STROKE PREDICTION USING MACHINE LEARNING

A Project Report submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

P. Praneet 221910301043
Ch. Aryan Reddy 221910301017
K. Vivek Reddy 221910301032
P. Sree Paada Reddy 221910301044

Under the esteemed guidance of

Dr. P. K. Bhagat

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE &ENGINEERING

GITAM

(Deemed to be University)

HYDERABAD

NOVEMBER 2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM INSTITUTE OF TECHNOLOGY

GITAM

(Deemed to be University)

DECLARATION

I/We, hereby declare that the project report entitled "**STROKE PREDICTION USING MACHINE LEARNING**" is an original work done in the Department of Computer Science and Engineering, GITAM Institute of Technology, GITAM (Deemed to be University) submitted in partial fulfillment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date: 07-11-2022

Registration No(s).	Name(s)	Signature(s)
221910301043	**P. Praneet**	
221910301017	**Ch. Aryan Reddy**	
221910301032	**K. Vivek Reddy**	
221910301044	**P. Sree Paada Reddy**	

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM INSTITUTE OF TECHNOLOGY

GITAM

(Deemed to be University)

CERTIFICATE

This is to certify that the project report entitled "**STROKE PREDICTION USING MACHINE LEARNING**" is a bonafide record of work carried out by **P. Praneet (221910301043),Ch. Aryan Reddy (221910301017),K. Vivek Reddy (221910301032),P. Sree Paada Reddy (221910301044)** students submitted in partial fulfillment of requirement for the award of degree of Bachelors of Technology in Computer Science and Engineering.

Project Guide	**Head of the Department**
Dr. P. K. Bhagat	**Dr. S. Phani Kumar**
Assistant Professor	**Professor**

TABLE OF CONTENTS

TABLE OF FIGURES

1. ABSTRACT

Stroke is among the leading causes of death in the world. It is a condition that causes a tear in blood vessels in the vital organs of the human body such as the brain. Technology has been increasingly widening its scope, there is a tremendous potential to increase the collaboration with the medical industry to get valuable insights. The ever-growing availability of medical records has enabled us to understand how correlative various lifestyle decisions are and their contribution to the stroke prediction. This project has made use of various risk factors that medically affect stroke along with modern machine learning algorithms like Logistic Regression, Adaboost, Naïve Bayes Classification, XGBoost to train different models for accurate stroke prediction. The XGBoost algorithm has given the highest accuracy as compared to the other algorithms with an accuracy of 86.5%.

2. INTRODUCTION

Stroke is among the leading causes of death in the world. A stroke, sometimes called a brain attack, occurs when something blocks blood supply to part of the brain or when a blood vessel in the brain bursts. In either case, parts of the brain become damaged or die. In an adult human being even when the body is set to rest 20% of the oxygen and glucose also about 2% of the entire body weight is constituted by the brain. It is carried out through the internal carotid and vertebral arteries. The blood is then altered from head to heart through the internal jugular veins. The loss of the blood might be observed in two scenarios in which the flow of blood among the blood tissues decreases resulting in ischemic stroke whereas if internal bleeding occurs among the brain tissues known as hemorrhagic stroke.

A stroke can cause lasting brain damage, long-term disability, or even death. There is a highly negative impact of stroke among the public and hence has led to an increased focus on identifying risk factors of stroke.With the help of ever increasing recording of patient's data. Most of the past research has more or less focussed on heart stroke, very rarely some research has been done on brain stroke.

Technology has been increasingly widening its scope, there is a tremendous potential to increase collaboration with the medical industry to get valuable insights. The different machine learning approaches used in the project include four different algorithms - Logistic Regression, Adaboost, Naïve Bayes Classification, XGBoost. The patient management has significantly improved in the past few years by systematically mining and archiving the patient’s medical records. Such analysis has helped the medical practitioners to make an accurate prognosis of any medical condition.

3. LITERATURE REVIEW

Various studies from the past have been analyzed in order to understand the problems in the existing models. The use of ML Prediction here refers to the output of an algorithm after it has been trained on a historical dataset and applied to new data when forecasting the likelihood of a particular outcome. ML has been used in healthcare to help to analyze thousands of different data points and suggest outcomes, provide timely risk scores, precise resource allocation, and has many other applications.

In [1] the basic undersampling technique is used which further constrains the already small dataset. Further only four features are used (age, heart_disease, hypertension and avg_glucose_level) on the sole basis of correlation. This further leads to drop in accuracy scores predicted by the ML model as each and every factor has a very high contribution to the stroke cause. Medically ignoring a risk factor is a sustainable way of eliminating the cause of contribution to the disease and hence it has to be given importance even though they may not be significantly independent.

Further use of neural networks has already been outperformed by past studies and hence neural network fails as an important algorithm in this prediction as

1. Implementation of not enough nodes may be a reason behind this issue because models with fewer nodes need to change their architecture drastically to model the data better and fail to converge.
2. The amount of the training data is low or the data we are pushing on the model is corrupted or not collected with the data integrity.
3. Inappropriate weight application in the network can also cause a failure in convergence. The weights we are applying to the network should be well calculated according to the activation function.

In the existing research work, the authors have done only a classification of multiple types of strokes together. This leads to improper classification and we do not attain decent accuracy to predict the stroke severity. Due to this, there is a gap identified for predicting the risk levels of stroke factors which are low, moderate, high and severe. To overcome the challenge of sampling we have used SMOTE technique along with the proposed algorithms to predict the risk levels of stroke factors.

4. PROBLEM IDENTIFICATION AND OBJECTIVES

4.1 PROBLEM IDENTIFICATION

Stroke being one of the leading causes of sudden death in the world causes a tremendous impact on society. As of today, the only way to lessen the impact of stroke is its prevention. For preventing a stroke there is a need to identify the impact of various risk factors that lead to a person getting a stroke. Deployment of the Machine Learning algorithms would be done to train the machine so that it can predict the likelihood of getting a stroke as accurately as possible based on the risk factors of an individual thereby contributing to a life saving cause.

4.2 OBJECTIVES

1. Employ suitable Preprocessing that can improve efficiency of Machine Learning techniques
2. Train an accurate Machine Learning Algorithm using the historic stroke patients data.
3. Test the performance of every algorithm.
4. Compare and analyze multiple Machine Learning techniques
5. Calculate the risk of the user getting a stroke based on the above inputs using the final outcome

5. SYSTEM METHODOLOGIES

The project has followed the given flow

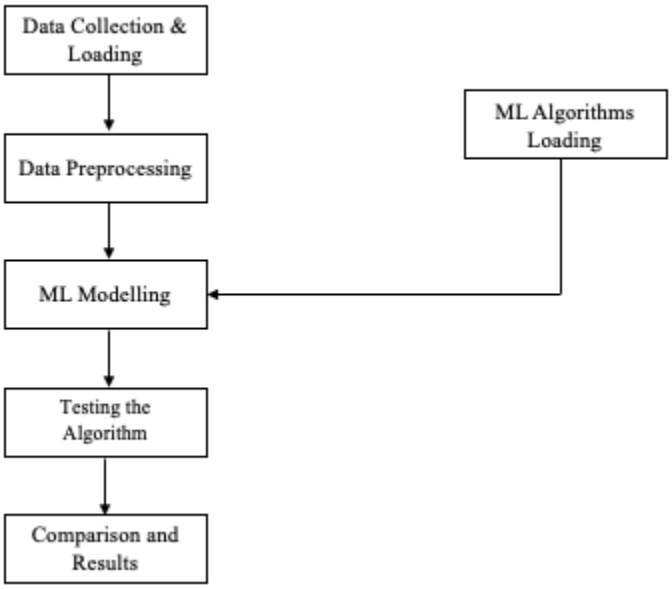

Figure 5.1 Project Flowchart

The dataset is first loaded into the colab. The preprocessing is done next where we check for duplicates and check them. In addition we handle missing values, handling imbalanced data takes place and performing label encoding that are specific for this particular dataset.Now that the data is preprocessed, it is ready for model building. For model building, we need to split the dataset into training and test data, here we consider a 70:30 split between train and test data. In addition since the number of people with stroke condition in the dataset is extremely low as compared to people without stroke condition we use the SMOTE technique for Data Sampling. The sampled data is now given to ML Algorithms. Naïve Bayes Classification, Adaboost, Logistic Regression and XGBoost algorithms are used. After building these four different models we compare them using four accuracy metrics namely Accuracy Score, F1 Score, Recall Score and Precision Score to determine the best performing model.

6. OVERVIEW OF TECHNOLOGIES

The various technologies used are given below

6.1 TOOLS

6.1.1 Google Colab

Colab is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUs. It mainly allows teams to work together on python notebooks.

6.1.2 Python

Python is a very popular programming language. It was discovered by Guido Van Rossum. Python is an open source language that has many inbuilt functions. It is the easiest language because it has huge libraries and modules. Opencv ,tesseract, Matplotlib and Numpy are mostly used in the project for image recognition and text extraction and creating a web form with the obtained information.

6.2 ALGORITHMS

A total of four machine learning algorithms are used in this project these include :

1. Adaboost
2. Naive Bayes Classification
3. Logistic Regression
4. XGBoost

6.2.1 Adaboost

AdaBoost algorithm is a Boosting technique used as an Ensemble Method in Machine Learning. It is called Adaptive Boosting as the weights are re-assigned to each instance, with higher weights assigned to incorrectly classified instances. Boosting is used to reduce bias as well as variance for supervised learning. AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers. The individual learners can be weak, but as long as the performance of each one is slightly better than random guessing, the final model can be proven to converge to a strong learner.

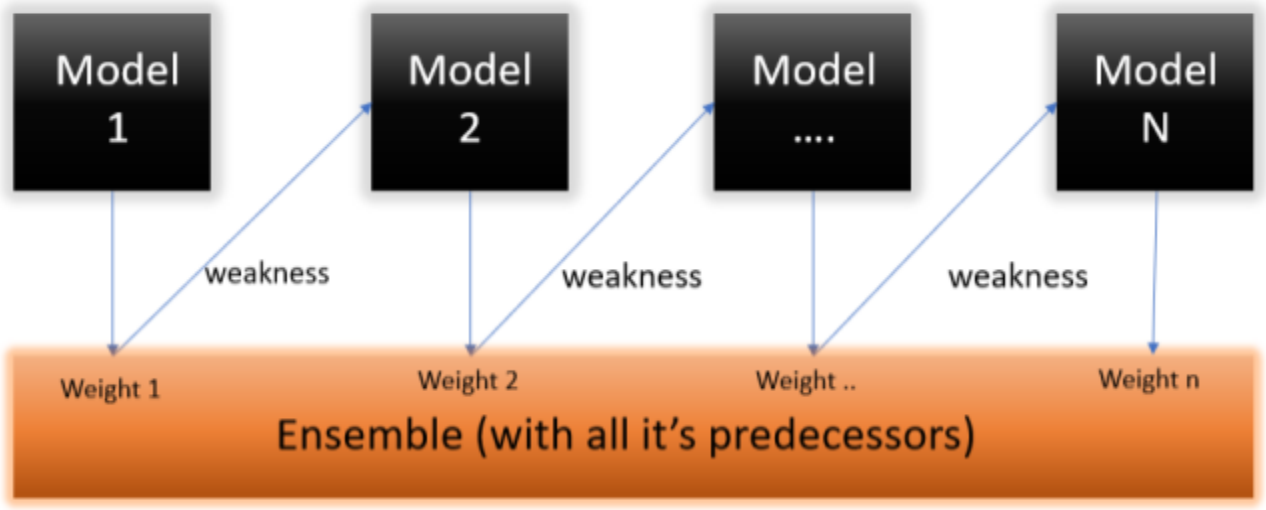

Fig 6.1 Adaboost Algorithm

Adaboost algorithm makes 'n' number of decision trees during the data training period. As the first decision tree/model is made, the incorrectly classified record in the first model is given priority. Only these records are sent as input for the second model. The process goes on until we specify a number of base learners we want to create.

6.2.2 Naive Bayes

The Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. It is mainly used in text classification that includes a high-dimensional training dataset. It is one of the most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object. The naive bayes algorithm assumes that each feature makes an independent and equal contribution to the outcome.

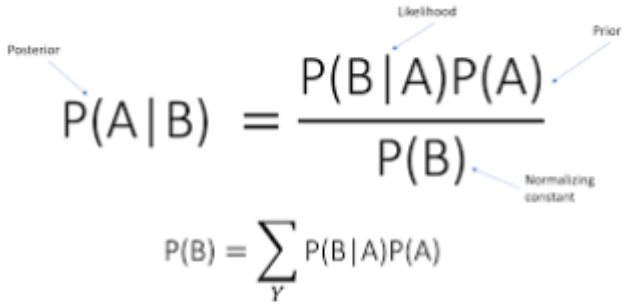

Figure 6.2 Bayes Theorem

6.2.3 Logistic Regression

Logistic regression is an example of supervised learning. It is used to calculate or predict the probability of a binary (yes/no) event occurring. An example of logistic regression could be applying machine learning to determine if a person is likely to be infected with COVID-19 or not. In logistic regression, a logit transformation is applied on the odds—that is, the probability of success divided by the probability of failure.

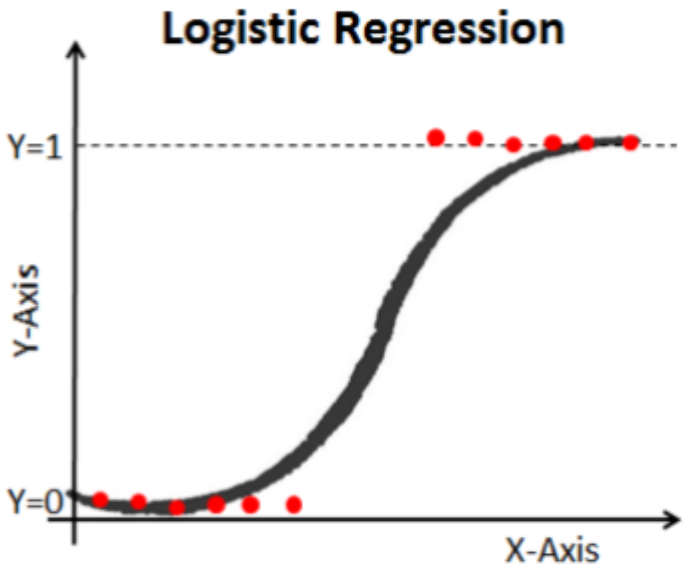

Figure 6.3 Logistic Regression

In this algorithm the dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no). Mathematically, the logistic regression model predicts P(Y=1) as a function of X.

6.2.4 XGBoost

XGBoost, which stands for Extreme Gradient Boosting, is a scalable, distributed gradient-boosted decision tree (GBDT) machine learning library. It provides parallel tree boosting and is the leading machine learning library for regression, classification, and ranking problems.The term “gradient boosting” comes from the idea of “boosting” or improving a single weak model by combining it with a number of other weak models in order to generate a collectively strong model. Gradient boosting is an extension of boosting where the process of additively generating weak models is formalized as a gradient descent algorithm over an objective function. Gradient boosting sets targeted outcomes for the next model in an effort to minimize errors.With XGBoost, trees are built in parallel, instead of sequentially.

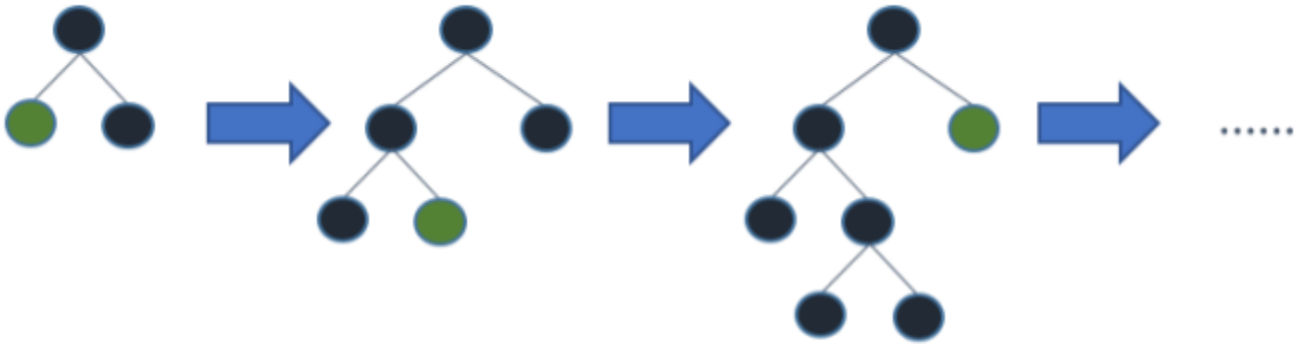

Figure 6.4 XGBoost

6.3 LIBRARIES

1. Pandas: It is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. Its name is a play on the phrase "Python data analysis" itself

2. Numpy: It is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. It was created by incorporating the features of the competing Numarray into Numeric, with extensive modifications. The core functionality of NumPy is its "ndarray", for n-dimensional array, data structure. These arrays are strided views on memory. In contrast to Python's built-in list data structure, these arrays are homogeneously typed: all elements of a single array must be of the same type.

3. Sklearn: It is a free software machine learning library for the Python programming language. Sklearn is a NumFOCUS fiscally sponsored project. It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

4. Imblearn: Imbalanced-learn is an open source library relying on sklearn and provides tools when dealing with classification with imbalanced classes. In machine learning, while building a classification model we sometimes come to situations where we do not have an equal proportion of classes. That means when we have class imbalance issues for example we have 500 records of 0 class and only 200 records of 1 class, here Imblearn techniques help to either upsample the minority class or downsample the majority class to match the equal proportion.

5. Collections: The collection Module in Python provides different types of containers. A Container is an object that is used to store different objects and provide a way to access the contained objects and iterate over them. It provides a rich set of specialized container data types carefully designed to approach specific programming problems in a Pythonic and efficient way.

6. Xgboost: XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting that solves many machine learning problems in a fast and accurate way. It is the leading machine learning library for regression, classification, and ranking problems.

7. IMPLEMENTATION

Stroke Prediction using Machine Learning - CSEBNUM_A12

Packages Import

```
[1] import pandas as pd
    import numpy as np
    from sklearn.preprocessing import LabelEncoder
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import confusion_matrix
    from sklearn.preprocessing import MinMaxScaler
    from imblearn.over_sampling import SMOTE
    from collections import Counter
    from sklearn.naive_bayes import GaussianNB
    from sklearn.metrics import accuracy_score, classification_report
    from sklearn.ensemble import AdaBoostClassifier
    from sklearn.linear_model import LogisticRegression
    from xgboost import XGBClassifier
```

```
[2] df = pd.read_csv('healthcare-dataset-stroke-data.csv')
    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

Data Preprocessing

```
[3] df.shape
```

```
(5110, 12)
```

```
[4] # removing neccessary duplicates
    df = df.drop_duplicates()
    df.shape
```

```
(5110, 12)
```

```
[5] # checking missing values
    df.isnull().sum()
```

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

```
[6] #Filling the missing values
    df.fillna(value = df['bmi'].median(), inplace = True)
    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                  0
smoking_status       0
stroke               0
dtype: int64
```

```
[7] #checking all unique values
    print('gender : ',df.gender.unique())
    print('hypertension : ',df.hypertension.unique())
    print('heart disease : ',df.heart_disease.unique())
    print('ever married : ',df.ever_married.unique())
    print('work type : ',df.work_type.unique())
    print('residence type : ',df.Residence_type.unique())
    print('smoking status : ',df.smoking_status.unique())
```

```
gender :  ['Male' 'Female' 'Other']
hypertension :  [0 1]
heart disease :  [1 0]
ever married :  ['Yes' 'No']
work type :  ['Private' 'Self-employed' 'Govt_job' 'children' 'Never_worked']
residence type :  ['Urban' 'Rural']
smoking status :  ['formerly smoked' 'never smoked' 'smokes' 'Unknown']
```

```
# Removing unnecessary values
s = df[df['smoking_status'] == 'Unknown'].index
s
```

```
Int64Index([   8,    9,   13,   19,   23,   27,   31,   38,   46,   50,
            ...
            5088, 5089, 5094, 5095, 5097, 5098, 5101, 5103, 5104, 5109],
           dtype='int64', length=1544)
```

```
[55] d = df[df['gender'] == 'Other'].index
     d
```

```
Int64Index([3116], dtype='int64')
```

```
[56] df.drop(d, inplace = True)
     df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5109 entries, 0 to 5109
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   id                 5109 non-null   int64  
 1   gender             5109 non-null   object 
 2   age                5109 non-null   float64
 3   hypertension       5109 non-null   int64  
 4   heart_disease      5109 non-null   int64  
 5   ever_married       5109 non-null   object 
 6   work_type          5109 non-null   object 
 7   Residence_type     5109 non-null   object 
 8   avg_glucose_level  5109 non-null   float64
 9   bmi                5109 non-null   float64
 10  smoking_status     5109 non-null   object 
 11  stroke             5109 non-null   int64  
dtypes: float64(3), int64(4), object(5)
memory usage: 518.9+ KB
```

```
[57] #converting categorical data into numeric data
     enc=LabelEncoder()
```

```
[58] gender=enc.fit_transform(df['gender'])
     smoking_status=enc.fit_transform(df['smoking_status'])
     work_type=enc.fit_transform(df['work_type'])
     Residence_type=enc.fit_transform(df['Residence_type'])
     ever_married=enc.fit_transform(df['ever_married'])
```

```
[59] df['ever_married']=ever_married
     df['Residence_type']=Residence_type
     df['smoking_status']=smoking_status
     df['gender']=gender
     df['work_type']=work_type
```

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

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	1	67.0	0	1	1	2	1	228.69	36.6	1	1
1	51676	0	61.0	0	0	1	3	0	202.21	28.1	2	1
2	31112	1	80.0	0	1	1	2	0	105.92	32.5	2	1
3	60182	0	49.0	0	0	1	2	1	171.23	34.4	3	1
4	1665	0	79.0	1	0	1	3	0	174.12	24.0	2	1

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

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smok
count	5109.000000	5109.000000	5109.000000	5109.000000	5109.000000	5109.000000	5109.000000	5109.000000	5109.000000	5109.000000	
mean	36513.985516	0.413975	43.229986	0.097475	0.054022	0.656293	2.167743	0.508123	106.140399	28.863300	
std	21162.008804	0.492592	22.613575	0.296633	0.226084	0.474991	1.090398	0.499983	45.285004	7.699785	
min	67.000000	0.000000	0.080000	0.000000	0.000000	0.000000	0.000000	0.000000	55.120000	10.300000	
25%	17740.000000	0.000000	25.000000	0.000000	0.000000	0.000000	2.000000	0.000000	77.240000	23.800000	
50%	36922.000000	0.000000	45.000000	0.000000	0.000000	1.000000	2.000000	1.000000	91.880000	28.100000	
75%	54643.000000	1.000000	61.000000	0.000000	0.000000	1.000000	3.000000	1.000000	114.090000	32.800000	
max	72940.000000	1.000000	82.000000	1.000000	1.000000	1.000000	4.000000	1.000000	271.740000	97.600000	

```
[62] scaler = MinMaxScaler()
     cn = ['age','work_type','avg_glucose_level','bmi','smoking_status']
     df[cn] = scaler.fit_transform(df[cn])
```

```
[63] df.drop(columns = 'id', inplace = True)
```

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

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	1	0.816895	0	1	1	0.50	1	0.801265	0.301260	0.333333	1
1	0	0.743652	0	0	1	0.75	0	0.679023	0.203895	0.666667	1
2	1	0.975586	0	1	1	0.50	0	0.234512	0.254296	0.666667	1
3	0	0.597168	0	0	1	0.50	1	0.536008	0.276060	1.000000	1
4	0	0.963379	1	0	1	0.75	0	0.549349	0.156930	0.666667	1

```
[65] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5109 entries, 0 to 5109
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   gender             5109 non-null   int64  
 1   age                5109 non-null   float64
 2   hypertension       5109 non-null   int64  
 3   heart_disease      5109 non-null   int64  
 4   ever_married       5109 non-null   int64  
 5   work_type          5109 non-null   float64
 6   Residence_type     5109 non-null   int64  
 7   avg_glucose_level  5109 non-null   float64
 8   bmi                5109 non-null   float64
 9   smoking_status     5109 non-null   float64
 10  stroke             5109 non-null   int64  
dtypes: float64(5), int64(6)
memory usage: 479.0 KB
```

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

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status
count	5109.000000	5109.000000	5109.000000	5109.000000	5109.000000	5109.000000	5109.000000	5109.000000	5109.000000	5109.000000
mean	0.413975	0.526733	0.097475	0.054022	0.656293	0.541936	0.508123	0.235529	0.212638	0.458994
std	0.492592	0.276045	0.296633	0.226084	0.474991	0.272599	0.499983	0.209053	0.088199	0.357209
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.304199	0.000000	0.000000	0.000000	0.500000	0.000000	0.102114	0.154639	0.000000
50%	0.000000	0.548340	0.000000	0.000000	1.000000	0.500000	1.000000	0.169698	0.203895	0.666667
75%	1.000000	0.743652	0.000000	0.000000	1.000000	0.750000	1.000000	0.272228	0.257732	0.666667
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

```
#checking for extreme cases of correlation
df.corr()
```

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
gender	1.000000	-0.027752	0.021223	0.085685	-0.030171	0.056576	-0.006105	0.054722	-0.026452	-0.062423	0.009081
age	-0.027752	1.000000	0.276367	0.263777	0.679084	-0.361686	0.014031	0.238323	0.324211	0.265165	0.245239
hypertension	0.021223	0.276367	1.000000	0.108292	0.164187	-0.051772	-0.007980	0.174540	0.158252	0.111018	0.127891
heart_disease	0.085685	0.263777	0.108292	1.000000	0.114601	-0.028031	0.003045	0.161907	0.036879	0.048445	0.134905
ever_married	-0.030171	0.679084	0.164187	0.114601	1.000000	-0.352831	0.005988	0.155329	0.334770	0.259604	0.108299
work_type	0.056576	-0.361686	-0.051772	-0.028031	-0.352831	1.000000	-0.007348	-0.050492	-0.299218	-0.305942	-0.032323
Residence_type	-0.006105	0.014031	-0.007980	0.003045	0.005988	-0.007348	1.000000	-0.004783	-0.000444	0.008168	0.015415
avg_glucose_level	0.054722	0.238323	0.174540	0.161907	0.155329	-0.050492	-0.004783	1.000000	0.167033	0.063498	0.131991
bmi	-0.026452	0.324211	0.158252	0.036879	0.334770	-0.299218	-0.000444	0.167033	1.000000	0.218928	0.036075
smoking_status	-0.062423	0.265165	0.111018	0.048445	0.259604	-0.305942	0.008168	0.063498	0.218928	1.000000	0.028108
stroke	0.009081	0.245239	0.127891	0.134905	0.108299	-0.032323	0.015415	0.131991	0.036075	0.028108	1.000000

Making Data suitable for ML algorithm

```
[68] X = df.drop(['stroke'], axis=1)
     y = df['stroke']
```

```
[69] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=12)
```

```
[70] print('X train - ',X_train.shape)
     print('Y train - ',y_train.shape)
     print('X test - ',X_test.shape)
     print('Y test - ',y_test.shape)
```

```
X train -  (3576, 10)
Y train -  (3576,)
X test -  (1533, 10)
Y test -  (1533,)
```

SMOTE

```
[71] counter = Counter(y_train)
     print('Before Sampling',counter)
     smt = SMOTE(random_state=12)
     X_train_sm, y_train_sm = smt.fit_resample(X_train, y_train)

     counter = Counter(y_train_sm)
     print('After Sampling',counter)
```

```
Before Sampling Counter({0: 3404, 1: 172})
After Sampling Counter({0: 3404, 1: 3404})
```

Naive Bayes

```
[72] gnb = GaussianNB()
     gnb.fit(X_train_sm,y_train_sm)
```

```
GaussianNB()
```

```
[73] y_pred = gnb.predict(X_train_sm)
     pred = [round(value) for value in y_pred]

     accuracy = accuracy_score(y_train_sm, pred)
     print('Train Accuracy',accuracy)
```

```
Train Accuracy 0.7532314923619271
```

```
[74] pred = gnb.predict(X_test)
     pred = [round(value) for value in pred]

     accuracy = accuracy_score(y_test, pred)
     print('Test Accuracy' ,accuracy)
```

```
Test Accuracy 0.7318982387475538
```

```
[75] print(classification_report(y_test, pred))
```

```
                  precision    recall  f1-score   support

               0       0.99      0.73      0.84      1456
               1       0.14      0.86      0.24        77

        accuracy                           0.73      1533
       macro avg       0.57      0.79      0.54      1533
    weighted avg       0.95      0.73      0.81      1533
```

```
[76] c = confusion_matrix(y_test, pred)
     print(c)
```

```
[[1056  400]
 [  11   66]]
```

Adaboost

```
[77] adbo = AdaBoostClassifier(random_state=12)
     adbo.fit(X_train_sm,y_train_sm)
```

```
AdaBoostClassifier(random_state=12)
```

```
[78] y_pred = adbo.predict(X_train_sm)
     pred = [round(value) for value in y_pred]

     accuracy = accuracy_score(y_train_sm, pred)
     print('Train Accuracy',accuracy)
```

```
Train Accuracy 0.868096357226792
```

```
[79] pred = adbo.predict(X_test)
     pred = [round(value) for value in pred]

     accuracy = accuracy_score(y_test, pred)
     print('Test Accuracy', accuracy)
```

```
Test Accuracy 0.8258317025440313
```

```
[80] print(classification_report(y_test, pred))
```

```
                  precision    recall  f1-score   support

               0       0.97      0.84      0.90      1456
               1       0.16      0.56      0.24        77

        accuracy                           0.83      1533
       macro avg       0.56      0.70      0.57      1533
    weighted avg       0.93      0.83      0.87      1533
```

```
[81] c = confusion_matrix(y_test, pred)
     print(c)
```

```
[[1223  233]
 [  34   43]]
```

Logistic Regression

```
[82] logis = LogisticRegression(solver = 'liblinear')
     logis.fit(X_train_sm, y_train_sm)
```

```
LogisticRegression(solver='liblinear')
```

```
[83] y_pred = logis.predict(X_train_sm)
     pred = [round(value) for value in y_pred]

     accuracy = accuracy_score(y_train_sm, pred)
     print('Train Accuracy',accuracy)
```

```
Train Accuracy 0.7740893066980024
```

```
[84] pred = logis.predict(X_test)
     pred = [round(value) for value in pred]

     accuracy = accuracy_score(y_test, pred)
     print('Test Accuracy', accuracy)
```

```
Test Accuracy 0.7273320287018917
```

```
[85] print(classification_report(y_test, pred))
```

```
                  precision    recall  f1-score   support

               0       0.99      0.72      0.83      1456
               1       0.14      0.84      0.24        77

        accuracy                           0.73      1533
       macro avg       0.56      0.78      0.54      1533
    weighted avg       0.95      0.73      0.80      1533
```

```
[86] c = confusion_matrix(y_test, pred)
     print(c)
```

```
[[1050  406]
 [  12   65]]
```

XGboost

```
[87] xgb_model = XGBClassifier()
     xgb_model.fit(X_train_sm, y_train_sm)
```

```
XGBClassifier()
```

```
[88] y_pred = xgb_model.predict(X_train_sm)
     pred = [round(value) for value in y_pred]

     accuracy = accuracy_score(y_train_sm, pred)
     print("Train Accuracy",accuracy)
```

```
Train Accuracy 0.9164218566392479
```

```
[89] y_pred = xgb_model.predict(X_test)
     pred = [round(value) for value in y_pred]

     accuracy = accuracy_score(y_test, pred)
     print("Test Accuracy",accuracy)
```

```
Test Accuracy 0.8656229615133725
```

```
[90] print(classification_report(y_test, pred))
```

```
                  precision    recall  f1-score   support

               0       0.97      0.89      0.93      1456
               1       0.19      0.49      0.27        77

        accuracy                           0.87      1533
       macro avg       0.58      0.69      0.60      1533
    weighted avg       0.93      0.87      0.89      1533
```

```
[91] c = confusion_matrix(y_test, pred)
     print(c)
```

```
[[1289  167]
 [  39   38]]
```

8. RESULT AND DISCUSSIONS

The results obtained by the algorithms are tabulated below

Algorithm	F1	Accuracy	Recall	Precision
Naïve Bayes	0.81	0.73	0.73	0.95
Adaboost	0.87	0.83	0.83	0.93
Logistic Regression	0.8	0.72	0.73	0.95
XGBoost	0.89	0.87	0.87	0.93

8.1 Accuracy

Model accuracy is the measurement used to determine which model is best at identifying relationships and patterns between variables in a dataset based on the input, or training, data. The better a model can generalize to 'unseen' data, the better predictions and insights it can produce, which in turn deliver more business value.

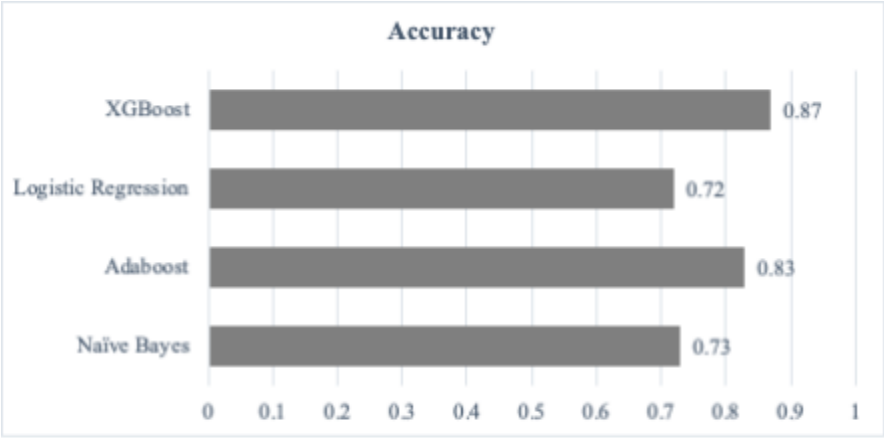

Fig 8.1 Accuracy Comparison

On comparison of the accuracy scores it is clearly distinguished that the XGBoost algorithm is the best performing algorithm followed by the Adaboost.

8.2 F1 Score

In statistical analysis of binary classification, the F-score or F-measure is a measure of a test's accuracy. It is calculated from the precision and recall of the test, where the precision is the number of true positive results divided by the number of all positive results, including those not identified correctly, and the recall is the number of true positive results divided by the number of all samples that should have been identified as positive. Precision is also known as positive predictive value, and recall is also known as sensitivity in diagnostic binary classification.

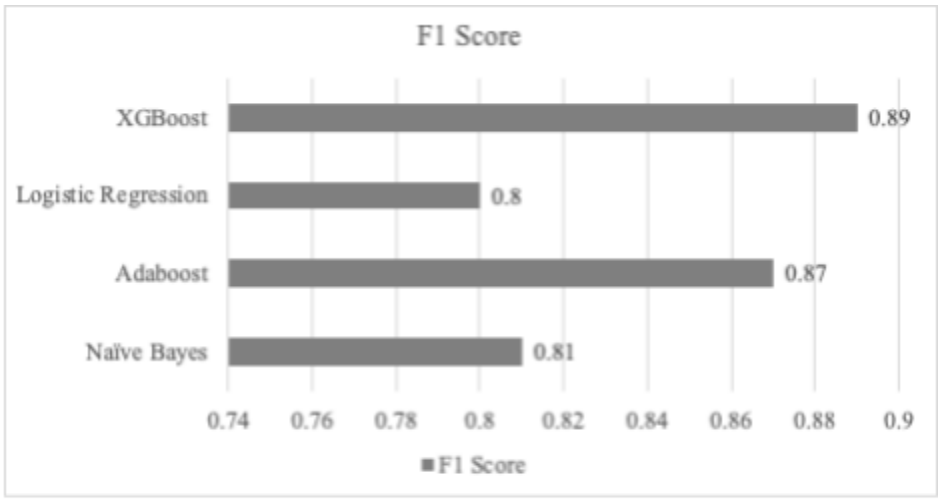

Fig 8.2 F1 Score Comparison

8.3 Recall

The recall is calculated as the ratio between the numbers of Positive samples correctly classified as Positive to the total number of Positive samples. The recall measures the model's ability to detect positive samples. The higher the recall, the more positive samples detected. Unlike Precision, Recall is independent of the number of negative sample classifications. Further, if the model classifies all positive samples as positive, then Recall will be 1.

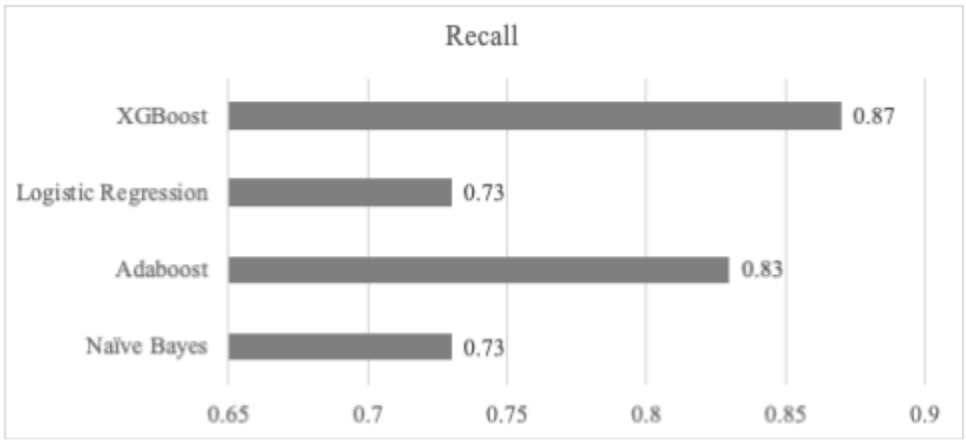

Fig 8.3 Recall Comparison

XGBoost has a higher recall over the other algorithms. Ensemble based methods tend to perform better over the classification techniques.

8.4 Precision

Precision is defined as the ratio of correctly classified positive samples (True Positive) to a total number of classified positive samples (either correctly or incorrectly).Precision, hence, calculates the accuracy for the minority class.

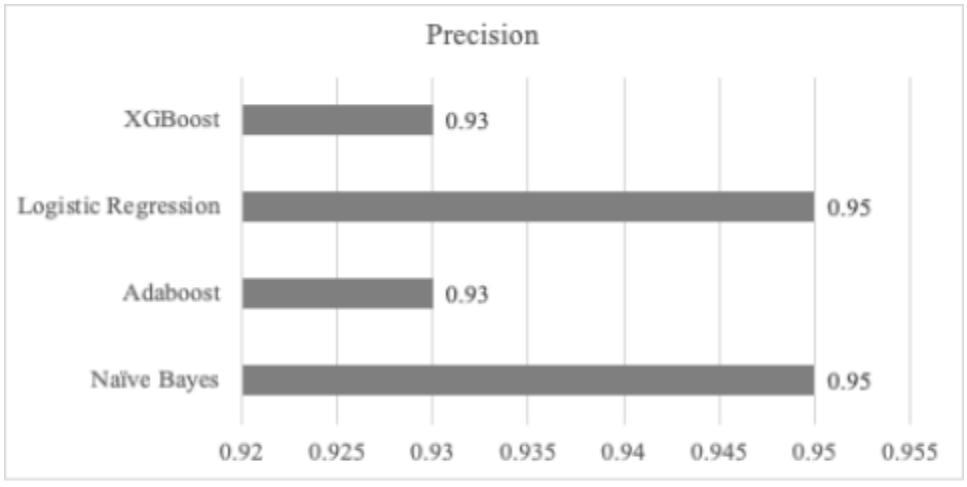

Fig 8.4 Precision Comparison

We can see that all the algorithms have almost a similar precision score.

Clearly the XGBoost algorithm has outperformed other algorithms and hence is the model that shall be used for the prediction of stroke.

9. CONCLUSION AND FUTURE SCOPE

The XGBoost algorithm is seemingly the best suited algorithm for stroke prediction. An accuracy of 86.5 % is achieved through the stroke predictor model. Using the ensemble learning techniques, we obtained efficient results with improved prediction accuracy. Going by the famous medical philosophy of "Prevention is better than cure", These outcomes of the project will aid the efforts made by researchers to prevent deaths due to stroke.

The project can further be improved by creating a visual interface for people to enter their details with regards to the risk factors and the display of the final stroke risk being displayed on the screen. A new dataset needs to be collected which can help us benchmark the results achieved in this project. A larger dataset is essential to train the algorithm in a more effective way and it can be done via both traditional and a more modern way of directly scraping data from hospitals and insurance records where stroke conditions would be explicitly mentioned. In addition we can also train methods for different types of strokes along with risk levels using an image data set.

10. REFERENCES

A. Journals/Articles

1. Soumyabrata Dev, Hewei Wang, Chidozie Shamrock Nwosue, Nishtha Jain, Bharadwaj Veeravalli, Deepu Johng, "A predictive analytics approach for stroke prediction using machine learning and neural networks", *Elsevier Healthcare Analytics* **Vol. 2(2022)doi:10.1016/j.health.2022.100032**
2. J.F. Meschia, C. Bushnell, B. Boden-Albala, L.T. Braun, D.M. Bravata, S. Chaturvedi, M.A Creager, R.H. Eckel, M.S. Elkind, M. Fornage, et al., "Guidelines for the primary prevention of stroke: a statement for healthcare professionals from the American heart association/American stroke association", *Stroke* **45 (12) (2014) 3754–3832**.

B. Books

3. Brian K. Jones and David M. Beazley, Python Cookbook: Recipes for Mastering Python 3 (3rd Edition)
4. Aurelion Geron, Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools and Techniques to build Intelligent Systems
5. Tom M. Mitchell, Machine Learning, McGraw Hill, 2017.
6. EthemAlpaydin, Introduction to Machine Learning, 3/e, PHI, 2015.

C. e-Websites / Downloads

7. https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset
8. https://towardsdatascience.com/sampling-techniques-a4e34111d808
9. https://scikit-learn.org/stable/modules/naive_bayes.html
10. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html
11. https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
12. https://xgboost.readthedocs.io/en/stable/python/python_api.html