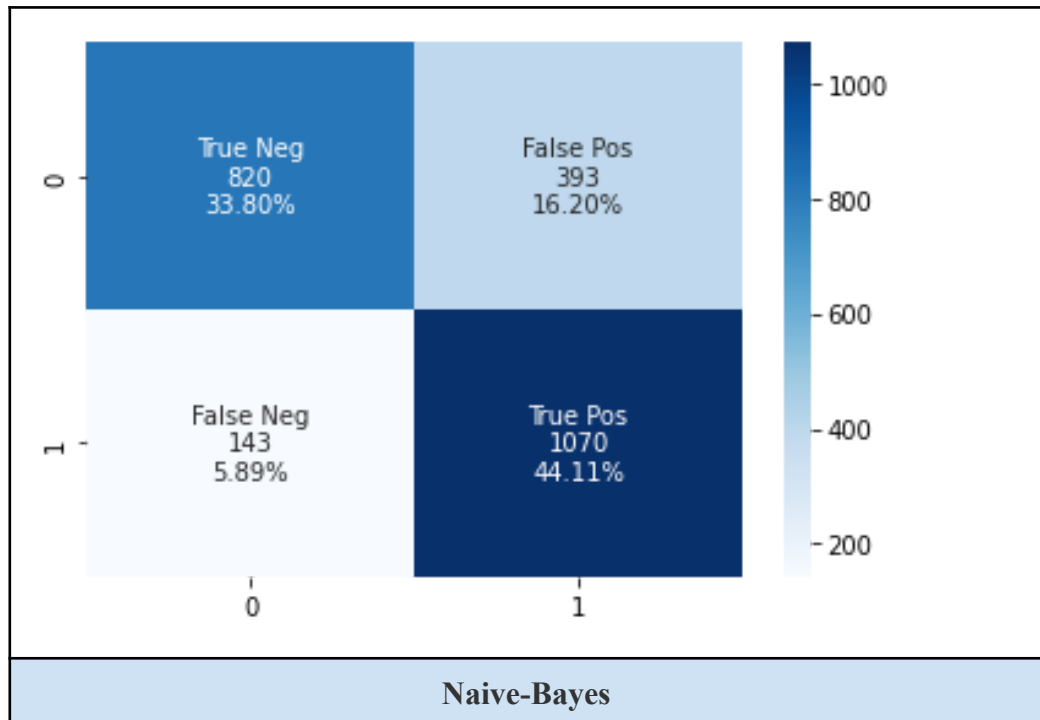
If two models have low precision and high recall or vice versa, it is difficult to compare these models. So, for this purpose, we can use F1-score. F1-Score helps us to evaluate the recall and precision at the same time. The F-score is maximum if the recall is equal to the precision.[2]

$$F1 - Score = \frac{2*Recall*Precision}{Recall + Precision}$$

6.2. Experimental Results and Comparison

■ Confusion Matrix





■ Accuracy

Models	Accuracy (%)
K-NN	63.1904369332
Decision Tree	73.7015663643
Naive-Bayes	77.9060181368

■ Precision

Models	Precision (%)
K-NN	72.4089635854
Decision Tree	85.4500616522
Naive-Bayes	73.1373889268

■ Recall

Models	Recall (%)
K-NN	42.6215993404
Decision Tree	57.1310799670
Naive-Bayes	88.2110469909

■ F1-Score

Models	F1-score (%)
K-NN	53.6585365853
Decision Tree	68.4782608695
Naive-Bayes	79.9701046337

Chapter 7

INFERENCES AND CONCLUSION

Based on the findings in Chapter 6 the Naive-Bayes model has the highest accuracy among the other models used. Also in the confusion matrix the True negatives and the True positives are highest for the Naive Bayes model.

The attributes 'age', 'hypertension', 'heart_disease', 'ever_married', 'avg_glucose_level' have the highest influence on determining whether the patient will suffer from stroke or not, as written in the order.

References

1. Tan, P. N., Steinbach, M., & Kumar, V. (2016). *Introduction to data mining*. Pearson Education India.
2. Website: <https://www.javatpoint.com>
3. Website: <https://www.geeksforgeeks.org/>
4. Website: <https://towardsdatascience.com/7-ways-to-handle-missing-values-in-machine-learning-1a6326adf79e#:~:text=Missing%20values%20can%20be%20handled,null%20can%20also%20be%20dropped.>
5. Han, J., Pei, J., & Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.
6. Kesavaraj, G., & Sukumaran, S. (2013, July). A study on classification techniques in data mining. In *2013 fourth international conference on computing, communications and networking technologies (ICCCNT)* (pp. 1-7). IEEE.
7. Website: himadri.cmsdu.org/documents/DataMining_Metrics.pdf
8. Website : <https://online.datasciencedojo.com/course/Data-Mining/data-preprocessing-transformation/data-sampling-types>
9. Website : [kaggle.com](https://www.kaggle.com/)
Owner of dataset: [fedesoriano](#)

Appendix

.py file

```
"""Stroke_Prediction.ipynb
```

Automatically generated by Colaboratory.

Original file is located at

```
https://colab.research.google.com/drive/1FD1Hd4ykyYyFniJZXnDH3uytmNcErdhr
"""
```

```
from google.colab import files
uploaded=files.upload()
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from imblearn.pipeline import Pipeline
from imblearn.over_sampling import SMOTE
from sklearn.feature_selection import SelectKBest,f_classif
from sklearn.metrics import
accuracy_score,f1_score,classification_report,precision_score,recall_score,confusion_matrix
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
```

```
stroke=pd.read_excel('/content/healthcare-dataset-stroke-data.xlsx')
```

```
stroke.tail()
```

```
len(stroke)
```

```

stroke.columns

stroke['smoking_status'].unique()

stroke['smoking_status'].replace('Unknown', np.nan, inplace=True)

stroke['smoking_status']

stroke.columns

stroke.drop(['id'], axis=1, inplace=True)

stroke.shape[1]

stroke.isnull().sum()

stroke.dtypes

stroke['gender'].value_counts()

"""#Visualisation"""

sns.countplot(data=stroke, x='gender', hue='stroke')

sns.displot(stroke['age'])

sns.boxplot(data=stroke, x='stroke', y='age')

sns.countplot(data=stroke, x='hypertension', hue='stroke')

sns.countplot(data=stroke, x='heart_disease', hue='stroke')

sns.countplot(data=stroke, x='ever_married', hue='stroke')

sns.countplot(data=stroke, x='work_type', hue='stroke')

sns.countplot(data=stroke, x='Residence_type', hue='stroke')

sns.displot(stroke['avg_glucose_level'])

sns.boxplot(data=stroke, x='stroke', y='avg_glucose_level')

```



```

sns.displot(stroke['bmi'])

sns.boxplot(data=stroke,x='stroke',y='bmi')

sns.countplot(data=stroke,x='smoking_status',hue='stroke')

fig=plt.figure(figsize=(6,6))

plt.title('Gender')
stroke['gender'].value_counts().plot.bar(color='red')

plt.title('Hypertension')
heightertension=[stroke['hypertension'].value_counts()[0],stroke['hypertension'].value_counts()[1]]
bhottension=['Negative','Positive']
plt.xlabel('Hypertension status')
plt.ylabel("No. of individuals")
plt.bar(bhottension,heightertension,color='blue')

plt.title('Heart disease')
heightertension=[stroke['heart_disease'].value_counts()[0],stroke['heart_disease'].value_counts()[1]]
bhottension=['Negative','Positive']
plt.xlabel('Heart disease status')
plt.ylabel("No. of individuals")
plt.bar(bhottension,heightertension,color='orange')

plt.title('Marital Status')
heightertension=[stroke['ever_married'].value_counts()[0],stroke['ever_married'].value_counts()[1]]
bhottension=['Married','Unmarried']
plt.ylabel("No. of individuals")
plt.bar(bhottension,heightertension,color='magenta')

plt.title('Work Type')
plt.ylabel("No. of individuals")
stroke['work_type'].value_counts().plot.bar(color='brown')

plt.title('Residence Type')
plt.ylabel("No. of individuals")
stroke['Residence_type'].value_counts().plot.bar(color='green')

```

```

plt.title('Smoking Status')
plt.ylabel("No. of individuals")
stroke['smoking_status'].value_counts().plot.bar(color='grey')

plt.title('Stroke Status')
plt.ylabel("No. of individuals")
heightertension=[stroke['stroke'].value_counts()[0],stroke['stroke'].value_counts()[1]]
bhottension=['Did not happen','Happened']
plt.bar(bhottension,heightertension,color='teal')

sns.boxplot(x='age',data=stroke,color='#6bab0c')
plt.title('Age distribution')

sns.boxplot(x='avg_glucose_level',data=stroke,color='#f53f07')
plt.title('Average Glucose level distribution')

sns.boxplot(x='bmi',data=stroke,color='#8d11c2')
plt.title('BMI distribution')

"""#Predicting values

##Encoding
"""

cols=stroke.select_dtypes(include=['object']).columns
ohe=LabelEncoder()
stroke[cols]=stroke[cols].apply(ohe.fit_transform)
stroke.head()

"""##Correlation heatmap"""

plt.figure(figsize=(12,10))
sns.heatmap(stroke.corr(),annot=True,fmt='.2')

"""##Handling null values"""

stroke['bmi'].fillna(stroke.bmi.median(),inplace=True)

stroke['smoking_status'].fillna(stroke['smoking_status'].mode()[0],inplace=True)

```

```

stroke.isnull().sum()

"""##Filtering attributes which affects stroke most"""

skb=SelectKBest(score_func=f_classif,k=5)
fits=skb.fit(stroke.drop('stroke',axis=1),stroke['stroke'])
scpd=pd.DataFrame(fits.scores_)
col=pd.DataFrame(stroke.drop('stroke',axis=1).columns)
fsc=pd.concat([col,scpd],axis=1)
fsc.columns=['Attribute','Score']
fsc.sort_values(by='Score',ascending=False)

colus=fsc[fsc['Score']>50].Attribute
colus

"""##Splitting dataset"""

train_x,test_x,train_y,test_y=train_test_split(stroke[colus]
,

stroke['stroke'],

random_state=1266,

test_size=0.25)
train_x.shape,test_x.shape,train_y.shape,test_y.shape

"""##Sampling dataset"""

smote=SMOTE()
train_x,train_y=smote.fit_resample(train_x,train_y)
test_x,test_y=smote.fit_resample(test_x,test_y)
train_x.shape,test_x.shape,train_y.shape,test_y.shape

"""##Naive-Bayes Classifier"""

gnb=GaussianNB()
gnb.fit(train_x,train_y)
pred=gnb.predict(test_x)
accuracy_score(pred,test_y)

cf_nb=confusion_matrix(test_y,pred)

group_names = ['True Neg','False Pos','False Neg','True

```

```

Pos']
group_counts = ["{0:0.0f}".format(value) for value in
                 cf_nb.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in
                     cf_nb.flatten()/np.sum(cf_nb)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(cf_nb, annot=labels, fmt='', cmap='Blues')

precision_score(test_y,pred)

recall_score(test_y,pred)

f1_score(test_y,pred)

"""##K-NN Classifier"""

knn=KNeighborsClassifier(n_neighbors=1)
knn.fit(train_x,train_y)
predi=knn.predict(test_x)
accuracy_score(predi,test_y)

cf_knn=confusion_matrix(test_y,predi)

group_names = ['True Neg','False Pos','False Neg','True
Pos']
group_counts = ["{0:0.0f}".format(value) for value in
                 cf_knn.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in
                     cf_knn.flatten()/np.sum(cf_knn)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(cf_knn, annot=labels, fmt='', cmap='Blues')

precision_score(test_y,predi)

recall_score(test_y,predi)

f1_score(test_y,predi)

"""##Decision Tree Classifier"""

```

```

dtr=DecisionTreeClassifier()
dtr.fit(train_x,train_y)
ohyeah=dtr.predict(test_x)
accuracy_score(ohyeah,test_y)

cf_dtr=confusion_matrix(test_y,ohyeah)

group_names = ['True Neg','False Pos','False Neg','True
Pos']
group_counts = ["{0:0.0f}".format(value) for value in
                 cf_dtr.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in
                     cf_dtr.flatten()/np.sum(cf_dtr)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(cf_dtr, annot=labels, fmt='', cmap='Blues')

precision_score(test_y,ohyeah)

recall_score(test_y,ohyeah)

f1_score(test_y,ohyeah)

```