BRAIN STROKE PREDICTION

A MINI PROJECT REPORT

18CSC305J - ARTIFICIAL INTELLIGENCE

Submitted by

Ritesh Mishra(RA2111027010014)
K.Akash (RA2111027010015)
Mudaliar Saurabh Ravi(RA2111027010016)
Arunima Mishra(RA2111027010017)

Under the guidance of

Dr. Arthy M

Assistant Professor, Department of Data Science And Business Systems

in partial fulfillment for the award of the

degreeof

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled **"BRAIN STROKE PREDICTION"** is the bona fide work of **Ritesh Mishra(RA2111027010014) ,K.Akash (RA2111027010015) , Mudaliar Saurabh Ravi(RA2111027010016) ,Arunima Mishra(RA2111027010017)** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. Arthy M
Assistant Professor
Department of Data Science and Business Systems

Dr.M Lakshmi
Professor and Head
Department of Data Science and Business Systems

Table of Contents

CHAPTER 1

Introduction

Our project focuses on the domain of brain stroke prediction, utilizing data analysis techniques and machine learning algorithms. Brain strokes are a critical medical condition with potentially severe consequences. Early prediction and detection can significantly improve patient outcomes and reduce the risk of debilitating effects.

Existing Problem:

The existing problem in this domain lies in the challenges of accurately predicting and diagnosing brain strokes before they occur. Traditional medical approaches often rely on identifying risk factors and symptoms, which may not always be conclusive or timely. Additionally, the sheer volume and complexity of data involved in stroke prediction pose significant challenges for manual analysis and interpretation.

Problem Statement:

Our project aims to address these challenges by developing a robust predictive model for brain stroke occurrence. This model will leverage advanced data analysis techniques and machine learning algorithms to identify patterns and risk factors indicative of an impending stroke.

Project Objective:

The primary objective of our project is to create a predictive model capable of accurately forecasting the likelihood of a brain stroke for individuals based on their medical history, lifestyle factors, and other relevant data points. By doing so, we aim to facilitate early intervention and preventive measures, ultimately reducing the incidence and severity of strokes.

Motivation:

The motivation behind our project stems from the urgent need to improve stroke prediction and prevention strategies. Brain strokes represent a significant public health concern worldwide, causing immense suffering and economic burden. By developing an effective predictive model, we can empower healthcare professionals to intervene proactively and potentially save lives.

Proposed Solution:

Our proposed solution involves leveraging data analysis techniques and machine learning algorithms to analyze and interpret relevant data sources for stroke prediction. Specifically, we plan to utilize algorithms such as linear regression, decision trees, random forests, and XGBoost to build a comprehensive predictive model. By combining these algorithms, we aim to capture complex relationships and patterns within the data that may indicate stroke risk.

Scope:

The scope of our project encompasses several key aspects:

- Data Collection: Gathering comprehensive datasets containing medical history, lifestyle factors, demographic information, and other relevant variables.
- Data Preprocessing: Cleaning and preparing the data for analysis, including handling missing values, encoding categorical variables, and normalizing features.
- Feature Selection: Identifying the most relevant features that contribute to stroke prediction using techniques such as correlation analysis and feature importance ranking.
- Model Development: Implementing and fine-tuning machine learning algorithms including linear regression, decision trees, random forests, and XGBoost to build an accurate predictive model.

CHAPTER 2

Literature Survey

- Earlier studies in the literature have examined various aspects of stroke prediction. Jeena et al. [3] investigated several risk factors to understand their association with stroke occurrence. Employing a regression-based methodology, they analyzed the impact of each factor on the likelihood of stroke. Adam et al. [8] explored stroke prediction using the decision tree method and the k-nearest neighbor algorithm. Their research revealed the decision tree method as more effective in predicting stroke occurrences.

- Singh and Choudhary [9] utilized the Cardiovascular Health Study (CHS) dataset to predict strokes in individuals. Emon et al. [10] implemented learning-based classification algorithms, including XGBoost, Random Forest, Naive Bayes, Logistic Regression, and Decision Tree, on a dataset obtained from Kaggle. Kansadub et al. [11] examined stroke likelihood using decision trees, neural networks, and Naive Bayes analysis, evaluating the precision and AUC of their models.

- Tazin et al. [12] proposed early-stage stroke prediction using Logistic Regression, Decision Tree Classification, Random Forest Classification, and Voting Classifier, with Random Forest performing the best. Chetan Sharma et al. [13] suggested using the random forest algorithm for short-term stroke prediction. They also explored a feed-forward multi-layer artificial neural network-based deep learning model for stroke prediction.

- Similarly, research on developing intelligent systems for stroke prediction using patient records was explored in [14]. Hung et al. [15] compared machine learning and deep learning models for constructing stroke prediction models from electronic medical claims databases. Fang et al. [16] applied three deep learning approaches (CNN, LSTM, Resnet) and compared them with machine learning algorithms (Deep Forest, Random

Forest, Support Vector Machine, etc.) for clinical prediction. Mahesh et al. [12] utilized various deep learning algorithms such as CNN, Densenet, and VGG16 to evaluate performance metrics for automatic brain stroke prediction.

TABLE III: COMPARISON AMONG THE MACHINE LEARNING APPROACHES

Algorithm	Accuracy	Precision	Recall	F1-score	AUC
LR	0.71	0.69	0.73	0.71	0.79
DT	0.98	1.00	0.95	0.97	0.98
RF	0.99	1.00	0.98	0.99	1.00
KNN	0.96	1.00	0.92	0.96	0.98
SVM	0.82	0.84	0.77	0.81	-
GaussianNB	0.70	0.68	0.73	0.70	0.78
BernoulliNB	0.67	0.69	0.59	0.63	0.71
XGBoost	0.97	1.00	0.93	0.97	0.98
AdaBoost	0.78	0.75	0.82	0.78	0.76
LGBM	0.95	1.00	0.90	0.95	0.96

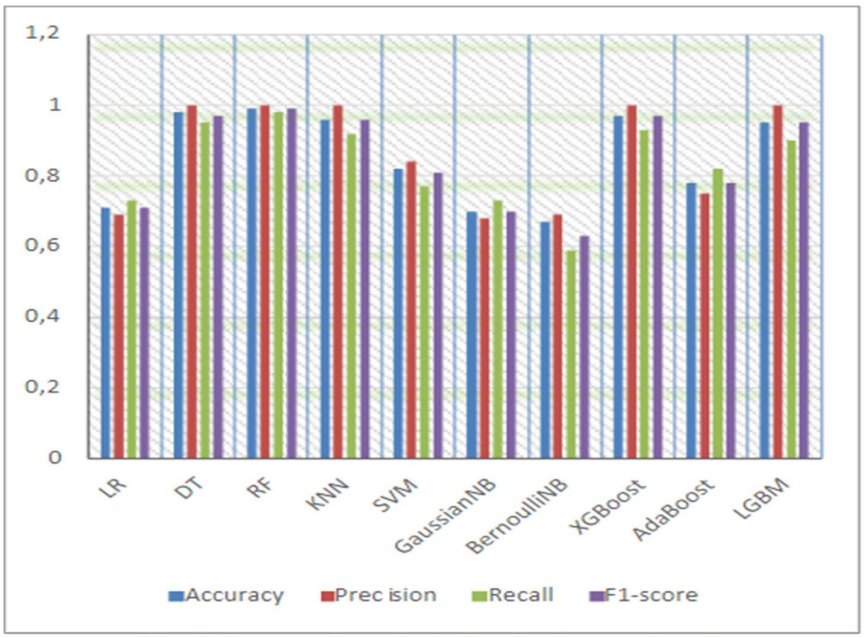


Fig. 6. A comparison chart of evaluation metrics of machine learning algorithms.

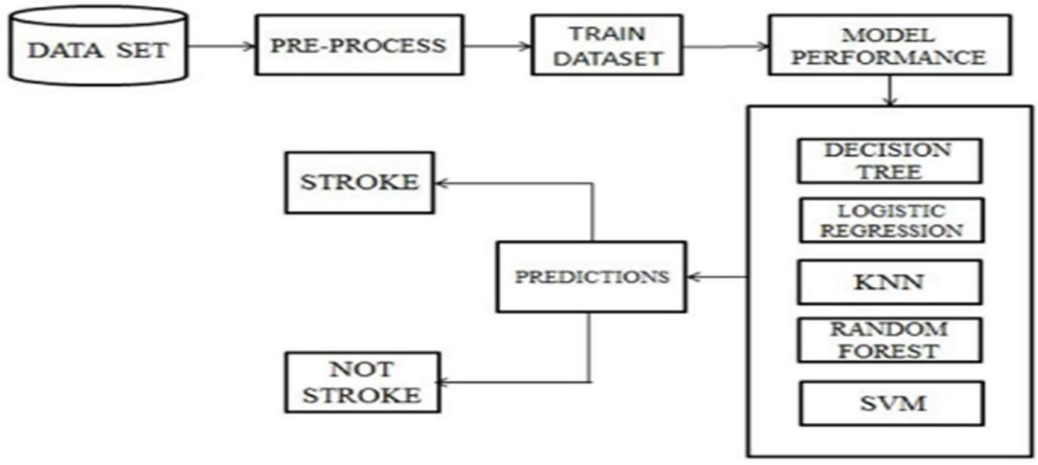
- **Results** : Random Forest outperforms other classifiers in terms of accuracy (0.99) which is calculated using equation (1). It shows the

highest accuracy among the machine learning algorithms, whereas 3-layer ANN demonstrated promising results among deep learning techniques. The comparison among the machine learning approaches is shown in Table III and a chart is shown (in Fig. 6) to represent the superiority of RF method over the other ML algorithms using the performance metrics which were calculated using equation (1-4). The area under roc curve for Random Forest method is given in Fig. 7. The performance metrics for the ANN approaches are described in Table IV. The Area under roc curve for the 4-layer ANN is shown in Figure 8. The comparison between the Random Forest and 3-layer ANN method is depicted in the bar chart of Fig. 9. From the comparison, it is clear that RF algorithm outperforms all the Boosting algorithms and deep neural approaches in all aspects

- **Conclusion**: Stroke is a potentially fatal medical condition that needs to be treated right away to prevent future consequences. The creation of a machine learning (ML) and Deep Learning model could help with stroke early diagnosis and subsequent reduction of its severe consequences. This study examines how well different machine learning (ML) as well as Boosting algorithms predict stroke based on various biological factors. With a classification accuracy of 99%, and AUC of 1, random forest classification exceeds the other investigated techniques. According to the study, the random forest method performs better than other methods when forecasting brain strokes using cross-validation measures.

CHAPTER 3

System Architecture and Design



SYSTEM ARCHIETECTURE

Data Set

A data set is a collection of data. In the case of tabular data, a data set corresponds to one or more database tables, where every column of a table represents a particular variable, and each row corresponds to a given record of the data set In question. The data set lists values for each of the variables, such as gender, age and bmi of the person. Data sets can also consist of a collection of documents or files. The stroke prediction dataset was used to perform the study. There were 5110rows and 12 columns in this dataset. The value of the output column stroke is either 1 or0. The number 0 indicates that no stroke risk was identified, while the value 1 indicates that a stroke risk was detected. The probability of 0 in the output column (stroke)exceeds the possibility of 1 in the same column in this dataset. 249 rows alone in the stroke column have the value 1, whereas 4861 rows have the value 0. To improve accuracy, data pre-processing is used to balance the data. Figure 3.2

shows the total number of stroke and non-stroke records in the output column before preprocessing

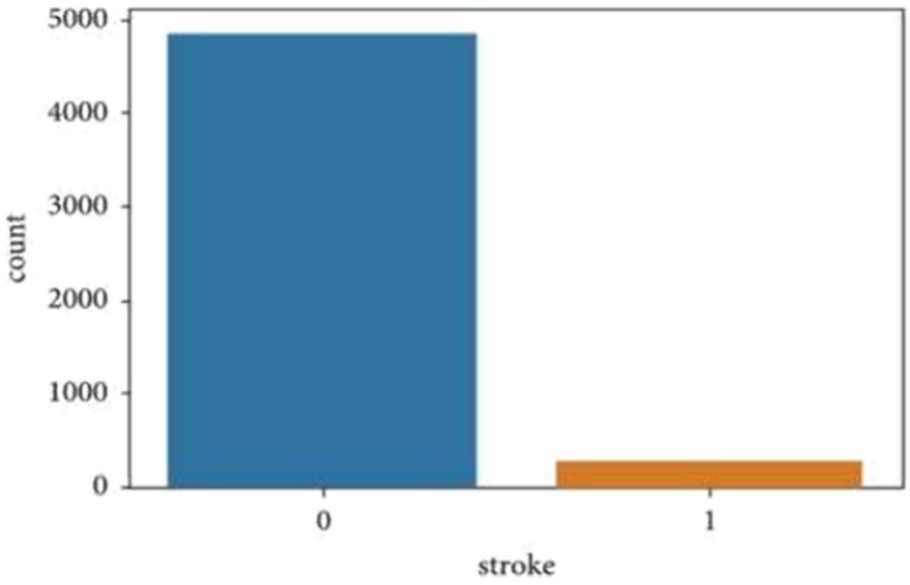


Fig 3.2 Total number of stroke and normal data

Pre- Processing

Before building a model, data pre-processing is required to remove unwanted noise andoutliers from the dataset that could lead the model to depart from its intended training.This stage addresses everything that prevents the model from functioning moreefficiently. Following the collection of the relevant dataset, the data must be cleaned and prepared for model development. As stated before, the dataset used has twelve characteristics. To begin with, the column id is omitted since its presence has no bearing on model construction. The dataset is then inspected for null values and filled if any are detected. The null values in the column BMI are filled using the data column's mean in this case.Label encoding converts the dataset's string literals to integer values that the computercan comprehend. As the computer is frequently trained on numbers, the strings must be converted to integers. The gathered dataset has five columns of the data type string. All strings are encoded during label encoding, and the whole dataset is transformed into a collection of numbers. The dataset used for stroke prediction is very imbalanced. The dataset

has a total of 5110 rows, with 249 rows indicating the possibility of a stroke and 4861 rows confirming the lack of a stroke. While using such data to train a machine-level model may result in accuracy, other accuracy measures such as precision and recallare inadequate. If such an unbalanced data is not dealt with properly, the findings will beinaccurate, and the forecast will be ineffective. As a result, to obtain an efficient model,this unbalanced data must be dealt with first. The SMOTE technique was employed forthis purpose. Figure 3.3 depicts the dataset's balance output column.

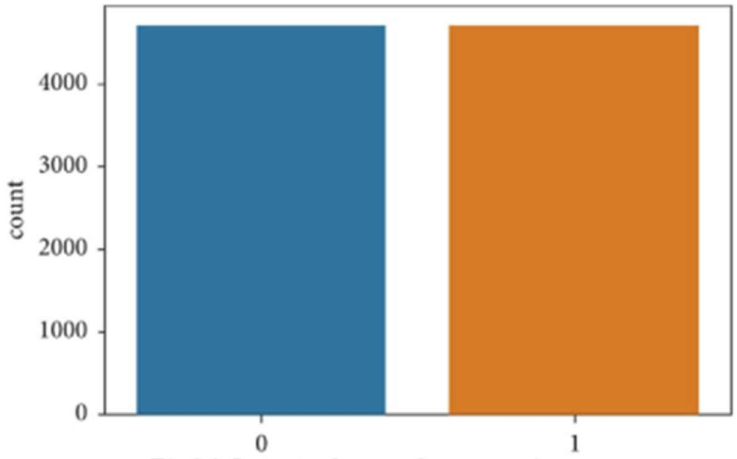


Fig 3.3 Output columns after processing

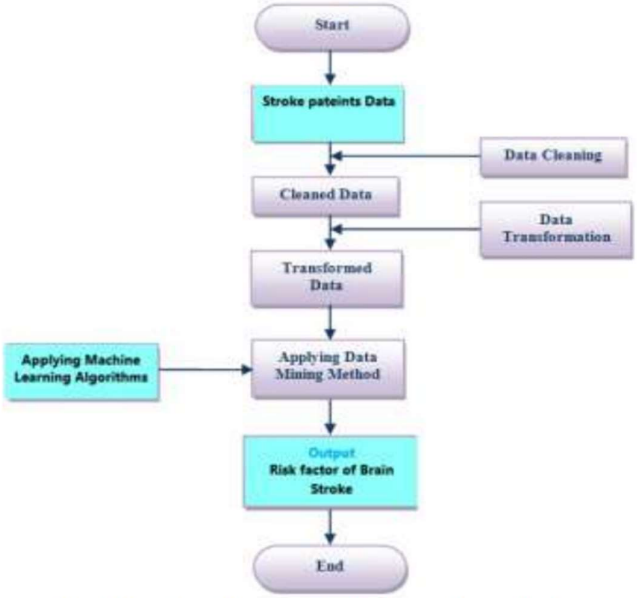


Fig-2:Dataflow Diagram for Brain Stroke prediction

Train Dataset

- Training data (or a training dataset) is the initial data used to train machine learningmodels

- Training datasets are fed to machine learning algorithms to teach them how to makepredictions or perform a desired task

CHAPTER 4

Methodology

The methodology for predicting brain strokes using machine learning involves several steps from data acquisition to model evaluation. This section outlines the use of various machine learning models, including K-Nearest Neighbors (KNN), Linear Regression, XGBoost, Random Forest, and Decision Trees, detailing each phase of the process.

4.1 Dataset Description:

Our ML model was based on a dataset from Kaggle. From this dataset, we focused on participants who are over 18 years old. The number of participants was 3254, and all of the attributes (10 as input to ML models and 1 for target class) are described as follows:

- **Age** (years): This feature refers to the age of the participants who are over 18 years old.
- **Gender**: This feature refers to the participant's gender. The number of men is 1260, whereas the number of women is 1994.
- **Hypertension**: This feature refers to whether this participant is hypertensive or not. The percentage of participants who have hypertension is 12.54%.
- **Heart_disease**: This feature refers to whether this participant suffers from heart disease or not. The percentage of participants suffering from heart disease is 6.33%.
- **Ever married**: This feature represents the marital status of the participants, 79.84% of whom are married.
- **Work type**: This feature represents the participant's work status and has 4 categories (private 65.02%, self-employed 19.21%, govt_job 15.67% and never_worked 0.1%).
- **Residence type**: This feature represents the participant's living status and has 2 categories (urban 51.14%, rural 48.86%).
- **Avg glucose level** (mg/dL): This feature captures the participant's average glucose level.

- **BMI** (Kg/m^2): This feature captures the body mass index of the participants.
- **Smoking Status**: This feature captures the participant's smoking status and has 3 categories (smoke 22.37%, never smoked 52.64% and formerly smoked 24.99%).
- **Stroke**: This feature represents if the participant previously had a stroke or not. The percentage of participants who have suffered a stroke is 5.53%.

Most features are nominal except for the age, average glucose level and BMI, which are numerical.

4.2 Data Collection

The initial step involves gathering data that can be used to predict strokes. This data typically includes patient demographics, medical history, lifestyle factors, and other relevant clinical parameters known to influence stroke risk.

4.3 Data Preprocessing

Data preprocessing is crucial to prepare the raw data for modeling. This phase includes:

- Cleaning: Removing or imputing missing values, eliminating duplicate entries.

- Transformation: Normalizing or scaling numerical data to bring everything to a comparable range.

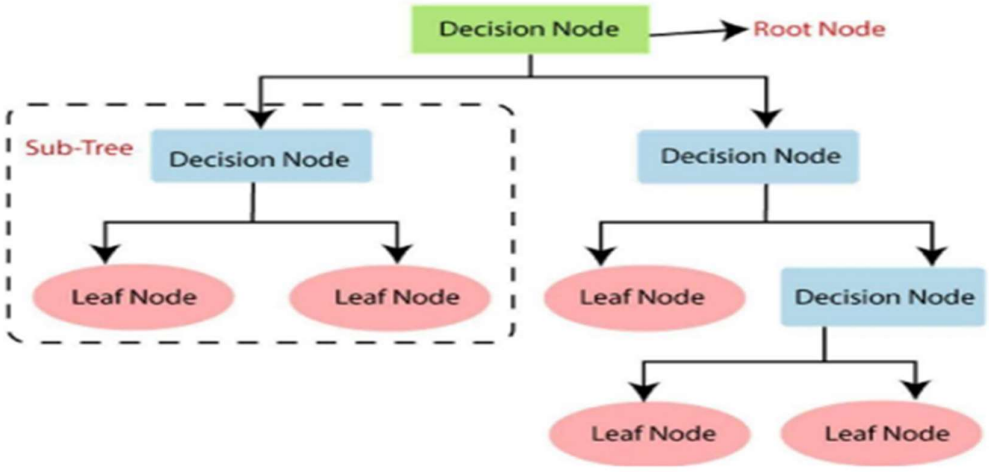
- Feature Selection:Identifying the most relevant features that contribute to stroke prediction based on statistical tests and domain knowledge.

- Data Splitting: Dividing the data into training and testing sets to ensure the model can be trained and tested on different datasets.

4.4 ALGORITHMS

4.4.1 Decision Tree

- Decision Tree is a Supervised learning technique that can be used for both classificationand Regression problems, but mostly it is preferred 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
- In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node.
- Decision nodes are used to make any decision and have multiple branches, whereas Leafnodes are the output of those decisions and do not contain any further branches
- The decisions or the test are performed on the basis of features of the given dataset.
- It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- A decision tree simply asks a question, and based on the answer (Yes/No), it further splitthe tree into subtrees.



4.4.2 Random Forest

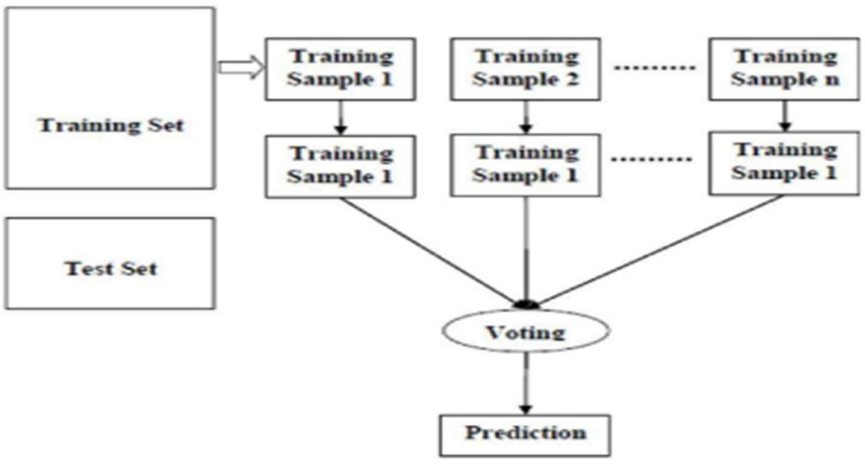
Random forest is a supervised learning algorithm which is used for both classification as well as regression. But however, it is mainly used for classification problems. As we know that a forest is made up of trees and more trees means more robust forest. Similarly, random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result. Working of Random Forest Algorithm We can understand the working of Random Forest algorithm with the help of following steps:

1. First, start with the selection of random samples from a given dataset.

2. Next, this algorithm will construct a decision tree for every sample. Then it will get the prediction result from every decision tree.

3. In this step, voting will be performed for every predicted result.

4. At last, select the most voted prediction result as the final prediction result.



4.4.3 K-Nearest Neighbour (KNN)

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.
- It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

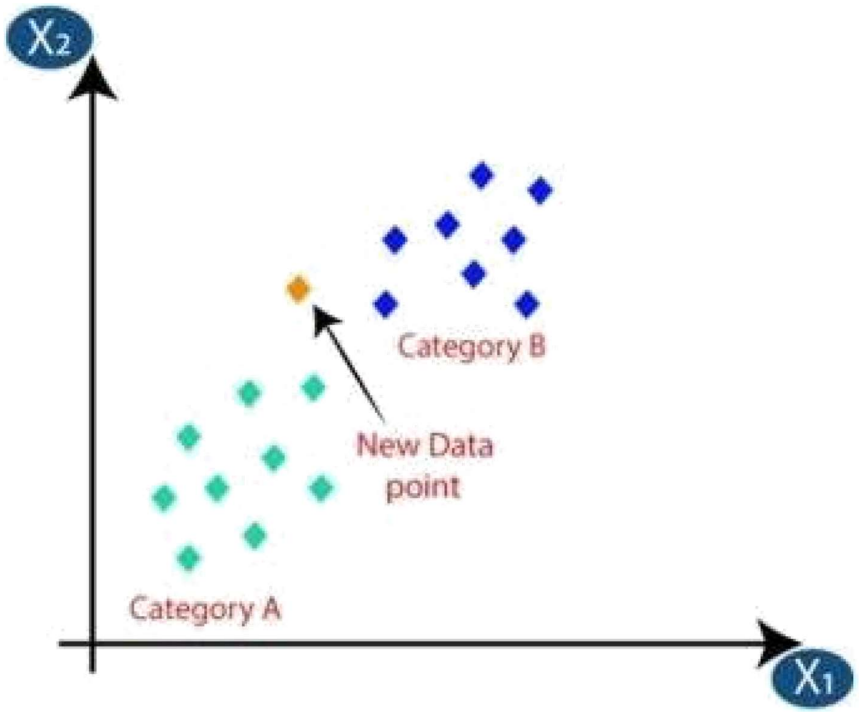
The K-NN working can be explained on the basis of the below algorithm:

1. Select the number K of the neighbours
2. Calculate the Euclidean distance of K number of neighbors
3. Take the K nearest neighbors as per the calculated Euclidean distance.
4. Among these k neighbors, count the number of the data points in each category.

5. Assign the new data points to that category for which the number of the neighbor is maximum.
6. Our model is ready.

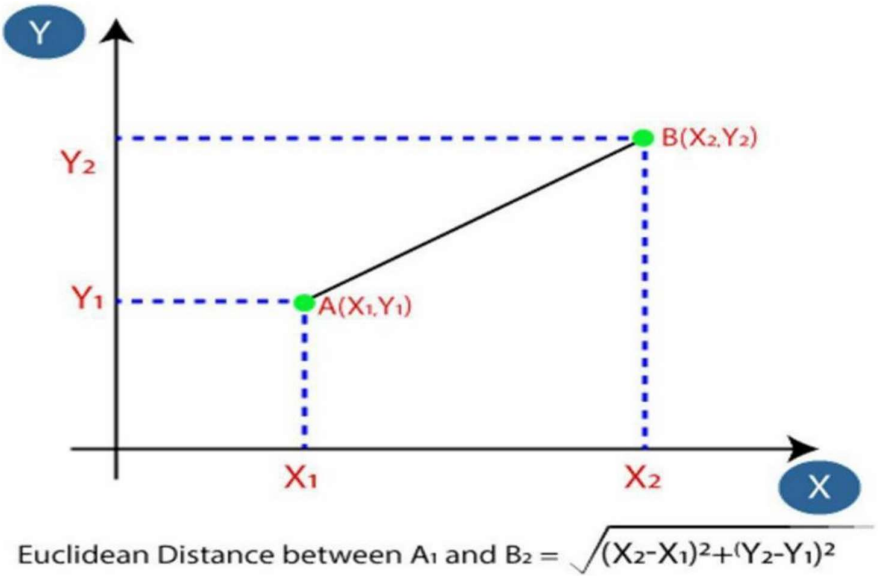
Suppose we have a new data point and we need to put it in the required category.

Consider the below image



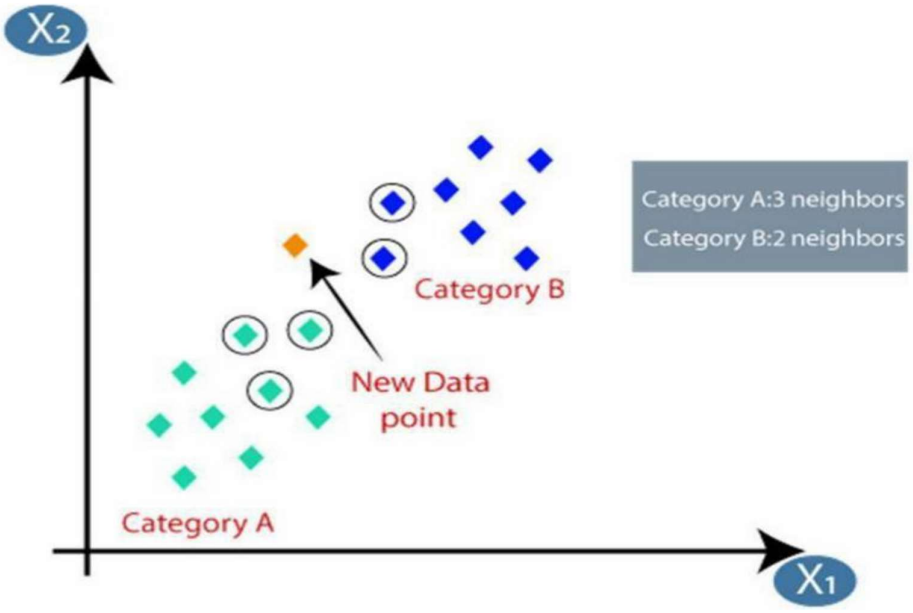
Example of KNN

- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the Euclidean distance between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



Finding Euclidean distance

By calculating the Euclidean distance, we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



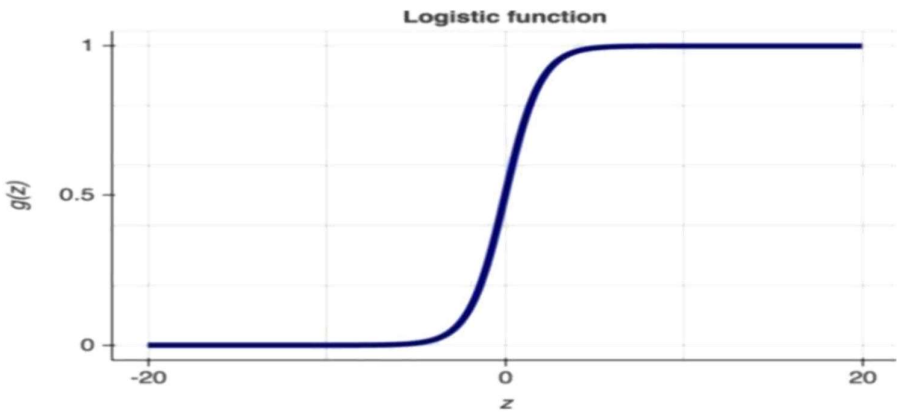
Finding near neighbors

4.4.4 Logistic regression

Logistic regression is a Machine Learning classification algorithm that is used to predictthe probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.). In other words, the logistic regression model predicts P(Y=1) as a function of X.

Step 1: Logistic regression hypothesis

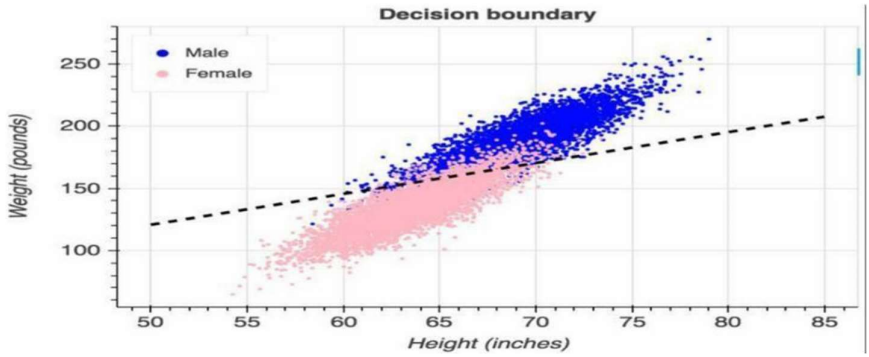
The logistic regression classifier can be derived by analogy to the logistic *regression* the function **g(z)** is the logistic function also known as the *sigmoid function*. The logistic function has asymptotes at 0 and 1, and it crosses the y-axis at 0.5.



Logistic regression hypothesis

Step 2: Logistic regression decision boundary

Since our data set has two features: height and weight, the logistic regression hypothesis is the following:



Logistic regression decision boundary

Steps for executing the Project

1. Install the required packages
2. Load the datasets.
3. Pre-process the data.
4. Split the dataset into train and test.
5. Use the train dataset to train the ml models.
6. Use the test data to test the model prediction and accuracy generation.

4.4.5 XGBoost (eXtreme Gradient Boosting)

Introduction:

XGBoost is a powerful and scalable machine learning algorithm known for its efficiency and effectiveness in predictive modeling tasks. It belongs to the family of gradient boosting algorithms and is widely used in both classification and regression problems.

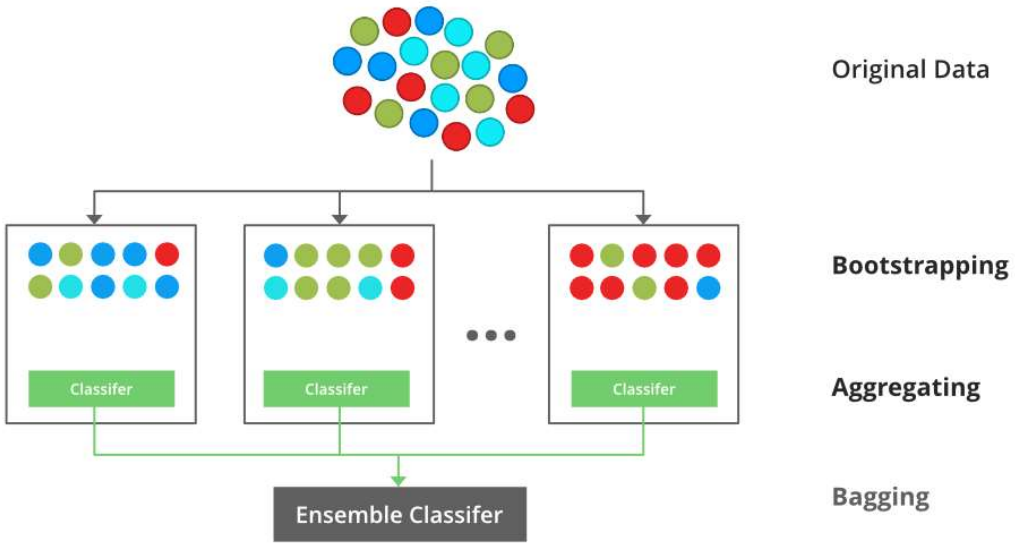
How XGBoost Works:

Gradient Boosting Framework: XGBoost is based on the gradient boosting framework, which builds a strong predictive model by combining multiple weak learners (typically decision trees) sequentially.

Objective Function: XGBoost minimizes a predefined loss function by adding new weak learners to the ensemble that reduces the overall error.

Regularization: Incorporates regularization terms in the objective function to control model complexity and prevent overfitting.

Parallel Processing: Utilizes parallel computing techniques to accelerate training and prediction, making it highly scalable.



Key Concepts:

Weak Learners: Typically decision trees, known as base learners, are used in XGBoost.

Boosting: Weak learners are added sequentially to correct errors made by previous models.

Shrinkage (Learning Rate): Controls the contribution of each new tree added to the ensemble, preventing overfitting.

Feature Importance: XGBoost provides a mechanism to quantify the importance of each feature in the model.

Formulas Used in XGBoost:

Objective Function: Obj=$\sum i=1n$loss(yi,$y\hat{}i$)+$\sum k=1K\Omega(fk)$

Loss Function: Depends on the specific task (e.g., logistic loss for binary classification, squared error for regression)

Regularization Term: $\Omega(f)=\gamma T+12\lambda\sum j=1Twj2$

Conclusion:

XGBoost is a versatile and powerful algorithm suitable for a wide range of machine learning tasks. Its efficiency, scalability, and accuracy make it a popular choice among data scientists and machine learning practitioners.

CHAPTER 5

HARDWARE DESCRIPTION

5.1 Processor

Fig 5.1 Processor

For the most part, you'll get faster CPU performance from the Core i5 parts over Core i3. Some Core i5 processors are dual-core and some are quad-core. Most of the time, a true quad- core CPU will perform better than a dual-core processor, especially on multimedia tasks like video transcoding or photo editing. All Core i3 processor are dual core. Occasionally, you'll find an older Ivy Bridge processor like the Intel Core i33130M in a system that's the same price as system with a newer Haswell CPU like the Intel Core i3-4012Y.

5.2 Hard Disk

Fig 5.2 Hard Disk

A computer's hard drive is a device consisting of several hard disks, read/write heads, a drive motor to spin the disks, and a small amount of circuitry, all sealed in a metal case to protect the disks from dust. In addition to referring to the disks themselves, the term hard disk is also used to refer to the whole of a computer's internal data storage Beginning in the early 21st century, some personal computers and laptops were produced that used solid-state drives (SSDs) that relied on flash memory chips instead of hard disks to store information.

5.3 Ram

With 8 GB of RAM, you will have enough memory to run several programs at once. You can open lots of browser tabs at once, use photo or video editing programs, stream content, and play mid-to-high-end games.

Many Windows 10 and macOS computers or laptops come with 8 GB of memory installed these days. So, 8 GB of memory should be more than enough to run most productivity programs. It's also the minimum amount of memory recommended by Adobe to run Creative Cloud programs like Photoshop.

5.4 Monitor

A computer monitor is an output device that displays information in pictorial or text form. A monitor usually comprises a visual display, some circuitry, a casing, and a power supply. The display device in modern monitors is typically a thin film transistor liquid crystal display (TFT-LCD) with LED backlighting having replaced cold-cathode fluorescent lamp (CCFL) backlighting. Previous monitors used a cathode ray tube (CRT) and some Plasma (also called Gas-Plasma) displays. Monitors are connected to the computer via VGA, HDMI, DisplayPort, USB-C, low-voltage differential signaling (LVDS) or other proprietary connectors and signals.

Fig 4.3 Monitor

4.5 BP MACHINE

Blood Pressure Chart by Age

Age	Min Systolic/Diastolic	Normal Systolic/Diastolic	Max Systolic/Diastolic
1 to 12 months	75 / 50	90 / 60	100 / 75
1 to 5 years	80 / 55	95 / 65	110 / 79
6 to 13 years	90 / 60	105 / 70	115 / 80
14 to 19 years	105 / 73	117 / 77	120 / 81
20 to 24 years	108 / 75	120 / 79	132 / 83
25 to 29 years	109 / 76	121 / 80	133 / 84
30 to 34 years	110 / 77	122 / 81	134 / 85
35 to 39 years	111 / 78	123 / 82	135 / 86
40 to 44 years	112 / 79	125 / 83	137 / 87
45 to 49 years	115 / 80	127 / 84	139 / 88
50 to 54 years	116 / 81	129 / 85	142 / 89
55 to 59 years	118 / 82	131 / 86	144 / 90
60 to 64 years	121 / 83	134 / 87	147 / 91

www.free-printable-paper.com

4.6 GLUCOSE MONITOR

Blood Glucose Chart			
Mg/DL	Fasting	After Eating	2-3 Hours After Eating
Normal	80-100	170-200	120-140
Impaired Glucose	101-125	190-230	140-160
Diabetic	126+	220-300	200+

4.7 GLUCOSE STRIPS

CHAPTER 6

SOFTWARE DESCRIPTION

Operating System	: Windows 7/8/ 10/11
Server side Script	: Python
IDE	: PyCharm, Anakonda
Libraries Used	: SKlearn, pandas, numpy, Matplotlib, Collections
Technology	: Python 3.7
Front End	: HTML
Dataset	: Kaggle

5.1 PYTHON

Python is a general purpose, dynamic, high level and interpreted programming language. It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures. It is easy to learn yet powerful and versatile scripting language which makes it attractive for Application Development. It's syntax and dynamic typing with its interpreted nature, makes it an ideal language for scripting and rapid application development. It supports multiple programming patterns, including object oriented, imperative and functional or procedural programming styles. It is not intended to work on special area such as web programming. That is why it is known as multipurpose because it can be used with web, enterprise, 3D CAD etc. We don't need to use data types to declare variable because it is dynamically typed so we can write a=10to assign an integer value in an integer variable. It makes the development and debugging fast because there is no

compilation step included in python development and edit-test-debug cycle is very fast

5.2 HTML

The HyperText Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.Webbrowsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by *tags*, written using angle brackets. Tags such as <img /> and <input/>directly introduce content into the page. Other tags such as surround <p> and provide information about document text and may include other tags as sub- elements. Browsers do not display the HTML tags but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript, which affects the behaviour and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.[2] A form of HTML, known as HTML5, is used to display video and audio, primarily using the <canvas> element, in collaboration with javascript.

CHAPTER 7

APPLICATIONS

1. Hospital applications
2. Medical appliances
3. Ayurveda Treatments

CHAPTER 8

IMPLEMENTATION

Jupyter notebook code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)

df = pd.read_csv("healthcare-dataset-stroke-data.csv")

# The first 5 instances of the dataframe
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

Printing the number of N/A values in eacg column

print(df.isna().sum())

Graphical representation of the na values present in the attribute - bar graph

df.isna().sum().plot.barh()

To check the statistical analysis of all numerical type attributes (count, mean, standaard deviation, minimum values, all quartiles, maximum values)

df.describe()

	id	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	4909.000000	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	7.854067	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	23.500000	0.000000
50%	36932.000000	45.000000	0.000000	0.000000	91.885000	28.100000	0.000000
75%	54682.000000	61.000000	0.000000	0.000000	114.090000	33.100000	0.000000
max	72940.000000	82.000000	1.000000	1.000000	271.740000	97.600000	1.000000

Provides the data type of all attributes and the number of NOT NULL values count is obtained

df.info()

Checking the values in the gender column

df['gender'].value_counts()

Removing the 'other' gender instance inorder to reduce the dimension

df['gender'] = df['gender'].replace('Other','Female')

plotting a pie chart to see the gender count distribution

df['gender'].value_counts().plot(kind="pie")

Value count in the stroke attribute

df['stroke'].value_counts()

Graphical representation of the value count distribution of the target attribute

df['stroke'].value_counts().plot(kind="bar",color = "cyan")

```
print("% of people who actualy got a stroke : ",(df['stroke'].value_counts()[1]/df['stroke'].value_counts().sum()).round(3)*100)
# Graphical representation of the value counts of the hypertension attribute
df['hypertension'].value_counts().plot(kind="bar",color = "red")

# Value of count of work-type attribute
df['work_type'].value_counts()
# Graphical representation of the value counts of the work-type attribute
df['work_type'].value_counts().plot(kind="pie")

# Value of count of somoking status attribute
df['smoking_status'].value_counts()
# Graphical representation of the value counts of the smoking staus attribute
df['smoking_status'].value_counts().plot(kind="pie")
# Graphical representation of the value counts of the residence attribute
df['Residence_type'].value_counts().plot(kind="pie")
# Number of BMI - NULL values
df['bmi'].isnull().sum()
# Graphical representation of bmi attribute
sns.histplot(data=df['bmi'])
sns.boxplot(data=df['bmi'])
# Finding the count of outliers based on those instances which are out of iqr
Q1 = df['bmi'].quantile(0.25)
Q3 = df['bmi'].quantile(0.75)
# Finding IQR
IQR = Q3 - Q1
da=(df['bmi'] < (Q1 - 1.5 * IQR)) | (df['bmi'] > (Q3 + 1.5 * IQR))
da.value_counts()
# Percentage of NULL values in bmi
```

```
df['bmi'].isna().sum()/len(df['bmi'])*100
df_na=df.loc[df['bmi'].isnull()]
g=df_na['stroke'].sum()
print("People who got stroke and their BMI is NA:",g)
h=df['stroke'].sum()
print("People who got stroke and their BMI is given:",h)
print("percentage of people with stroke in Nan values to the overall dataset:",g/h*100)
# Percentage of instances who got stroke
df['stroke'].sum()/len(df)*100
# Analysing whether to drop NA values in Bmi column
df_na=df.loc[df['bmi'].isnull()]
print("Nan BMI values where people have stroke:",df_na['stroke'].sum())
print("overall BMI values where people have stroke:",df['stroke'].sum())
# Imputing the missing N/A values using the median of bmi column
print("median of bmi",df['bmi'].median())
df['bmi']=df['bmi'].fillna(df['bmi'].median())
# Graphical representation fo the data in age column
# histogram
sns.histplot(data=df['age'])
# boxplot
sns.boxplot(data=df['age'])
# Graphical representation fo the data in glucose level column
# histogram
sns.histplot(data=df['avg_glucose_level'])
# Boxplot
sns.boxplot(data=df['avg_glucose_level'])
# Finding the count of outliers based on those instances which are out of iqr
Q1 = df['avg_glucose_level'].quantile(0.25)
Q3 = df['avg_glucose_level'].quantile(0.75)
```

IQR = Q3 - Q1

da=(df['avg_glucose_level'] < (Q1 - 1.5 * IQR)) | (df['avg_glucose_level'] > (Q3 + 1.5 * IQR))

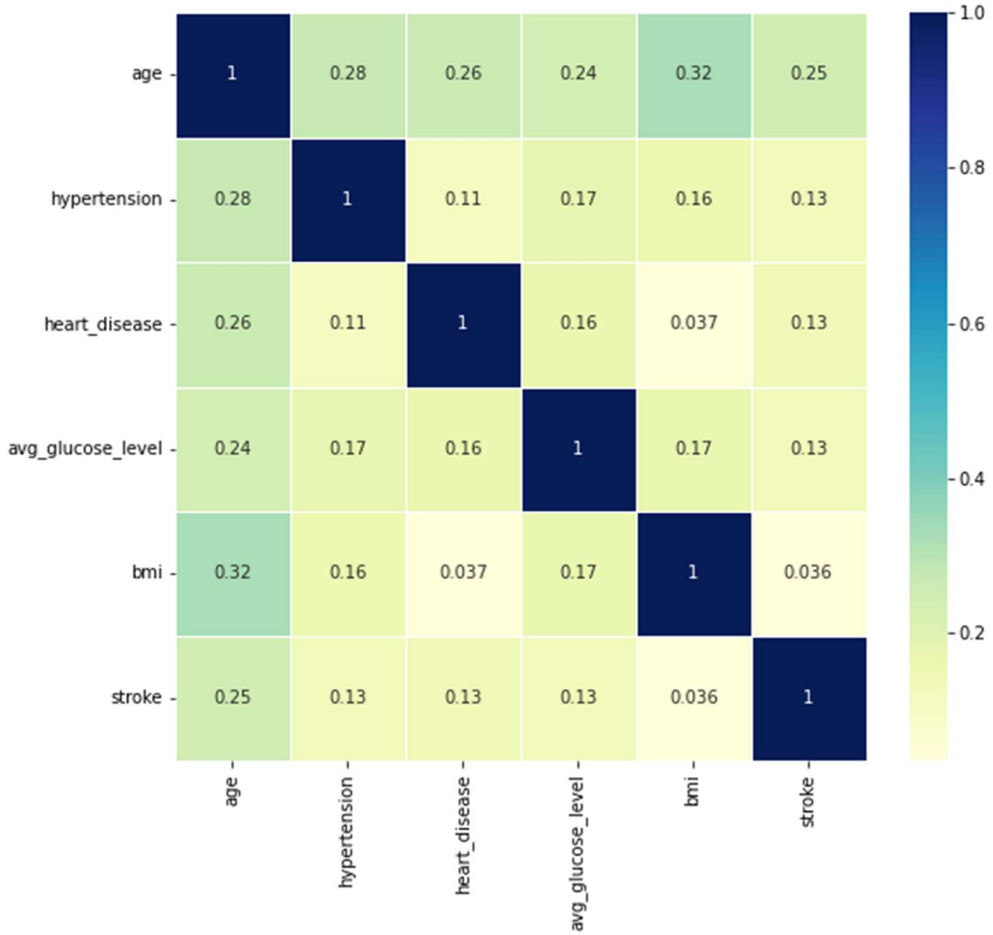
da.value_counts()

Correlation matrix between the attributes in the dataset to find if any attributes are correlated

corrmat=df.corr()

f,ax=plt.subplots(figsize=(9,8))

sns.heatmap(corrmat,ax=ax,cmap="YlGnBu",linewidth=0.8,annot=True)



Value count of heart disease attribute

df['heart_disease'].value_counts()

df['heart_disease'].value_counts().plot(kind="pie")

Value count of evver married attribute

df['ever_married'].value_counts()

```
# Graphical representation
df['ever_married'].value_counts().plot(kind="pie")
# Comparing stroke with gender
sns.countplot(x='stroke', hue='gender', data=df)
# Comparing stroke with work-type
sns.countplot(x='stroke', hue='work_type', data=df)
# Comparing stroke with somking_status
sns.countplot(x='stroke', hue='smoking_status', data=df)
# Comparing stroke with residence type
sns.countplot(x='stroke', hue='Residence_type', data=df)
# Comparing stroke with heart disease
sns.countplot(x='stroke', hue='heart_disease', data=df)
# Comparing stroke with married status
sns.countplot(x='stroke', hue='ever_married', data=df)
# Converting numeric-binary value attributes to string
df[['hypertension', 'heart_disease', 'stroke']] = df[['hypertension', 'heart_disease',
'stroke']].astype(str)
# Generating dummy attributes - one hot encoding format
df = pd.get_dummies(df, drop_first= True)
# Since our Dataset is highly undersampled (based on target instances) we are
going to perform a over sampling method to have equal representation of both
the target classes
# Using random oversampling - importing the library
from imblearn.over_sampling import RandomOverSampler


# Performing a minority oversampling
oversample = RandomOverSampler(sampling_strategy='minority')
X=df.drop(['stroke_1'],axis=1)
y=df['stroke_1']


# Obtaining the oversampled dataframes - testing and training
```

```
X_over, y_over = oversample.fit_resample(X, y)


# importing a scaling modeule
from sklearn.preprocessing import StandardScaler


# Since the numeric attributes in the dataset is in different ranges and three are outliers persent we are usign a scaler to get all the values into the same range.
s = StandardScaler()
# Scaling the numeric attributes
df[['bmi', 'avg_glucose_level', 'age']] = s.fit_transform(df[['bmi', 'avg_glucose_level', 'age']])
# creating dataset split for training and testing the model
from sklearn.model_selection import train_test_split
# Performing a 80-20 test-train split
X_train, X_test, y_train, y_test = train_test_split(X_over, y_over, test_size= 0.20, random_state= 42)
# Checking the size of the splits
print('X_train:', X_train.shape)
print('y_train:', y_train.shape)
print('X_test:', X_test.shape)
print('y_test:', y_test.shape)
```

#**DECISION TREE**

```
#importing the Decision Tree Classifier module
from sklearn.tree import DecisionTreeClassifier
# Libraries for calculating performance metrics
from sklearn import metrics
from sklearn.metrics import auc,roc_auc_score,roc_curve,precision_score,recall_score,f1_score


# Create the classifier object
```

```
clf = DecisionTreeClassifier()

# Training the classifier
clf = clf.fit(X_train,y_train)

#predicting result using the test dataset
y_pred = clf.predict(X_test)

# Printing the accuracyof the model
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9794344473007712
```

```
#KNN
#importing the KNN Classifier module
from sklearn.neighbors import KNeighborsClassifier
# Libraries for calculating performance metrics
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix
from sklearn.metrics import auc,roc_auc_score,roc_curve,precision_score,recall_score,f1_score

# Create the classifier object
# 2 neighbours because of the 2 classes
knn = KNeighborsClassifier(n_neighbors = 2)
# Training the classifier
knn.fit(X_train,y_train)
#predicting result using the test dataset
```

```
y_pred_knn = knn.predict(X_test)
y_pred_prob_knn = knn.predict_proba(X_test)[:, 1]

# Printing the accuracy and roc-auc score of the model
confusion_matrix(y_test, y_pred_knn)
print('Accuracy:',accuracy_score(y_test, y_pred_knn))
print('ROC AUC Score:', roc_auc_score(y_test, y_pred_prob_knn))
```

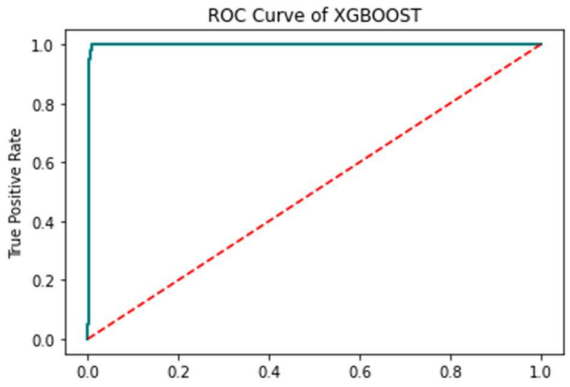
```
Accuracy: 0.9722365038560411
ROC AUC Score: 0.9723076923076923
```

#XGBoost

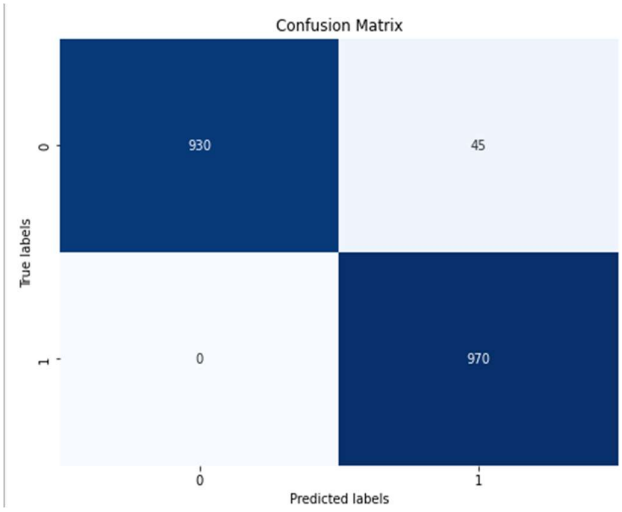
```
#importing the XGBoost Classifier module
from xgboost  import XGBClassifier

# Create the classifier object
xgb = XGBClassifier()
# Training the classifier
xgb.fit(X_train,y_train)
#predicting result using the test dataset
y_pred_xgb = xgb.predict(X_test)
y_pred_prob_xgb = xgb.predict_proba(X_test)[:, 1]

# Printing the accuracy and roc-auc score of the model
print('Accuracy:', accuracy_score(y_test, y_pred_xgb))
print('ROC AUC Score:', roc_auc_score(y_test, y_pred_prob_xgb))

# plots of roc_auc
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob_xgb)

plt.figure(figsize=(6,4))
```

```
plt.plot(fpr, tpr, linewidth=2, color= 'teal')
plt.plot([0,1], [0,1], 'r--' )
plt.title('ROC Curve of XGBOOST')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')

plt.show()
```

```
Accuracy: 0.9768637532133676
ROC AUC Score: 0.9985863071636267
```




```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Compute confusion matrix
cm = confusion_matrix(y_test, y_pred_xgb)

# Plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.xlabel("Predicted labels")
plt.ylabel("True labels")
plt.title("Confusion Matrix")
plt.show()
```



Printing the precision,recall,f1score and support values of the model based on the confusion matrix

from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score

print("Accuracy_score:",accuracy_score(y_test,y_pred_xgb))

print("Precision_score:",precision_score(y_test,y_pred_xgb))

print("Recall_score:",recall_score(y_test,y_pred_xgb))

print("f1_score:",f1_score(y_test,y_pred_xgb))

print('ROC AUC Score:', roc_auc_score(y_test, y_pred_prob_xgb))

```
Accuracy_score: 0.9768637532133676
Precision_score: 0.9556650246305419
Recall_score: 1.0
f1_score: 0.9773299748110831
ROC AUC Score: 0.9985863071636267
```

#**RANDOM FOREST**

importing random forest classifier module for training

from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import RandomizedSearchCV

Create the classifier object

rf_clf = RandomForestClassifier(n_estimators = 100)

```
# Train the model using the training sets
rf_clf.fit(X_train, y_train)
# performing predictions on the test dataset
y_pred_rf = rf_clf.predict(X_test)
# Printing accuracy of the model
print('Accuracy:', accuracy_score(y_test, y_pred_rf))
```

```
Accuracy: 0.9953727506426735
```

Logistic regression

```
from sklearn.linear_model import LogisticRegression

classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)

y_pred_lr = classifier.predict(X_test)

confusion_matrix(y_test, y_pred_lr)
print('Accuracy:', accuracy_score(y_test, y_pred_lr))
```

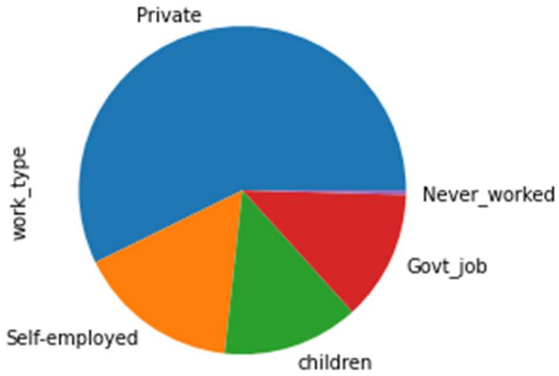
```
Accuracy: 0.7701799485861183
```

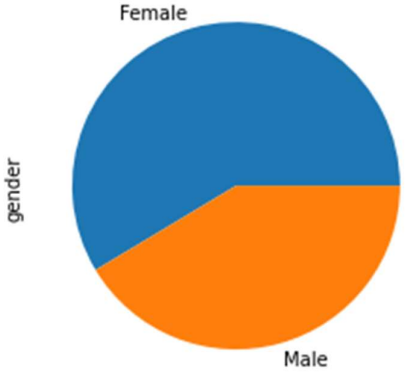
 # Making sample predictions based on manual value entry

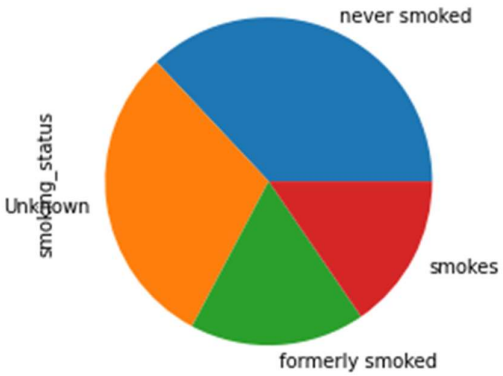
```
age = 40
avg_glucose_level = 100
bmi = 20
gender_Male = 1
ever_married_Yes = 1
work_type_Never_worked = 0
```

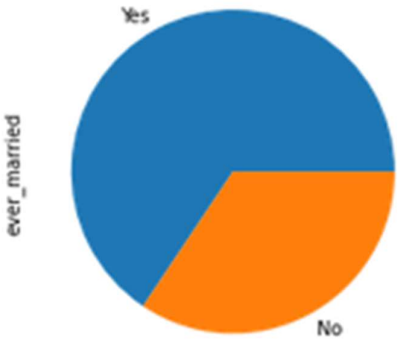
```
work_type_Private = 1
work_type_Self_employed = 0
work_type_children = 0
Residence_type_Urban = 1
smoking_status_formerly_smoked = 0
smoking_status_never_smoked = 1
smoking_status_smokes = 0
hypertension_1 = 0
heart_disease_1 = 0

input_features = [age, avg_glucose_level, bmi, gender_Male, hypertension_1, heart_disease_1,
        ever_married_Yes, work_type_Never_worked, work_type_Private, work_type_Self_employed,
        work_type_children, Residence_type_Urban, smoking_status_formerly_smoked,
        smoking_status_never_smoked, smoking_status_smokes]

features_value = [np.array(input_features)]
features_name = ['age', 'avg_glucose_level', 'bmi', 'gender_Male', 'hypertension_1', 'heart_disease_1',
        'ever_married_Yes', 'work_type_Never_worked', 'work_type_Private', 'work_type_Self-employed',
        'work_type_children', 'Residence_type_Urban', 'smoking_status_formerly smoked',
        'smoking_status_never smoked', 'smoking_status_smokes']

df = pd.DataFrame(features_value, columns=features_name)
prediction = rf_clf.predict(df)[0]
print(prediction)
```

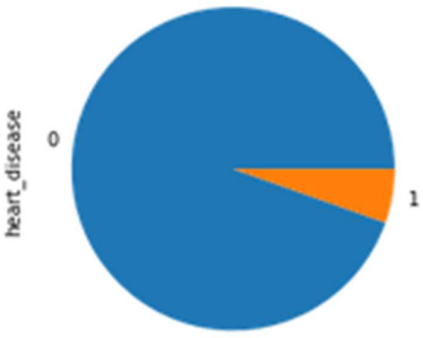
RESULTS

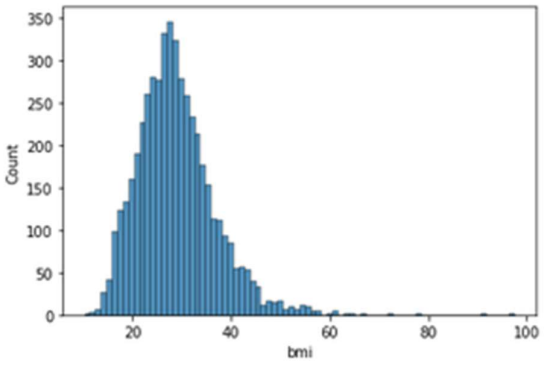

Private
work_type
Never_worked
Govt_job
Self-employed
children

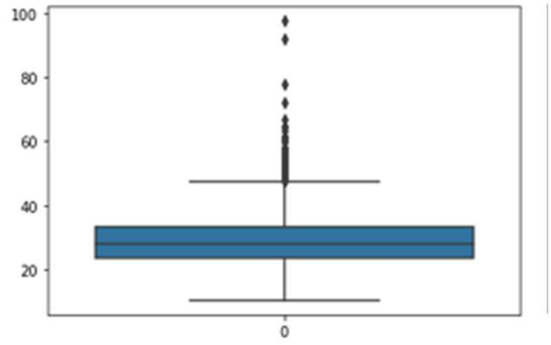

Female
gender
Male

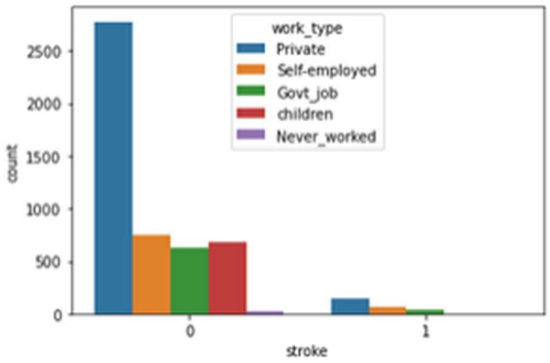

never smoked
smoking_status
Unknown
smokes
formerly smoked

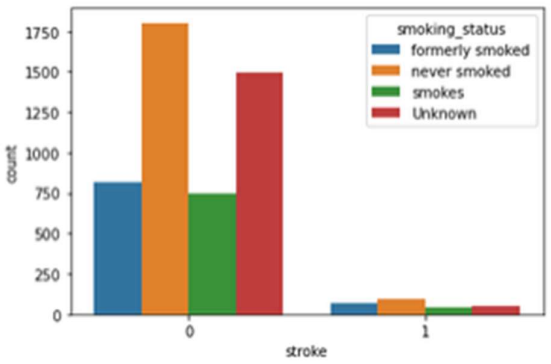

Yes
ever_married
No

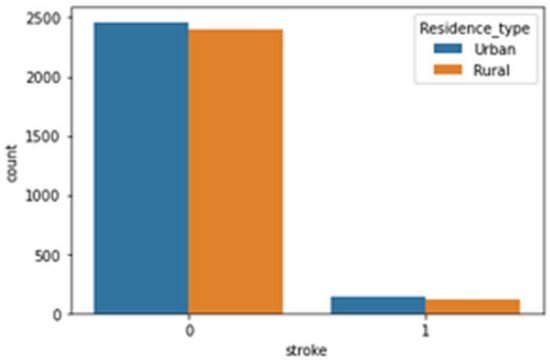

heart_disease
0
1

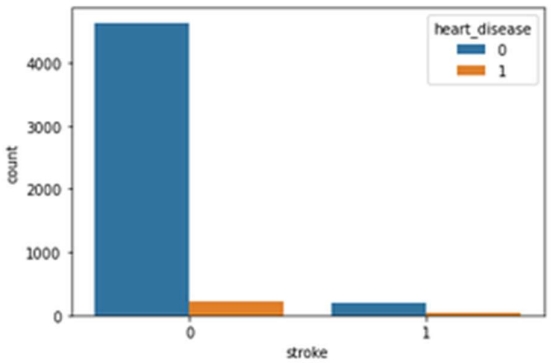

Count
bmi
0
50
100
150
200
250
300
350
20
40
60
80
100

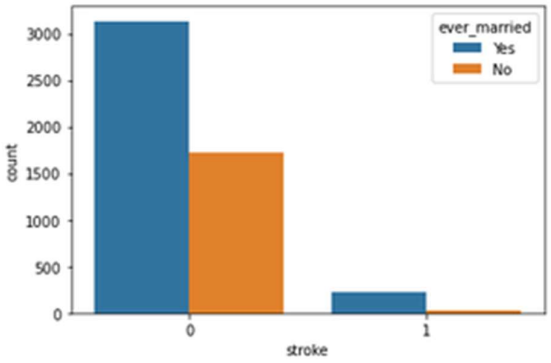

100
80
60
40
20
0


work_type
Private
Self-employed
Govt_job
children
Never_worked
count
2500
2000
1500
1000
500
0
0
1
stroke


smoking_status
formerly smoked
never smoked
smokes
Unknown
count
1750
1500
1250
1000
750
500
250
0
0
1
stroke


Residence_type
Urban
Rural
count
2500
2000
1500
1000
500
0
0
1
stroke


heart_disease
0
1
count
4000
3000
2000
1000
0
0
1
stroke


ever_married
Yes
No
count
3000
2500
2000
1500
1000
500
0
0
1
stroke

REFERENCES

[1] Singh, M.S., Choudhary, P., Thongam, K.: A comparative analysis for various stroke prediction techniques. In: Springer, Singapore (2020)

[2] Pradeepa, S., Manjula, K. R., Vimal, S., Khan, M. S., Chilamkurti, N., &Luhach, A. K.: DRFS: Detecting Risk Factor of Stroke Disease from Social Media Using Machine Learning Techniques. In Springer (2020).

[3] Dataset named „Stroke Prediction Dataset" from Kaggle: https://www.kaggle.com/fedesoriano/strokeprediction-dataset

[4] Wang, S., Li, Y., Tian, J., Peng, X., Yi, L., Du, C., Feng, C., Liang, X. (2020). A randomized controlled trial of brain and heart health manager-led mHealth secondary stroke prevention. Cardiovascular Diagnosis and Therapy, 10(5): 1192-1199. https://doi.org/10.21037/cdt-20-423

[5] Wilkinson, D.A., Daou, B.J., Nadel, J.L., Chaudhary, N., Gemmete, J.J., Thompson, B.G., Pandey, A.S. (2020). Abdominal aortic aneurysm is associated with subarachnoid hemorrhage. Journal of Neuro Interventional Surgery. https://doi.org/10.1136/neurintsurg2020-016757

[6] Verma, A., Jaiswal, S., Sheikh, W.R. (2020). Acute thrombotic occlusion of subclavian artery presenting as a stroke mimic. Journal of the American College of Emergency Physicians Open, 1(5): 932-934. https://doi.org/10.1002/emp2.12085

[7] Yu, J., Park, S., Lee, H., Pyo, C.S., Lee, Y.S. (2020). An elderly health monitoring system using machine learning and in-depth analysis techniques on the NIH stroke scale. Mathematics, 8(7): 1-16. https://doi.org/10.3390/math8071115

[8] Xie, Y., Jiang, B., Gong, E., Li, Y., Zhu, G., Michel, P., Wintermark, M., Zaharchuk, Z. (2019). Use of gradient boosting machine learning to predict patient outcome in acute ischemic stroke on the basis of imaging, demographic, and clinical information. American Journal of Roentgenology, 212(1): 44-51. https://doi.org/10.2214/AJR.18.20260

[9] Boukobza, M., Nahmani, S., Decschamps, L., Laissy, J.P. (2019). Brain abscess complicating ischemic embolic stroke in a patient with cardiac papillary fibroelastoma - Case report and literature review. Journal of Clinical Neuroscience, 66: 277-279. https://doi.org/10.1016/j.jocn.2019.03.041

[10] S. Sung, S.M., Kang, Y.J., Cho, H.J., Kim, N.R., Lee, S.M., Choi, B.K., Cho, G. (2020). Prediction of early neurological deterioration in acute minor ischemic stroke by machine learning algorithms. Clinical Neurology and Neurosurgery, 195: 105892. https://doi.org/10.1016/j.clineuro.2020.10589 2