

# **RECOGNITION AND LISTING OF ACUTE STROKE PROGRESSION BASED ON OCT IMAGES USING CURVELET ANALYSIS**

Submitted in partial fulfillment of the requirements for the award of  
Bachelor of Engineering degree in Computer Science and Engineering

by

S.V. CHAITANYA (Reg. No. 38110498)

A. RANADHEERAJ (Reg. No. 3811057)



**DEPARTMENT OF COMPUTER SCIENCE AND  
ENGINEERING SCHOOL OF COMPUTING**

## **SATHYABAMA**

**INSTITUTE OF SCIENCE AND TECHNOLOGY  
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade “A” by NAAC  
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,  
CHENNAI – 600 119**

**MARCH - 2020**



# **SATHYABAMA**

INSTITUTE OF SCIENCE AND TECHNOLOGY  
(DEEMED TO BE UNIVERSITY)

Accredited with “A” grade by NAAC  
JeppiaarNagar, Rajiv Gandhi Salai, Chennai – 600 119

[www.sathyabama.ac.in](http://www.sathyabama.ac.in)



## **DEPARTMENT OF COMPUTER SCIENCEENGINEERING**

### **BONAFIDE CERTIFICATE**

This is to certify that this Project Report is the bonafide work of who carried  
S.V. CHAITANYA (Reg. No. 38110498)A. RANADHEERAJ  
(Reg. No. 3811057) out the project entitled “**RECOGNITION AND  
LISTING OF ACUTE STROKE PROGRESSION BASED ON OCT  
IMAGES USINGCURVELET ANALYSIS**” under our supervision from Nov  
2019 to April 2020.

**Internal Guide**

**Dr. A.C. SANTHA SHEELA M.Tech**

**Head of the Department**

---

**Submitted for Viva voce Examination held on\_\_\_\_\_**

**Internal Examiner**

**External Examiner**

## **DECLARATION**

I **M.R.V.VIJAY** hereby declare that the Project Report entitled “**RECOGNITION AND LISTING OF ACUTE STROKE PROGRESSION BASED ON OCT IMAGES USING CUREVELET ANALYSIS**” is done by me under the guidance of **Dr. A.C. SANTHA SHEELA M.Tech.**, Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

**DATE:**

**PLACE: CHENNAI**

**SIGNATURE OF THE CANDIDATE**

## ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean**, School of Computing, **Dr. S. Vigneswari, M.E., Ph.D.,** and **Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. A.C. SANTHA SHEELA M.Tech.,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

## **ABSTRACT**

The estimation of multivariate distributions of from high dimensional data with low sample size is at the core of many image analysis applications. The proposed system estimates a multivariate Gaussian distribution of diffusion tensor features in a set of brain regions. This distribution is used to identify the imaging abnormalities in subjects with brain injury.

Based on endogenous optical scattering signals provided by OCT imaging, we have developed a single, integrated imaging platform enabling the measurement of changes in blood perfusion, blood flow, erythrocyte velocity, and light attenuation within cortical tissue, during focal cerebral ischemia in a mouse model. During the acute phase (from 5 minutes to the first few hours following blood occlusion), the multi-parametric OCT imaging revealed multiple hemodynamic and tissue scattering responses in vivo, including cerebral blood flow deficits, capillary non-perfusion, displacement of penetrating vessels, and increased light attenuation in the cortical tissue at risk that are spatially correlated with the infarct core, as determined by postmortem staining with triphenyltetrazolium chloride (TTC).

## TABLE OF CONTENTS

| CHAPTER No. | TITLE                                | PAGE No.  |
|-------------|--------------------------------------|-----------|
|             | <b>ABSTRACT</b>                      | <b>v</b>  |
|             | <b>LIST OF FIGURES</b>               | <b>ix</b> |
| <b>1.</b>   | <b>INTRODUCTION</b>                  | <b>1</b>  |
|             | 1.1 DATA IMPORT AND PREPROCESSING    | 1         |
|             | 1.2 DATA AUGMENTATION                | 1         |
|             | 1.3 MODEL BUILDING                   | 2         |
|             | 1.4 MODEL PERFORMANCE                | 3         |
| <b>2.</b>   | <b>LITERATURE SURVEY</b>             | <b>4</b>  |
|             | 2.1 LITERATURE 1                     | 5         |
|             | 2.2 LITERATURE 2                     | 6         |
|             | 2.3 LITERATURE 3                     | 7         |
|             | 2.4 LITERATURE 4                     | 8         |
|             | 2.5 LITERATURE 5                     | 9         |
| <b>3.</b>   | <b>SYSTEM ANALYSIS</b>               | <b>10</b> |
|             | 3.1 EXISTING SYSTEM                  | 10        |
|             | 3.2 DRAWBACKS OF EXISTING PLAN       | 10        |
|             | 3.3 PROPOSED SYSTEM                  | 11        |
|             | 3.4 ADVANTAGE OF PROPOSED SYSTEM     | 11        |
|             | 3.5 TECHNOLOGY USED                  | 13        |
|             | 3.6 EXISTING ALGORITHM               | 13        |
|             | 3.7 PROPOSED ALGORITHM               | 13        |
|             | 3.8 ADVANTAGES OF PROPOSED ALGORITHM | 13        |
|             | 3.9 WATERFALL MODEL                  | 15        |
|             | 3.10 REQUIREMENT GATHERING           | 15        |
|             | 3.11 SYSTEM DESIGN                   | 16        |
|             | 3.12 IMPLEMENTATION                  | 16        |
|             | 3.13 TESTING                         | 16        |

|           |                                                         |           |
|-----------|---------------------------------------------------------|-----------|
|           | 3.14 DELIVERY/ DEVELOPMENT                              | 16        |
|           | 3.15 MAINTANENCE                                        | 16        |
|           | 3.16 RAD MODEL                                          | 17        |
|           | 3.17 PHASE:1 REQUIREMENTS PLANNING                      | 17        |
|           | 3.18 PHASE: 2 USER DESIGN                               | 18        |
|           | 3.19 PHASE: 3 RAPID CONSTRUCTION                        | 19        |
|           | 3.20 PHASE: 4 CUTOVER                                   | 19        |
| <b>4.</b> | <b>RESULTS AND DISCUSSION</b>                           | <b>21</b> |
|           | 4.1 .DATABASE STORAGE                                   | 21        |
|           | 4.2 ENROLLING DATA INTO JUPYTER PLATFORM                | 21        |
|           | 4.3 OVERVIEW OF PATIENT MEDICAL DETAILS                 | 23        |
| <b>5</b>  | <b>TESTING</b>                                          |           |
|           | 5.1 TESTING                                             | 24        |
|           | 5.2 BENEFITS OF USING DOCUMENTATION                     | 24        |
|           | 5.3 FUNCTIONAL TESTING TYPES AND<br>PERFORMANCE TESTING | 31        |
|           | 5.4 PERFORMANCE TESTING                                 | 32        |
| <b>6</b>  | <b>LANGUAGE DESCRIPTION</b>                             | <b>43</b> |
|           | 6.1 ABOUT PYTHON                                        | 43        |
|           | 6.2 PYTHON LINE STRUCTURE                               | 44        |
|           | 6.3 EXPLICIT LINE JOINING                               | 45        |
|           | 6.4 IMPLICIT LINE JOINING                               | 45        |
|           | 6.5 WHITESPACES AND IDENTATION                          | 45        |
|           | 6.6 PYTHON DATA TYPES                                   | 47        |
|           | 6.7 ADVANTAGES OF PYTHON                                |           |
|           | 6.8 SELECTION OF INTEGRATED DEVELOPMENT<br>ENVIRONMENT  | 49        |
|           | 6.9 LICLIPSE                                            | 51        |

|                 |                                         |           |
|-----------------|-----------------------------------------|-----------|
| 6.10            | SFKHLSEHFLIDS                           | 51        |
| 6.11            | SELECTION OF OPERATING SYSTEM           | 54        |
| 6.12            | SELECTION OF WEB APPLICATION FRAME WORK |           |
| 6.12.1          | FULL STACK FRAME WORK                   | 52        |
| 6.12.2          | NON-FULL STACK                          | 52        |
| 6.12.3          | MICROFRAME WORK                         | 53        |
| 6.12.4          | ASYNCHRONOUS FRAME WORK                 | 54        |
| 6.13            | SELECTION BROWSER                       | 55        |
| 6.14            | ABOUT PANDAS                            | 56        |
| 6.15            | NUMPY                                   | 59        |
| 6.16            | MATHPLOTLIB                             | 61        |
| 6.17            | MACHINE LEARNING                        | 62        |
| 6.17.1          | SUPERVISED LEARNING                     | 64        |
| 6.18            | CONCLUSION                              | 66        |
| <b>6.19</b>     | <b>REFERENCES</b>                       | <b>67</b> |
| <br>            |                                         |           |
| <b>APPENDIX</b> |                                         | <b>68</b> |
| A.              | SOURCE CODE                             |           |

## LIST OF FIGURES

| FIGURE No. | FIGURE NAME                   | PAGE No. |
|------------|-------------------------------|----------|
| 3.1        | ARCHITECTURE                  | 12       |
| 3.2        | WATERFALL MODEL               | 15       |
| 3.3        | RAPID APPLICATION DEVELOPMENT | 17       |
| 4.1        | DATABASE STORAGE              | 21       |
| 4.2        | ENROLLING DATA INTO JUPYTER   | 22       |
| 4.3        | GRAPHICAL REPRESENTATION      | 22       |
| 4.4        | FINAL OUTPUT                  | 23       |
| 5.1        | TESTING CYCLE                 | 27       |
| 5.2        | POPULAR UNIT TESTING TOOLS    | 29       |
| 5.3        | INTEGRATION TESTING           | 34       |
| 5.4        | SYSTEM TESTING                | 35       |
| 5.5        | PERFORMANCE TESTING           | 40       |
| 6.1        | PYTHON ARCHITECTURE-1         | 42       |
| 6.2        | PYTHON ARCHITECTURE-2         | 43       |
| 6.3        | MACHINE LEARNING ALGORITHM    | 63       |
| 6.4        | SUPERVISED LEARNING           | 64       |

# CHAPTER 1

## INTRODUCTION

### 1.1 Data Import and Preprocessing

Pre-processing is a common name for operations with images at the lowest level of abstraction both input and output are intensity images. The aim of pre-processing is an improvement of the image data that suppresses unwanted distortions or enhances some image features important for further processing.

Convert color images to grayscale to reduce computation complexity: in certain problems you'll find it useful to lose unnecessary information from your images to reduce space or computational complexity.

For example, converting your colored images to grayscale images. This is because in many objects, color isn't necessary to recognize and interpret an image. Grayscale can be good enough for recognizing certain objects. Because color images contain more information than black and white images, they can add unnecessary complexity and take up more space in memory (Remember how color images are represented in three channels, which means that converting it to grayscale reduces the number of pixels that need to be processed).

One important constraint that exists in some machine learning algorithms, such as CNN, is the need to resize the images in your dataset to a unified dimension. This implies that our images must be preprocessed and scaled to have identical widths and heights before fed to the learning algorithm.

### 1.2 Data Augmentation

We use effective methods that you can use to build a powerful image classifier, using only very few training examples --just a few hundred or thousand pictures from each class you want to be able to recognize. In order to make the

most of our few training examples, we will "augment" them via a number of random transformations, so that our model would never see twice the exact same picture. This helps prevent overfitting and helps the model generalize better.

The right tool for an image classification job is a convnet, so let's try to train one on our data, as an initial baseline. Since we only have few examples, our number one concern should be overfitting. Overfitting happens when a model exposed to too few examples learns patterns that do not generalize to new data, i.e. when the model starts using irrelevant features for making predictions. For instance, if you, as a human, only see three images of people who are lumberjacks, and three, images of people who are sailors, and among them only one lumberjack wears a cap, you might start thinking that wearing a cap is a sign of being a lumberjack as opposed to a sailor. You would then make a pretty lousy lumberjack/sailor classifier.

Data augmentation is one way to fight overfitting, but it isn't enough since our augmented samples are still highly correlated. Your main focus for fighting overfitting should be the entropic capacity of your model --how much information your model is allowed to store. A model that can store a lot of information has the potential to be more accurate by leveraging more features, but it is also more at risk to start storing irrelevant features. Meanwhile, a model that can only store a few features will have to focus on the most significant features found in the data, and these are more likely to be truly relevant and to generalize better.

There are different ways to modulate entropic capacity. The main one is the choice of the number of parameters in your model, i.e. the number of layers and the size of each layer. Another way is the use of weight regularization, such as L1 or L2 regularization, which consists in forcing model weights to take smaller values.

### **1.3 Model Building**

VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". The model achieves

92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It was one of the famous model submitted to ILSVRC-2014. It makes the improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. VGG16 was trained for weeks and was using NVIDIA Titan Black GPU's.

The input to cov1 layer is of fixed size 224 x 224 RGB image. The image is passed through a stack of convolutional (conv.) layers, where the filters were used with a very small receptive field: 3×3 (which is the smallest size to capture the notion of left/right, up/down, center). In one of the configurations, it also utilizes 1×1 convolution filters, which can be seen as a linear transformation of the input channels (followed by non-linearity). The convolution stride is fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1-pixel for 3×3 conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the conv. layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2×2 pixel window, with stride 2.

ModelCheckpoint helps us to save the model by monitoring a specific parameter of the model. In this case I am monitoring validation accuracy by passing val\_acc to ModelCheckpoint. The model will only be saved to disk if the validation accuracy of the model in current epoch is greater than what it was in the last epoch.

## **1.4 Model Performance**

As we train your classification predictive model, we want to assess how good it is. Interestingly, there are many different ways of evaluating the performance. Most data scientists that use Python for predictive modeling use the Python package called scikit-learn. Scikit-learn contains many built-in functions for analyzing the performance of models.

`confusion_matrix`

Given an actual label and a predicted label, the first thing we can do is divide our

samples in 4 buckets:

True positive — actual = 1, predicted = 1

False positive — actual = 0, predicted = 1

False negative — actual = 1, predicted = 0

True negative — actual = 0, predicted = 0

## **CHAPTER 2**

### **LITERATURE SURVEY**

#### **2.1 LITERATURE 1**

**TITLE: AN OFB-BASED FRAGILE WATERMARKING SCHEME FOR 3D POLYGONAL MESHES.**

**AUTHORS:**

Jen-Tse Wang, Yi-Ching Chang, Chih-Wei Lu, Shyr-Shen Yu

**PUBLISHED YEAR:** October 2016

**EFFICIENCY:**

- Achieve both the authentication and verification
- Simple to implement

**DRAWBACKS:**

- Fails in locating the changed regions.

**DESCRIPTION:**

Watermarking is an important technique for protecting the ownership and authorization of digital content. In this paper, an output feedback (OFB) based fragile watermarking scheme for 3D models in spatial domain is proposed. The proposed scheme employs the output feedback (OFB) to generate the embedded watermark. The least significant bit (LSB) substitution technique is employed for watermark data embedding. Instead of the hash function, the tampering region can be verified and located by the output feedback (OFB) based watermark check. The output feedback (OFB) based watermark is embedded to achieve both the authentication and verification. Experimental results show that the proposed scheme can achieve authentication of the original model and tampering detection of the stego model with insignificant visual distortion. Moreover, the verification information is not needed for verification task of the proposed scheme.

## **2.2 LITERATURE 2**

**TITLE: AUDIO ZERO WATERMARKING SCHEME BASED ON SUB BAND MEAN ENERGY COMPARISON USING DWT-DCT**

**AUTHORS:**

Jeebananda Panda, Shrikishan Choudhary, Kaberi Nath, Dr. Sunil Kumar

**PUBLISHED YEAR:** June 2018

**EFFICIENCY:**

- Watermark does not modify the original audio file
- Provides high robustness
- Affect imperceptibility

**DRAWBACKS:**

- More robust against the attacks

**DESCRIPTION:**

Digital format of data has become very popular now a days with the advancement of technology. But the problem arises when unauthorized copying and distribution of these digital contents took place. As a result the question of copyright protection came. In order to cope with this situation, the era of Digital Watermarking came where the authorized owner hides some kind of information in an imperceptible way into the digital data. Digital Audio Watermarking is a technique to hide information in a host audio file in such a way that it is imperceptible to the listener. It is used for the copyright protection, tamper proofing of any multimedia file. In this paper a robust and efficient audio zero-watermarking algorithm is introduced. Audio zero-watermarking is an embedding technique in which the watermark does not modify the original audio file as it is hidden into a secret key. In Experimental results show that the proposed scheme is efficient and the watermarked signal is indistinguishable from the host audio signal as no modification is done.

## **2.3 LITERATURE 3**

**TITLE:** COLLUSION ATTACK RESISTANT SVD BASED SECURED AUDIO WATERMARKING SCHEME WITH CHAOTIC ENCRYPTION

**AUTHORS:** Tribhuwan Kumar Tewari , Vikas Saxena

**PUBLISHED YEAR:** October 2018

**EFFICIENCY:**

- Produces security as well as collusion attack resistance
- The intruder can't remove the watermark

**DRAWBACKS:**

- Not sufficient for protecting

**DESCRIPTION:**

In this paper, a blind, robust, and inaudible audio watermarking algorithm based on Singular Value Decomposition (SVD) is proposed. The algorithm produce multiple copies with different fingerprints/ watermarks on the same audio using different SVD values making it robust to collusion attack. The proposed algorithm uses modified mean value and quantization index modulation technique of the singular values for watermark embedding. Instead of using the SVD matrix or the SVD matrix in cascade with the other transform directly for watermark embedding, the proposed schemes uses Baker's Chaotic mapping for encryption of the matrix without the cost of imperceptibility. The proposed method thus produces security as well as collusion attack resistance over all the SVD based methods where the location for the watermark embedding is known as the diagonal elements of the SVD matrix produced from the audio directly. Experimental results presented in this paper demonstrate the effectiveness of the proposed algorithm against signal processing attacks with good imperceptibility.

## 2.4 LITERATURE 4

**TITLE:** ROBUST WATERMARKING OF RELATIONAL DATABASES WITH ONTOLOGY-GUIDED DISTORTION CONTROL

**AUTHORS:** Javier Franco, Gouenou Coatrieux

**PUBLISHED YEAR:** September 2017

**EFFICIENCY:**

- Do not modify attributes
- Impact of the deletion attack variably depends on each tuple

**DRAWBACKS:**

- Embedder does not insert any symbols of the message
- Modification is carried out by means of an iterative process

**DESCRIPTION:**

In this paper, we present a new robust database watermarking scheme the originality of which stands on a semantic control of the data distortion and on the extension of quantization index modulation (QIM) to circular histograms of numerical attributes. The semantic distortion control of the embedding process we propose relies on the identification of existing semantic links in between values of attributes in a tuple by means of an ontology. By doing so, we avoid incoherent or very rare record occurrences which may bias data interpretation or betray the presence of the watermark. In a second time, we adapt QIM to database watermarking. Watermark embedding is conducted by modulating the relative angular position of the circular histogram center of mass of one numerical attribute. We theoretically demonstrate the robustness performance of our scheme against most common attacks (i.e., tuple insertion and deletion). This makes it suitable for copyright protection, owner identification, or traitor tracing purposes. We further verify experimentally these theoretical limits within the framework of a medical database of more than one half million of inpatient hospital stay records. Under the assumption imposed by the central limit theorem, experimental results.

## 2.5 LITERATURE 5

### **TITLE: TARGETED WATERMARK REMOVAL OF A SVD-BASED IMAGE WATERMARKING SCHEME**

**AUTHORS:** Pegah Nikbakht, Mojtaba Mahdavi

**PUBLISHED YEAR:** November 2018

#### **EFFICIENCY:**

- Enable us to store, transfer and process digital content in less time
- Watermarking scheme are robustness and security

#### **DRAWBACKS:**

- Attacker needs to know the watermarking algorithm
- Attack tries to change the exact space

#### **DESCRIPTION:**

Digital watermarking is widely used to verify the authenticity or integrity of multimedia contents, such as images and videos. In the last few years, a well-known numerical tool called singular value decomposition (SVD) has received much attention from the watermarking community. Designers of SVDbased schemes usually use dewatermarking tools such as Stirmark to prove the robustness of their schemes. Although, these tools are valuable but their attacks usually reduce the quality of watermarked images. On the other hand, there is another group of attacks in which the attacker needs to know the watermarking algorithm to perform the attack. In this paper, an attack in this group has been designed to remove the watermark from a specific SVD-based watermarking scheme [1]. The experiments show our attack makes fewer changes to remove embedded watermark than other distortions available in Stirmark, so the quality of attacked image is better than processed image by using Stirmark in terms of PSNR.

## **CHAPTER 3**

### **SYSTEM ANALYSIS**

#### **3.1 EXISTING SYSTEM**

Functional optical coherence tomography (OCT) has emerged as a potential candidate for monitoring acute ischemic strokes in preclinical investigations. OCT is a noninvasive, label-free optical imaging technique capable of resolving microanatomy of living biological tissues over a field-of-view of several millimeters, with micron scale resolution.

The OCT signal received from the tissue sample is a result of linear superposition of static, dynamic scattering components and additive system noise, in which the dynamic scattering component can be assumed to be due mainly to moving RBC scatters. OCTA works to extract the moving scattering elements of the OCT signal in order to localize blood perfusion within the scanned tissue.

To process the data, differential operations between the adjacent frames of the repeated ensemble B-scans were performed, and the absolute values were averaged along the transverse y location. Finally, the processed OCTA cubic data was collapsed into two-dimensional data with a maximum amplitude projection (MAP), where the highest amplitude along the depth was chosen for each A-line, creating an en face (x-y) vascular map involving the perfused microvessel network.

#### **3.2 DRAWBACKS OF EXISTING SYSTEM**

- High Computation cost
- Need Longer time to detect
- Not a cost effective solution
- Not Accurate
- Cannot handle unbalanced data

### **3.3 PROPOSED SYSTEM**

This system propose a new algorithm and a set of features to help clinicians determine which patients evaluated for TBI need a head CT scan using cost sensitive random forest (CSRF) classifier. We show that random forest (RF) and CSRF are useful methods for identifying patients likely to have a positive head CT scan.

The proposed algorithm has superior diagnostic accuracy in comparison to the Canadian head CT algorithm, which is currently the most accurate and widely used algorithm for determining which TBI patients need a head CT scan.

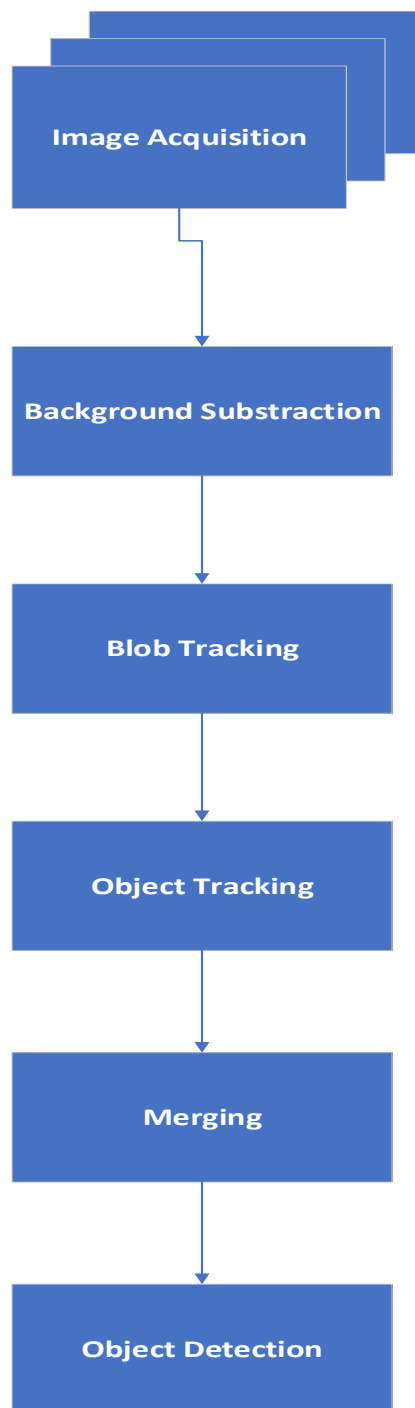
In bagging methods, the input data is re-sampled using a method called re-sampling by replacement (boot-strapping) and then many of the trees are constructed in a way that each tree is independent of the already constructed trees. Each tree computes a response(vote) and final prediction will be computed based on majority voting.

In addition to use of different boot-strap sample of input data, RF also use a different way to construct the classification trees. RF chooses the best split in each node among all the subsets of the variables that randomly selected at that point, while in the standard method, the best split selection is based on best split on the all variables.

### **3.4 ADVANTAGES OF PROPOSED SYSTEM**

- It is one of the most accurate learning algorithms available. ...
- It runs efficiently on large databases.
- It can handle thousands of input variables without variable deletion.
- It gives estimates of what variables are important in the classification.
- Quick Prediction/Training Speed
- Robust to Outliers and Non-linear Data

### 3.1 ARCHITECTURE



***Fig.3.1 : ARCHITECTURE***

### **3.5 TECHNOLOGY USED**

#### ***Backend Technologies***

- Python
- Numpy
- Sci-learn
- Eclipse IDE

#### ***Frontend Technologies***

- Web Technologies
- Bootstrap

### **3.6 EXISTING ALGORITHM**

- Doppler Algorithm

### **3.7 PROPOSED ALGORITHM**

- Cost sensitive random forest

### **3.8 ADVANTAGES OF PROPOSED ALGORITHM**

- Impressive in Versatility
- Parallelizable
- Great with High dimensionality

One of the primary choices we face for our project implementations is “Which development methodology should we use?” There are various software

development methodologies are available in the ICT industry. They are

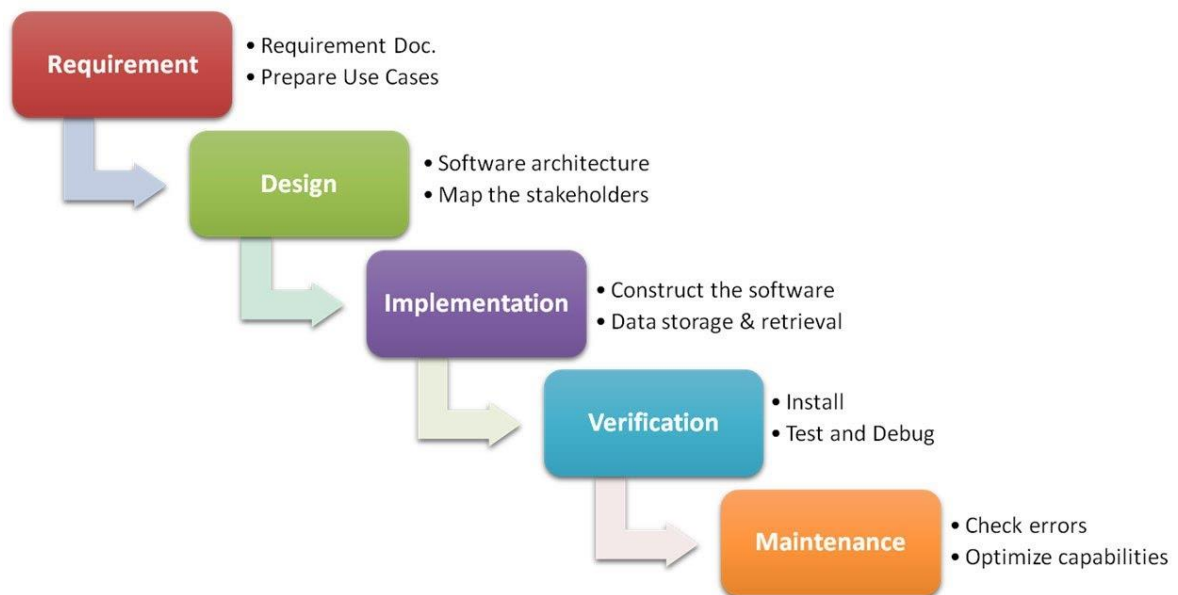
- Agile Software Development
- Crystal Methods
- Dynamic Systems Development Model (DSDM)
- Extreme Programming (XP)
- Feature Driven Development (FDD)
- Joint Application Development (JAD)
- Lean Development (LD)
- Rapid Application Development (RAD)
- Rational Unified Process (RUP)
- Scrum
- Spiral
- Systems Development Life Cycle (SDLC)
- Waterfall (a.k.a. Traditional)

Among all these methodologies, we have decided to compare the two most popular and widely used methodologies:

- Waterfall Model
- RAD Model

### 3.9 WATERFALL MODEL

It is called as traditional approach. Waterfall is a linear method (sequential model) for any software application development. Here application development is segregated into a sequence of pre-defined phases.



**Fig. 3.2: WATERFALL MODEL (TAKEN FROM GOOGLE.COM)**

(Reference: <https://www.lucidchart.com/blog/waterfall-project-management-methodology>)

### 3.10 REQUIREMENT GATHERING AND DOCUMENTATION

In this stage, you should gather comprehensive data approximately what this challenge requires. You may gather this data in a diffusion of ways, from interviews to questionnaires to interactive brainstorming. By means of the stop of this section, the task requirements have to be clean, and also you have to have a necessities file that has been allotted on your team.

### **3.11 SYSTEM DESIGN**

Using the well-known requirements, your team designs the solutions. During this phase, no development will be happening. But the project team starts specification such as programming language or hardware requirements.

### **3.12 IMPLEMENTATION**

During this phase software development coding will be happening. Web application programmers take data from the previous stage and create a functional product. Web application programmers write source code in small pieces, which are integrated at the end of this phase or the beginning of the next.

### **3.13 TESTING**

Once all coding is done, testing of the product can begin. Testers methodically find and report any problems. If serious issues arise, your project may need to return to phase one for reevaluation.

### **3.14 DELIVERY/DEPLOYMENT**

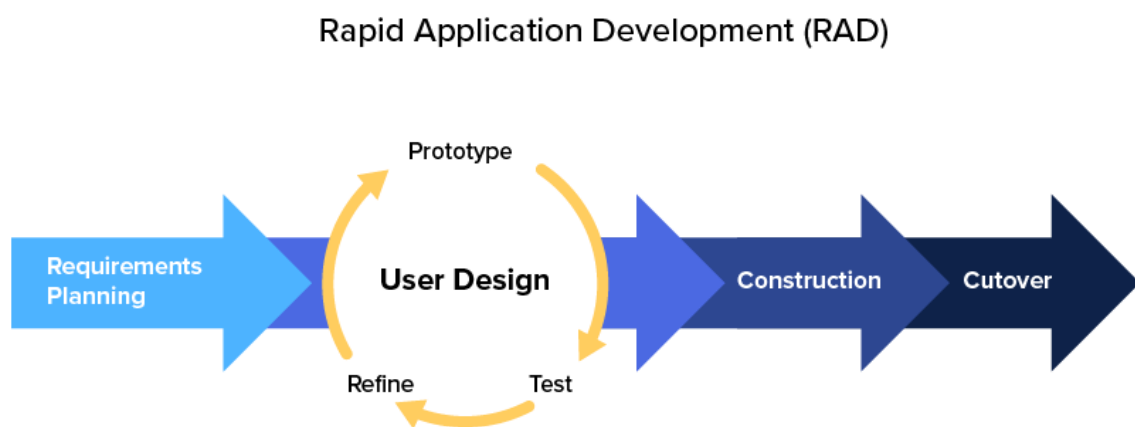
In this phase, the solution is complete, and your project team submits the deliverables to be deployed or released.

### **3.15 MAINTENANCE**

The final solution has been implemented to the client and is being used. As troubles arise, the project team might also want to create patches and updates might also to deal with them. Again, huge troubles may also necessitate a return to segment one.

### 3.16 RAD MODEL

Agile is an iterative, team-based approach to development. This method highlights the speedy delivery of an software application in comprehensive functional components.



**Fig. 3.3: RAPID APPLICATION DEVELOPMENT (RAD)**

References (<https://www.lucidchart.com/blog/rapid-application-development-methodology>)

### 3.17 PHASE 1: REQUIREMENTS PLANNING

This phase is equivalent to a project scoping meeting. Although the planning phase is condensed compared to other project management methodologies, this is a critical step for the ultimate success of the project.

During this stage, web application programmers, clients (software users), and team members communicate to determine the goals and expectations for the project as well as current and potential issues that would need to be addressed

during the build.

A basic breakdown of this stage involves:

Researching the current problem

Defining the requirements for the project

Finalizing the requirements with each stakeholder's approval

It is important that everyone has the opportunity to evaluate the goals and expectations for the project and weigh in. By getting approval from each key stakeholder and web application programmer, teams can avoid miscommunications and costly change orders down the road.

### **3.18 PHASE 2: USER DESIGN**

Once the project is scoped out, it's time to jump right into development, building out the user design through various prototype iterations.

This is the meat and potatoes of the RAD methodology—and what sets it apart from other project management strategies. During this phase, clients work hand in hand with web application programmers to ensure their needs are being met at every step in the design process. It's almost like customizable software development where the users can test each prototype of the product, at each stage, to ensure it meets their expectations.

All the bugs and kinks are worked out in an iterative process. The web application programmer designs a prototype, the client (user) tests it, and then they come together to communicate on what worked and what didn't.

This method gives web application programmers the opportunity to tweak the model as they go until they reach a satisfactory design.

Both the software web application programmers and the clients learn from the experience to make sure there is no potential for something to slip through the cracks.

### **3.19 PHASE 3: RAPID CONSTRUCTION**

Phase 3 takes the prototypes and beta systems from the design phase and converts them into the working model.

Because the majority of the problems and changes were addressed during the thorough iterative design phase, web application programmers can construct the final working model more quickly than they could by following a traditional project management approach.

The phase breaks down into several smaller steps:

Preparation for rapid construction

Program and application development

Coding

Unit, integration, and system testing

The software development team of programmers, coders, testers, and web application programmers work together during this stage to make sure everything is working smoothly and that the end result satisfies the client's expectations and objectives.

This third phase is important because the client still gets to give input throughout the process. They can suggest alterations, changes, or even new ideas that can solve problems as they arise

### **3.20 PHASE 4: CUTOVER**

This is the implementation phase where the finished product goes to launch. It includes data conversion, testing, and changeover to the new system, as well as user training.

All final changes are made while the coders and clients continue to look for bugs in the system.

Benefits of RAD methodology

RAD is one of the most successful software development programs available today, with numerous benefits for both software development teams as well as their clients.

Here are just a few advantages:

RAD lets you break the project down into smaller, more manageable tasks.

The task-oriented structure allows project managers to optimize their team's efficiency by assigning tasks according to members' specialties and experience.

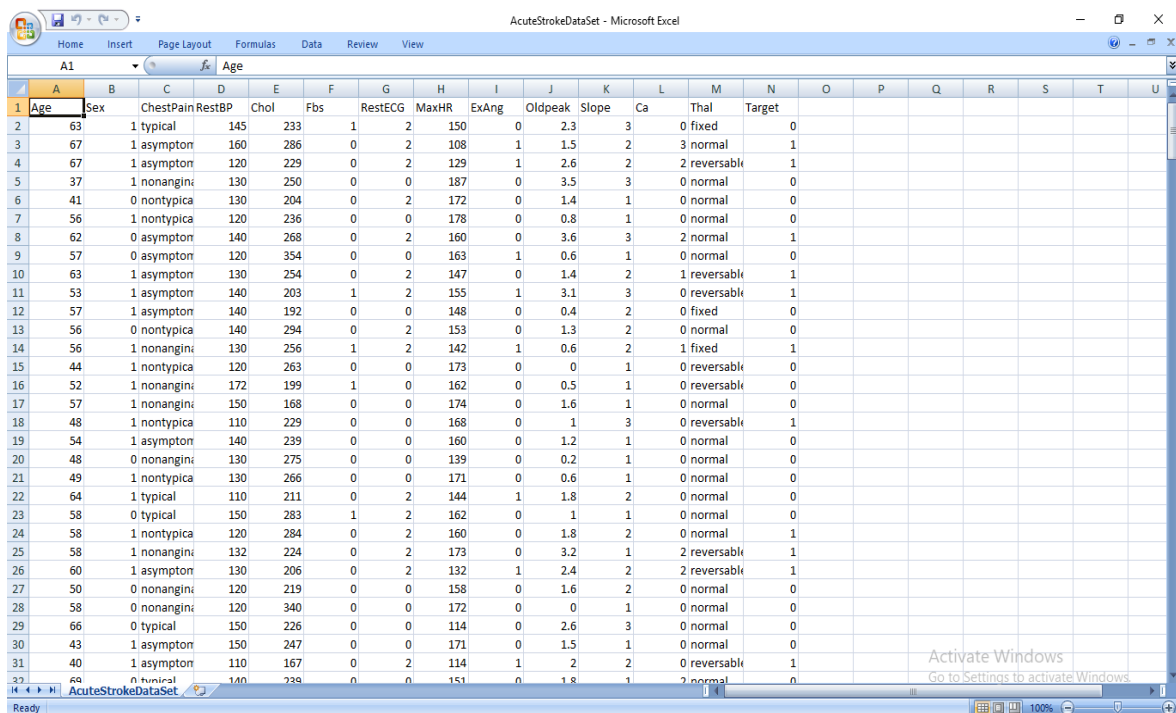
Clients get a working product delivered in a shorter time frame.

## CHAPTER 4

### RESULTS AND DISCUSSION

#### 4.1 DATABASE STORAGE SNAPSHOT

First we have to take the complete data about patient in an excel sheet. Data is nothing but having medical reports of patient like age of patient, cholesterol reports, Fbs, RestECG, MaxHR, ExAng, Oldpeak etc.,

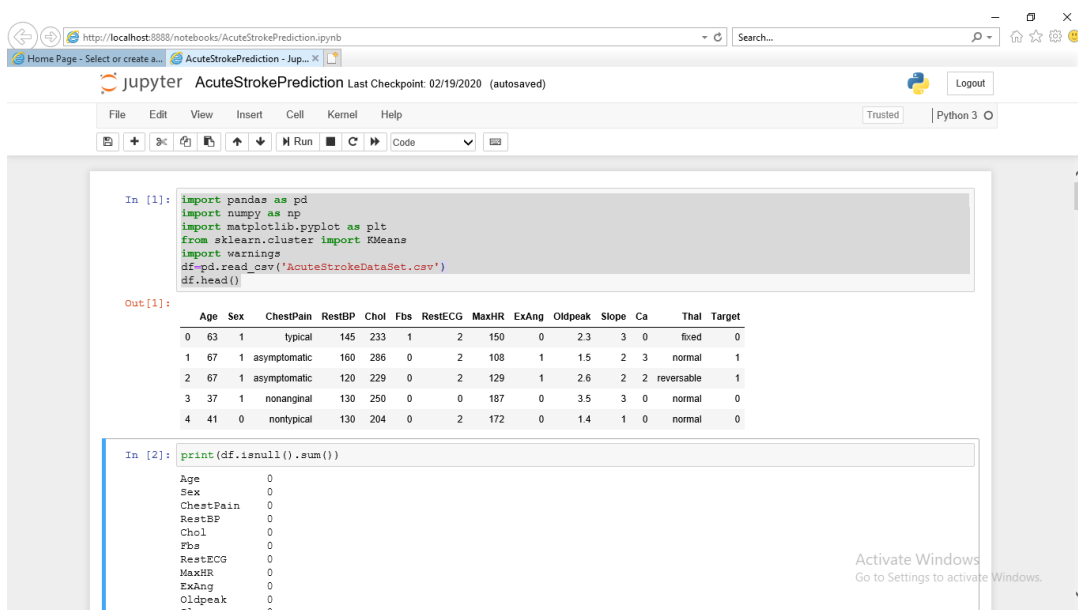


|    | A   | B   | C            | D      | E    | F   | G       | H     | I     | J       | K     | L  | M          | N      | O | P | Q | R | S | T | U |
|----|-----|-----|--------------|--------|------|-----|---------|-------|-------|---------|-------|----|------------|--------|---|---|---|---|---|---|---|
| 1  | Age | Sex | ChestPain    | RestBP | Chol | Fbs | RestECG | MaxHR | ExAng | Oldpeak | Slope | Ca | Thal       | Target |   |   |   |   |   |   |   |
| 2  | 63  | 1   | typical      | 145    | 233  | 1   | 2       | 150   | 0     | 2.3     | 3     | 0  | fixed      | 0      |   |   |   |   |   |   |   |
| 3  | 67  | 1   | asymptomatic | 160    | 286  | 0   | 2       | 108   | 1     | 1.5     | 2     | 3  | normal     | 1      |   |   |   |   |   |   |   |
| 4  | 67  | 1   | asymptomatic | 120    | 229  | 0   | 2       | 129   | 1     | 2.6     | 2     | 2  | reversible | 1      |   |   |   |   |   |   |   |
| 5  | 37  | 1   | nonanginal   | 130    | 250  | 0   | 0       | 187   | 0     | 3.5     | 3     | 0  | normal     | 0      |   |   |   |   |   |   |   |
| 6  | 41  | 0   | nontypical   | 130    | 204  | 0   | 2       | 172   | 0     | 1.4     | 1     | 0  | normal     | 0      |   |   |   |   |   |   |   |
| 7  | 56  | 1   | nontypical   | 120    | 236  | 0   | 0       | 178   | 0     | 0.8     | 1     | 0  | normal     | 0      |   |   |   |   |   |   |   |
| 8  | 62  | 0   | asymptomatic | 140    | 268  | 0   | 2       | 160   | 0     | 3.6     | 3     | 2  | normal     | 1      |   |   |   |   |   |   |   |
| 9  | 57  | 0   | asymptomatic | 120    | 354  | 0   | 0       | 163   | 1     | 0.6     | 1     | 0  | normal     | 0      |   |   |   |   |   |   |   |
| 10 | 63  | 1   | asymptomatic | 130    | 254  | 0   | 2       | 147   | 0     | 1.4     | 2     | 1  | reversible | 1      |   |   |   |   |   |   |   |
| 11 | 53  | 1   | asymptomatic | 140    | 203  | 1   | 2       | 155   | 1     | 3.1     | 3     | 0  | reversible | 1      |   |   |   |   |   |   |   |
| 12 | 57  | 1   | asymptomatic | 140    | 192  | 0   | 0       | 148   | 0     | 0.4     | 2     | 0  | fixed      | 0      |   |   |   |   |   |   |   |
| 13 | 56  | 0   | nontypical   | 140    | 294  | 0   | 2       | 153   | 0     | 1.3     | 2     | 0  | normal     | 0      |   |   |   |   |   |   |   |
| 14 | 56  | 1   | nonanginal   | 130    | 256  | 1   | 2       | 142   | 1     | 0.6     | 2     | 1  | fixed      | 1      |   |   |   |   |   |   |   |
| 15 | 44  | 1   | nontypical   | 120    | 263  | 0   | 0       | 173   | 0     | 0       | 1     | 0  | reversible | 0      |   |   |   |   |   |   |   |
| 16 | 52  | 1   | nonanginal   | 172    | 199  | 1   | 0       | 162   | 0     | 0.5     | 1     | 0  | reversible | 0      |   |   |   |   |   |   |   |
| 17 | 57  | 1   | nonanginal   | 150    | 168  | 0   | 0       | 174   | 0     | 1.6     | 1     | 0  | normal     | 0      |   |   |   |   |   |   |   |
| 18 | 48  | 1   | nontypical   | 110    | 229  | 0   | 0       | 168   | 0     | 1       | 3     | 0  | reversible | 1      |   |   |   |   |   |   |   |
| 19 | 54  | 1   | asymptomatic | 140    | 239  | 0   | 0       | 160   | 0     | 1.2     | 1     | 0  | normal     | 0      |   |   |   |   |   |   |   |
| 20 | 48  | 0   | nonanginal   | 130    | 275  | 0   | 0       | 139   | 0     | 0.2     | 1     | 0  | normal     | 0      |   |   |   |   |   |   |   |
| 21 | 49  | 1   | nontypical   | 130    | 266  | 0   | 0       | 171   | 0     | 0.6     | 1     | 0  | normal     | 0      |   |   |   |   |   |   |   |
| 22 | 64  | 1   | typical      | 110    | 211  | 0   | 2       | 144   | 1     | 1.8     | 2     | 0  | normal     | 0      |   |   |   |   |   |   |   |
| 23 | 58  | 0   | typical      | 150    | 283  | 1   | 2       | 162   | 0     | 1       | 1     | 0  | normal     | 0      |   |   |   |   |   |   |   |
| 24 | 58  | 1   | nontypical   | 120    | 284  | 0   | 2       | 160   | 0     | 1.8     | 2     | 0  | normal     | 1      |   |   |   |   |   |   |   |
| 25 | 58  | 1   | nonanginal   | 132    | 224  | 0   | 2       | 173   | 0     | 3.2     | 1     | 2  | reversible | 1      |   |   |   |   |   |   |   |
| 26 | 60  | 1   | asymptomatic | 130    | 206  | 0   | 2       | 132   | 1     | 2.4     | 2     | 2  | reversible | 1      |   |   |   |   |   |   |   |
| 27 | 50  | 0   | nonanginal   | 120    | 219  | 0   | 0       | 158   | 0     | 1.6     | 2     | 0  | normal     | 0      |   |   |   |   |   |   |   |
| 28 | 58  | 0   | nonanginal   | 120    | 340  | 0   | 0       | 172   | 0     | 0       | 1     | 0  | normal     | 0      |   |   |   |   |   |   |   |
| 29 | 66  | 0   | typical      | 150    | 226  | 0   | 0       | 114   | 0     | 2.6     | 3     | 0  | normal     | 0      |   |   |   |   |   |   |   |
| 30 | 43  | 1   | asymptomatic | 150    | 247  | 0   | 0       | 171   | 0     | 1.5     | 1     | 0  | normal     | 0      |   |   |   |   |   |   |   |
| 31 | 40  | 1   | asymptomatic | 110    | 167  | 0   | 2       | 114   | 1     | 2       | 2     | 0  | reversible | 1      |   |   |   |   |   |   |   |
| 32 | 60  | 0   | nontypical   | 140    | 238  | 0   | 0       | 151   | 0     | 1.8     | 1     | 2  | normal     | 0      |   |   |   |   |   |   |   |

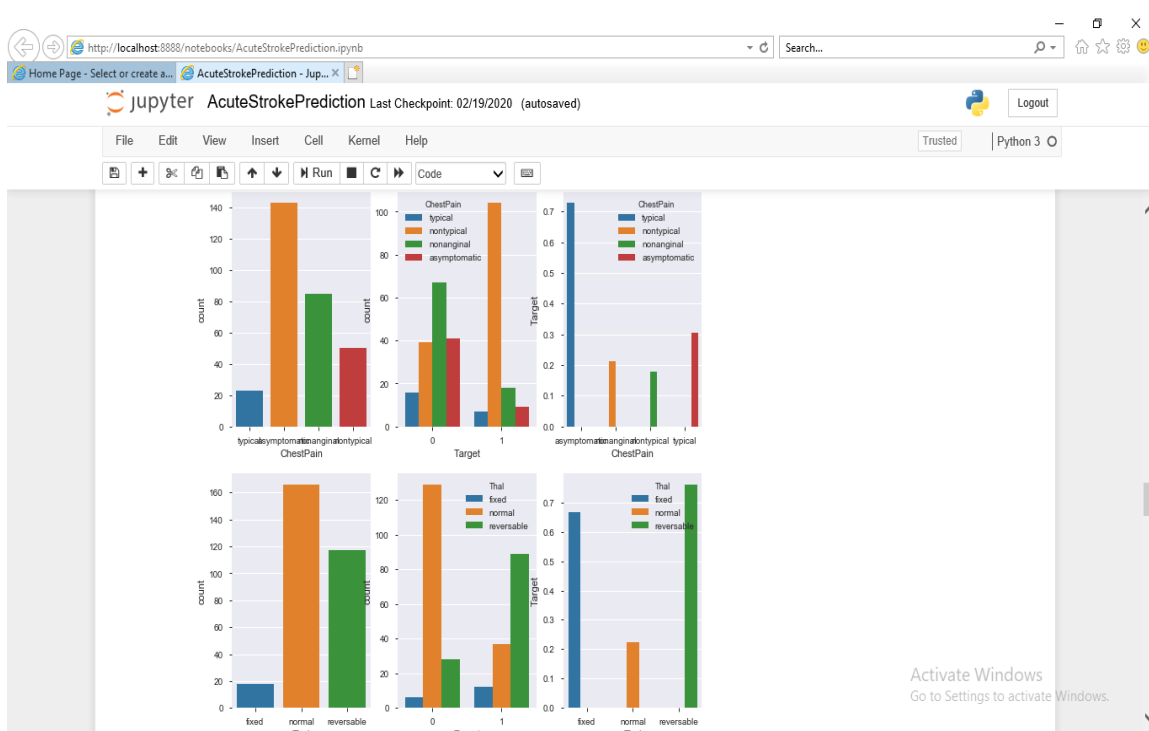
**Fig.4.1 : DATABASE STORAGE**

#### 4.2 ENROLLING DATA IN TOJUPYTER PLATFORM

The patient data which we have taken from testing, we kept in Microsoft excel sheet. Not the same data in excel sheet need to enroll in jupyter platform. From there by using this data and python code we going to get output.



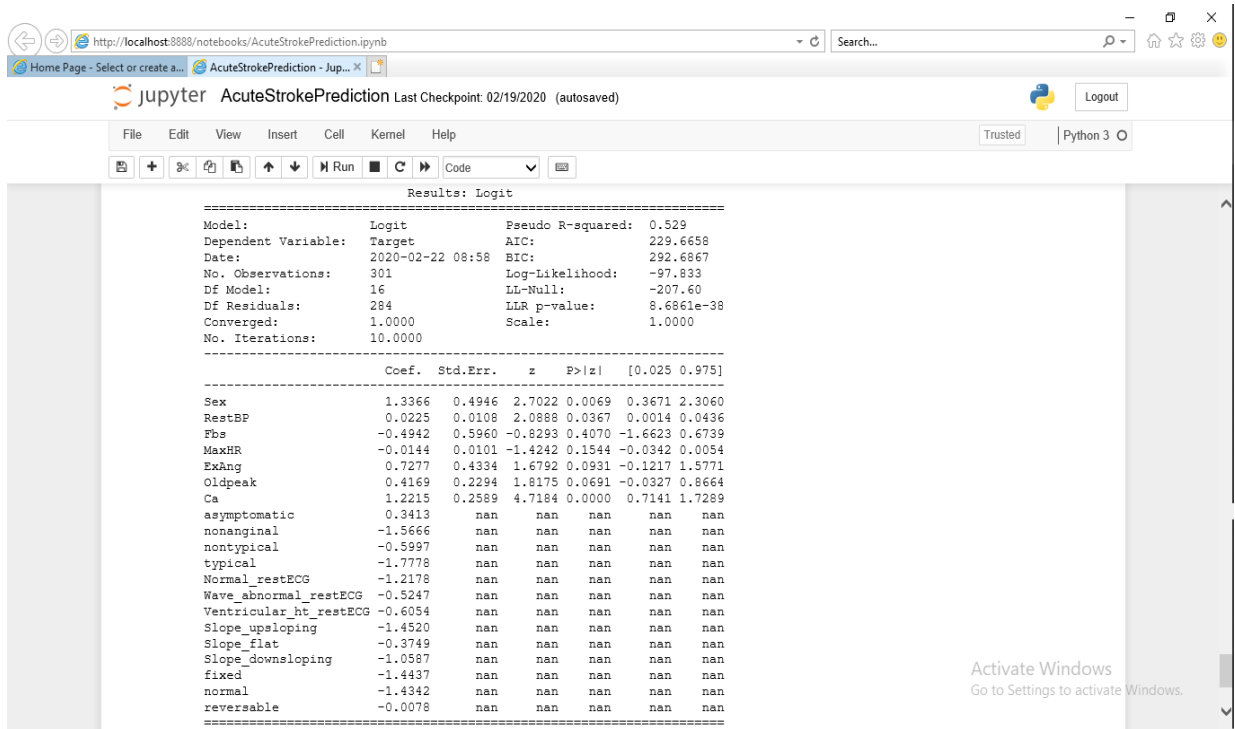
**Fig.4.2 : ENROLLING DATA INTO JUPYTER**



**Fig.4.3 : GRAPHICAL REPRESENTATION**

## 4.3 OVERVIEW OF PATIENT MEDICAL DETAILS

By running python code and which we have given an algorithm called as doopler algorithm we can get the complete details of patient.



**Fig.4.4 : FINAL OUTPUT**

## **CHAPTER 5**

### **TESTINIG**

#### **5.1 TESTING**

Testing documentation is the documentation of artifacts that are created during or before the testing of a software application. Documentation reflects the importance of processes for the customer, individual and organization. Projects which contain all documents have a high level of maturity. Careful documentation can save the time, efforts and wealth of the organization.

If the testing or development team gets software that is not working correctly and developed by someone else, so to find the error, the team will first need a document. Now, if the documents are available then the team will quickly find out the cause of the error by examining documentation. But, if the documents are not available then the tester need to do black box and white box testing again, which will waste the time and money of the organization. More than that, Lack of documentation becomes a problem for acceptance.

#### **5.2 BENEFITS OF USING DOCUMENTATION**

- Documentation clarifies the quality of methods and objectives.
- It ensures internal coordination when a customer uses software application.
- It ensures clarity about the stability of tasks and performance.
- It provides feedback on preventive tasks.
- It provides feedback for your planning cycle.
- It creates objective evidence for the performance of the quality management system.

The test scenario is a detailed document of test cases that cover end to end functionality of a software application in liner statements. The liner statement is considered as a scenario. The test scenario is a high-level classification of testable requirements. These requirements are grouped on the basis of the functionality of a module and obtained from the use cases.

In the test scenario, there is a detailed testing process due to many associated test cases. Before performing the test scenario, the tester has to consider the test cases for each scenario.

In the test scenario, testers need to put themselves in the place of the user because they test the software application under the user's point of view. Preparation of scenarios is the most critical part, and it is necessary to seek advice or help from customers, stakeholders or developers to prepare the scenario.

As per the IEEE Documentation describing plans for, or results of, the testing of a system or component, Types include test case specification, test incident report, test log, test plan, test procedure, test report. Hence the testing of all the above mentioned documents is known as documentation testing.

This is one of the most cost effective approaches to testing. If the documentation is not right: there will be major and costly problems. The documentation can be tested in a number of different ways to many different degrees of complexity. These range from running the documents through a spelling and grammar checking device, to manually reviewing the documentation to remove any ambiguity or inconsistency.

Documentation testing can start at the very beginning of the software process and hence save large amounts of money, since the earlier a defect is found the less it will cost to be fixed.

the most popular testing documentation files are test reports, plans, and checklists. These documents are used to outline the team's workload and keep track of the process. Let's take a look at the key requirements for these files and see how they contribute to the process.

### **5.2.1 Test Strategy:**

An outline of the full approach to product testing. As the project moves along, developers, designers, product owners can come back to the document and see if the actual performance corresponds to the planned activities.

### **5.2.2 Test data:**

The data that testers enter into the software to verify certain features and their outputs. Examples of such data can be fake user profiles, statistics, media content, similar to files that would be uploaded by an end-user in a ready solution.

### **5.2.3 Test plans:**

A file that describes the strategy, resources, environment, limitations, and schedule of the testing process. It's the fullest testing document, essential for informed planning. Such a document is distributed between team members and shared with all stakeholders.

### **5.2.4 Test scenarios:**

In scenarios, testers break down the product's functionality and interface by modules and provide real-time status updates at all testing stages. A module can be described by a single statement, or require hundreds of statuses, depending on its size and scope.

### **5.2.5 Test cases:**

If the test scenario describes the object of testing (what), a scenario describes a procedure (how). These files cover step-by-step guidance, detailed conditions, and current inputs of a testing task. Test cases have their own kinds that depend on the type of testing — functional, UI, physical, logical cases, etc. Test cases compare available resources and current conditions with desired outcomes and determine if the functionality can be released or not.

### **5.2.6 Traceability Matrix:**

This software testing documentation maps test cases and their requirements. All entries have their custom IDs — team members and stakeholders can track the progress of any tasks by simply entering its ID to the search.

External documentation collects information from inner documentation but also emphasize on providing a visual data representation — graphs, diagrams, etc.

External reports — these documents collect information on test results and can describe an entire project or a particular piece of functionality.

Test summary report — the file with final test results and findings, presented to stakeholders.

Bug reports — such files keep track of newly encountered bugs and their fixes. We prefer to keep our bug documentation numbered, so it's easier to mention them in further documentation. Reports are concise and focus on offering tangible solutions. Sometimes, bug reports can only include issue description, if the team hasn't yet found the best approach to fixing the problem.

The combination of internal and external documentation is the key to a deep understanding of all testing processes. Although stakeholders typically have access to the majority of documentation, they mostly work with external files, since they are more concise and tackle tangible issues and results. Internal files, on the other hand, are used by team members to optimize the testing process.

Black Box testers don't care about Unit Testing. Their main goal is to validate the application against the requirements without going into the implementation details.

But as a curiosity or Out of the box thinking, have you ever wondered how developers test their own code? What method do they use to test before releasing code for testing? How is dev-testing important in an agile process? The answer to all this is Unit Testing. I want to educate you on the importance of Unit Testing so

that development and testing teams can work more collaboratively to design, test and release an excellent application.

Unit Testing is not a new concept. It's been there since the early days of programming. Usually, developers and sometimes White box testers write Unit tests to improve code quality by verifying each and every unit of the code used to implement functional requirements (aka test driven development TDD or test-first development).

Most of us might know the classic definition of Unit Testing

“Unit Testing is the method of verifying the smallest piece of testable code against its purpose.” If the purpose or requirement failed then the unit test has failed.

In simple words, Unit Testing means – writing a piece of code (unit test) to verify the code (unit) written for implementing requirements.



**Fig. 5.1: TESTING CYCLE**

Unit Testing is used to design robust software components that help maintain code and eliminate the issues in code units. We all know the importance of finding and

fixing defects in the early stage of the software development cycle. Unit Testing serves the same purpose.

Unit Testing is an integral part of the agile software development process. When a nightly build run unit test suite should run and report should be generated. If any of the unit tests have failed then the QA team should not accept that build for verification.

If we set this as a standard process, many defects would be caught in the early development cycle, saving much testing time.

I know many developers hate to write unit tests. They either ignore or write bad unit test cases due to tight scheduled or lack of seriousness (yea they write empty unit tests, so 100% of them pass successfully ;-)). It's important to write good unit tests or don't write them at all. It's even more important to provide sufficient time and a supportive environment for real benefits.

- Testing can be done in the early phases of the software development lifecycle when other modules may not be available for integration
- Fixing an issue in Unit Testing can fix many other issues occurring in later development and testing stages
- Cost of fixing a defect found in Unit Testing is very less than the one found in the system or acceptance testing
- Code coverage can be measured
- Fewer bugs in the System and Acceptance testing
- Code completeness can be demonstrated using unit tests. This is more useful in the agile process. Testers don't get the functional builds to test until integration is completed. Code completion cannot be justified by showing that you have written and checked in the code. But running Unit tests can demonstrate code completeness.
- Expect robust design and development as developers write test cases by understanding the specifications first.



Functional Testing is a type of black box testing whereby each part of the system is tested against functional specification/requirements. For instance, seek answers to the following questions,

Are you able to login to a system after entering correct credentials?

Does your payment gateway prompt an error message when you enter incorrect card number?

Does your “add a customer” screen adds a customer to your records successfully?

Well, the above questions are mere samples to perform full-fledged functional testing of a system.

Test Driven Development, or TDD, is a code design technique where the programmer writes a test before any production code, and then writes the code that will make that test pass. The idea is that with a tiny bit of assurance from that initial test, the programmer can feel free to refactor and refactor some more to get the cleanest code they know how to write. The idea is simple, but like most simple things, the execution is hard. TDD requires a completely different mind set from what most people are used to and the tenacity to deal with a learning curve that may slow you down at first.

### **5.3 FUNCTIONAL TESTING TYPES INCLUDE:**

- Unit Testing
- Integration Testing
- System Testing
- Sanity Testing
- Smoke Testing
- Interface Testing

- Regression Testing
- Beta/Acceptance Testing
- Non-functional Testing types include

## **5.4 PERFORMANCE TESTING**

- Load Testing
- Stress Testing
- Volume Testing
- Security Testing
- Compatibility Testing
- Install Testing
- Recovery Testing
- Reliability Testing
- Usability Testing
- Compliance Testing
- Localization Testing

### **5.3.1 Unit Testing**

Unit Testing is a level of software testing where individual units/components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method,

which may belong to a base/ super class, abstract class or derived/ child class. (Some treat a module of an application as a unit. This is to be discouraged as there will probably be many individual units within that module.) Unit testing frameworks, drivers, stubs, and mock/ fake objects are used to assist in unit testing.

A unit can be almost anything you want it to be -- a line of code, a method, or a class. Generally though, smaller is better. Smaller tests give you a much more granular view of how your code is performing. There is also the practical aspect that when you test very small units, your tests can be run fast; like a thousand tests in a second fast.

Black Box testers don't care about Unit Testing. Their main goal is to validate the application against the requirements without going into the implementation details.

Unit Testing is not a new concept. It's been there since the early days of programming. Usually, developers and sometimes White box testers write Unit tests to improve code quality by verifying each and every unit of the code used to implement functional requirements (aka test driven development TDD or test-first development).

Most of us might know the classic definition of Unit Testing –

“Unit Testing is the method of verifying the smallest piece of testable code against its purpose.” If the purpose or requirement failed then the unit test has failed.

In simple words, Unit Testing means – writing a piece of code (unit test) to verify the code (unit) written for implementing requirements.

### **5.3.2 Black box Testing**

During functional testing, testers verify the app features against the user specifications. This is completely different from testing done by developers which is unit testing. It checks whether the code works as expected. Because unit testing focuses on the internal structure of the code, it is called the white box testing. On

the other hand, functional testing checks app's functionalities without looking at the internal structure of the code, hence it is called black box testing. Despite how flawless the various individual code components may be, it is essential to check that the app is functioning as expected, when all components are combined. Here you can find a detailed comparison between functional testing vs unit testing.

### **5.3.3 Integration Testing**

INTEGRATION TESTING is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.

Integration testing: Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems. See also component integration testing, system integration testing.

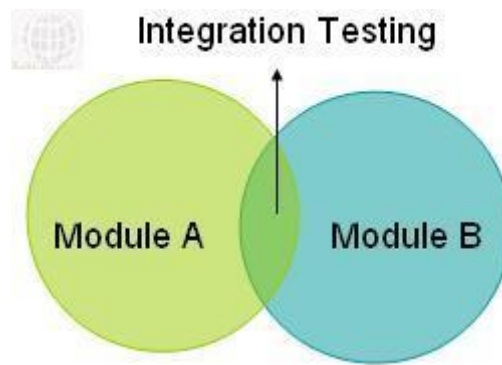
Component integration testing: Testing performed to expose defects in the interfaces and interaction between integrated components. System integration testing: Testing the integration of systems and packages; testing interfaces to external organizations (e.g. Electronic Data Interchange, Internet).

Integration tests determine if independently developed units of software work correctly when they are connected to each other. The term has become blurred even by the diffuse standards of the software industry, so I've been wary of using it in my writing. In particular, many people assume integration tests are necessarily broad in scope, while they can be more effectively done with a narrower scope.

As often with these things, it's best to start with a bit of history. When I first learned about integration testing, it was in the 1980's and the waterfall was the dominant influence of software development thinking. In a larger project, we would have a design phase that would specify the interface and behavior of the various modules in the system. Modules would then be assigned to developers to program. It was not unusual for one programmer to be responsible for a single module, but this would be big enough that it could take months to build it. All this work was done in isolation, and when the programmer believed it was finished they would hand it

over to QA for testing.

Integration testing tests integration or interfaces between components, interactions to different parts of the system such as an operating system, file system and hardware or interfaces between systems. Integration testing is a key aspect of software testing.

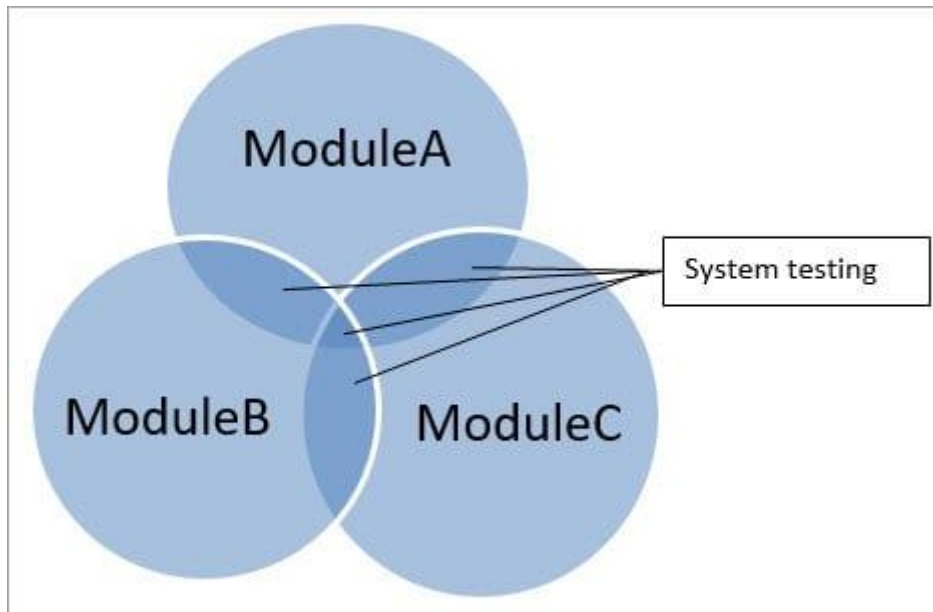


**Fig. 5.3: INTEGRATION TESTING**

#### **5.3.4 System Testing**

SYSTEM TESTING is a level of software testing where a complete and integrated software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements. System Testing means testing the system as a whole. All the modules/components are integrated in order to verify if the system works as expected or not.

System Testing is done after Integration Testing. This plays an important role in delivering a high-quality product.



**Fig. 5.4: SYSTEM TESTING**

System testing is a method of monitoring and assessing the behaviour of the complete and fully-integrated software product or system, on the basis of pre-decided specifications and functional requirements. It is a solution to the question "whether the complete system functions in accordance to its pre-defined requirements?"

It's comes under black box testing i.e. only external working features of the software are evaluated during this testing. It does not requires any internal knowledge of the coding, programming, design, etc., and is completely based on users-perspective.

A black box testing type, system testing is the first testing technique that carries out the task of testing a software product as a whole. This System testing tests the integrated system and validates whether it meets the specified requirements of the client.

System testing is a process of testing the entire system that is fully functional, in order to ensure the system is bound to all the requirements provided by the client in the form of the functional specification or system specification documentation. In most cases, it is done next to the Integration testing, as this testing should be

covering the end-to-end system's actual routine. This type of testing requires a dedicated Test Plan and other test documentation derived from the system specification document that should cover both software and hardware requirements. By this test, we uncover the errors. It ensures that all the system works as expected. We check System performance and functionality to get a quality product. System testing is nothing but testing the system as a whole. This testing checks complete end-to-end scenario as per the customer's point of view. Functional and Non-Functional tests also done by System testing. All things are done to maintain trust within the development that the system is defect-free and bug-free. System testing is also intended to test hardware/software requirements specifications. System testing is more of a limited type of testing; it seeks to detect both defects within the "inter-assemblages".

#### **5.3.5 Sanity Testing**

Sanity Testing is done when as a QA we do not have sufficient time to run all the test cases, be it Functional Testing, UI, OS or Browser Testing. Sanity testing is a subset of regression testing. After receiving the software build, sanity testing is performed to ensure that the code changes introduced are working as expected. This testing is a checkpoint to determine if testing for the build can proceed or not.

The main purpose of this testing is to determine that the changes or the proposed functionality are working as expected. If the sanity test fails, the build is rejected by the testing team to save time and money. It is performed only after the build has cleared the smoke test and been accepted by the Quality Assurance team for further testing. The focus of the team during this testing process is to validate the functionality of the application and not detailed testing. Let's assume that in an e-commerce project, there are mainly five modules- like login page, home page, user profile page, new user creation, etc. There is a defect in the login page when the password field accepts less than four alpha numeric characters and the requirement mentions that this password field should not be below eight characters. So, the bug is reported by the testing team to the development team to resolve it. When the development team fixes the defect and passes it to the testing team. Then the testing team checks if the changes done are working fine. It also

checks if it does not have an impact on other related functionalities. Now there is a functionality to update the password in the user profile page. As part of the sanity check, we would need to validate the login page as well as the profile page to ensure that the checks are working fine at both the places. Sanity tests are generally performed on build where the production deployment is required immediately - like a critical bug fix.

Smoke Testing is done to make sure if the build we received from the development team is testable or not. It is also called as “Day 0” check. It is done at the “build level”.

It helps not to waste the testing time to simply testing the whole application when the key features don't work or the key bugs have not been fixed yet. Here our focus will be on primary and core application work flow.

To conduct smoke testing, we don't write test cases. We just pick the necessary test cases from already written test cases.

Do we really write test cases for all testing types? Here in this article, we have given clear idea on choosing testing types to write test cases.

As mentioned earlier, here in Smoke Testing, our main focus will be on core application work flow. So we pick the test cases from our test suite which cover major functionality of the application. In general, we pick minimal number of test cases that won't take more than half an hour to execute.

After receiving a Software build with the minor issues fixes in code or functionality, Sanity testing is carry out to check whether the bugs reported in previous build are fixed & there is regression introduced due to these fixes i.e. not breaking any previously working functionality. The main aim of Sanity testing to check the planned functionality is working as expected. Instead of doing whole regression testing the Sanity testing is perform.

Sanity tests helps to avoid wasting time and cost involved in testing if the build is failed. Tester should reject the build upon build failure.

After completion of regression testing the Sanity testing is started to check the defect fixes & changes done in the software application is not breaking the core functionality of the software. Typically this is done nearing end of SDLC i.e. while releasing the software. You can say that sanity testing is a subset of acceptance testing. We can also say Tester Acceptance Testing for Sanity testing.

### **5.3.6 Regression Testing**

Regression Testing is a type of testing that is done to verify that a code change in the software does not impact the existing functionality of the product. This is to make sure the product works fine with new functionality, bug fixes or any change in the existing feature. Previously executed test cases are re-executed in order to verify the impact of change.

Regression Testing is a Software Testing type in which test cases are re-executed in order to check whether the previous functionality of the application is working fine and the new changes have not introduced any new bugs.

This test can be performed on a new build when there is a significant change in the original functionality that too even in a single bug fix.

For regression testing to be effective, it needs to be seen as one part of a comprehensive testing methodology that is cost-effective and efficient while still incorporating enough variety—such as well-designed frontend UI automated tests alongside targeted unit testing, based on smart risk prioritization—to prevent any aspects of your software applications from going unchecked. These days, many Agile work environments employing workflow practices such as XP (Extreme Programming), RUP (Rational Unified Process), or Scrum appreciate regression testing as an essential aspect of a dynamic, iterative development and deployment schedule.

But no matter what software development and quality-assurance process your organization uses, if you take the time to put in enough careful planning up front, crafting a clear and diverse testing strategy with automated regression testing at its core, you can help prevent projects from going over budget, keep your team on

track, and, most importantly, prevent unexpected bugs from damaging your products and your company's bottom line.

Performance testing is the practice of evaluating how a system performs in terms of responsiveness and stability under a particular workload. Performance tests are typically executed to examine speed, robustness, reliability, and application size. The process incorporates "performance" indicators such as:

- Browser, page, and network response times
- Server request processing times
- Acceptable concurrent user volumes
- Processor memory consumption; number and type of errors that might be encountered with app.

Performance testing gathers all the tests that verify an application's speed, robustness, reliability, and correct sizing. It examines several indicators such as a browser, page and network response times, server query processing time, number of acceptable concurrent users architected, CPU memory consumption, and number/type of errors which may be encountered when using an application.

Performance testing is the testing that is performed to ascertain how the components of a system are performing under a certain given situation.

Resource usage, scalability, and reliability of the product are also validated under this testing. This testing is the subset of performance engineering, which is focused on addressing performance issues in the design and architecture of a software product.



**Fig. 5.5: PERFORMANCE TESTING**

Software Performance testing is type of testing perform to determine the performance of system to major the measure, validate or verify quality attributes of the system like responsiveness, Speed, Scalability, Stability under variety of load conditions. The system is tested under a mixture of load conditions and check the time required responding by the system under varying workloads. Software performance testing involves the testing of application under test to ensure that application is working as expected under variety of load conditions. The goal of performance testing is not only find the bugs in the system but also eliminate the performance bottlenecks from the system.

Load Testing is type of performance testing to check system with constantly increasing the load on the system until the time load is reaches to its threshold value. Here Increasing load means increasing number of concurrent users, transactions & check the behavior of application under test. It is normally carried out underneath controlled environment in order to distinguish between two different systems. It is also called as “Endurance testing” and “Volume testing”. The main purpose of load testing is to monitor the response time and staying

power of application when system is performing well under heavy load. Load testing comes under the Non Functional Testing & it is designed to test the non-functional requirements of a software application.

Load testing is perform to make sure that what amount of load can be withstand the application under test. The successfully executed load testing is only if the specified test cases are executed without any error in allocated time.

Simple examples of load testing:

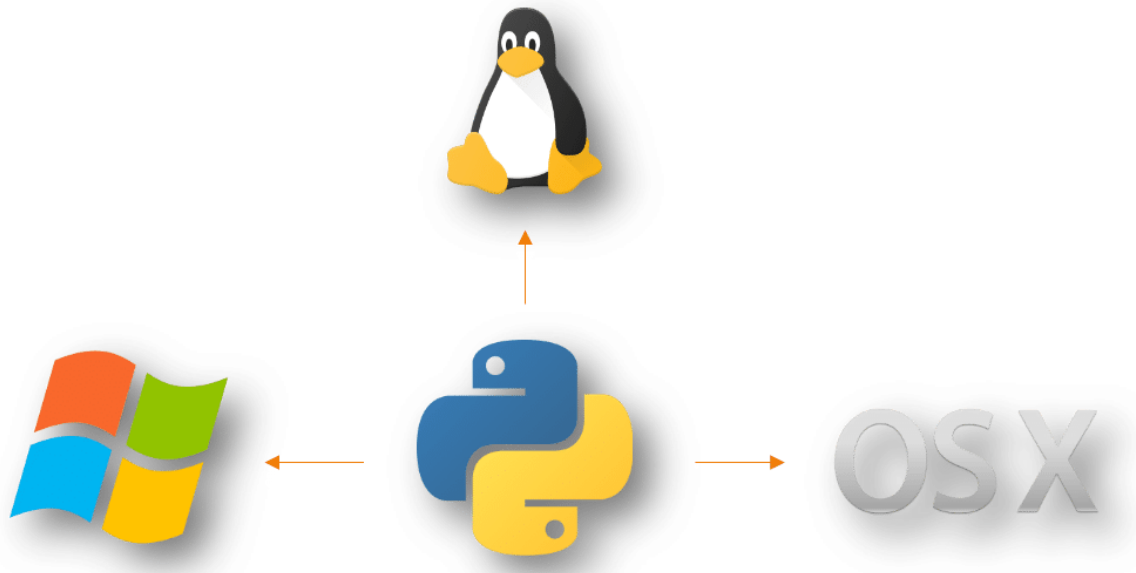
- Testing printer by sending large job.
- Editing a very large document for testing of word processor
- Continuously reading and writing data into hard disk.
- Running multiple applications simultaneously on server.
- Testing of mail server by accessing thousands of mailboxes
- In case of zero-volume testing & system fed with zero load..

## CHAPTER 6

### LANGUAGE DESCRIPTION

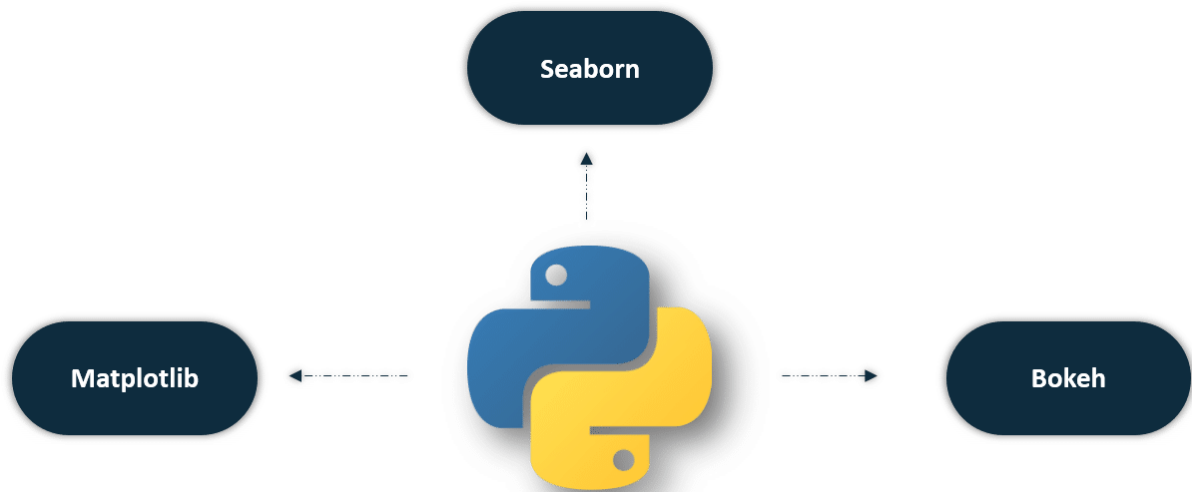
#### 6.1 ABOUT PYTHON

Python is a free, open-source programming language. Therefore, all you have to do is install Python once, and you can start working with it. Not to mention that you can contribute your own code to the community. Python is also a cross-platform compatible language. So, what does this mean? Well, you can install and run Python on several operating systems. Whether you have a Windows, Mac or Linux, you can rest assure that Python will work on all these operating systems.



**Fig. 6.1: PYTHON ARCHITECTURE-1**

Python is also a great visualization tool. It provides libraries such as Matplotlib, seaborn and bokeh to create stunning visualizations.



**Fig. 6.2: PYTHON ARCHITECTURE- 2**

In addition, Python is the most popular language for machine learning and deep learning. As a matter of fact, today, all top organizations are investing in Python to implement machine learning in the back-end.

## **6.2 PYTHON LINE STRUCTURE**

Python coding style comprises physical lines as well as logical lines or statements. A physical line in a Python program is a sequence of characters, and the end of the line terminates the line sequence as opposed to some other languages, such as C and C++ where a semi-colon is used to mark the end of the statement. A logical line, on the other hand, is composed of one or more physical lines. The use of a semi-colon is not prohibited in Python, although it's not mandatory. The NEWLINE token denotes the end of the logical line. A logical line that only contains spaces, comments, or tabs are called blank lines and they are ignored by the interpreter.

As we saw that in Python, a new line simply means that a new statement has started. Although, Python does provide a way to split a statement into a multiline statement or to join multiple statements into one logical line. This can be helpful to increase the readability of the statement. Following are the two ways to split a line into two or more lines:

### 6.3 EXPLICIT LINE JOINING

In explicit line joining, we use a backward slash to split a statement into a multiline statement.

### 6.4 IMPLICIT LINE JOINING

Statements that reside inside [], {}, or () parentheses can be broken down into two or more physical lines without using a back slash.

#### Multiple Statements on a Single Line

In Python, it is possible to club multiple statements in the same line using a semi-colon; however, most programmers do not consider this to be a good practice as it reduces the readability of the code.

### 6.5 WHITESPACES AND INDENTATION

Unlike most of the programming languages, Python uses indentation to mark a block of code. According to Python coding style guideline or PEP8, we should keep an indent size of four.

Most of the programming languages provide indentation for better code formatting and don't enforce to have it. But in Python it is mandatory. This is why indentation is so crucial in Python.

Comments in any programming language are used to increase the readability of the code. Similarly, in Python, when the program starts getting complicated, one of the best ways to maintain the readability of the code is to use Python comments. It is considered a good practice to include documentations and notes in the python syntax since it makes the code way more readable and understandable to other programmers as well, which comes in handy when multiple programmers are simultaneously working on the same project.

The code can only explain how it does something and not why it does that, but Python comments can do that. With Python comments, we can make documentations for various explanations in our code itself. Comments are nothing but tagged lines of codes which increase the readability of a code and make it self-explanatory. There are different ways of creating comments depending on the type of comment we want to include in our code. Following are different kinds of comments that can be included in our Python program:

- Single Line Comments
- Docstring Comments
- Multiline Comments

#### **6.5.1 *Single line Python comments:***

These are marked with # character. These comments end at the end of the physical line, which means that all characters starting after the # character (and lasts till the end of the line) are part of the comment.

Python has the documentation strings (or docstrings) feature which is usually the first statement included in functions and modules. Rather than being ignored by the Python Interpreter like regular comments, docstrings can actually be accessed at the run time using the dot operator.

It gives programmers an easy way of adding quick notes with every Python module, function, class, and method. To use this feature, we use triple quotes in the beginning of the documentation string or comment and the closing triple quotes at the end of the documentation comment. Docstrings can be one-liners as well as multi-liners.

Unlike some programming languages that support multiline comments, such as C, Java, and more, there is no specific feature for multiline comments in Python. But that does not mean that it is totally impossible to make multiline comments in Python. There are two ways we can include comments that can span across

multiple lines in our Python code.

### **6.5.2 Python Block Comments:**

We can use several single line comments for a whole block. This type of comment is usually created to explain the block of code that follows the Block comment. Python Block comment is the only way of writing a real comment that can span across multiple lines. It is supported and preferred by Python's PEP8 style guide since Block comments are ignored by Python interpreter or parser. However, nothing is stopping programmers from using the second 'non-real' way of writing multiline comments in Python which is explained below.

### **6.5.3 Using Docstrings**

Docstrings are largely used as multiline comments in Python by many programmers since it is the closest thing to having a multiline comment feature in Python. While it is not wrong to use docstrings when we need to make multiline comments, it is important to keep in mind that there is a significant difference between docstrings and comments. Comments in Python are totally ignored by the Python Interpreter, while docstrings, when used inside the Python function, can be accessed at the run time.

## **6.6 Python Data Types**

One of the most crucial part of learning any programming language is to understand how data is stored and manipulated in that language. Users are often inclined toward Python because of its ease of use and the number of versatile features it provides. One of those features is dynamic typing.

In Python, unlike statically typed languages like C or Java, there is no need to

specifically declare the data type of the variable. In dynamically typed languages such as Python, the interpreter itself predicts the data type of the Python Variable based on the type of value assigned to that variable.

## **6.7 ADVANTAGES OF PYTHON**

- Universal Language Construct
- Support both High Level and Low Level Programming
- Language Interoperability
- Fastest Development life cycle therefore more productive coding environment
- Less memory used because a single container hold multiple data types and each type doesn't require its own function
- Learning Ease and open source development
- Speed and user-friendly data structure
- Extensive and extensible libraries.
- Simple & support IoT
- And many more

Based on the 3 languages that are described above, we decided to use Python as programming language for developing this e-Voting web based application. The important motives are

- Easy to learn, even non experienced programmers can use it. Ex: spacing and tabbing instead of extra syntax
- Interactive mode
- Large and comprehensive standard libraries
- Python programs resemble to that of pseudo-code. This makes it a basis and a must have for beginner programmers due to its extreme ease a difficulty when compared to C++ Java, Perl, and so forth.

## 6.8 SELECTION OF INTEGRATED DEVELOPMENT ENVIRONMENT

IDE – Integrated Development Environment. It is software suiteIt contains different instruments for web application software engineers to compose and test applications.

Programming Web application software engineers utilize wide scope of devices habitually all through application source code creation, building and application testing. Programming Development apparatuses frequently incorporate word processors for source code creation, code libraries for reuse, compilers and test stages.

An Integrated Development Environment (IDE) is a product application that gives a programming domain to streamline creating and troubleshooting programming. Instead of playing out every one of the means required to make an executable program as disconnected individual assignments, it brings every one of the devices required into one application and workspace. Every one of the devices has a familiarity with the earth, and they cooperate to display a consistent improvement set for the designer.

Without an IDE, a product web application software engineer must choose, send, incorporate and deal with these instruments independently. An IDE brings huge numbers of those improvement related instruments all together system, application or administration. The incorporated toolset is intended to streamline programming advancement and can distinguish and limit coding missteps and grammatical errors.

- Komodo
- NetBeans
- PyCharm

- Python for Visual Studio.NET
- PyDev
- PyStudio
- LiClipse (Eclipse)

## 6.9 Liclipse

LiClipse is at the core for all Python based application development activities. It's been available for many years. LiClipse is a set of plugins to enhance Eclipse. It improves the overall Eclipse experience. It is best IDE tool to develop Python Applications.

### 6.9.1 *Advantages of LiClipse:*

- Perfect Debugger
- Error checking features
- An enhanced ability to analyze the code for quality and security concerns
- Auto complete feature

Based on the various IDE that are described above, we decided to use LiClipse as IDE (Integrated Development Environment) for developing this e-Voting web based application.

Important Features That Make LiClipse our IDE of Choice

- Rich Reusable Libraries are available
- Model Driven Development
- Enterprise Python Tooling
- Powerful Debugging options

## 6.10 SELECTION OF RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS):

There are various open source and commercial database management systems are available in the market. Normally, databases are divided into three categories

- Hierarchical databases
- Network databases
- Relational databases
- Object-oriented databases
- Graph databases
- ER model databases
- Document databases

Out of which Relational database management system is widely used. RDBMS represents social database the board framework. A social model can be spoken to as a table of lines and segments.

RDBMS is a data the board framework which is arranged on an information model. Here all the data is appropriately put away as tables. It offers data to be spared in the tables.

A relational database contains tables which store information that is connected here and there. SQL is the language that permits recovery and control of table information in a social database.

SQL can be used for storing and retrieving information from one spot considered database and utilize that information at whatever point it required in other programming applications.

There are approximately 121 relational database management systems are available in the market.

We have decided to use SQL Server for our e-Voting application.

SQL Server is a relational database technology developed by Microsoft. SQL Server can be used to develop from small scale to large enterprises with complex data requirements, data warehousing and Web-enabled databases.

SQL Server uses Transact-SQL (T-SQL) which is an extension of SQL.

Important Features That Make SQL Server our Database of Choice

- Better Performance Features
- Better Security Features
- Lower Cost Of Ownership
- Excellent Data Restoration and Recovery Mechanism

## **6.11 SELECTION OF OPERATING SYSTEM**

There are various open source and commercial Operating systems are available in the market. Computer Operating systems can be broadly classified into two types:

- Normal Operating System
- Real Time Operating System

We have decided to use Windows 10 for our e-Voting application.

Windows 10 is a is the most powerful Microsoft operating system it can be used for desktop/laptop computers as well as IoT devices.

Important Features That Make Windows 10 our Operating System of Choice

- The loading time is very fast compare to other OS
- Supports built-in mobile device executive facilities
- Stability and performance
- Have hologram technology

## **6.12 SELECTION OF WEB APPLICATION FRAMEWORK:**

Framework provides a structure for application development. This makes web application programmers' lives easier when developing consistent, accessible, and workable enterpirse wide web applications. They automate the implementation of redundant tasks or extensions for common operations, reduces the development

and testing time and allowing programmers to concentration more on application logic instead of routine works.

Python framework can be classified as two types:

#### **6.12.1 Full-Stack Framework**

The full-stack systems give total help to engineers, including vital components, for example, structure approval, structure generators, and format designs.

- Web2py
- Django
- Flask
- TurboGears
- Zope2
- CubicWeb
- Grok
- Pylon

#### **6.12.2 Non-Full Stack**

The non-full-stack systems don't give extra functionalities and highlights to the clients. Designers need to include a great deal of code and different things physically.

#### **6.12.3 Microframework**

Miniaturized scale structures are little, straightforward, and simple to utilize. They are laconic and have straightforward documentation. URL directing is frequently RESTful web services. Miniaturized scale systems use WSGI and work through HTTP request/Response. They are a decent decision for little applications, or as a feature of a bigger undertaking.

- CherryPy
- Flask
- Bottle

- Pyramid
- Bobo

#### 6.12.4 *Asynchronous Framework*

In situations where solicitation handling pace assumes a significant job or an undertaking should manage long reaction times, nonconcurrent structure ought to work.

- Aiohttp
- Quart
- Tornado
- FastAPI
- Sanic
- Vibora

We have decided to use Flask Microframework for our e-Voting application.

Flask is a lightweight popular Python WSGI web application framework. It is intended to make beginning fast and simple, with the capacity to scale up to complex applications. It started as a basic wrapper around Werkzeug and Jinja and has turned out to be one of the most well known Python web application systems.

It is viewed as more Pythonic than the Django web system on the grounds that in like manner circumstances the identical Flask web application is increasingly express.

Flask offers proposals, yet doesn't uphold any conditions or venture format. It is dependent upon the engineer to pick the apparatuses and libraries they need to utilize. There are numerous expansions given by the network that make including new usefulness simple.

Applications that use the Flask framework include Pinterest LinkedIn and the community web page for Flask itself.

## Important Features That Make Flask Microframework our microframework of Choice

- Easy to get started
- Little Boilerplate code needed
- built-in development server and fast debugger
- provides simplicity, flexibility and fine-grained control.
- Jinja2 templating

### 6.13 SELECTION BROWSER

There are various open source and commercial Operating systems are available in the market. There are five different Types of Web Browsers:

- WebKit Browser Types
- Blink Browser Types
- Gecko Browsers
- Goanna Browsers
- EdgeHTML Browsers

We have decided to use Google Chrome for our e-Voting application. Google Chrome is a free and one of the popular web browser created by Google that uses the WebKit layout engine.

## Important Features That Make Google Chrome our Browser of Choice

- Extremely fast web browser; it loads and displays pages very quickly.
- Tab independence
- Support for latest EcmaScript
- Modern Layout
- Cross Platform Browser

## 6.14 ABOUT PANDAS

Pandas is a popular Python package for data science, and with good reason: it offers powerful, expressive and flexible data structures that make data manipulation and analysis easy, among many other things. The DataFrame is one of these structures.

Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the DataFrame. DataFrames allow you to store and manipulate tabular data in rows of observations and columns of variables.

Pandas is built on top of the NumPy package, meaning a lot of the structure of NumPy is used or replicated in Pandas. Data in pandas is often used to feed statistical analysis in SciPy, plotting functions from Matplotlib, and machine learning algorithms in Scikit-learn.

Jupyter Notebooks offer a good environment for using pandas to do data exploration and modeling, but pandas can also be used in text editors just as easily.

Jupyter Notebooks give us the ability to execute code in a particular cell as opposed to running the entire file. This saves a lot of time when working with large datasets and complex transformations. Notebooks also provide an easy way to visualize pandas' DataFrames and plots. As a matter of fact, this article was created entirely in a Jupyter Notebook.

There are two types of data structures in pandas: Series and DataFrames.

1. Series: a pandas Series is a one dimensional data structure (“a one dimensional ndarray”) that can store values — and for every value it holds a unique index, too.
2. DataFrame: a pandas DataFrame is a two (or more) dimensional data structure — basically a table with rows and columns. The columns have names and the rows have indexes.

Those who are familiar with R know the data frame as a way to store data in

rectangular grids that can easily be overviewed. Each row of these grids corresponds to measurements or values of an instance, while each column is a vector containing data for a specific variable. This means that a data frame's rows do not need to contain, but can contain, the same type of values: they can be numeric, character, logical, etc.

Now, DataFrames in Python are very similar: they come with the Pandas library, and they are defined as two-dimensional labeled data structures with columns of potentially different types.

In general, you could say that the Pandas DataFrame consists of three main components: the data, the index, and the columns.

Firstly, the DataFrame can contain data that is:

- a Pandas DataFrame
- a Pandas Series: a one-dimensional labeled array capable of holding any data type with axis labels or index. An example of a Series object is one column from a DataFrame.
- a NumPy ndarray, which can be a record or structured
- a two-dimensional ndarray
- dictionaries of one-dimensional ndarray's, lists, dictionaries or Series.

Some of the key features of Python Pandas are as follows:

It provides DataFrame objects with default and customized indexing which is very fast and efficient.

- There are tools available for loading data of different file formats into in-memory data objects.
- It is easy to perform data alignment and integrated handling of missing data in Python Pandas.
- It is very simple to perform pivoting and reshaping of data sets in Pandas.

- It also provides indexing, label-based slicing, and sub-setting of large data sets.
- We can easily insert and delete columns from a data structure.
- Data aggregation and transformations can be done using group by.
- High-performance merging and joining of data can be done using Pandas.
- It also provides time series functionality.
- Inserting and deleting columns in data structures.
- Merging and joining data sets.
- Reshaping and pivoting data sets.
- Aligning data and dealing with missing data.
- Manipulating data using integrated indexing for DataFrame objects.
- Performing split-apply-combine on data sets using the group by engine.
- Manipulating high-dimensional data in a data structure with a lower dimension using hierarchical axis indexing.
- Subsetting, fancy indexing, and label-based slicing data sets that are large in size.
- Generating data range, converting frequency, date shifting, lagging, and other time-series functionality.
- Reading from files with CSV, XLSX, TXT, among other formats.
- Arranging data in an order ascending or descending.
- Filtering data around a condition.
- Analyzing time series.
- Iterating over a data set.

## 6.15 NUMPY:

Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays. If you are already familiar with MATLAB, you might find this tutorial useful to get started with Numpy.

A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension.

NumPy is, just like SciPy, Scikit-Learn, Pandas, etc. one of the packages that you just can't miss when you're learning data science, mainly because this library provides you with an array data structure that holds some benefits over Python lists, such as: being more compact, faster access in reading and writing items, being more convenient and more efficient.

NumPy is a Python library that is the core library for scientific computing in Python. It contains a collection of tools and techniques that can be used to solve on a computer mathematical models of problems in Science and Engineering. One of these tools is a high-performance multidimensional array object that is a powerful data structure for efficient computation of arrays and matrices. To work with these arrays, there's a vast amount of high-level mathematical functions operate on these matrices and arrays.

an array is basically nothing but pointers. It's a combination of a memory address, a data type, a shape, and strides:

- The data pointer indicates the memory address of the first byte in the array,
- The data type or dtype pointer describes the kind of elements that are contained within the array,
- The shape indicates the shape of the array, and

- The strides are the number of bytes that should be skipped in memory to go to the next element. If your strides are (10,1), you need to proceed one byte to get to the next column and 10 bytes to locate the next row.

Or, in other words, an array contains information about the raw data, how to locate an element and how to interpret an element.

With NumPy, we work with multidimensional arrays. We'll dive into all of the possible types of multidimensional arrays later on, but for now, we'll focus on 2-dimensional arrays. A 2-dimensional array is also known as a matrix, and is something you should be familiar with. In fact, it's just a different way of thinking about a list of lists. A matrix has rows and columns. By specifying a row number and a column number, we're able to extract an element from a matrix.

We can create a NumPy array using the `numpy.array` function. If we pass in a list of lists, it will automatically create a NumPy array with the same number of rows and columns. Because we want all of the elements in the array to be float elements for easy computation, we'll leave off the header row, which contains strings. One of the limitations of NumPy is that all the elements in an array have to be of the same type, so if we include the header row, all the elements in the array will be read in as strings. Because we want to be able to do computations like find the average quality of the wines, we need the elements to all be floats.

NumPy has several advantages over using core Python mathematical functions, a few of which are outlined here:

1. NumPy is extremely fast when compared to core Python thanks to its heavy use of C extensions.
2. Many advanced Python libraries, such as Scikit-Learn, Scipy, and Keras, make extensive use of the NumPy library. Therefore, if you plan to pursue a career in data science or machine learning, NumPy is a very good tool to master.

NumPy comes with a variety of built-in functionalities, which in core Python would take a fair bit of custom code.

## 6.16 MATHPLOTLIB

Plotting of data can be extensively made possible in an interactive way by Matplotlib, which is a plotting library that can be demonstrated in Python scripts. Plotting of graphs is a part of data visualization, and this property can be achieved by making use of Matplotlib.

Matplotlib makes use of many general-purpose GUI toolkits, such as wxPython, Tkinter, QT, etc., in order to provide object-oriented APIs for embedding plots into applications. John D. Hunter was the person who originally wrote Matplotlib, and its lead developer was Michael Droettboom. One of the free and open-source Python library which is basically used for technical and scientific computing is Python SciPy. Matplotlib is widely used in SciPy as most scientific calculations require plotting of graphs and diagrams.

Matplotlib is a plotting library like GNUplot. The main advantage towards GNUplot is the fact that Matplotlib is a Python module. Due to the growing interest in python the popularity of matplotlib is continually rising as well.

Another reason for the attractiveness of Matplotlib lies in the fact that it is widely considered to be a perfect alternative to MATLAB, if it is used in combination with Numpy and Scipy. Whereas MATLAB is expensive and closed source, Matplotlib is free and open source code. It is also object-oriented and can be used in an object oriented way. Furthermore it can be used with general-purpose GUI toolkits like wxPython, Qt, and GTK+. There is also a procedural "pylab", which designed to closely resemble that of MATLAB. This can make it extremely easy for MATLAB users to migrate to matplotlib.

Matplotlib can be used to create publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

Another characteristic of matplotlib is its steep learning curve, which means that users usually make rapid progress after having started. The official website has to say the following about this: "matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc, with just a few lines of code."

## 6.17 MACHINE LEARNING

Machine Learning is the most popular technique of predicting the future or classifying information to help people in making necessary decisions. Machine Learning algorithms are trained over instances or examples through which they learn from past experiences and also analyze the historical data. Therefore, as it trains over the examples, again and again, it is able to identify patterns in order to make predictions about the future.

Data is the core backbone of machine learning algorithms. With the help of the historical data, we are able to create more data by training these machine learning algorithms. For example, Generative Adversarial Networks are an advanced concept of Machine Learning that learns from the historical images through which they are capable of generating more images. This is also applied towards speech and text synthesis. Therefore, Machine Learning has opened up a vast potential for data science applications.

Machine Learning combines computer science, mathematics, and statistics. Statistics is essential for drawing inferences from the data. Mathematics is useful for developing machine learning models and finally, computer science is used for implementing algorithms.

However, simply building models is not enough. You must also optimize and tune the model appropriately so that it provides you with accurate results. Optimization techniques involve tuning the hyperparameters to reach an optimum result.

The world today is evolving and so are the needs and requirements of people. Furthermore, we are witnessing a fourth industrial revolution of data. In order to derive meaningful insights from this data and learn from the way in which people and the system interface with the data, we need computational algorithms that can churn the data and provide us with results that would benefit us in various ways. Machine Learning has revolutionized industries like medicine, healthcare, manufacturing, banking, and several other industries. Therefore, Machine Learning has become an essential part of modern industry.

Data is expanding exponentially and in order to harness the power of this data, added by the massive increase in computation power, Machine Learning has

added another dimension to the way we perceive information. Machine Learning is being utilized everywhere. The electronic devices you use, the applications that are part of your everyday life are powered by powerful machine learning algorithms.

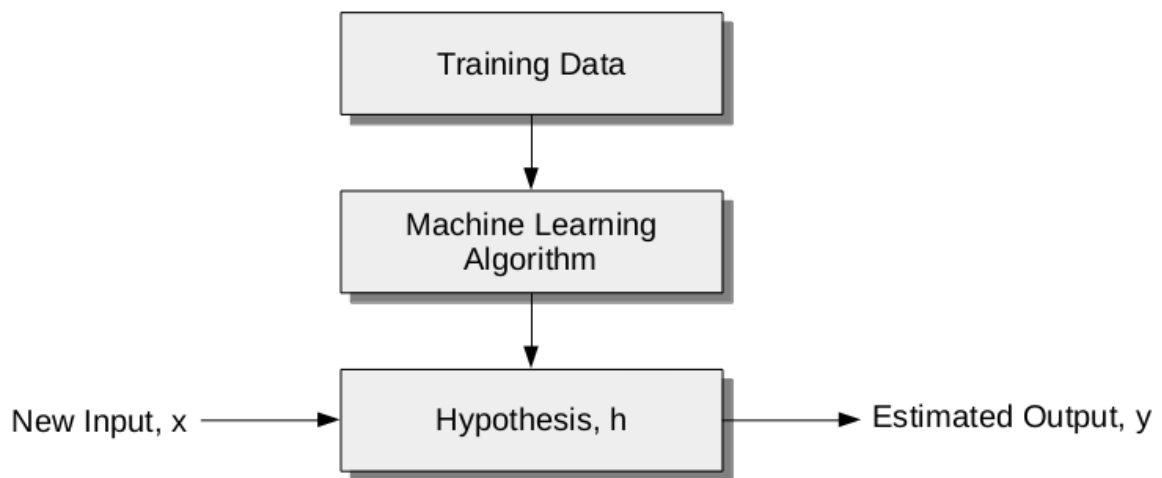
With an exponential increase in data, there is a need for having a system that can handle this massive load of data. Machine Learning models like Deep Learning allow the vast majority of data to be handled with an accurate generation of predictions. Machine Learning has revolutionized the way we perceive information and the various insights we can gain out of it.

These machine learning algorithms use the patterns contained in the training data to perform classification and future predictions. Whenever any new input is introduced to the ML model, it applies its learned patterns over the new data to make future predictions. Based on the final accuracy, one can optimize their models using various standardized approaches. In this way, Machine Learning model learns to adapt to new examples and produce better results.

#### Types of Machine Learning

Machine Learning Algorithms can be classified into 3 types as follows –

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning



**Fig. 6.3: MACHINE LEARNING ALGORITHM**

#### 6.17.1 Supervised Learning

In the majority of supervised learning applications, the ultimate goal is to develop a finely tuned predictor function  $h(x)$  (sometimes called the “hypothesis”). “Learning” consists of using sophisticated mathematical algorithms to optimize this function so that, given input data  $x$  about a certain domain (say, square footage of a house), it will accurately predict some interesting value  $h(x)$  (say, market price for said house).

$$h(x_1, x_2, x_3, x_4) = \theta_0 + \theta_1 x_1 + \theta_2 x_3^2 + \theta_3 x_3 x_4 + \theta_4 x_1^3 x_2^2 + \theta_5 x_2 x_3^4 x_4^2$$

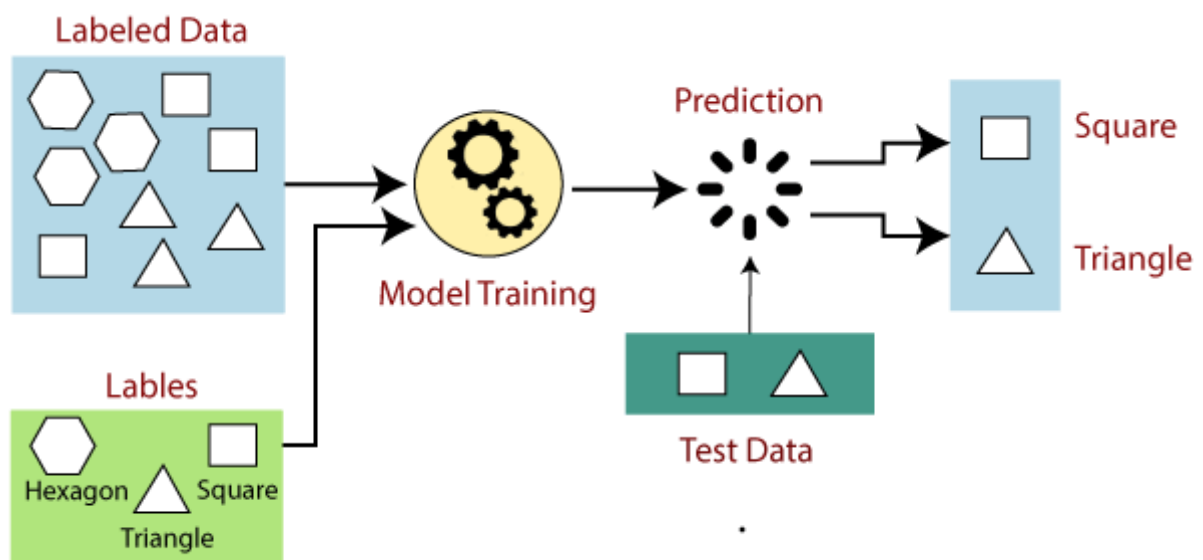
This function takes input in four dimensions and has a variety of polynomial terms. Deriving a normal equation for this function is a significant challenge. Many modern machine learning problems take thousands or even millions of dimensions of data to build predictions using hundreds of coefficients. Predicting how an organism’s genome will be expressed, or what the climate will be like in fifty years, are examples of such complex problems.

Under supervised ML, two major subcategories are:

- Regression machine learning systems: Systems where the value being predicted falls somewhere on a continuous spectrum. These systems help us with questions of “How much?” or “How many?”.

- Classification machine learning systems: Systems where we seek a yes-or-no prediction, such as “Is this tumor cancerous?”, “Does this cookie meet our quality standards?”, and so on.

In practice,  $x$  almost always represents multiple data points. So, for example, a housing price predictor might take not only square-footage ( $x_1$ ) but also number of bedrooms ( $x_2$ ), number of bathrooms ( $x_3$ ), number of floors ( $x_4$ ), year built ( $x_5$ ), zip code ( $x_6$ ), and so forth. Determining which inputs to use is an important part of ML design. However, for the sake of explanation, it is easiest to assume a single input value is used.



**FIG. 6.4: SUPERVISED LEARNING**

Steps Involved in Supervised Learning:

- First Determine the type of training dataset
- Collect/Gather the labelled training data.
- Split the training dataset into training dataset, test dataset, and validation dataset.
- Determine the input features of the training dataset, which should have enough knowledge so that the model can accurately predict the output.
- Determine the suitable algorithm for the model, such as support vector machine,

decision tree, etc.

- Execute the algorithm on the training dataset. Sometimes we need validation sets as the control parameters, which are the subset of training datasets.
- Evaluate the accuracy of the model by providing the test set. If the model predicts the correct output, which means our model is accurate.

## Regression

Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc. Below are some popular Regression algorithms which come under supervised learning:

- Linear Regression
- Regression Trees
- Non-Linear Regression
- Bayesian Linear Regression
- Polynomial Regression

## Classification

Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc.

- Spam Filtering,
- Random Forest
- Decision Trees
- Logistic Regression
- Support vector Machines

## 6.18 CONCLUSION

Head CT scans is more often used for evaluating patients with suspected TBI which represents high cost and low value endeavor. We have developed an algorithm for helping clinicians determine which patients evaluated for TBI need a head CT scan. This algorithm has superior diagnostic accuracy in comparison to

the CCHR algorithm, which is currently the most accurate and widely used algorithm for determining which TBI patient needs a head CT scan.

Using readily available clinical information, our algorithm is capable of helping clinicians avoid nearly two times the number of head CT scans that can be avoided when the CCHR algorithm is utilized.

Clinical implementation of this algorithm can help decrease financial costs associated with ED evaluations for traumatic brain injury, while decreasing patient exposure to avoidable ionizing radiation.

## **6.19 REFERENCES:**

- ▶ T. Kumagai, M. Walberer, H. Nakamura, H. Endepols, M. Sué, S. Vollmar, S. Adib, G. Mies, T. Yoshimine, M. Schroeter, and R. Graf, “Distinct spatiotemporal patterns of spreading depolarizations during early infarct evolution: evidence from real-time imaging,” *J. Cereb. Blood Flow Metab.*, vol. 31, no. 2, pp. 580-592, Feb. 2011.
- ▶ L. Shi, J. Qin, R. Reif, and R. K. Wang, “Wide velocity range Doppler optical microangiography using optimized step-scanning protocol with phase variance mask,” *J. Biomed. Opt.*, vol. 18, no. 10, pp. 106015, Oct. 2013.
- ▶ J. Xu, S. Song, Y. Li, and R. K. Wang, “Complex-based OCT angiography algorithm recovers microvascular information better than amplitude- or phase-based algorithms in phase-stable systems,” *Phys. Med. Biol.*, vol. 63, no. 1, pp. 015023, Dec. 2017.
- ▶ W. J. Choi, Z. Zhi, and R. K. Wang, “In vivo OCT microangiography of rodent iris,” *Opt. Lett.*, vol. 39, no. 8, pp. 2455-2458, Apr. 2014.

- ▶ G. Llovera, S. Roth, N. Plesnila, R. Veltkamp, and A. Liesz, “Modeling stroke in mice: permanent coagulation of the distal middle cerebral artery,” *J. Vis. Exp.*, no. 89, pp. e51729, Jul. 2014.
  
- ▶ Y. Li, U. Baran, and R. K. Wang, “Application of thinned-skull cranial window to mouse cerebral blood flow imaging using optical microangiography,” *PLoS One*, vol. 9, no. 11, pp. e113658, Nov. 2014.
  
- ▶ S. P. Chong, C. W. Merkle, D. F. Cooke, T. Zhang, H. Radhakrishnan, L. Krubitzer, and V. J. Srinivasan, “Noninvasive, in vivo imaging of subcortical mouse brain regions with 1.7  $\mu\text{m}$  OCT,” *Opt. Lett.*, vol. 40, no. 21, pp. 4911-4914, Nov. 2015.
  
- ▶ D. W. Wei, A. J. Deegan, and R. K. Wang, “Automatic motion correction for in vivo human skin OCTA through combined rigid and nonrigid registration,” *J. Biomed. Opt.*, vol. 22, no. 6, pp. 66013, Jun. 2017.
  
- ▶ Frederick P Rivara, Nathan Kuppermann, and Richard G Ellenbogen, “Use of clinical prediction rules for guiding use of computed tomography in adults with head trauma,” *JAMA*, vol. 314, no. 24, pp. 2629– 2631, 2015.
  
- ▶ Shu-Ling Chong, Su Yah Chew, Jasmine Xun Yi Feng, Penny Yun Lin Teo, Sock Teng Chin, Nan Liu, and Marcus Eng Hock Ong, “A prospective surveillance of paediatric head injuries in singapore: a dual-centre study,” *BMJ open*, vol. 6, no. 2, pp. e010618, 2016.

## APPENDIX

### A Source Code

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

import warnings

df=pd.read_csv('AcuteStrokeDataSet.csv')

df.head()

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

import warnings

df=pd.read_csv('AcuteStrokeDataSet.csv')

df.head()

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

import warnings
```

```
df=pd.read_csv('AcuteStrokeDataSet.csv')
```

```
df.head()import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
from sklearn.cluster import KMeans
```

```
import warnings
```

```
df=pd.read_csv('AcuteStrokeDataSet.csv')
```

```
df.head()import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
from sklearn.cluster import KMeans
```

```
import warnings
```

```
df=pd.read_csv('AcuteStrokeDataSet.csv')
```

```
df.head()import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
from sklearn.cluster import KMeans
```

```
import warnings
```

```
df=pd.read_csv('AcuteStrokeDataSet.csv')
```

```
df.head()import pandas as pd
```

```
import numpy as np
```

```

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

import warnings

df=pd.read_csv('AcuteStrokeDataSet.csv')

df.head()import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

import warnings

df=pd.read_csv('AcuteStrokeDataSet.csv')

df.head()import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

import warnings

```

```
df=pd.read_csv('AcuteStrokeDataSet.csv')
```

```
df.head()import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
from sklearn.cluster import KMeans
```

```
import warnings
```

```
df=pd.read_csv('AcuteStrokeDataSet.csv')
```

```
df.head()
```

```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
from sklearn.cluster import KMeans
```

```
import warnings
```

```
df=pd.read_csv('AcuteStrokeDataSet.csv')
```

```
df.head()
```

```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
from sklearn.cluster import KMeans
```

```
import warnings
```

```
df=pd.read_csv('AcuteStrokeDataSet.csv')
```

```

df.head()

print(df.isnull().sum())

df=df.dropna()

df.info()

df.describe()

import pandas_profiling

pandas_profiling.ProfileReport(df)

d=df['Target'].value_counts()

print(d)

import seaborn as sns

def plotTarget():

    sns.countplot(x='Target', data=df, ax=ax)

    for i, p in enumerate(ax.patches):

        count=df['Target'].value_counts().values[i]

        x=p.get_x()+ p.get_width() /2.

        y=p.get_height() + 3

        label='{:.1.2f}'.format(count / float(df.shape[0]))

        ax.text(x, y,label, ha='center')

fig_target,ax=plt.subplots(nrows=1, ncols=1, figsize=(5, 2))

plotTarget()

```

```
df.corr()
```

```
def plotAge():
```

```
    facet_grid = sns.FacetGrid(df, hue='Target')
```

```
    facet_grid.map(sns.kdeplot, "Age", shade=True, ax=axes[0])
```

```
    legend_labels = ['Disease false', 'Disease true']
```

```
    for t, l in zip(axes[0].get_legend().texts, legend_labels):
```

```
        t.set_text(l)
```

```
    axes[0].set(xlabel='Age', ylabel='Density')
```

```
    avg = df[["Age", "Target"]].groupby(['Age'], as_index=False).mean()
```

```
    sns.barplot(x='Age', y='Target', data=avg, ax=axes[1])
```

```
    axes[1].set(xlabel='Age', ylabel='disease probability')
```

```
plt.clf()
```

```
fig_age, axes = plt.subplots(nrows=2, ncols=1, figsize=(15, 8))
```

```
plotAge()
```

```
x=df['ChestPain']
```

```
x.value_counts()
```

```
x=df['Thal']
```

```
x.value_counts()
```

```

import seaborn as sns

category=[('ChestPain',
['typical','nontypical','nonanginal','asymptomatic']),('Thal',['fixed','normal','re
versible',])]

continuous = [('Age', 'Age in year'),

                ('Sex','1 for Male 0 for Female'),

                ('RestBP','BP in Rest State'),

                ('Fbs','Fasting blood glucose'),

                ('RestECG','ECG at rest'),

                ('Chol', 'serum cholestoral in mg/d'),

                ('MaxHR','Max Heart Rate'),

                ('ExAng','Exchange Rate'),

                ('Slope','Slope of Curve'),

                ('Oldpeak', 'ST depression by exercise relative to rest'),

                ('Ca', '# major vessels: (0-3) colored by flourosopy')]

def plotCategorical(attribute, labels, ax_index):

    sns.countplot(x=attribute, data=df, ax=axes[ax_index][0])

    sns.countplot(x='Target', hue=attribute, data=df, ax=axes[ax_index][1])

    avg = df[[attribute, 'Target']].groupby([attribute], as_index=False).mean()

    sns.barplot(x=attribute, y='Target', hue=attribute, data=avg,
ax=axes[ax_index][2])

```

```

for t, l in zip(axes[ax_index][1].get_legend().texts, labels):

    t.set_text(l)

for t, l in zip(axes[ax_index][2].get_legend().texts, labels):

    t.set_text(l)


def plotContinuous(attribute, xlabel, ax_index):

    sns.distplot(df[[attribute]], ax=axes[ax_index][0])

    axes[ax_index][0].set(xlabel=xlabel, ylabel='density')

    sns.violinplot(x='Target', y=attribute, data=df, ax=axes[ax_index][1])


def plotGrid(isCategorical):

    if isCategorical:

        [plotCategorical(x[0], x[1], i) for i, x in enumerate(category)]

    else:

        [plotContinuous(x[0], x[1], i) for i, x in enumerate(continuous)]

fig_categorical, axes = plt.subplots(nrows=len(category), ncols=3, figsize=(10,
10))

plotGrid(isCategorical=True)

fig_continuous, axes = plt.subplots(nrows=len(continuous), ncols=2,

```

```

figsize=(10,10))

plotGrid(isCategorical=False)

chestpain_dummy = pd.get_dummies(df.loc[:, 'ChestPain'])

chestpain_dummy.rename(columns={1: 'Typical', 2: 'Asymptomatic', 3:
'Nonanginal', 4: 'Nontypical'}, inplace=True)

restecg_dummy = pd.get_dummies(df.loc[:, 'RestECG'])

restecg_dummy.rename(columns={0: 'Normal_restECG', 1:
'Wave_abnormal_restECG', 2: 'Ventricular_ht_restECG'}, inplace=True)

slope_dummy = pd.get_dummies(df['Slope'])

slope_dummy.rename(columns={1: 'Slope_upsloping', 2: 'Slope_flat', 3:
'Slope_downsloping'}, inplace=True)

thal_dummy = pd.get_dummies(df['Thal'])

thal_dummy.rename(columns={3: 'Thal_Normal', 6: 'Thal_fixed', 7:
'Thal_reversible'}, inplace=True)

df = pd.concat([df, chestpain_dummy, restecg_dummy, slope_dummy,
thal_dummy], axis=1)

df.drop(['ChestPain', 'RestECG', 'Slope', 'Thal'], axis=1, inplace=True)

df.info()

```

```

df.head()

df_X= df.loc[:, df.columns != 'Target']

df_y= df.loc[:, df.columns == 'Target']

import statsmodels.api as sm

from sklearn.feature_selection import RFE

from sklearn.linear_model import LogisticRegression

selected_features=[]

lr=LogisticRegression()

rfe=RFE(lr,20)


warnings.simplefilter('ignore')

rfe.fit(df_X.values,df_y.values)

print(rfe.support_)

print(rfe.ranking_)


for i, feature in enumerate(df_X.columns.values):

    if rfe.support_[i]:

        selected_features.append(feature)


df_selected_X = df_X[selected_features]

df_selected_y=df_y

```

```
lm=sm.Logit(df_selected_y,df_selected_X)
```

```
result = lm.fit()
```

```
print(result.summary2())
```

```
warnings.simplefilter('ignore')
```