A

Major Project Report

On

START-UP SUCCESS PREDICTION USING MACHINE LEARING

Submitted to **CMREC, HYDERABAD**

In Partial Fulfillment of the requirements for the Award of Degree of
BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

Submitted By

U. NIKHITHA	**(218R1A6762)**
K. NAGARAJU	**(218R1A6735)**
CH. SRIHITH	**(218R1A6719)**
SK. RAHAMATH	**(218R1A6758)**

Under the Esteemed guidance of

Mrs. P. RENUKA

Assistant Professor, Department of CSE (Data Science)

Department of Computer Science & Engineering (DataScience)

CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

(Approved by AICTE, NEW DELHI, Affiliated to JNTU, Hyderabad) Kandlakoya, Medchal Road, R.R. Dist. Hyderabad-501401.

2024-2025

CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

(Accredited by NBA,Approved by AICTE NEW DELHI, Affiliated to JNTU, Hyderabad)

Kandlakoya, Medchal Road, Hyderabad-501 401

Department of Computer Science & Engineering (Data Science)

CERTIFICATE

This is to certify that the project entitled **"Start-up Success Prediction Using Machine Learning"** is a Bonafide work carried out by

U. Nikhitha	**(218R1A6762)**
K. Nagaraju	**(218R1A6735)**
Ch. Srihith	**(218R1A6719)**
Sk. Rahamath	**(218R1A6758)**

in partial fulfillment of the requirement for the award of the degree of **BACHELOR OFTECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)** from CMR Engineering College, affiliated to JNTU, Hyderabad, under our guidance and supervision.

The results presented in this Major project have been verified and are found to be satisfactory. The results embodied in this Major project have not been submitted to any other university for the award of any other degree or diploma.

Internal Guide	**Project Coordinator**	**Head of the Department**	**External Examiner**
Mrs. P. Renuka	**Mrs. G. Shruthi**	**Dr. M. Laxmaiah**	
Assistant Professor	Assistant Professor	Professor & H.O.D	
CSE (Data Science), CMREC	CSE (Data Science), CMREC	CSE (Data Science), CMREC	

DECLARATION

This is to certify that the work reported in the present Major project entitled " **Start-Up Success Prediction Using Machine Learning"** is a record of Bonafide work done by us in the Department of Computer Science and Engineering (Data Science), CMR Engineering College, JNTU Hyderabad. The reports are based on the project work done entirely by us and not copied from any other source. We submit our project for further development by any interested students who share similar interests to improve the project in the future.

The results embodied in this Major project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of our knowledge and belief.

U. Nikhitha (218R1A6762)

K. Nagaraju (218R1A6735)

Ch. Srihith (218R1A6719)

Sk. Rahamath (218R1A6758)

ACKNOWLEDGMENT

We are extremely grateful to **Dr. A. Srinivasula Reddy**, Principal, and **Dr. M. Laxmaiah**, Professor HOD, **Department of CSE (DS), CMR Engineering College,** for their constant support**.**

We are extremely thankful to **Mrs. P. Renuka,** Assistant Professor, Internal Guide, Department of CSE(DS), for her constant guidance, encouragement and moral support throughout the project.

We will be failing in duty if We do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this Project.

We thank **Mrs. G. Shruthi**, Assistant Professor, CSE(DS) Department**,** Major Project Coordinator for her constant support in carrying out the project activities and reviews.

We express my thanks to all staff members and friends for all the help and co-ordination extended in bringing out this project successfully in time.

Finally, We are very much thankful to our parents who guided me for every step.

U. Nikhitha (218R1A6762)
K. Nagaraju (218R1A6735)
Ch. Srihith (218R1A6719)
Sk. Rahamath (218R1A6758)

ABSTRACT

Startup refers to a company that is in their first stages of operations. Startups can be founded by one or more entrepreneurs who are working on developing a product or service. Startups normally have very high cost and limited revenue. As they require a lot of capital to take it off the ground, they look for capital from a lot of sources like venture capitalists.

Startups go through multiple rounds of funding to raise capital. The different funding rounds that let outside investors the opportunity to invest cash in exchange of equity or partial ownership of the company. Other types of investments are debt, convertible note, stock or dividends. Startups can start off with “seed” funding or angel investor funding at the beginning. The next funding rounds can be followed by Series A, B, C and so on. Goal of most startups is to get acquired by a different company or become a publicly traded company.

90% of startups fail due to bad product market fit, marketing problems, team problems or other issues. They also fail within the first few years. This makes startup investment very risky. Historically only venture capitalists could invest in startups but due to the recent trend in crowdfunding sites, an average investor can easily grab a piece of an exciting startup.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 OVERVIEW

In today's fast-paced and dynamic business landscape, startups play a pivotal role in driving innovation, economic growth, and job creation. Defined as companies in their early stages of operations, startups are often characterized by their agility, creativity, and disruptive potential. These fledgling ventures are typically founded by visionary entrepreneurs who identify gaps in the market and seek to develop innovative products or services to address unmet needs or challenges.

The startup journey is marked by a myriad of challenges and opportunities, with success often hinging on a delicate balance of factors such as market demand, product-market fit, team dynamics, and access to capital. While some startups soar to unprecedented heights, achieving unicorn status and garnering global recognition, others falter and fade into obscurity, succumbing to the harsh realities of competition, resource constraints, or misaligned strategies.

Despite the inherent risks associated with startup investment, the allure of backing the next big disruptor has captivated the imagination of investors, venture capitalists, and angel investors alike. The prospect of reaping substantial returns on investment, coupled with the excitement of nurturing groundbreaking innovations, has fueled a vibrant ecosystem of startup funding and entrepreneurship worldwide.

However, navigating the treacherous waters of startup investment is far from straightforward. With a staggering 90% of startups failing within the first few years of operation, according to some estimates, the odds are stacked against investors seeking to identify the next success story. The inherent uncertainty and unpredictability of startup outcomes make investment decisions inherently risky, requiring investors to carefully evaluate a multitude of factors and make informed strategic decisions.

Traditionally, startup investors have relied on a combination of market research, due diligence, and gut instinct to assess the potential viability of a startup. Factors such as the founding team's experience, market opportunity, competitive landscape, and revenue projections are often taken into consideration when evaluating investment opportunities. However, the subjective nature of these assessments and the limited predictive power of traditional methods leave investors vulnerable to biases and inaccuracies.

By harnessing the power of advanced algorithms and vast troves of data, investors now have the ability to analyze complex patterns and trends, uncover hidden insights, and make

more accurate predictions about startup success.

One such machine learning technique that holds promise in the domain of startup prediction is Ada Boosting. As a supervised learning algorithm belonging to the ensemble learning family, Ada Boosting combines multiple weak learners sequentially to build a robust predictive model. By iteratively adjusting the weights of misclassified data points, Ada Boosting aims to improve the overall predictive accuracy of the model, making it particularly well-suited for tackling complex and nuanced problems such as startup success prediction.

In this context, the present study seeks to explore the potential of Ada Boosting and other machine learning techniques in predicting startup success. By analyzing a diverse array of factors ranging from team composition and market dynamics to product differentiation and funding history, we aim to develop a predictive model that can assist investors, entrepreneurs, and decision-makers in allocating resources more effectively and making informed strategic decisions.

Through a comprehensive examination of historical data and real-world case studies, we endeavor to uncover the critical factors that contribute to startup success and develop actionable insights that can guide investment decisions in a rapidly evolving and increasingly competitive startup landscape. By bridging the gap between data science and startup investment, we aspire to empower stakeholders with the tools and knowledge needed to navigate the complexities of the startup ecosystem and unlock the full potential of innovative ventures.

The intersection of machine learning and startup investment represents a paradigm shift in how stakeholders approach decision-making in the venture capital landscape. By harnessing the predictive capabilities of advanced algorithms, investors can gain deeper insights into the underlying factors that drive startup success and failure, thereby mitigating risks and maximizing returns on investment.

The advent of crowdfunding platforms and the democratization of startup investment have further democratized access to capital, enabling individual investors to participate in the high-stakes world of startup funding. However, with increased accessibility comes heightened uncertainty, as novice investors may lack the expertise and resources to evaluate investment opportunities effectively.

In this context, the development of robust predictive models that leverage machine learning techniques holds immense potential to level the playing field and empower investors of all stripes to make more informed and data-driven decisions. By analyzing vast quantities of data from diverse sources, including financial statements, market trends, and social media sentiment, these models can uncover hidden patterns and correlations that elude traditional.

Moreover, the predictive power of machine learning extends beyond mere speculation, offering tangible benefits to entrepreneurs, startups, and the broader economy. By identifying early warning signs of potential failure and opportunities for growth, these models can help entrepreneurs refine their strategies, optimize resource allocation, and increase their chances of success.

Furthermore, by facilitating more efficient allocation of capital and resources, machine learning can drive innovation and economic growth, fostering a thriving ecosystem of startups that fuel job creation, spur technological advancement, and drive societal progress.

However, it is essential to recognize the inherent limitations and challenges associated with machine learning-based startup prediction. Data quality, availability, and bias are perennial concerns that must be addressed to ensure the integrity and reliability of predictive models. Moreover, the dynamic and unpredictable nature of the startup landscape poses unique challenges to modeling and forecasting, requiring ongoing refinement and adaptation of predictive algorithms. these challenges, the potential benefits of leveraging machine learning in startup investment are too significant to ignore. By embracing data-driven decision-making and harnessing the power of predictive analytics, investors, entrepreneurs, and decision-makers can unlock new opportunities, mitigate risks, and drive sustainable growth in the ever-evolving world of startups.

1.2 PROBLEM DEFINITION

Startup investment can be very risky due to the high failure rate of startups. People like angel investors and venture capitalists have a very high risk while they are investing in startups. To assist startup investors with their decisions, in this project we aim to find the important features that lead to startup success and forecast a company's success with supervised machine learning methods.

ADABOOST (ADAPTIVE BOOSTING) ALGORITHM

Adaboost (Adaptive Boosting) is a machine learning ensemble technique that improves the accuracy of weak classifiers by combining multiple weak learners into a strong classifier. It assigns different weights to training samples and iteratively adjusts these weights to focus more on difficult examples.

How AdaBoost Works

1. **Initialize Weights**:
    - Assign equal weights to all training samples.
2. **Train Weak Classifier:**

- Using weighted training data, ◦ Train a weak model (e.g., decision stump).

3. **Compute Error:**
    - Calculate the weighted error of the weak classifier.

4. **Update Weights:**
    - Increase the weight of misclassified samples so that the next classifier focuses more on difficult cases.
5. **Combine Weak Classifiers:**
    - Assign a weight to the weak classifier based on its accuracy.
    - Repeat the process for multiple weak classifiers and combine them into a final strong classifier.
6. **Final Decision:**
    - The final prediction is made using a weighted vote of all weak classifiers.

Key Features

- Uses weak learners (e.g., decision stumps).
- Focuses more on difficult samples by adjusting weights.
- Reduces bias and variance, improving accuracy.
- Works well with structured data.

Advantages of AdaBoost

1. **Improves Weak Learners**:
    - Converts weak classifiers into a strong classifier, improving overall accuracy.
2. **Reduces Bias and Variance**:
    - Helps in reducing both bias (underfitting) and variance (overfitting) by focusing on misclassified samples.
3. **Versatile & Can Work with Different Models**:
    - Can be combined with various weak learners like decision trees, SVM, or neural networks.
4. **Feature Selection**:
    - Assigns higher weights to important features, making it useful for feature selection.
5. **Handles Class Imbalances**:
    - Gives more weight to misclassified instances, helping in class im balance .

2. LITERATURE SURVEY

LITERATURE REVIEW ON ADA BOOSTING ALGORITHM

Ada Boosting, short for Adaptive Boosting, is a powerful supervised machine learning algorithm widely used in various domains for classification and regression tasks. Developed by Freund and Schapire in 1996, Ada Boosting belongs to the ensemble learning family, specifically boosting algorithms. The following literature review explores the key concepts, methodologies, and applications of Ada Boosting as reported in the academic literature.

ALGORITHM OVERVIEW:

Ada Boosting operates by sequentially combining multiple weak learners or base classifiers to create a strong learner. Weak learners are typically simple classifiers, such as decision trees with limited depth or linear classifiers, that perform slightly better than random guessing. In each iteration, AdaBoost assigns equal weights to all observations in the training dataset and focuses on correctly classifying previously misclassified data points. By iteratively adjusting the weights of misclassified observations, AdaBoost places greater emphasis on difficult-to-classify instances, thereby improving overall predictive performance.

PERFORMANCE ENHANCEMENT:

The adaptability and sequential nature of AdaBoosting contribute to its remarkable performance in handling complex classification problems. By iteratively refining the decision boundaries based on the distribution of misclassified instances, AdaBoosting effectively learns from its mistakes and adapts to the underlying data distribution. This adaptive learning process enables AdaBoosting to achieve high accuracy even with relatively simple weak learners, making it particularly suitable for ensemble-based classification tasks.

APPLICATIONS:

Ada Boosting has found widespread applications across various domains, including computer vision, natural language processing, bioinformatics, and finance. In computer vision, Ada Boosting has been successfully applied to object detection, face recognition, and image segmentation tasks, where accurate classification of visual patterns is critical. In natural language processing, Ada Boosting has been used for text classification, sentiment analysis, and document categorization, leveraging its ability to handle high-dimensional feature spaces and noisy data.

Furthermore, Ada Boosting has been deployed in bioinformatics for protein classification, gene expression analysis, and disease diagnosis, where accurate prediction of biological phenomena is essential for understanding complex biological systems. In finance, Ada Boosting has been

Study	Image Modality	Task	Method	Strengths	Weaknesses
Wrobel, M., Plotka, W.	Crowdfunding Platforms	Analyze startup growth strategies.	Equity-based Crowdfunding	Provides insights on startup funding via crowdfunding	Lacks analysis of alternative crowdfunding models
Dushnitsky, G., Fitze, M.	Venture Capital Ecosystem	Study VC impact on startups	VC Funding Lifecycle Analysis	Explores investor roles in ecosystem evolution	Limited focus on regional and sector-specific VC trends
Ribeiro, F., Raposo, M.	Startup Success Factors	Identify key success factors.	Machine Learning Techniques	Uses AI to identify success factors in startups	Uses a limited dataset, reducing generalizability
Garcia, R., Paris-Ortiz, M.	Cryptocurrency-based Funding	Investigate crypto in startup financing	Decentralized Funding Mechanisms	Highlights emerging trends in decentralized finance	Does not deeply evaluate regulatory and adoption issues

employed for credit scoring, fraud detection, and stock market prediction, leveraging its robustness

Table No: 2.1 Literature Survey table of Startup Success prediction

CHALLENGES AND FUTURE DIRECTIONS:

While AdaBoosting has demonstrated remarkable success in a wide range of applications, several challenges remain to be addressed. These include scalability issues with large datasets, sensitivity to noisy data and outliers, and potential overfitting when weak learners are too complex. Future research directions may focus on developing scalable and robust variants of AdaBoosting, incorporating domain-specific knowledge to enhance model interpretability, and exploring ensemble learning techniques beyond traditional boosting algorithms.

In conclusion, AdaBoosting stands as a versatile and effective machine learning algorithm for

classification tasks, offering adaptive learning capabilities and high predictive accuracy. Its widespread adoption across diverse domains underscores its utility and relevance in addressing real-world challenges. As machine learning continues to evolve, AdaBoosting remains a valuable tool in the data scientist's arsenal, providing a powerful framework for ensemble-based learning

FURTHER EXPLORATION OF APPLICATIONS:

In addition to the previously mentioned applications, AdaBoosting has been extensively applied in various other fields due to its versatility and effectiveness. For instance, in the field of marketing and customer relationship management, AdaBoosting has been utilized for customer segmentation, churn prediction, and targeted marketing campaigns. By leveraging AdaBoosting's ability to handle high-dimensional data and complex decision boundaries, marketers can identify patterns and trends in customer behavior, personalize marketing strategies, and optimize customer engagement efforts.

Moreover, in healthcare and medical research, AdaBoosting has been employed for disease diagnosis, medical image analysis, and drug discovery. With the increasing availability of electronic health records and medical imaging data, AdaBoosting offers a powerful tool for analyzing patient data, predicting disease outcomes, and assisting clinical decision-making. By integrating AdaBoosting with advanced data analytics techniques such as deep learning and transfer learning, researchers can unlock new insights into disease mechanisms, drug responses, and personalized treatment strategies.

Despite its widespread adoption and success, AdaBoosting faces several challenges and limitations that warrant further investigation. For example, the algorithm may struggle with imbalanced datasets, where one class is significantly more prevalent than others, leading to biased predictions and suboptimal performance.

Addressing this challenge requires the development of advanced techniques for handling class imbalance, such as resampling methods, cost-sensitive learning, or ensemble pruning strategies.

Moreover, AdaBoosting's reliance on decision stumps or shallow trees as weak learners may limit its ability to capture complex nonlinear relationships in the data. Future research efforts may focus on exploring alternative weak learners, such as deep neural networks, kernel methods, or ensemble models, to enhance AdaBoosting's flexibility and expressiveness. Additionally, incorporating domain-specific knowledge and incorporating interpretable models into the ensemble could improve the transparency and explainability of AdaBoosting's predictions, facilitating trust and adoption in real-world applications.

1. EXISTING SYSTEM

Currently, startup investors rely on various factors and their own expertise to make investment decisions. They may consider aspects such as the team's experience, the uniqueness of the product or service, market demand, financial projections, and industry trends. However, decision-making in this context can be subjective and prone to biases. Additionally, predicting the success of a startup accurately remains challenging due to the complex and dynamic nature of the startup ecosystem.

Startup investors base their decisions on a combination of qualitative and quantitative factors, leveraging their experience, industry knowledge, and analytical skills. Key considerations include the founding team’s expertise, track record, and ability to execute their vision effectively. Investors also evaluate the uniqueness and competitive advantage of the product or service, ensuring it addresses a real market need with a scalable business model.

Market demand plays a crucial role, as investors analyze industry trends, customer adoption potential, and competitive landscape to determine long-term viability. Financial projections, including revenue models, profitability timelines, and capital efficiency, are scrutinized to assess the startup’s financial health and growth potential.

DISADVANTAGES:

Subjectivity and Bias: Investment decisions are often influenced by subjective judgments and biases, leading to potential inaccuracies.

Limited Predictive Power: Traditional methods may lack the predictive power to accurately forecast startup success, as they may not consider all relevant factors or account for dynamic market conditions.

Time-Consuming and Resource-Intensive: Analyzing multiple variables manually can be time-consuming and resource-intensive for investors, limiting their ability to evaluate a large number of potential investment opportunities effectively.

2. PROPOSED METHODOLOGY

The proposed system aims to leverage machine learning techniques, specifically Ada Boosting, to enhance the prediction of startup success. By utilizing a supervised learning approach, the system will analyze a diverse set of features and historical data to identify patterns and trends associated with successful startups. AdaBoosting, as an ensemble learning method, combines multiple weak learners sequentially to build a strong predictive model.

ADVANTAGES

1. Enhanced Predictive Accuracy

- By combining multiple weak learners, AdaBoost reduces bias and variance, leading to higher accuracy compared to traditional decision-making.
- It considers multiple factors systematically, reducing human subjectivity.

2. Automated and Scalable Analysis

- Traditional startup analysis is manual and time-consuming. The proposed system automates data analysis, allowing investors to evaluate multiple startups quickly.
- Scalability: Can process thousands of startups in a short time, unlike human analysts who have limitations.

3. Adaptability to Changing Market Conditions

- The model continuously learns from new data, adjusting to evolving market trends and improving predictions over time.
- Investors can receive real-time insights, helping them make informed decisions.

4. Reduction of Investment Risks

- By leveraging machine learning, the system minimizes subjective biases, ensuring more objective and data-backed investment choices.
- Early warning signals: Detects potential risks in startups that might fail based on historical patterns.

Challenges and Considerations

While AdaBoost enhances predictive power, the system must also address:

- Data Availability & Quality: Ensuring accurate and high-quality historical data for training.
- Handling Noisy and Imbalanced Data: Some startups may have incomplete or misleading data, requiring data preprocessing techniques.

3. METHODOLOGY

3.1 SDLC (Software Development Life Cycle) – Umbrella Model

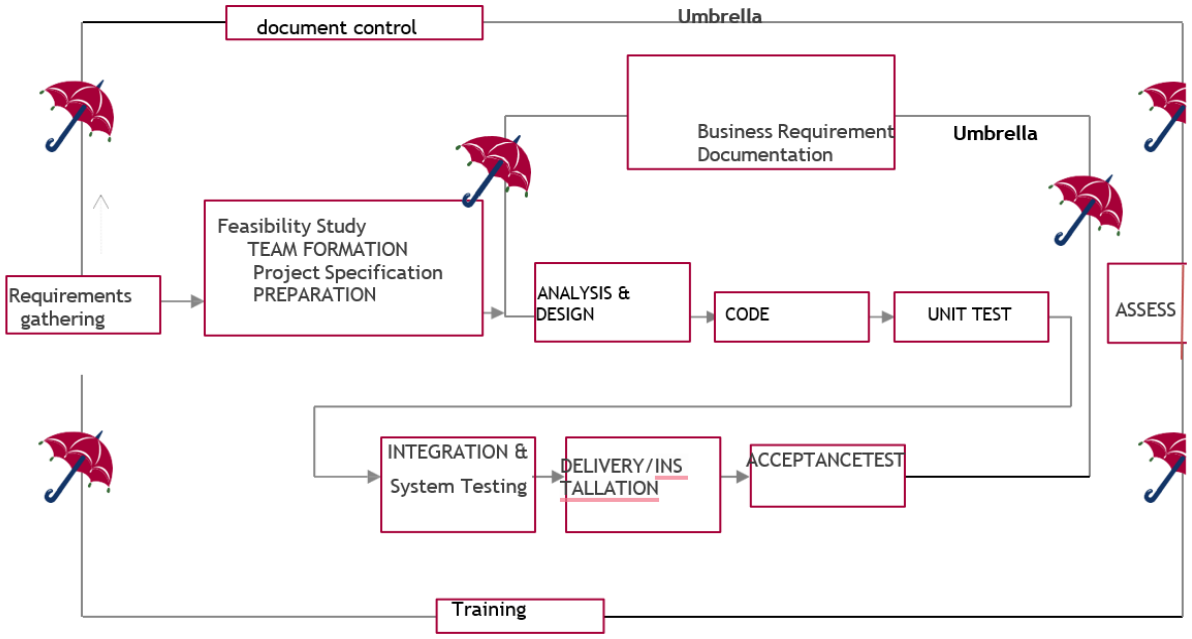


Fig no. 5.1 Umbrella model for methodology.

SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

3.2 REQUIREMENTS GATHERING STAGE:

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title a Textual description.

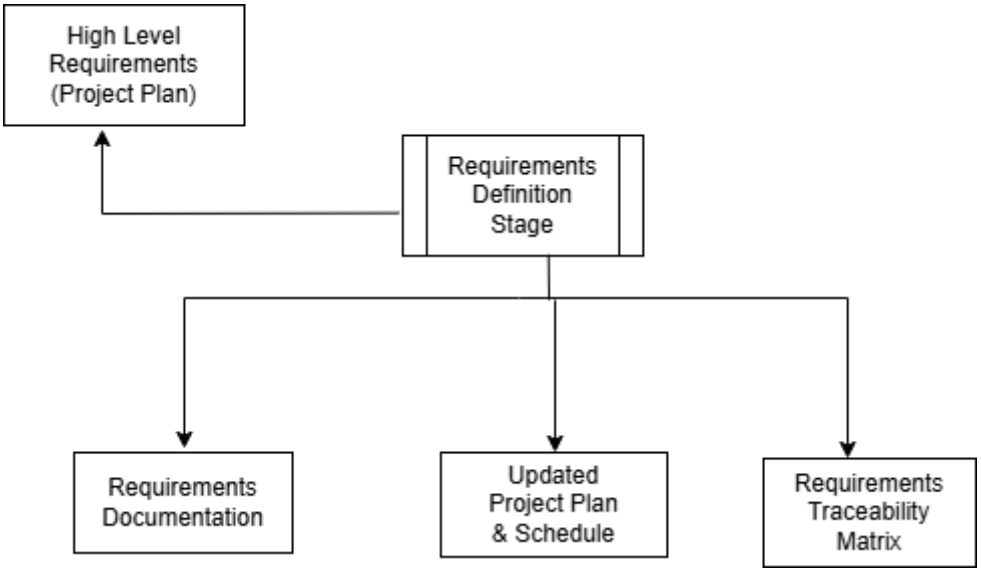


Fig no. 5.2 Requirements Gathering stage

These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are not included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term requirements traceability.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

Feasibility study is all about identification of problems in a project, number of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.

Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

3.3 ANALYSIS STAGE:

The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.

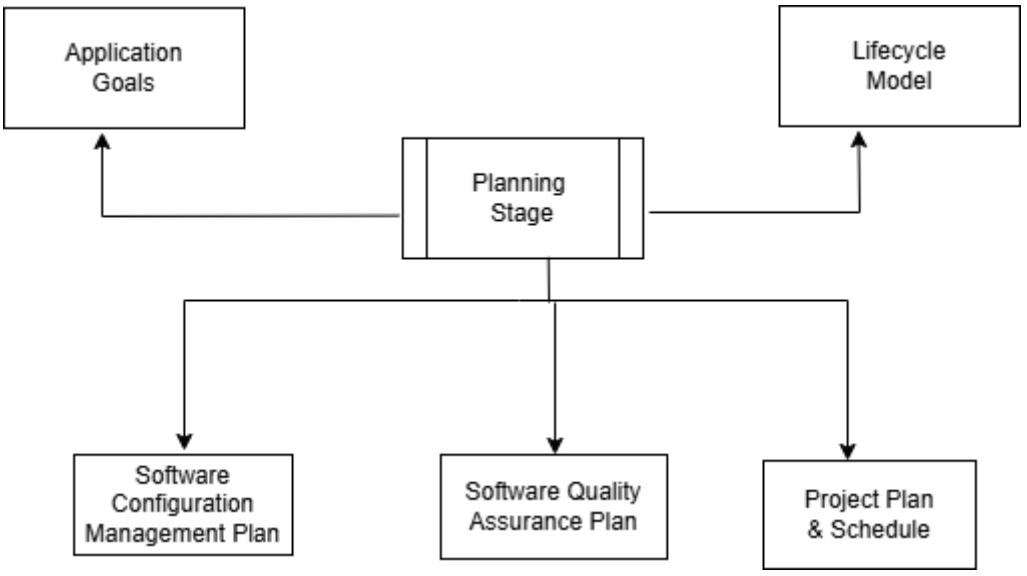


Fig no. 5.3 Analysis stage

The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage and high level estimates of effort for the out stages.

3.4 DESIGNING STAGE:

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.

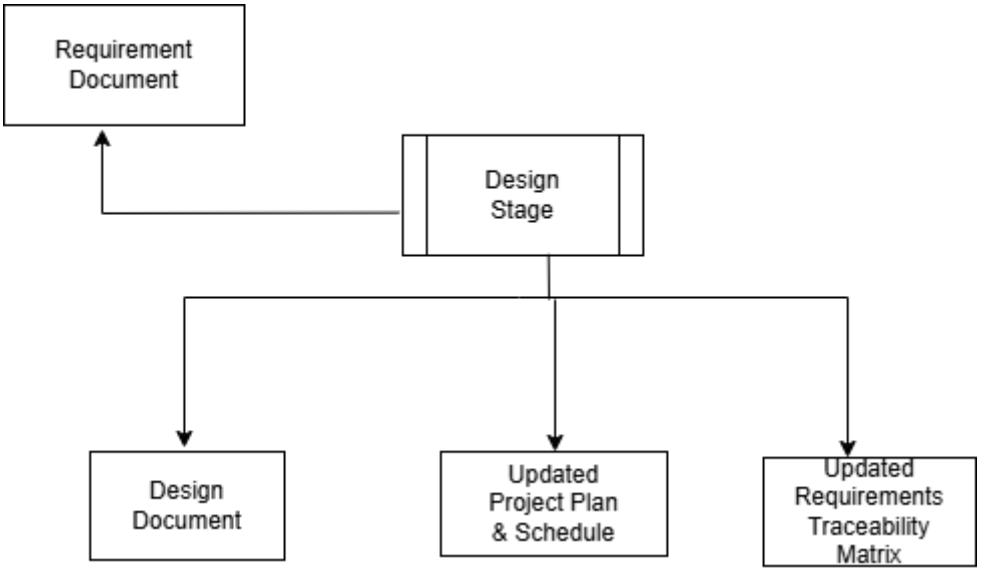


Fig no. 5.4 Designing stage

Software artifacts include but are not limited to menus, dialogs, data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.

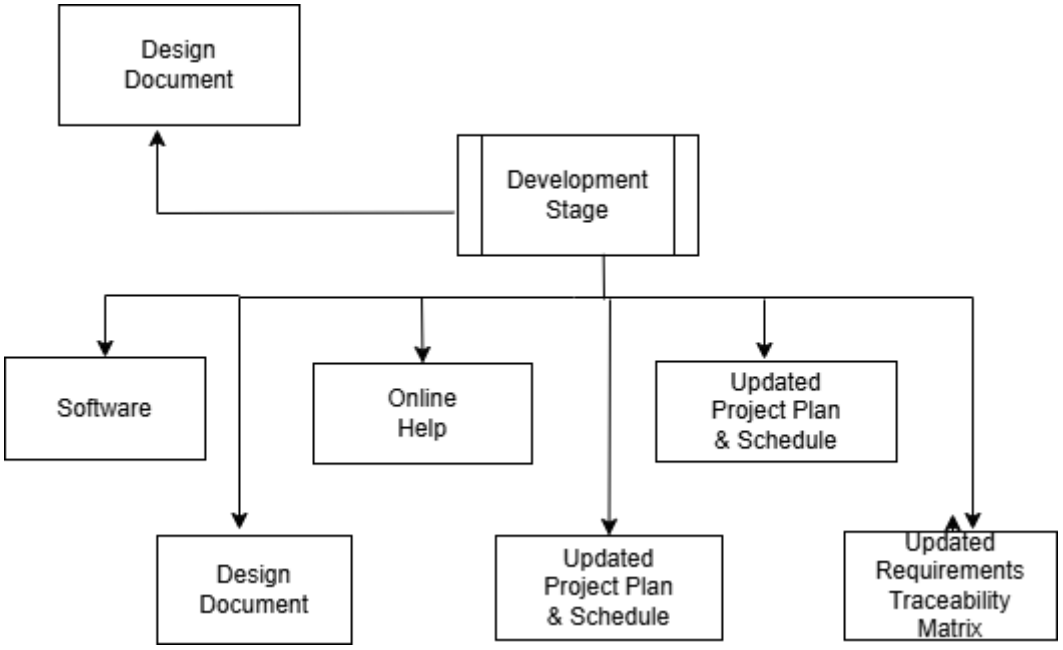


Fig no. 5.5 Coding stage

3.5 . Integration & Test Stage:

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan**.**

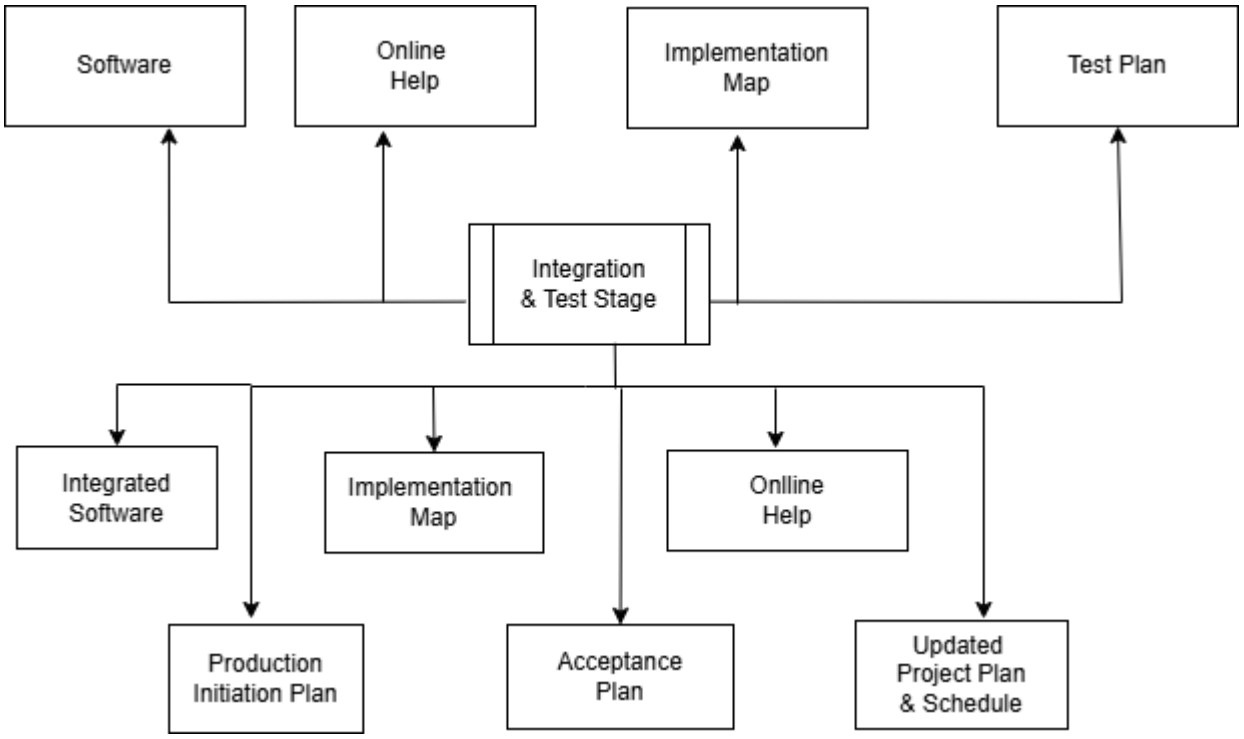


Fig no. 5.6 Integration and Testing Stage

3.6 . Installation & Acceptance Test

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.

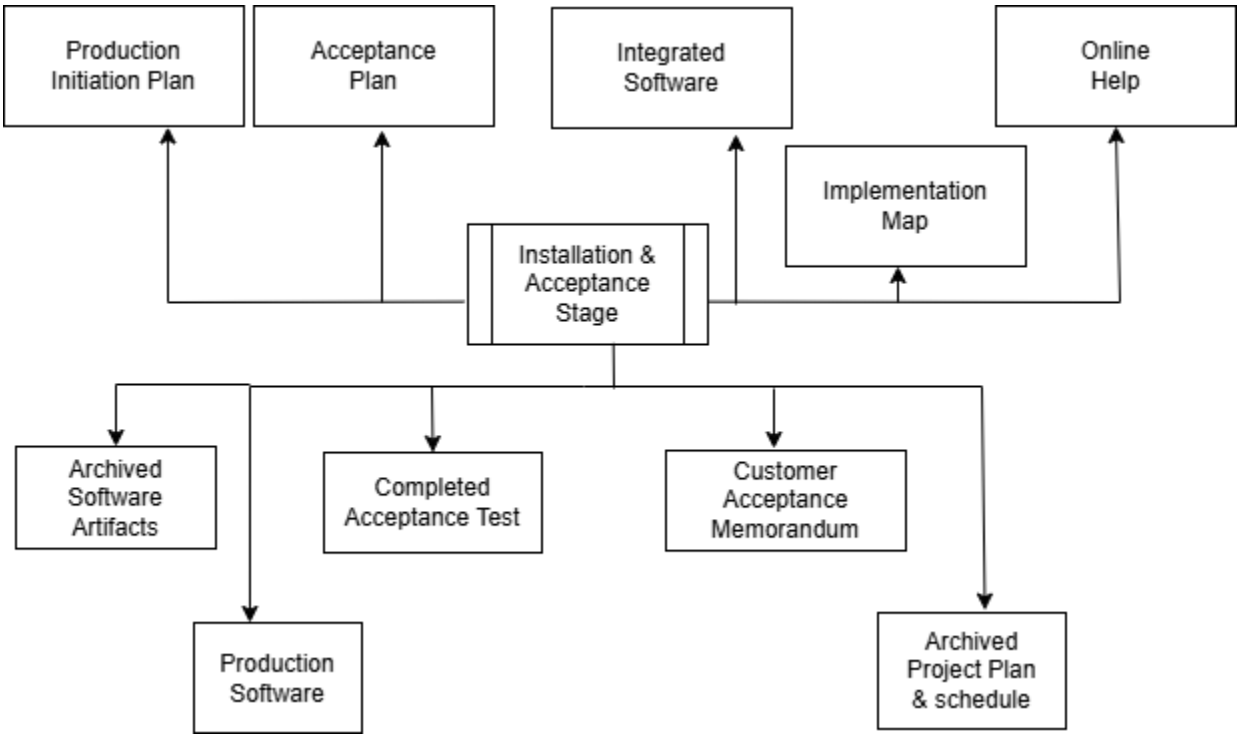


Fig no. 5.7 Installation

3.7 . MAINTENANCE:

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category.

4. SYSTEM DESIGN AND UML DIAGRAMS

4.1 SYSTEM ARCHITECTURE:

The purpose of the design phase is to arrange an answer of the matter such as by the necessity document. This part is that the opening moves in moving the matter domain to the answer domain. The design phase satisfies the requirements of the system. The design of a system is probably the foremost crucial issue warm heartedness the standard of the software package. It's a serious impact on the later part, notably testing and maintenance.

The output of this part is that the style of the document. This document is analogous to a blueprint of answer and is employed later throughout implementation, testing and maintenance. The design activity is commonly divided into 2 separate phases System Design and Detailed Design.

System Design conjointly referred to as top-ranking style aims to spot the modules that ought to be within the system, the specifications of those modules, and the way them move with one another to supply the specified results.

At the top of the system style all the main knowledge structures, file formats, output formats, and also the major modules within the system and their specifications square measure set. System design is that the method or art of process the design, components, modules, interfaces, and knowledge for a system to satisfy such as needs. Users will read it because the application of systems theory to development.

Detailed Design, the inner logic of every of the modules laid out in system design is determined. Throughout this part, the small print of the info of a module square measure sometimes laid out in a high-level style description language that is freelance of the target language within which the software package can eventually be enforced.

In system design the main target is on distinguishing the modules, whereas throughout careful style the main target is on planning the logic for every of the modules.

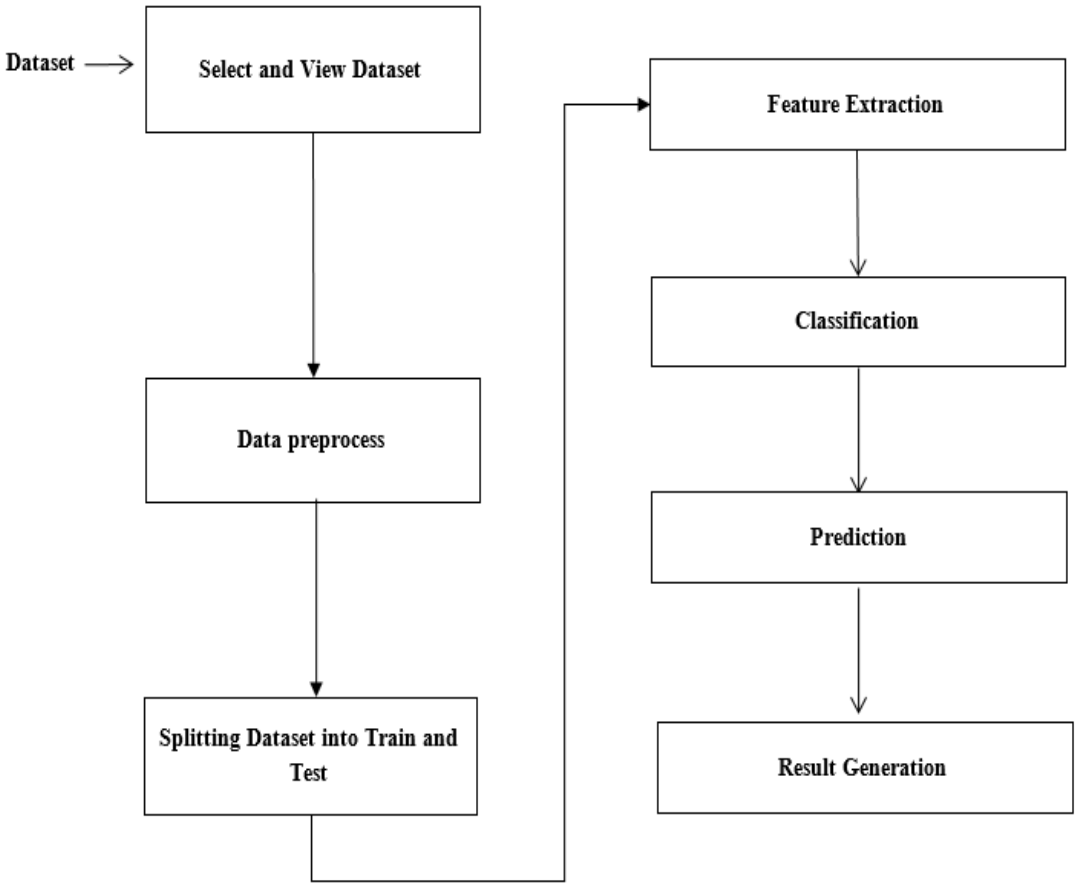


Figure 6.1.1: Architecture diagram

UML DIAGRAMS:

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling 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 Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling 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:

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

4.2 CLASS DIAGRAM:

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities.

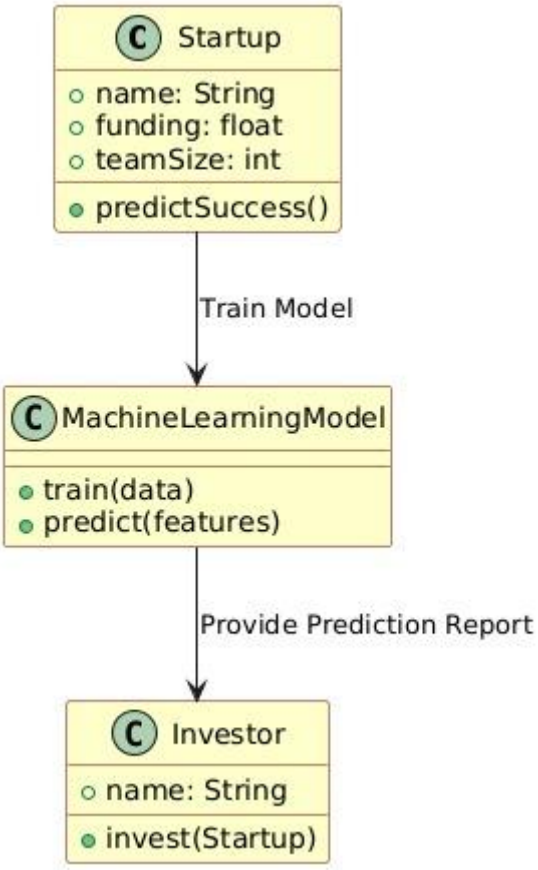


Figure 6.2: Class Diagram

4.3 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.

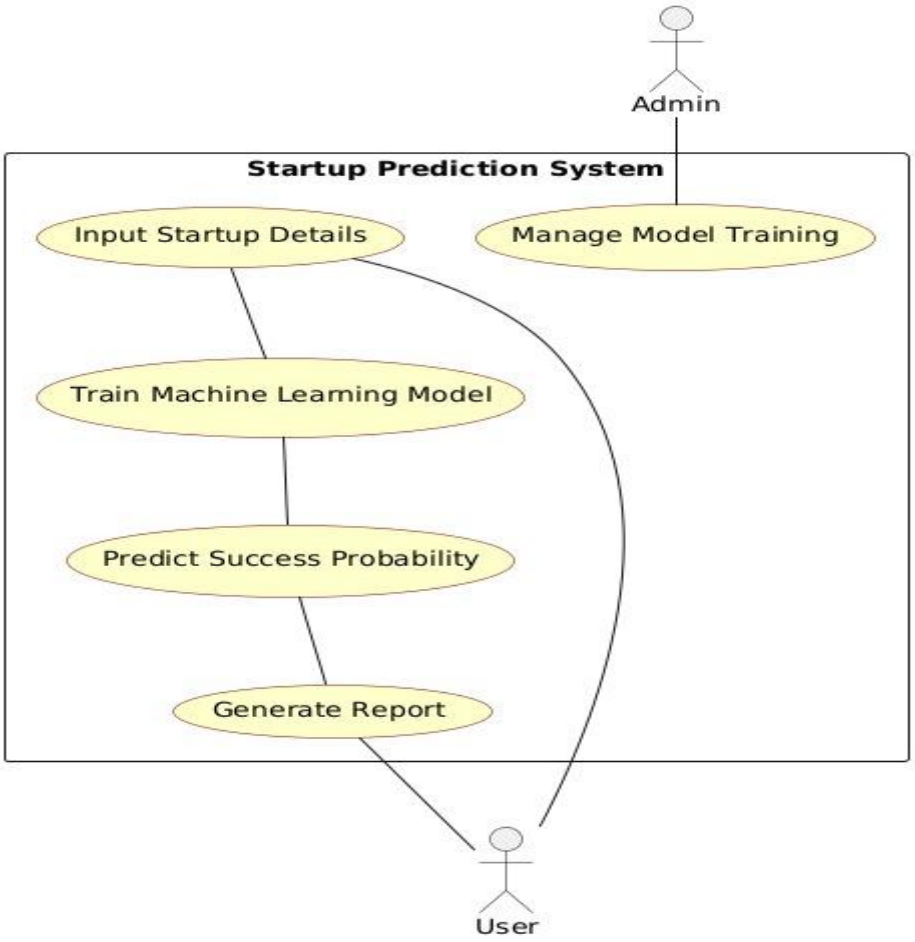


Figure 6.3: Use Case Diagram

4.4 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. A sequence diagram shows, as parallel vertical lines ("lifelines"), different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

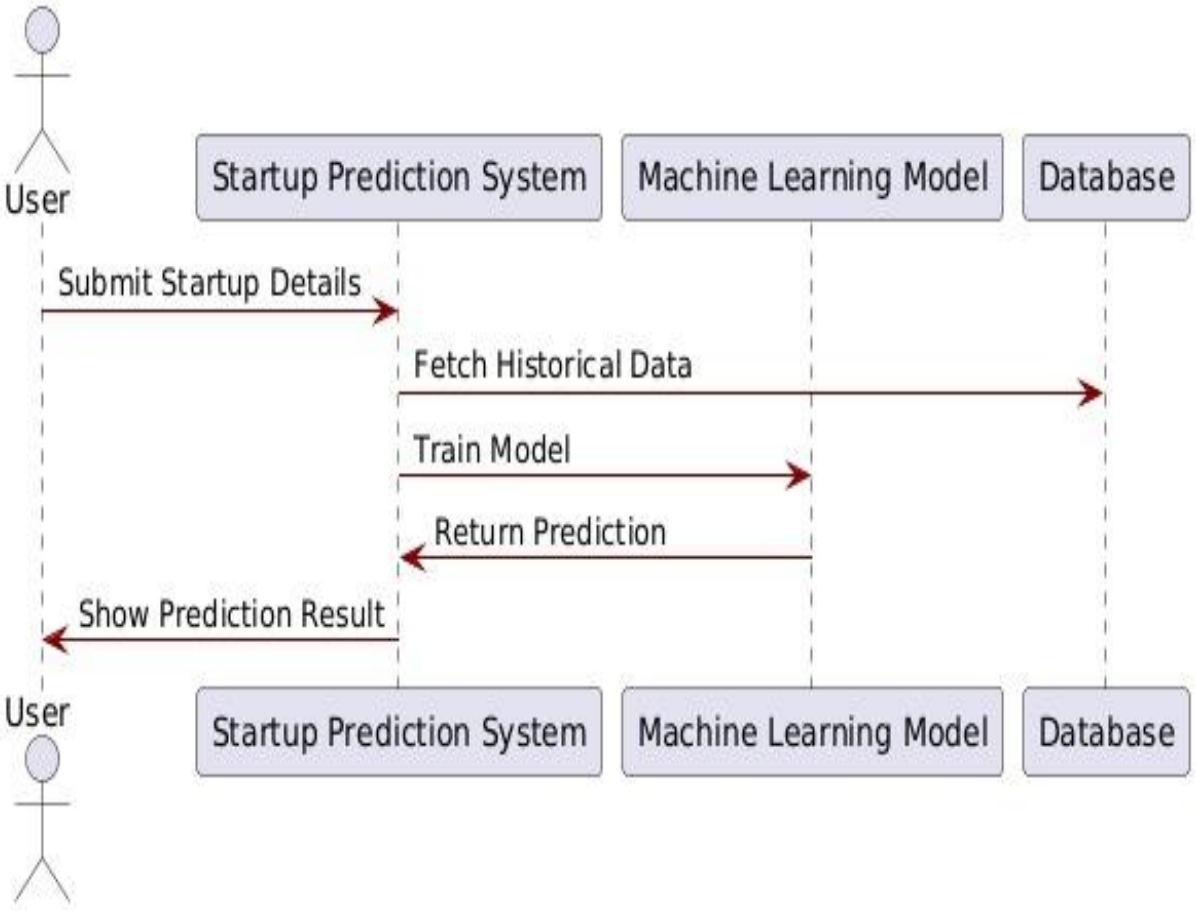


Figure 6.4: Sequence Diagram

4.5 ACTIVITY DIAGRAM:

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

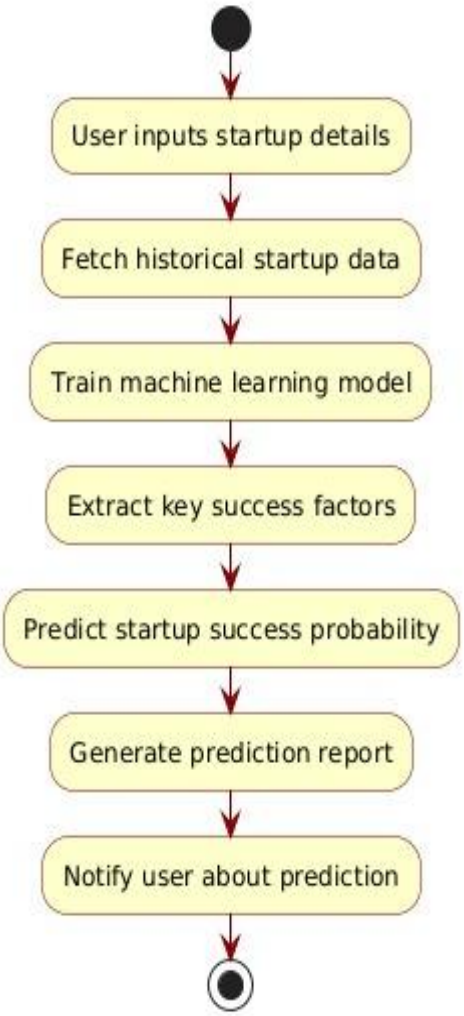


Figure 6.5: Activity Diagram

7. SOFTWARE ENVIRONMENT

7.1 What is Python?

Python is currently the most widely used multi-purpose, high-level programming language.Python allows programming in Object-Oriented and Procedural paradigms. Python programs are generally smaller than those in other programming languages like Java.Programmers have to type relatively less, and the indentation requirement of the language makes Python code more readable.Python is used by almost all tech giants, including Google, Amazon, Facebook, Instagram, Dropbox, and UberThe biggest strength of Python is huge collection of standard libraries which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Opencv, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

ADVANTAGES OF PYTHON

Let’s see how Python dominates over other languages.

1.Extensive Libraries

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, CGI, email, image manipulation, and more. So, we don’t have to write the complete code for that manully.

2. Extensible

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3.Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

4.Improved Productivity

The language's simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5.IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet of Things. This is a way to connect the language with the real world.

6.Simple and Easy

When working with Java, you may have to create a class to print 'Hello World'. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

7. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. These further aids the readability of the code.

8. Object-Oriented

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

9. Free and Open-Source

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

11. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

ADVANTAGES OF PYTHON OVER OTHER LANGUAGES

1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

3. **Python is for Everyone**

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations.

DISADVANTAGES OF PYTHON:

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

1. **Speed Limitations**

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2. **Weak in Mobile Computing and Browsers**

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.

The reason it is not so famous despite the existence of Brython is that it isn't that secure.

3. **Design Restrictions**

As you know, Python is dynamically-typed. This means that you don't need to declare the type of variable while writing the code. It uses duck-typing. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

4. **Underdeveloped Database Access Layers**

Compared to more widely used technologies like JDBC (Java Database Connectivity) and ODBC (Open Database Connectivity), Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

5. **Simple**

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

7.2 HISTORY OF PYTHON

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners[1], Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it. "Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So, I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

PYTHON DEVELOPMENT STEPS:

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of lists, dict, str and others. It was also object oriented and had a module system. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released.

Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it."Some changes in Python 7.3:

- Print is now a function.
- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g., a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
- There is only one integer type left, i.e., int. long is int as well.
- The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.
- Text Vs. Data Instead of Unicode Vs. 8-bit

Purpose

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- 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 also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

7.3 MODULES IN PROJECTS

Module 1: Data Collection and Preprocessing

- Function to collect historical rainfall data from reliable sources or APIs.
- Data cleaning and preprocessing functions to handle missing values and anomalies.
- Feature engineering to extract relevant features like season, temperature, humidity, etc.
- Splitting the dataset into training and testing sets for model evaluation.

Module 2: Machine Learning Models

- To implement traditional machine learning algorithms (e.g., Linear Regression,
- Decision Trees, Random Forests) for rainfall prediction.
- Hyperparameter tuning functions to optimize the model's performance.
- Evaluation metrics functions (e.g., Mean Squared Error, R-squared) to assess the model's
- Accuracy.
- Saving and loading trained ML models for future predictions.

Module 3: Deep Learning Models

- Function to build deep learning architectures (e.g., LSTM, GRU, CNN) for rainfall prediction.
- Training functions to train deep learning models on the preprocessed dataset.
- Early stopping and model checkpointing to prevent overfitting and save the best model

during training.

- Visualization functions to analyze the model's performance and loss over epochs.

Module 4: Integration and Comparison

- Function to combine the outputs of machine learning and deep learning models for ensemble approach an.
- Performance comparison functions to compare the predictions of different models.
- Choosing the best model for final rainfall prediction based on evaluation metrics.

Module 5: Real-Time Prediction

- Function to integrate the chosen model into a real-time prediction system.
- Collecting and preprocessing real-time weather data.
- Utilizing the trained model to predict rainfall for the given weather conditions.
- Displaying the predictions and confidence intervals for user interpretation.

Module 6: Model Updates and Maintenance

- Function to periodically retrain and update the machine learning and deep learning models with new data.
- Implementing version control for models and managing model deployments.
- Monitoring the model's performance over time and addressing any issues that arise.

TENSOR FLOW

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

NumPy

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It contains various features including these important one. Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data.

Pandas:

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib:

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object-oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn:

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- 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 also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

7.4 Install Python Step-by-Step in Windows and Mac

Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

How to Install Python on Windows and Mac

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e., operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So, the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here. The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

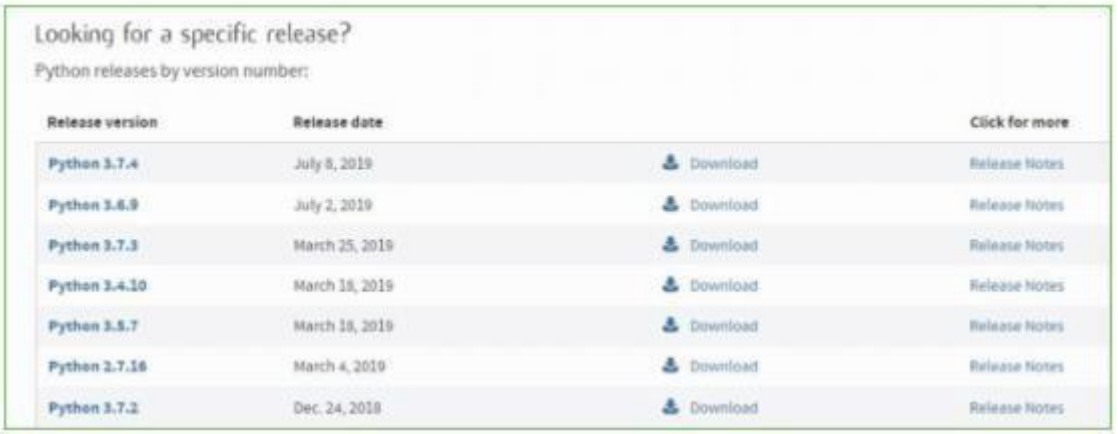
Download the Correct version into the system

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: https://www.python.org

Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4



Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.

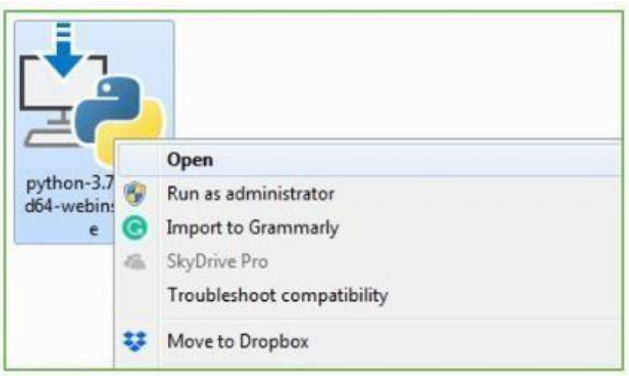
- To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
- To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e., Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.



Step 2: Before you click on Install Now, make sure to put a tick on Add Python 3.7 to PATH.

Step 3: Click on Install NOW After the installation is successful. Click on Close.

With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

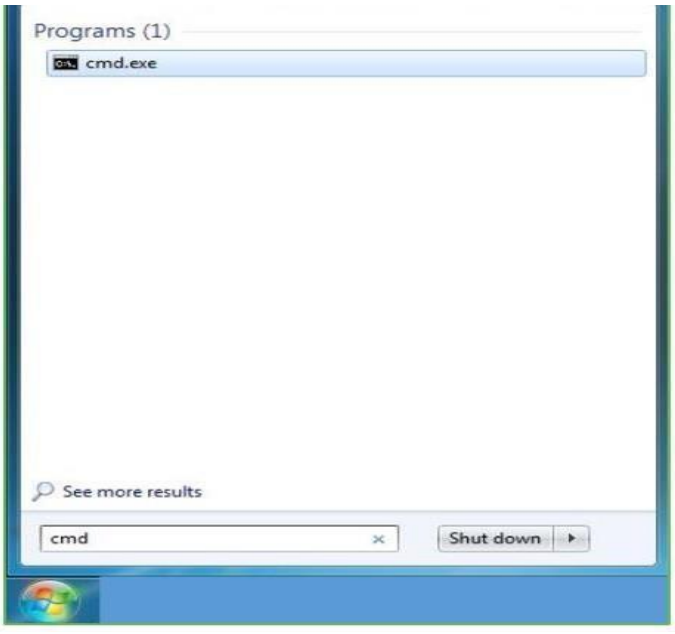
Note: The installation process might take a

couple of minutes. Verify the Python

Installation

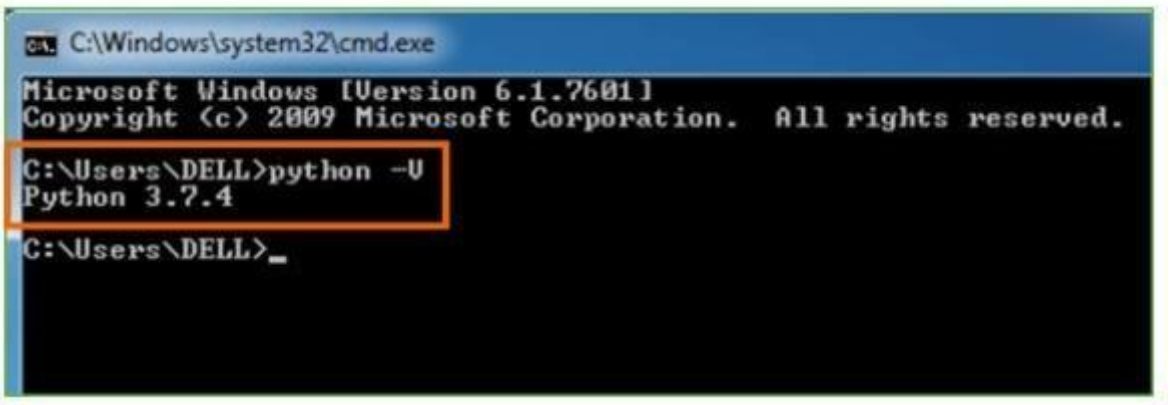
Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.



Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type python –V and press Enter.



Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

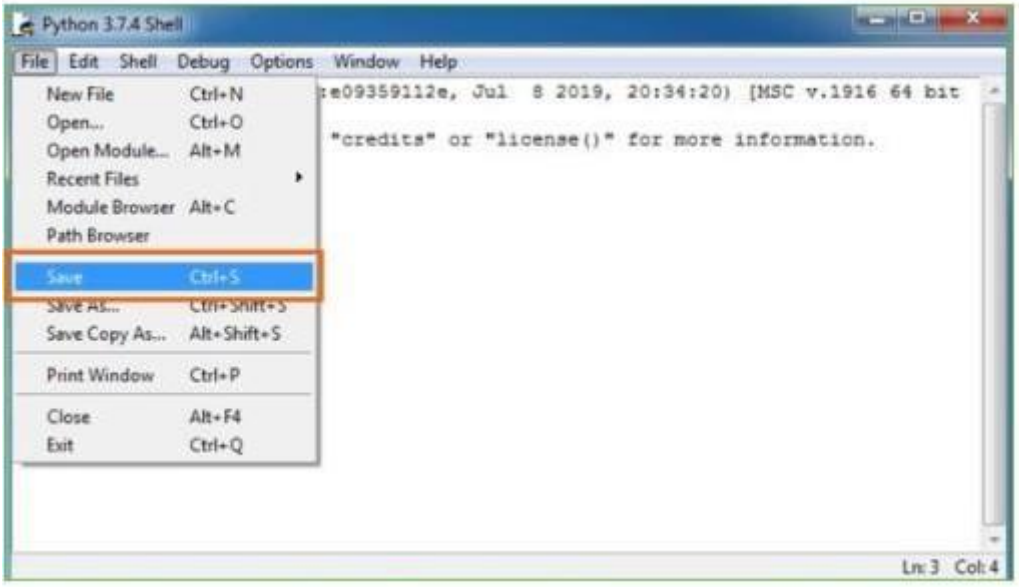
Check how the Python IDLE works

Step 1: Click on Start

Step 2: In the Windows Run command, type “python idle”.

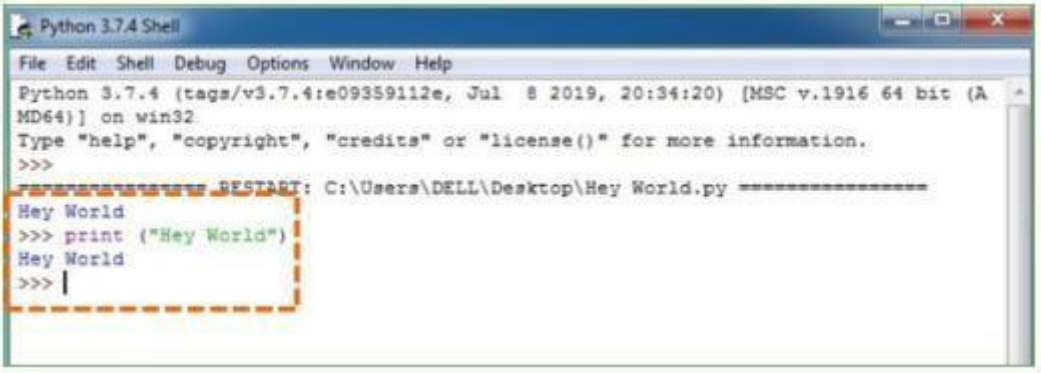
Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. Click on File > Click on Save



Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g., enter print ("Hey World") and Press Enter.



You will see that the command given is launched. With this, we end our tutorial on how to install Python. You have learned how to download python for windows into your respective operating system.

Note: Unlike Java, Python does not need semicolons at the end of the statements otherwise it won't work.

8.MACHINE LEARNING

8. 1. WHAT IS MACHINE LEARNING :

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models tunable parameters that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain.Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

Categories Of Machine Leaning :-

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning. Supervised Learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into *classification* tasks and *regression* tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section. Unsupervised Learning involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as dimensionality reduction. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data.

8.2 Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven't surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, "to make decisions, based on data, with efficiency and scale".

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programing logic, in the problems that cannot be programmed inherently. The fact is that we can't do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

Challenges in Machines Learning :-

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are

Quality of data : Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

Time-Consuming task : Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

Lack of specialist persons : As ML technology is still in its infancy stage, availability of expert resources is a tough job.

No clear objective for formulating business problems : Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

Issue of overfitting & underfitting : If the model is overfitting or underfitting, it cannot be represented well for the problem.

Curse of dimensionality : Another challenge ML model faces is too many features of data points. This can be a real hindrance.

Difficulty in deployment : Complexity of the ML model makes it quite difficult to be deployed in real life.

8.3 APPLICATIONS OF MACHINES LEARNING :-

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML

- Emotion analysis
- Sentiment analysis
- Error detection and prevention
- Weather forecasting and prediction
- Stock market analysis and forecasting
- Speech synthesis
- Speech recognition
- Customer segmentation
- Object recognition

HOW TO START LEARNING MACHINE LEARNING?

Arthur Samuel coined the term **"**Machine Learning" in 1959 and defined it as a "Field of study that gives computers the capability to learn without being explicitly programmed".

And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to Indeed, Machine Learning Engineer Is The Best Job of 2019 with a 344% growth and an average base salary of $146,085 per year.

But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So this article deals with the Basics of Machine Learning and also the path you can follow to eventually become a full-fledged Machine Learning Engineer. Now let's get started!!!

This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

Step 1 – Understand the Prerequisites

In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python. And if you don't know these, never fear! You don't need a Ph.D. degree in these topics to get started but you do need a basic understanding.

(a). Learn Linear Algebra and Multivariate Calculus:

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily focused on maths as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is very important as you will have to implement many ML algorithms from scratch.

(b). Learn Statistics:

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the collection, analysis, and presentation of data. So it is no surprise that you need to learn it!!! Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

(c). Learn Python:

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is Python! While there are other languages you can use for Machine Learning like R, Scala, etc. Python is currently the most popular language for ML. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning such as Keras, TensorFlow, Scikit-learn, etc.

(a). Terminologies of Machine Learning:

- **Model :** A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.
- **Feature :** A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.
- **Target (Label) :** A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.
- **Training :** The idea is to give a set of inputs(features) and it's expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
- **Prediction :** Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

(b). Types of Machine Learning:

- **Supervised Learning :** This involves learning from a training dataset with labeled data using classification and regression models. This learning process continues until the required level of performance is achieved.
- **Unsupervised Learning :** This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.

- **Semi-supervised Learning :**This involves using unlabelled data like Unsupervised Learning with a small amount of labeled data. Using labeled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.
- **Reinforcement Learning :** This involves learning optimal actions through trial and error. So the next action is decided by learning behaviors that are based on the current state and that will maximize the reward in the future.

8.4. Advantages of Machine learning :-

1. Easily identifies trends and patterns

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation)

With ML, you don't need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus softwares; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

3. Continuous Improvement

As ML algorithms gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

4. Handling multi-dimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

5. Wide Applications

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

8.5 Disadvantages of Machine Learning :-

1. Data Acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

2. Time and Resources

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

9. SYSTEM REQUIREMENTS & SPECIFICATIONS

9.1. Software Requirements

The software requirements specify the use of all required software products like data management system. The required software product specifies the numbers and version. Each interface specifies the purpose of the interfacing software as related to this software product.

- Technologies: Python
- Operating System: Microsoft Windows, Linux or Mac any version

9.2Hardware Requirements

The hardware requirement specifies each interface of the software elements and the hardware elements of the system. These hardware requirements include configuration characteristics.

- Processer : Pentium IV 2.4 GHz.
- Hard Disk : 40 GB.
- Floppy Drive : 1.44 Mb.
- Monitor : 15 VGA Colour.
- Mouse : Logitech.
- Ram : 512 Mb.

9.3.Feasibility Study

The feasibility of the project is analyzed 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

Economical 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.

10.FUNCTIONAL REQUIREMENTS

10.1. Output Design

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provides a permanent copy of the results for later consultation. The various types of outputs in general are:

- External Outputs, whose destination is outside the organization
- Internal Outputs whose destination is within organization and they are the
- User's main interface with the computer.
- Operational outputs whose use is purely within the computer department.
- Interface outputs, which involve the user in communicating directly.

Output Definition

The outputs should be defined in terms of the following points:

- Type of the output
- Content of the output
- Format of the output
- Location of the output
- Frequency of the output
- Volume of the output
- Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

10.2 Input Design

Input design is a part of overall system design. The main objective during the input design is as given below:

- To produce a cost-effective method of input.
- To achieve the highest possible level of accuracy.
- To ensure that the input is acceptable and understood by the user.

Input Stages

The main input stages can be listed as below:

- Data recording
- Data transcription
- Data conversion
- Data verification
- Data control
- Data transmission
- Data validation
- Data correction

Input Types

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

- External inputs, which are prime inputs for the system.
- Internal inputs, which are user communications with the system.
- Operational, which are computer department's communications to the system?
- Interactive, which are inputs entered during a dialogue.

Input Media

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

- Type of input
- Flexibility of format
- Speed
- Accuracy
- Verification methods
- Rejection rates
- Ease of correction
- Storage and handling requirements
- Security
- Easy to use
- Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

Error Avoidance

At this stage care is to be taken to ensure that input data remains accurate form the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

Error Detection

Even though every effort is made to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

Data Validation

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

10.3 User Interface Design

It is essential to consult the system users and discuss their needs while designing the user interface:

User Interface Systems Can Be Broadly Classified As:

- User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.Computer initiated interfaces
- In the computer-initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

User Initiated Interfaces

User initiated interfaces fall into two approximate classes:

- Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.
- Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms-oriented interface is chosen because it is the best choice.

Computer-Initiated Interfaces

The following computer – initiated interfaces were used:

- The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
- Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

Error Message Design

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

This application must be able to produce output at different modules for different inputs.

10.4 Performance Requirements:

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is

because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

- The system should be able to interface with the existing system
- The system should be accurate
- The system should be better than the existing system
- The existing system is completely dependent on the user to perform all the duties.

11. SOURCE CODE IMPLEMENTATION

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys
def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'startup.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)
if __name__ == '__main__':
    main()
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 1,
   "id": "35e424e1",
   "metadata": {},
   "outputs": [],
   "source": [
    "# age first funding year is the age of the company in years since it got first funding. similar for age last funding year.\n",
    "\n",
```

"# milestonefor any startup is a tracking mark for startups. Just like a milestone on the side of a road marks how\n",
"#far you've gone, a milestone in startups tracks progress as an startup grow and implement their plan.\n",
"\n",
"# relationships- it says how many relationship does a startup have. For example a start up can have \n",
"#relationships with accountants, investors, vendors, mentors, etc.\n",
"\n",
"#funding rounds for an startup can be better understood here: https://www.forbes.com/sites/alejandr\n",
"#ocremades/2018/12/26/how-funding-rounds-work-for-startups/?sh=57858cc73866"
]
},
{
"cell_type": "markdown",
"id": "dfaee4ec",
"metadata": {},
"source": [
"<divstyle=\"padding:25px;color:black;margin:0;font-size:250%;text-align:center;display:fill;border-radius:10px;background-color:#D4C8BA;overflow:hidden;font-weight:500;font-family:magra\">Imports "
]
},
{
"cell_type": "code",
"execution_count": 2,
"id": "6f6943a6",
"metadata": {},
"outputs": [],
"source": [
"import pandas as pd\n",
"import seaborn as sns\n",
"import matplotlib.pyplot as plt\n",
"\n",

```
   "from sklearn.linear_model import LogisticRegression\n",
   "from sklearn.tree import DecisionTreeClassifier\n",
   "from sklearn.ensemble import RandomForestClassifier\n",
   "from sklearn.svm import SVC\n",
   "from sklearn.ensemble import AdaBoostClassifier\n",
   "\n",
   "from sklearn.model_selection import train_test_split\n",
   "from sklearn.model_selection import GridSearchCV\n",
   "from sklearn.preprocessing import LabelEncoder\n",
   "from sklearn.metrics import confusion_matrix,classification_report,recall_score,precision_score,accuracy_score\n",
   "\n",

   "import warnings\n",
   "warnings.filterwarnings('ignore')"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 3,
  "id": "ffb5f7f6",
  "metadata": {},
  "outputs": [],
  "source": [
   "data=pd.read_csv('startup dataog.csv')"
  ]
"""
Django settings for startup project.
Generated by 'django-admin startproject' using Django 4.2.
For more information on this file, see
https://docs.djangoproject.com/en/4.2/topics/settings/
For the full list of settings and their values, see
https://docs.djangoproject.com/en/4.2/ref/settings/
"""
from pathlib import Path
```

```
import os
# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent
# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-v(ly9nj$apj0kh6-jpx&i9qnl78e-cdp8_+z+y549m$2*0891f'
# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True
ALLOWED_HOSTS = []
# Application definition
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'Home'
]

IDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'startup.urls'
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': ['templates'],
```

```
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'startup.wsgi.application'
# Database
# https://docs.djangoproject.com/en/4.2/ref/settings/#databases
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }

}
# Password validation
# https://docs.djangoproject.com/en/4.2/ref/settings/#auth-password-validators
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
```

```
    },
]
# Internationalization
# https://docs.djangoproject.com/en/4.2/topics/i18n/
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_TZ = True
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.2/howto/static-files/

STATIC_URL = 'static/'
STATICFILES_DIRS = [
    os.path.join(BASE_DIR,'static')
    ]
# Default primary key field type
# https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

12.SYSTEM TESTING

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property function as a unit. The test data should be chosen such that it passed through all possible condition.

12.1. SYSTEM TESTING:

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to user the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

12.2. MODULE TESTING:

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

12.3. INTEGRATION TESTING:

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

12.4. ACCEPTANCE TESTING:

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

13. RESULTS

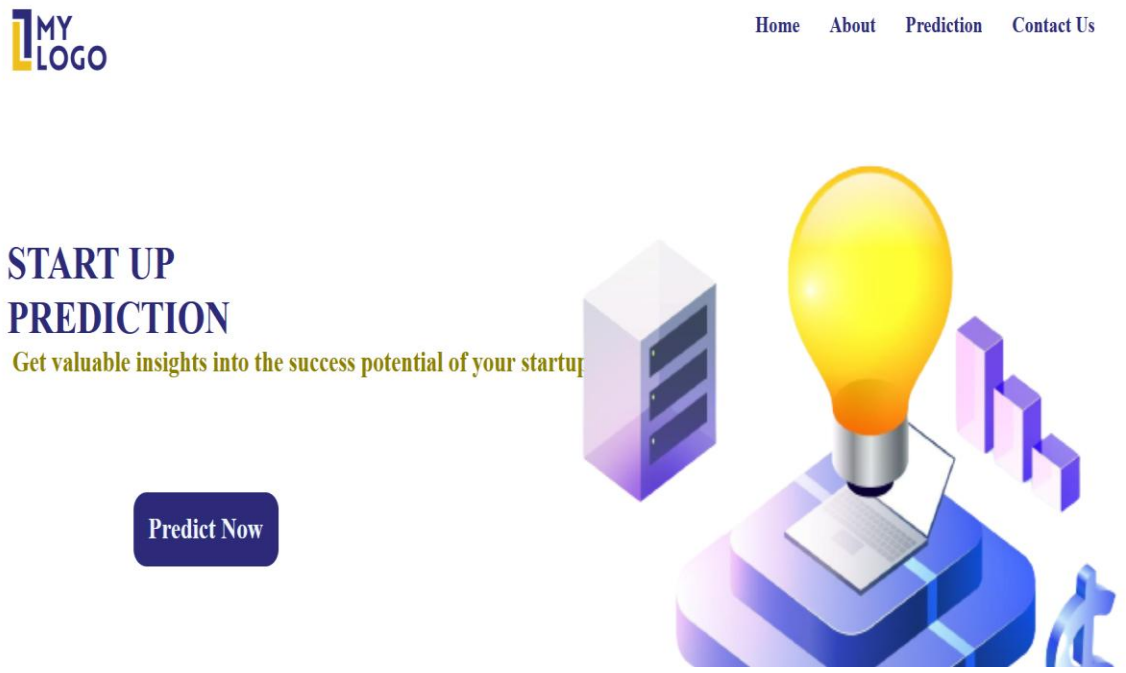


Figure 13.1: Home Page of StartUp Success Prediction

Figure 13.2: About Page of StartUp Success Prediction

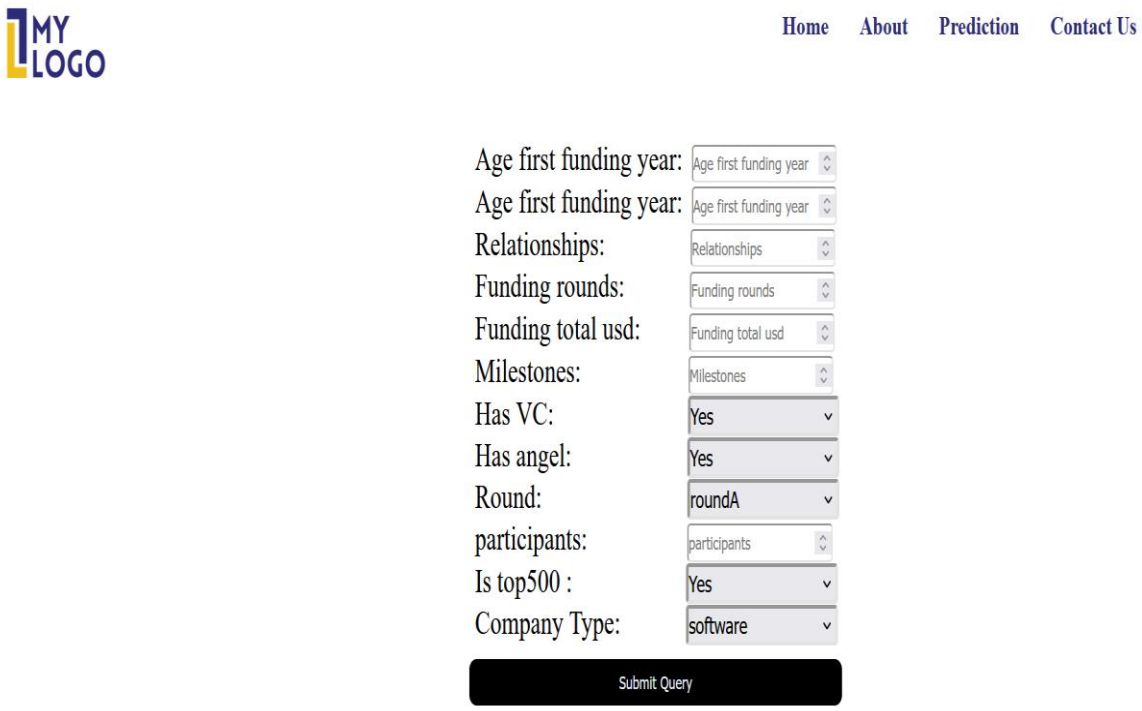


Figure 13.3: Startup success Prediction web page

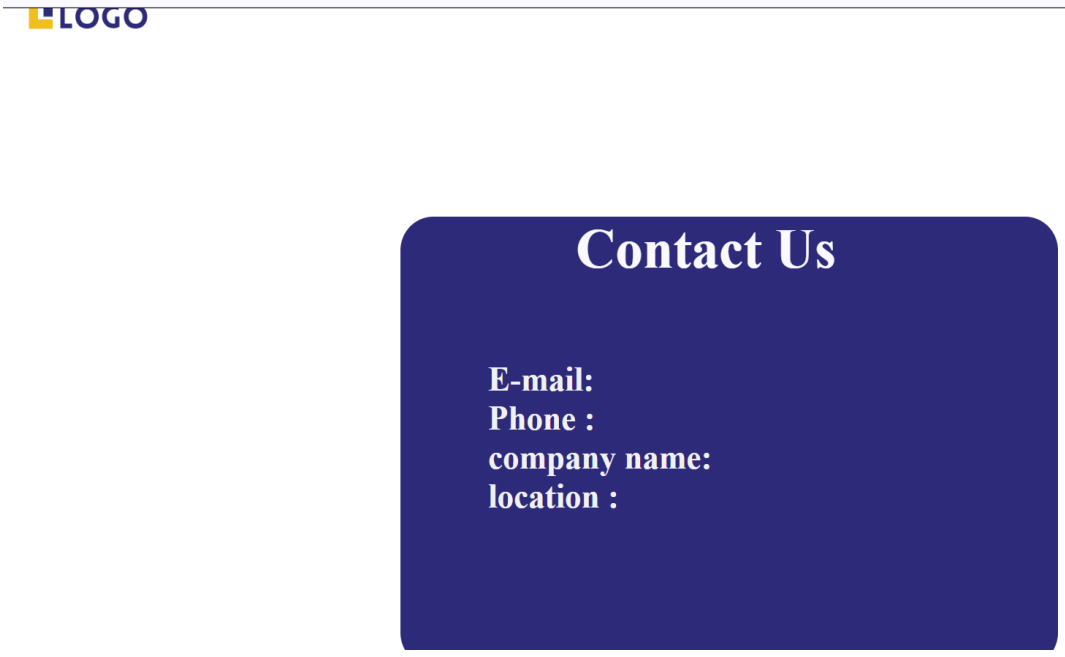


Figure 13.4: StartUp Success Prediction Contact Page

14. CONCLUSION AND FUTURE SCOPE

CONCLUSION

Predicting startup success can help investors, entrepreneurs, and decision-makers allocate resources more effectively and make informed strategic decisions. As the field of machine learning continues to advance, it will undoubtedly bring new opportunities and challenges for predicting startup success, leading to a more dynamic and informed startup ecosystem.

FUTURE ENHANCEMENTS:

Machine learning algorithms can analyse vast amounts of historical startup data to identify patterns that correlate with success or failure. AI can assess factors such as market timing, founder experience, and financial metrics to provide data-driven probability scores for startup success.

Unique and unpredictable factors: Startups operate in dynamic and rapidly changing environments. Factors such as market trends, competition.

15. REFERENCES

[1] Malhar Bangdiwala, Yashvi Mehta, Smrithi Agrawal, Sunil Ghane , “Predicting Success Rate of Startups using some Machine Learning Alogorithms” in *IEEE Access*, DOI:10.1109/ASIANCON55314.2022.9908921

[2] Daniel Bennet, Shella Aulla Anjani, Ora Pertiwi Daeli, Dedi Martano, “Predictive Analysis of startup Ecosytems with Random Forest Techniques" ,2024

[3] Athanasios Davalas, “Scoring Card Methodologies for Startup Evaluation:A Machine learning based Approach,DOI: 10.46609/IJSSER.2023.v08i12.013

[4] Alsolaim M. Barriers to Survival for Small Start-up Businesses in Saudi Arabia. PhD diss., University of Brighton, 2019.

[5] Tomy S, Pardede E. From Uncertainties to Successful Start Ups: A Data Analytic Approach to Predict Success in Technological Entrepreneurship. Sustainability. 2018; 10(3): 602. doi:10.3390 su10030602.

[6] Kim B, Kim H, Jeon Y. Critical Success Factors of a Design Startup Business. Sustainability. 2018; 10(9): 2981. doi:10. 3390/su10092981.B. Branco,

P. Abreu, A. S. Gomes, M. S. C. Almeida, J. T. Ascensão, and P. Bizarro,

[7] Cohen B, Amorós JE, Lundy L. The Generative Potential of Emerging Technology to Support Startups and New Ecosystems. ScienceDirect. 2017; 60(6): 714–745. doi:10. 1016/j.bushor.2017.06.004.

[8] Kuzmianok D. Socioeconomic Impact of Startup Companies: The Republic of Belarus - Prospects and Challenges of Startup Ecosystem.2016. Available online: https://monami.hsmittweida.de/frontdoor/index/index/docId/8383

[9] Xu J, Peng B, Cornelissen J. Modelling the Network Economy: A Population Ecology Perspective on Network Dynamics. Technovation. 2021; 102: 102212. doi:10.1016/ j.technovation.2020.102212.

[10] Li J. Prediction of the Success of Startup Companies Based on Support Vector Machine and Random Forest. 2020 2nd International Workshop on Artificial Intelligence and Education. 2020. doi:10.1145/3447490.3447492.

[11] Żbikowski K, Antosiuk P. A Machine Learning, Bias-Free Approach for Predicting Business Success Using Crunchbase Data. Information Processing & Management. 2021; 58(4): 102555. doi:10.1016/j.ipm.2021.102555.

[12] Pan C, Gao Y, Luo Y. Machine Learning Prediction of Companies’ Business Success. CS229: Machine Learning. Fall 2018. Stanford University, CA.

[13] Shah V. Predicting the Success of a Startup Company. support.sas.com. 2019. Available online: https://support.sas. com/resources/papers/proceedings19/3878-2019.pdf (accessed Nov. 27, 2023).

[14] Veloso F. Predicting Startup Success in the US. The University of North Carolina at Charlotte, 2020.
[15] Zoayed MT, Arshe S, Rahman F. Startup Success Prediction Using Classification Algorithms. Jun. 2022.
[16] Vasquez E, Santisteban J, Mauricio D. Predicting the Success of a Startup in Information Technology Through Machine Learning. International Journal of Information Technology and Web Engineering. 2023; 18(1): 1–17. doi:10.4018/ijitwe. 323657.
[17] Fidder D. Finding the Most Significant Predictors of Startup Success with Machine Learning. Eindhoven University of Technology. Jan. 2022.
[18] Ghee WY. An Application of Timmons Model in the Mini Entrepreneurial Logistics Project. Advances in Social Sciences Research Journal. 2018; 5(10). doi:10.14738/assrj. 510.5541.
[19] S. Social Entrepreneurship from the Perspective of Opportunity: Integration Analysis Based on Timmons Process Model. Journal of Human Resource and Sustainability Studies. 2019; 07(03): 438–461. doi:10.4236/jhrss.2019. 73029.
[20] Shang Z. The Research of Financial Forecasting and Valuation Models. Proceedings of the 2021 International Conference on Enterprise Management and Economic Development (ICEMED 2021). 2021. doi:10.2991/amber.k. 210601.012.