*A Major Project Report*

*on*

**STROKE PREDICTION USING MACHINE LEARNING**

*submitted in partial fulfillment of the requirements for the award of degree of*

**BACHELOR OF TECHNOLOGY**

*in*

**COMPUTER SCIENCE & ENGINEERING**

*by*

G. SATWIK (19211A0583)

G. AKHIL (19211A0588)

D. NIKHIL KUMAR (19211A0564)

G. GAGAN (19211A05A3)

*Under the guidance of*

**Mrs. D. DEEPIKA**

****Associate Professor

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**B.V. RAJU INSTITUTE OF TECHNOLOGY**

(UGC Autonomous, Accredited by NBA & NAAC)

Vishnupur, Narspur, Medak (Dist.), Telangana State, India – 502313

2022 - 2023

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**CERTIFICATE**

This is to certify that the Major Project entitled **“STROKE PREDICTIONUSING MACHINE LEARNING”**, being submitted by

G. SATWIK (19211A0583)

G. AKHIL (19211A0588)

D. NIKHIL KUMAR (19211A0564)

G. GAGAN (19211A05A3)

In partial fulfillment of the requirements for the award of degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING to B.V. RAJU INSTITUTE OF TECHNOLOGY is a record of bonafide work carried out during a period from **August 2022 to May 2023** by them under the guidance of **Mrs. D. DEEPIKA**, Associate Professor, CSE Department.

This is to certify that the above statement made by the students is/are correct to the best of my knowledge.

**Mrs. D. DEEPIKA**

Associate Professor

The Project Viva-Voce Examination of this team has been held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

**Dr. Ch. Madhu Babu**

EXTERNAL EXAMINER Professor & HoD-CSE

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**CANDIDATE’S DECLARATION**

We hereby certify that the work which is being presented in the project entitled **“STROKE PREDICTION USING MACHINE LEARNING”** in partial fulfillment of the requirements for the award of Degree of Bachelor of Technology and submitted in the Department of Computer Science and Engineering, B. V. Raju Institute of Technology, Narsapur is an authentic record of my own work carried out during a period from **August 2022 to May 2023** under the guidance of **Mrs. D. DEEPIKA**, Associate Professor. The work presented in this project report has not been submitted by us for the award of any other degree of this or any other Institute/University.

G. SATWIK (19211A0583)

G. AKHIL (19211A0588)

D. NIKHIL KUMAR (19211A0564)

G. GAGAN (19211A05A3)

**ACKNOWLEDGEMENT**

The success and final outcome of this project required a lot of guidance and assistance from many people, and I am extremely fortunate to have got this all along the completion. Whatever we have done is due to such guidance and assistance. We would not forget to thank them.

We thank **Mrs.** **D. DEEPIKA** for guiding us and providing all the support in completing this project. We are thankful to **Mrs. Ch. Sreedevi**, our section project coordinator for supporting us in doing this project. We are thankful to **Dr. Pradeep Kumar V**, project coordinator for helping us in completing the project in time. We thank the person who has our utmost gratitude is Dr**. Ch. Madhu Babu**, Head of CSE Department.

We are thankful to and fortunate enough to get constant encouragement, support and guidance from all the staff members of CSE Department.

G. SATWIK (19211A0583)

G. AKHIL (19211A0588)

D. NIKHIL KUMAR (19211A0564)

G. GAGAN (19211A05A3)

**STROKE PREDICTION USING MACHINE LEARNING**

**ABSTRACT**

Stroke, a medical emergency that occurs due to the interruption of flow of blood to a part of brain because of bleeding or blood clots. Worldwide, it is the second major reason for deaths with an annual mortality rate of 5.5 million. Every year, more than 15 million people worldwide have a stroke, and in every 4 minutes, someone dies due to stroke. A stroke is generally a consequence of a poor style of living and hence, preventable in up to 80% of the cases. Therefore, the prediction of stroke becomes necessary and should be used to prevent permanent damage by stroke. The current work predicted the stroke using the different machine learning models namely, Gaussian Naive Bayes, Logistic Regression, Decision Tree Classifier, K-Nearest Neighbors, AdaBoost Classifier, XGBoost Classifier, and Random Forest Classifier. The paper presents the comparison among all machine learning algorithms. Analysis of results revealed that the AdaBoost, XGBoost and Random Forest Classifier made the least value of incorrect predictions and had the greatest accuracy scores 95%, 96% and 97% respectively. Hence, they were the best suited model for stroke prediction and can feasibly be used by physicians to predict stroke in real world.

**TABLE OF CONTENTS**

[CHAPTER-1 1](#_Toc135132242)

[INTRODUCTION 1](#_Toc135132243)

[1.1 Introduction 1](#_Toc135132244)

[CHAPTER- 2 3](#_Toc135132245)

[literature survey 3](#_Toc135132246)

[2.1 literature review 3](#_Toc135132247)

[Chapter-3 5](#_Toc135132251)

[Theoretical background 5](#_Toc135132252)

[3.1 Introduction: 5](#_Toc135132253)

[3.2 Introduction to PYTHON 5](#_Toc135132254)

[Chapter-4 20](#_Toc135132255)

[System analysis 20](#_Toc135132256)

[4.1 EXISTING SYSTEM: 20](#_Toc135132257)

[4.1.1 DISADVANTAGES OF EXISTING SYSTEM: 20](#_Toc135132258)

[4.2 PROPOSED SYSTEM: 21](#_Toc135132259)

[4.2.1 ADVANTAGES OF PROPOSED SYSTEM: 21](#_Toc135132260)

[CHAPTER- 5 22](#_Toc135132261)

[SYSTEM design 22](#_Toc135132262)

[5.1 introduction 22](#_Toc135132263)

[5.2 modules 22](#_Toc135132264)

[5.2.1 Data COLLECTION: 22](#_Toc135132265)

[5.2.2 Pre-processing: 23](#_Toc135132266)

[5.2.3 Label Encoding: 23](#_Toc135132267)

[5.3 system architecture 29](#_Toc135132268)

[5.4 UML DAIGRAMS 31](#_Toc135132269)

[5.4.1 Construction of Use case diagrams: 35](#_Toc135132270)

[5.4.2 SEQUENCE DIAGRAMS: 36](#_Toc135132271)

[5.4.3 CLASS DIAGRAM: 36](#_Toc135132272)

[5.4.4 ACTIVITY DIAGRAM: 37](#_Toc135132273)

[CHAPTER-6 39](#_Toc135132274)

[system requirements 39](#_Toc135132275)

[6.1 SYSTEM REQUIREMENTS 39](#_Toc135132276)

[6.1.1 HARDWARE REQUIREMENTS: 39](#_Toc135132277)

[6.1.2 SOFTWARE REQUIREMENTS: 39](#_Toc135132278)

[Chapter-7 40](#_Toc135132279)

[System implementation 40](#_Toc135132280)

[7.1 input and output designs 40](#_Toc135132281)

[7.1.1 Logical design 40](#_Toc135132282)

[7.1.2 Physical design 40](#_Toc135132283)

[7.2 INPUT & OUTPUT REPRESENTATION 42](#_Toc135132284)

[7.2.1 Input Design 42](#_Toc135132285)

[7.2.2 Objectives 42](#_Toc135132286)

[1.2.3 Output Design 43](#_Toc135132287)

[Chapter-8 47](#_Toc135132288)

[System testing 47](#_Toc135132289)

[8.1 INTRODUCTION: 47](#_Toc135132290)

[8.2 Levels of Testing 48](#_Toc135132291)

[8.2.1 BLACK BOX TESTING 49](#_Toc135132292)

[8.2.2 WHITE BOX TESTING 51](#_Toc135132293)

[CHAPTER-9 60](#_Toc135132294)

[Output Screens 60](#_Toc135132295)

[9.1 Dataset SCREEN 60](#_Toc135132296)

[CONCLUSION and Future work 64](#_Toc135132297)

[REFERENCES 65](#_Toc135132298)

**List of Figures**

**Name of the figure Pg.no**

[Figure 5. 1 System Architecture 33](#_Toc80179735)

[Figure 5. 2 Use Case Diagram 39](#_Toc80179736)

[Figure 5. 3 Sequence diagram 41](#_Toc80179737)

[Figure 5. 4 Class Diagram 42](#_Toc80179738)

[Figure 5. 5 Activity Diagram 43](#_Toc80179739)

# CHAPTER-1

# INTRODUCTION

## Introduction

Stroke, also known as brain attack, happens when blood flow to the brain is blocked, preventing it from getting oxygen and nutrients from it and causing the death of brain cells within minutes [1]. According to the World Health Organization (WHO), it is the second cause of death worldwide after ischemic heart disease [2]. Stroke victims can experience paralysis, impaired speed, or loss of vision. While some of the Stroke risk factors cannot be modified, such as family history of cerebrovascular diseases, age, gender and race, others can and are estimated to account for 60% -80% of stroke risk in the general population [3]. Therefore, predicting stroke outcome for new cases can be determining for them to be treated early enough and avoid disabling and mortal consequences. The purpose of this research is to create an accurate model for predicting Stroke outcome with Data Science and Machine Learning (ML) based on previous data and individual characteristics. Providing useful information for the medical staff to deploy the needed treatment and decrease risks and consequences. A. Related work Data Science have been studied previously as a tool for predictions about Stroke. One of them is the development of a hybrid machine learning approach for cerebral stroke prediction [4]. Despite the similarity with the research described in this paper, the method used was different, and looking for a better performing model is another objective for this research. Not only Stroke prediction but its consequences have been widely studied. For example, Kim et al. [5] developed a Deep Neural Network (DNN) for predicting the motor outcome at six months after a Stroke by analyzing the data from 1,056 consecutive stroke patients with the objective of providing important information for clinicians to establish appropriate rehabilitation strategies for patients. Also, in [6] Hbid et al. evaluated a mixed-effect linear model to predict the risk of cognitive decline post-stroke. B. ML algorithms Sometimes, it can be difficult to interpret the patterns or extract useful information from data. This can be because of the amount of information or because the relationship between its components might not be easily perceptible. This is where ML techniques are very useful in order to teach, as its name implies, machines to handle data more efficiently [7]. Although there are many ML algorithms and variants out there, its applicability depends on the data composition and the research question to be answered. Supervised classification is one of the tasks more frequently carried out by Intelligent Systems, which goal is to build concise model of the distribution of class labels in terms of predictor features. Then the resulting classifier is used to assign class labels to the testing instances where the values of the predictors are known, but the value of the class label is not [8].

**Problem Identification & Objectives**

It’s not only very expensive for the medical treatments and a permanent disability but can at last prompt demise. By and large, Data Mining assumes an imperative part in the forecast of illnesses in medical care industry.

# HAPTER- 2

# literature survey

## 2.1 literature review

# In [4], stroke prediction was made on Cardiovascular Health Study (CHS) dataset using five machine learning techniques. As an optimal solution, the authors used a combination of the Decision Tree with the C4.5 algorithm, Principal Component Analysis, Artificial Neural Networks, and Support Vector Machine. But the CHS Dataset taken for this work had a smaller number of input parameters. In [5], stroke prediction has been carried out from the social media posts posted by people. In this particular work, the authors have used the DRFS method to find the various symptoms associated with stroke disease. The usage of Natural Language Processing to extract the text from the social media posts adds up to the overall execution time of the model which is not desirable. In [6], the authors have performed the task of stroke prediction by using an improvised random forest algorithm. This was used to analyze the levels of risks obtained with the strokes. As suggested by the authors, this method is said to have performed better when compared to the existing algorithms. This particular research is limited to very few types of strokes and cannot be used for any new stroke type in the future.

# Research paper [7] shows that the model was trained using Decision Tree, Random Forest, and Multi-layer perceptron for stroke prediction. The obtained accuracies for the three methods were quite close, with slight differences. The calculated accuracy for Decision Tree was 74.31%, Random Forest was 74.53%, and Multi-layer perceptron was 75.02%. This paper suggests that Multi-layer perceptron is more accurate than the other two methods. Accuracy score was the only metric used for calculating the performance that might not always give favorable results. Research carried out in [8] shows the implementation of machine learning model to predict heart stroke. They used various machine learning techniques like Decision tree, Naïve Bayes, SVM to build the model and later compared their performance. They obtained a maximum accuracy of 60% from the used algorithms which is pretty less.

# In [9], the authors have used different data mining classification techniques to predict the possibility of a stroke. The dataset was taken from the Ministry of National Guards Health Affairs Hospitals, Kingdom of Saudi Arabia. The three classification algorithms used were C4.5, Jrip and Multi layers perceptron (MLP). With these algorithms, the model obtained an accuracy of around 95%. Even though the paper claims to obtain an accuracy of 95%, the time taken for training and predicting is higher as the authors have used a combination of complex algorithms. Research carried out in [10], suggests the usage of three different algorithms to predict the possibility of stroke. These algorithms are Naïve Bayes, Decision Tree, and Neural Networks. This paper concluded that the Decision tree has the highest accuracy (about 75%) of the other two algorithms. But this model could not suit the real-world examples based on the values obtained from the confusion matrix.

# In [11], the researchers have performed stroke prediction on Cardiovascular Health Study (CHS) dataset. They proposed a novel automatic feature selection algorithm that selects robust features based on their proposed conservative mean. They have combined this method with the Support Vector Machine algorithm for better efficiency. But this resulted in the generation of a number of vectors that tend to reduce the performance of the model. Research in [12] proposes the prediction of thromboembolic stroke disease using Artificial Neural Networks. The method used for prediction was the Back-propagation algorithm. This model was able to get an accuracy of around 89%. But Neural Networks need more time to be trained and require higher processing time because of the complex structure with increasing number of neurons.

# 

# Chapter-3

# Theoretical background

## 3.1 Introduction:

## 3.2 Introduction to PYTHON

**Python**

Python technology is both a programming language and a platform.

**The python Programming Language**

THE PYTHON PROGRAMMING LANGUAGE IS A HIGH-LEVEL LANGUAGE THAT CAN BE CHARACTERIZED BY ALL OF THE FOLLOWING BUZZWORDS:

* + - Simple
    - Architecture neutral
    - Object oriented
    - Portable
    - Distributed
    - High performance
    - Interpreted
    - Multithreaded
    - Robust
    - Dynamic
    - Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Python programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called Python byte codes —the platform-independent codes interpreted by the interpreter on the Python platform. The interpreter parses and runs each Python byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.

FEATURES OF MACHINE LEARNING

• It is nothing but automating the Automation.

• Getting computers to program themselves.

• Writing Software is bottleneck.

• Machine leaning models involves machines learning from data without the help of humans or any kind of human intervention.

• Machine Learning is the science of making of making the computers learn and act like humans by feeding data and information without being explicitly programmed.

• Machine Learning is totally different from traditionally programming, here data and output is given to the computer and in return it gives us the program which provides solution to the various problems. Below is the figure.

**Traditional Programming vs Machine Learning**

• Machine Learning is a combination of Algorithms, Datasets, and Programs.

• There are Many Algorithms in Machine Learning through which we will provide us the exact solution in predicting the disease of the patients.

• How Does Machine Learning Works?

• Solution to the above question is Machine learning works by taking in data, finding relationships within that data and then giving the output.

**Machine Learning Model**

There are various applications in which machine learning is implemented such as Web search, computing biology, finance, e-commerce, space exploration, robotics, social networks, debugging and much more.

There are 3 types of machine learning supervised, unsupervised, and reinforcement.

**BENEFITS OF PYTHON**

• Presence of Third-Party Modules

• Extensive Support Libraries

• Open Source and Community Development

• Learning Ease and Support Available

• User-friendly Data Structures

• Productivity and Speed

• Highly Extensible and Easily Readable Language.

**Python**

Python is high level language, and it is also integrated version of the program. Python is an object-oriented approach and its main aim to help programmers to write the code clearly, logical code for small and large scale of project.

Python is dynamically typed and garbage collected it also support multiple programming and it is both procedure and object oriented and also functional programming. And structural programming also supported. It has many built in function it also supports filter, map and reduce function. All the machine learning algorithm and the libraries are being supported by the python programming language. Python also support list, dict, sets and other generators. Python code can be run in different platform such as anaconda, PyCharm etc.

The main goal of this programing language is as follows:

• Python is simple, object-oriented programming language.

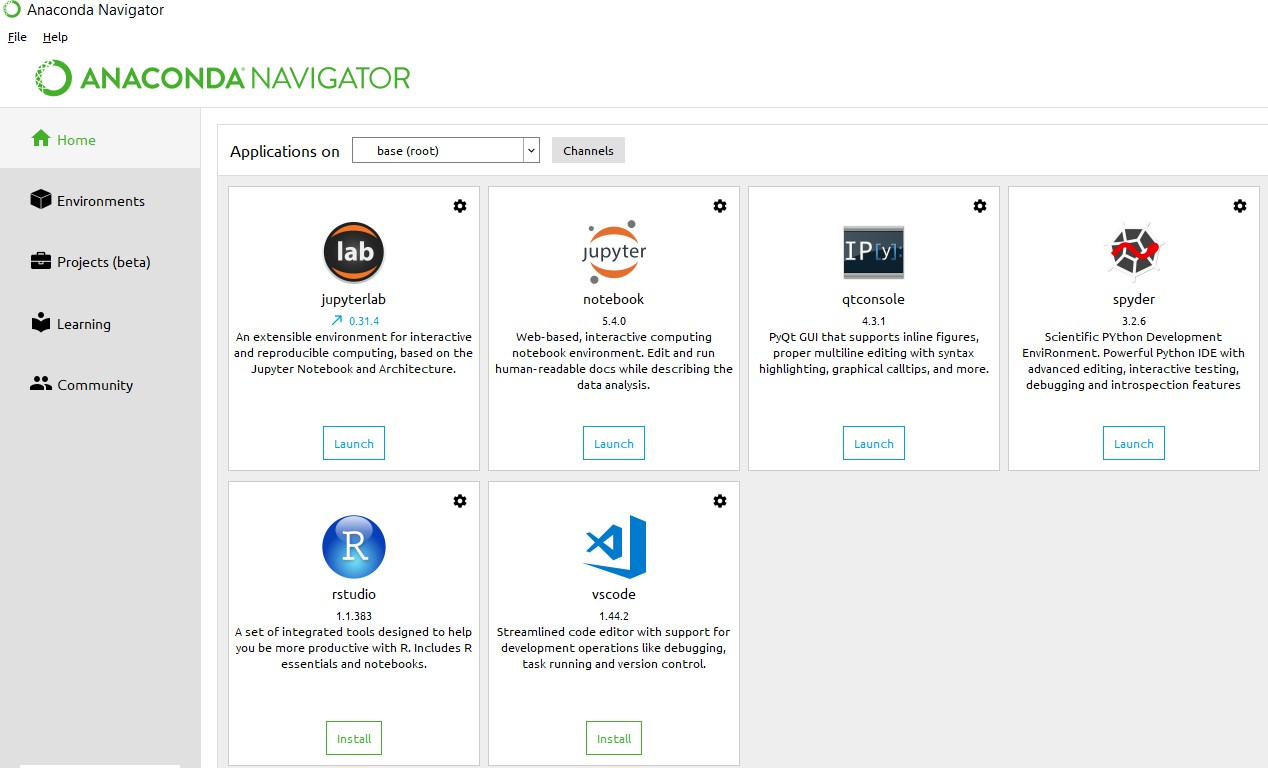
• The language and implementation should provide support for software engineering principles such as strong type library preset for different machine learning algorithm, and all other algorithm in simple manner.

• Coding will be smooth in python and the data analysis can be easily done in python.

This is so much so to the point where we now have modules and APIs at our disposal, and you can engage in machine learning very easily without almost any knowledge at all of how it works. With the defaults from Scikit-learn, you can get 90-95% accuracy on many tasks right out of the gate. Machine learning is a lot like a car, you do not need to know much about how it works in order to get an incredible amount of utility from it.

Despite the apparent age and maturity of machine learning, I would say there's no better time than now to learn it, since you can actually use it. Machines are quite powerful, the one you are working on can probably do most of this series quickly. Data is also very plentiful lately.

**Anaconda**

Anaconda is free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine Learning applications, Large- scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. It is developed and maintained by Anaconda, Inc. The distribution incudes data-science packages suitable for Windows, Linux, and macOS. Packaged versions are required and are managed by the package management system anaconda. This package manager was spun out as a separate open-source package as it ended up being useful on its own and for other things than Python. There is also a small, bootstrap version of Anaconda called Miniconda, which includes only conda, Python, the packages they depends on, and a small number of other packages.

**Anaconda Console**

Machine learning is the ability that gives the computer to learn without being explicitly programmed. There are two types of machine learning:

**Supervised Learning:** supervised learning is the learning of the labelled data. It is the types of machine learning that maps the input and output based on the examples input-output pairs. In supervised learning each training data having pairs of input and desired outputs values. Supervised learning algorithm analyzes the training data and produces a function which can be used for mapping of new data.

Supervised Learning The output to solve the supervised learning algorithm are as:

• Determine the types of data, before doing anything else the user should understand which types of data set is to be used for training the data.

• Gathered the training data sets either in form of human experts or from measurements.

• Determine the feature of inputs from the learned data and depends on the inputs it changed into feature vector; number of features should not be large but should contains enough information to accurately predict the outputs.

• Check the learned function and the learned algorithm for example we use support vector machines or decisions tree.

• Complete the design and run the trained data sets.

• Analyzed the output and verify the data sets to get the accurate outputs.

**Unsupervised Learning:** Unsupervised learning is a type of machine learning that helps in finding the previously unknown patterns in the data set without any known labels. It is known as self- organization and allows modelling probability densities of given inputs.

unsupervised Learning Some of the algorithm used in unsupervised learning are:

• Clustering

• Anomaly detection

• Neural networks

• Approach for learning latent variable models

• Non labelled data

Semi Supervised Machine Learning algorithm: It’s like the middleman which have some labeled data and some unlabeled which can be prosed by the both the structured and unsupervised learning.

The algorithms have been compared based upon the parameters: Size of the dataset and Number of technical indicators used. Accuracy and F-measure values have been computed for each algorithm. Long term model has been used to compute the accuracy and F-measure.

**import NumPy as np.**

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

At the core of the NumPy package, is the ndarray object. This encapsulates n-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of a ndarray will create a new array and delete the original.

The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.

NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python’s built-in sequences.

A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today’s scientific/mathematical Python-based software, just knowing how to use Python’s built-in sequence types is insufficient - one also needs to know how to use NumPy arrays.

**import time.**

This module provides various time-related functions. For related functionality, see also the datetime and calendar modules.

Although this module is always available, not all functions are available on all platforms. Most of the functions defined in this module call platform C library functions with the same name. It may sometimes be helpful to consult the platform documentation, because the semantics of these functions varies among platforms.

An explanation of some terminology and conventions is in order.

The epoch is the point where the time starts, and is platform dependent. For Unix, the epoch is January 1, 1970, 00:00:00 (UTC). To find out what the epoch is on a given platform, look at time.gmtime(0).

The term seconds since the epoch refers to the total number of elapsed seconds since the epoch, typically excluding leap seconds. Leap seconds are excluded from this total on all POSIX-compliant platforms.

The functions in this module may not handle dates and times before the epoch or far in the future. The cut-off point in the future is determined by the C library; for 32-bit systems, it is typically in 2038.

Function strptime() can parse 2-digit years when given %y format code. When 2-digit years are parsed, they are converted according to the POSIX and ISO C standards: values 69–99 are mapped to 1969–1999, and values 0–68 are mapped to 2000–2068.

UTC is Coordinated Universal Time (formerly known as Greenwich Mean Time, or GMT). The acronym UTC is not a mistake but a compromise between English and French.

DST is Daylight Saving Time, an adjustment of the timezone by (usually) one hour during part of the year. DST rules are magic (determined by local law) and can change from year to year. The C library has a table containing the local rules (often it is read from a system file for flexibility) and is the only source of True Wisdom in this respect.

The precision of the various real-time functions may be less than suggested by the units in which their value or argument is expressed. E.g., on most Unix systems, the clock “ticks” only 50 or 100 times a second.

On the other hand, the precision of time () and sleep () is better than their Unix equivalents: times are expressed as floating point numbers, time () returns the most accurate time available (using Unix gettimeofday() where available), and sleep() will accept a time with a nonzero fraction (Unix select() is used to implement this, where available).

The time value as returned by gmtime(), localtime(), and strptime(), and accepted by asctime(), mktime() and strftime(), is a sequence of 9 integers. The return values of gmtime(), localtime(), and strptime() also offer attribute names for individual fields.

See struct\_time for a description of these objects.

Changed in version 3.3: The struct\_time type was extended to provide the tm\_gmtoff and tm\_zone attributes when platform supports corresponding struct tm members.

Changed in version 3.6: The struct\_time attributes tm\_gmtoff and tm\_zone are now available on all platforms.

**import os**

This module provides a portable way of using operating system dependent functionality. If you just want to read or write a file see open(), if you want to manipulate paths, see the os.path module, and if you want to read all the lines in all the files on the command line see the fileinput module. For creating temporary files and directories see the tempfile module, and for high-level file and directory handling see the shutil module.

Notes on the availability of these functions:

The design of all built-in operating system dependent modules of Python is such that as long as the same functionality is available, it uses the same interface; for example, the function os.stat(path) returns stat information about path in the same format (which happens to have originated with the POSIX interface).

Extensions peculiar to a particular operating system are also available through the os module but using them is of course a threat to portability.

All functions accepting path or file names accept both bytes and string objects, and result in an object of the same type, if a path or file name is returned.

# Chapter-4

# System analysis

## 4.1 EXISTING SYSTEM:

Tasfia Ismail Shoily et al., has compared the different models between Naive Bayes, J48, k-NN, and Random Forest, we observe Naive Bayes has better precision. Different medical reports are observed to obtain the dataset which was cross referenced by medical experts and used with WEKA (Waikato Environment for Knowledge Analysis). The model which is developed will help patients to be cautious whether they may get a stroke or not. Trained 4 different models such as Naive Bayes, J48, k-NN and Random Forest. Precision and accuracy were observed to validate the models. The dataset is applied to the machine learning models [1].

### DISADVANTAGES OF EXISTING SYSTEM:

* + 1. Accuracy of these Models are less and prediction of results are not correct

All types of features are not taken as input.

Pre-processing was done to remove redundant and incompatible data, 350 inputs were taken for the prediction. It was run on MATLAB which has given less accuracy below 50 percent.

. • Prediction is not accurate or given input.

## 4.2 PROPOSED SYSTEM:

Stroke is a destructive illness that typically influences individuals over the age of 65 years age. Prediction of stroke is a time consuming and tedious for doctors. Therefore, the project mainly aims at predicting the chances of occurrence of stroke using the emerging Machine Learning techniques. Five different algorithms are used, and a comparison is made for better accuracy. Aim is to create an application with a user friendly interface which is easy to navigate and enter inputs.

### 4.2.1 ADVANTAGES OF PROPOSED SYSTEM:

* Prediction accuracy is above 80 percent.
* Preprocessing data is easy and data analysis is done in proposed system.

# 

# CHAPTER- 5

# SYSTEM design

## 5.1 introduction

**System Design Introduction:**

The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces.

## 5.2 modules

### 5.2.1 Data COLLECTION:

The dataset for stroke prediction is from Kaggle [3]. This particular dataset has 5110 rows and 12 columns. The columns have 'id', 'gender', 'age', 'hypertension', 'heart\_disease', 'ever\_marri ed', 'work\_type', 'Residence\_type', 'avg\_glucose\_level', 'bmi', 'smoking\_status' and 'stroke' as the main attributes. The output column 'stroke' has the value as either '1' or '0'. The value '0' indicates no stroke risk detected, whereas the value '1' indicates a possible risk of stroke. This dataset is highly imbalanced as the possibility of '0' in the output column ('stroke') outweighs that of '1' in the same column. Only 249 rows have the value '1' whereas 4861 rows with the value '0' in the stroke column. For better accuracy, data pre-processing is performed to balance the data.

### 5.2.2 Pre-processing:

Data Preprocessing is required before model building to remove the unwanted noise and outliers from the dataset, resulting in a deviation from proper training. Anything that interrupts the model from performing with less efficiency is taken care of in this stage. After collecting the appropriate dataset, the next step lies in cleaning the data and making sure that it is ready for model building. The dataset taken has 12 attributes, as mentioned in Table I. Firstly, the column 'id' is dropped because its existence does not make much difference in model building. Then the dataset is checked for null values and filled if any found. In this case, the column 'bmi' has null values filled with the mean of the column data. After removing the null values from the dataset, the next task is Label Encoding.

### 5.2.3 Label Encoding:

Label encoding encodes the string literals in the dataset into integer values for the machine to understand them. As the machine is usually trained in numbers, the strings have to be converted into integers. There are five columns in the collected dataset that have strings as their data type. On performing label encoding, all the strings get encoded, and the entire dataset becomes a combination of numerals.

**Weight Calculation Using TF-IDF**

The dataset chosen for the task of stroke prediction is highly imbalanced. The entire dataset has 5110 rows, of which 249 rows are suggesting the occurrence of a stroke and 4861 rows having the possibility of no stroke. The graphical representation of the imbalance is in Fig. 2. Training a machine-level model with such data might give accuracy, but other accuracy metrics like precision and recall are shallow. If such imbalanced data is not handled, the results are not accurate, and the prediction is inefficient. Therefore, to get an efficient model, this imbalanced data is to be first handled. For this purpose, the method of under sampling is used. Under sampling [13] balances the data wherein the majority class is under sampled to match the minority class. In this case, the class with a value as '0' is under sampled for the class with the value' 1'. So after under sampling the resulting dataset will have 249 rows with value ‘0’ and 249 rows with value ‘1’.

**Splitting the Data:**

After completing data preprocessing and handling the imbalanced dataset, the next step is building the model. The under sampled data is split into training and testing data for better accuracy and efficiency for this task keeping the ratio as 80% training data and 20% testing data. After splitting, various classification algorithms are used to train the model. The classification algorithms used for this purpose are Logistic Regression, Decision Tree Classification algorithm, Random Forest Classification, K-Nearest Neighbors Classification, Support Vector Machine and Naïve Bayes Classification.

**Classification Algorithms:**

Random forest classification: The following classification algorithm chosen is Random Forest Classification . Random Forests are composed of multiple independent decision trees trained independently on a random subset of data. These trees are generated at the time of training, and the outputs are obtained from each decision tree. For the final prediction from this algorithm, a method called "voting" takes place. This method means that each decision tree votes for an output class (in this case, the two classes are: 'stroke' and 'no stroke'). The random forest chooses the class with the maximum number of votes as the final prediction. The accuracy obtained by training the model using this particular algorithm is 73%. The precision and recall scores are 72% and 73.5%, respectively. The F1 Score obtained with this algorithm is 72.7%. The Receiver operating characteristic (ROC) curve for Random Forest Classification is 73%.

**Identifying stroke using RFC Algorithm**

#### Random Forest

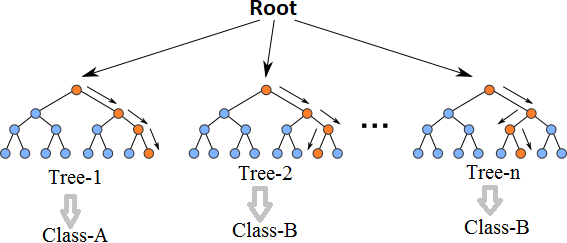
Random forest is a tree-based algorithm which involves building several trees (decision trees), then combining their output to improve generalization ability of the model. The method of combining trees is known as an ensemble method. Ensembling is nothing but a combination of weak learners (individual trees) to produce a strong learner.

Definition: A random forest is a classifier consisting of a collection of tree structured classifiers *h*(*x,* Θ*k*)*, k* = 1*, ...* where the Θ*k* are independent identically distributed (*i.i.d*) random vectors and each tree casts a unit vote for the most popular class at input [[4](#_bookmark61)].

Random Forest Algorithm: The following are the basic steps involved in performing the random forest algorithm:

* + - * Pick N random records from the dataset.
      * Build a decision tree based on these N records.
      * Choose the number of trees you want in your algorithm and repeat steps (i) and (ii).
      * In case of a classification problem, each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the majority vote.

Figure [2.1](#_bookmark0) shows different trees labelling the class differently. What ensemble does is take the mode (maximum occurring class) of the output produced by n different trees to create a better model. To say it in simple words: Random Forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.



Even though decision trees are pretty intuitive and easier to understand, they can be very noisy. Few changes in the data can lead to different splits and completely different models. The instability of the tree makes it unrealistic as a prediction model by itself. A single decision tree is insufficient and generally overfits the data, that is it can capture the structure of the in-sample data very well, but it tends to work poorly out-of-sample. In the context of statistics, decisions trees have low bias (as it can fit the data well) but high variances (the predictions are noisy).

Understanding the working principle of decision trees is imperative in the understanding of Random Forest Algorithm. The most popular algorithm for decision trees is ID3 algorithm. It finds the best attributes/features that best classifies the target attribute. One of the most commonly used way to figure out the best attribute is by calculating Information Gain which is, in turn, calculated using another property called Entropy.

The calculation of entropy of a system is done as follows:

*Entropy*(*S*) = ∑ *pilog*2 *pi*

−

*i*=1

Here, c is the total number of classes or attributes, and *pi* is number of examples belonging to the *ith* class. Information gain is simply the expected reduction in entropy caused by partitioning all our examples according to a given attribute. Mathematically, it is defined as:

*Gain* (*S, A*) ≡ *Entropy*(*S*) − ∑|Sv|/|S|­­­­­­ *Entropy* (*Sv*) (2.2)

*v*a*lues*(*A*)

S refers to the entire set of examples that we have. A is the attribute we want to partition or split. |S| is the number of examples and |*Sv*| is the number of examples for the current value of attribute A. The attribute with the highest information gain sits at the root node, and the tree is first split based on that attribute.

**Experimental Results**

In our experiments, we first used the ground truth set of 400 Android apps, including 200 benign apps from official Android market and 200 malwares from Drebin dataset, to train a SVM model. Then, we used the trained SVM model to test a new unlabelled set of 300 Android apps, which contains 150 benign apps and 150 malwares. Figure 3 shows the SVM classification result, and the accuracy of the SVM classifier is 80 percent and Neural network accuracy is 92 percent.

## 

## 5.3 system architecture

A system architecture or systems architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system. Organized in a way that supports reasoning about the structures and behaviors of the system.

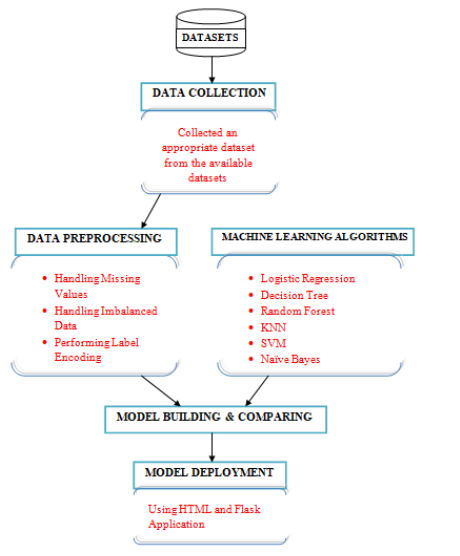
****

Figure 5. 1 System Architecture

**3-Tier Architecture:**

The three-tier software architecture (a three-layer architecture) emerged in the 1990s to overcome the limitations of the two-tier architecture. The third tier (middle tier server) is between the user interface (client) and the data management (server) components. This middle tier provides process management where business logic and rules are executed and can accommodate hundreds of users (as compared to only 100 users with the two tier architecture) by providing functions such as queuing, application execution, and database staging.

The three tier architecture is used when an effective distributed client/server design is needed that provides (when compared to the two tier) increased performance, flexibility, maintainability, reusability, and scalability, while hiding the complexity of distributed processing from the user. These characteristics have made three layer architectures a popular choice for Internet applications and net-centric information systems.

**Advantages of Three-Tier:**

* Separates functionality from presentation.
* Clear separation – better understanding.
* Changes limited to well define components.
* Can be running on WWW.
* Effective network performance.

## 5.4 UML DAIGRAMS

Global Use Case Diagrams:

Identification of actors:

Actor: Actor represents the role a user plays with respect to the system. An actor interacts with, but has no control over the use cases.

Graphical representation:



An actor is someone or something that:

Interacts with or uses the system.

* Provides input to and receives information from the system.
* Is external to the system and has no control over the use cases.

Actors are discovered by examining:

* Who directly uses the system?
* Who is responsible for maintaining the system?
* External hardware used by the system.
* Other systems that need to interact with the system.

Questions to identify actors:

* + Who is using the system? Or, who is affected by the system? Or, which groups need help from the system to perform a task?
  + Who affects the system? Or, which user groups are needed by the system to perform its functions? These functions can be both main functions and secondary functions such as administration.
  + Which external hardware or systems (if any) use the system to perform tasks?
  + What problems does this application solve (that is, for whom)?
  + And, finally, how do users use the system (use case)? What are they doing with the system?

The actors identified in this system are:

1. System Administrator
2. Customer
3. Customer Care

Identification of use cases:

Use case: A use case can be described as a specific way of using the system from a user’s (actor’s) perspective.

Graphical representation:



**1.2 Flow of Events**

A flow of events is a sequence of transactions (or events) performed by the system. They typically contain very detailed information, written in terms of what the system should do, not how the system accomplishes the task. Flow of events are created as separate files or documents in your favorite text editor and then attached or linked to a use case using the Files tab of a model element. A flow of events should include:

* When and how the use case starts and ends
* Use case/actor interactions
* Data needed by the use case
* Normal sequence of events for the use case
* Alternate or exceptional flows.

### 5.4.1 Construction of Use case diagrams:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

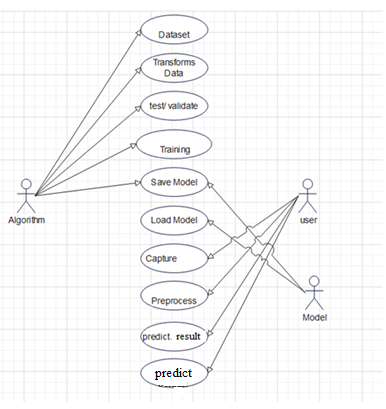


Figure 5. 2 Use Case Diagram

### 5.4.2 SEQUENCE DIAGRAMS:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

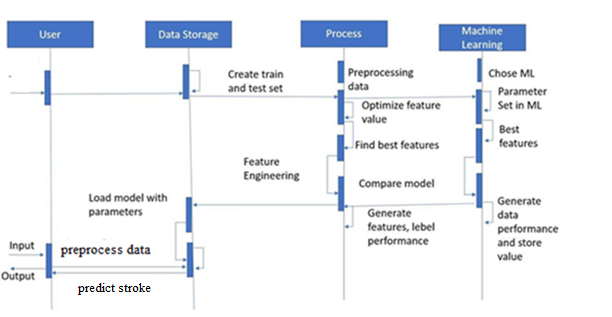


Figure 5. 3 Sequence diagram

### 5.4.3 CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

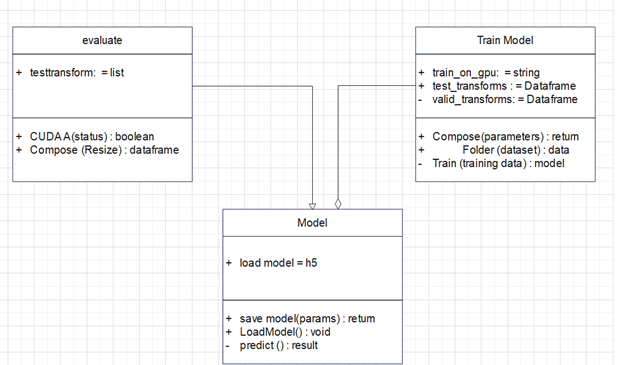


Figure 5. 4 Class Diagram

### 5.4.4 ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

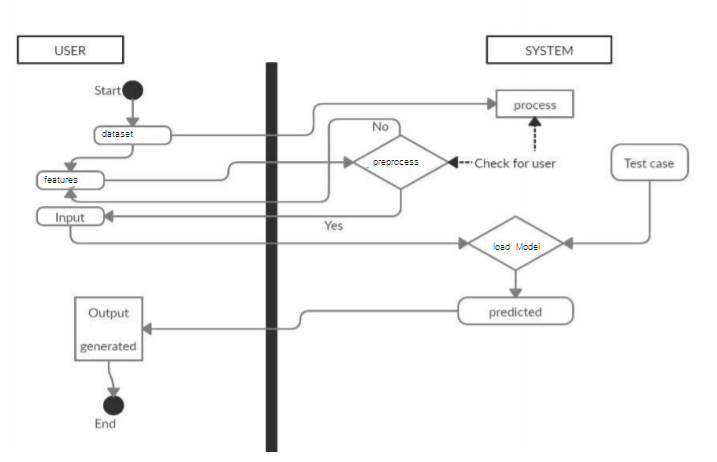
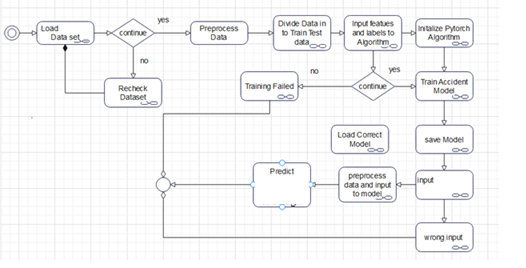
****

Figure 5. 5 Activity Diagram

****

# CHAPTER-6

# system requirements

## 6.1 SYSTEM REQUIREMENTS

### 6.1.1 HARDWARE REQUIREMENTS:

* System : Intel(R) Core (TM) i3-7020U CPU @ 2.30GHz
* Hard Disk : 1 TB.
* Input Devices : Keyboard, Mouse
* Ram : 4 GB.

### 6.1.2 SOFTWARE REQUIREMENTS:

* Operating system : Windows XP/7/10.
* Coding Language : Python
* Tool : Anaconda
* Interface : flask

# Chapter-7

# System implementation

To conduct studies and analyses of an operational and technological nature, and to promote the exchange and development of methods and tools for operatio nal analysis as applied to defense problems.

## 7.1 input and output designs

### 7.1.1 Logical design

The logical design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system. This is often conducted via modeling, using an over-abstract (and sometimes graphical) model of the actual system. In the context of systems design are included. Logical design includes ER Diagrams i.e. Entity Relationship Diagrams

### 7.1.2 Physical design

The physical design relates to the actual input and output processes of the system. This is laid down in terms of how data is input into a system, how it is verified / authenticated, how it is processed, and how it is displayed as output. In Physical design, following requirements about the system are decided.

1. Input requirement,
2. Output requirements,
3. Storage requirements,
4. Processing Requirements,
5. System control and backup or recovery.

Put another way, the physical portion of systems design can generally be broken down into three sub-tasks:

1. User Interface Design
2. Data Design
3. Process Design

User Interface Design is concerned with how users add information to the system and with how the system presents information back to them. Data Design is concerned with how the data is represented and stored within the system. Finally, Process Design is concerned with how data moves through the system, and with how and where it is validated, secured and/or transformed as it flows into, through and out of the system. At the end of the systems design phase, documentation describing the three sub-tasks is produced and made available for use in the next phase.

Physical design, in this context, does not refer to the tangible physical design of an information system. To use an analogy, a personal computer's physical design involves input via a keyboard, processing within the CPU, and output via a monitor, printer, etc. It would not concern the actual layout of the tangible hardware, which for a PC would be a monitor, CPU, motherboard, hard drive, modems, video/graphics cards, USB slots, etc. It involves a detailed design of a user and a product database structure processor and a control processor. The H/S personal specification is developed for the proposed system.

## 7.2 INPUT & OUTPUT REPRESENTATION

### 7.2.1 Input Design

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

* What data should be given as input?
* How the data should be arranged or coded?
* The dialog to guide the operating personnel in providing input.
* Methods for preparing input validations and steps to follow when error occur.

### 7.2.2 Objectives

Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

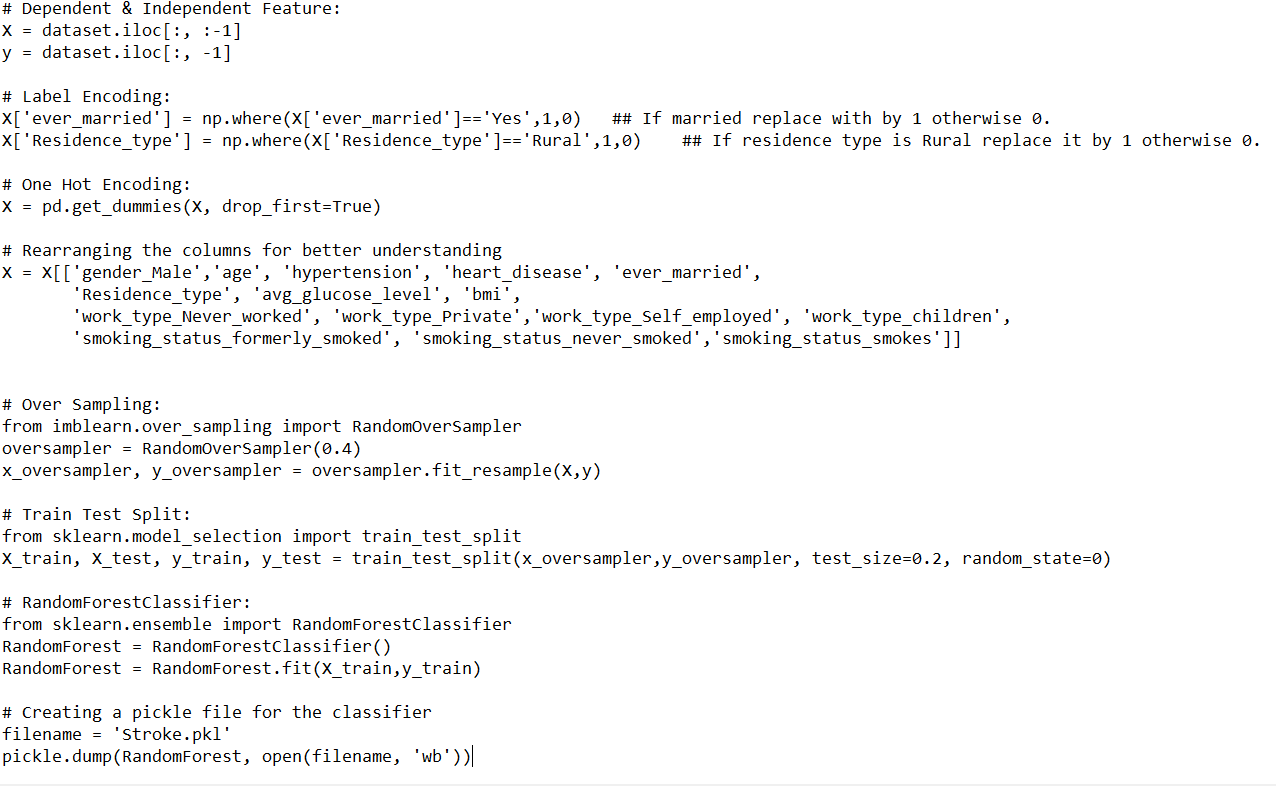
When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

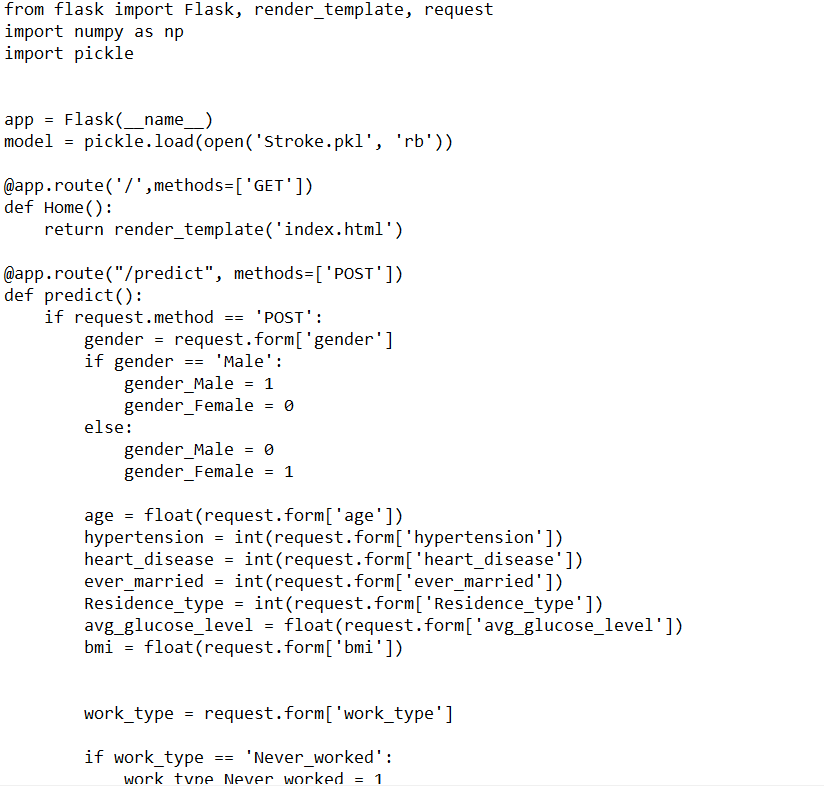
### Output Design

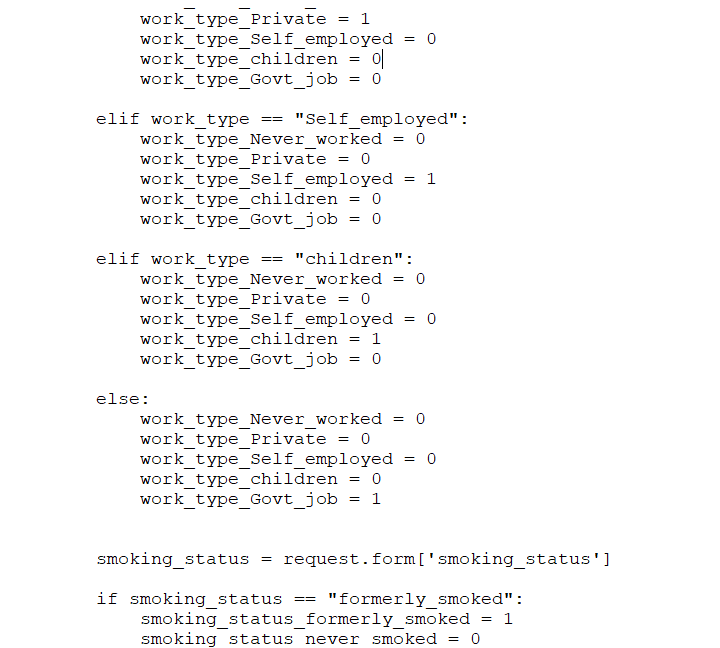
A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

* 1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
  2. Select methods for presenting information.
  3. Create document, report, or other formats that contain information produced by the system.

**Code**







# Chapter-8

# System testing

## 8.1 INTRODUCTION:

Testing is the debugging program is one of the most critical aspects of the computer programming triggers, without programming that works, the system would never produce an output of which it was designed. Testing is best performed when user development is asked to assist in identifying all errors and bugs. The sample data are used for testing. It is not quantity but quality of the data used the matters of testing. Testing is aimed at ensuring that the system was accurately an efficiently before live operation commands.

Testing objectives:

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say, testing is a process of executing a program with intent of finding an error.

1. A successful test is one that uncovers an as yet undiscovered error.
2. A good test case is one that has probability of finding an error, if it exists.
3. The test is inadequate to detect possibly present errors.
4. The software more or less confirms to the quality and reliable standards.

## 8.2 Levels of Testing

**Code testing:**

This examines the logic of the program. For example, the logic for updating various sample data and with the sample files and directories were tested and verified.

**Specification Testing:**

Executing this specification starting what the program should do and how it should performed under various conditions. Test cases for various situation and combination of conditions in all the modules are tested.

**Unit testing:**

In the unit testing we test each module individually and integrate with the overall system. Unit testing focuses verification efforts on the smallest unit of software design in the module. This is also known as module testing. The module of the system is tested separately. This testing is carried out during programming stage itself. In the testing step each module is found to work satisfactorily as regard to expected output from the module. There are some validation checks for fields also. For example the validation check is done for varying the user input given by the user which validity of the data entered. It is very easy to find error debut the system.

Each Module can be tested using the following two Strategies:

1. Black Box Testing
2. White Box Testing

### 8.2.1 BLACK BOX TESTING

What is Black Box Testing?

Black box testing is a software testing techniques in which functionality of the software under test (SUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on the software requirements and specifications.

In Black Box Testing we just focus on inputs and output of the software system without bothering about internal knowledge of the software program.



The above Black Box can be any software system you want to test. For example : an operating system like Windows, a website like Google ,a database like Oracle or even your own custom application. Under Black Box Testing , you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

### 8.2.2 WHITE BOX TESTING

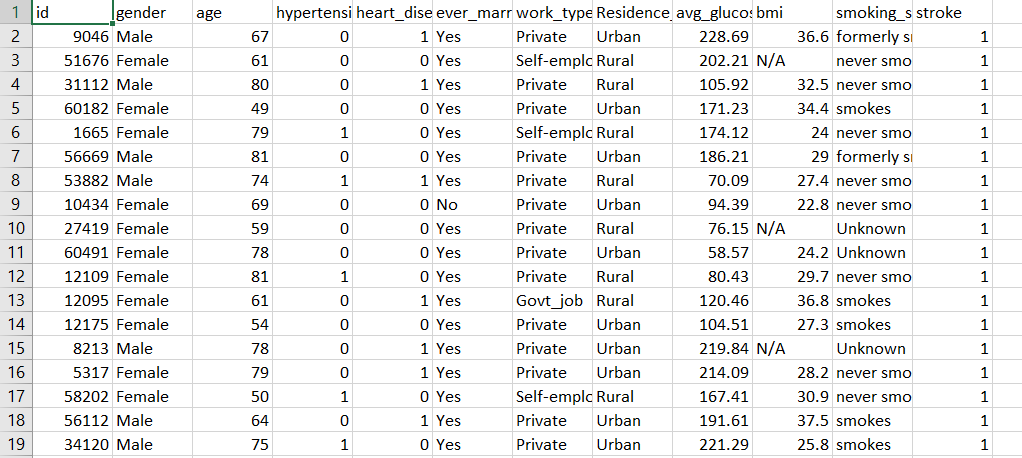
White Box Testing is the testing of a software solution's internal coding and infrastructure. It focuses primarily on strengthening security, the flow of inputs and outputs through the application, and improving design and usability.White box testing is also known as clear, open, structural, and glass box testing.

It is one of two parts of the "box testing" approach of software testing. Its counter-part, blackbox testing, involves testing from an external or end-user type perspective. On the other hand, Whitebox testing is based on the inner workings of an application and revolves around internal testing. The term "whitebox" was used because of the see-through box concept. The clear box or whitebox name symbolizes the ability to see through the software's outer shell (or "box") into its inner workings. Likewise, the "black box" in "black box testing" symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested.

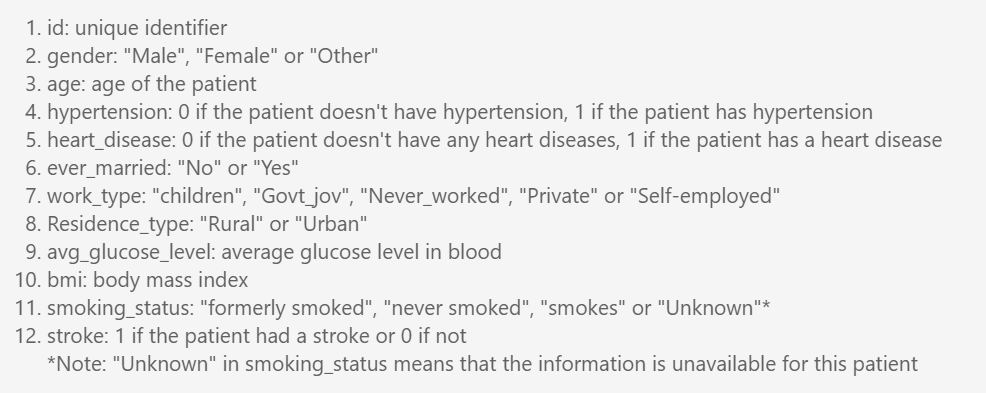
# CHAPTER-9

# Output Screens

## 9.1 Dataset SCREEN



**9.2 Dataset types**

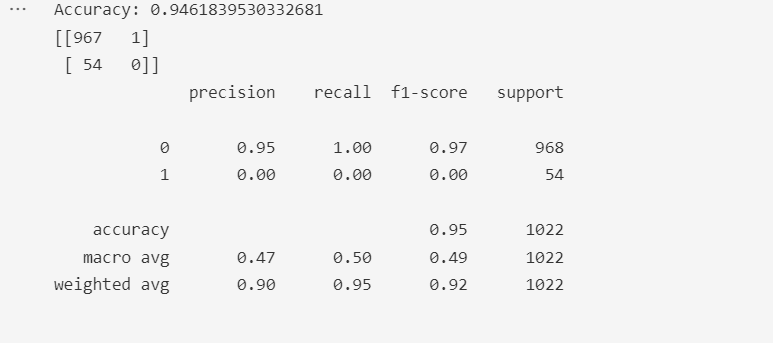


**9.3 correlation graph:**

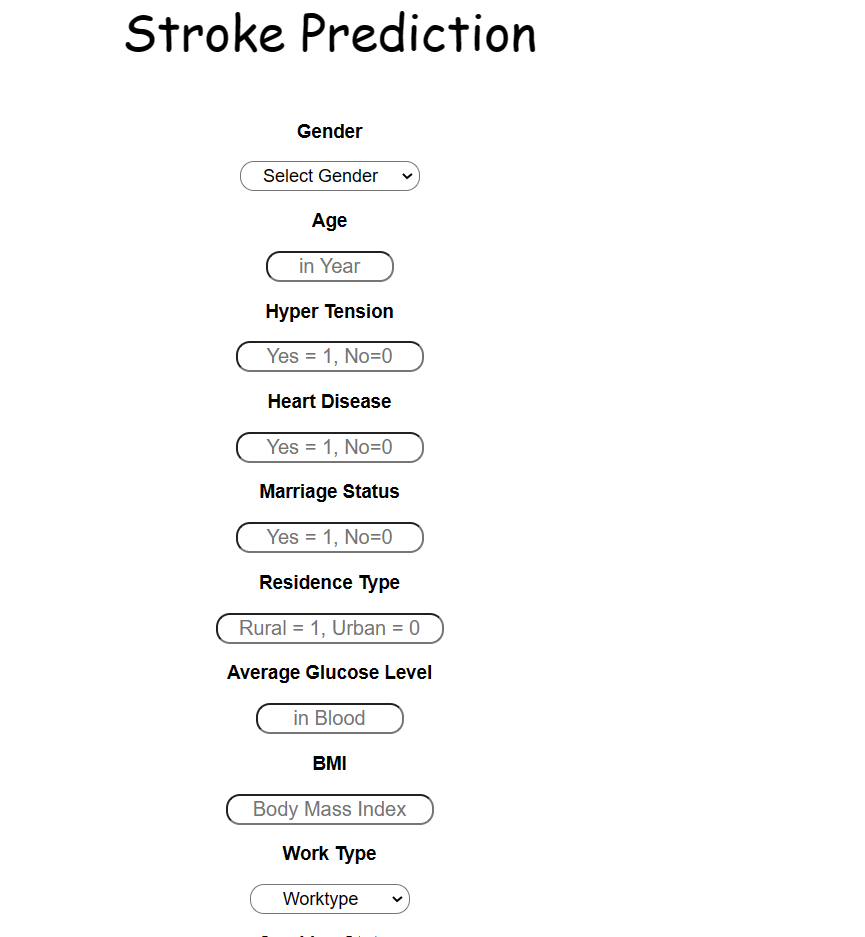
****

**Accuracy of the model which shows 89 percent accuracy**

**9.4 Prediction:**

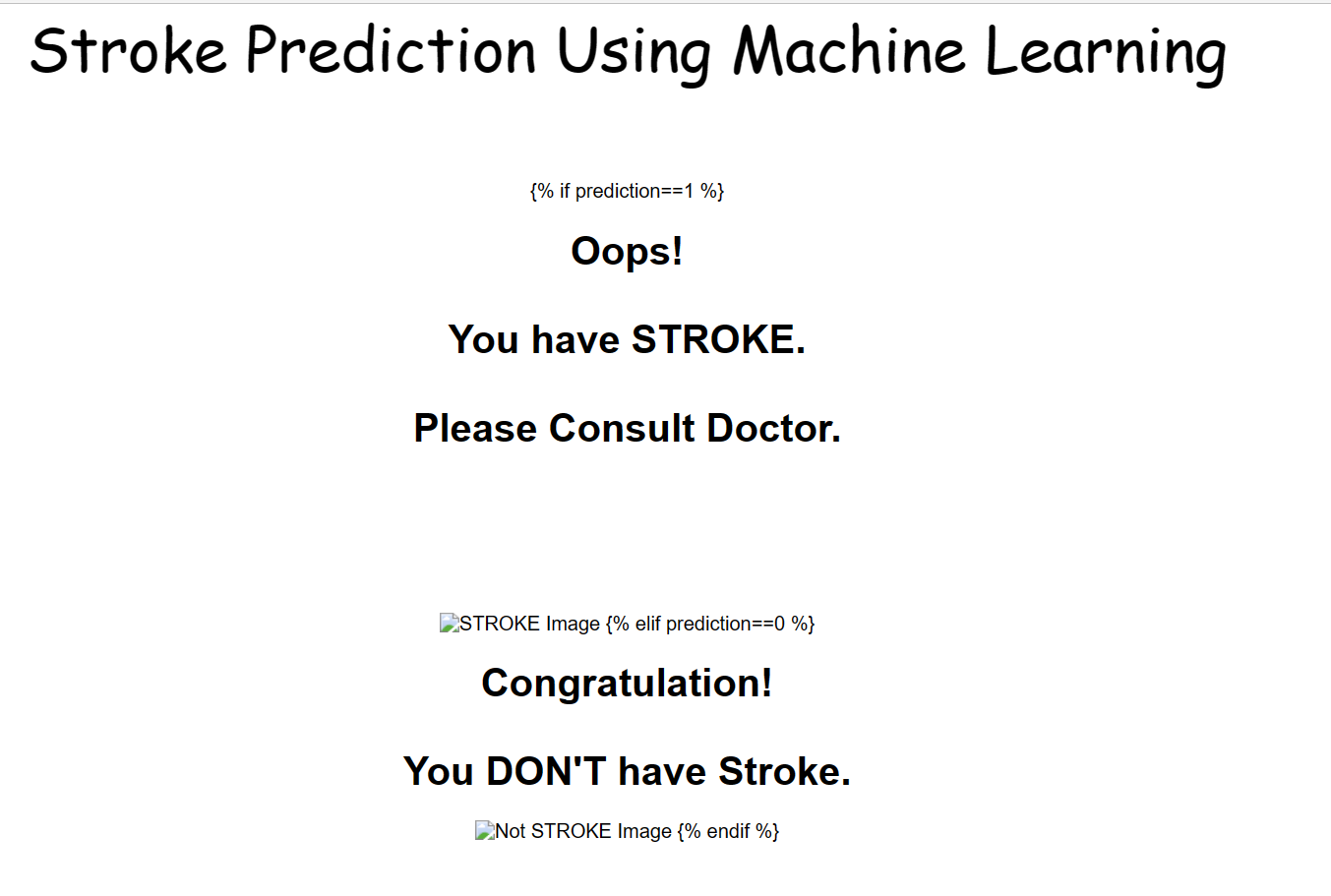
****

**9.5 Flask Webpage:**



Flask webpage to predict

**9.6 Predict result**

****

Predicted result

# 

# CONCLUSION and Future work

Before it worsens. Building a machine learning model can help in the early prediction of stroke and reduce the severe impact of the future. This paper shows the performance of various machine learning algorithms in successfully predicting stroke based on multiple physiological attributes. Out of all the algorithms chosen, Random forest Classification performs best with an accuracy of 95%.

# REFERENCES

Dataset named ‘Stroke Prediction Dataset’ from Kaggle: https://www.kaggle.com/fedesoriano/stroke-prediction-dataset. [4] Singh, M.S., Choudhary, P., Thongam, K.: A comparative analysis for various stroke prediction techniques. In: Springer, Singapore (2020).

[5] 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).

[6] Vamsi Bandi, Debnath Bhattacharyya, Divya Midhunchakkravarthy: Prediction of Brain Stroke Severity Using Machine Learning. In: International Information and Engineering Technology Association (2020).

[7] Nwosu, C.S., Dev, S., Bhardwaj, P., Veeravalli, B., John, D.: Predicting stroke from electronic health records. In: 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society IEEE (2019).

[8] Fahd Saleh Alotaibi: Implementation of Machine Learning Model to Predict Heart Failure Disease. In: International Journal of Advanced Computer Science and Applications (IJACSA) (2019).

[9] Ohoud Almadani, Riyad Alshammari: Prediction of Stroke using Data Mining Classification Techniques. In: International Journal of Advanced Computer Science and Applications (IJACSA) (2018).

[10] Kansadub, T., Thammaboosadee, S., Kiattisin, S., Jalayondeja, C.: Stroke risk prediction model based on demographic data. In: 8th Biomedical Engineering International Conference (BMEiCON) IEEE (2015).

[11] Aditya Khosla, Yu Cao, Cliff Chiung-Yu Lin, Hsu-Kuang Chiu, Junling Hu, Honglak Lee:An Integrated Machine Learning Approach to Stroke Prediction. In: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining (2010).

[12] Shanthi, D., Sahoo, G., Saravanan, N.: Designing an artificial neural network model for the prediction of thrombo-embolic stroke. Int. J. Biometric Bioinform. (IJBB) (2009)