PREDICTION OF HEART DISEASE USING MACHINE LEARNING ALGORITHMS

A Project Report submitted to

The Principal, Sree Vidyanikethan Engineering College
(AUTONOMOUS)

in partial fulfilment of the requirements for
the award of the degree of

MASTER OF COMPUTER APPLICATIONS

By

BUSETTI SIVAIAH
[21121F0016]

Under the Guidance of

Dr. M. Sunil Kumar
Professor & Head of MCA

Department of MCA

SREE VIDYANIKETHAN ENGINEERING COLLEGE
(AUTONOMOUS)
(Affiliated to JNTUA, Anantapuramu)
Sree Sainath Nagar, A. Rangampet-517 102. Chittoor (Dist), A.P.
2022–2023

Department of Master of Computer Application

SREE VIDYANIKETHAN ENGINEERING COLLEGE

(AUTONOMOUS)

Sree Sainath Nagar, Tirupati – 517102, (2021-2023)

Certificate

This is to certify that the Project report entitled

"PREDICTION OF HEART DISEASE USING MACHINE LEARNING ALGORITHMS"

is the bonafied work done and submitted by

BUSETTI SIVAIAH

(21121F0016)

In the Department of Master of computer Applications, Sree Vidyanikethan Engineering College, A. Rangampet, affiliated to Jawaharlal Nehru Technological University Anantapur, Anantapuramu in partial fulfilment of the requirements for the award of the degree of Master of computer Applications during 2022-2023.

External Guide

Internal Guide

Dr. M. Sunil Kuma
Professor & Head,
Department of MCA

Head of the Department

Dr. M. Sunil Kumar
Professor & Head,
Department of MCA

Master of Computer Applications (MCA) is a four-semester full-time post-graduate Program spread over two years.

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

PEO1. Enrolled or completed higher education/research studies in the core and allied areas of computer science.

PEO2. Successful entrepreneurs and professionally excelled in diverse application skills in the core or allied area of computer science of societal importance.

PEO3. Professionals in industry, academia and organizations with ability to adapt to evolving technologies in the core and allied areas of computer science.

PROGRAM OUTCOMES (POs)

After completion of the program, a successful student will be able to:

PO1. Apply knowledge of computing fundamentals, computing specialization, mathematics, and domain knowledge appropriate for the computing specialization to the abstraction and conceptualization of computing models from defined problems and requirements. (**Computational Knowledge**)

PO2. Identify, formulate, research literature, and solve complex computing problems reaching substantiated conclusions using fundamental principles of mathematics, computing sciences, and relevant domain disciplines. **(Problem Analysis)**

PO3. Design and evaluate solutions for complex computing problems, and design and evaluate systems, components, or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal, and environmental considerations. (**Design /Development of Solutions**)

PO4. Use research-based knowledge and research methods including design of experiments, analysis, and interpretation of data, and synthesis of the information to provide valid conclusions. (**Conduct Investigations of Complex Computing Problems**)

PO5. Create, select, adapt and apply appropriate techniques, resources, and modern computing tools to complex computing activities, with an understanding of the limitations. (**Modern Tool Usage**)

PO6. Understand and commit to professional ethics and cyber regulations, responsibilities, and norms of professional computing practices. (**Professional Ethics**)

PO7. Recognize the need, and have the ability, to engage in independent learning for continual development as a computing professional. (**Life-long Learning**)

PO8. Demonstrate knowledge and understanding of the computing and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments. (**Project management and finance**)

PO9. Communicate effectively with the computing community, and with society at large, about complex computing activities by being able to comprehend and write effective reports, design documentation, make effective presentations, and give and understand clear instructions. (**Communication Efficacy**)

PO10. Understand and assess societal, environmental, health, safety, legal, and cultural issues within local and global contexts, and the consequential responsibilities relevant to professional computing practices. (**Societal and Environmental Concern**)

PO11. Function effectively as an individual and as a member or leader in diverse teams and in multidisciplinary environments. (**Individual and Team Work**)

PO12. Identify a timely opportunity and use innovation to pursue that opportunity to create value and wealth for the betterment of the individual and society at large. (**Innovation and Entrepreneurship)**

PROGRAM SPECIFIC OUTCOMES (PSOs)

On successful completion of M.C.A Program, MCA graduates will be able to:

PSO1. Design, implement and test applications for complex computing problems for desired specifications through modern tool usage, appropriate technologies and programming skills.

PSO2: Use managerial and domain Skills of Information Management to model an application's data requirements using domain specific modelling tools, Transaction & Query processing, Indexing & Searching techniques, and extract information for interpreting the datasets for Decision Making.

PSO3: Apply suitable techniques and algorithms to Integrate Operating systems, Services, Network devices, Security mechanisms, and Infrastructure to meet the requirements for the deployment of an application and to communicate on computer networks.

COURSE OUTCOMES:

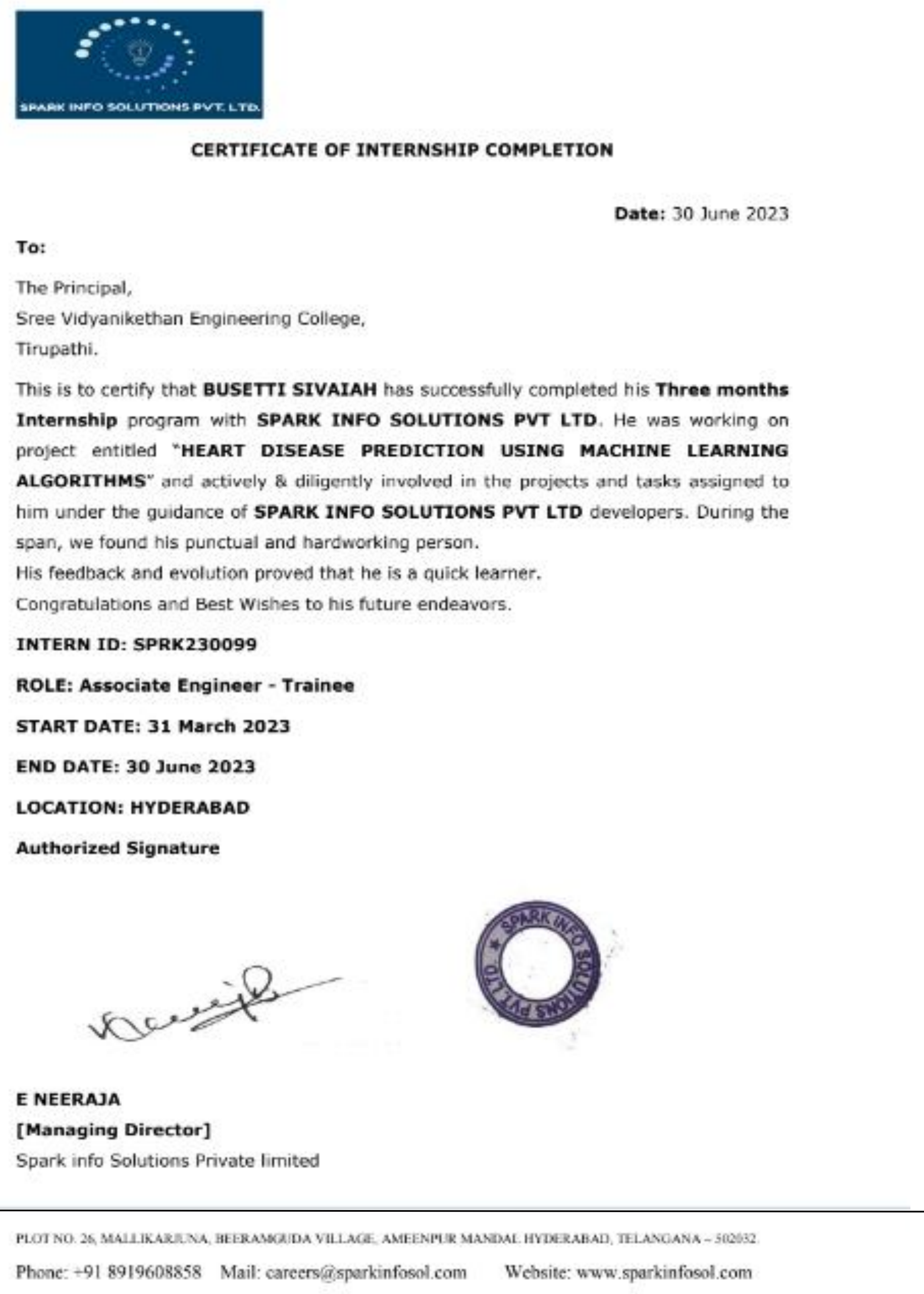
Completion of the project work enables a successful student to demonstrate:

CO1. Design and Develop software systems or processes to solve complex computing and allied problems using appropriate tools and techniques following relevant standards, codes, policies, regulations, and latest developments.

CO2. Consider society, health, safety, environment, legal, economics and project management in solving complex computing and allied problems.

CO3. Perform individually or in a team besides communicating effectively in written, oral and graphical forms on Software systems or processes.

Company offers Letter



SPARK INFO SOLUTIONS PVT. LTD.

CERTIFICATE OF INTERNSHIP COMPLETION

Date: 30 June 2023

To:

The Principal,
Sree Vidyanikethan Engineering College,
Tirupathi.

This is to certify that **BUSETTI SIVAIAH** has successfully completed his **Three months Internship** program with **SPARK INFO SOLUTIONS PVT LTD**. He was working on project entitled "**HEART DISEASE PREDICTION USING MACHINE LEARNING ALGORITHMS**" and actively & diligently involved in the projects and tasks assigned to him under the guidance of **SPARK INFO SOLUTIONS PVT LTD** developers. During the span, we found his punctual and hardworking person.
His feedback and evolution proved that he is a quick learner.
Congratulations and Best Wishes to his future endeavors.

INTERN ID: SPRK230099

ROLE: Associate Engineer - Trainee

START DATE: 31 March 2023

END DATE: 30 June 2023

LOCATION: HYDERABAD

Authorized Signature

[illegible]

SPARK INFO SOLUTIONS PVT LTD

E NEERAJA
[Managing Director]
Spark info Solutions Private limited

PLOT NO. 26, MALLIKARJUNA, BEERAMGUDA VILLAGE, AMEENPUR MANDAL HYDERABAD, TELANGANA – 502032.

Phone: +91 8919608858 Mail: careers@sparkinfosol.com Website: www.sparkinfosol.com

ACKNOWLEDGEMENTS

This acknowledgement transcends the reality of formality when I would like to express my deep gratitude and respect to all those people behind the screen who guided, inspired, and helped me in the completion of my project work.

I express my deep sense of gratitude to our beloved chairman **Dr. M. Mohan Babu** Padma Shri awardee for his encouragement throughout the course.

I owe my gratitude and special thanks to the Principal **Dr. B. M. Satish**, for his special encouragement and advice to shape myself for my future career.

I am extremely thankful to **Dr. M. Sunil Kumar**, HoD, Department of **Master of Computer Applications** for all provisions made and for his constant encouragement throughout my work.

I wish to express my deep sense of gratitude to my Project supervisor **Dr. M. Sunil Kumar**, Professor & Head, Department of MCA, Sree Vidyanikethan Engineering College for extending her/his valuable cooperation, moral support, kind attention, guidance, suggestions, and encouragement to complete my Project Work successfully.

I thank all my beloved **Faculty**, Department of MCA for giving their valuable suggestions and maximum co-operation.

I owe a deep sense of gratitude to my beloved **Parents** for extending their moral support in this Endeavour. I would like to thank all my **friends** who extended their help, encouragement, and moral support either directly or indirectly in completing my project work.

DECLARATION

I, **BUSETTI SIVAIAH** hereby declare that, the project entitled **"Prediction of heart disease using machine learning algorithms"** developed by me at **Spark Info Solutions Pvt Ltd**, **Hyderabad,** during the Academic year 2022-2023 and submitted to The Principal, Sree Vidyanikethan Engineering College (Autonomous) for partial fulfilment for the award of Master of Computer Applications (MCA).

I also declare that, the project is resulted by my own effort and that it has not been copied from any one and not been submitted by anybody in any of the University or Institution or Research Centre.

Place:

Date:

Busetti Sivaiah

(21121F0016)

ABSTRACT

Heart disease is one of the leading health illnesses in humans. It is a one of the major causes of mortality, millions of people are dying in every year around the world. Heart disease can include conditions such as coronary artery disease, heart failure, arrhythmias, and heart valve problems. There are over 30 different types of heart disease are there. Early and accurate prediction of heart disease can significantly improve patient outcomes by enabling timely intervention and appropriate treatment. In this model, we investigate the application of machine learning algorithms for the prediction of heart disease. We explore a comprehensive dataset consisting of patient information, including demographic factors, medical history, and clinical measurements. Machine learning algorithms have demonstrated remarkable potential in accurately predicting the occurrence and diagnosis of heart disease. Various machine learning algorithms, including logistic regression, decision trees, random forests, support vector machines (SVM), Gradient Boosting, XGBoost and artificial neural networks (ANN), are employed to develop predictive models. The predictive model is a hybrid model, in this we can combined Logistic regression, decision tree random forests, support vector machines (SVM), Gradient Boosting, XGBoost and artificial neural networks (ANN). To improve the accuracy the model. The dataset is pre-processed to handle missing values, normalize features, and address class imbalance. Feature selection techniques are applied to identify the most informative predictors for heart disease. The models are trained, validated, and evaluated using performance metrics such as accuracy, precision, recall, and area under the receiver operating characteristic curve (AUC-ROC). The main objective of this model is to propose an innovative approach for developing a model that effectively addresses real-world problems.

Keywords: logistic regression, decision trees, random forests, support vector machines (SVM), and artificial neural networks (ANN), performance Analysis.

Table of Contents

List of Figures

List of Tables

List of screenshots

1.INTRODUCTION

Objective: To create a predictive model that can precisely identify those who are at a high risk of acquiring heart disease, machine learning algorithms are being used to forecast heart disease. The objective is to train a machine learning model to properly forecast a person's likelihood of getting heart disease in the future using a variety of data inputs, including demographic data, medical history, lifestyle choices, and clinical test results. The aim is to provide early identification of individuals who are at high risk of heart disease so that preventative measures can be taken to reduce the risk of developing the condition. This could include lifestyle changes such as diet and exercise, as well as medical interventions such as medication or surgery.

Heart disease is a significant public health issue and a leading cause of death worldwide. It is also commonly known as cardiovascular disease. According to the world health organization estimated 17.9 million people died every year, which will be almost 31% of all global deaths [5]. Several risk factors contribute to the development of heart disease, including smoking, high blood pressure, high cholesterol levels, obesity, diabetes, a sedentary lifestyle, a family history of heart disease, and age. Early identification of individuals at high risk of developing heart disease is critical for effective prevention and management of the condition.

Traditional risk factors such as age, gender, and family history have limited predictive power, and there is a need for more accurate and effective predictive models. Prevention and management of heart disease involve adopting a healthy lifestyle, including regular physical activity, a balanced diet, avoiding smoking, maintaining a healthy weight, managing stress, and controlling other underlying conditions such as high blood pressure and diabetes. Treatment options for heart disease can vary depending on the specific condition but may include medications, lifestyle changes, medical procedures, or surgery.

The dataset name will be Heart Disease Cleveland it is available in the UCI machine learning repository. The dataset consists of 303 rows and 14 columns with the label Target. Data contains categorical as well as continuous data. the dataset has fields of Age, Sex, ChestPain, RestBP, Chol, Fbs, RestECG, MaxHR, ExAng, Oldpeak, Slope, Ca, Thal, and Target [6].

Machine learning algorithms have shown great potential in the prediction of heart disease. These algorithms can analyse large amounts of data, including medical history, lifestyle habits, and clinical test results, to develop more accurate predictive models for identifying individuals at high risk of heart disease. The use of machine learning algorithms has the potential to improve patient outcomes by enabling healthcare providers to intervene earlier in disease progression. However, the development of accurate machine learning models requires access to high-quality data, appropriate feature selection, and careful model training and validation. Additionally, the use of machine learning algorithms in healthcare raises ethical considerations around data privacy, bias, and transparency.

In this model, we use machine learning algorithms like logistic regression, decision trees, random forests, support vector machines (SVM), Gradient Boosting, XGBoost and artificial neural networks (ANN).

Logistic regression is a widely used statistical model employed for binary classification tasks. Despite its name, it is a regression algorithm that is commonly used for classification tasks. Logistic regression models the relationship between a set of input variables (features) and a binary target variable by estimating the probabilities of the target variable belonging to each class.

A decision tree is a widely used and well-known supervised machine learning algorithm employed for both classification and regression tasks. It constructs a tree-like model of decisions and their potential outcomes using a given set of input features. Decision trees are easy to understand and interpret, making them popular for both analytical and predictive tasks.

Random Forest is an ensemble learning method that combines multiple decision trees to create a robust and accurate predictive model. It is a supervised learning algorithm used for both classification and regression tasks. Random Forest improves upon the decision tree algorithm by reducing over fitting and increasing predictive performance.

Support Vector Machines (SVM) is a widely recognized and highly effective supervised machine learning algorithm utilized for both classification and regression purposes. It effectively separates and categorizes data points by constructing decision boundaries in a high-dimensional space, based on provided input features. It is particularly effective in scenarios where the data is not linearly separable in the input feature space.

Gradient Boosting is a machine learning algorithm that combines the predictions of multiple weak learners (typically decision trees) to create a strong predictive model. It works by iteratively training new models that focus on the errors made by previous models, gradually reducing the overall prediction error.

XGBoost (Extreme Gradient Boosting) is an optimized implementation of gradient boosting, which is an ensemble learning method that combines multiple weak models (typically decision trees) to create a strong predictive model. XGBoost is widely used in machine learning competitions and has gained popularity for its performance and scalability.

2.LITERATURE SURVEY

A literature survey on heart disease prediction using various machine learning algorithm.in this surveys we collect data from various research papers, publications and journals. Some of them are represented bellow.

Title: "Heart Disease Identification Method Using Machine Learning Classification in E-Healthcare"

Author: SALAH UD DIN, ASIF KHAN, AMIN UL HAQ, JALALUDDIN KHAN AND JALALUDDIN KHAN

The authors work This project at the University of Electronic Science and Technology in China, this model main intention is low Redundancy and high Relevance. In this model the authors use the algorithm like ANN, KNN and decision tree that can be used in conjunction with classification algorithms like SVM and Logistic Regression. The decision tree may be improved by adopting these feature selection strategies by concentrating on the most relevant and informative characteristics, increasing its accuracy and understandability [1].

Title:” Efficient Prediction of Cardiovascular Disease Using Machine Learning Algorithms with Relief and LASSO Feature Selection Techniques”

Author: MIRJAM JONKMAN, ABHIJITH REDDY BEERAVOLU, F. M. JAVED MEHEDI SHAMRAT, ASIF KARIM, EVA IGNATIOUS, SHAHANA SHULTANA, SAMI AZAM, AND FRISO DE BOER

The researchers done on Daffodil International University's in Dhaka, Bangladesh (1225). The right features can be selected using the Relief and Least Absolute Shrinkage and Selection Operator (LASSO) techniques by combining the conventional classifiers with bagging techniques like the Decision Tree, KNN, Random Forest, AdaBoost and Gradient Boosting [2].

Title: “Comparative Study of Optimum Medical Diagnosis of Human Heart Disease Using Machine Learning Technique with and Without Sequential Feature Selection”

Author: SYED MD. HUMAYUN AKHTER, HIRA FATIMA, GHULAB NABI AHMAD, SHAFIULLAH, AND ABDULLAH ALGETHAMI

the authors do this work on Mangalayatan University, Uttar Pradesh, India. It was the site of the authors' work on this project. Researchers used a number of methods to create this model to predict cardiac illness, including Decision Tree, SVM, Linear Discriminant Analysis, Random Forest, Gradient Boosting Classifier, and K-Nearest Neighbours. To choose pertinent characteristics for the prediction job, they also included the sequential feature selection method [3].

Title: “Efficient Medical Diagnosis of Human Heart Diseases Using Machine Learning Techniques with and Without GridSearchCV”

Author: SHAFIULLAH, ABDELAZIZ SALAH SAIDI, GHULAB NABI AHMAD, AND IMDADULLAH

The authors prepare this model at the Mangalayatan University, Uttar Pradesh, India. They used the GridSearchCV and many Machine Learning algorithms to predict cardiac illness, including Logical regression, KNN and SVM. Verification is done using 5-fold cross-validation approach to assess performance of models. The effectiveness of the various models in predicting and diagnosing heart illnesses is examined and compared using these datasets. The study intends to evaluate the generalization and robustness capabilities of the models across various populations and data sources [4] by utilizing a variety of datasets.

3.PROBLEM DEFINITION

The problem addressed by “PREDICTION OF HEART DISEASE USING MACHINE LEARNING ALGORITHMS” is a leading cause of death worldwide, and early identification of individuals at high risk is critical for effective prevention and management of the condition. Current methods for identifying individuals at risk of heart disease rely on traditional risk factors such as age, gender, and family history, which have limited predictive power.

3.1. EXISTING SYSTEM

The majority of cardiac problems are mostly avoidable, and early diagnosis and treatment considerably improve the prognosis. Examples of such changes in lifestyle include quitting smoking, eating healthier, exercising, and avoiding obesity. Due to the multifactorial nature of many contributing risk factors such as diabetes, BP, cholesterol, etc., it is challenging to identify high-risk individuals [7].

Due to its superiority in pattern identification and classification, ML has demonstrated that it is useful in helping to make judgements and predictions from huge amount of data produced by the health sector about cardiac disease [7]. Using the Framingham dataset, they will investigate various machine learning techniques in this model to determine if a patient has a chance of getting coronary disease (CHD). The dataset may be found on the Kaggle website. [8].

In order to select the most accurate outcome, we analyse four computational methods for this model: logistic regression, KNN, decision trees, and SVM.

Logistic regression

We get a 67.6% accuracy rate.

The logistic regression f1 score is 62.41%

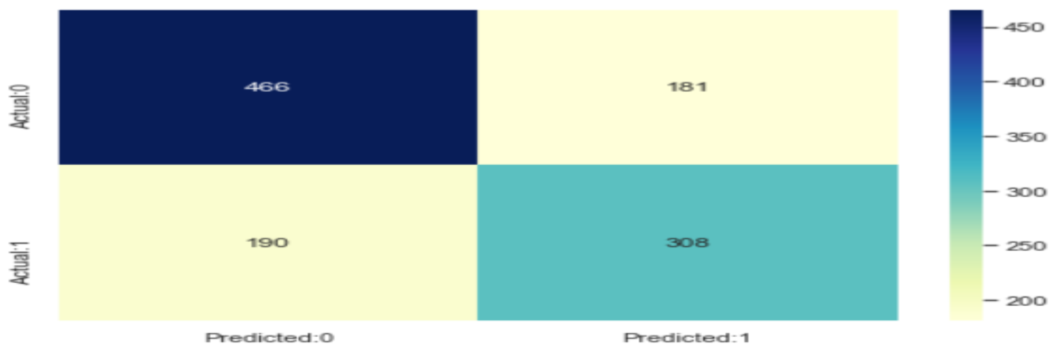


Figure 3.1.1: Accuracy rate of logistic regression. scale is 50, x-axes is prediction and y-axes are Actual.

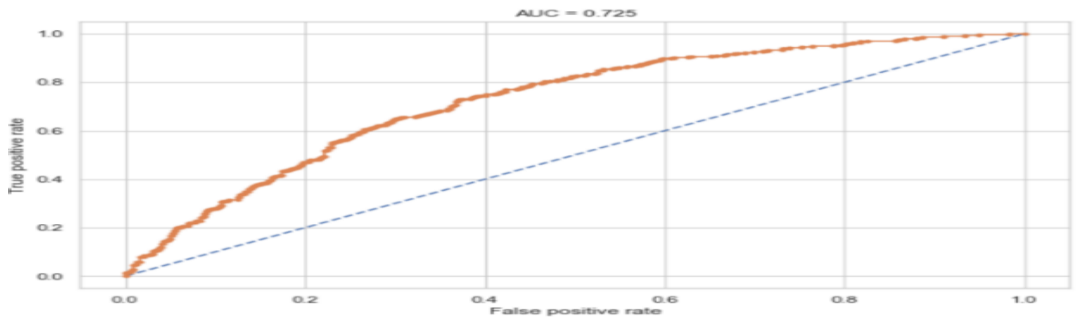


Figure 3.1.2: Graph consist x-Axes false value and y-axes is true value. scale is 0.2

k-Nearest Neighbors

Accuracy rate is 82.53%

f1 score is 82.27%.

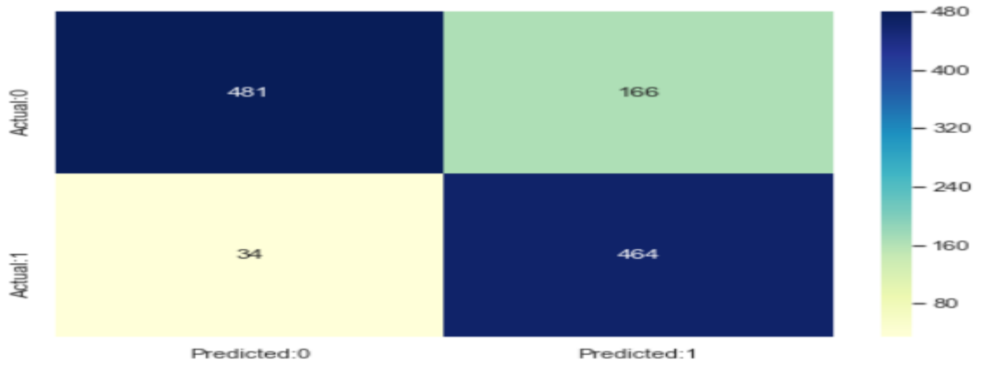


Figure 3.1.3: Accuracy rate of KNN. scale is 80, x-axes is prediction and y-axes are Actual.

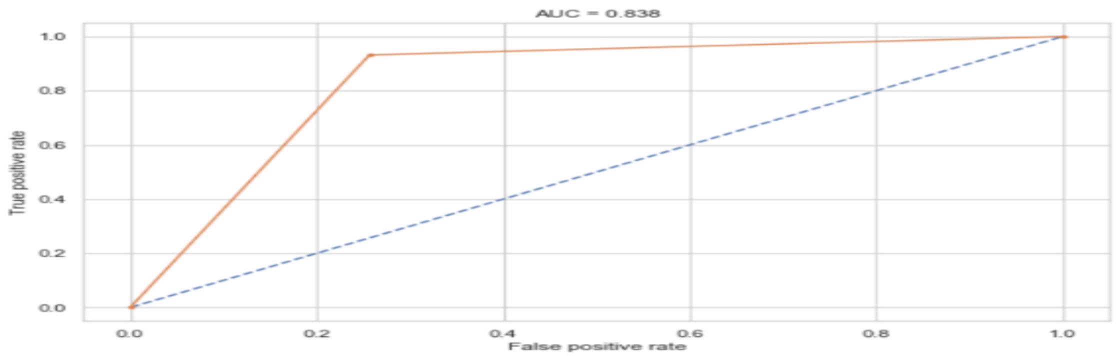


Figure 3.1.4: Graph consist x-Axes false value and y-axes is true value. scale is 0.2

Decision Trees

Accuracy rate is 72.4%.

f1 score is 67.62%.

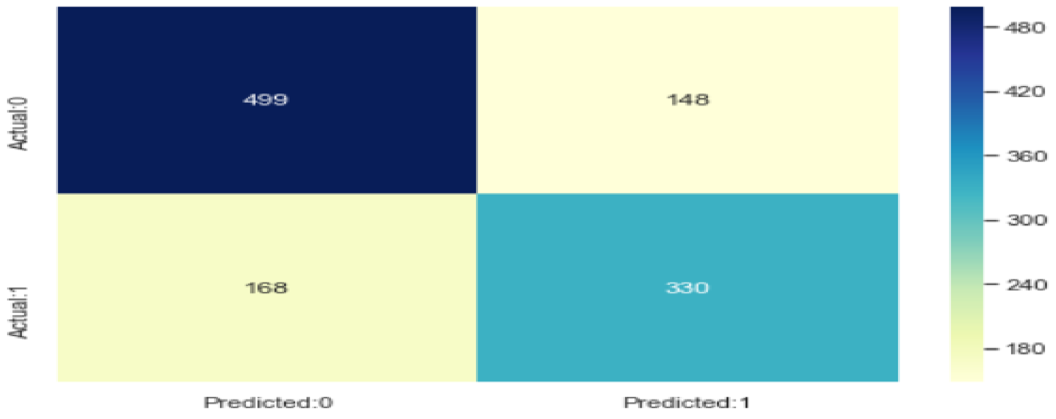


Figure 3.1.5: Accuracy rate of Decision Tree. scale is 60, x-axes is prediction and y-axes are Actual.

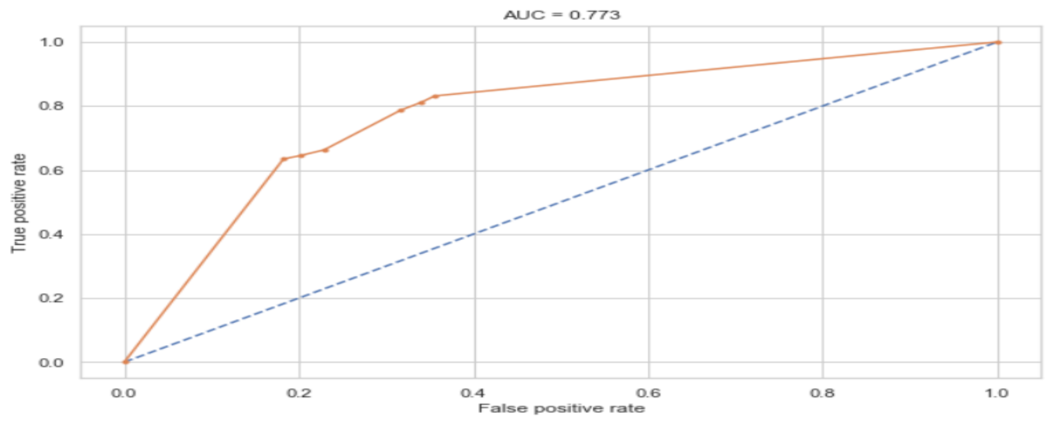


Figure 3.1.6: Graph consist x-Axes false value and y-axes is true value. scale is 0.2

Support Vector Machine

Accuracy rate 86.46%.

f1 score is 85.31%.

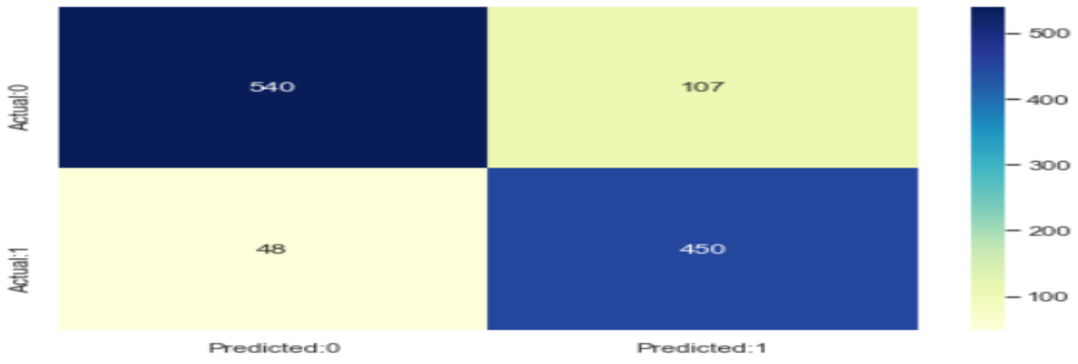


Figure 3.1.7: Accuracy rate of SVM. scale is 100, x-axes is prediction and y-axes are Actual.

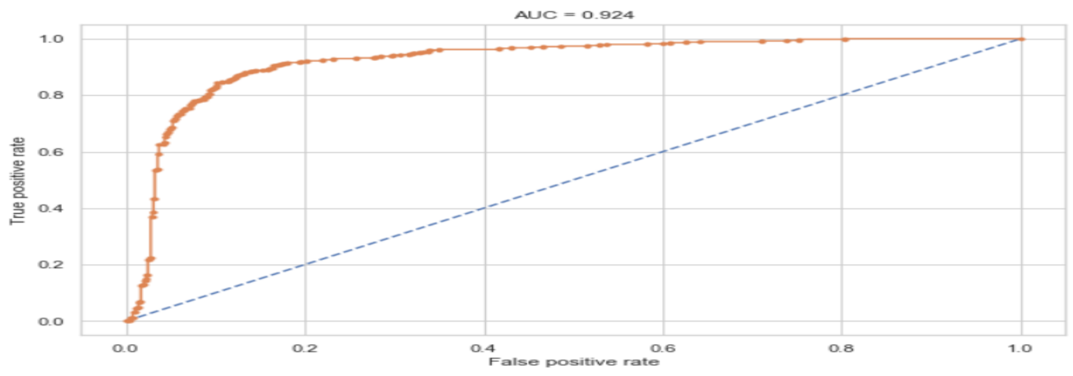
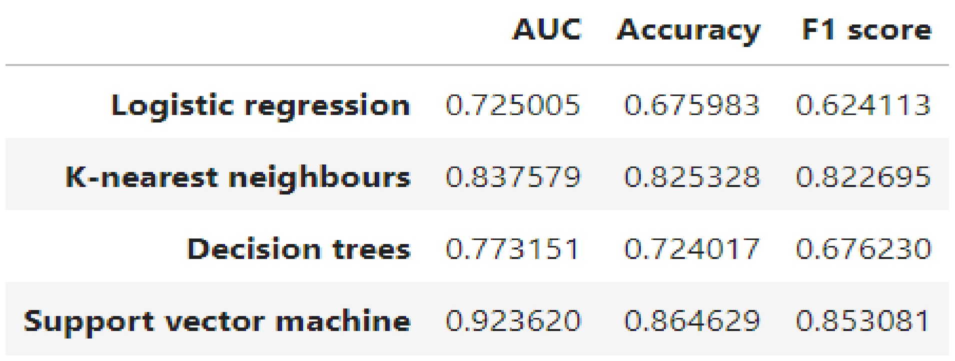


Figure 3.1.8: Graph consist x-Axes false value and y-axes is true value. scale is 0.2

Model Comparison



	AUC	Accuracy	F1 score
Logistic regression	0.725005	0.675983	0.624113
K-nearest neighbours	0.837579	0.825328	0.822695
Decision trees	0.773151	0.724017	0.676230
Support vector machine	0.923620	0.864629	0.853081

Advantages

1. The most important features in predicting the ten years risk of developing CHD were age and systolic blood pressure.
2. The Support vector machine with the radial kernel was the best performing model in terms of accuracy and the F1 score. Its high AUC and this show that it has a high true positive rate.
3. Balancing the dataset by using the smote technique helped in improving the model’s sensitivity
4. With more data (especially that of the minority class) better models can be built

Disadvantages

1. Time complexity and space complexity is more, because the model use all the algorithm models each and every time.
2. Developing accurate machine learning models requires access to high-quality data. If the data used to train the model is incomplete, inaccurate, or biased, it can lead to unreliable or incorrect predictions.

3.2. PROPOSED SYSTEM

The proposed method for the prediction of heart disease using hybrid machine learning algorithm which means combination of multiple algorithms. In this model we use combination of logistic regression, decision trees, random forests, support vector machines (SVM), Gradient Boosting, XGBoost and artificial neural networks (ANN).

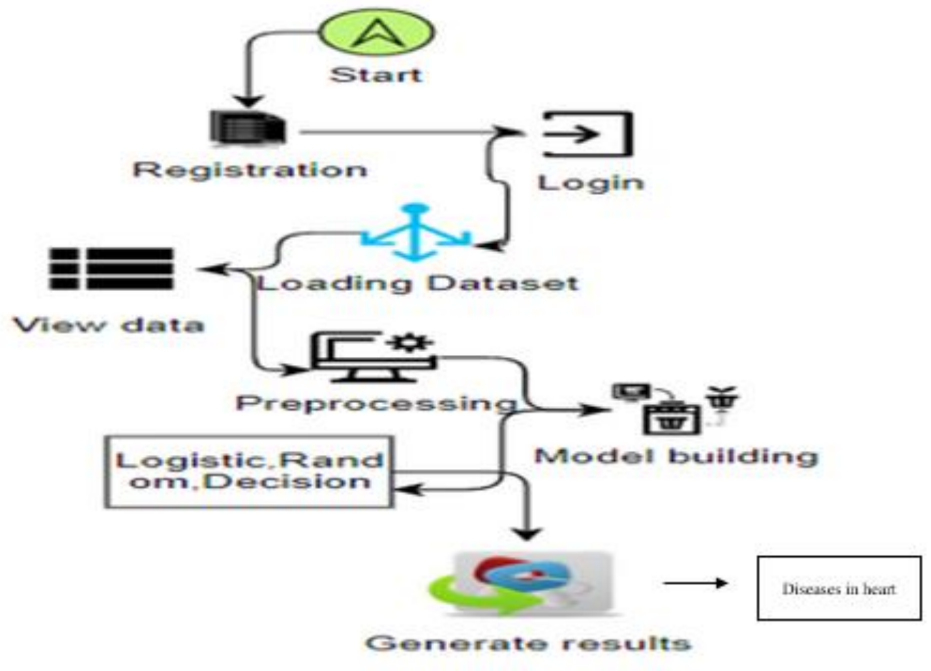


Figure-3.2.1: System Architecture

Logistic Regression

Logistic regression is indeed an example of supervised learning and is commonly used for binary classification problems. The goal of binary logistic regression is to predict the probability of a binary event occurring, such as predicting whether an outcome will be "yes" or "no," "true" or "false," or "positive" or "negative.

Mathematics behind logistic regression

Probability always ranges between 0 (does not happen) and 1 (happens).

$$h\Theta(x) = 1/1 + e- (\beta 0+\beta 1X)$$

'$h\Theta(x)$' is output of logistic function, where $0<= h\Theta(x)>=1$

'$\beta 1$' is the slope

'$\beta 0$' is the y-intercept

'x' is the independent variable

$(\beta 0+\beta 1*X)$-derived from equation of a line Y(prediction)= $(\beta 0+\beta 1*x)$ + Error value

Decision Tree

A decision tree is a hierarchical model utilized in decision support, which represents decisions and their potential outcomes, encompassing chance events, resource expenses, and utility. This algorithmic model employs conditional control statements and is non-parametric, supervised learning, making it applicable for both classification and regression tasks. The tree structure consists of a root node, branches, internal nodes, and leaf nodes, creating a hierarchical, tree-like arrangement. Decision tree outputs typically consist of binary outcomes, such as "yes" or "no".

Random Forest

Random Forest is a machine learning technique employed for addressing regression and classification problems. It leverages ensemble learning, a methodology that combines multiple classifiers to deliver solutions for intricate problems. By utilizing an ensemble of decision trees, Random Forest integrates their predictions to make more accurate and robust predictions or classifications. This technique improves the performance and generalization of the model by reducing overfitting and increasing the diversity of the individual trees' predictions.

- Here's an overview of how Random Forest works:

- Data Preparation
- Random Sampling
- Building Decision Trees
- Feature Subset Selection
- Ensemble of Decision Trees
- Out-of-Bag Evaluation
- Prediction

Support Vector Machine

Support Vector Machine (SVM) is a robust supervised algorithm that demonstrates its strength particularly on complex datasets, even though it performs well on smaller datasets as well. SVM can be utilized for both regression and classification tasks, but it is renowned for its effectiveness in solving classification problems. By finding the optimal hyperplane that maximally separates different classes, SVM is capable of handling datasets with complex decision boundaries and nonlinear relationships between variables. This makes SVM a powerful tool for tackling classification problems, offering high accuracy and robustness in various domains.

Gradient Boosting

Gradient Boosting is a machine learning algorithm that combines the predictions of multiple weak learners (typically decision trees) to create a strong predictive model. It works by iteratively training new models that focus on the errors made by previous models, gradually reducing the overall prediction error.

- The key steps in gradient boosting are as follows:
- Initialization
- Model Training

- Update the Weights
- Predict Ensemble Building
- Repeat Steps 2-4
- Final Prediction

XGBoost

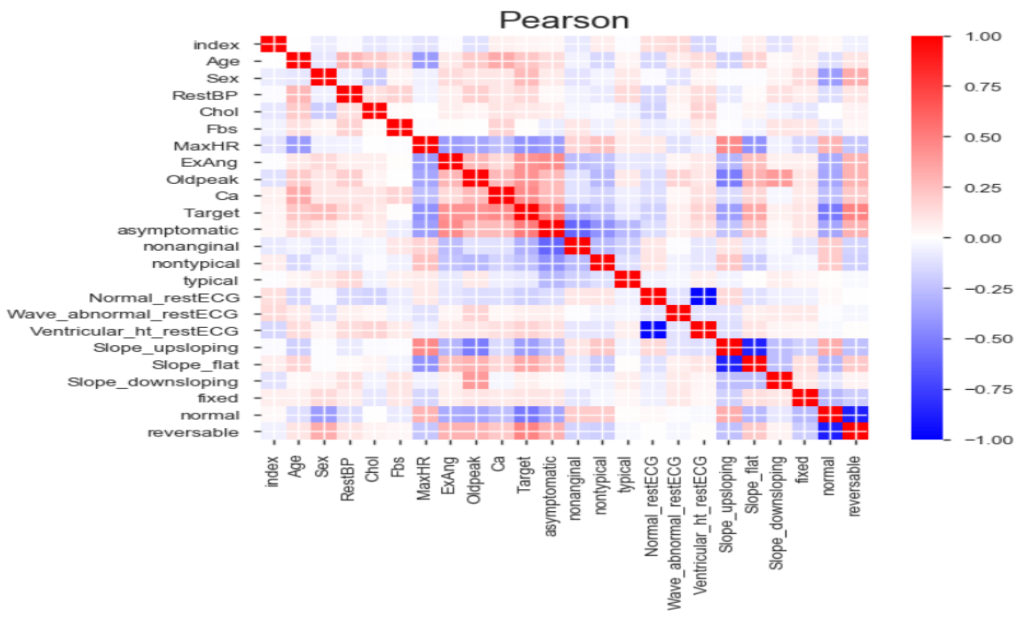
XGBoost (Extreme Gradient Boosting) is an optimized implementation of gradient boosting, which is an ensemble learning method that combines multiple weak models (typically decision trees) to create a strong predictive model. XGBoost is widely used in machine learning competitions and has gained popularity for its performance and scalability.

- Gradient Boosting
- Decision Tree Base Learners
- Regularization
- Feature Importance
- Handling Missing Values
- Parallel Processing and Scalability
- Cross-Validation
- Tree Pruning

Tunable Hyper parameters

In this model we use several steps is to collect relevant data on heart disease from different sources such as hospitals, medical records, and research studies. The data should include information on risk factors such as Age, Sex, ChestPain, RestBP, Chol, Fbs, RestECG, MaxHR, ExAng, Oldpeak, Slope, Ca, Thal.

Correlations



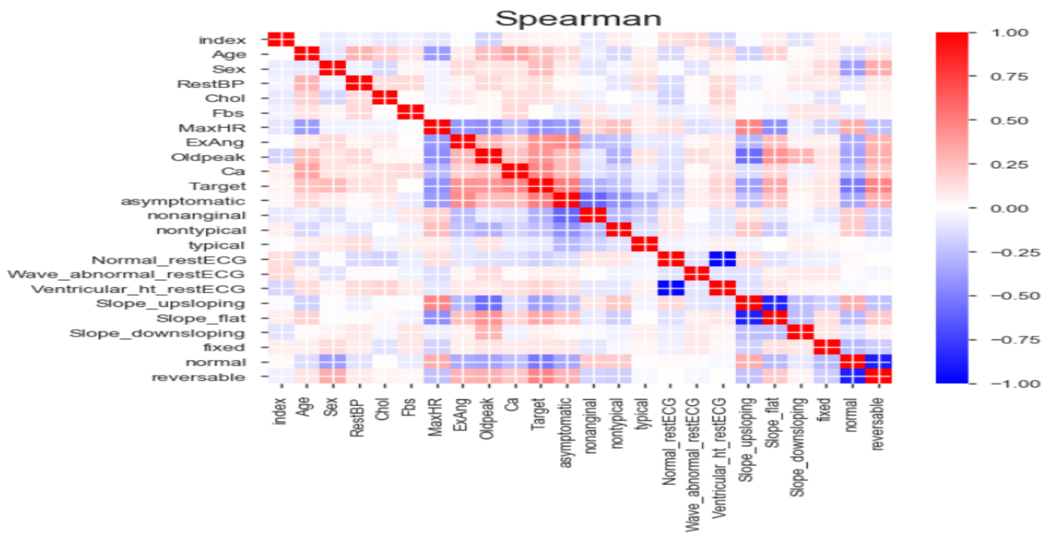


Figure 3.2.2: Pearson correlation and Spearman correlation. scale is 0.25

Advantages

1. **Improved Accuracy:** the proposed model use hybrid algorithm, it will be helps to improve accuracy.
2. **Low latency:** The proposed system has low time complexity when compared to the Existed model.

3.3 Hardware Requirements

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

3.4 Software Requirements

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

4. DATA COLLECTION

The dataset taken from UCI Machine learning repository. This database contains 76 attributes, but all published experiments refer to using a subset of 14 of them. In particular, the Cleveland database is the only one that has been used by ML researchers to this date. The "goal" field refers to the presence of heart disease in the patient. It is integer valued from 0 (no presence) to 4. Experiments with the Cleveland database have concentrated on simply attempting to distinguish presence (values 1,2,3,4) from absence (value 0). The names and social security numbers of the patients were recently removed from the database, replaced with dummy values. One file has been "processed", that one containing the Cleveland database. All four unprocessed files also exist in this directory.

Attribute Information

1. #3 (age)
2. #4 (sex)
3. #9 (cp)
4. #10 (trestbps)
5. #12 (chol)
6. #16 (fbs)
7. #19 (restecg)
8. #32 (thalach)
9. #38 (exang)
10. #40 (oldpeak)
11. #41 (slope)
12. #44 (ca)
13. #51 (thal)
14. #58 (num)

Complete attribute information:

1 id: patient identification number

2 ccf: social security number (I replaced this with a dummy value of 0)

3 age: age in years

4 sex: sex (1 = male; 0 = female)

5 painloc: chest pain location (1 = substernal; 0 = otherwise)

6 painexer (1 = provoked by exertion; 0 = otherwise)

7 relrest (1 = relieved after rest; 0 = otherwise)

8 pncaden (sum of 5, 6, and 7)

9 cp: chest pain type

-- Value 1: typical angina

-- Value 2: atypical angina

-- Value 3: non-anginal pain

-- Value 4: asymptomatic

10 trestbps: resting blood pressure (in mm Hg on admission to the hospital)

11 htn

12 chol: serum cholestoral in mg/dl

13 smoke: I believe this is 1 = yes; 0 = no (is or is not a smoker)

14 cigs (cigarettes per day)

15 years (number of years as a smoker)

16 fbs: (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)

17 dm (1 = history of diabetes; 0 = no such history)

18 famhist: family history of coronary artery disease (1 = yes; 0 = no)

19 restecg: resting electrocardiographic results

-- Value 0: normal

-- Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)

-- Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria

20 ekgmo (month of exercise ECG reading)

21 ekgday(day of exercise ECG reading)

22 ekgyr (year of exercise ECG reading)

23 dig (digitalis used furing exercise ECG: 1 = yes; 0 = no)

24 prop (Beta blocker used during exercise ECG: 1 = yes; 0 = no)

25 nitr (nitrates used during exercise ECG: 1 = yes; 0 = no)

26 pro (calcium channel blocker used during exercise ECG: 1 = yes; 0 = no)

27 diuretic (diuretic used used during exercise ECG: 1 = yes; 0 = no)

28 proto: exercise protocol

1 = Bruce

2 = Kottus

3 = McHenry

4 = fast Balke

5 = Balke

6 = Noughton

7 = bike 150 kpa min/min (Not sure if "kpa min/min" is what was written!)

8 = bike 125 kpa min/min

9 = bike 100 kpa min/min

10 = bike 75 kpa min/min

11 = bike 50 kpa min/min

12 = arm ergometer

29 thaldur: duration of exercise test in minutes

30 thaltime: time when ST measure depression was noted

31 met: mets achieved

32 thalach: maximum heart rate achieved

33 thalrest: resting heart rate

34 tpeakbps: peak exercise blood pressure (first of 2 parts)

35 tpeakbpd: peak exercise blood pressure (second of 2 parts)

36 dummy

37 trestbpd: resting blood pressure

38 exang: exercise induced angina (1 = yes; 0 = no)

39 xhypo: (1 = yes; 0 = no)

40 oldpeak = ST depression induced by exercise relative to rest

41 slope: the slope of the peak exercise ST segment

-- Value 1: upsloping

-- Value 2: flat

-- Value 3: downsloping

42 rldv5: height at rest

43 rldv5e: height at peak exercise

44 ca: number of major vessels (0-3) colored by flourosopy

45 restckm: irrelevant

46 exerckm: irrelevant

47 restef: rest raidonuclid (sp?) ejection fraction

48 restwm: rest wall (sp?) motion abnormality

0 = none

1 = mild or moderate

2 = moderate or severe

3 = akinesis or dyskmem (sp?)

49 exeref: exercise radinalid (sp?) ejection fraction

50 exerwm: exercise wall (sp?) motion

51 thal: 3 = normal; 6 = fixed defect; 7 = reversable defect

52 thalsev: not used

53 thalpul: not used

54 earlobe: not used

55 cmo: month of cardiac cath (sp?) (perhaps "call")

56 cday: day of cardiac cath (sp?)

57 cyr: year of cardiac cath (sp?)

58 num: diagnosis of heart disease (angiographic disease status)

-- Value 0: < 50% diameter narrowing

-- Value 1: > 50% diameter narrowing

(in any major vessel: attributes 59 through 68 are vessels)

59 lmt

60 ladprox

61 laddist

62 diag

63 cxmain

64 ramus

65 om1

66 om2

67 rcaprox

68 rcadist

69 lvx1: not used

70 lvx2: not used

71 lvx3: not used

72 lvx4: not used

73 lvf: not used

74 cathef: not used

75 junk: not used

76 name: last name of patient (I replaced this with the dummy string "name")

5. SYSTEM DESIGN

5.1 High Level Design Documentation

Input Design:

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

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

It should serve specific purpose effectively such as storing, recording, and retrieving the information.

It ensures proper completion with accuracy.

It should be easy to fill and straightforward.

It should focus on user's attention, consistency, and simplicity.

All these objectives are obtained using the knowledge of basic design principles regarding −

What are the inputs needed for the system?

How end users respond to different elements of forms and screens.

Objectives for Input Design:

The objectives of input design are −

To design data entry and input procedures

To reduce input volume

To design source documents for data capture or devise other data capture methods

To design input data records, data entry screens, user interface screens, etc.

To use validation checks and develop effective input controls.

Output Design:

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

Objectives of Output Design:

The objectives of input design are:

To develop output design that serves the intended purpose and eliminates the production of unwanted output.

To develop the output design that meets the end user's requirements.

To deliver the appropriate quantity of output.

To form the output in appropriate format and direct it to the right person.

To make the output available on time for making good decisions.

5.2 Flowchart

A flowchart is a diagram that depicts a process, system or computer algorithm. They are widely used in multiple fields to document, study, plan, improve and communicate often complex processes in clear, easy-to-understand diagrams. Flowcharts, sometimes spelled as flow charts, use rectangles, ovals, diamonds and potentially numerous other shapes to define the type of step, along with connecting arrows to define flow and sequence.

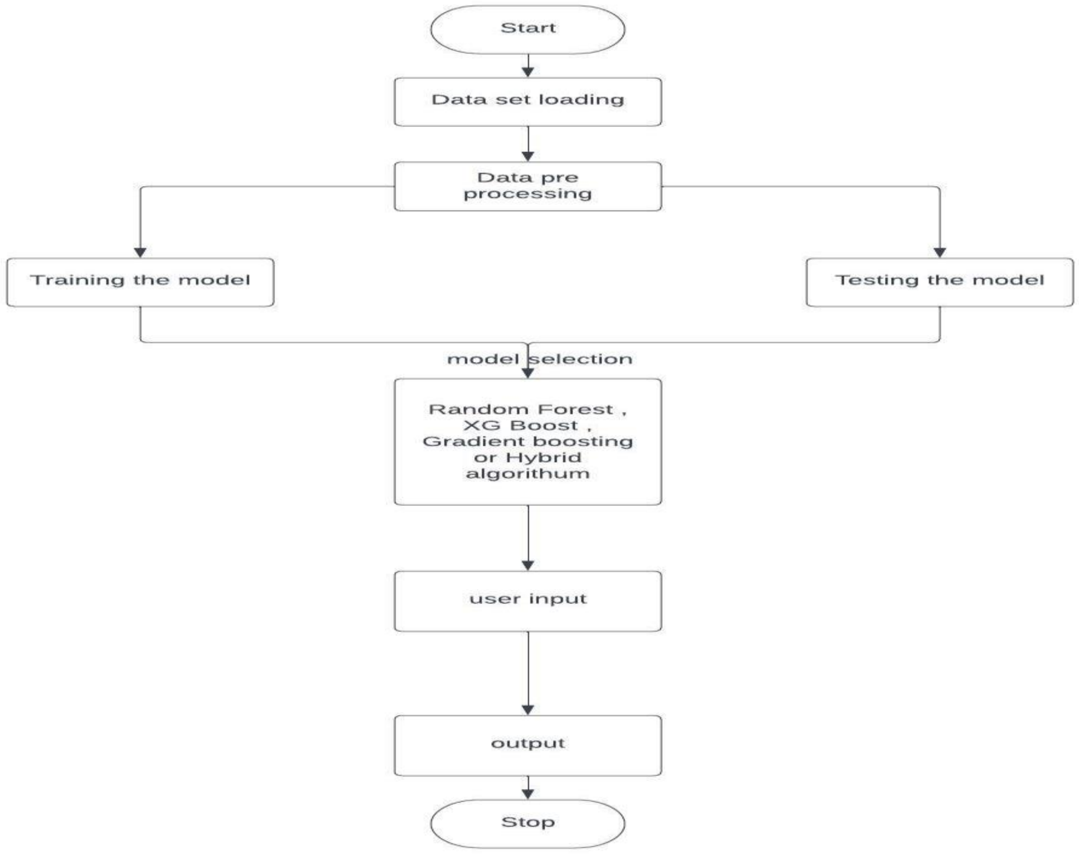


Figure 5.2.1: flow chart

6. IMPLEMENTATION

Home: Home page of the project.

Admin:

Admin: Operator needs to login.

Dash Board: Operator can view the Model performance graph and registered user's details.

Dataset access: Operator needs to upload the dataset regarding project.

Data Grooming: Operator needs to clean the data to implementing models(algorithms) to view the accuracy which fits for training / testing with algorithms.

Log-out: Logout from the web application.

Customer:

Home: Home page of the project.

Register: Customer needs to give his/her credentials to the registration.

Login: Customer needs to give credentials for login

Dashboard: Customer has option like change the password and as well as the prediction page for Outcome Verification.

Log-Out: Customer can logout.

Software Development Life Cycle – SDLC

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

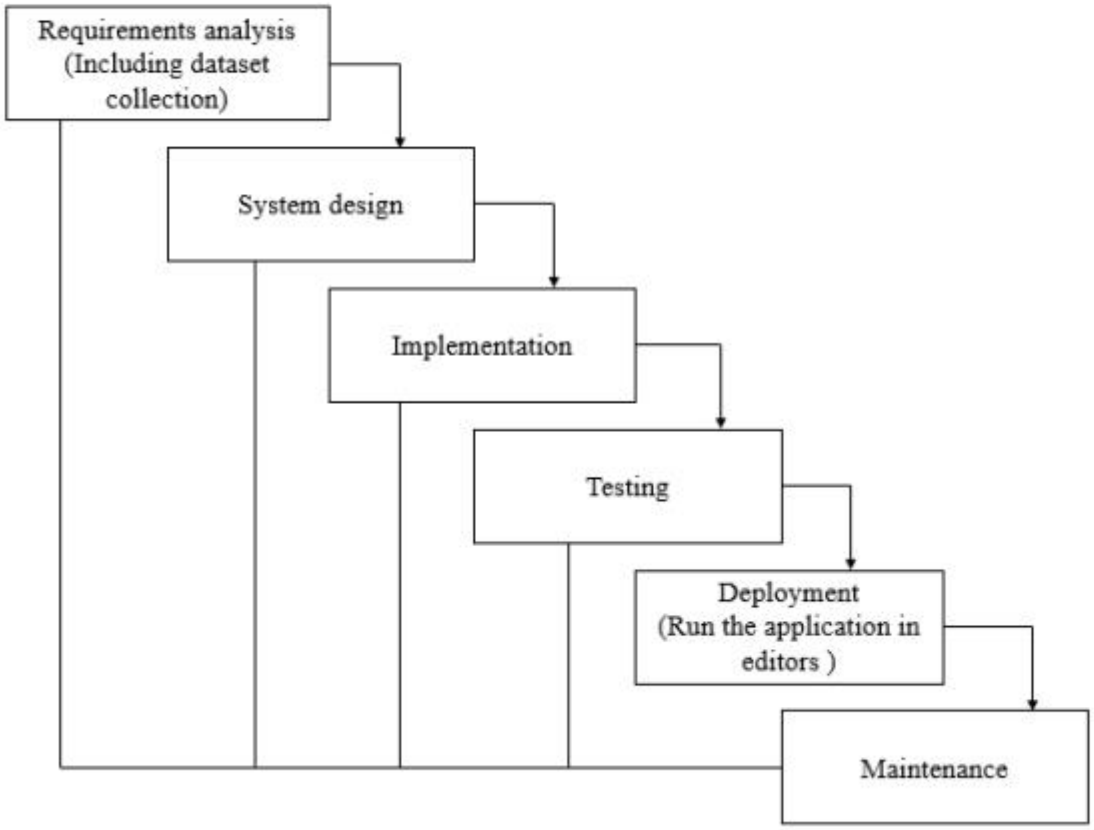

Requirements analysis (Including dataset collection)
System design
Implementation
Testing
Deployment (Run the application in editors)
Maintenance

Figure-6.1: software development life cycle

Requirement Gathering and analysis − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.

System Design − the requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.

Implementation − with inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.

Integration and Testing − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

Deployment of system − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

Maintenance − There are some issues which come up in the client environment. To fix those issues, patches are released. Also, to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

Feasibility Study

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

Three key considerations involved in the feasibility analysis are

ECONOMICAL FEASIBILITY
TECHNICAL FEASIBILITY
SOCIAL FEASIBILITY

Economic feasibility:

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

Technical feasibility:

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

Social feasibility:

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

System Requirements Specification

Functional and non-functional requirements:

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

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

Examples of functional requirements:

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

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

They basically deal with issues like:

Portability
Security
Maintainability
Reliability
Scalability
Performance
Reusability
Flexibility

Examples of non-functional requirements:

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

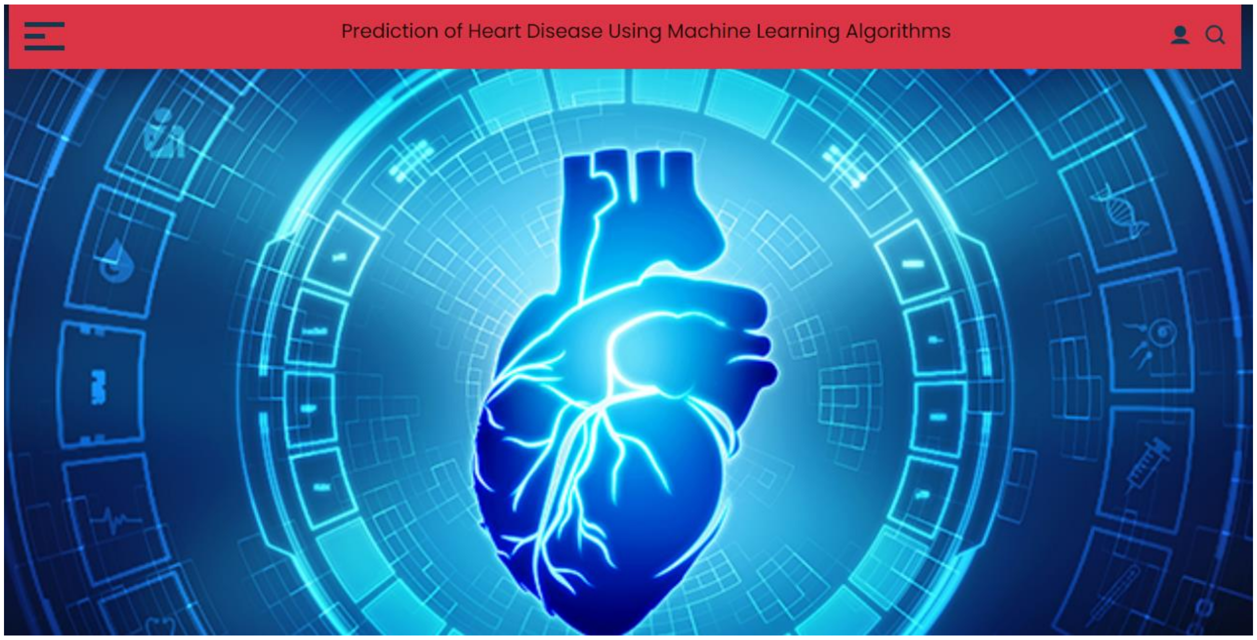
Learning Outcome:

Regarding Machine Learning
Supervised Machine Learning
Classification Technique
About PyCharm

Flask Frame work

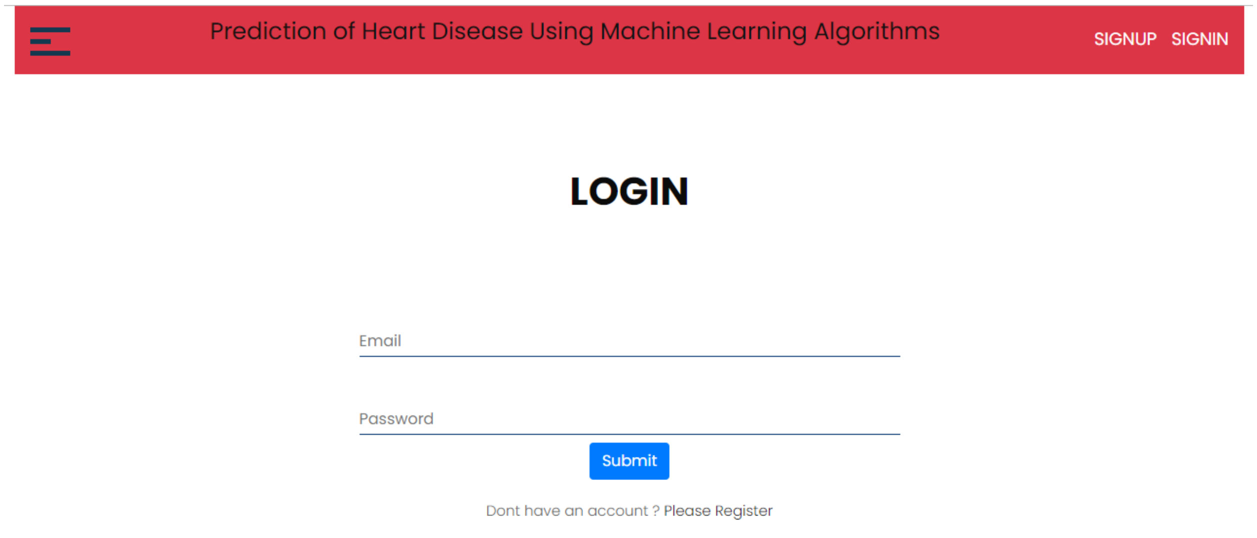
6.1 Screen Shots

Homepage:


Prediction of Heart Disease Using Machine Learning Algorithms

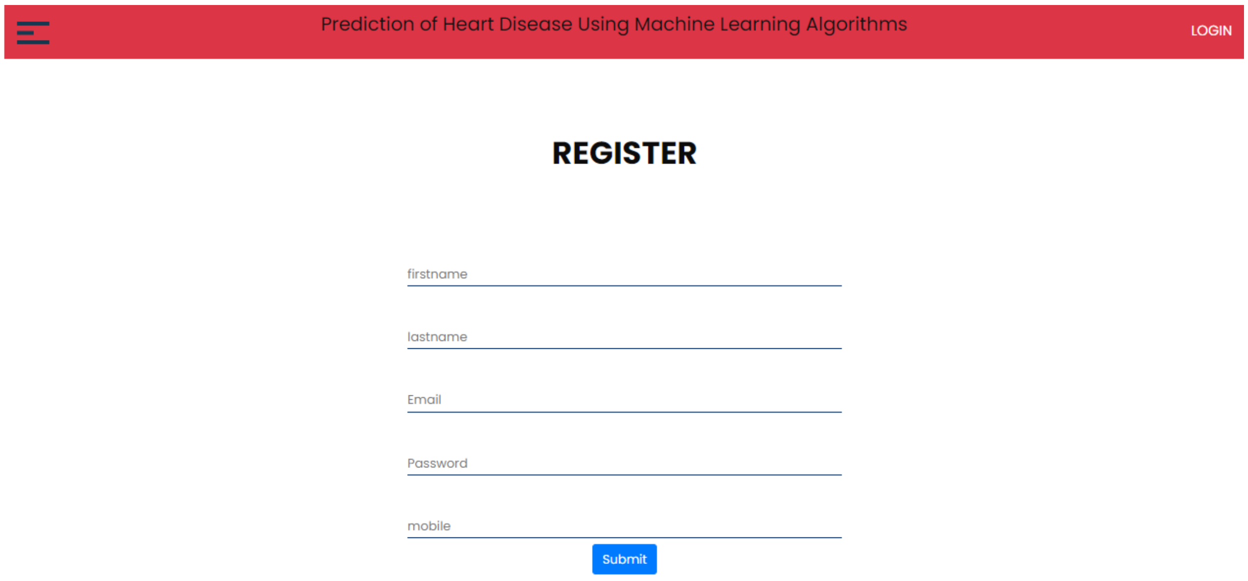
Screenshot-6.1.1: Home page of the project.

Login Page:


Prediction of Heart Disease Using Machine Learning Algorithms
SIGNUP
SIGNIN
LOGIN
Email
Password
Submit
Dont have an account ? Please Register

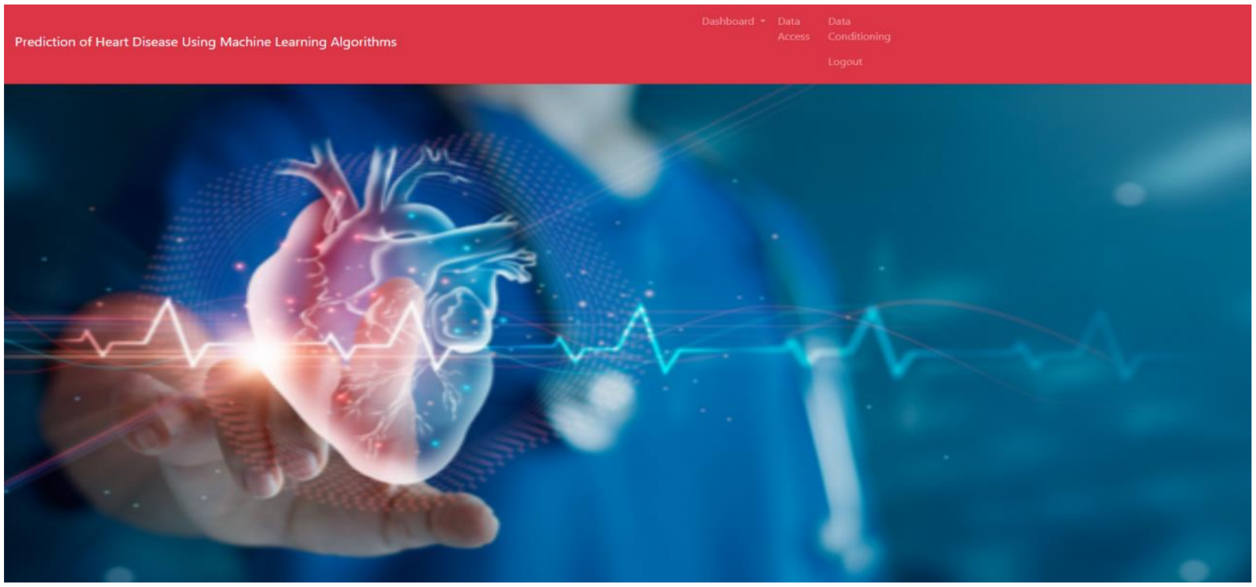
Screenshot-6.1.2: The user needs to give credentials for login.

Register page:


Prediction of Heart Disease Using Machine Learning Algorithms
LOGIN
REGISTER
firstname
lastname
Email
Password
mobile
Submit

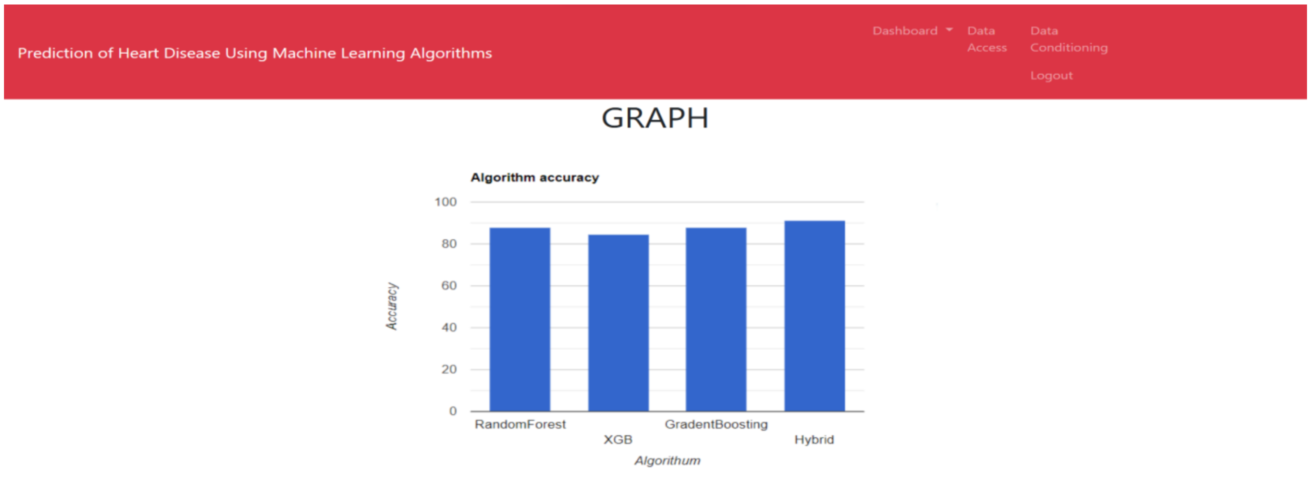
Screenshot-6.1.3: The user needs to give his/her credentials to the registration.

Admin Homepage:


Prediction of Heart Disease Using Machine Learning Algorithms
Dashboard
Data Access
Data Conditioning
Logout

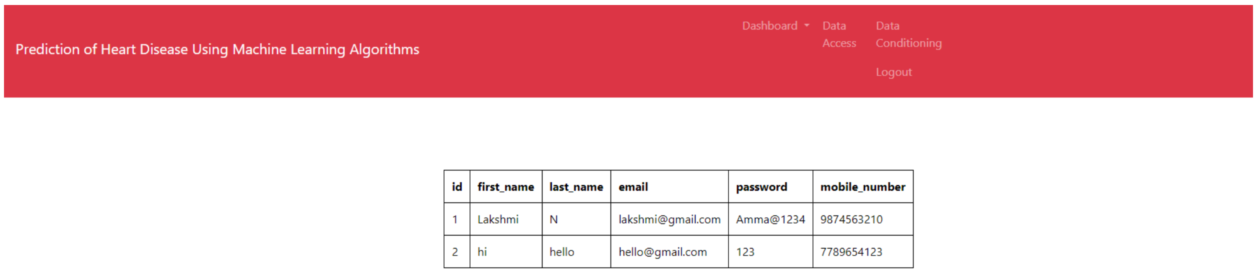
Screenshot-6.1.4: The admin Homepage

Graph:



Screenshot-6.1.5: Accuracy graph

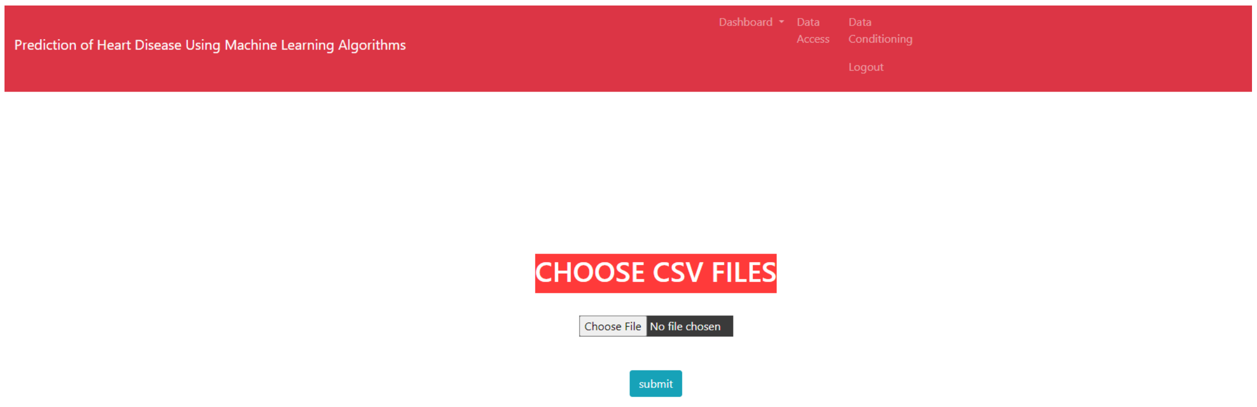
All user:


Prediction of Heart Disease Using Machine Learning Algorithms
Dashboard
Data Access
Data Conditioning
Logout

id	first_name	last_name	email	password	mobile_number
1	Lakshmi	N	lakshmi@gmail.com	Amma@1234	9874563210
2	hi	hello	hello@gmail.com	123	7789654123

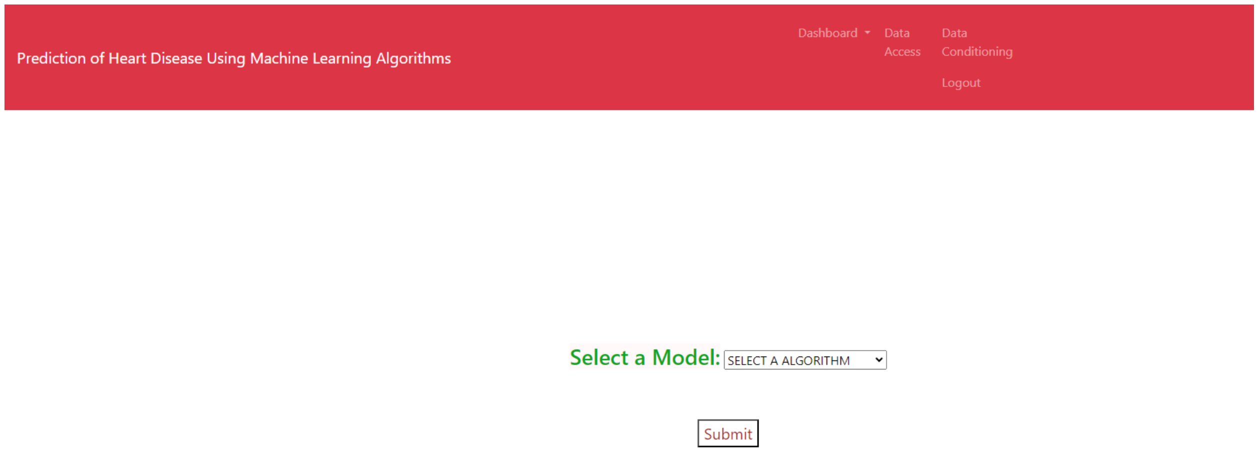
Screenshot-6.1.6: Users details

Data Access:



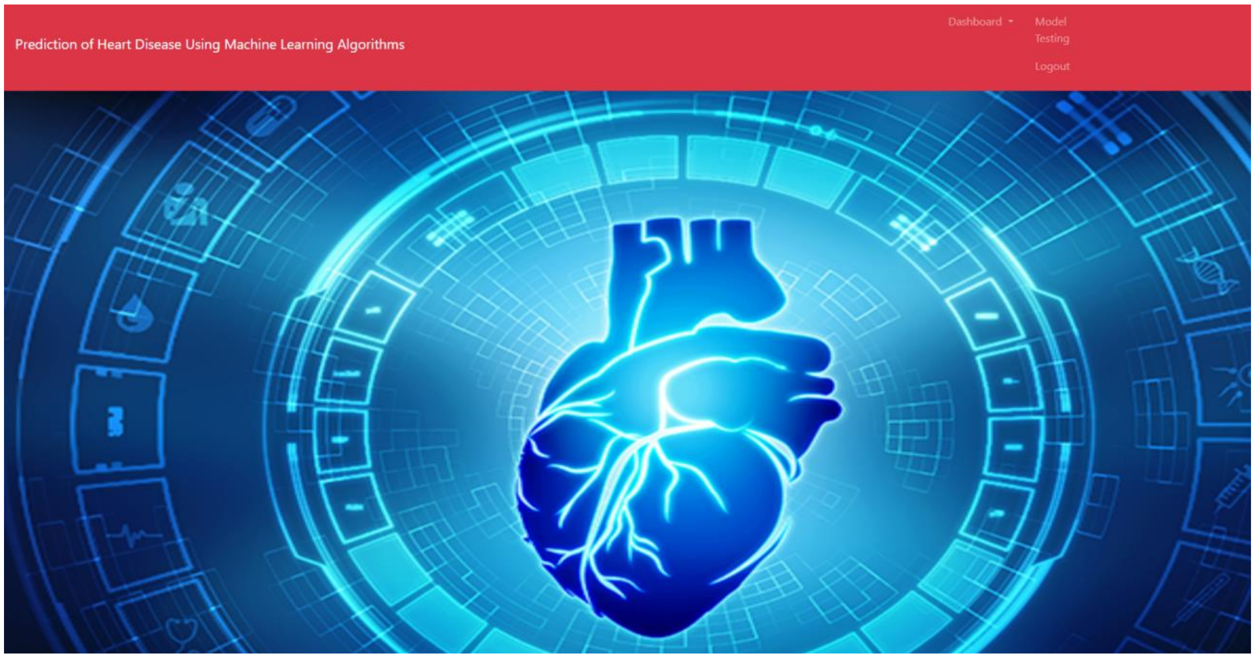
Screenshot-6.1.7: Data CSV

Data Conditioning:



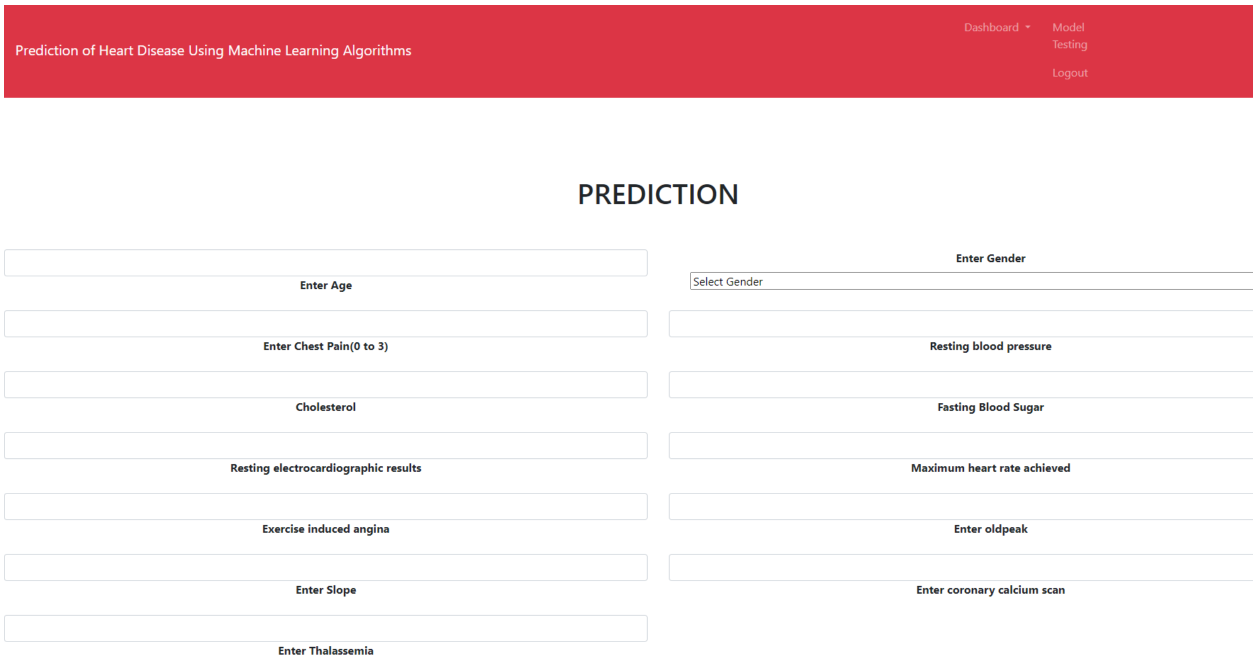
Screenshot-6.1.8: Algorithm selection

Customer Homepage:


Prediction of Heart Disease Using Machine Learning Algorithms
Dashboard
Model Testing
Logout

Screenshot-6.1.9: Customer Homepage

Prediction:


Prediction of Heart Disease Using Machine Learning Algorithms
Dashboard
Model Testing
Logout

PREDICTION

Enter Age	Enter Gender: Select Gender
Enter Chest Pain(0 to 3)	Resting blood pressure
Cholesterol	Fasting Blood Sugar
Resting electrocardiographic results	Maximum heart rate achieved
Exercise induced angina	Enter oldpeak
Enter Slope	Enter coronary calcium scan
Enter Thalassemia	

Screenshot-6.1.10: user input page

7. SYSTEM TESTING

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

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

7.1TYPES OF TESTS

Unit testing

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

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

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

Functional testing is centred on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems/Procedures:	interfacing systems or procedures must be invoked.

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

SYSTEM TEST

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

White Box Testing

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

Black Box Testing

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

Unit Testing:

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

Test strategy and approach

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

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

Features to be tested

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

Integration Testing

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

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

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

Acceptance Testing

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

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

7.2 Test Cases

Input	**Output**	**Result**
Input for the anomalous behaviour	Predicting the anomalous behaviour from the user input	Success

Figure 7.2.1: result table

Test cases model building

S.NO	Test cases	I/O	Expected O/T	Actual O/T	P/F
1	Read the datasets.	Need to provide the dataset path and the data should be in the form of CSV.	Data loaded successfully	Valid data format	P
2	Read the datasets.	Need to provide the dataset path and the data should be in the form of CSV.	Data loaded successfully	Dataset is not in CSV format	F
3	Preprocess Data	Need to enter the split size (20-30%) for training and testing	Data pre-processed and splits successfully	Data pre-processed and splits successfully	P
4	Preprocess Data	If the split size is more or less than the mentioned size	Data pre-processed and splits successfully	Model may get under fit or over fit	F
5	Model	Models need to trained with the training dataset	Models trained successfully	Accuracy obtained by each model	P
6	Model	If no model selected for training	Select any model for training	Select any model for training Select any model for training	F
7	Prediction	Enter details to predict the anomalous behaviour	Predict result as anomalous behavior or not	Predict result as anomalous behavior or not	P

Figure 7.2.2: Test case table

8. CONCLUSION

This model highlights the effectiveness of the hybrid machine learning algorithm in the prediction of heart disease. The combination of multiple algorithms and the incorporation of various risk factors enhance the accuracy and reliability of predictive model, potentially aiding in early detection and proactive management of heart disease.

8.1 Limitations

While heart disease prediction systems have made significant advancements in recent years, there are still some limitations to be aware of. Here are a few:

1. Data availability: The accuracy and reliability of prediction systems heavily rely on the quality and quantity of data available for analysis. If the system lacks access to comprehensive and diverse data, it may not be as effective in identifying all risk factors.

2. False Positives and False Negatives: Prediction systems may occasionally produce false positives (indicating heart disease when it is not present) or false negatives (failing to detect heart disease when it exists). These errors can impact patient care and create unnecessary anxiety or complacency.

3. In this model, the accuracy is varying between 85% to 90%.

4. Limited scope: heart disease prediction systems typically focus on known risk factors such as age, gender, family history, blood pressure, cholesterol levels, and smoking habits. However, there may be other risk factors or emerging factors that are not yet considered or included in the system, potentially leading to incomplete risk assessments.

5. Lack of prevention strategies: While prediction systems can identify individuals at risk, they may not provide personalized prevention strategies or interventions to mitigate the risk effectively. Collaborating with healthcare professionals is necessary to implement appropriate preventive measures.

8.2 Future Work

Future work in heart disease prediction systems can focus on addressing the limitations and enhancing the effectiveness of the existing systems. Here are some potential areas for improvement and future developments:

1. Integration of advanced technologies: heart disease prediction systems can benefit from incorporating advanced technologies such as artificial intelligence (AI), machine learning, and deep learning algorithms. These technologies can analyse large datasets more efficiently, identify complex patterns and relationships, and improve the accuracy of predictions.

2. Incorporation of additional risk factors: Expanding the scope of risk factors considered in heart disease prediction systems can enhance their accuracy. Researchers can explore including novel risk factors such as genetic markers, biomarkers, social determinants of health, psychological factors, and environmental factors to provide a more comprehensive risk assessment.

3. Integration with electronic health records (EHRs): Integrating heart disease prediction systems with EHRs can facilitate seamless data exchange and enhance the accuracy of predictions. Access to comprehensive medical histories, laboratory results, medication records, and other clinical data can provide a more holistic view of an individual's health and improve risk assessment accuracy.

4. Ethical considerations: As heart disease prediction systems advance, it is crucial to address ethical considerations. This includes ensuring data privacy, informed consent, transparency in algorithms and predictions, and mitigating potential biases or discrimination in the use of these systems.

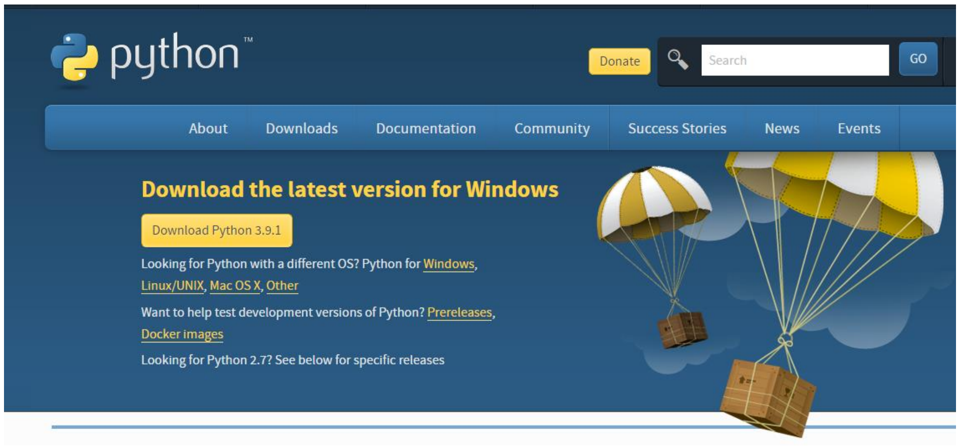
APPENDIX

A. USER MANUAL

SOFTWARE INSTALLATION FOR MACHINE LEARNING PROJECTS:

Installing Python:

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



Download the latest version for Windows

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Prereleases, Docker images

Looking for Python 2.7? See below for specific releases

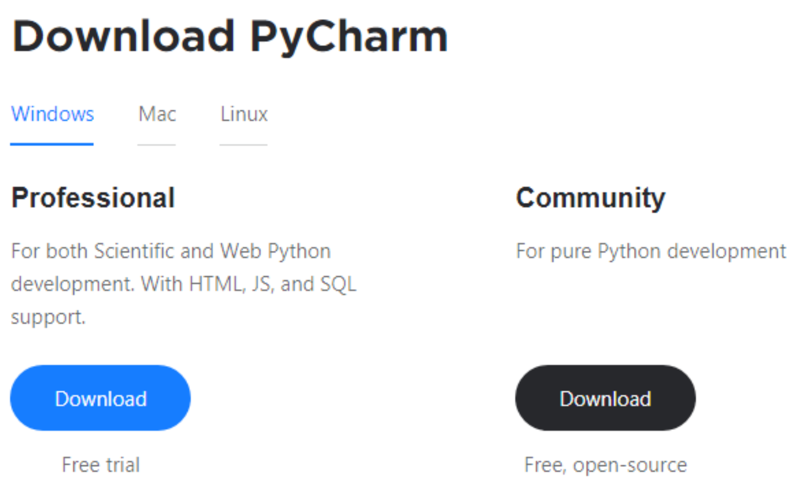
2. Once the download is complete, run the exe to install Python. Now click on Install Now.

3. You can see Python installing at this point.

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

Installing PyCharm:

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



Download PyCharm

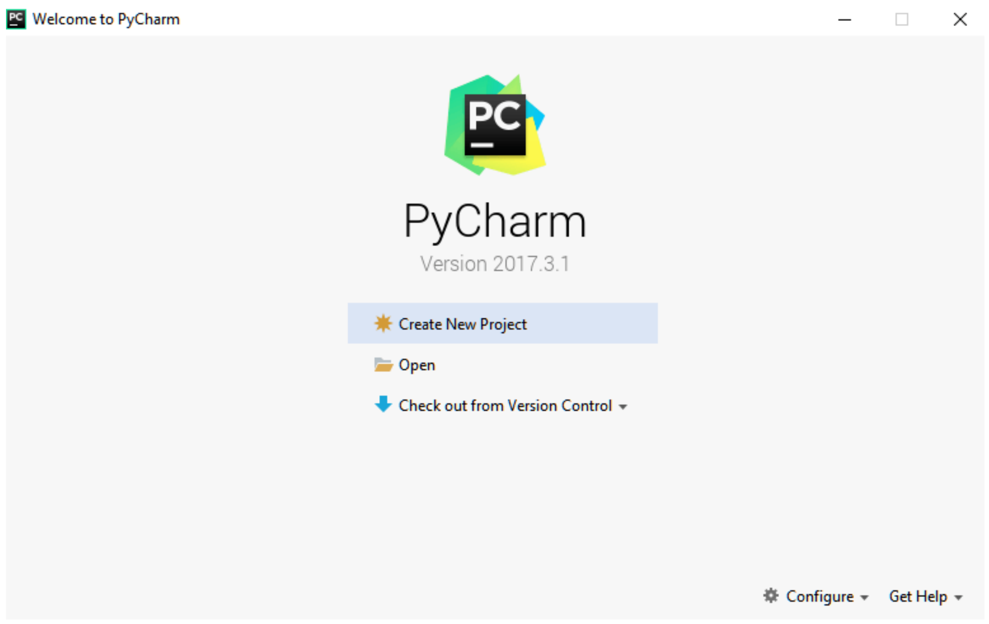
Professional

For both Scientific and Web Python development. With HTML, JS, and SQL support.

Community

For pure Python development

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

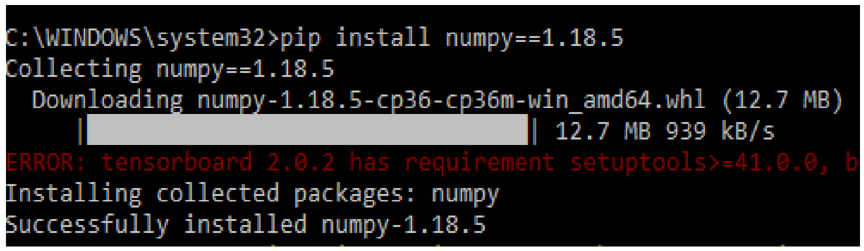


9. You need to install some packages to execute your project in a proper way.

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

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

Ex: Pip install NumPy



```
C:\WINDOWS\system32>pip install numpy==1.18.5
Collecting numpy==1.18.5
  Downloading numpy-1.18.5-cp36-cp36m-win_amd64.whl (12.7 MB)
     |████████████████████████████████| 12.7 MB 939 kB/s
ERROR: tensorboard 2.0.2 has requirement setuptools>=41.0.0, b
Installing collected packages: numpy
Successfully installed numpy-1.18.5
```

<u>Introduction To Python</u>

✓ Python

What Is a Script?

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

Scripts are reusable

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

Scripts are editable

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

You will need a text editor

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

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

Difference between a script and a program

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

Program: The program has an executable form that the computer can use directly to execute the instructions. The same program in its human-readable source code form, from which executable programs are derived (e.g., compiled)

Python

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

Python concepts

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

• Open-source general-purpose language.

• Object Oriented, Procedural, Functional

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

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

• Great interactive environment

• Great interactive environment

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

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented − Python supports an Object-Oriented style or technique of programming that encapsulates code within objects.

- Python is a Beginner's Language − Python is a great language for beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

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

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

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

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

Python Features

Python's features include −

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

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

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

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

Dynamic vs. Static

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

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

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

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

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

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

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

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

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

Variables

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

Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types −

- Numbers
- String
- List
- Tuple
- Dictionary

Python Numbers

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

Python Strings

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

Python Lists

Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data types.

The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 at the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

Python Tuples

A tuple is another sequence data type that is similar to a list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated. Tuples can be thought of as read-only lists.

Python Dictionary

Python's dictionaries are a kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type but is usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

Different Modes in Python

Python has two basic modes: normal and interactive.

The normal mode is the mode where the scripted and finished .pie files are run in the Python interpreter.

Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory. As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole

20 Python libraries

1. Requests. The most famous http library written by Kenneth remits. It's a must have for every python developer.

2. Scrappy. If you are involved in web scraping then this is a must have library for you. After using this library, you won't use any other.

3. Python. A guy toolkit for python. I have primarily used it in place of tinder. You will really love it.

4. Pillow. A friendly fork of PIL (Python Imaging Library). It is more user friendly than PIL and is a must have for anyone who works with images.

5. SQL Alchemy. A database library. Many love it and many hate it. The choice is yours.

6. Beautiful Soup. I know it’s slow but this xml and html parsing library is very useful for beginners.
7. Twisted. The most important tool for any network application developer. It has a very beautiful ape and is used by a lot of famous python developers.
8. Numbly. How can they leave this very important library? It provides some advance math functionalities to python.
9. Skippy. When talk about numbly then have to talk about spicy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.
10. Matplotlib. A numerical plotting library. It is very useful for any data scientist or any data analyser.
11. Pygmy. Which developer does not like to play games and develop them? This library will help you achieve your goal of 2d game development.
12. Piglet. A 3d animation and game creation engine. This is the engine in which the famous python port of mine craft was made
13. Pit. A GUI toolkit for python. It is my second choice after python for developing GUIs for my python scripts.
14. Pit. Another python GUI library. It is the same library in which the famous Bit torrent client is created.
15. Scaly. A packet sniffer and analyser for python made in python.
16. Pywin32. A python library which provides some useful methods and classes for interacting with windows.
17. Notch. Natural Language Toolkit – I realize most people won’t be using this one, but it’s generic enough. It is a very useful library if you want to manipulate strings. But its capacity is beyond that. Do check it out.
18. Nose. A testing framework for python. It is used by millions of python developers. It is a must have if you do test driven development.
19. Simply. Simply can-do algebraic evaluation, differentiation, expansion, complex numbers, etc. It is contained in a pure Python distribution.

20. I Python. I just can’t stress enough how useful this tool is. It is a python prompt on steroids. It has completion, history, shell capabilities, and a lot more. Make sure that you take a look at it.

NumPy

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

• Offers Matlab-ish capabilities within Python

• Fast array operations

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

Matplotlib

• High quality plotting library.

Python class and objects

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

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

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

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

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

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

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

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

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

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

Here’s a brief list of Python OOP ideas:

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

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

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

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

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

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

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

<u>Inheritance</u>

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

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

As they’ve seen, an instance of a class inherits the attributes of that class.

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

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

Operator Overloads

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

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

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

Exceptions

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

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

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

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

Here are just a few examples:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

User-Defined Exceptions

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

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

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

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

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

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

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

Python modules

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

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

Testing code

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

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

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

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

His entire process is known as the development cycle.

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

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

Functions in Python

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

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

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

There are three types of functions in Python: Help (), min (), and print ().

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

Some namespaces in Python:

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

Garbage Collection

Garbage Collector exposes the underlying memory management mechanism of Python, the automatic garbage collector. The module includes functions for controlling how the collector operates and examining the objects known to the system, either pending collection or stuck in reference cycles and unable to be freed.

Python XML Parser

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

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

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

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

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

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

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

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

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

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

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

Python Web Frameworks

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

Why are web frameworks useful?

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

Common web framework functionality

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

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

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

Comparing web frameworks

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

Web framework resources

- When you are learning how to use one or more web frameworks it's helpful to have an idea of what the code under the covers is doing.
- Frameworks is a really well-done short video that explains how to choose between web frameworks. The author has some particular opinions about what should be in a framework. For the most part, I agree although I've found sessions and database ORMs to be a helpful part of a framework when done well.
- What is a web framework? Is an in-depth explanation of what web frameworks being and their relation to web servers?
- Jingo vs. Flash vs. Pyramid: Choosing a Python web framework contains background information and code comparisons for similar web applications built in these three big Python frameworks.
- This fascinating blog post takes a look at the code complexity of several Python web frameworks by providing visualizations based on their code bases.

- Python’s web frameworks benchmarks are a test of the responsiveness of a framework by encoding an object to JSON and returning it as a response as well as retrieving data from the database and rendering it in a template. There were no conclusive results but the output is fun to read about nonetheless.
- What web frameworks do you use and why are they awesome? Is a language-agnostic Reedit discussion on web frameworks? It's interesting to see what programmers in other languages like and dislike about their suite of web frameworks compared to the main Python frameworks.
- This user-voted question & answer site asked "What are the best general-purpose Python web frameworks usable in production?” The votes aren't as important as the list of the many frameworks that are available to Python developers.

Web Frameworks learning checklist

- Choose a major Python web framework (Jingo or Flask are recommended) and stick with it. When you're just starting it's best to learn one framework first instead of bouncing around trying to understand every framework.
- Work through a detailed tutorial found within the resource links on the framework's page.
- Study open-source examples built with your framework of choice so you can take parts of those projects and reuse the code in your application.
- Build the first simple iteration of your web application then go to the deployment section to make it accessible on the web.

BIBLIOGRAPHY

[1] JIAN PING LI, AMIN UL HAQ, SALAH UD DIN, JALALUDDIN KHAN, ASIF KHAN, AND ABDUS SABOOR, "Heart Disease Identification Method Using Machine Learning Classification in E-Healthcare" and Digital Object Identifier 10.1109/ACCESS.2020.3001149

[2] PRONAB GHOSH, SAMI AZAM, MIRJAM JONKMAN, (Member, IEEE), ASIF KARIM, F. M. JAVED MEHEDI SHAMRAT, EVA IGNATIOUS, SHAHANA SHULTANA, ABHIJITH REDDY BEERAVOLU, AND FRISO DE BOER," Efficient Prediction of Cardiovascular Disease Using Machine Learning Algorithms with Relief and LASSO Feature Selection Techniques" and Digital Object Identifier 10.1109/ACCESS.2021.3053759

[3] GHULAB NABI AHMAD, SHAFIULLAH, ABDULLAH ALGETHAMI, HIRA FATIMA, AND SYED MD. HUMAYUN AKHTER, "Comparative Study of Optimum Medical Diagnosis of Human Heart Disease Using Machine Learning Technique with and Without Sequential Feature Selection" and Digital Object Identifier 10.1109/ACCESS.2022.3153047

[4] GHULAB NABI AHMAD, HIRA FATIMA1, SHAFIULLAH, ABDELAZIZ SALAH SAIDI, (Member, IEEE), AND IMDADULLAH, (Senior Member, IEEE), "Efficient Medical Diagnosis of Human Heart Diseases Using Machine Learning Techniques with and Without GridSearchCV" and Digital Object Identifier 10.1109/ACCESS.2022.3165792

[5] WORLD HEALTH ORGANIZATION https://www.who.int/india/health-topics/cardiovascular-diseases

[6] UCI machine learning repository, heart disease and https://archive.ics.uci.edu/dataset/45/heart+disease

[7] Amayo mode repository, HEART DISEASE PREDICTION USING MACHINE LEARNI and https://github.com/amayomode/Heart-Disease-Risk-Prediction