A MAJOR PROJECT REPORT

ON

"Precision Clinical Medicine Using Machine Learning: Using High and Low Quantile Ranges of Vital Signs for ICU Patient Risk Stratification"

Project report submitted
In partial fulfillment of the requirement for the award of the degree

MASTER OF COMPUTER APPLICATIONS
SEMESTER IV

Submitted By

J.BHEEMESH **(6012163055)**

Under the valuable guidance of

Prof. G. ANJAN BABU

M.Sc., MTech., Ph.D.

DEPARTMENT OF COMPUTER SCIENCE
SRI VENKATESWARA UNIVERSITY
TIRUPATI-517502
(2021-2023)

DEPARTMENT OF COMPUTER SCIENCE
S.V.U COLLEGE OF COMMERCE MANAGEMENT
AND COMPUTER SCIENCE

SRI VENKATESWARA UNIVERSITY
TIRUPATI-517502
2021-2023

CERTIFICATE

This is to certify that this project entitled "**Precision Clinical Medicine Using Machine Learning: Using High and Low Quantile Ranges of Vital Signs for ICU Patient Risk Stratification** " is a bonafied work carried out by J.BHEEMESH (6012163055) **and** R.THABASSUM KOWSAR(6012163093) of Sri Venkateswara University, Tirupati, for the award of degree of **MASTER OF COMPUTER APPLICATIONS** is a record of bonafied minor project work carried out by the candidates under my supervision and guidance. The minor project has reached the standard fulfilling the requirements of the regulation for the award of the degree of **SRI VENKATESWARA UNIVERSITY**.

Project Guide

Prof.G.ANJAN BABU

M.Sc, M.Tech, Ph.D

Dept. of Computer Science
S V University, Tirupati.

Head of the Department

Prof.G.ANJAN BABU

M.Sc, M.Tech, Ph.D

Dept. of Computer Science
S V University, Tirupati.

Submitted to the practical Examination held on…….

Examiner

External Examiner

1.
2.

DECLARATION

We hereby declare that the thesis entitled "ANTI-CRIME MANAGEMENT SYSTEM" under the valuable guidance and supervision of **Prof.G.ANJAN BABU** *Department of Computer Science, S V University, Tirupati, is submitted in partial fulfillment of the degree of Master of Computer Applications to Sri Venkateswara University and this minor project is result of our own effort and has been submitted earlier for the award of "MASTER OF COMPUTER APPLICATIONS" degree.*

Place: Tirupati
Date:

(J.BHEEMESH)

ACKNOWLEDGEMENT

"Task Successful" makes everyone happy. But the happiness will be gold without glitter if we didn't state the persons who have supported us to make it a success.

This acknowledgement transcends the reality of formality when we would like to express deep gratitude and respect to all those people behind the screen who guided, inspired and helped us for the completion of our project work.

We consider ourselves lucky enough to get such a good project. This project would add as an asset to our academic profile.

We would like to express our thankful to our project guide, **Prof .G. ANJAN BABU, Department of computer science, S.V University, Tirupati** for her constant motivation and valuable help through the project work.

We are extremely thankful to **Prof .G. ANJAN BABU, Head of the department, Department of Computer Science, S.V University, Tirupati** for all provisions made and for his constant encouragement throughout our work.

We thank all our beloved **Faculty Members, Department of Computer Science, S.V University, Tirupati.**

Finally we would like to thank our **friends** for their co-operation to complete this project.

Domain: Machine Learning
Technology: Python

Precision Clinical Medicine Using Machine Learning: Using High and Low Quantile Ranges of Vital Signs for ICU Patient Risk Stratification

Domain: Machine Learning
Technology: Python

OJECTIVE:

The main objective of this research is to Feature Evaluation of Precision Clinic Medicine Using Machine Learning Using High and Low Quantile Ranges of Vital Signs for ICU Patient Risk Stratification. Here we are going to predicting based on health condition whether the person has a chance to Survive or not.

ABSTRACT:

The abstract presents a study on the use of machine learning in precision clinical medicine for intensive care unit (ICU) patient risk stratification based on high and low quantile ranges of vital signs. The study aimed to evaluate the performance of machine learning algorithms in predicting ICU patient outcomes using different ranges of vital signs as features. The study used a dataset of ICU patient records and employed logistic regression, random forest, and gradient boosting algorithms to predict patient outcomes. The results showed that using high and low quantile ranges of vital signs as features improved the performance of the machine learning algorithms in predicting ICU patient outcomes. The study suggests that machine learning can be a valuable tool in precision clinical medicine for ICU patient risk stratification, and using quantile ranges of vital signs as features can improve the accuracy of the predictions.

Keywords: Machine Learning, Random Forest, KNN,LDA,MLP,ANN, Evaluation .

INTRODUCTION

Precision medicine is a medical approach that uses a patient's unique genetic, environmental, and lifestyle factors to tailor individualized treatments. One promising area within precision medicine is precision clinic medicine, which uses advanced technology and data analysis to monitor a patient's vital signs and detect changes that may indicate an impending health problem. Vital signs are a set of measurements that provide important information about a patient's overall health. The four main vital signs are temperature, blood pressure, heart rate, and respiratory rate. In addition,

other vital signs such as oxygen saturation, pain level, and urine output may also be monitored. By continuously monitoring a patient's vital signs, precision clinic medicine can detect subtle changes in their health that may not be noticeable otherwise. For example, a patient's blood pressure may rise slightly before they develop hypertension, or their heart rate may increase before they have a heart attack. By detecting these changes early, healthcare providers can intervene before a serious health problem develops. One technology that is commonly used in precision clinic medicine is wearable sensors. These devices can be worn on the body or implanted under the skin to continuously monitor vital signs and other health indicators. For example, a smartwatch can track a patient's heart rate and activity level, while an implanted glucose sensor can monitor blood sugar levels in patients with diabetes. Another key component of precision clinic medicine is data analysis. The large amount of data generated by wearable sensors and other monitoring devices must be analyzed to identify patterns and trends that may indicate a health problem. Machine learning algorithms can be used to analyze this data and generate alerts when significant changes are detected. Precision clinic medicine has the potential to transform healthcare by enabling earlier detection of health problems and more targeted treatments. By monitoring vital signs in real-time and analyzing the data, healthcare providers can personalize treatment plans for each patient and provide more proactive care. However, there are also challenges to implementing precision clinic medicine in practice. For example, there may be concerns about patient privacy and data security when using wearable sensors and other monitoring devices. In addition, healthcare providers will need to be trained on how to interpret the large amounts of data generated by these devices and integrate it into their clinical decision-making.In conclusion, precision clinic medicine has the potential to revolutionize healthcare by leveraging advanced technology and data analysis to monitor a patient's vital signs and detect changes that may indicate an impending health problem. While there are challenges to implementing this approach in practice, the potential benefits are significant and could lead to more personalized and effective healthcare.

LITERATURE SURVEY

Domain: Machine Learning
Technology: Python

[1.] Y.-W. Lin, Y. Zhou, F. Faghri, M. J. Shaw, and R. H. Campbell, ‘‘Analysis and prediction of unplanned intensive care unit readmission using recurrent neural networks with long short-term memory,’’ PLoS ONE, vol. 14, no. 7, Jul. 2019, Art. no. e0218942

Unplanned readmission of a hospitalized patient is a risk indication and an unnecessary waste of medical resources. Aside from hospital readmission, intensive care unit (ICU) readmission poses additional cost risk, as well as morbidity and mortality hazards. Identifying high-risk patients who are likely to be readmitted can be extremely beneficial to both patients and medical professionals. The introduction of machine learning techniques for detecting hidden patterns in large, multi-dimensional datasets opens up unprecedented prospects for establishing an effective discharge decision-making assistance system for physicians and ICU experts. We used supervised machine learning approaches for ICU readmission prediction. We used machine learning methods on comprehensive, longitudinal clinical data from the MIMIC-III to predict the ICU readmission of patients within 30 days of their discharge.

[2] O. Lo, L. Fan, W. Buchanan, and C. Thuemmler, ‘‘Technical evaluation of an e-health platform,’’ in Proc. ADIS Int. Conf. e-Health, Lisbon, Portugal, Jul. 2012, pp. 21–28.

Until recently, the use of technology in health care was mostly driven by assumptions about the benefits of electronic health (eHealth) rather than evidence. The volume of proof demonstrating eHealth's efficacy and efficiency is not proportional to the number of interventions that are frequently undertaken. Reliable evidence created via comprehensive assessment of eHealth treatments may serve to accelerate the expansion of eHealth for long-term effective adoption and increase the experience of eHealth advantages. To investigate the assessment methodologies for eHealth therapies, a comprehensive literature review was done. The standards for preferred reporting items for systematic reviews and meta-analyses (PRISMA) were followed. We looked via Google Scholar and Scopus for published publications that addressed eHealth assessment or

detailed an eHealth intervention trial. A qualitative study of the selected papers was carried out in stages.

[3] S. N. Golmaei and X. Luo, ‘‘DeepNote-GNN: Predicting hospital readmission using clinical notes and patient network,’’ in Proc. 12th ACM Conf. Bioinf., Comput. Biol., Health Informat., Aug. 2021, pp. 1–9.

Measures to predict 30-day readmission are regarded as an essential quality component for hospitals because precise forecasts can save total healthcare costs by identifying high-risk patients before they are released. While recent deep learning-based studies have shown promising empirical results on readmission prediction, several limitations exist that may limit widespread clinical utility, such as (a) only patients with specific conditions are considered, (b) existing approaches do not leverage data temporality, (c) individual admissions are assumed to be independent of each other, which is unrealistic, and (d) prior studies are typically limited to a single source of data and data from a single center. To overcome these constraints, we present a multimodal, modality-agnostic spatiotemporal graph neural network (MM-STGNN) for predicting 30-day all-cause hospital readmission that fuses multimodal in-patient longitudinal data with spatiotemporal graph neural networks. Finally, qualitative model interpretability study shows that, while patients' main diagnoses were not explicitly utilized to train the model, node attributes critical for model prediction do. Our MM-STGNN, which is agnostic to node feature modalities, might be used to incorporate multimodal data for triaging patients in different downstream resource allocation tasks.

[4] A. S. Fialho, F. Cismondi, S. M. Vieira, S. R. Reti, J. M. C. Sousa, and S. N. Finkelstein, ‘‘Data mining using clinical physiology at discharge to predict ICU readmissions,’’ Expert Syst. Appl., vol. 39, no. 18, pp. 13158–13165, Dec. 2012.

Increased mortality, morbidity, and expenditures are connected with patient readmissions to critical care units (ICUs). Current models for predicting ICU readmissions have a modest predictive value and can use up to twelve factors that can be tested at various periods throughout the inpatient stay in the ICU. We hypothesize that more predictive value may be obtained with

fewer physiological indicators, some of which can be evaluated within 24 hours after discharge. A large retrospectively gathered ICU database was subjected to a data mining technique that combined fuzzy modeling with tree search feature selection. The collection of the six selected predictive factors is not difficult in modern ICUs, and their assessment may aid in the creation of clinical treatment strategies that may reduce the likelihood of readmission. These features include genetic data as well as molecular, cellular, and imaging profiling, among others, on which individuals may be divided into subpopulations for specialized decision making in illness treatment and management. Precision medicine is one of the most intriguing and promising fields in medicine, with the greatest promise in precision preventative discovery, development, and implementation.

[5] A. E. Johnson, T. J. Pollard, L. Shen, H. L. Li-Wei, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark, ‘‘MIMIC-III, a freely accessible critical care database,’’ Sci. Data, vol. 3, May 2016, Art. no. 160035

MIMIC-III ('Medical Information Mart for Intensive Care') is a big, single-center database that contains data on patients admitted to critical care units at a large tertiary care hospital. Vital signs, medicines, laboratory measures, observations and comments recorded by care providers, fluid balance, procedure codes, diagnostic codes, imaging reports, hospital length of stay, survival statistics, and other information are included in the data. Academic and industry research, quality improvement programs, and higher education courses are all supported by the database. Current models for predicting ICU readmissions have a modest predictive value and can use up to twelve factors that can be tested at various periods throughout the inpatient stay in the ICU. We hypothesize that more predictive value may be obtained with fewer physiological indicators, some of which can be evaluated within 24 hours after discharge. A data mining strategy combining fuzzy modeling and tree search feature selection was used on a large retrospectively collected ICU database (MIMIC II), which included data from four separate ICUs at Beth Israel Deaconess Medical Center in Boston. The purpose was to predict ICU readmission between 24 and 72 hours

after release from the ICU. The area under the receiver-operating curve for readmissions was predicted using fuzzy modeling and sequential forward selection.

SYSTEM ANALYSIS & FEASIBILITY STUDY

EXISTING SYSTEM

In the existing system, implementation of machine learning algorithms is bit complex to build due to the lack of information about the data visualization. Mathematical calculations are used in existing system for model building this may takes the lot of time and complexity. To overcome all this, we use machine learning packages available in the scikit-learn library.

Disadvantages:

- High complexity.
- Time consuming.

PROPOSED SYSTEM

Proposed several machine learning models to classify whether the patient has a chance to Survive or not. Also, similar studies that have proposed models for evaluation of such performance classification mostly do not consider the heterogeneity and the size of the data Therefore, we propose a Random Forest, ANN,LDA,MLP,KNN algorithms to predict the precision medicine.

Advantages:

- Highest accuracy
- Reduces time complexity.

- Easy to use

Block Diagram:

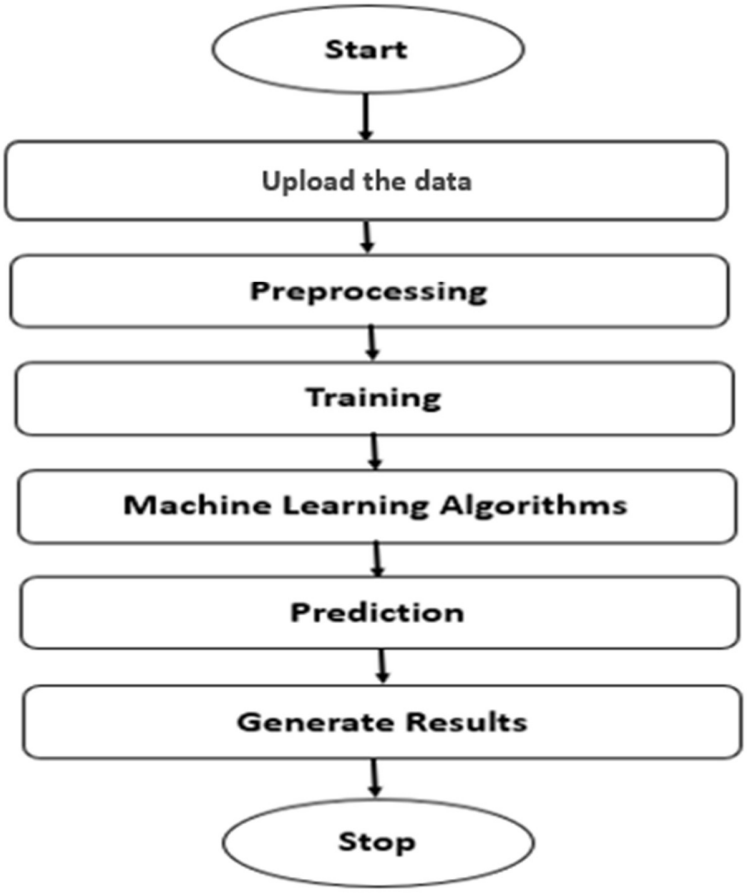
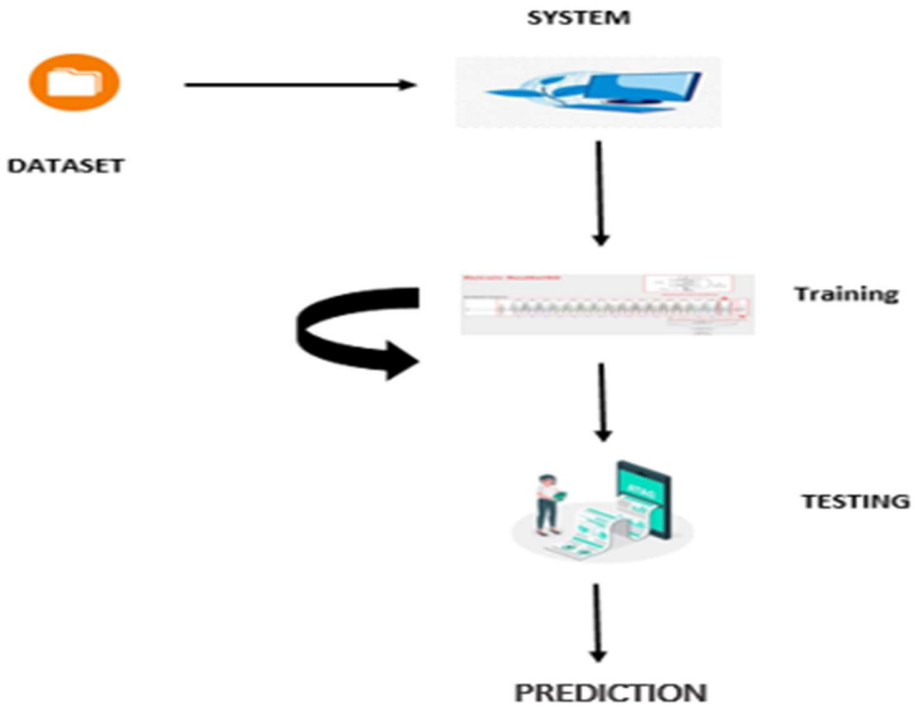


Fig: Block Diagram

Domain: Machine Learning
Technology: Python

Architecture:



METHODOLOGY AND ALGORITHMS:

1. Random Forest Classifier:

A random forest is a machine learning technique that's used to solve regression and classification problems. It utilizes ensemble learning, which is a technique that combines many classifiers to provide solutions to complex problems.

A random forest algorithm consists of many decision trees. The 'forest' generated by the random forest algorithm is trained through bagging or bootstrap aggregating. Bagging is an ensemble meta-algorithm that improves the accuracy of machine learning algorithms.

The (random forest) algorithm establishes the outcome based on the predictions of the decision trees. It predicts by taking the average or mean of the output from various trees. Increasing the number of trees increases the precision of the outcome.

A random forest eradicates the limitations of a decision tree algorithm. It reduces the over fitting of datasets and increases precision. It generates predictions without requiring many configurations in packages (like Scikit-learn).

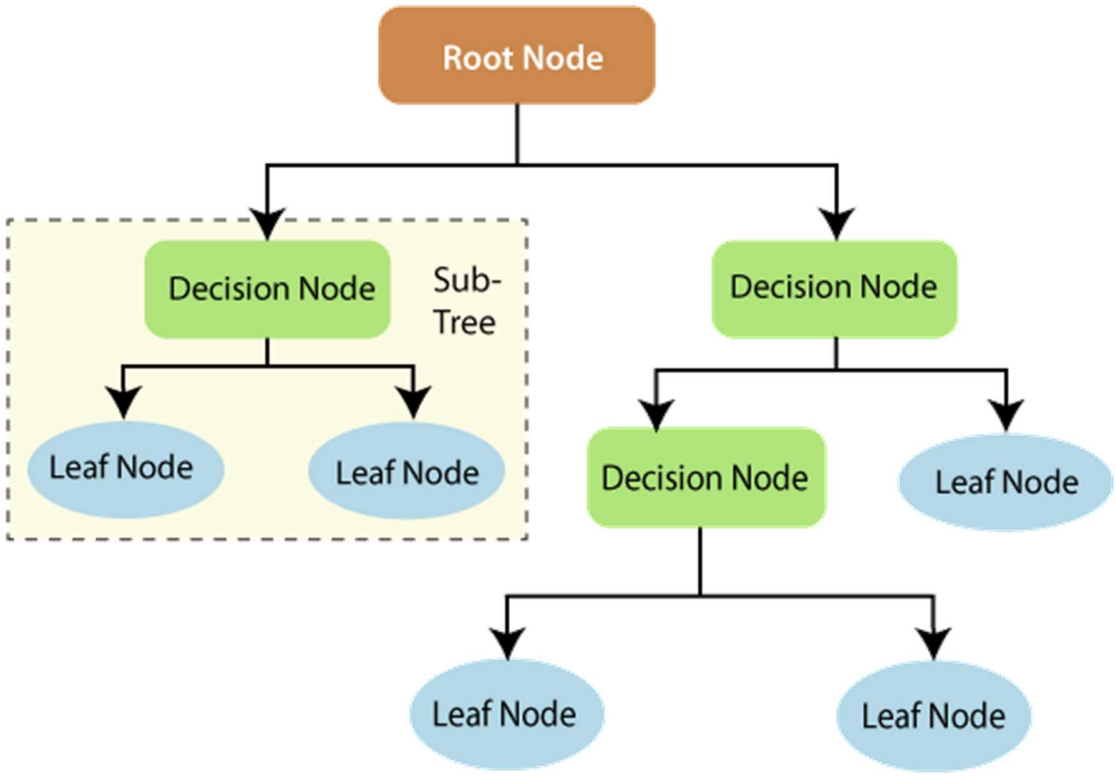
Features of a Random Forest Algorithm:

- It's more accurate than the decision tree algorithm.
- It provides an effective way of handling missing data.
- It can produce a reasonable prediction without hyper-parameter tuning.
- It solves the issue of over fitting in decision trees.
- In every random forest tree, a subset of features is selected randomly at the node's splitting point.

Decision trees are the building blocks of a random forest algorithm. A decision tree is a decision support technique that forms a tree-like structure. An overview of decision trees will help us understand how random forest algorithms work.

A decision tree consists of three components: decision nodes, leaf nodes, and a root node. A decision tree algorithm divides a training dataset into branches, which further segregate into other branches. This sequence continues until a leaf node is attained. The leaf node cannot be segregated further.

The nodes in the decision tree represent attributes that are used for predicting the outcome. Decision nodes provide a link to the leaves. The following diagram shows the three types of nodes in a decision tree.



The information theory can provide more information on how decision trees work. Entropy and information gain are the building blocks of decision trees. An overview of these fundamental concepts will improve our understanding of how decision trees are built.

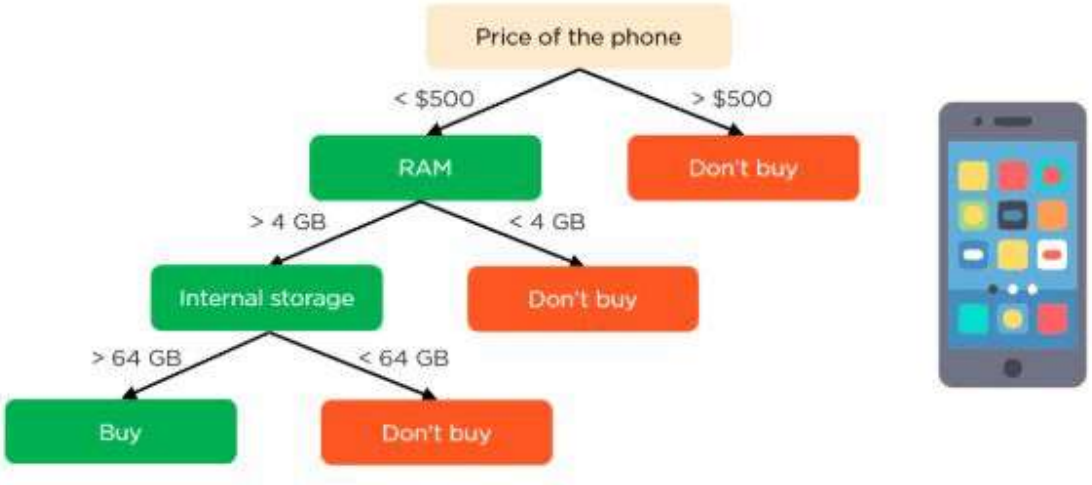
Entropy is a metric for calculating uncertainty. Information gain is a measure of how uncertainty in the target variable is reduced, given a set of independent variables.

The information gain concept involves using independent variables (features) to gain information about a target variable (class). The entropy of the target variable (Y) and the conditional entropy of Y (given X) are used to estimate the information gain. In this case, the conditional entropy is subtracted from the entropy of Y.

Information gain is used in the training of decision trees. It helps in reducing uncertainty in these trees. A high information gain means that a high degree of uncertainty (information entropy) has been removed. Entropy and information gain are important in splitting branches, which is an important activity in the construction of decision trees.

Let's take a simple example of how a decision tree works. Suppose we want to predict if a customer will purchase a mobile phone or not. The features of the phone form the basis of his decision. This analysis can be presented in a decision tree diagram.

The root node and decision nodes of the decision represent the features of the phone mentioned above. The leaf node represents the final output, either *buying* or *not buying*. The main features that determine the choice include the price, internal storage, and Random Access Memory (RAM). The decision tree will appear as follows.



Applying decision trees in random forest

The main difference between the decision tree algorithm and the random forest algorithm is that establishing root nodes and segregating nodes is done randomly in the latter. The random forest employs the bagging method to generate the required prediction.

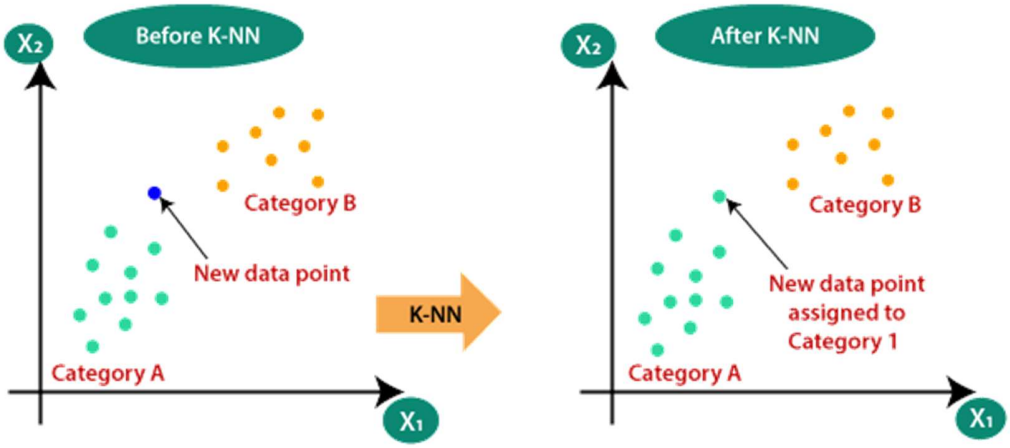
Bagging involves using different samples of data (training data) rather than just one sample. A training dataset comprises observations and features that are used for making predictions. The decision trees produce different outputs, depending on the training data fed to the random forest algorithm. These outputs will be ranked, and the highest will be selected as the final output.

Our first example can still be used to explain how random forests work. Instead of having a single decision tree, the random forest will have many decision trees. Let's assume we have only four decision trees. In this case, the training data comprising the phone's observations and features will be divided into four root nodes.

The root nodes could represent four features that could influence the customer's choice (price, internal storage, camera, and RAM). The random forest will split the nodes by selecting features randomly. The final prediction will be selected based on the outcome of the four trees.
The outcome chosen by most decision trees will be the final choice. If three trees predict *buying*, and one tree predicts *not buying*, then the final prediction will be *buying*. In this case, it's predicted that the customer will buy the phone.

KNN:

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



Advantages of KNN Algorithm:

- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

Disadvantages of KNN Algorithm:

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

LINEAR DISCRIMINANT ANALYSIS

Linear Discriminant Analysis (LDA) is one of the commonly used dimensionality reduction techniques in machine learning to solve more than two-class classification problems. It is also known as Normal Discriminant Analysis (NDA) or Discriminant Function Analysis (DFA).This can be used to project the features of higher dimensional space into lower-dimensional space in order to reduce resources and dimensional costs. In this topic, "Linear Discriminant Analysis (LDA) in machine learning", we will discuss the LDA algorithm for classification predictive modeling problems, limitation of logistic regression, representation of linear Discriminant analysis model, how to make a prediction using LDA, how to prepare data for LDA, extensions to LDA

and much more. So, let's start with a quick introduction to Linear Discriminant Analysis (LDA) in machine learning. Although the logistic regression algorithm is limited to only two-class, linear Discriminant analysis is applicable for more than two classes of classification problems.Linear Discriminant analysis is one of the most popular dimensionality reduction techniques used for supervised classification problems in machine learning. It is also considered a pre-processing step for modeling differences in ML and applications of pattern classification.

Whenever there is a requirement to separate two or more classes having multiple features efficiently, the Linear Discriminant Analysis model is considered the most common technique to solve such classification problems. For e.g., if we have two classes with multiple features and need to separate them efficiently. When we classify them using a single feature, then it may show overlapping.

Why is LDA used?

Logistic Regression is a common classification approach that works well for binary classification but struggles with multiple classification issues with well-separated classes. At the same time, LDA manages these fairly well.LDA, like PCA, may be used in data pre-processing to minimize the number of features, lowering computation costs dramatically.Face detection techniques also make use of LDA. LDA is used in Fisherfaces to extract relevant data from various faces. It offers good results when used with eigenfaces.It maximizes the distance between means of two classes.It minimizes the variance within the individual class.

MULTILAYER PERCEPTRON:

In the world of deep learning, TensorFlow, Keras, Microsoft Cognitive Toolkit (CNTK), and PyTorch are very popular. Most of us may not realise that the very popular machine learning library **Scikit-learn** is also capable of a basic deep learning modelling

- Salient points of Multilayer Perceptron (MLP) in Scikit-learn

- There is no activation function in the output layer.
- For regression scenarios, the square error is the loss function, and cross-entropy is the loss function for the classification
- It can work with single as well as multiple target values regression.
- Unlike other popular packages, likes Keras the implementation of MLP in Scikit doesn't support GPU.

We cannot fine-tune the parameters like different activation functions, weight initializers etc. for each layer.

A multilayer perceptron (MLP) is a class of feedforward artificial neural network (ANN). The term MLP is used ambiguously, sometimes loosely to mean any feedforward ANN, sometimes strictly to refer to networks composed of multiple layers of perceptrons (with threshold activation); see § Terminology. Multilayer perceptrons are sometimes colloquially referred to as "vanilla" neural networks, especially when they have a single hidden layer.

An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training.[2][3] Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable

The term "multilayer perceptron" does not refer to a single perceptron that has multiple layers. Rather, it contains many perceptrons that are organized into layers. An alternative is "multilayer perceptron network". Moreover, MLP "perceptrons" are not perceptrons in the strictest possible sense. True perceptrons are formally a special case of artificial neurons that use a threshold activation function such as the Heaviside step function. MLP perceptrons can employ arbitrary activation functions. A true perceptron performs binary classification, an MLP neuron is free to either perform classification or regression, depending upon its activation function.

The term "multilayer perceptron" later was applied without respect to nature of the nodes/layers, which can be composed of arbitrarily defined artificial neurons, and not perceptrons specifically.

This interpretation avoids the loosening of the definition of "perceptron" to mean an artificial neuron in general.

MLPs are useful in research for their ability to solve problems stochastically, which often allows approximate solutions for extremely complex problems like fitness approximation.

MLPs are universal function approximators as shown by Cybenko's theorem,[4] so they can be used to create mathematical models by regression analysis. As classification is a particular case of regression when the response variable is categorical, MLPs make good classifier algorithms.

MLPs were a popular machine learning solution in the 1980s, finding applications in diverse fields such as speech recognition, image recognition, and machine translation software,[9] but thereafter faced strong competition from much simpler (and related[10]) support vector machines. Interest in backpropagation networks returned due to the successes of deep learning.

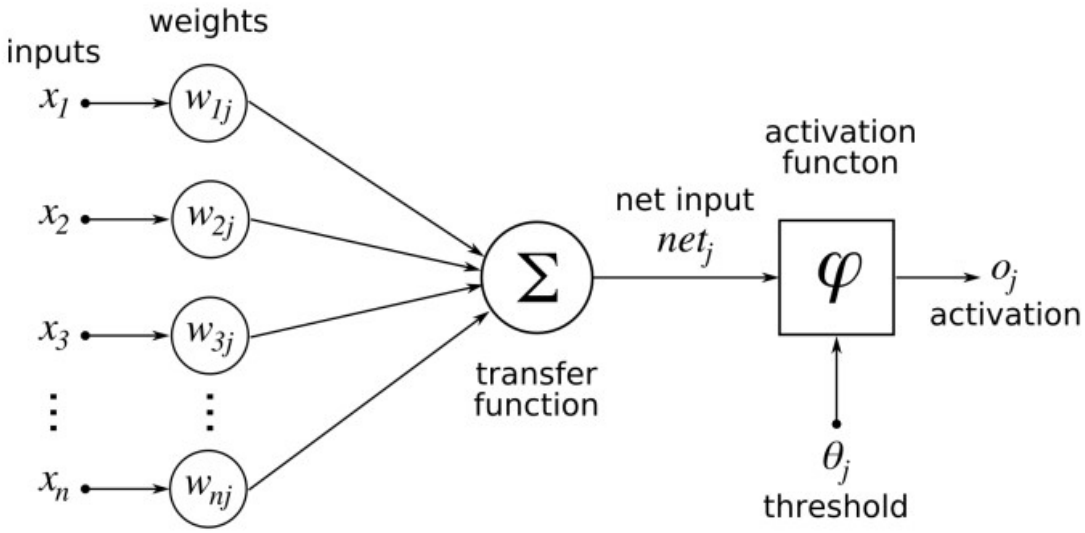
Neural Networks:

An artificial neural network (ANN) is the piece of a computing system designed to simulate the way the human brain analyzes and processes information. It is the foundation of artificial intelligence (AI) and solves problems that would prove impossible or difficult by human or statistical standards. ANNs have self-learning capabilities that enable them to produce better results as more data becomes available.

An ANN has hundreds or thousands of artificial neurons called processing units, which are interconnected by nodes. These processing units are made up of input and output units. The input units receive various forms and structures of information based on an internal weighting system, and the neural network attempts to learn about the information presented to produce one output report. Just like humans need rules and guidelines to come up with a result or output, ANNs also use a set of learning rules called backpropagation, an abbreviation for backward propagation of error, to perfect their output results.

An ANN initially goes through a training phase where it learns to recognize patterns in data, whether visually, aurally, or textually. During this supervised phase, the network compares its actual output produced with what it was meant to produce—the desired

output. The difference between both outcomes is adjusted using backpropagation. This means that the network works backward, going from the output unit to the input units to adjust the weight of its connections between the units until the difference between the actual and desired outcome produces the lowest possible error.



Whenever we increase the layers in our ANN then it is nothing but our Deep Neural Networks.

A **deep neural network** (DNN) is an artificial **neural network** (ANN) with multiple layers between the input and output layers. There are different types of **neural networks** but they always consist of the same components: neurons, synapses, weights, biases, and functions.

Domain: Machine Learning
Technology: Python

SOFTWARE DEVELOPMENT LIFE CYCLE – SDLC:

In our project we use waterfall model as our software development cycle because of its step-by-step procedure while implementing.

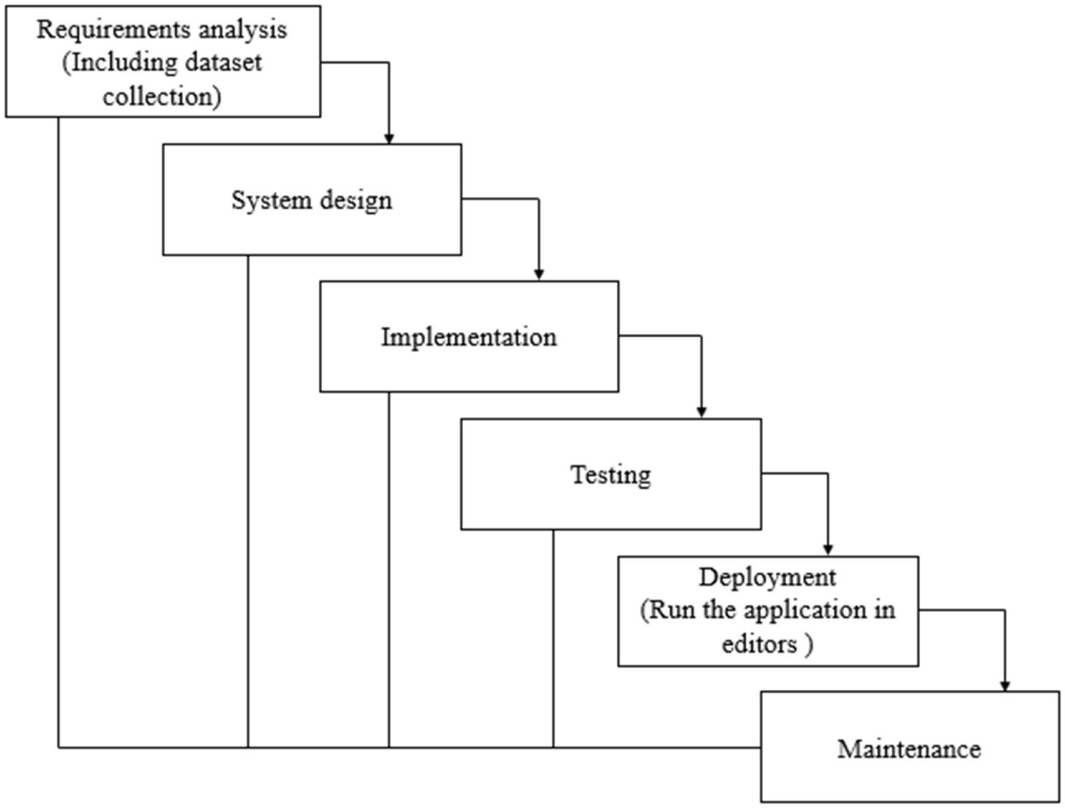


Fig1: Waterfall Model

- **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design** − the requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- **Implementation** − with inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also, to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

FEASIBILITY STUDY

The feasibility of the project is analysed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

Economic feasibility:

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

Technical feasibility:

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

Social feasibility:

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

Domain: Machine Learning
Technology: Python

SYSTEM REQUIREMENTS SPECIFICATION

Functional and non-functional requirements:

Requirement's analysis is very critical process that enables the success of a system or software project to be assessed. Requirements are generally split into two types: Functional and non-functional requirements.

Functional Requirements: These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

Examples of functional requirements:

1) Authentication of user whenever he/she logs into the system
2) System shutdown in case of a cyber-attack
3) A verification email is sent to user whenever he/she register for the first time on some software system.

Non-functional requirements: These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioral requirements.

They basically deal with issues like:

- Portability
- Security
- Maintainability
- Reliability
- Scalability
- Performance
- Reusability

- Flexibility

Examples of non-functional requirements:

1) Emails should be sent with a latency of no greater than 12 hours from such an activity.
2) The processing of each request should be done within 10 seconds
3) The site should load in 3 seconds whenever of simultaneous users are > 10000

SOFTWARE AND HARDWARE REQUIREMENTS

H/W Configuration:

Operating system	: Windows 7 or 7+
RAM	: 8 GB
Hard disc or SSD	: More than 500 GB
Processor	: Intel 3rd generation or high or Ryzen with 8 GB Ram

S/W Configuration:

Software's	: Python 3.6 or high version
IDE	: PyCharm.
Framework	: Flask, pandas, numpy and Scikit-Learn

Learning Outcome:

1. Regarding Machine Learning
2. Supervised Machine Learning
3. Classification Technique
4. About Pycharm

5. Flask Frame work

SYSTEM DESIGN:

Input Design:

In an information system, input is the raw data that is processed to produce output. During the input design, the developers must consider the input devices such as PC, MICR, OMR, etc.

Therefore, the quality of system input determines the quality of system output. Well-designed input forms and screens have following properties −

- It should serve specific purpose effectively such as storing, recording, and retrieving the information.
- It ensures proper completion with accuracy.
- It should be easy to fill and straightforward.
- It should focus on user's attention, consistency, and simplicity.
- All these objectives are obtained using the knowledge of basic design principles regarding –
    - What are the inputs needed for the system?
    - How end users respond to different elements of forms and screens.

Objectives for Input Design:

The objectives of input design are −

- To design data entry and input procedures
- To reduce input volume
- To design source documents for data capture or devise other data capture methods
- To design input data records, data entry screens, user interface screens, etc.
- To use validation checks and develop effective input controls.

Output Design:

The design of output is the most important task of any system. During output design, developers identify the type of outputs needed, and consider the necessary output controls and prototype report layouts.

Objectives of Output Design:

The objectives of input design are:

- To develop output design that serves the intended purpose and eliminates the production of unwanted output.
- To develop the output design that meets the end user's requirements.
- To deliver the appropriate quantity of output.
- To form the output in appropriate format and direct it to the right person.
- To make the output available on time for making good decisions.

MODULES:

1. **User**:

1.1 View Home page:

Here user view the home page of the web application.

View Registration Page :

Here user needs to register details.

View Login Page:

Here user needs to login details\.

1.2 View about page:

In the about page, users can learn more about the ICU platform.

1.3 View load_data page:

In the load_data page , the user will load the dataset for modelling.By loading it goes for preprocessing for splitting dataset for training and testing.

1.4 Input Model:

The user must provide input values for the certain fields in order to get results.

1.5 View Results:

User view's the generated results from the model.

1.6 View score:

Here user have ability to view the accuracy score in %

2. System

2.1 Working on dataset:

System checks for data whether it is available or not and load the data in csv files.

2.2 Pre-processing:

Data need to be pre-processed according the models it helps to increase the accuracy of the model and better information about the data.

2.3 Training the data:

After pre-processing the data will split into two parts as train and test data before training with the given algorithms.

2.4 Model Building

To create a model that predicts the personality with better accuracy, this module will help user.

2.5 Generated Score:

2.6 Here user view the score in %

2.7 Generate Results:

We train the machine learning algorithm and predict the result.

UML DIAGRAMS

UML stands for Unified Modelling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artefacts of software system, as well as for business modelling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems.

The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

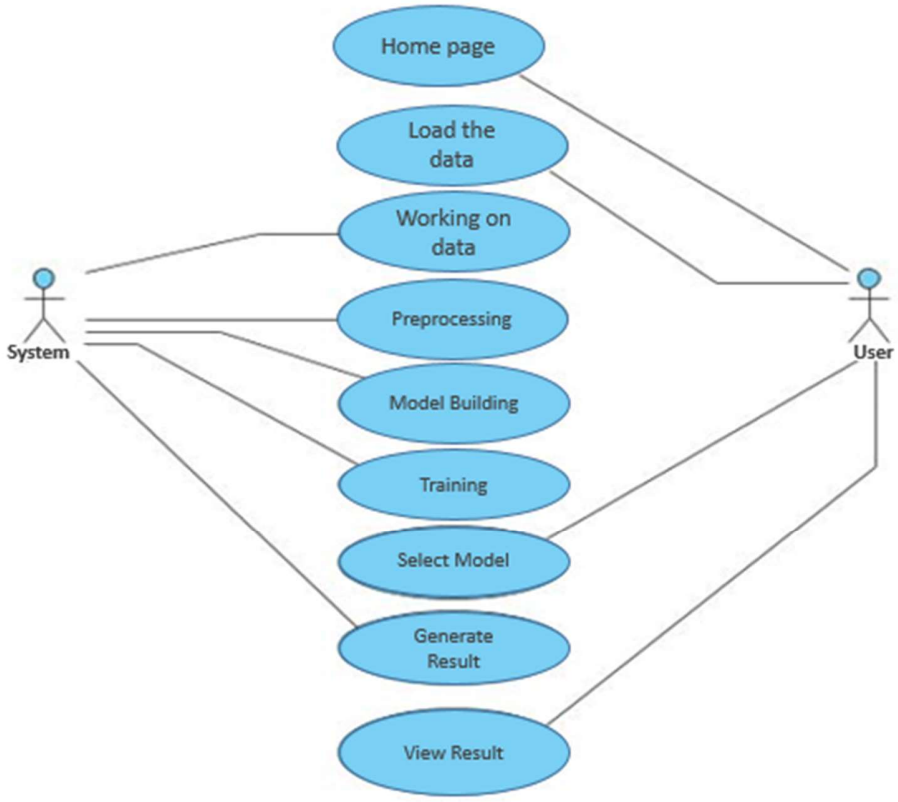
The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modelling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modelling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

USE CASE DIAGRAM

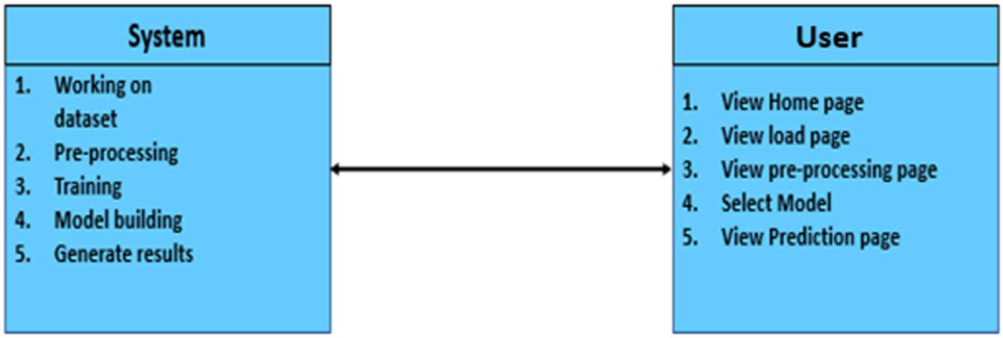
- A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis.

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



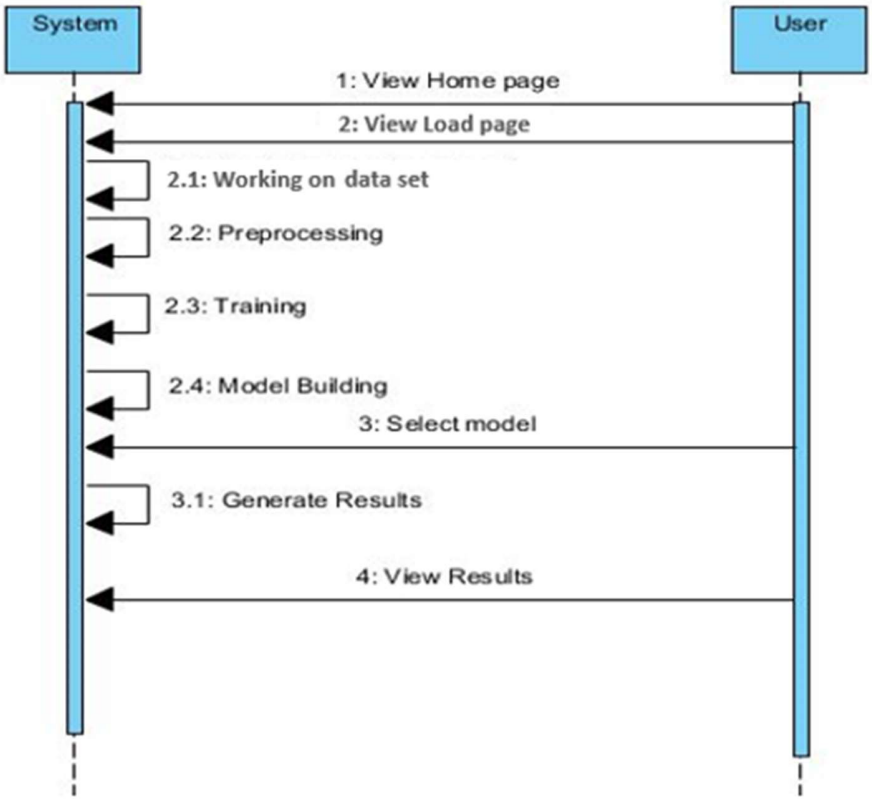
CLASS DIAGRAM

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



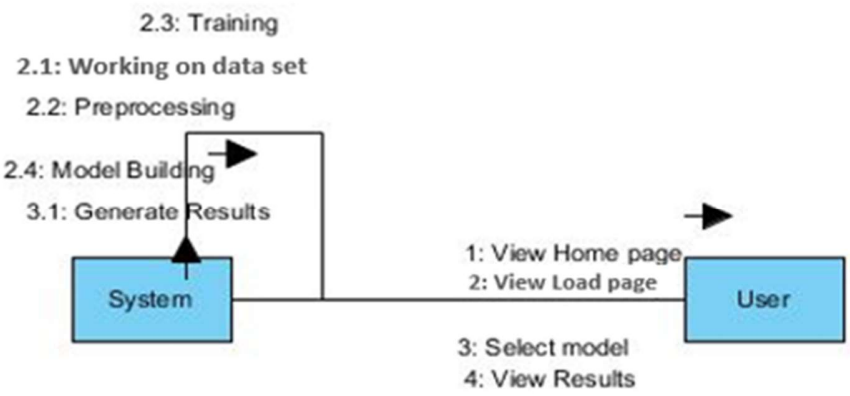
SEQUENCE DIAGRAM

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



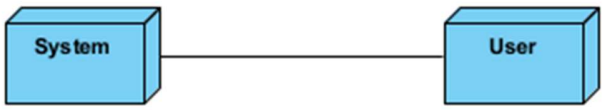
COLLABORATION DIAGRAM:

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram. The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization whereas the collaboration diagram shows the object organization.



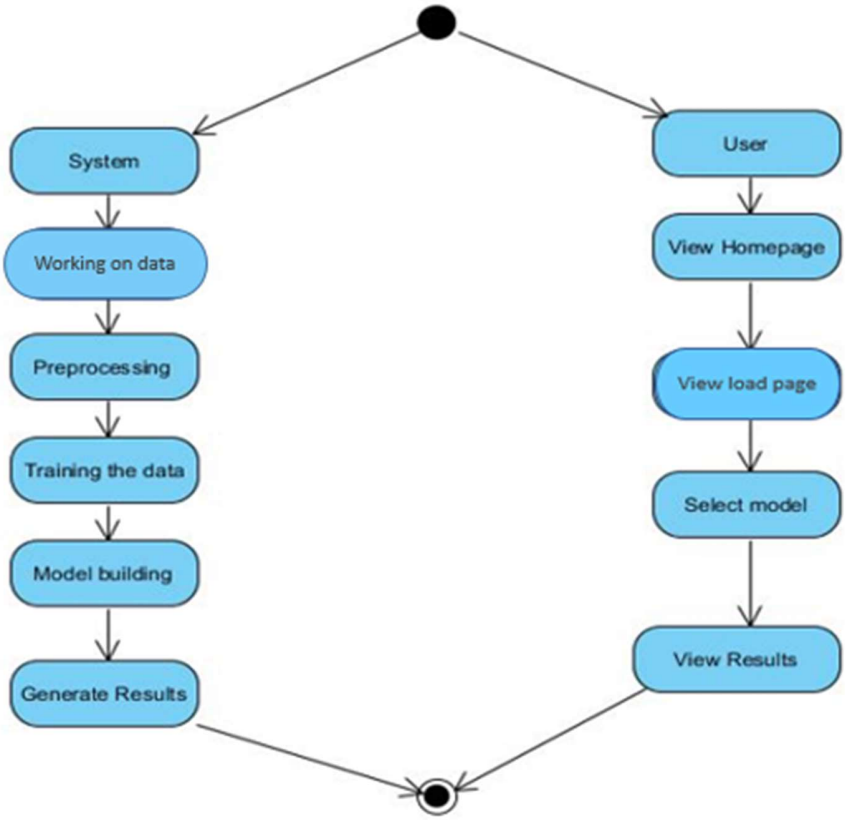
DEPLOYMENT DIAGRAM

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware's used to deploy the application.



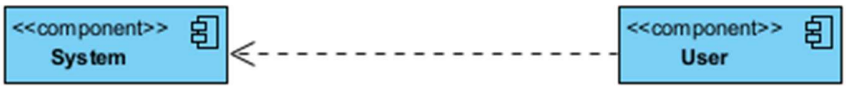
ACTIVITY DIAGRAM:

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



COMPONENT DIAGRAM:

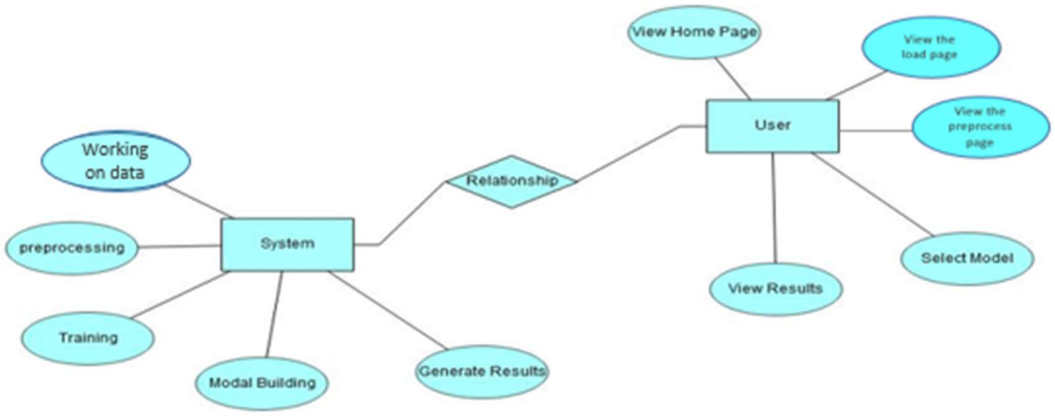
A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical components in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required function is covered by planned development.



ER DIAGRAM:

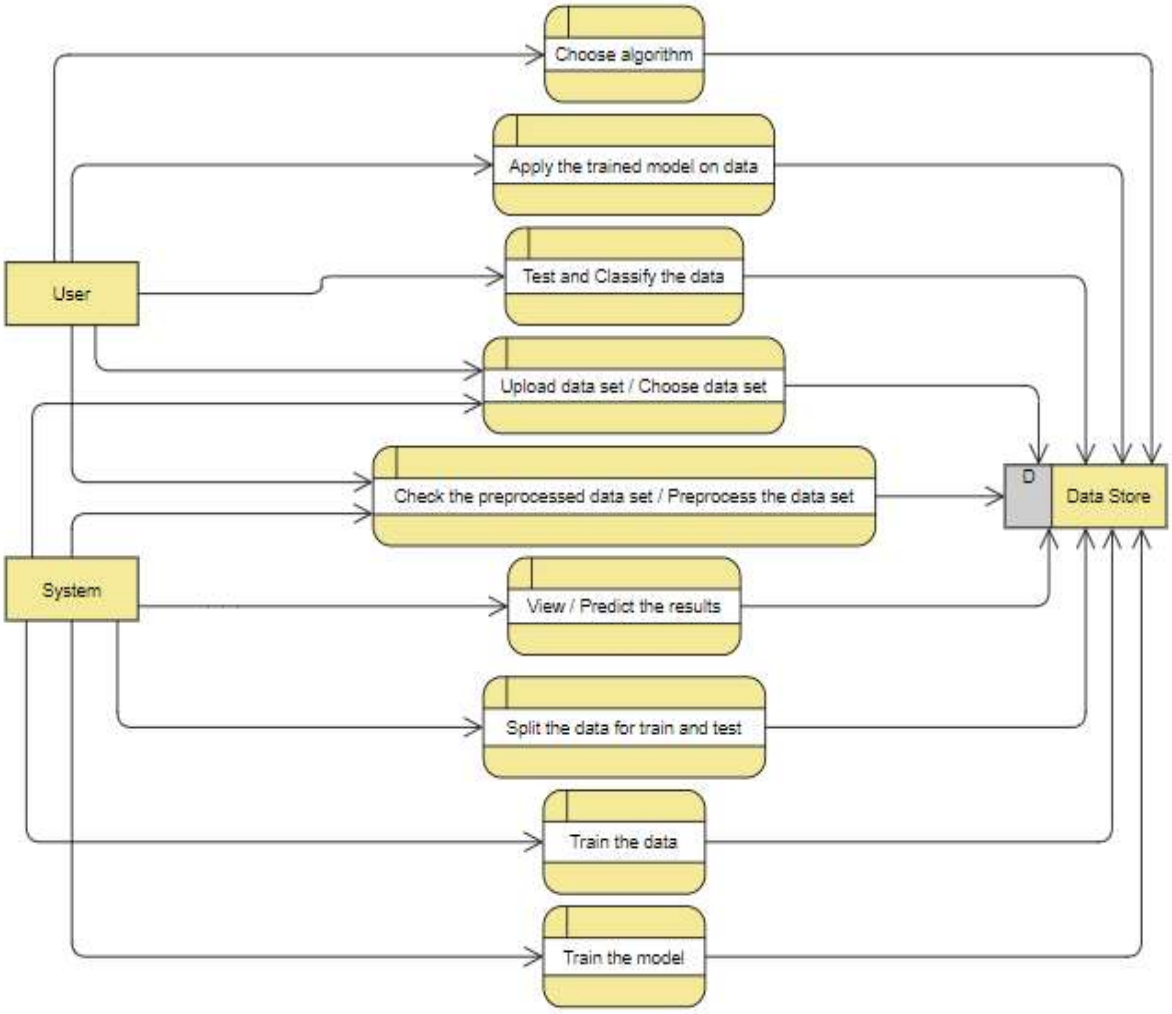
An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Let’s have a look at a simple ER diagram to understand this concept.



DFD DIAGRAM:

A Data Flow Diagram (DFD) is a traditional way to visualize the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.



CONCLUSION:

Machine learning methods applied to clinical datasets providehuge promise for tailored drug deliveryHuman illness therapy that is aimed Adding to our foundationWe propose three replicable risk indicators as part of our recent creation of an intelligent ICU patient monitoring (IICUPM) platform.ML models for stratification Our findings suggest that we canconstruct balanced predictive models for ICU patient readmissionand anomaly with more precision by combiningCombining ML and a quantiles technique that focused solely on important variablessigns. To prevent incorrect findings and low precision in theWe presented two data

options for the readmission model.Having unequal classes: one that employs under-samplingapproach, as well as one that use the clustering re-sampling method. We also provide three methods for identifying predictors ofVital sign measures taken the next day in relation to a baseline.We created regression models utilizing two distinct classifications.algorithms to each method In general, we discovered that theThe error compensation strategy outperformed the average.The measured method fared the poorest. The outcome suggeststhat employing the most current vital sign measures achieves the least error, especially when accounting forfor prior blunders In addition to offering three clearThis work adds a functionality to replicable ML models.developing an algorithm that may be used in a variety of critical care contexts.

REFERENCES:

[1] A. Shaban-Nejad, M. Michalowski, N. Peek, J. S. Brownstein, and D. L. Buckeridge, ‘‘Seven pillars of precision digital health and medicine,’’ Artif. Intell. Med., vol. 103, Mar. 2020, Art. no. 101793.

[2] G. Gutierrez, ‘‘Artificial intelligence in the intensive care unit,’’ Crit. Care, vol. 24, no. 1, p. 101, 2020, doi: 10.1186/s13054-020-2785-y.

[3] A. Shaban-Nejad, H. Mamiya, A. Riazanov, A. J. Forster, C. J. O. Baker, R. Tamblyn, and D. L. Buckeridge, ‘‘From cues to nudge: A knowledgebased framework for surveillance of healthcare-associated infections,’’ J. Med. Syst., vol. 40, no. 1, pp. 1–12, Jan. 2016. VOLUME 10, 2022 52429 K. Alghatani et al.: Precision Clinical Medicine Through Machine Learning .

[4] T. J. Pollard and L. A. Celi, ‘‘Enabling machine learning in critical care,’’ ICU Manage. Pract., vol. 17, no. 3, p. 198, 2017.

[5] D. Plant and A. Barton, ‘‘Machine learning in precision medicine: Lessons to learn,’’ Nature Rev. Rheumatol., vol. 17, no. 1, pp. 5–6, Jan. 2021.

[6] K. Alghatani, N. Ammar, A. Rezgui, and A. Shaban-Nejad, ‘‘Predicting intensive care unit length of stay and mortality using patient vital signs: Machine learning model development and validation,’’ JMIR Med. Informat., vol. 9, no. 5, May 2021, Art. no. e21347.

[7] A. Rajkomar et al., ‘‘Scalable and accurate deep learning with electronic health records,’’ npj Digit. Med., vol. 1, no. 1, p. 18, 2018.

[8] A. E. Johnson, T. J. Pollard, L. Shen, H. L. Li-Wei, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark, ‘‘MIMIC-III, a freely accessible critical care database,’’ Sci. Data, vol. 3, May 2016, Art. no. 160035.

[9] T. Pollard and A. Johnson, III. (2016). The MIMIC-III Clinical Database. [Online]. Available: https://doi.org/10.13026/c2xw26

[10] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, ‘‘PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals,’’ Circulation, vol. 101, no. 23, pp. e215–e220, 2000.

[11] Y.-W. Lin, Y. Zhou, F. Faghri, M. J. Shaw, and R. H. Campbell, ‘‘Analysis and prediction of unplanned intensive care unit readmission using recurrent neural networks with long short-term memory,’’ PLoS ONE, vol. 14, no. 7, Jul. 2019, Art. no. e0218942.

[12] A. S. Fialho, F. Cismondi, S. M. Vieira, S. R. Reti, J. M. C. Sousa, and S. N. Finkelstein, ‘‘Data mining using clinical physiology at discharge to predict ICU readmissions,’’ Expert Syst. Appl., vol. 39, no. 18, pp. 13158–13165, Dec. 2012.

[13] E. K. Shin, R. Mahajan, O. Akbilgic, and A. Shaban-Nejad, ‘‘Sociomarkers and biomarkers: Predictive modeling in identifying pediatric asthma patients at risk of hospital revisits,’’ npj Digit. Med., vol. 1, no. 1, pp. 1–5, Dec. 2018.

[14] S. N. Golmaei and X. Luo, ‘‘DeepNote-GNN: Predicting hospital readmission using clinical notes and patient network,’’ in Proc. 12th ACM Conf. Bioinf., Comput. Biol., Health Informat., Aug. 2021, pp. 1–9.

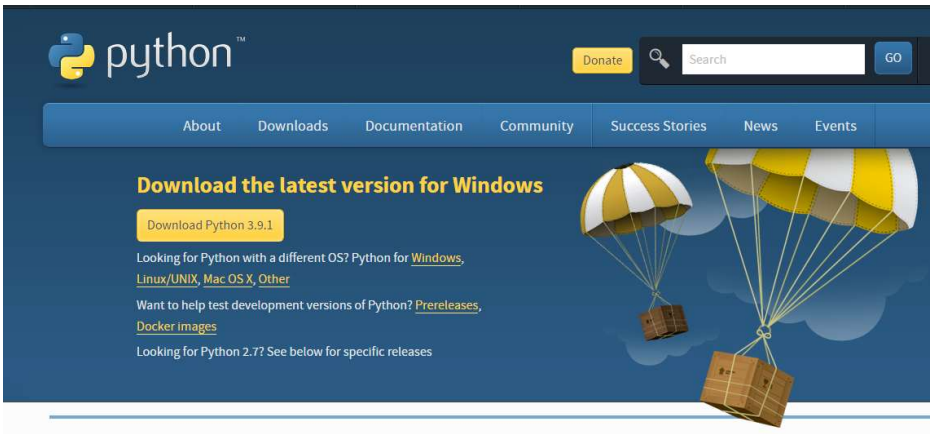
[15] A. Pakbin, P. Rafi, N. Hurley, W. Schulz, M. H. Krumholz, and J. B. Mortazavi, ‘‘Prediction of ICU readmissions using data at patient discharge,’’ in Proc. 40th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC), Jul. 2018, pp. 4932–4935.

BIBLIOGRAPHY:

SOFTWARE INSTALLATION FOR MACHINE LEARNING PROJECTS:

Installing Python:

1. To download and install Python visit the official website of Python https://www.python.org/downloads/ and choose your version.



2. Once the download is complete, run the exe for install Python. Now click on Install Now.

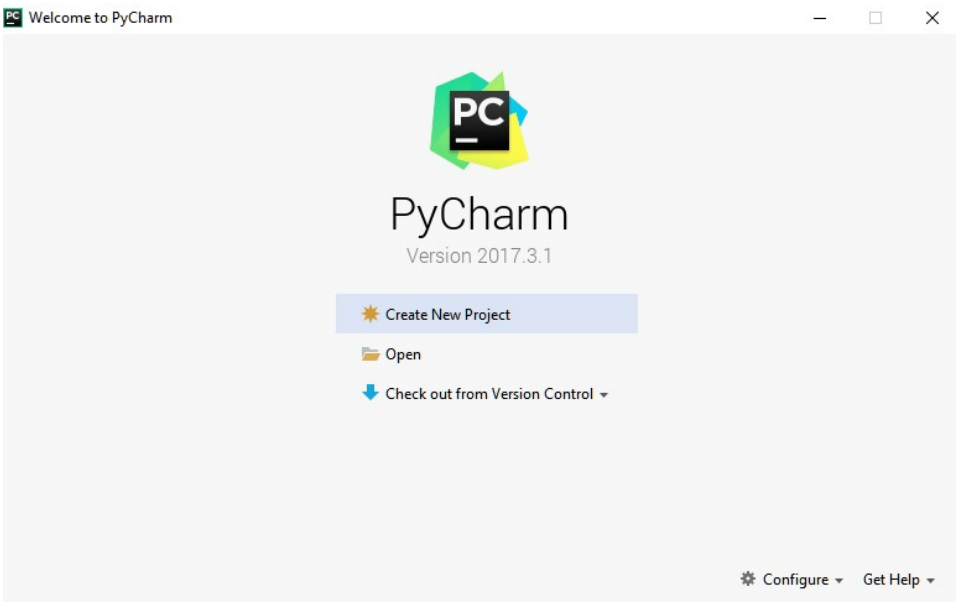
3. You can see Python installing at this point.

4. When it finishes, you can see a screen that says the Setup was successful. Now click on "Close".

Installing PyCharm:

1. To download PyCharm visit the website https://www.jetbrains.com/pycharm/download/ and click the "DOWNLOAD" link under the Community Section.

2. Once the download is complete, run the exe for install PyCharm. The setup wizard should have started. Click “Next”.
3. On the next screen, Change the installation path if required. Click “Next”.
4. On the next screen, you can create a desktop shortcut if you want and click on “Next”.
5. Choose the start menu folder. Keep selected JetBrains and click on “Install”.
6. Wait for the installation to finish.
7. Once installation finished, you should receive a message screen that PyCharm is installed. If you want to go ahead and run it, click the “Run PyCharm Community Edition” box first and click “Finish”.
8. After you click on "Finish," the Following screen will appear.

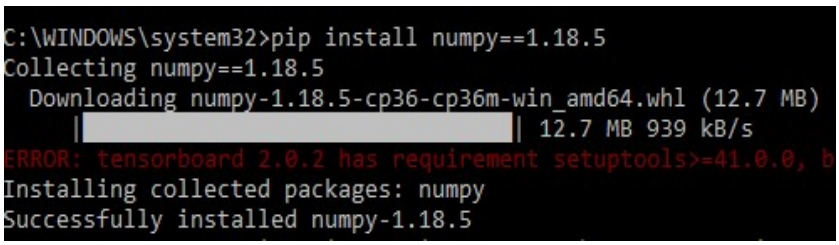


9. You need to install some packages to execute your project in a proper way.

10. Open the command prompt/ anaconda prompt or terminal as administrator.

11. The prompt will get open, with specified path, type "pip install package name" which you want to install (like NumPy, pandas, sea born, scikit-learn, Matplotlib, Pyplot)

Ex: Pip install NumPy



INTRODUCTION TO PYTHON

- ✓ Python

What Is a Script?

Up to this point, I have concentrated on the interactive programming capability of Python. This is a very useful capability that allows you to type in a program and to have it executed immediately in an interactive mode

Scripts are reusable

Basically, a script is a text file containing the statements that comprise a Python program. Once you have created the script, you can execute it over and over without having to retype it each time.

Scripts are editable

Perhaps, more importantly, you can make different versions of the script by modifying the statements from one file to the next using a text editor. Then you can execute each of the individual versions. In this way, it is easy to create different programs with a minimum amount of typing.

You will need a text editor

Just about any text editor will suffice for creating Python script files.

You can use *Microsoft Notepad, Microsoft WordPad, Microsoft Word,* or just about any word processor if you want to.

Difference between a script and a program

Script:

Scripts are distinct from the core code of the application, which is usually written in a different language, and are often created or at least modified by the end-user. Scripts are often interpreted from source code or byte code, whereas the applications they control are traditionally compiled to native machine code.

Program:

The program has an executable form that the computer can use directly to execute the instructions.

The same program in its human-readable source code form, from which executable programs are derived (e.g., compiled)

Python

What is Python? Chances you are asking yourself this. You may have found this book because you want to learn to program but don't know anything about programming languages. Or you may have heard of programming languages like C, C++, C#, or Java and want to know what Python is and how it compares to "big name" languages. Hopefully I can explain it for you.

Python concepts

If you're not interested in the how's and whys of Python, feel free to skip to the next chapter. In this chapter I will try to explain to the reader why I think Python is one of the best languages available and why it's a great one to start programming with.

• Open-source general-purpose language.

• Object Oriented, Procedural, Functional

• Easy to interface with C/ObjC/Java/Fortran

• Easy-is to interface with C++ (via SWIG)

• Great interactive environment

• Great interactive environment

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- Python is a Beginner's Language − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Possum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, and UNIX shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Possum still holds a vital role in directing its progress.

Python Features

Python's features include −

- Easy-to-learn − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- Easy-to-read − Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain − Python's source code is fairly easy-to-maintained.
- A broad standard library − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- Interactive Mode − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

- Portable − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- Extendable − you can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- Databases − Python provides interfaces to all major commercial databases.
- GUI Programming − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- Scalable − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Dynamic vs. Static

Types Python is a dynamic-typed language. Many other languages are static typed, such as C/C++ and Java. A static typed language requires the programmer to explicitly tell the computer what type of "thing" each data value is.

For example, in C if you had a variable that was to contain the price of something, you would have to declare the variable as a "float" type.

This tells the compiler that the only data that can be used for that variable must be a floating point number, i.e. a number with a decimal point.

If any other data value was assigned to that variable, the compiler would give an error when trying to compile the program.

Python, however, doesn't require this. You simply give your variables names and assign values to them. The interpreter takes care of keeping track of what kinds of objects your program is using. This also means that you can change the size of the values as you develop the program. Say you have another decimal number (a.k.a. a floating point number) you need in your program.

With a static typed language, you have to decide the memory size the variable can take when you first initialize that variable. A double is a floating point value that can handle a much larger number than a normal float (the actual memory sizes depend on the operating environment).

If you declare a variable to be a float but later on assign a value that is too big to it, your program will fail; you will have to go back and change that variable to be a double.

With Python, it doesn't matter. You simply give it whatever number you want and Python will take care of manipulating it as needed. It even works for derived values.

For example, say you are dividing two numbers. One is a floating point number and one is an integer. Python realizes that it's more accurate to keep track of decimals so it automatically calculates the result as a floating point number

Variables

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.
Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data

types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types −

- Numbers
- String
- List
- Tuple
- Dictionary

Python Numbers

Number data types store numeric values. Number objects are created when you assign a value to them

Python Strings

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

Python Lists

Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type. The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

Python Tuples

A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.
The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated. Tuples can be thought of as read-only lists.

Python Dictionary

Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.
Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

Different modes in python

Python has two basic modes: normal and interactive.
The normal mode is the mode where the scripted and finished .pie files are run in the Python interpreter.
Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory. As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole

20 Python libraries

1. Requests. The most famous http library written by Kenneth remits. It's a must have for every python developer.
2. Scrappy. If you are involved in web scraping then this is a must have library for you. After using this library you won't use any other.

3. Python. A guy toolkit for python. I have primarily used it in place of tinder. You will really love it.
4. Pillow. A friendly fork of PIL (Python Imaging Library). It is more user friendly than PIL and is a must have for anyone who works with images.
5. SQL Alchemy. A database library. Many love it and many hate it. The choice is yours.
6. Beautiful Soup. I know it's slow but this xml and html parsing library is very useful for beginners.
7. Twisted. The most important tool for any network application developer. It has a very beautiful ape and is used by a lot of famous python developers.
8. Numbly. How can we leave this very important library? It provides some advance math functionalities to python.
9. Skippy. When we talk about numbly then we have to talk about spicy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.
10. Matplotlib. A numerical plotting library. It is very useful for any data scientist or any data analyser.
11. Pygmy. Which developer does not like to play games and develop them? This library will help you achieve your goal of 2d game development.
12. Piglet. A 3d animation and game creation engine. This is the engine in which the famous python port of mine craft was made
13. Pit. A GUI toolkit for python. It is my second choice after python for developing GUI's for my python scripts.
14. Pit. Another python GUI library. It is the same library in which the famous Bit torrent client is created.
15. Scaly. A packet sniffer and analyser for python made in python.
16. Pywin32. A python library which provides some useful methods and classes for interacting with windows.
17. Notch. Natural Language Toolkit – I realize most people won't be using this one, but it's generic enough. It is a very useful library if you want to manipulate strings. But its capacity is beyond that. Do check it out.

18. Nose. A testing framework for python. It is used by millions of python developers. It is a must have if you do test driven development.
19. Simply. Simply can-do algebraic evaluation, differentiation, expansion, complex numbers, etc. It is contained in a pure Python distribution.
20. I Python. I just can't stress enough how useful this tool is. It is a python prompt on steroids. It has completion, history, shell capabilities, and a lot more. Make sure that you take a look at it.

NumPy

Humpy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In numbly dimensions are called *axes*. The number of axes is *rank*.

• Offers Matlab-ish capabilities within Python

• Fast array operations

• 2D arrays, multi-D arrays, linear algebra etc.

Matplotlib

• High quality plotting library.

Python class and objects

These are the building blocks of OOP. Class creates a new object. This object can be anything, whether an abstract data concept or a model of a physical object, e.g. a chair. Each class has individual characteristics unique to that class, including variables and methods. Classes are very powerful and currently "the big thing" in most programming languages. Hence, there are several chapters dedicated to OOP later in the book.

The class is the most basic component of object-oriented programming. Previously, you learned how to use functions to make your program do something.

Now will move into the big, scary world of Object-Oriented Programming (OOP). To be honest, it took me several months to get a handle on objects.

When I first learned C and C++, I did great; functions just made sense for me.

Having messed around with BASIC in the early ’90s, I realized functions were just like subroutines so there wasn’t much new to learn.

However, when my C++ course started talking about objects, classes, and all the new features of OOP, my grades definitely suffered.

Once you learn OOP, you’ll realize that it’s actually a pretty powerful tool. Plus many Python libraries and APIs use classes, so you should at least be able to understand what the code is doing.

One thing to note about Python and OOP: it’s not mandatory to use objects in your code in a way that works best; maybe you don’t need to have a full-blown class with initialization code and methods to just return a calculation. With Python, you can get as technical as you want.

As you’ve already seen, Python can do just fine with functions. Unlike languages such as Java, you aren’t tied down to a single way of doing things; you can mix functions and classes as necessary in the same program. This lets you build the code

Objects are an encapsulation of variables and functions into a single entity. Objects get their variables and functions from classes. Classes are essentially a template to create your objects.

Here’s a brief list of Python OOP ideas:

• The class statement creates a class object and gives it a name. This creates a new namespace.

• Assignments within the class create class attributes. These attributes are accessed by qualifying the name using dot syntax: ClassName.Attribute.

• Class attributes export the state of an object and its associated behaviour. These attributes are shared by all instances of a class.

• Calling a class (just like a function) creates a new instance of the class.

This is where the multiple copy's part comes in.

• Each instance gets ("inherits") the default class attributes and gets its own namespace. This prevents instance objects from overlapping and confusing the program.

• Using the term self identifies a particular instance, allowing for per-instance attributes. This allows items such as variables to be associated with a particular instance.

Inheritance

First off, classes allow you to modify a program without really making changes to it.

To elaborate, by sub classing a class, you can change the behaviour of the program by simply adding new components to it rather than rewriting the existing components.

As we've seen, an instance of a class inherits the attributes of that class.

However, classes can also inherit attributes from other classes. Hence, a subclass inherits from a superclass allowing you to make a generic superclass that is specialized via subclasses.

The subclasses can override the logic in a superclass, allowing you to change the behaviour of your classes without changing the superclass at all.

Operator Overloads

Operator overloading simply means that objects that you create from classes can respond to actions (operations) that are already defined within Python, such as addition, slicing, printing, etc.

Even though these actions can be implemented via class methods, using overloading ties the behavior closer to Python's object model and the object interfaces are more consistent to Python's built-in objects, hence overloading is easier to learn and use.

User-made classes can override nearly all of Python's built-in operation methods

Exceptions

I've talked about exceptions before but now I will talk about them in depth. Essentially, exceptions are events that modify program's flow, either intentionally or due to errors.

They are special events that can occur due to an error, e.g. trying to open a file that doesn't exist, or when the program reaches a marker, such as the completion of a loop.

Exceptions, by definition, don't occur very often; hence, they are the "exception to the rule" and a special class has been created for them. Exceptions are everywhere in Python.

Virtually every module in the standard Python library uses them, and Python itself will raise them in a lot of different circumstances.

Here are just a few examples:

• Accessing a non−existent dictionary key will raise a Key Error exception.

• Searching a list for a non−existent value will raise a Value Error exception

. • Calling a non−existent method will raise an Attribute Error exception.

• Referencing a non−existent variable will raise a Name Error exception.

• Mixing data types without coercion will raise a Type Error exception.

One use of exceptions is to catch a fault and allow the program to continue working; we have seen this before when we talked about files.

This is the most common way to use exceptions. When programming with the Python command line interpreter, you don't need to worry about catching exceptions.

Your program is usually short enough to not be hurt too much if an exception occurs.

Plus, having the exception occur at the command line is a quick and easy way to tell if your code logic has a problem.

However, if the same error occurred in your real program, it will fail and stop working. Exceptions can be created manually in the code by raising an exception.

It operates exactly as a system-caused exceptions, except that the programmer is doing it on purpose. This can be for a number of reasons. One of the benefits of using exceptions is that, by their nature, they don't put any overhead on the code processing.

Because exceptions aren't supposed to happen very often, they aren't processed until they occur.

Exceptions can be thought of as a special form of the if/elf statements. You can realistically do the same thing with if blocks as you can with exceptions.

However, as already mentioned, exceptions aren't processed until they occur; if blocks are processed all the time.

Proper use of exceptions can help the performance of your program.

The more infrequent the error might occur, the better off you are to use exceptions; using if blocks requires Python to always test extra conditions before continuing.

Exceptions also make code management easier: if your programming logic is mixed in with error-handling if statements, it can be difficult to read, modify, and debug your program.

User-Defined Exceptions

I won't spend too much time talking about this, but Python does allow for a programmer to create his own exceptions.

You probably won't have to do this very often but it's nice to have the option when necessary.

However, before making your own exceptions, make sure there isn't one of the built-in exceptions that will work for you.

They have been "tested by fire" over the years and not only work effectively, they have been optimized for performance and are bug-free.

Making your own exceptions involves object-oriented programming, which will be covered in the next chapter

. To make a custom exception, the programmer determines which base exception to use as the class to inherit from, e.g. making an exception for negative numbers or one for imaginary numbers would probably fall under the Arithmetic Error exception class.

To make a custom exception, simply inherit the base exception and define what it will do.

Python modules

Python allows us to store our code in files (also called modules). This is very useful for more serious programming, where we do not want to retype a long function definition from the very beginning just to change one mistake. In doing this, we are essentially defining our own modules, just like the modules defined already in the Python library.

To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a *module*; definitions from a module can be *imported* into other modules or into the *main* module.

Testing code

As indicated above, code is usually developed in a file using an editor.

To test the code, import it into a Python session and try to run it.

Usually there is an error, so you go back to the file, make a correction, and test again.

This process is repeated until you are satisfied that the code works. T

His entire process is known as the development cycle.

There are two types of errors that you will encounter. Syntax errors occur when the form of some command is invalid.

This happens when you make typing errors such as misspellings, or call something by the wrong name, and for many other reasons. Python will always give an error message for a syntax error.

Functions in Python

It is possible, and very useful, to define our own functions in Python. Generally speaking, if you need to do a calculation only once, then use the interpreter. But when you or others have need to perform a certain type of calculation many times, then define a function.

You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-program and called when needed. That means that a function is a piece of code written to carry out a specified task.

To carry out that specific task, the function might or might not need multiple inputs. When the task is carved out, the function can or cannot return one or more values.

There are three types of functions in python:

Help (), min (), print ().

Namespaces in Python are implemented as Python dictionaries, this means it is a mapping from names (keys) to objects (values). The user doesn't have to know this to write a Python program and when using namespaces.

Some namespaces in Python:

- global names of a module
- local names in a function or method invocation
- built-in names: this namespace contains built-in functions (e.g. abs(), camp(), ...) and built-in exception names

Garbage Collection

Garbage Collector exposes the underlying memory management mechanism of Python, the automatic garbage collector. The module includes functions for controlling how the collector

operates and to examine the objects known to the system, either pending collection or stuck in reference cycles and unable to be freed.

Python XML Parser

XML is a portable, open source language that allows programmers to develop applications that can be read by other applications, regardless of operating system and/or developmental language.

What is XML? The Extensible Markup Language XML is a markup language much like HTML or SGML.

This is recommended by the World Wide Web Consortium and available as an open standard.

XML is extremely useful for keeping track of small to medium amounts of data without requiring a SQL-based backbone.

XML Parser Architectures and APIs the Python standard library provides a minimal but useful set of interfaces to work with XML.

The two most basic and broadly used APIs to XML data are the SAX and DOM interfaces.

Simple API for XML SAX: Here, you register callbacks for events of interest and then let the parser proceed through the document.

This is useful when your documents are large or you have memory limitations, it parses the file as it reads it from disk and the entire file is never stored in memory.

Document Object Model DOM API : This is a World Wide Web Consortium recommendation wherein the entire file is read into memory and stored in a hierarchical tree − based form to represent all the features of an XML document.

SAX obviously cannot process information as fast as DOM can when working with large files. On the other hand, using DOM exclusively can really kill your resources, especially if used on a lot of small files.

SAX is read-only, while DOM allows changes to the XML file. Since these two different APIs literally complement each other, there is no reason why you cannot use them both for large projects.

Python Web Frameworks

A web framework is a code library that makes a developer's life easier when building reliable, scalable and maintainable web applications.

Why are web frameworks useful?
Web frameworks encapsulate what developers have learned over the past twenty years while programming sites and applications for the web. Frameworks make it easier to reuse code for common HTTP operations and to structure projects so other developers with knowledge of the framework can quickly build and maintain the application.
Common web framework functionality

Frameworks provide functionality in their code or through extensions to perform common operations required to run web applications. These common operations include:

1. URL routing
2. HTML, XML, JSON, and other output format tinplating
3. Database manipulation
4. Security against Cross-site request forgery (CSRF) and other attacks
5. Session storage and retrieval

Not all web frameworks include code for all of the above functionality. Frameworks fall on the spectrum from executing a single use case to providing every known web framework feature to every developer. Some frameworks take the "batteries-included" approach where everything possible comes bundled with the framework while others have a minimal core package that is amenable to extensions provided by other packages.

Comparing web frameworks

There is also a repository called compare-python-web-frameworks where the same web application is being coded with varying Python web frameworks, tinplating engines and object.

Web framework resources

- When you are learning how to use one or more web frameworks it's helpful to have an idea of what the code under the covers is doing.
- Frameworks is a really well done short video that explains how to choose between web frameworks. The author has some particular opinions about what should be in a framework. For the most part I agree although I've found sessions and database ORMs to be a helpful part of a framework when done well.
- What is a web framework? Is an in-depth explanation of what web frameworks are and their relation to web servers?
- Jingo vs. Flash vs. Pyramid: Choosing a Python web framework contains background information and code comparisons for similar web applications built in these three big Python frameworks.
- This fascinating blog post takes a look at the code complexity of several Python web frameworks by providing visualizations based on their code bases.
- Python's web frameworks benchmarks is a test of the responsiveness of a framework with encoding an object to JSON and returning it as a response as well as retrieving data from the database and rendering it in a template. There were no conclusive results but the output is fun to read about nonetheless.
- What web frameworks do you use and why are they awesome? Is a language agnostic Reedit discussion on web frameworks? It's interesting to see what programmers in other languages like and dislike about their suite of web frameworks compared to the main Python frameworks.
- This user-voted question & answer site asked "What are the best general purpose Python web frameworks usable in production?" The votes aren't as important as the list of the many frameworks that are available to Python developers.

Web frameworks learning checklist

1. Choose a major Python web framework (Jingo or Flask are recommended) and stick with it. When you're just starting it's best to learn one framework first instead of bouncing around trying to understand every framework.

2. Work through a detailed tutorial found within the resources links on the framework's page.
3. Study open source examples built with your framework of choice so you can take parts of those projects and reuse the code in your application.
4. Build the first simple iteration of your web application then go to the deployment section to make it accessible on the web.

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually

satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

SYSTEM_TEST

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

6.1 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

6.2 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

6.3 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Domain: Machine Learning
Technology: Python