AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY

AMITY UNIVERSITY JHARKHAND

TERM PAPER

TITLE: BRAIN STROKE PREDICTION USING MACHINE LEARNING ALGORITHMS.

Under the guidance of:
Ms. KANIKA THAKUR
Asst. professor
AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY

Compiled by:

NAME :SUBHAJIT TEWARY
PROGRAM : B.TECH (CSE)
ENROLLMENT NO: A35705218006

ACKNOWLEDGEMENT

I owe a debt of gratitude to AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY, JHARKHAND and my mentor MS. KANIKA THAKUR for the vision and foresight they had in me which inspired me to complete this term paper.

I want to thank my family and friends for trusting in me and helping me during the whole research paper completion process.

This was an exceptional learning opportunity for me and I would like to thank all those engaged in making this research possible once again.

DECLARATION BY THE STUDENT

I Subhajit Tewary student of B.Tech.(CSE), hereby declare that the project titled “BRAIN STROKE PREDICTION USING MACHINE LEARNING ALGORITHMS” which is submitted by me to AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY, AMITY UNIVERSITY JHARKHAND in partial fulfilment of requirement has not been previously formed the basis for the award of any degree, diploma or other similar title or recognition.

The researcher attests that approval on use of any copy-righted content appearing in the research / project report has been gained other than brief extracts requiring only proper acknowledgement in academic writing and all such use is recognition.

DATE:

Name of student: SUBHAJIT TEWARY

Signature of student:

CERTIFICATE

On the basis of declaration submitted by SUBHAJIT TEWARY student of B.Tech (CSE),hereby certify that the project titled "BRAIN STROKE PREDICTION USING MACHINE LEARNING ALGORITHMS" which is submitted to AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY, AMITY UNIVERSITY JHARKHAND is an original contribution with existing knowledge and faithful record of work carried out by him under my guidance and supervision.

To the best of my knowledge, this analysis has not been submitted in part of full to this university or elsewhere for any degree or diploma.

DATE:

Name of guide: MS. KANIKA THAKUR

Signature of guide:

TABLE OF CONTENT

TABLE OF FIGURES

ABSTRACT:

In today's world, brain stroke is a huge health issue, with nearly one person dying from it every minute. There is a wealth of data available in the health-care system about brain stroke, and if we use it correctly, we can diagnose the problem early on, provide effective therapy to the patient, and save many lives. The goal of this project is to provide a graphical user interface that allows the client to enter medical data and, using Machine Learning techniques, predict whether or not a person is having a brain stroke. The data for this project was taken from Kaggle's "Heart-disease-dataset," which has 1024 instances and 15 attributes. I utilised a variety of Machine Learning techniques in this research, and the Decision Tree and Random Forest algorithms fared the best, with a 94 percent accuracy. However, I chose the Random Forest technique to build my model over the Decision Tree algorithm.

1. INTRODUCTION:

1.1. BRIFE HISTORY OF MACHINE LEARNING:

In 1943, neurophysiologist Warren McCulloch and mathematician Walter Pitts published a study on neurons and how they work, which was the first example of neural organisation. They decided to simulate this using an electrical circuit, and the neural organisation was born as a result.

Alan Turing created the world-famous Turing Test in 1950. This is a simple test: for a computer to pass, it must be able to persuade a human that it is a human, not a computer.The first computer software that could learn as it ran was released in 1952. It was a checkers game created by Arthur Samuel.

In 1958, Frank Rosenblatt devised the Perceptron, the first fake neural network. The main goal of this was to set an example and recognise shapes.

Another extremely early instance of neuronal organisation occurred in 1959, when Bernard Widrow and Marcian Hoff of Stanford University created two models of them. The first was ADELINE, and it was capable of recognising binary patterns. It could, for example, predict what the next bit would be in a burst of bits. The next one was named MADELINE, and it could reduce reverberation on telephone lines, thus it was actually useful. It is still in use today.

Despite MADELINE's accomplishments, there was little progress until the late 1970s for a variety of reasons, the most prominent of which was the fame of Von Neumann engineering. This is a design in which directions and information are stored in a comparable memory or neural networks, which appears to be easier to understand than a neural organisation, and as a result, many people have created projects based on it.

1.2. OVERVIEW:

The heart is an important organ in the human body since it is responsible for the circulation of blood to all parts of the body. If it fails to do so, a person can suffer from heart failure, arrhythmias, and a variety of other issues. According to the World Health Organization, brain stroke has been the leading cause of mortality in recent years, with an estimated 17 million deaths worldwide each year.

Artificial intelligence, machine learning, and data mining are becoming increasingly popular in the healthcare industry. As the healthcare industry today creates a large amount of complicated data every second, including information about patients, disease diagnoses, medical devices, and so on. Now, if we process and analyse these enormous data sets, we may learn a lot. For example, we can discover the relationship between the fetchers and the parameters of each characteristic that are responsible for brain stroke. Not only can we get information on brain strokes, but we can also find information about any other ailments. By doing so, we can save a lot of money and, more importantly, make wiser decisions. Now, machine learning provides a variety of approaches that may be used to evaluate these data in order to uncover previously unknown relationships and provide the results to healthcare professionals so that they can make better informed decisions.

This research aims to provide a thorough examination of several Machine Learning approaches that can be used to automate the prediction of brain stroke. This automation will reduce the number of tests required of the patient, saving not only money but also time in the analysis process.

1.3. MACHINE LEARNING:

Machine Learning is a technique for teaching a computer to think like a human brain, or a field of research in which computers may learn without being explicitly taught.

Machine Learning is not the same as traditional programming. With traditional programming, we give some input data plus some logic, run it, and receive some output; however, in Machine Learning, we provide the data plus output relevant to that data, run the programme, and the machine trains itself, creating its own logic, which is then assessed during testing.

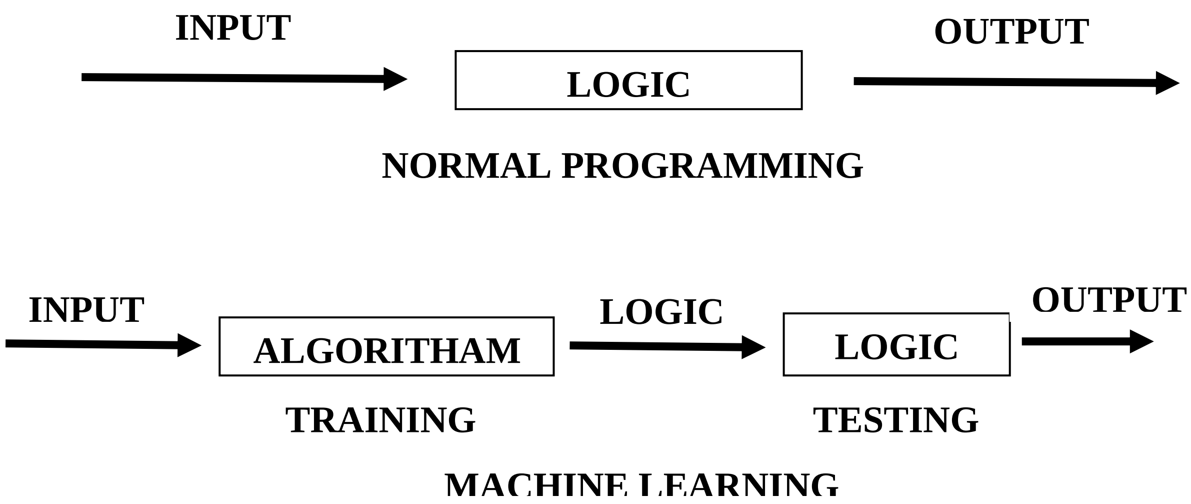
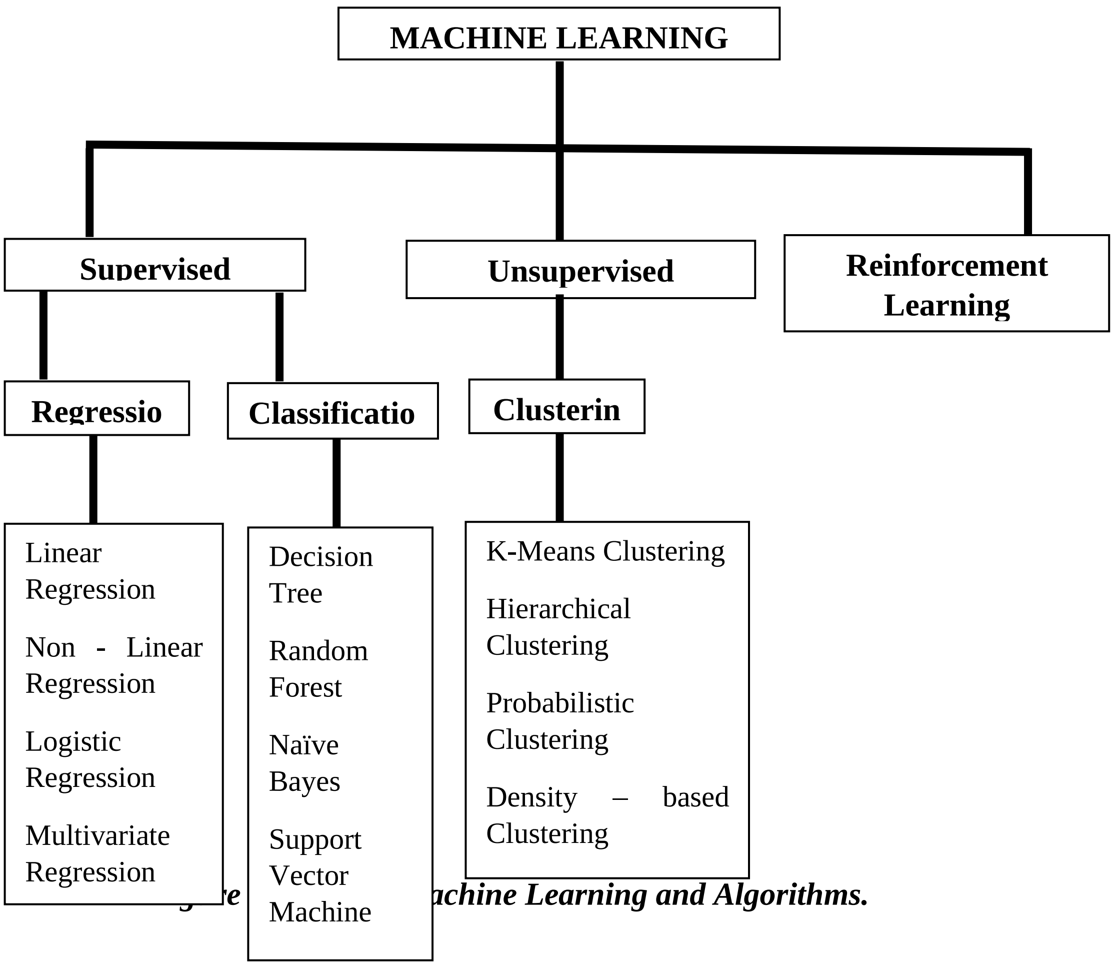


Figure 1. Normal programming vs Machine learning.

Now a machine can adopt any of the three learning approaches.

1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning



re achine Learning and Algorithms.

1.3.1. Supervised learning:

Supervised Learning is a learning method in which we train our model by giving it a labelled dataset to work with. The term "labelled dataset" refers to a dataset that contains both input and output for a given input.

For example, suppose we have a dataset of house prices in a specific area, and the dataset contains labels such as "area of the house in square feet," "where the house is located," and "what is the area's corruption rate," and we have our output label, "price," based on all the values that are present in all of these features. So, as the "area of the house in sq. feet" increases, and the property is located in a respectable location with a low crime rate, all of these factors will have a direct impact on the output "price." Our model will now train itself and construct a logic based on these inputs and outputs. Now you may use this reasoning to predict the output for fresh inputs.

There are two types of supervised learning.

- Regression
- Classification

1.3.1.1. Regression:

Regression is a supervised learning method that produces a continuous output within a defined range. The purpose of regression is to anticipate values as close to the original value as feasible, and the accuracy is measured by the error, which is the difference between the actual and predicted values. The less the mistake, the better the accuracy.

1.3.1.2. Classification:

Classification is a supervised learning strategy that produces discrete output. The purpose is to forecast discrete values belonging to a specific class, which can be binary (i.e. 0 or 1), multiclass (i.e. true or false), or binary (i.e. 0 or 1).

1.3.1.3. Unsupervised Learning:

We train the model via unsupervised learning by giving it raw data, unsystematic data, or data without a label. The machine's job here is to sort the data into groups based on their commonalities.

1.4. LITERATURE REVIEW

On the Cardiovascular Health Study (CHS) dataset, five machine learning algorithms were employed to predict cerebral stroke in [1.] To arrive at the best answer, the authors used the Decision Tree with the C4.5 approach, Principal Component Analysis, Artificial Neural Networks, and Support Vector Machine. The CHS Dataset employed in this study, on the other hand, has fewer input parameters.

Stroke prediction was carried out in [2] utilising user-generated social media posts. In this study, the authors used the DRFS method to identify the various symptoms associated with stroke disease. To extract text from social media postings, Natural Language Processing is required, which increases the model's total execution time, which is undesired.

Most of the publications outline some strategies for digging up information to infer stroke predictions. B. Support vector machines (Yan, Zhengetal. 2003; Andreeva2006, Das, SitarTight) showing varying degrees of accuracy such as decision trees, naive Bayes, neural networks, kernel densities, automatically characterized collections, bagging algorithms, etc. , Zdrenghea et al 2009; Turkoglu et al 2009; Srinivas Rani et al 2010; Raj Kumar and Reena 2010) Patients from various information bases around the world. Various parameters and data bases have been established by a number of creators in order to test the exactnesses. Specialists have been studying the use of the Decision Tree approach in the diagnosis of heart disease and have had a lot of success. Sitair-Tight et al. used the Weka tool to compare the use of Nave Bayes and J48 Decision Trees for the detection of heart illness. They also used the bagging method in the Weka tool and compared it to the J4.8 Decision Tree. [9] uses the Random forest algorithm to successfully analyse the dynamic cycle of a brain stroke. Heart disease is predicted based on the possibility of receiving help of one's choosing. As a result, the creator concluded that Decision Tree performs well, and that the exactness is sometimes comparable to Bayesian categorization.

2. METHODOLOGY

2.1. Data Flow Diagram

The graphic below depicts the data flow that occurs during the model development process. The data was obtained from "Kaggle.com.".

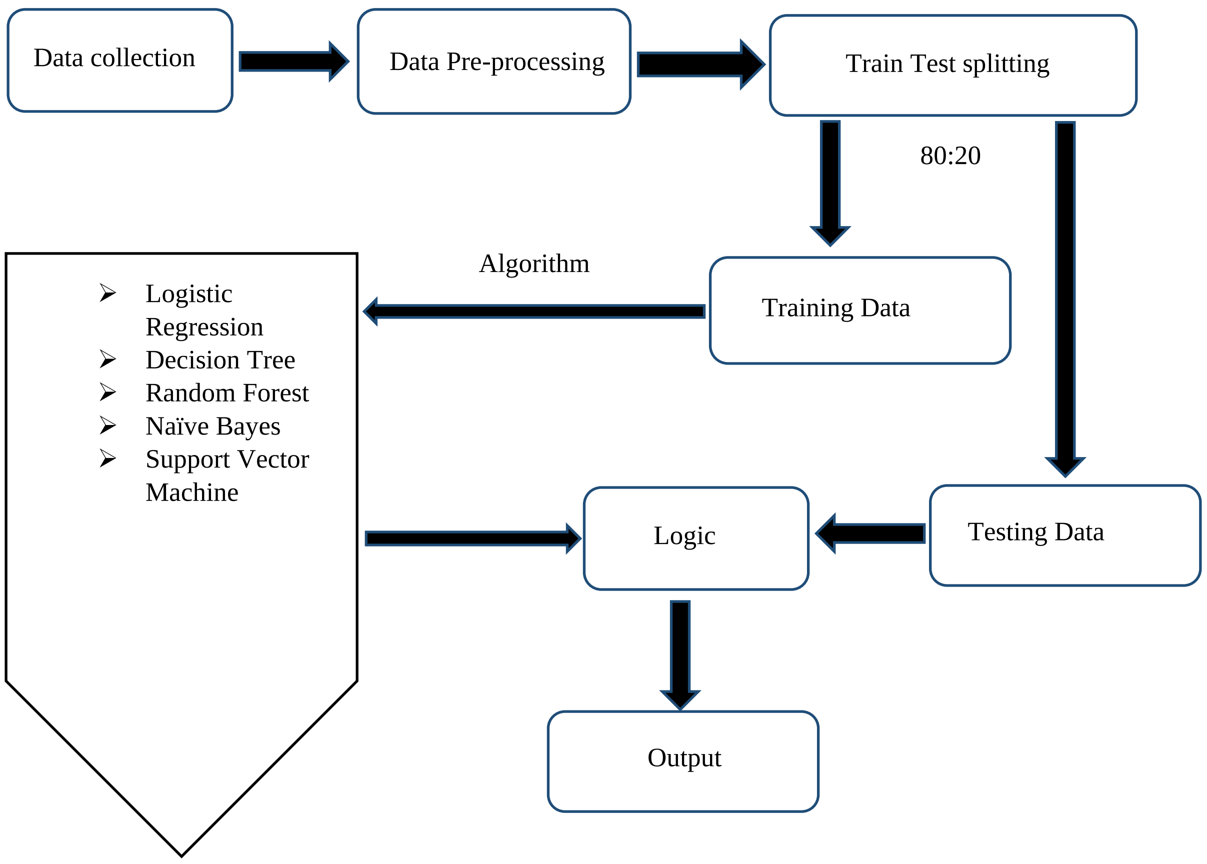


Figure 3. Data Flow Diagram

2.1.1. Data Collection:

This is the first step toward building a machine learning model that can predict brain stroke, and I used Kaggle.com to gather data..

2.1.2. Data pre-processing:

Data pre-processing, also known as data wrangling or data cleaning, is the most significant and crucial phase in machine learning. We search for missing values, pick relevant characteristics for the model, normalise the data, and so on in this step.

2.1.2.1 Feature Selection:

The data attributes you use to create your Machine Learning models have an impact on the type of display you may put on. Feature selection is the process of selecting those features that contribute the most to the forecast variable or yield that you are interested in..

2.1.2.2 Feature Scaling:

Feature scaling is a technique for normalising the free features found in a set of data. If feature scaling isn't done, a Machine Learning computation will weight more prominent characteristics higher and less prominent qualities lower, regardless of the unit of the qualities..

2.1.3. Train Test Splitting:

The train-test split is a method for evaluating a Machine Learning calculation's presentation. It is commonly used for classification or classification problems, but it may also be used for any supervised learning computation. The approach entails isolating a dataset into two parts, namely training data and testing data.

2.1.4. Training Data:

Training data is the information used to create a computation or machine learning model that predicts the outcome you want your model to predict..

2.1.5. Testing Data:

Test data is used to quantify the display of the computation you're using to create the machine, such as its exactness or efficacy.

2.1.6 Confusion Matrix:

Examining the confusion matrix is a much improved technique to assessing the performance of a classifier. The general idea is to count the number of times examples of class A are classified as class B. In a confusion matrix, each row represents an actual class, while each column represents an expected class.

	0	1
0	TP	FP
1	FN	TN

Figure 4. Confusion Matrix.

- True Positive (TP): [0][0] represents the values which are predicted to be true and are actually true.
- True Negative (TN): [1][1] represents the values which are predicted to be false and are actually false.
- False Positive (FP): [0][1] represents the values which are predicted to be false, but are true.
- False Negative (FN): [1][0] represents the values which are predicted to be true, but are false.

2.1.6. Accuracy Score:

Accuracy score is the measure of accuracy of a model.

$$Accracy\ Score = \frac{TP+TN}{TP+FP+FN+TN}$$

2.2. ALGORITHMS AND TECHNIQUES USED

2.2.1 Logistic Regression:

Logistic regression is a supervised learning classification algorithm. Logistic regression produces results in a binary format which is used to predict the outcome of a categorical dependent variable. So the outcome should be in discrete or categorical such as:

- 0 or 1
- True or False

- Yes or No

Based on the number of categories Logistic regression can be classified in three types:

- Binomial: Only two possible output i.e. 0 or 1, yes or no, true or false, win or loss.
- Multinomial: Target variable will have 3 or more than three possible types which are not ordered.
- Ordinal: Target variable deals with ordered category and each category can be given some specific values like 0,1,2,3.

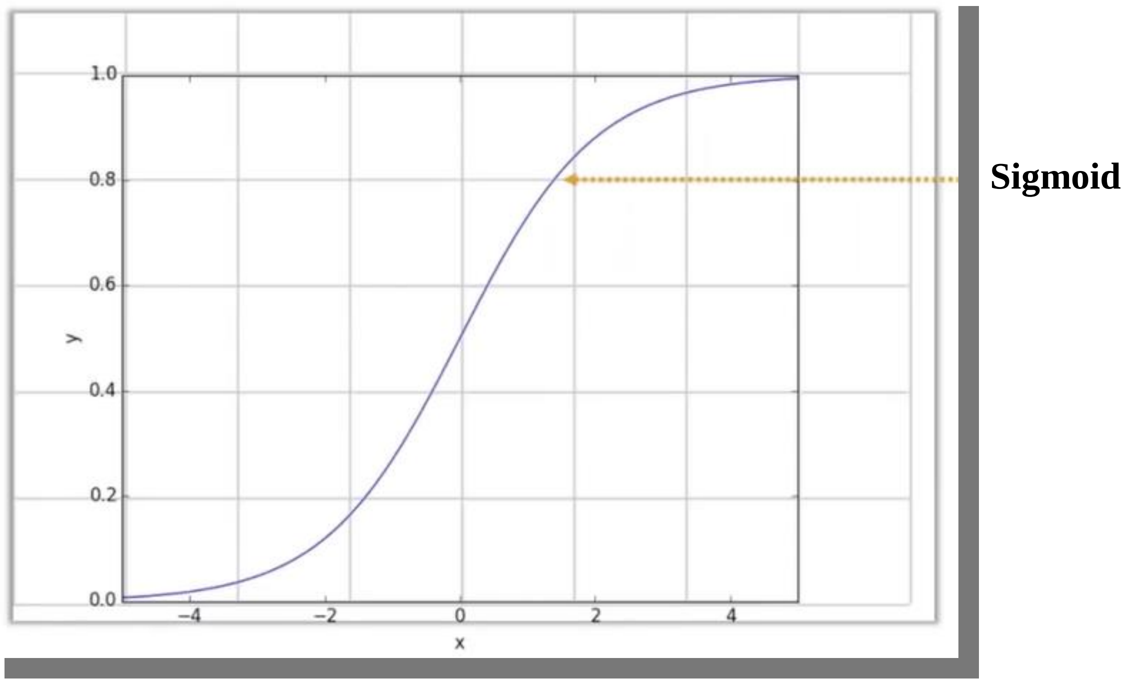


Figure 5. Sigmoid curve

Logistic regression models the data using the sigmoid function.

$$d(z) = \frac{1}{1 + e^{-z}}$$

Where, $z = y * W^T x$

Now you might be wondering how you'll know whether the value is 0 or 1, or true or false. So we have the concept of threshold value, which is nothing more than a number that reflects the likelihood of winning or losing.

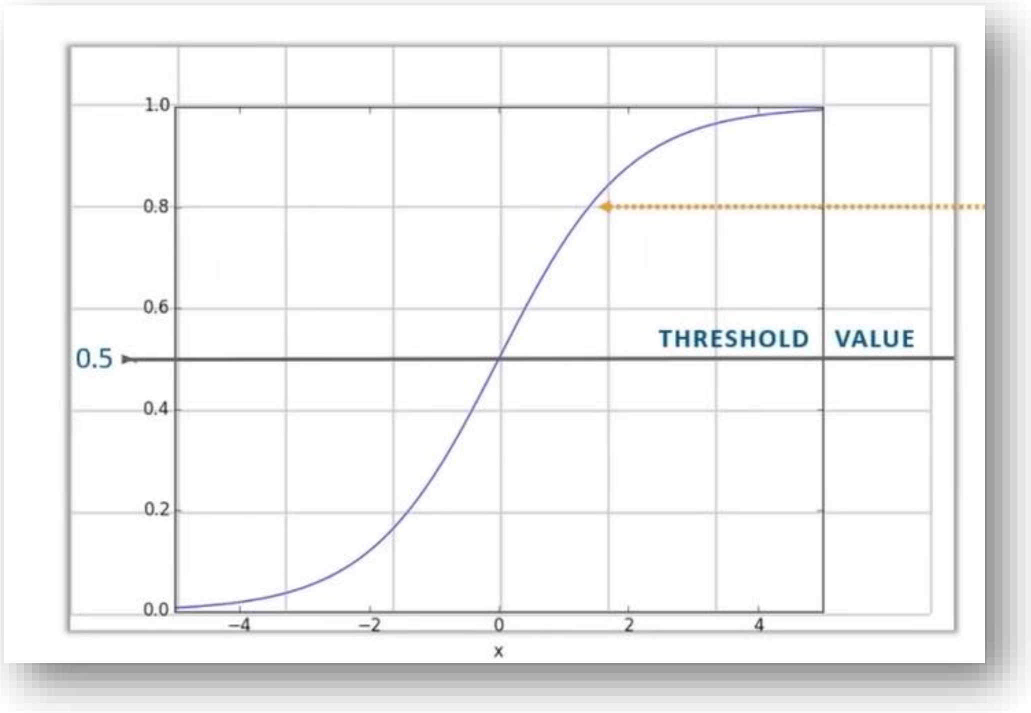


Figure 6. Threshold value

What does it indicate that we have a threshold value of 0.5 on the graph above (i.e. figure 6)? Simply said, any value larger than 0.5 is regarded as 1 and any value less than 0.5, including 0.5, is regarded as 0..

2.2.2. Decision Tree:

The most popular and powerful tool for classification and prediction is the decision tree, which is a supervised learning algorithm. This statement breaks the problem into smaller sub-problems and creates a tree-like structure with "yes" or "no" as the solution for each node based on specified requirements.

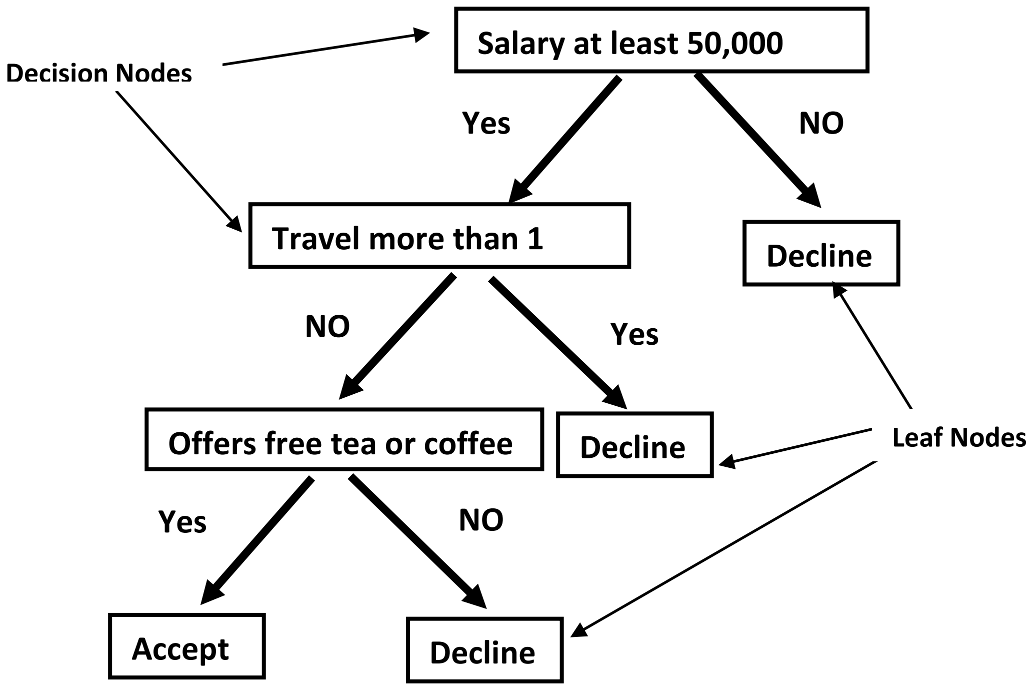


Figure 7. Should you accept a job offer?

Consider the example above, where our problem statement is whether or not we will accept a job offer. So in order to decide what we did is we make some list of decisions starting with the root node i.e. whether the company is providing 50,000 Rs per month if it is less than 50,000 than we are not accepting the offer if it is more than or equal to 50,000 than we will go to our next node or next decision i.e. whether we have to travel for more than one hour or not in case if it is more than one hour than we are not accepting the offer if it is less than one our than we are getting close to accepting the offer and further go to the next node or next decision i.e. whether the company offers free tea or coffee if it done than obviously we are accepting the offer or else we will decline the offer.

The important test in a decision tree is identifiable proof of the trait for the root node at each level. Property determination is the name for this cycle. We have two primary traits to choose from:

1. Information Gain
2. Gini impurity

1. **Information Gain:** The entropy changes when a node in a decision tree is used to segment the training set into smaller subsets. This change in entropy has a feature called information gain.

   Entropy: The impurity of a subjective assortment of models is depicted by entropy, which is the count of vulnerability of a random variable. The more information there is, the higher the entropy..

$$\boldsymbol{Entropy\ (s) = -P(yes)\log_2 P(yes)\ - P(No)\log_2 P(yes)}$$

   Where,

   - S is the total sample space
   - P(yes) is probability of yes
   - P(No) is probability of No

$$Gain(S, A) = Entropy(S) - \sum_{v \epsilon Values(A)} \frac{|S_v|}{|S|}.Entropy(S_v)$$

2. **Gini index:** It is a measure of impurity or purity that is employed in the construction of a decision tree. Low gini index attributes are recommended.

$$GiniIndex = 1 - \sum_j p_j^2$$

2.2.3. Random Forest:

Random Forest is an ensemble classifier that uses numerous decision trees and a technique called bagging to do both regression and classification tasks. The main idea is to use a combination of several decision trees to determine the output rather than relying on a single decision tree. By combining many models, ensemble learning improves machine learning results. In comparison to a single model, this methodology allows for the production of superior predictive execution. The main idea is to become acquainted with a group of classifiers (experts) and to allow them to vote. The most favourable outcome is then chosen or taken into account as the final result.

As a fundamental learning model, Random Forest uses a variety of decision trees. Row and feature sampling are done at random from the dataset, resulting in datasets for each model..

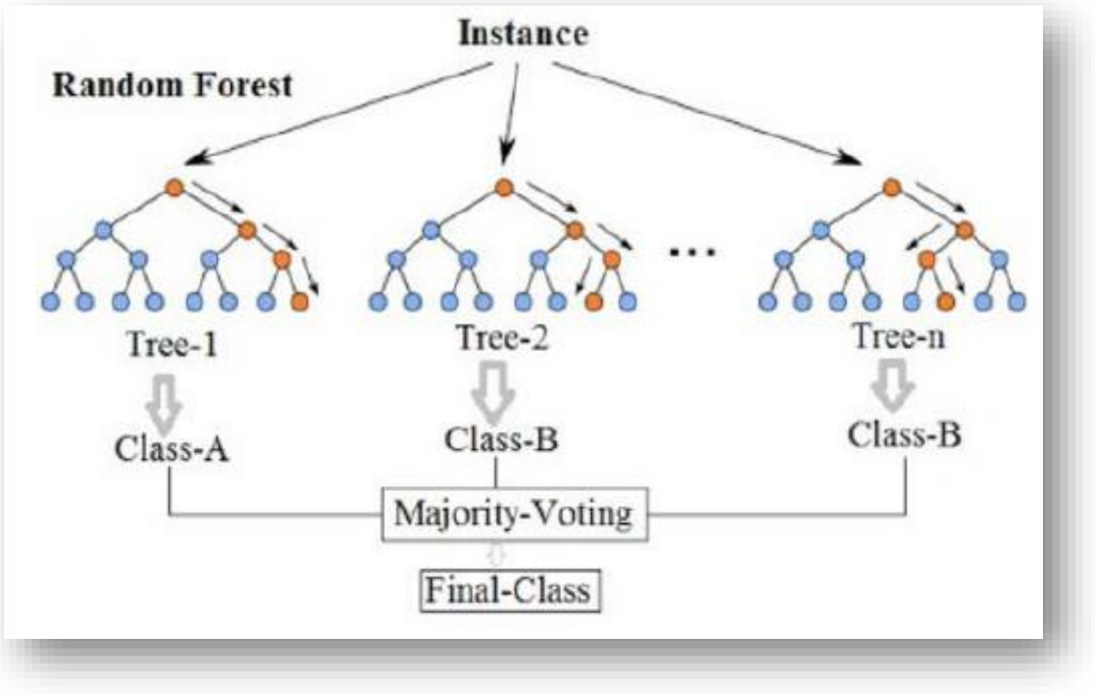


Figure 8. Random Forest.

From the first informative index or the original data, a large number of subsets are created by replacing perceptions. A subset of features is picked at random, and the feature with the best split is used to iteratively part the node. The tree has grown to its full potential. Reverse the procedure, and a forecast is made based on the collection of expectations from n trees..

2.2.4. Naïve Bayes

The Naive Bayes classifiers are a collection of calculations based on Bayes' Theorem. It's not a single computation, but rather a collection of calculations with a common standard, such as each pair of highlights being ranked being independent of the others.

The essential Naive Bayes hypothesis is that each component makes an equal and free commitment to the outcome.

We assume that no two features are dependent. For example, imagine we have F1, F2, F3, and Outcome in a dataset. According to independence, F1 has nothing to do with F2 and F3, F2 has nothing to do with F1 and F3, and F3 has nothing to do with F1 and F2. As a result, we may claim that each feature is self-contained.

Second, each feature is given equal weight, therefore feature F1 cannot define the Outcome by itself, and feature F2 cannot define the Outcome by itself. We needed all of the features to define the outcome. As a result, we can conclude that all qualities are equally weighted when it comes to establishing an Outcome.

Let us first define Bayes Theorem before moving on to the Nave Bayes formula as Naïve Bayes is based on Bayes theorem.

Bayes Theorem

The Bayes' Theorem calculates the probability of a function occurring given the probability of a previously occurring function..

$$\boldsymbol{P(A|B) = \frac{P(B|A)P(A)}{P(B)}}$$

Where,

- A and B are the events.
- We are attempting to discover the likelihood of occasion A, given the occasion B is valid.
- P (A) is the priori of A (the earlier likelihood, for example Likelihood of function before evidence is seen). The proof is a property estimation of an obscure instance (here, it is function B).
- P (A|B) is a posteriori likelihood of B, for example likelihood of function after proof is seen.

Now, with regards to our dataset, we can apply Bayes' theorem in following way:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

Where, y is class variable and X is a dependent feature vector (of size *n*) where:

$$X = (x_1, x_2, x_3, \ldots.., x_n)$$

Naive assumption

Currently, it is possible to apply a Nave Bayes model to the Bayes hypothesis, which is one of the highlights. As a result, we've divided proof into its free pieces.

Now, if any two events A and B are independent, then,

$$\boldsymbol{P(A,B) = P(A)P(B)}$$

Hence, we reach to the result:

$$P(y|x_1,...,x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

$$P(y|x_1,...,x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1)P(x_2)...P(x_n)}$$

Which can be expressed as:

We can now eliminate that term because the denominator remains constant for every given input:

We currently need to create a classification model. For this, we find the greatest extreme likelihood of a given arrangement of contributions for all possible estimations of the class variable y and derive the yield. This can be expressed mathematically as follows::

$$P(y|x_1,...,x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|y)$$

$$y = argmax_y P(y)\prod_{i=1}^{n} P(x_i|y)$$

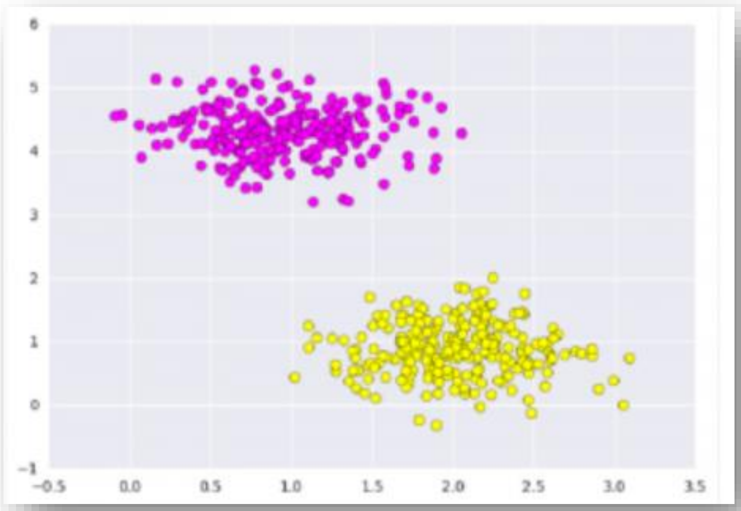
2.2.5. Support Vector Machine:

Support vector machines (SVMs, also known as support vector networks) are supervised learning models with a linked learning algorithm that analyse data for classification and regression analysis in machine learning.

An isolating hyperplane defines a Support Vector Machine (SVM), which is a discriminative classifier. Overall, the technique provides an ideal hyperplane that sorts new models given labelled training data (supervised learning).

A SVM model is a representation of the models as points in space, with the instances of the various classes partitioned by an unmistakable gap as large as is reasonably possible..

Figure 9. Support Vector Machine



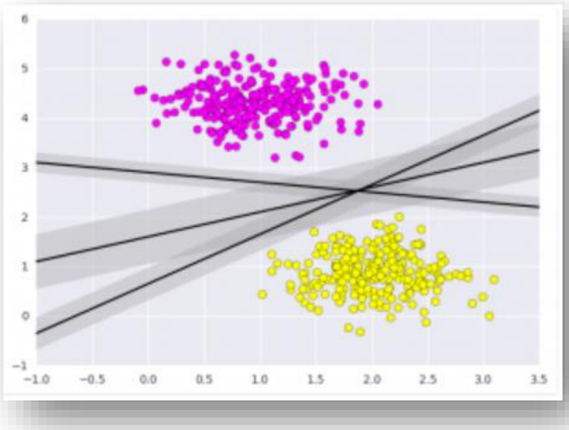


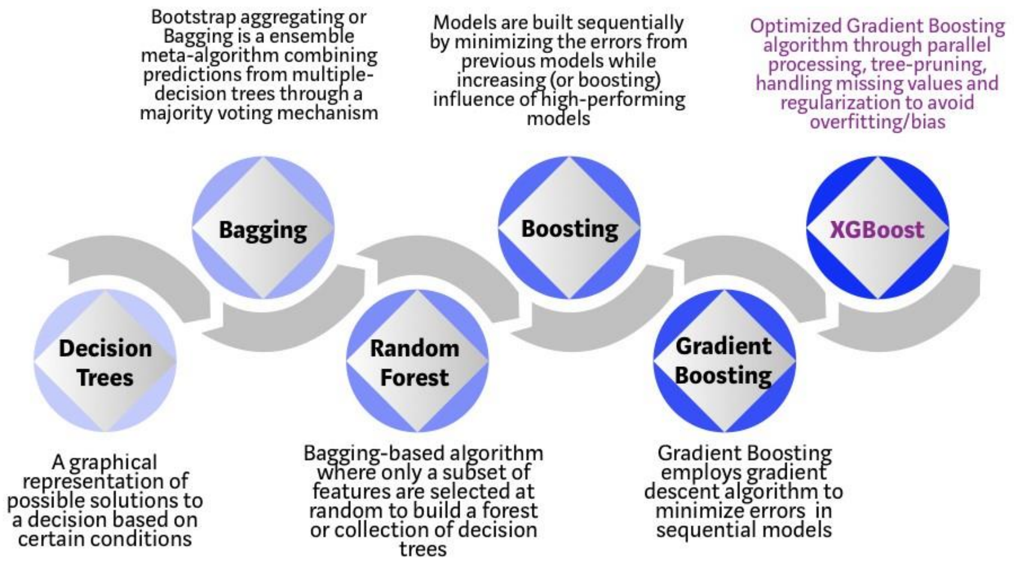
Figure 9.1. Support vector Machin-2

So, if we look at the diagram above, we can see that there are two types of outcomes, one marked by "pink" dots and the other by "yellow" dots. As shown in the diagram, the aim of SVM is to design a line or a hyperplane that can isolate both locations (Figure 9.1)

BOOSTING ALGORITHMS:

1. XG BOOST- XGBoost is a method of ensemble learning. It may not always be enough to depend on the outcomes of a single machine learning model. Ensemble learning is a method for combining the predictive abilities of numerous learners in a systematic way. The end result is a single model that combines the outputs of numerous models.

The foundation learners, or models that make up the ensemble, might be from the same learning algorithm or from distinct learning algorithms. Bagging and boosting are two types of ensemble learners that are commonly utilised. Though these two strategies may be used to a variety of statistical models, decision trees have been the most popular.


Bootstrap aggregating or Bagging is a ensemble meta-algorithm combining predictions from multiple-decision trees through a majority voting mechanism
Models are built sequentially by minimizing the errors from previous models while increasing (or boosting) influence of high-performing models
Optimized Gradient Boosting algorithm through parallel processing, tree-pruning, handling missing values and regularization to avoid overfitting/bias
Bagging
Boosting
XGBoost
Decision Trees
Random Forest
Gradient Boosting
A graphical representation of possible solutions to a decision based on certain conditions
Bagging-based algorithm where only a subset of features are selected at random to build a forest or collection of decision trees
Gradient Boosting employs gradient descent algorithm to minimize errors in sequential models

Unique features of XG Boost:

1. **Regularization**
2. **Handling sparse data**
3. **Weighted quantile sketch**
4. **Block structure for parallel learning**
5. **Cache awareness**
6. **Out-of-core computin**

2. CAT Boost- Yandex's CatBoost machine learning algorithm was just open-sourced. It's simple to interface with deep learning frameworks like TensorFlow from Google and Core ML from Apple. It can work with a variety of data formats to assist organisations address a variety of challenges. To top it off, it has the highest accuracy in the industry.

It is particularly effective in two ways:

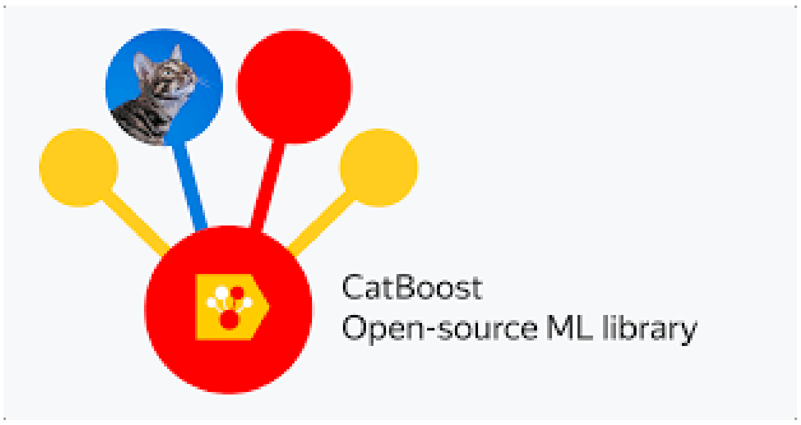
It provides robust out-of-the-box support for the more descriptive data formats that accompany many business challenges, and it produces state-of-the-art outcomes without the substantial data training that other machine learning approaches require.

The term "CatBoost" is derived from the phrases "Category" and "Boosting."

As previously stated, the library works well with a variety of data types, including audio, text, and picture, as well as historical data.

Advantages of CatBoost Library

1. **Performance**
2. **Handling Categorical features automatically**



3. **Robust**
4. **Easy to use**

3. DATA ANALYSIS

3.1. Importing all modules:

We needed to load certain crucial modules in order to develop a Machine Learning model.

Now that Python has a plethora of modules that may be used to create Machine Learning models, one must be specific in importing the model or, to put it another way, one must import the correct model in order to create a solid machine learning model.

The following is a list of the models that were utilised in this project..

1. Panda:

Wes McKinney designed Pandas, a high-level information control apparatus. It is built on the Numpy package, and the DataFrame is its primary data structure. DataFrames allow you to store and manipulate data in lines of perception and factor segments..

2. Numpy:

NumPy is a Python library for manipulating arrays. It also has the ability to work in the areas of straight polynomial math, fourier transformation, and networks. Travis Oliphant created NumPy in 2005. It is an open source project that you are free to use. Numerical Python is represented by NumPy. NumPy intends to provide a cluster object that is up to 50 times faster than standard Python records. The ndarray exhibit object in NumPy provides a wealth of supporting capabilities that make working with ndarray simple. Arrays are commonly used in data science, where speed and resources are important..

3. Matplotlib:

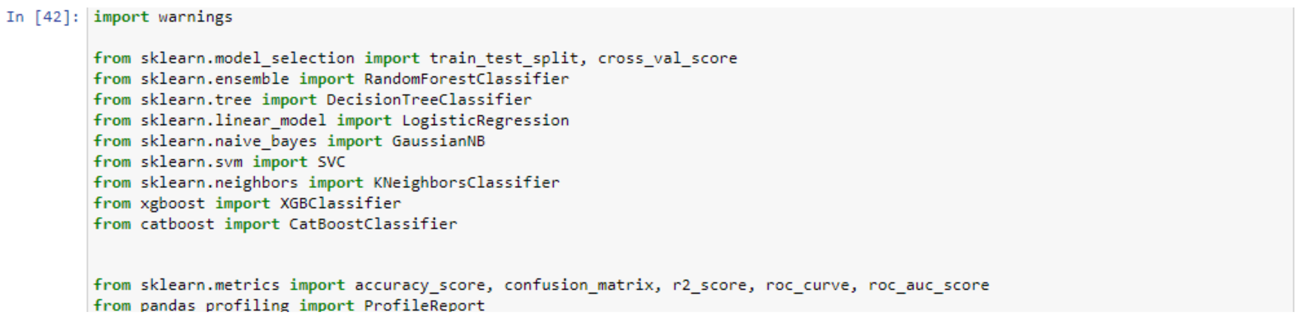
Matplotlib is a charting library for the Python programming language and NumPy, which is its mathematical augmentation. It provides an item-specific API for embedding plots into programmes using widely used GUI toolkits such as Tkinter, wxPython, Qt, or GTK+.t..

4. Seaborn:

Seaborn is a Python module for creating real designs. It builds on top of matplotlib and works hand-in-hand with pandas data structures. Seaborn encourages you to research and understand your data.

5. **Sklearn.**

Sci-kit learn popularly known as “sklearn” is a popular machine learning library for python.

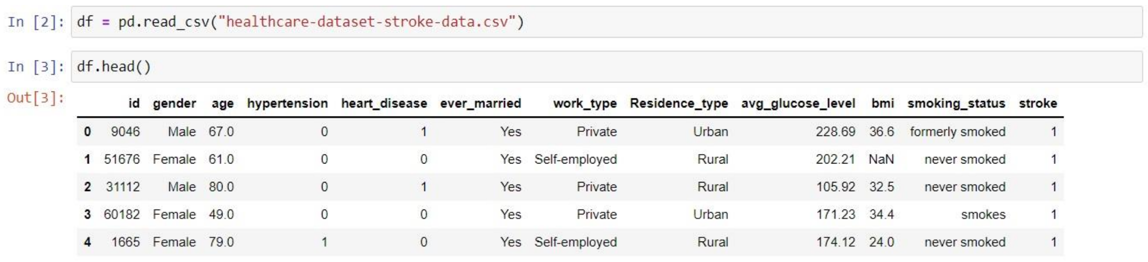


```
In [42]: import warnings

         from sklearn.model_selection import train_test_split, cross_val_score
         from sklearn.ensemble import RandomForestClassifier
         from sklearn.tree import DecisionTreeClassifier
         from sklearn.linear_model import LogisticRegression
         from sklearn.naive_bayes import GaussianNB
         from sklearn.svm import SVC
         from sklearn.neighbors import KNeighborsClassifier
         from xgboost import XGBClassifier
         from catboost import CatBoostClassifier


         from sklearn.metrics import accuracy_score, confusion_matrix, r2_score, roc_curve, roc_auc_score
         from pandas_profiling import ProfileReport
```

Figure 10. Importing Libraries and Modules

3.2. Loading the data and gathering Information

Now, the data is in .csv format so we will use the “read_csv” function from pandas library to read the data.



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

In [3]: df.head()
```

Out[3]:

	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

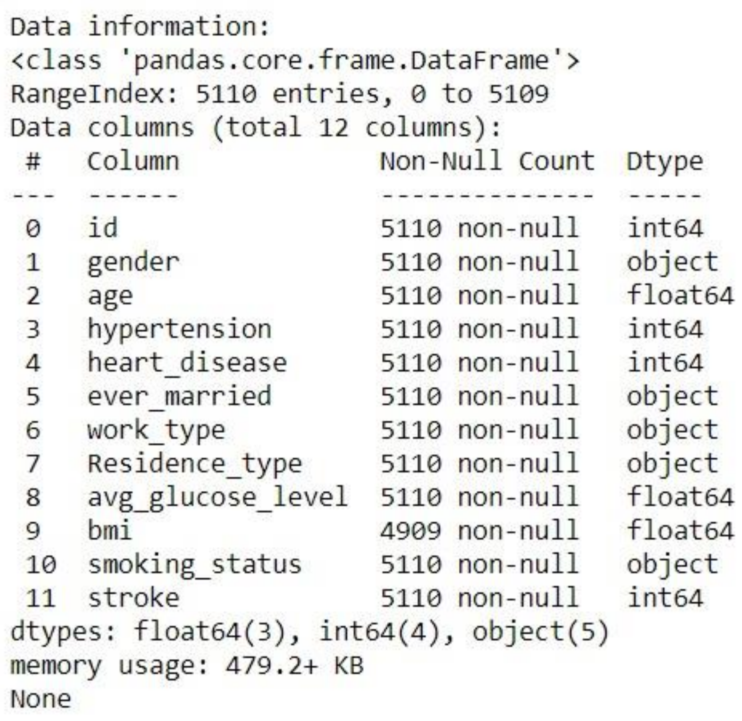
Figure 11. Loading the data.

In the image above, we can see that I've put all of the data into a variable called "data."

Now, we have several options for seeing all of the information contained within that csv file. We may simply input the name of the variable and see all of the information contained within the dataset, as seen in the above picture (i.e. Figure 11).

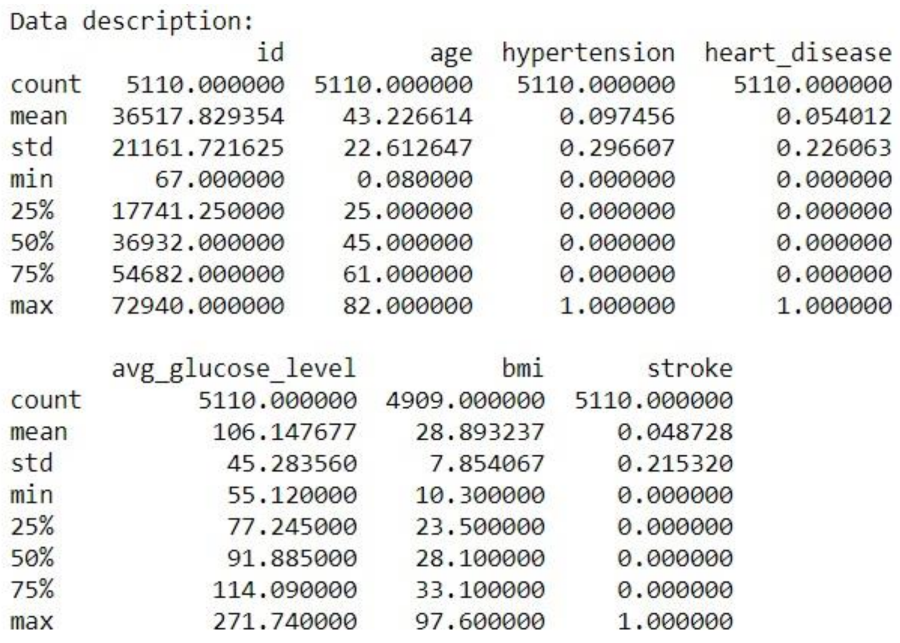
Now, if we only want to see the first few rows of data, we can use the "head()" method, which by default returns the first five rows of the dataset. If we input values like 10,20,30, etc. into the head function, it will return the first 10 rows, 20 rows, 30 rows, etc.

Now, if we want to examine more details about the dataset or obtain more information about it, we may do so in a variety of methods, such as using "data.info()" or "data.describe()," as seen in the examples below (i.e. figure 12 and figure 13).



```
Data information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 5110 non-null   int64
 1   gender             5110 non-null   object
 2   age                5110 non-null   float64
 3   hypertension       5110 non-null   int64
 4   heart_disease      5110 non-null   int64
 5   ever_married       5110 non-null   object
 6   work_type          5110 non-null   object
 7   Residence_type     5110 non-null   object
 8   avg_glucose_level  5110 non-null   float64
 9   bmi                4909 non-null   float64
 10  smoking_status     5110 non-null   object
 11  stroke             5110 non-null   int64
dtypes: float64(3), int64(4), object(5)
memory usage: 479.2+ KB
None
```

Figure 12. Data.info() function



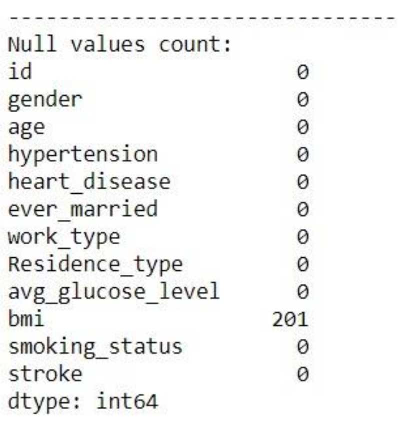
```
Data description:
                 id          age  hypertension  heart_disease
count   5110.000000  5110.000000   5110.000000    5110.000000
mean   36517.829354    43.226614      0.097456       0.054012
std    21161.721625    22.612647      0.296607       0.226063
min       67.000000     0.080000      0.000000       0.000000
25%    17741.250000    25.000000      0.000000       0.000000
50%    36932.000000    45.000000      0.000000       0.000000
75%    54682.000000    61.000000      0.000000       0.000000
max    72940.000000    82.000000      1.000000       1.000000

       avg_glucose_level          bmi       stroke
count        5110.000000  4909.000000  5110.000000
mean          106.147677    28.893237     0.048728
std            45.283560     7.854067     0.215320
min            55.120000    10.300000     0.000000
25%            77.245000    23.500000     0.000000
50%            91.885000    28.100000     0.000000
75%           114.090000    33.100000     0.000000
max           271.740000    97.600000     1.000000
```

Figure 13. Data.describe() function.

3.3. Data Pre-processing

The term "pre-preparing" refers to the alterations made to our data before applying it to the calculation. Data Pre-processing is a procedure for transforming raw data into a flawless data set. As a result, if information is obtained from numerous sources, it is done so in raw form, which is insufficient for the research.

The arrangement of data must be done properly in order to achieve better results from the used model in Machine Learning initiatives. Some preconfigured Machine Learning models demand data in a specified design, for example, Random Forest classification can not tolerate null values, so null values must be monitored from the start of the raw data collection.



```
------------------------------
Null values count:
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
```

Figure 14. Data.isnull().sum()

Another viewpoint is that data gathering should be organised in such a way that multiple Machine Learning and Deep Learning algorithms are used in a single informational collection, with the best of them being chosen.

We're looking for null values in the dataset in the image above (figure 14).

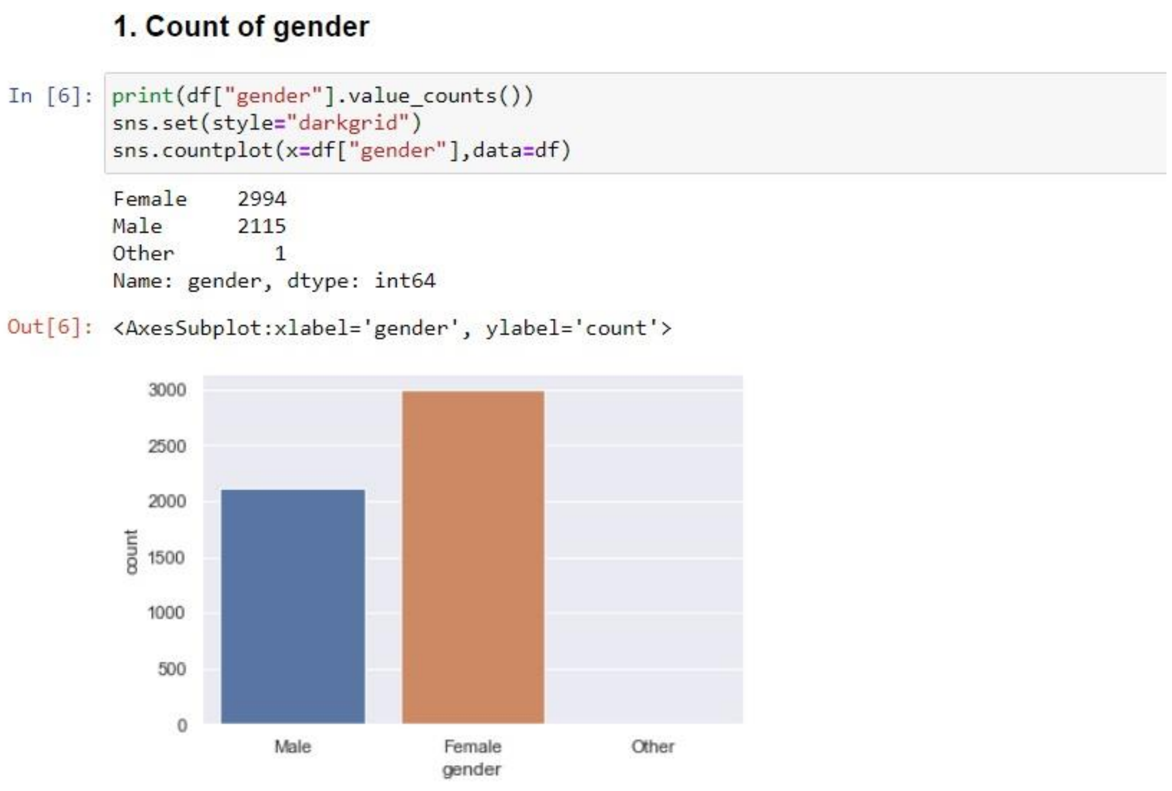
3.4. Data visualization

. The representation of information or data in a diagram, graph, or other visual layout is known as data visualisation. Machine learning makes it easier to lead investigations, such as prescient investigation, by allowing supporting representations to fill in the gaps.

It's usually a good idea to visualise your data so you can see how one feature affects another. Visualization assists you in selecting only the data that is necessary for making predictions.

3.5. Count Plot

The below figure will give you the total number of counts of patients having Brain stroke and persons not having brain stroke.



1. Count of gender

```
In [6]: print(df["gender"].value_counts())
        sns.set(style="darkgrid")
        sns.countplot(x=df["gender"],data=df)
```

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

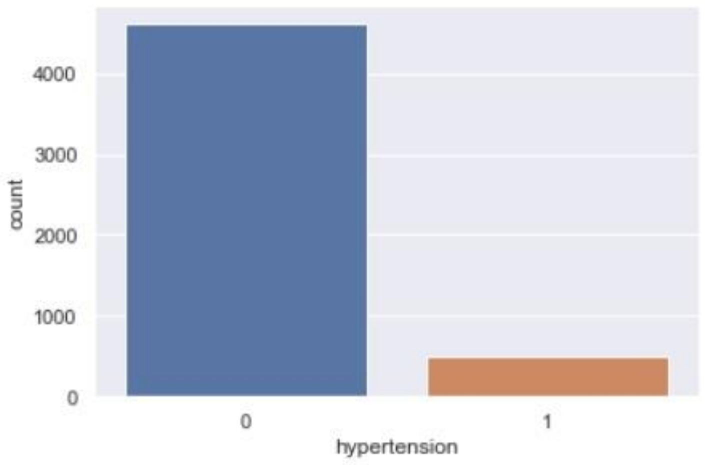
```
Out[6]: <AxesSubplot:xlabel='gender', ylabel='count'>
```

2. Count of hypertension

```
In [7]: print(df["hypertension"].value_counts())
        sns.countplot(x=df["hypertension"], data=df)
```

```
0    4612
1     498
Name: hypertension, dtype: int64
```

```
Out[7]: <AxesSubplot:xlabel='hypertension', ylabel='count'>
```

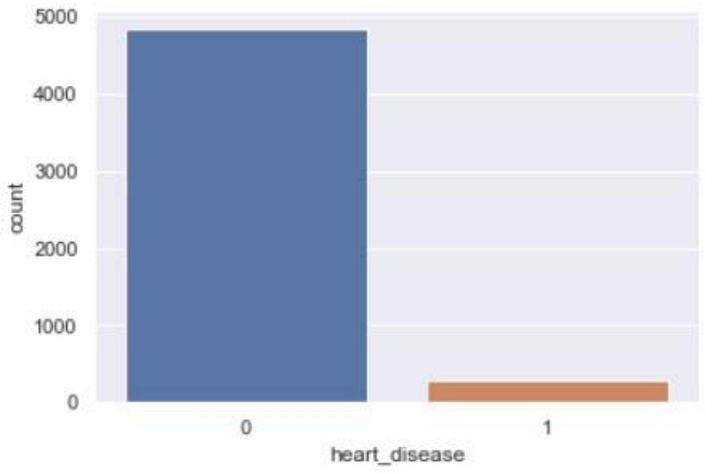


3. Heart Disease

```
In [8]: print(df["heart_disease"].value_counts())
        sns.countplot(x=df["heart_disease"], data=df)
```

```
0    4834
1     276
Name: heart_disease, dtype: int64
```

```
Out[8]: <AxesSubplot:xlabel='heart_disease', ylabel='count'>
```

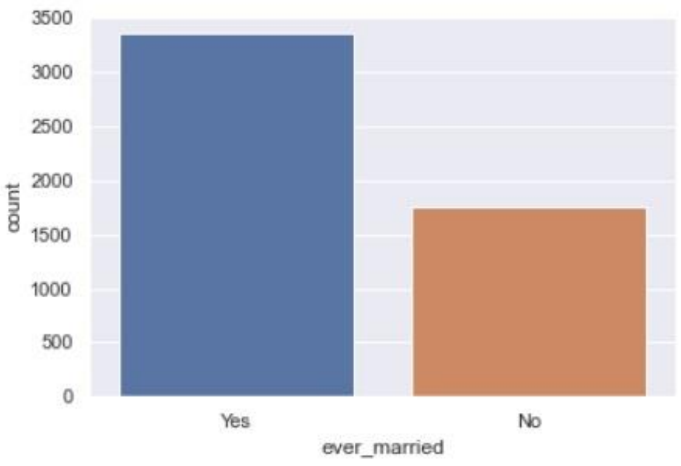


4. Married status

```
In [9]: print(df["ever_married"].value_counts())
        sns.countplot(x=df["ever_married"], data=df)
```

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

```
Out[9]: <AxesSubplot:xlabel='ever_married', ylabel='count'>
```

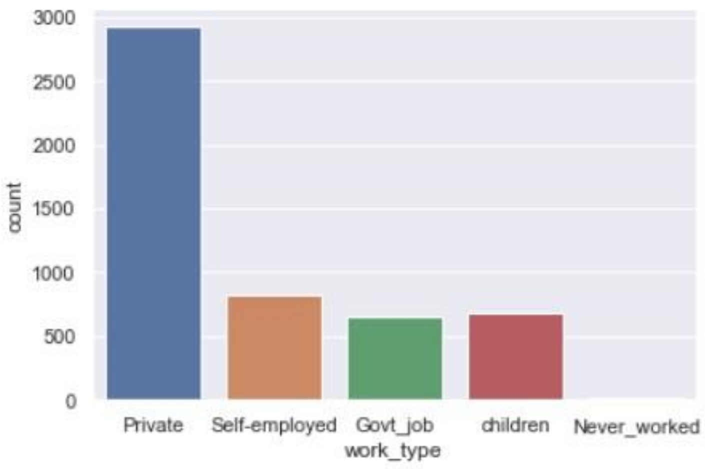


5. Work type

```
In [10]: print(df["work_type"].value_counts())
         sns.countplot(x=df["work_type"], data=df)
```

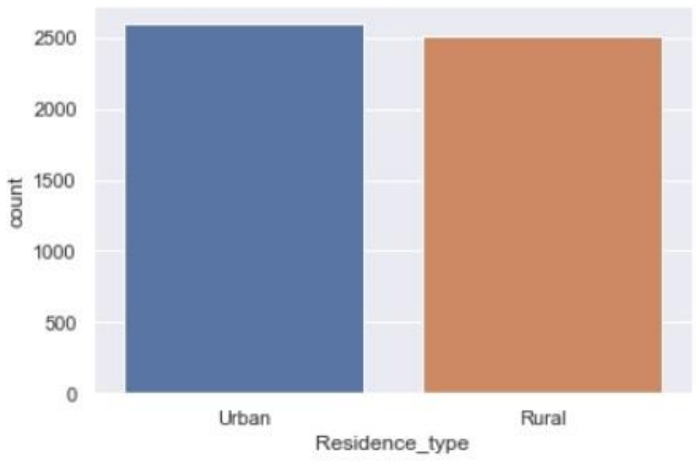
```
Private          2925
Self-employed     819
children          687
Govt_job          657
Never_worked       22
Name: work_type, dtype: int64
```

```
Out[10]: <AxesSubplot:xlabel='work_type', ylabel='count'>
```



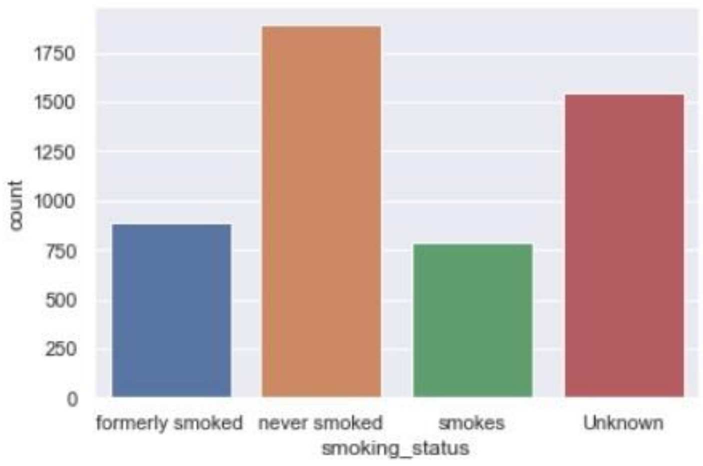
6. Residence type

```
In [11]: print(df["Residence_type"].value_counts())
         sns.countplot(x=df["Residence_type"], data=df)

         Urban     2596
         Rural     2514
         Name: Residence_type, dtype: int64

Out[11]: <AxesSubplot:xlabel='Residence_type', ylabel='count'>
```



7. Smoking type

```
In [12]: print(df["smoking_status"].value_counts())
         sns.countplot(x=df["smoking_status"], data=df)

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

Out[12]: <AxesSubplot:xlabel='smoking_status', ylabel='count'>
```

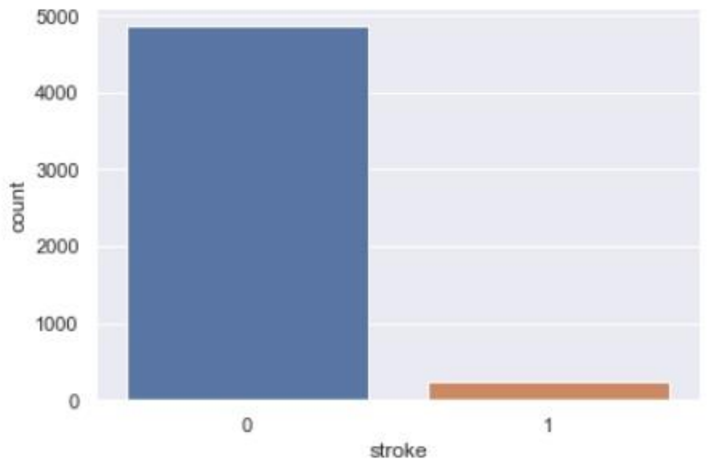


8. Stroke ¶

```
In [13]: print(df["stroke"].value_counts())
         sns.countplot(x=df["stroke"], data=df)
```

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

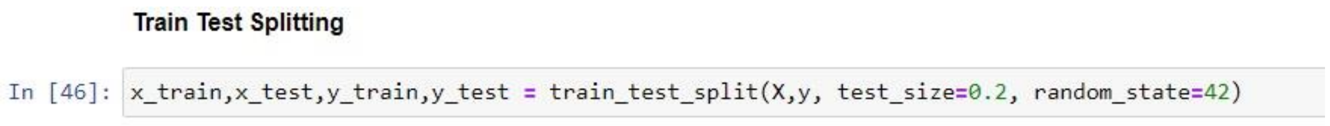
```
Out[13]: <AxesSubplot:xlabel='stroke', ylabel='count'>
```



As a result, we can see that we have 4861 people who have had no problems and 249 individuals who have had a brain stroke. "0" denotes patients who have had no brain stroke, while "1" denotes patients who have had a brain stroke.

3.6.Train Test Splitting

The train test split approach divides our dataset into two subsets: training data and testing data. Now the question is, how much data should be stored in training and testing data, or, in other words, what should be the ratio between the two? So, the easy answer is either a 70:30 ratio, in which 70% of the data is used for training and 30% for testing, or an 80:20 ratio, in which 80% of the data is used for training and 20% for testing. We'll use the "train test split()" function to split the dataset.



Train Test Splitting

```
In [46]: x_train,x_test,y_train,y_test = train_test_split(X,y, test_size=0.2, random_state=42)
```

Figure 25 . Splitting dataset of all features.

So, as you can see in the above picture, I used "test size = 0.2," which means that my testing data consists of 20% of the overall dataset's testing data and the remaining 80% is training data.

3.7.Training and Testing of models.

Now that we've successfully partitioned our data into training and testing subsets, it's time to develop and test those datasets using various Machine Learning techniques.

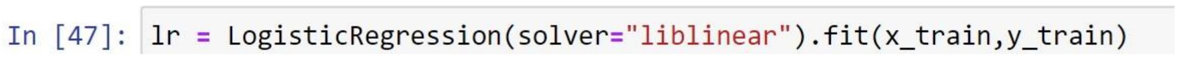
4. IMPLIMENTATION

4.1. Logistic Regression

Logistic regression is the first machine learning approach I used to develop the brain stroke prediction model. So, as shown in the figure below (figure 26), we acquire an accuracy of 81 percent when we employ all of the features in the data set.

Figure 26. Logistic regression.

Now, let’s check the confusion matrix of Logistic regression model we are considering the

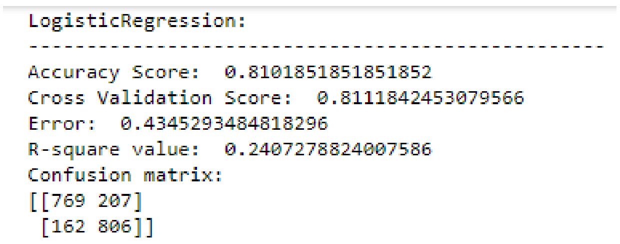


```
In [47]: lr = LogisticRegression(solver="liblinear").fit(x_train,y_train)
```

model having high accuracy i.e. the all feature one.

Figure 27. Confusion matrix of Logistic regression.

So, from the confusion matrix we can conclude the followings.

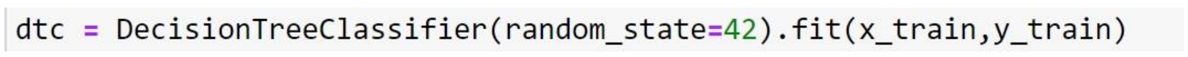


```
LogisticRegression:
-------------------------------------------------
Accuracy Score:  0.8101851851851852
Cross Validation Score:  0.8111842453079566
Error:  0.4345293484818296
R-square value:  0.2407278824007586
Confusion matrix:
[[769 207]
 [162 806]]
```

- [0][0] we have 769 people who are having a stroke and who have been appropriately classified as heat disease patients by our approach.
- [0][1] We have 207 patients that have no disease but have been recognised as brain stroke patients by our model.
- [1][0] we are having 162 Patients who are suffering a brain stroke, although our model predicts that they will not have one.
- [1][1] we have 806 patients who haven’t had a brain stroke and who have been appropriately diagnosed as such by our model.

4.2. Decision Tree

Now that we've completed our first algorithm, let's move on to our second algorithm, the decision tree. So, for the prediction, we'll utilise "DecisionTreeClassifier()" for the decision tree model..

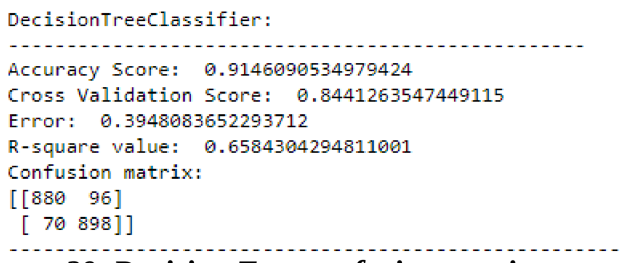


```
dtc = DecisionTreeClassifier(random_state=42).fit(x_train,y_train)
```

Figure 28. Decision Tree

So, when we utilise a decision tree as our model, we can see that we achieve a 91 percent accuracy. As can be seen, we have a high level of accuracy when compared to Logistic Regression. As a result, we will use the Decision Tree Model rather than the Logistic Regression Model to obtain accurate results.

Now, let’s see the confusion matrix for the same.



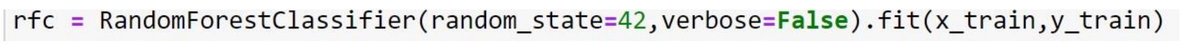
```
DecisionTreeClassifier:
-------------------------------------------------
Accuracy Score:  0.9146090534979424
Cross Validation Score:  0.8441263547449115
Error:  0.3948083652293712
R-square value:  0.6584304294811001
Confusion matrix:
[[880  96]
 [ 70 898]]
----------------------------------------------------
```

Figure 29. Decision Tree confusion matrix

- [0][0] we have 880 people who are having a stroke and who have been appropriately classified as heat disease patients by our approach.
- [0][1] We have 96 patients that have no disease but have been recognised as brain stroke patients by our model.
- [1][0] we are having 70 patients who are suffering a brain stroke, although our model predicts that they will not have one.
- [1][1] we have 898 patients who haven’t had a brain stroke and who have been appropriately diagnosed as such by our model.

4.3.Random Forest

Now we'll move on to our next algorithm, Random Forest. So I'll use "RandomForestClassifier()" for this.



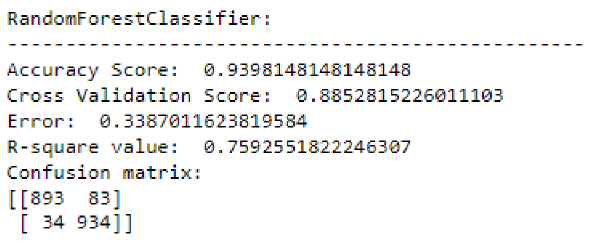
```
rfc = RandomForestClassifier(random_state=42,verbose=False).fit(x_train,y_train)
```

Figure 30. Random Forest

As a result, we can see that when we utilise random forest as our model, we achieve a 93 percent accuracy. So, while comparing Decision Tree and Logistic Regression, we can observe that we are achieving a high level of accuracy in random forest.

So far, Random Forest has shown to be the most effective.

Now, let’s check the confusion matrix for the same.



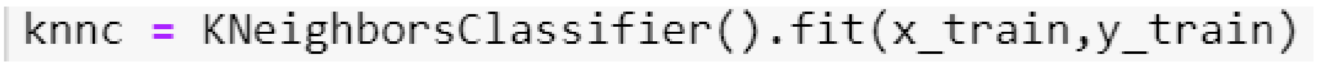
```
RandomForestClassifier:
-------------------------------------------------
Accuracy Score:  0.9398148148148148
Cross Validation Score:  0.8852815226011103
Error:  0.3387011623819584
R-square value:  0.7592551822246307
Confusion matrix:
[[893  83]
 [ 34 934]]
```

Figure 31. Random forest confusion matrix.

- [0][0] we have 893 people who are having a stroke and who have been appropriately classified as heat disease patients by our approach.
- [0][1] We have 83 patients that have no disease but have been recognised as brain stroke patients by our model.
- [1][0] we are having 34 patients who are suffering a brain stroke, although our model predicts that they will not have one.
- [1][1] we have 934 patients who haven’t had a brain stroke and who have been appropriately diagnosed as such by our model.

4.4. K-Nearest Neighbor(KNN)

Now, we will use K-Nearest Neighbor(KNN) algorithm as our next algorithm

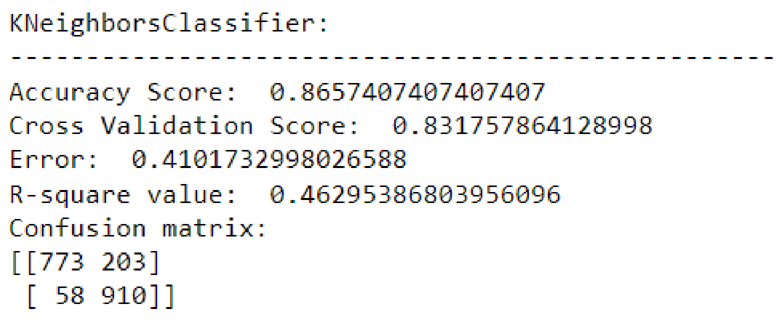


```
knnc = KNeighborsClassifier().fit(x_train,y_train)
```

Figure 32. KNN

So, when we utilise KNN algorithm to build our model, we can see that we achieve a 86 percent accuracy. Now comparing KNN with the like random forest and decision tree we can see that random forest and decision tree outperforms KNN in terms of accuracy. So we can that our KNN model didn't perform so well.

Now, let's check the confusion matrix for the KNN model



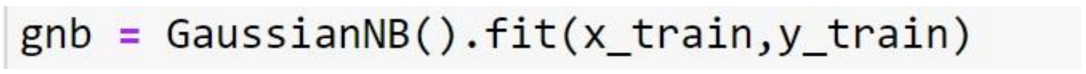
```
KNeighborsClassifier:
--------------------------------------------------
Accuracy Score:  0.8657407407407407
Cross Validation Score:  0.831757864128998
Error:  0.4101732998026588
R-square value:  0.46295386803956096
Confusion matrix:
[[773 203]
 [ 58 910]]
```

Figure 33. KNN Confusion Matrix.

- [0][0] we have 773 people who are having a stroke and who have been appropriately classified as heat disease patients by our approach.
- [0][1] We have 203 patients that have no disease but have been recognised as brain stroke patients by our model.
- [1][0] we are having 58 patients who are suffering a brain stroke, although our model predicts that they will not have one.
- [1][1] we have 910 patients who haven't had a brain stroke and who have been appropriately diagnosed as such by our model.

4.5 Naïve Bayes

Moving with our next algorithm i.e. Naïve Bayes. So, for this I have used "GaussianNB()".

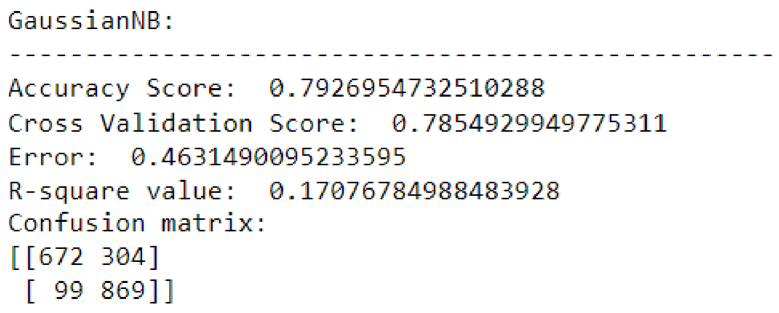


```
gnb = GaussianNB().fit(x_train,y_train)
```

Figure 34. Naïve Bayes

So in Naïve Bayes algorithm we see that is gives an accuracy of 79%. So we can see that this model didn't perform so well as compare with all the above algorithms.

Now, let's check the confusion matrix for the same.

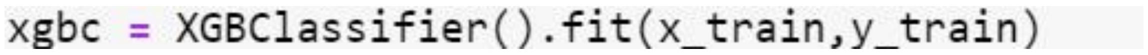


```
GaussianNB:
--------------------------------------------------
Accuracy Score:  0.7926954732510288
Cross Validation Score:  0.7854929949775311
Error:  0.4631490095233595
R-square value:  0.17076784988483928
Confusion matrix:
[[672 304]
 [ 99 869]]
```

Figure 35. Naïve Bayes confusion Matrix.

- [0][0] we have 672 people who are having a stroke and who have been appropriately classified as heat disease patients by our approach.
- [0][1] We have 304 patients that have no disease but have been recognised as brain stroke patients by our model.
- [1][0] we are having 99 patients who are suffering a brain stroke, although our model predicts that they will not have one.
- [1][1] we have 869 patients who haven't had a brain stroke and who have been appropriately diagnosed as such by our model.

4.6 XGBoost(Boosting Algorithm)

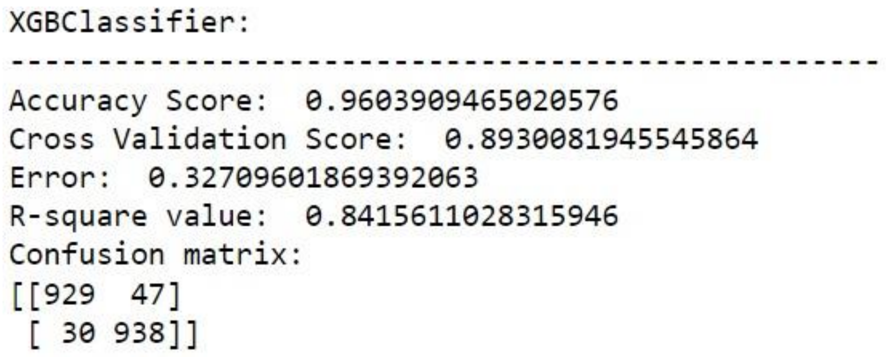


```
xgbc = XGBClassifier().fit(x_train,y_train)
```

Figure 36. XGBoost

Here we use boosting algorithm XGBoost to check the maximum accuracy of the model can be achieved from the dataset

Now, let’s check the confusion matrix for XGBoost Classifier



```
XGBClassifier:
----------------------------------------------------
Accuracy Score:  0.9603909465020576
Cross Validation Score:  0.8930081945545864
Error:  0.32709601869392063
R-square value:  0.8415611028315946
Confusion matrix:
[[929  47]
 [ 30 938]]
```

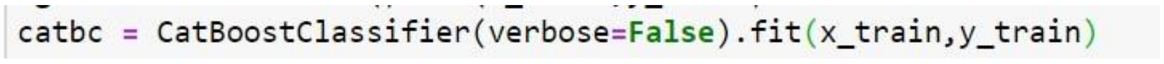
Figure 37. XGBoost confusion Matrix.

- [0][0] we have 929 people who are having a stroke and who have been appropriately classified as heat disease patients by our approach.
- [0][1] We have 47 patients that have no disease but have been recognised as brain stroke patients by our model.
- [1][0] we are having 30 patients who are suffering a brain stroke, although our model predicts that they will not have one.
- [1][1] we have 938 patients who haven’t had a brain stroke and who have been appropriately diagnosed as such by our model.

Even though we achieved an accuracy of 96 percent we will refrain from using this boosting algorithm as the algorithm sometimes overfits the dataset so that can lead to hampering of the overall result.

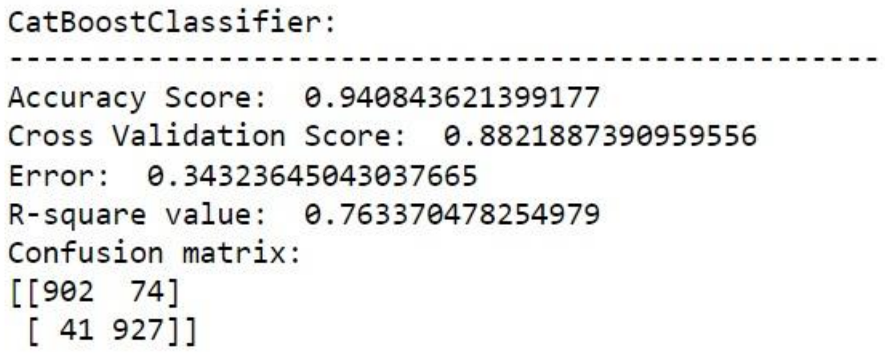
4.7 CAT Boost(Boosting Algorithm)

Now we will use our second boosting algorithm which is CAT Boost



```
catbc = CatBoostClassifier(verbose=False).fit(x_train,y_train)
```

Figure 38. CatBoost



```
CatBoostClassifier:
----------------------------------------------------
Accuracy Score:  0.940843621399177
Cross Validation Score:  0.8821887390959556
Error:  0.34323645043037665
R-square value:  0.763370478254979
Confusion matrix:
[[902  74]
 [ 41 927]]
```

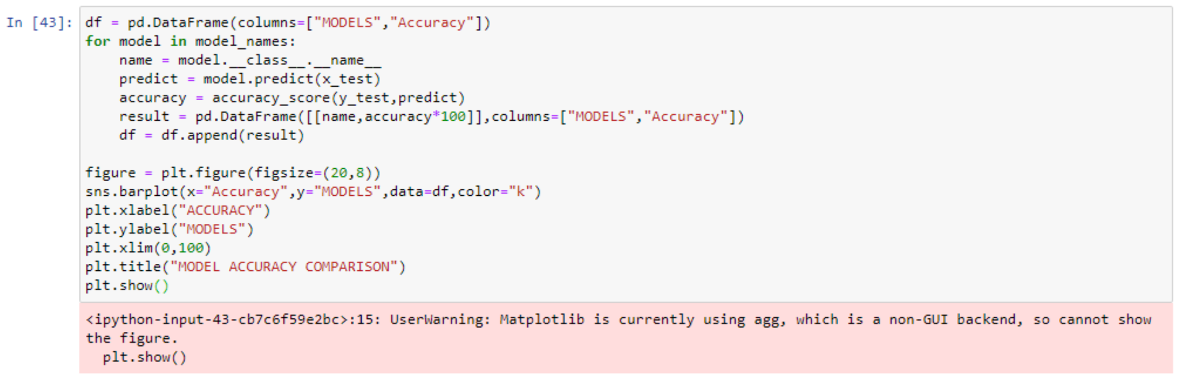
Figure 39. CatBoost Confusion Matrix

Now, let’s check the confusion matrix for CatBoost Classifier

- [0][0] we have 902 people who are having a stroke and who have been appropriately classified as heat disease patients by our approach.
- [0][1] We have 74 patients that have no disease but have been recognised as brain stroke patients by our model.
- [1][0] we are having 41 patients who are suffering a brain stroke, although our model predicts that they will not have one.
- [1][1] we have 927 patients who haven’t had a brain stroke and who have been appropriately diagnosed as such by our model.

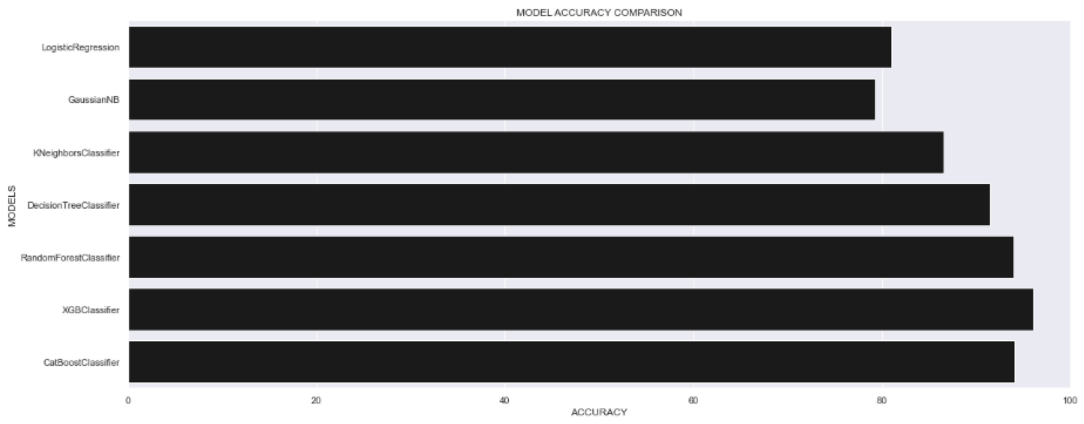
Even though we achieved an accuracy of 94 percent using CatBoost Algorithm, we will refrain from using this boosting algorithm as the algorithm sometimes overfits the dataset so that can lead to hampering of the overall result.
We will prefer using normal machine learning algorithms over boosting algorithms.

5. MODEL ACCURACY COMPARISON

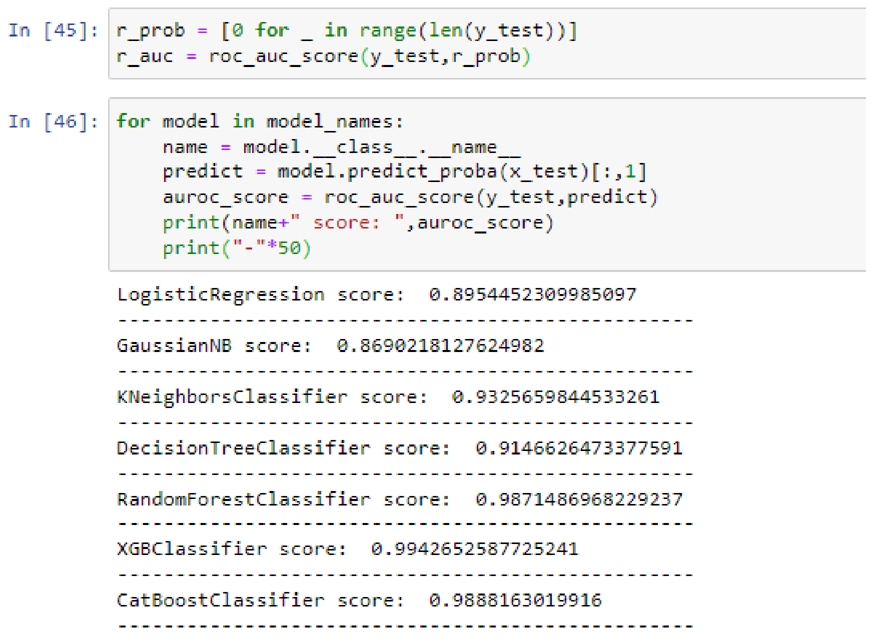


```
In [43]: df = pd.DataFrame(columns=["MODELS","Accuracy"])
         for model in model_names:
             name = model.__class__.__name__
             predict = model.predict(x_test)
             accuracy = accuracy_score(y_test,predict)
             result = pd.DataFrame([[name,accuracy*100]],columns=["MODELS","Accuracy"])
             df = df.append(result)

         figure = plt.figure(figsize=(20,8))
         sns.barplot(x="Accuracy",y="MODELS",data=df,color="k")
         plt.xlabel("ACCURACY")
         plt.ylabel("MODELS")
         plt.xlim(0,100)
         plt.title("MODEL ACCURACY COMPARISON")
         plt.show()
```

```
<ipython-input-43-cb7c6f59e2bc>:15: UserWarning: Matplotlib is currently using agg, which is a non-GUI backend, so cannot show
the figure.
  plt.show()
```

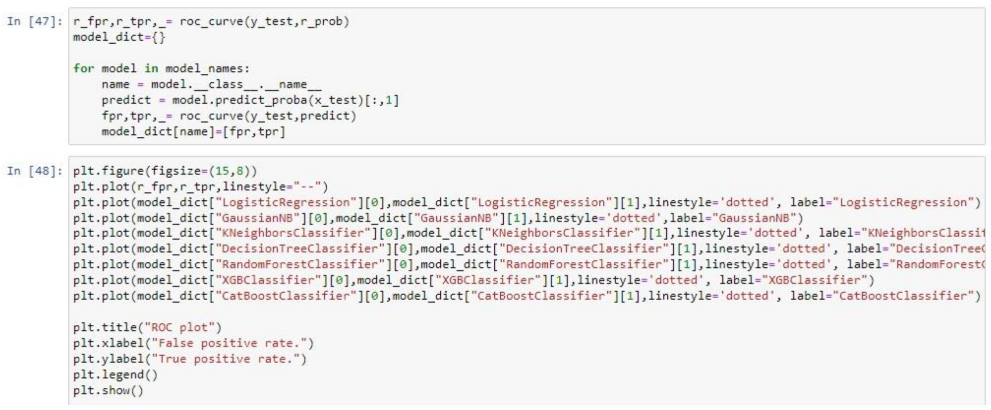


6. AUROC SCORE

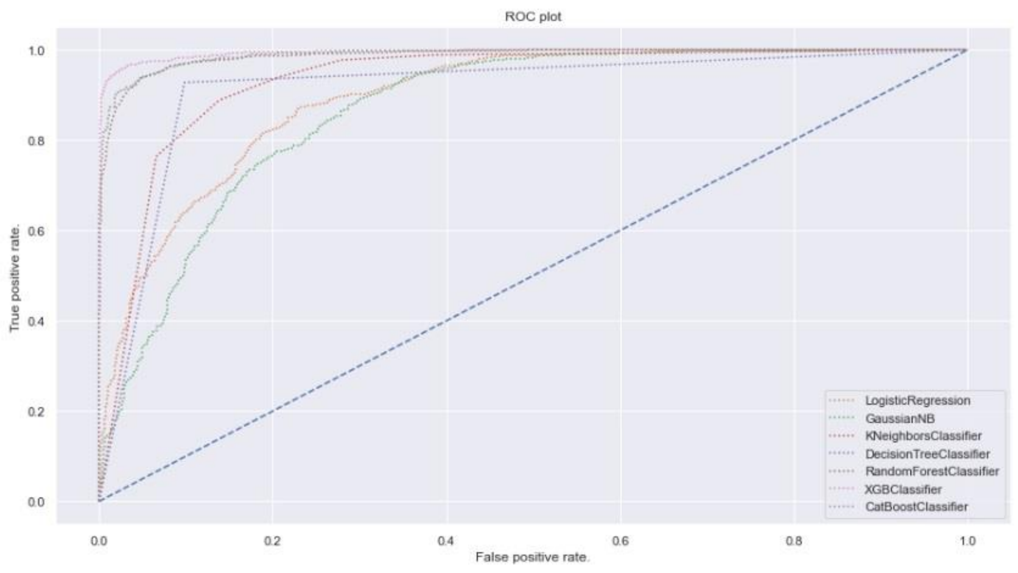


```
In [45]: r_prob = [0 for _ in range(len(y_test))]
         r_auc = roc_auc_score(y_test,r_prob)

In [46]: for model in model_names:
             name = model.__class__.__name__
             predict = model.predict_proba(x_test)[:,1]
             auroc_score = roc_auc_score(y_test,predict)
             print(name+" score: ",auroc_score)
             print("-"*50)

         LogisticRegression score:  0.8954452309985097
         --------------------------------------------------
         GaussianNB score:  0.8690218127624982
         --------------------------------------------------
         KNeighborsClassifier score:  0.9325659844533261
         --------------------------------------------------
         DecisionTreeClassifier score:  0.9146626473377591
         --------------------------------------------------
         RandomForestClassifier score:  0.9871486968229237
         --------------------------------------------------
         XGBClassifier score:  0.9942652587725241
         --------------------------------------------------
         CatBoostClassifier score:  0.9888163019916
         --------------------------------------------------
```

7. ROC CURVE



```
In [47]: r_fpr,r_tpr,_= roc_curve(y_test,r_prob)
         model_dict={}

         for model in model_names:
             name = model.__class__.__name__
             predict = model.predict_proba(x_test)[:,1]
             fpr,tpr,_= roc_curve(y_test,predict)
             model_dict[name]=[fpr,tpr]

In [48]: plt.figure(figsize=(15,8))
         plt.plot(r_fpr,r_tpr,linestyle="--")
         plt.plot(model_dict["LogisticRegression"][0],model_dict["LogisticRegression"][1],linestyle='dotted', label="LogisticRegression")
         plt.plot(model_dict["GaussianNB"][0],model_dict["GaussianNB"][1],linestyle='dotted',label="GaussianNB")
         plt.plot(model_dict["KNeighborsClassifier"][0],model_dict["KNeighborsClassifier"][1],linestyle='dotted', label="KNeighborsClassi
         plt.plot(model_dict["DecisionTreeClassifier"][0],model_dict["DecisionTreeClassifier"][1],linestyle='dotted', label="DecisionTree(
         plt.plot(model_dict["RandomForestClassifier"][0],model_dict["RandomForestClassifier"][1],linestyle='dotted', label="RandomForest(
         plt.plot(model_dict["XGBClassifier"][0],model_dict["XGBClassifier"][1],linestyle='dotted', label="XGBClassifier")
         plt.plot(model_dict["CatBoostClassifier"][0],model_dict["CatBoostClassifier"][1],linestyle='dotted', label="CatBoostClassifier")

         plt.title("ROC plot")
         plt.xlabel("False positive rate.")
         plt.ylabel("True positive rate.")
         plt.legend()
         plt.show()
```



8. CONCLUSION

The goal of this study is to look into several Machine Learning algorithms that can be used in computerised brain stroke prediction frameworks. This article characterises various techniques and Machine Learning classifiers that have recently emerged for effective and compelling Brain stroke analysis. According to the research, the Random Forest has also performed well, with a score of 93 percent.

Taking everything into account, as evidenced by the literature review, I believe that only a minor breakthrough has been made in the development of predictive models for brain stroke patients, and that there is a need for more complex and combinational models to improve the accuracy of predicting the early stages of brain stroke.

9. FUTURE WORK

We can use this model in the future to analyse different datasets by simply changing the name of the dataset file and the features provided in the training module. We can also use this model to determine what type of brain stroke a person is having once our model has predicted that a person is having a brain stroke. We can switch from Machine Learning algorithms to Deep Learning algorithms in the future to compute more complex datasets and even work on a combination of multiple algorithms to improve the accuracy score, allowing for much more accurate prediction and, based on that prediction, proper treatment of the patient.

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. Ordonez, Carlos. "Association rule discovery with the train and test approach for brain stroke prediction." *IEEE Transactions on Information Technology in Biomedicine* 10, no. 2 (2006): 334-343.
4. B Shantakumar, Y S Patil, Kumaraswamy “Intelligent and Effective Heart Attack Prediction System Using Data Mining and Artificial Neural Network” European Journal of Scientific Research, volume 31, issue 4, p. 642 – 656 Posted: 2009
5. Le Duff, Franck, Cristian Muntean, Marc Cuggia, and Philippe Mabo. "Predicting survival causes after out of hospital cardiac arrest using data mining method." In *Medinfo*, pp. 1256-1259. 2004.
6. Parthiban, Latha, and R. Subramanian. "Intelligent brain stroke prediction system using CANFIS and genetic algorithm." *International Journal of Biological, Biomedical and Medical Sciences* 3, no. 3 (2008).
7. Noh, Kiyong, Heon Gyu Lee, Ho-Sun Shon, Bum Ju Lee, and Keun Ho Ryu. "Associative classification approach for diagnosing cardiovascular disease." In *Intelligent computing in signal processing and pattern recognition*, pp. 721-727. Springer, Berlin, Heidelberg, 2006.
8. Shao, Yuehjen E., Chia-Ding Hou, and Chih-Chou Chiu. "Hybrid intelligent modeling schemes for brain stroke classification." *Applied Soft Computing* 14 (2014): 47-52.
9. Palaniappan, Sellappan, and Rafiah Awang. "Intelligent brain stroke prediction system using data mining techniques." In *2008 IEEE/ACS international conference on computer systems and applications*, pp. 108-115. IEEE, 2008.
10. Mahboob, Tahira, Rida Irfan, and Bazelah Ghaffar. "Evaluating ensemble prediction of coronary brain stroke using receiver operating characteristics." In *2017 Internet Technologies and Applications (ITA)*, pp. 110-115. IEEE, 2017.
11. Bashir, Saba, Usman Qamar, and M. Younus Javed. "An ensemble based decision support framework for intelligent brain stroke diagnosis." In *International Conference on Information Society (i-Society 2014)*, pp. 259-264. IEEE, 2014.
12. Irfan-ul-Haq, Ammar Asjad Raja, Madiha Guftar, Tamim Ahmed Khan, and Dominik Greibl. "Intelligent Syncope Disease Prediction Framework using DM-Ensemble Techniques."
13. Ganji, Mostafa Fathi, and Mohammad Saniee Abadeh. "Using fuzzy ant colony optimization for diagnosis of diabetes disease." In *2010 18th Iranian Conference on Electrical Engineering*, pp. 501-505. IEEE, 2010.
14. Rahman, Quazi Abidur, Larisa G. Tereshchenko, Matthew Kongkatong, Theodore Abraham, M. Roselle Abraham, and Hagit Shatkay. "Utilizing ECG-based heartbeat classification for hypertrophic cardiomyopathy identification." *IEEE transactions on nanobioscience* 14, no. 5 (2015): 505-512.

15. Saini, Ridhi, Namita Bindal, and Puneet Bansal. "Classification of brain strokes from ECG signals using wavelet transform and kNN classifier." In *International Conference on Computing, Communication & Automation*, pp. 1208-1215. IEEE, 2015.

16. Singh, Manpreet, Levi Monteiro Martins, Patrick Joanis, and Vijay K. Mago. "Building a cardiovascular disease predictive model using structural equation model & fuzzy cognitive map." In *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 1377-1382. IEEE, 2016.

17. Kavitha, R., and E. Kannan. "An efficient framework for brain stroke classification using feature extraction and feature selection technique in data mining." In *2016 international conference on emerging trends in engineering, technology and science (icetets)*, pp. 1-5. IEEE, 2016.

18. Kanchan, B. Dhomse, and M. Mahale Kishor. "Study of machine learning algorithms for special disease prediction using principal of component analysis." In *2016 international conference on global trends in signal processing, information computing and communication (ICGTSPICC)*, pp. 5-10. IEEE, 2016.

19. Rajathi, S., and G. Radhamani. "Prediction and analysis of Rheumatic brain stroke using kNN classification with ACO." In *2016 International Conference on Data Mining and Advanced Computing (SAPIENCE)*, pp. 68-73. IEEE, 2016.