**A Project Report**

**On**

# HYPERTENSION PREDICTION USING SVM AlGORITHM



Submitted in partial fulfillment of the requirements for the award of the degree of

# MASTER OF COMPUTER APPLICATIONS

#### SUBMITTED BY

**MONDU LAKSHMI SATYA MANASWINI (21P31F0036)**

##### Under the Esteemed Guidance of

**Mrs. N. Vani, M.Sc, PGDSE, DSE**

**Assistant Professor**



**DEPARTMENT OF MCA**

# ADITYA COLLEGE OF ENGINEERING AND TECHNOLOGY

(Approved by AICTE, Affiliated to JNTUK & Accredited by NBA, NAAC with ‘A+’ Grade) Recognized by UGC under the sections 2(f) and 12(B) of the UGC act 1956 SURAMPALEM-

533437, East Godavari District, ANDHRA PRADESH.

**2021-2023**

# ADITYA COLLEGE OF ENGINEERING AND TECHNOLOGY

(Approved by AICTE, Affiliated to JNTUK & Accredited by NBA, NAAC with ‘A+’ Grade) Recognized by UGC under the sections 2(f) and 12(B) of the UGC act 1956 SURAMPALEM- 533437, E.G.Dist, ANDHRAPRADESH

**DEPARTMENT OF MCA**



**CERTIFICATE**

This is to certify that the project work entitled, **"HYPERTENSION PREDICTION USING SVM ALGORITHM",** is a bonafide work carried out by **MONDU LAKSHMI SATYA MANASWINI** bearing Regd.No:**21P31F0036** submitted to the requirements for the award of the Computer Applications in partial fulfilment of the requirements for the award **o**f degree of **MASTER OF COMPUTER APPLICATIONS** from **Aditya College of Engineering and Technology**, surampalem during the academic year **2022-2023**.

**Project Guide Head of the Department**

**Mrs. N. VANI, M.sc, PGDSE, DSE** **Mr. R.V.V.N. Bheema Rao M.Tech**

Assistant Professor, Associate Professor,

Department of MCA, Department of MCA,

Aditya College Of Engg & Tech, Aditya College Of Engg & Tech,

Surampalem-533437. Surampalem-533437.

**EXTERNAL EXAMINER**

# DECLARATION

I hereby declare that the project entitled **“HYPERTENSION PREDICTION USING SVM ALGORITHM”** done on my own and submitted to **Aditya College of Engineering & Technology, Surampalem** has been carried out by me alone under the guidance of **Mrs. N. Vani.**

Place: Surampalem Date:

(M.L.S. Manaswini) 21P31F0036

# ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned our efforts with success. It is a pleasant aspect that I have now the opportunity to express my gratitude for all of them.

The first person I would like to thank is my guide **Mrs. N. VANI, Assistant Professor, Aditya College of Engineering and Technology, Surampalem,** her wide knowledge and logical way of thinking have made a deep impression on me. Her understanding, encouragement, and personal guidance have provided the basis for this project. She is a source of inspiration for innovative ideas and her kind support is well known to all her students and colleagues.

I wish to thank **Mr.R.V.V.N. Bheema Rao M.Tech, Head of the Department, Aditya College Of Engineering And Technology, Surampalem,** has extended his support for the success of this project.

I also wish to thank **Dr. Dola Sanjay S, Principal**, **Aditya College Of Engineering And Technology, Surampalem,** who has extended his support for the success of this project.

I would like to thank all the **Management & Technical Supporting Staff** for their timely help throughout the project.

ABSTRACT

The goal of this project is to predict hypertension in individuals based on their clinical information. Hypertension is also known as high blood pressure, it is a common medical condition the affects millions of people and can lead to serious health complications such as heart disease, stroke, and kidney failure. The dataset used for this project includes information on patients age, gender, race, body mass index (BMI), cholesterol levels, and blood pressure readings. Several machine learning algorithms will be trained on the dataset, including Logistic Regression, Naive Bayesian Classifiers and Support Vector Machine (SVM) to determine which algorithm performs best in predicting hypertension. The model will be evaluated based on several performance metrics, including accuracy, precision, recall and F1 score. Overall, this project is used to improve early detection and management of hypertension, leading to better health outcomes for patients. The data will be used to collect and analyze relevant data from users, including age, gender, family history of hypertension, lifestyle factors. The collected data will be used to generate a hypertension risk score for the user.

# CONTENTS

**PAGENO**

Chapter 1: INTRODUCTION 01

* 1. [Brief Information about the Project 01](#_TOC_250019)
  2. Motivation and Contribution of Project 01
  3. [Objective of the Project 03](#_TOC_250018)
  4. Organization of Project 05

Chapter 2: LITERATURE SURVEY 06

Chapter 3: SYSTEM ANALYSIS 09

* 1. [Existing System 09](#_TOC_250017)
  2. [Proposed System 10](#_TOC_250016)
  3. [Feasibility Study 18](#_TOC_250015)
  4. [Functional Requirements 19](#_TOC_250014)
  5. [Non-Functional Requirements 20](#_TOC_250013)
  6. [Requirements Specification 22](#_TOC_250012)
     1. Software Requirements 22
     2. Minimum Hardware Requirements 22

Chapter 4: SYSTEM DESIGN 23

* 1. [Introduction 23](#_TOC_250011)
  2. [System Architecture 24](#_TOC_250010)
  3. Modules Description 26
  4. [UML diagrams 28](#_TOC_250008)
     1. Use case Diagram 28
     2. [Class Diagram 33](#_TOC_250007)
     3. [Sequence Diagram 34](#_TOC_250006)
     4. Collaborative Diagram 37
     5. [Activity Diagram 39](#_TOC_250005)

Chapter 5: TECHNOLOGY DESCRIPTION 41

* 1. [Introduction to Python 41](#_TOC_250004)
  2. [Introduction to Machine Learning 48](#_TOC_250003)

Chapter 6: SAMPLE CODE 52

Chapter 7: TESTING 58

* 1. [Introduction 58](#_TOC_250002)
  2. Sample Test case Specifications 62

Chapter 8: SCREENSHOTS 63

[CONCLUSION 7](#_TOC_250001)2

[BIBLIOGRAPHY 73](#_TOC_250000)

# LIST OF TABLES

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **TABLE NO** | **NAME OF THE TABLE** | **PAGENO** |
| 1 | 4.4.1.1 | Use case Template for Data Uploading | 31 |
| 2 | 4.4.1.2 | Use case Template for Data Processing | 31 |
| 3 | 4.4.1.3 | Use case Template for Feature Selection | 32 |
| 4 | 4.4.1.4 | Use case Template for Feature Extraction | 32 |
| 5  6 | 4.4.1.5  7.2 | Use case Template for Optimal Clusters  Sample Test Cases Specifications | 33  62 |

**LIST OF FIGURES**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **FIGURE NO** | **NAME OF THE FIGURE** | **PAGE NO** |
| 01 | 4.2 | System Architecture | 24 |
| 02 | 4.5.1 | Use Case Diagram | 28 |
| 03 | 4.5.2 | Class Diagram | 33 |
| 04 | 4.5.3 | Sequence Diagram | 34 |
| 05 | 4.5.4 | Collaboration Diagram | 37 |
| 06 | 4.5.5 | Activity Diagram | 39 |

**LIST OF SCREENS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **SCREEN NO** | **NAME OF THE SCREEN** | **PAGE NO** |
| 01 | 7.2.1 | Screenshot for System uploading dataset | 63 |
| 02 | 7.2.2 | Screenshot For System not uploading dataset | 63 |
| 03  04  05 | 8.1  8.2  8.3 | Screenshot For Execution using dataset  Screenshot for System Execution using Confusion Matrix  Screenshot for system Execution using Guassian Naïve Bayesian Algorithm | 64    65  66 |
| 06 | 8.4 | Screenshot For System Execution using Logistic Algorithm | 67 |
| 07 | 8.5 | Screenshot For System Execution using Support Vector Machine Algorithm | 68 |

|  |  |  |  |
| --- | --- | --- | --- |
| 08  09 | 8.6  8.7 | Screenshot For Distribution of Age Vs HBP  Screenshot For Distribution of Body Weight Vs HBP | 69  70 |

|  |  |  |  |
| --- | --- | --- | --- |
| 08  09 | 8.8  8.9 | Screenshot For Distribution of Cholesterol Vs HBP  Screenshot For Distribution of Height Vs HBP | 73  74 |

|  |
| --- |
| **LIST OF ABBREVATIONS**  **S.NO ABBREVATION FULL FORM**  01 PD pandas  02 NBC Naïve Bayesian Classifier  03 LR Logistic Regression  04 SVM Support Vector Machine |

# INTRODUCTION

### Brief information about the project:

Hypertension is a common health condition that has become an issue in this present world. It is part of the metabolic syndrome and a multifactorial condition in which an individual is diagnosed with systolic blood pressure ≥140 mmHg and or a diastolic pressure ≥90 mmHg. It occurs when the force of blood against the walls of the arteries is consistently too high, leading to an increased risk of heart disease, stroke, and other health problems. It is often referred as the “silent killer” because it is often having no noticeable symptom until it has caused significantly to the body. It is a prevalent condition, affecting millions of people worldwide, and is a leading cause of death and disability globally.

Hypertension prediction involves utilizing various techniques and factors to assess the likelihood of an individual developing high blood pressure. These predictors can be broadly categorized into two types: non-modifiable risk factors and modifiable risk factors.

Although hypertension cannot be cured, it can be managed through lifestyle modifications and medications. Regular blood pressure monitoring, healthy diet, regular exercise, maintaining a healthy weight, reducing stress, and avoiding smoking and excessive alcohol consumption are essential in managing and reducing the risk of complications.

* 1. **Motivation of the project:**

The motivation for a hypertension prediction project is to address the significant public health burden of hypertension and its associated complications. Hypertension affects millions of people worldwide and is a leading cause of heart disease, stroke, kidney disease, and vision loss. Early detection and prevention of hypertension can help to reduce the risk of these complications, but many people with hypertension are undiagnosed and unaware of their condition. This prediction model can be used to identify individuals at high risk of hypertension and provide them with appropriate preventative measures, including lifestyle modifications and medication. By developing a hypertension prediction model, healthcare professionals can improve the

accuracy of hypertension diagnosis and treatment, reducing the risk of complications and improving overall health outcomes. This project has the potential to significantly impact public health by identifying individuals at high risk of hypertension and helping prevent its associated complications.

##### Health Consequences:

Hypertension is a major risk factor for cardiovascular diseases such as heart disease, stroke, and kidney problems. Uncontrolled high blood pressure can lead to serious complications and even be life-threatening. By managing and preventing hypertension, individuals can reduce their risk of developing these conditions and improve their overall health and longevity.

##### Quality of Life:

Hypertension can significantly impact an individual's quality of life. It may lead to symptoms such as headaches, fatigue, dizziness, and shortness of breath. By managing blood pressure levels, individuals can experience relief from these symptoms and improve their well-being, allowing them to engage in daily activities without limitations.

##### Preventive Approach:

Addressing hypertension with a preventive approach to healthcare. By identifying individuals at risk, predicting the likelihood of developing hypertension, and implementing appropriate interventions, the focus shifts from reactive treatment to proactive prevention. This approach can lead to better health outcomes, reduced healthcare costs, and improved overall population health.

##### Reducing Healthcare Burden:

Hypertension places a substantial burden on healthcare systems worldwide. It contributes to increased healthcare utilization, hospitalizations, and healthcare costs. By effectively managing hypertension, healthcare systems can alleviate the burden on resources, redirecting them to other areas of healthcare delivery.

##### Health Consequences:

Hypertension is a major risk factor for cardiovascular diseases such as heart disease, stroke, and kidney problems. Uncontrolled high blood pressure can lead to serious complications and even be life-threatening. By managing and preventing hypertension, individuals can reduce their risk of developing these conditions and improve their overall health and longevity.

##### Health Promotion:

Hypertension prediction serves as a tool for health promotion and education. It raises awareness about the risk factors associated with high blood pressure, encourages individuals to adopt healthier lifestyles, and promotes regular blood pressure monitoring. It empowers individuals to take an active role in their health and make informed decisions to prevent hypertension.

* 1. **Objective of the project:**

##### Early detection:

The objective could be to develop a predictive model that can identify individuals who are at high risk of developing hypertension before they develop the condition. This could allow for early preventive measures to be put in place to reduce the likelihood of hypertension developing.

##### Risk stratification:

Another objective could be to develop a model that can stratify individuals based on their risk of developing hypertension. This could help healthcare providers to prioritize their interventions and resources, and target those who are at highest risk. **Treatment optimization:**

The objective could be to develop a model that can predict which treatments will be most effective for individuals with hypertension. This could help healthcare providers to tailor their treatment plans to the individual, leading to better outcomes. **Population-level prediction:**

Another objective could be to develop a model that can predict the prevalence of hypertension in a population. This could help policymakers to plan and allocate resources for prevention and treatment programs, and to monitor the effectiveness of those programs over time.

##### Prevention:

The primary objective for hypertension is to prevent its occurrence whenever possible. This involves identifying individuals at risk, implementing preventive measures, and promoting healthy lifestyles to reduce the likelihood of developing high blood pressure.

##### Reduction of Complications:

Hypertension increases the risk of developing cardiovascular diseases, kidney

problems, and other related complications. The objective is to minimize the occurrence and severity of these complications through effective blood pressure control, adherence to prescribed treatments, and ongoing monitoring of organ function.

##### Education and Awareness:

A key objective is to educate individuals about hypertension, its risk factors, and preventive measures. Increasing awareness about the importance of blood pressure monitoring, lifestyle modifications, and guarantee to treatment can empower individuals to take an active role in managing their blood pressure and overall cardiovascular health.

##### Public Health Impact:

At a broader level, the objective for hypertension is to have a positive impact on public health. This involves implementing population-level strategies such as health promotion campaigns, policy changes, and community interventions to reduce the overall burden of hypertension and its associated complications in society.

##### Blood Pressure Control:

The main objective in managing hypertension is to achieve and maintain blood pressure within a healthy range. The goal is to lower and control elevated blood pressure levels through lifestyle modifications, such as adopting a balanced diet, engaging in regular physical activity, reducing sodium intake, managing stress, and, if necessary, using medication as prescribed by healthcare professionals.

##### Regular Monitoring and Follow-up:

Regular blood pressure monitoring and follow-up appointments are essential objectives for long-term blood pressure management. Monitoring allows for the assessment of treatment effectiveness and adjustment of treatment plans as needed. Follow-up appointments provide an opportunity for healthcare professionals to address any concerns, reinforce lifestyle modifications, and optimize blood pressure control.

By working towards these objectives, healthcare systems, healthcare providers, individuals, and communities can effectively prevent, manage, and reduce the impact of hypertension, improving health outcomes and quality of life for individuals at risk or already diagnosed with high blood pressure.

### Organization of the project:

* **Chapter 2 Literature Survey:** This Chapter consists of background of the project and possible approaches, Introduction, and comparison.
* **Chapter 3 System Analysis:** The description of the current system, the planned system, and the required specifications make up the majority of this chapter.
* **Chapter 4 System Design:** This chapter consists of modules description and algorithms with example and use case diagrams, class diagrams, sequence diagrams, Collaboration diagrams and activity diagrams. These diagrams will talks about the process by using these diagrams we can have an idea about the process that is done in the project.
* **Chapter 5 Technology Description:** This chapter mainly consists of the technology description of this project.
* **Chapter 6 Sample Code:** This chapter consists of sample code for the few modules.
* **Chapter 7 Testing:** This chapter mainly consists of testing techniques and test cases for modules.
* **Chapter 8 Screen Shots:** This chapter mainly consists of output screens of this project. It shows the main part of the project by having the predictions of each column and making the graphs through the actual and predicted values of the dataset.
* **Chapter 9 Conclusion:** Main Conclusion of the project is to predict the hypertension of the people. By finding the accuracy of the particular algorithm. We can decide the perfect algorithm by knowing the accuracy of the algorithm, can implement the algorithm for further findings in it.

# LITERATURE SURVEY

Predicting the Risk of Hypertension Based on Several Easy-to-Collect Risk Factors A Machine Learning Method done by author Huanhuan: Hypertension is a widespread chronic disease. Risk prediction of hypertension is an intervention that contributes to the early prevention and management of hypertension. The implementation of such intervention requires an effective and easy-to-implement hypertension risk prediction model. This study evaluated and compared the performance of four machine learning algorithms on predicting the risk of hypertension based on easy-to-collect risk factors. A dataset of 29,700 samples collected through a physical examination was used for model training and testing.

Firstly, we identified easy-to-collect risk factors of hypertension, through univariate logistic regression analysis. Then, based on the selected features, 10-fold cross-validation was utilized to optimize four models, random forest (RF), CatBoost, MLP neural network and logistic regression (LR), to find the best hyper-parameters on the training set.

Finally, the performance of models was evaluated by AUC, accuracy, sensitivity and specificity on the test set. The experimental results showed that the RF model outperformed the other three models, and achieved an AUC of 0.92, an accuracy of 0.82, a sensitivity of 0.83 and a specificity of 0.81. In addition, Body Mass Index (BMI), age, family history and waist circumference (WC) are the four primary risk factors of hypertension. These findings reveal that it is feasible to use machine learning algorithms, especially RF, to predict hypertension risk without clinical or genetic data. The technique can provide a non-invasive and economical way for the prevention and management of hypertension in a large population.

Predicting hypertension using machine learning Findings from Qatar Biobank Study done by author Latifa A. AlKaabi, Lina S. Ahmed: Hypertension, a global burden, is associated with several risk factors and can be treated by lifestyle modifications and medications. Prediction and early diagnosis is important to prevent related health complications. The objective is to construct and compare predictive models to identify individuals at high risk of developing hypertension without the need for invasive clinical procedures.

Development and validation of a hypertension risk prediction model and construction of a risk score in a Canadian population done by author [Mohammad Ziaul Islam Chowdhury](https://www.nature.com/articles/s41598-022-16904-x#auth-Mohammad_Ziaul_Islam-Chowdhury), [Alexander A. Leung](https://www.nature.com/articles/s41598-022-16904-x#auth-Alexander_A_-Leung): Identifying high-risk individuals for targeted intervention may prevent or delay hypertension onset. We developed a hypertension risk prediction model and subsequent risk sore among the Canadian population using measures readily available in a primary care setting. A Canadian cohort of 18,322 participants aged 35–69 years without hypertension at baseline was followed for hypertension incidence, and 625 new hypertension cases were reported. At a 2:1 ratio, the sample was randomly divided into derivation and validation sets. In the derivation sample, a Cox proportional hazard model was used to develop the model, and the model's performance was evaluated in the validation sample. Finally, a risk score table was created incorporating regression coefficients from the model.

The multivariable Cox model identified age, body mass index, systolic blood pressure, diabetes, total physical activity time, and cardiovascular disease as significant risk factors (*p* < 0.05) of hypertension incidence. The variable sex was forced to enter the final model. Some interaction terms were identified as significant but were excluded due to their lack of incremental predictive capacity. Our model showed good discrimination (Harrel’s C-statistic 0.77) and calibration (Grønnesby and Borgan test, χ2�2 statistic = 8.75, *p* = 0.07; calibration slope 1.006). A point-based score for the risks of developing hypertension was presented after 2-, 3-, 5-, and 6 years of observation. This simple, practical prediction score can reliably identify Canadian adults at high risk of developing incident hypertension in the primary care setting and facilitate discussions on modifying this risk most effectively.

A review of machine learning in hypertension detection and blood pressure estimation based on clinical and physiological data done by author Erick Martinez-Ríos a, Luis Montesinos, Tecnologico de Monterrey, School of Engineering and Sciences, Mexico City, 14380, Mexico: The use of [machine learning](https://www.sciencedirect.com/topics/computer-science/machine-learning) techniques in medicine has increased in recent years due to a rise in publicly available datasets.

These techniques have been applied in high blood pressure studies following two approaches: hypertension stage classification based on clinical data and blood pressure estimation based on related [physiological signals](https://www.sciencedirect.com/topics/computer-science/physiological-signal). This paper presents a literature review on such studies. We aimed to identify the best practices, challenges, and opportunities in developing machine learning models to detect hypertension or estimate blood pressure using clinical data and physiological signals. Hence, we identified and examined the [machine learning techniques](https://www.sciencedirect.com/topics/computer-science/machine-learning-technique), publicly available datasets, and predictors used in previous studies.

The feature selection techniques used to reduce model complexity are also reviewed. We found a lack of studies combining socio-demographic or clinical data with physiological signals, despite the correlation of blood pressure with photoplethysmography waveforms and variables such as age, gender, body mass index, and heart rate. Therefore, there is an opportunity to increase model performance by using both types of data for hypertension detection or blood pressure monitoring.

# SYSTEM ANALYSIS

### Existing System:

In the existing system, I have searched some findings for my project, at that time: The Qatar Biobank Study is a large-scale research initiative that aims to investigate the health and lifestyle of the Qatari population. One of the areas of focus is the prediction of hypertension using machine learning techniques. In a study published in the Journal of Medical Internet Research, researchers used data from the Qatar Biobank Study to develop and validate a machine learning model for predicting hypertension.

The results of the study showed that the machine learning model was able to accurately predict hypertension, with an overall accuracy of 86%. The model was also able to identify specific risk factors that were most strongly associated with hypertension, including age, body mass index, smoking status, and diabetes.

The Qatar Biobank is a national health initiative in Qatar that collects and stores biological samples, health information, and lifestyle data from Qatar's population. While the primary focus of the Qatar Biobank is to support medical research and advance personalized medicine, it also provides a valuable resource for studying various health conditions, including hypertension.

### Disadvantages:

##### False Positives and Negatives:

Predictive models for hypertension may not always be 100% accurate, resulting in false positives (predicting hypertension when it doesn't occur) or false negatives (failing to predict hypertension when it does occur). This can lead to unnecessary interventions for some individuals or a failure to identify those who may benefit from early intervention.

##### Overdiagnosis and Overtreatment:

There is a risk of overdiagnosis and overtreatment when relying solely on predictive models. Some individuals may be labelled as having a high risk of developing hypertension and receive unnecessary medical treatment or interventions, causing undue stress and healthcare costs.

##### Impact on Organ Systems:

Prolonged high blood pressure can damage and strain various organs and systems in the body. It can lead to thickening and narrowing of blood vessels, weakening of the heart muscle, kidney dysfunction, and impaired blood flow to vital organs. These effects can contribute to the development of cardiovascular diseases and other related conditions.

The risks and complications associated with the condition can be observed

through particular diagnosis, treatment, and lifestyle modifications. Regular monitoring, committing to medical advice, and proactive management can significantly reduce the impact of hypertension on an individual's health and being in a good way.

### Proposed System:

In the proposed system, I have done the process by using specific machine learning algorithms like Gaussian Naïve Bayes, Logistic Regression. These algorithms had different process and steps that which are worked on the dataset. The dataset is trained and tested by these algorithms and performs the result by producing the accuracy of different percentages. It calculates the precision, recall and f1-score values. Finally, the HBP values show least as the HBP levels should be as less as compared to both the algorithms. As I want to get a brief of both the algorithms and steps of it.

* Gather relevant data from multiple sources, including electronic health records patient medical history, lifestyle and physiological measurements.
* Cleanse and preprocess the collected data to ensure consistency, remove outliers, handle missing values, and normalize the data for further analysis.
* Choose appropriate machine learning algorithms for hypertension prediction based on the nature of the data and the problem. Commonly used models for classification tasks include logistic regression, support vector machines (SVM), and neural networks.
* Split the preprocessed data into training and testing sets. Trains selected machine learning model on the training set and evaluated its performance by using appropriate evaluation metrics such as accuracy, precision, recall, and F1-score.
* Validate the trained model on an independent validation dataset to ensure its generalizability and effectiveness in real-world scenarios. This step helps to identify potential overfitting or underfitting issues.

### Advantages:

##### Preventive Measures:

Predicting hypertension provides an opportunity for preventive measures. By identifying modifiable risk factors such as poor diet, sedentary lifestyle, smoking, or excessive alcohol consumption, individuals can make necessary lifestyle changes to lower their blood pressure and reduce the risk of developing hypertension.

##### Targeted Interventions:

Hypertension prediction can help healthcare providers target interventions and treatment plans for high-risk individuals. By identifying those most likely to develop hypertension, healthcare professionals can focus their efforts on providing education, counselling, and support to prevent or manage the condition effectively.

##### Reduced Health Risks:

Early identification and management of hypertension can significantly reduce the risk of associated health complications. By controlling blood pressure levels, individuals can lower their risk of heart disease, stroke, kidney problems, and other cardiovascular conditions.

### Gaussian Naïve Bayes:

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. It is mainly used in text classification that includes a high-dimensional training dataset. Naïve Bayes Classifier is one of the simplest and most effective Classification algorithms which helps in building fast machine learning models that can make quick predictions. Because there is a significant advantage with NB. Since it is a probabilistic model, the algorithm can be coded up easily and the predictions made quick. Real-time quick. In this, there are different steps to implement this algorithm:

##### Loading Initial Libraries:

I will start by loading some initial libraries to load and visualize the dataset.

##### Importing Dataset:

Now I will upload the dataset that I have obtained from previous patient’s data performing our Naive Bayes classification.

##### Exploring Dataset:

Let us take an initial look into the dataset with the help of head() function.

1. **Visualizing Dataset:**

To visualize the dataset, we will first split the dataframe into two parts. The first part contains information about patients who have no high blood pressure, and the second part contains those who had high blood pressure.

##### Preprocessing:

Now we will be assigning a value of ‘1’ to high blood pressure and ‘0’ to no blood pressure.

##### Data Normalization:

To improve the efficiency of the model it is always good to normalize data to bring them into a common scale.

##### Test Train Split:

Now with the help of sklearn library train\_test\_split module, we will split the dataset into training and testing parts.

##### Sklearn Gaussian Naive Bayes Model:

Now we will import the Gaussian Naive Bayes module of SKlearn GaussianNB and create an instance of it. We can pass x\_train and y\_train to fit the model.

##### Accuracy:

The following accuracy score on the test data shows how well our model Sklearn Gaussian Naive Bayes model has performed for predicting hypertension.

These algorithm steps are used in this project, by having these steps we may know what are doing and how it is processing in it. These are useful for knowing about the algorithm and steps carefully. Now, we can discuss the advantages and disadvantages of this algorithm.

### Advantages:

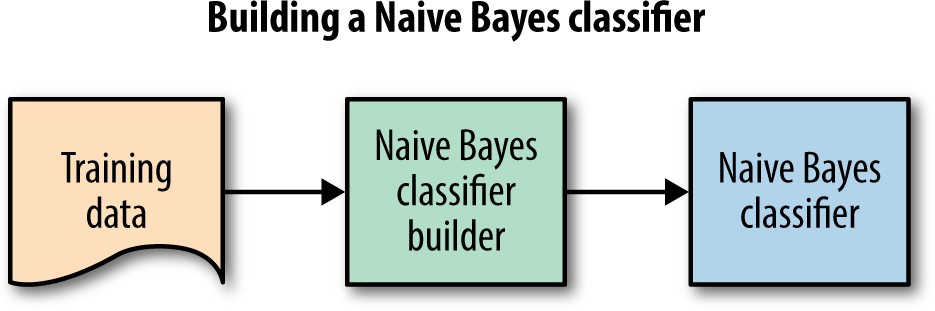
* + Very fast – no iterations since the probabilities can be directly computed. So this technique is useful where speed of training is important.
  + If the conditional Independence assumption holds, it could give great results.
  + Gaussian Naive Bayes is a simple and easy-to-understand algorithm. It is computationally efficient and has low memory requirements, making it suitable for large datasets.
  + The training time of Gaussian Naive Bayes is typically faster compared to more complex algorithms like decision trees or neural networks.
  + Gaussian Naive Bayes performs well even with a high number of features. It handles the curse of dimensionality reasonably well.
  + Effective with Small Datasets: Gaussian Naive Bayes can perform well even with limited training data.

### Drawbacks:

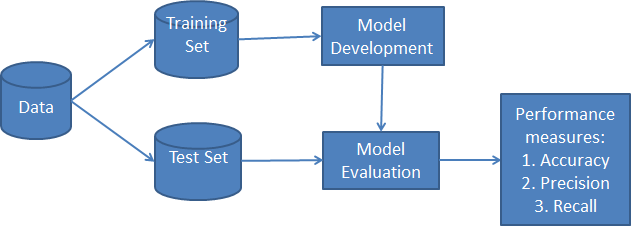
* + Conditional Independence Assumption does not always hold. In most situations, the feature shows some form of dependency.
  + Zero probability problem: When we encounter words in the test data for a particular class that are not present in the training data, we might end up with zero class probabilities.
  + Strong Independence Assumption: The main limitation of Gaussian Naive Bayes is its assumption of feature independence. This assumption may not hold in real- world scenarios where features are correlated, leading to suboptimal performance.
  + Sensitivity to Feature Distribution: Gaussian Naive Bayes assumes that continuous features follow a Gaussian distribution. If the distribution of features significantly deviates from Gaussian, the algorithm's performance may be negatively affected.
  + Lack of Model Complexity: Gaussian Naive Bayes is a simple algorithm and

may not capture complex relationships or interactions between features. It may struggle with datasets that exhibit intricate patterns or non-linear relationships.

* + Class Imbalance: Gaussian Naive Bayes may struggle with imbalanced datasets, where the number of instances in different classes varies significantly. Since it relies on the probability estimation of each class, imbalanced classes can skew the model's predictions.



The above figure shows that the Naïve Bayesian classifier trains the data of the dataset project by having the information of specified columns and rows in it. It processes and trains the dataset and classifies the patients data in it and performs the particular functions in it. It builds the data and gives some predicted values and actual values in the data processing. After building the data information it classifies all the values in it and takes it to the next process.



From this figure, first it gathers the data from the dataset and maintains the data in it. After gathering the data then it processes to trains the data completely and make

some values in it. After that, we will get some values in it and make values to some extent, after training the dataset we have to test the dataset and process the dataset and it develops the models as what the values get as an outcome. It evaluates the data and develops the outcomes, and it performs the measures by having accuracy, precision and recall.

### Logistic Regression:

Logistic regression is a supervised machine learning algorithm mainly used for classification tasks where the goal is to predict the probability. which analyzes the relationship between a set of independent variables and the dependent binary variables. It is a powerful tool for decision-making.

**Independent variables:** The input characteristics or predictor factors applied to the dependent variable’s predictions.

**Dependent variable:** The target variable in a logistic regression model, which we are trying to predict.

**Logistic function:** The formula used to represent how the independent and dependent variables relate to one another. The logistic function transforms the input variables into a probability value between 0 and 1, which represents the likelihood of the dependent variable being 1 or 0.

##### Import Packages, Functions, and Classes:

First, you must import Matplotlib for visualization and NumPy for array operations. You’ll also need Logistic Regression, classification\_report(),

and confusion\_matrix() from scikit-learn.

##### Get Data:

Import the logistic regression module from the scikit-learn library and create an instance of the logistic regression class.

##### Create a Model and Train It:

Once you have the input and output prepared, you can create and define your classification model. You’re going to represent it with an instance of the class Logistic Regression.

##### Make predictions:

For a given test data point, calculate the likelihood of each class label using the Naive Gaussian assumption, which assumes that the features are independent given the class label. Multiply the likelihoods for each feature together to get the joint

probability of the features given the class label, and then use Bayes' theorem to calculate the posterior probability of each class label given the features. The class label with the highest posterior probability is the predicted label for the data point.

1. **Evaluate the Model:** Using the testing set, evaluate the accuracy of the model by comparing the predicted labels to the true labels. Compute metrics such as accuracy, precision, recall, F1-score.

### Advantages:

1. **Simplicity:** Logistic regression is a relatively simple and easy to understand algorithm.
2. **Efficiency:** Logistic regression can be trained quickly, even on large datasets, making it a good choice for problems with a large number of features or a large number of training examples.
3. **Scalability:** Logistic regression scales well to large datasets and can be used for online learning, where new data is continuously added to the model.
4. **Probability estimates:** Logistic regression outputs a probability estimates for each allowing for more nuanced predictions and decision-making.
5. **Robustness:** Logistic regression is less prone to overfitting than other classification algorithms, especially when the number of features is small compared to the number of training examples.
6. **Feature selection:** Logistic regression can be used to identify important features in the dataset, which can help to improve model performance.

### Disadvantages:

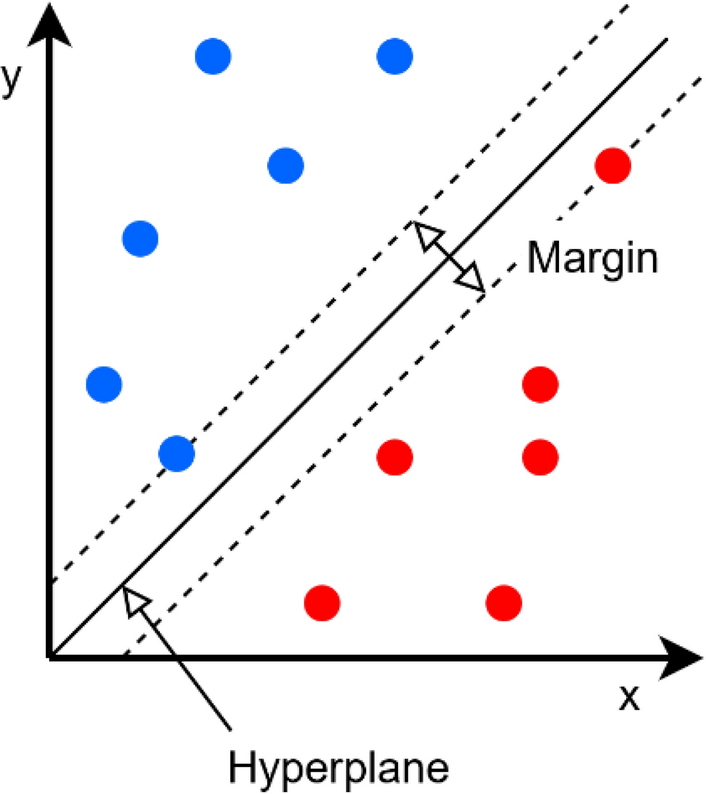
1. **Limited flexibility:** Logistic regression assumes a linear decision boundary between classes, which may not be appropriate for all datasets. It also assumes that the relationship between the features and the target variable is log-linear, which may not always hold in practice.
2. **Sensitive to outliers:** Logistic regression is sensitive to outliers, which can skew the decision boundary and affect model performance.
3. **Limited to binary classification:** Logistic regression is only suitable for binary classification problems, where there are two classes or labels. It cannot be directly used for multiclass classification without modifications.
4. **Requires feature engineering:** Logistic regression relies on carefully engineered

features, which can be time-consuming and require domain expertise.

1. **Can suffer from multicollinearity:** If the features are highly correlated with each other, logistic regression can suffer from multicollinearity, which can make the model unstable and difficult to interprete.
2. **Assumes independence of observations:** Logistic regression assumes that each observation is independent of the others, which may not be true for some datasets.
3. **Limited Handling of Missing Data:** Logistic regression requires complete data for all variables included in the model. It does not handle missing data inherently and may require imputation or data preprocessing techniques to handle missing values effectively.

### Support Vector Machine (SVM):

Support Vector Machine (SVM) is a powerful algorithm used in machine learning for classification and regression analysis. SVM has been used in various applications including hypertension prediction. The basic idea behind SVM is to find a hyperplane in a high-dimensional space that maximally separates the data points into different classes. In hypertension prediction, SVM can be used to classify patients as hypertensive or non-hypertensive based on their health records. SVM uses a set of training data to learn the characteristics of hypertensive and non-hypertensive patients. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or Hyperplane:



SVM can be used for both linear and non-linear classification problems. In

linear SVM, the hyperplane is a linear function of the input features, while in non- linear SVM, the input features are transformed using a kernel function to a higher- dimensional space where a linear classifier can be used.

### Advantages:

* SVM classifiers perform well in high-dimensional space and have excellent accuracy. SVM classifiers require less memory because they only use a portion of the training data.
* SVM performs reasonably well when there is a large gap between classes.
* High-dimensional spaces are better suited for SVM.
* When the number of dimensions exceeds the number of samples, SVM is useful.
* SVM uses memory effectively.

### Disadvantages:

* SVM requires a long training period; as a result, it is not practical for large datasets.
* The inability of SVM classifiers to handle overlapping classes is another drawback.
* Large data sets are not a good fit for the SVM algorithm.
* When the data set contains more noise, such as overlapping target classes, SVM does not perform as well.
* The SVM will perform poorly when the number of features for each data point is greater than the number of training data samples.

### Feasibility Study:

The feasibility study is the availability of data is crucial to building a hypertension prediction model. Data on individuals who have been diagnosed with hypertension and those who have not, as well as their medical history, lifestyle habits, and demographics, are essential for building a predictive model. The quality and quantity of data need to be sufficient to develop an accurate model. Availability of such data can be a challenge and requires data collaboration with healthcare providers, hospitals, and research institutions. There are aspects in the feasibility study portion of the preliminary investigation, they are shown in below as:

* **ECONOMIC FEASIBILITY**
* **OPERATIONAL FEASIBILITY**
* **TECHNICAL FEASIBILITY**

### Economic Feasibility:

The economic feasibility is Cost of Hardware and Software: The cost of hardware and software required to develop and deploy the hypertension prediction model can be substantial. The hardware requirements can include high-performance computing systems, data storage, and visualization tools. The software requirements can include data analytics software, machine learning libraries, and programming tools. The cost of hardware and software can be minimized by using cloud-based services, open-source software, and leveraging existing infrastructure.

### Operational Feasibility:

##### Operational Feasibility Integration with Clinical Workflow:

To be effective, the hypertension prediction model must be integrated into clinical workflow seamlessly. The model's output must be presented in a way that is easily understandable and actionable by healthcare providers. It is essential to involve healthcare providers in the design and testing of the model to ensure its usability and effectiveness in clinical practice.

##### Training and Support:

To ensure the successful adoption of the hypertension prediction model, healthcare providers and staff must be trained on how to use the model and interpret its output. Additionally, ongoing support and maintenance are necessary to ensure the model's continued effectiveness and accuracy.

### Technical Feasibility:

##### Model Development:

Model development involves training the selected algorithms on the prepared data and evaluating their performance. It is important to ensure that the models are accurate, reliable, and robust, and to address any issues that arise during development.

##### Model Maintenance and Updates:

The hypertension prediction model must be maintained and updated to ensure its continued accuracy and reliability. This process may require ongoing monitoring, data updates, and algorithm updates to address changing patient populations and medical practices.

##### Software Development and Integration:

Building a hypertension prediction system requires software development expertise and the ability to integrate machine learning models into the system's architecture. It may involve implementing data preprocessing pipelines, feature extraction, model training, and deploying the models within the system infrastructure. **Real-Time Monitoring and Alerts:**

If real-time hypertension risk monitoring and alerts are required, the technical feasibility will depend on the capability to continuously process and analyze patient data and trigger timely notifications or alerts based on the predicted risk levels.

### Functional Requirements:

In machine learning, the functional requirements characterize an element or a function of a software framework or its part. A function is portrayed as an arrangement of sources of info, inputs and outputs the conduct, and yields. These functional requirements might be counts, specialized points of interest, data and information control and handling, and other explicit usefulness that characterize what a system should achieve. Behavioral requirements portraying every one of the situations where the framework utilizes the functional requirements are caught being used cases.

* The system should support continuous monitoring of patient data and provide alerts or notifications when the predicted risk levels change significantly.
* It should enable timely interventions and follow-up actions for patients at increased risk.
* The system should incorporate mechanisms to assess the performance of the predictive models regularly.
* It should calculate metrics such as accuracy, sensitivity, specificity, and area under the receiver operating characteristic curve (AUC-ROC) to evaluate the effectiveness of the hypertension prediction.
* Collect patients’ data from the previous searching’s.
* Train the patient’s data.
* Divide them based on Characteristics.

### Non-Functional Requirements:

#### Efficiency:

It is a measurable concept and can often be expressed as a percentage of results that could ideally be expected**.**

#### Maintainability:

Maintainability is nothing but how the system which is developed can update itself concerning time, how the system corrects the defects and bugs which occurred after deployment and how it will meet users’ new requirements.

Since the program is developed using python it is easy to detect and correct the based on the user requirement just by adding required files or APIs for the existing software.

#### Scalability:

This will give an idea about how the system acts or how the system gives its throughput when the load of the input data is changed. System can work normally under any amount of input handwritten data.

#### Portability:

Portability is one of the important features of non-functional requirements, it will give the idea about whether the software needs to be rewritten when software moves from one device to another. Project uses python so the project can be easily installed and can be used in any other platform.

#### Usability:

The system should be user-friendly, with a well-designed interface that allows healthcare professionals or users to easily interact with the system. It should provide clear and understandable explanations of predictions and be intuitive to navigate.

#### Privacy and security:

The system should adhere to strict privacy regulations and maintain the confidentiality of patient data. It should implement appropriate security measures to protect sensitive information from unauthorized access, breaches, or misuse.

#### Accuracy:

The prediction system should have a high level of accuracy in identifying individuals at risk of hypertension. The accuracy of the prediction model should be measured by appropriate metrics such as sensitivity, specificity, positive predictive value, and negative predictive value.

#### Performance:

The system should be able to handle a large volume of data efficiently and provide timely predictions. It should have low latency and response time to ensure quick and efficient processing of patient data.

### Requirements Specification:

* + 1. **Hardware Requirements:**

Processor : Intel Core i5

Speed : 2.2 GHz

RAM : 8 GB

Hard Disk : 512 GB

### Software Requirements:

Operating System : Windows 10 Technology : Pytho3.9

IDE : Jupyter Notebook

# SYSTEM DESIGN

### Introduction:

The purpose of the design phase is to plan a solution to the problem specified by the requirement document. According to the World Health Organization, hypertension or high blood pressure (BP) is a global public health issue. This health condition affects 1.1 billion people worldwide, with two-thirds living in low and middle-income countries. Moreover, hypertension increases the risk of heart attack, heart failure, kidney disease, coronary heart disease, diabetes, and strokes. In 2013, high BP accounted for at least 45% of deaths due to heart disease and 51% of deaths due to stroke.

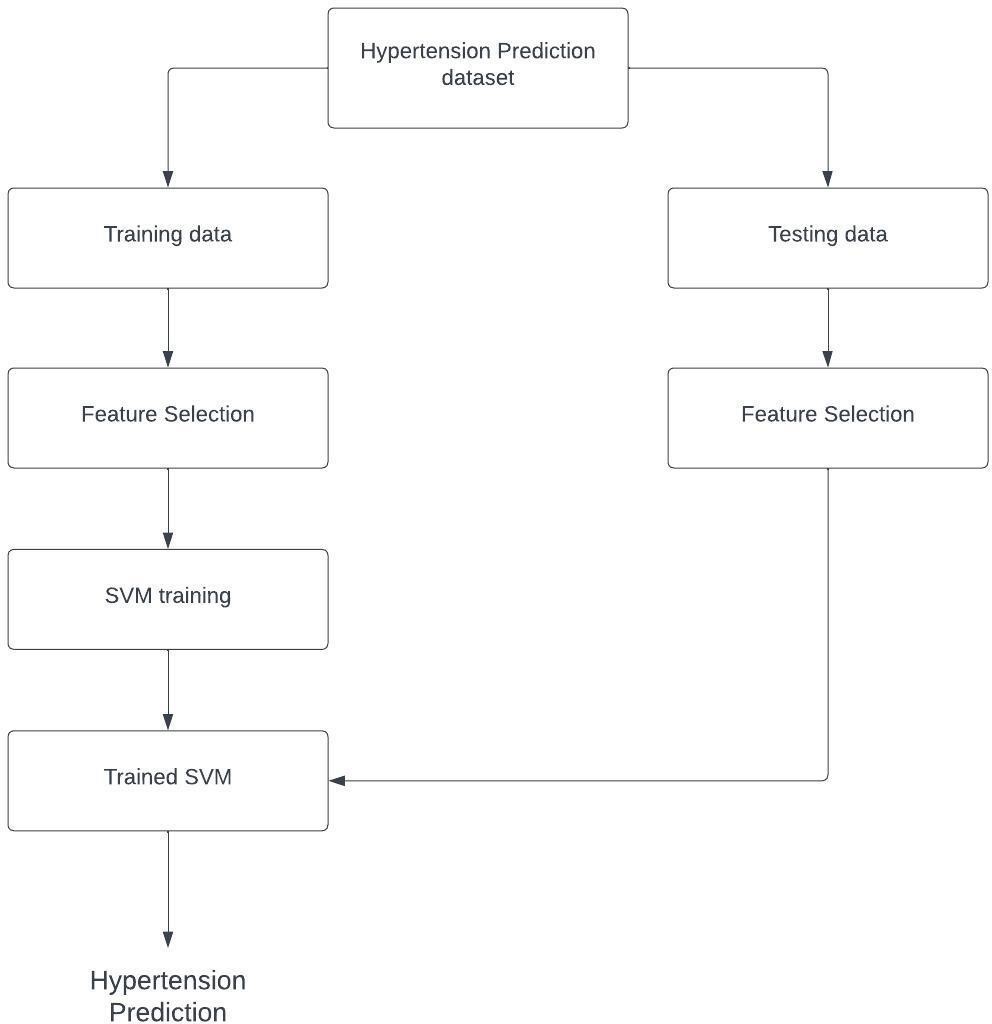
Moreover, managing this medical condition is challenging and costly. The diagnosis of hypertension is based on arterial BP readings measured with a sphygmomanometer, the standard method for measuring BP, which has an inflatable cuff, a mechanism of inflation that can be manually or automatically operated, and a mercury manometer. Table 1 shows the complete classification of BP values according to the systolic and diastolic BP readings. Some of the most common risk factors associated with hypertension include age, gender, body mass index (BMI), obesity, stress, lipoproteins, cholesterol, physical activity, smoking, and family history. Medical evidence suggests that early detection of hypertension, correction of life habits, and tight control could reduce its further development and consequences. Nevertheless, most people with hypertension have no signs or symptoms, which is the reason it is often referred to as a “silent killer”. A few people with hypertension could suffer dizziness, chest pain, headaches, breathing difficulties, nosebleeds, or heart palpitations. However, these signs and symptoms are not specific to this condition and usually do not occur until hypertension has reached a severe stage.

Therefore, the detection and monitoring of hypertension is still an open research topic, especially in lower-income countries with less pervasive and proactive healthcare services. The amount of clinical data available in electronic health records (EHRs) has enabled machine learning (ML) techniques to detect and monitor different medical conditions or diseases, hypertension not being the exception. The ML used in the medical field goes from traditional methods such as logistic and linear

regression to more complex techniques such as artificial neural networks (ANN) with their diverse architectures and characteristics. The resulting ML models are meant to provide medical experts with a tool to support clinical decision-making. Physiological signals, either raw or some features extracted from them, are also as frequently used as input data to produce ML models in medicine. Electrocardiography (ECG) and photoplethysmography (PPG) have been used extensively for BP estimation using ML models. They owe this popularity to two main features. First, their relative easiness of implementation on wearable devices (e.g., smartwatches) facilitates continuous BP monitoring through adequate processing.

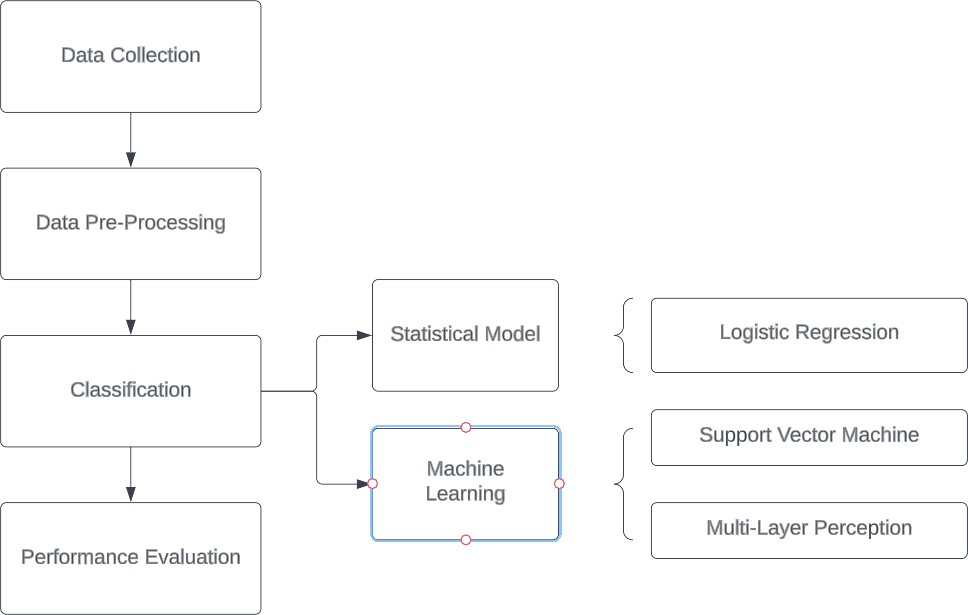
## SYSTEM ARCHITECTURE:

Initially we will see the dataset and then we will perform exploratory data analysis which deals with the missing data, duplicates values, and null values. By using Logistic Regression (LR), Naïve Bayesian Classifier (NBC) and Support Vector Machines (SVM).



The hypertension prediction system consists of the following components:

* Data Collection: Collect relevant health data from the individual.
* Data Pre-processing: Clean and pre-process the collected data.
* Feature Extraction: Extract relevant features from the pre-processed data.
* Feature Transformation: Transform the extracted features into a higher- dimensional space using a kernel function.
* Model Training: Train the SVM model using the transformed features and a labelled dataset.
* Model Evaluation: Evaluate the trained SVM model's performance on a separate test dataset.
* Model Optimization: Optimize the SVM model parameters to improve its performance.
* Deployment: Deploy the trained SVM model as an application or service.
* User Interface: Provide a user-friendly interface for the user to input their health data and receive a hypertension prediction.
* Data Storage: Store the health data of the individuals in a secure and scalable data storage system.
* Monitoring and Maintenance: Monitor the system's performance and maintain the model's accuracy.



The system architecture for logistic regression is:

* Data Collection: Collect relevant health data from the individual.
* Data Pre-processing: Clean and pre-process the collected data.
* Feature Extraction: Extract relevant features from the pre-processed data.
* Model Training: Train the logistic regression model using the extracted features and a labelled dataset.
* Model Evaluation: Evaluate the trained logistic regression model's performance on a separate test dataset.
* Model Optimization: Optimize the logistic regression model parameters to improve its performance.
* Deployment: Deploy the trained logistic regression model as an application or service.
* User Interface: Provide a user-friendly interface for the user to input their health data and receive a hypertension prediction.
* Data Storage: Store the health data of the individuals in a secure and scalable data storage system.
* Monitoring and Maintenance: Monitor the system's performance and maintain the model's accuracy.

## MODULE DESCRIPTION:

* Data Collection
* Data Pre-processing
* Data Cleaning

**Data Collection:**

It is a process of collecting the relevant data to your based criteria and this will be taken from the UCI Machine Learning Repository that is used for various data collections and using this module we can perform further actions like data extraction, selection etc.

In machine learning as it forms the foundation for training and developing accurate and reliable models. The quality, diversity, and representativeness of the collected data directly impact the performance and generalization capabilities of the machine learning algorithms.

**Data Pre-Processing:**

Data preprocessing is a critical step in machine learning that involves transforming

raw data into a format that can be effectively utilized by machine learning algorithms. It aims to improve data quality, remove inconsistencies, handle missing values, and prepare the data for model training.

Prior to using the data for machine learning, perform preprocessing steps to clean, transform, and standardize the collected data. This includes handling missing values, removing duplicates, normalizing or scaling numerical features, and encoding categorical variables into suitable representations for the machine learning algorithms. In this stage, the module performs data cleaning and preprocessing tasks to ensure the quality and consistency of the input data. It involves tasks such as removing duplicates, handling missing values, normalizing data, and transforming categorical variables into a suitable format for analysis.

* It consists of some basic steps like importing dataset, libraries, and finding missing values.
* This module provides the basic operations on data and provides effective output.
* Based on the analysis of data the outcome appears in this project we use of seaborn package to visualize the data efficiently and effectively.

**Essential Libraries:**

* + Pandas
  + Matplotlib
  + NumPy
  + sklearn
  + seaborn

**Data Cleaning:**

It is a process of removing the redundancy or duplicated data in the dataset to avoid the performance of the algorithm and there will be a no change in the performance and output will be generated effectively. By detecting the NULL values in the dataset to avoid any conflicts that will affect the output, by removing the null values or finding the missing values can improve the performance. Data cleaning, also known as data preprocessing or data cleansing, is an essential step in machine learning to ensure the quality, consistency, and reliability of the data used for training and model development.

## UML DIAGRAMS:

The unified modeling language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML have two separate views that depict the system from distinctly different angles are used to show it. UML is built using two distinct domains. UML Analysis modelling, which concentrates on the system's user model and structural model views. Behavioral modelling, implementation modelling, and environmental model views are the focus of UML design modelling. The following types are used to categorize them.

* Use case Diagram
* Class Diagram
* Sequence Diagram
* Collaboration Diagram
* Activity Diagram

## USE CASE DIAGRAMS:

Use case diagrams show the actors—the users who interact with the system— and theuse cases—the functionality—that the system offers. They also show how the actors and use cases are related. To express the system's high-level requirements, use cases are employed in the analysis phase of software development. Use Case Diagrams' main objectives include the following:

* + - * Providing a high-level view of what the system does.
* Identifying the users ("actors") of the system.
* Determining areas needing human-computer interfaces.

### Graphical Notation:

The basic components of Use Case diagrams are:

* Actor
* Use Case
* Association

### Actor:

An Actor, as mentioned, is a user of the system, and is depicted using a

stick figure. The role of the user is written beneath the icon. Actors are not limited to humans. If a system communicates with another application, and expects input or delivers output, then that application can also be considered an actor.

A picture containing circle  Description automatically generated

Actor role name

### Use Case:

A Use Case is functionality provided by the system; Use Cases are

depicted with an ellipse. The name of the use case is written within the ellipse.



Use Case Name

**Directed Association:**

These associations are used to link actors with use case and indicate that an actor participates in the use case in some form.

Behind each Use Case is a series of actions to achieve the proper functionality, as well as alternate paths for instances where validation fails, or errors occur. These actions can be further defined in a Use Case description. Because this is not addressed in UML, there are no standards for Use Case descriptions. However, there are some common templates can follow, and whole books on the subject writing of Use Case description.

A picture containing text, diagram, line, plot

Description automatically generated

### Description:

The above diagram represents the use case diagram of the project, normalization of duplicate records from multiple sources. The above use case diagram contains admin, publisher, end users as actors.

### Use case Template:

|  |  |
| --- | --- |
| **Use case name** | Data Uploading |
| **Participating actors** | User |
| **Flow of events** | User should login to the system.  User can browse and upload the dataset. |
| **Entry condition** | It searches for the data where it is located and load. |
| **Exit condition** | Dataset is successfully uploaded. |

**Table 4.4.1.1: Use case template for Upload dataset**

|  |  |
| --- | --- |
| **Use case name** | Data processing |
| **Participating actors** | User |
| **Flow of events** | It is a subpart in data preprocessing. Under this some operations are applied to handle unnecessary data like duplicates and missing values. |
| **Entry condition** | The file consists of features. |
| **Exit condition** | Noticing how much percentage of values are missing in all features. |

**Table 4.4.1.2: Use case template for Data processing**

|  |  |
| --- | --- |
| **Use case name** | Feature selection |
| **Participating actors** | ML engine |
| **Flow of events** | It is process of reduces the number of input values in the model. |
| **Entry condition** | The numerical input has been taken from user. |
| **Exit condition** | It will give successful results. |

**Table 4.4.1.3: Use case template for Feature selection**

|  |  |
| --- | --- |
| **Use case name** | Feature extraction |
| **Participating actors** | ML engine |
| **Flow of events** | Describes large set of data where it reduces number of resources while performing. |
| **Entry condition** | We will extract the features from the dataset. |
| **Exit condition** | It will give successful results. |

**Table 4.4.1.4: Use case template for Feature Extraction**

|  |  |
| --- | --- |
| **Use case name** | Optimal Clusters |
| **Participating actors** | ML engine |
| **Flow of events** | As there should be finding out the optimal clusters to visualize the data and it’s necessary to find out the clusters |
| **Entry condition** | We will find the optimal clusters based on pattern in data |
| **Exit condition** | It will find the clusters |

**Table 4.4.1.5: Use case template for Optimal Clusters**

## CLASS DIAGRAM:

The basic structure of object-oriented modelling is the class diagram. It is used for both precise modelling that converts the models into programming code and for general conceptual modelling of the structure of the application. Data modelling can also employ class diagrams. The major objects, interactions, and classes that need to be programmed are all represented by the classes in a class diagram. The diagram shows a class with three sections., classes is represented with boxes which contain three parts:

* + - * The upper part holds the name of the class.
      * The middle part contains the attributes of the class.
      * The bottom part gives the methods or operations the class can take or undertake.

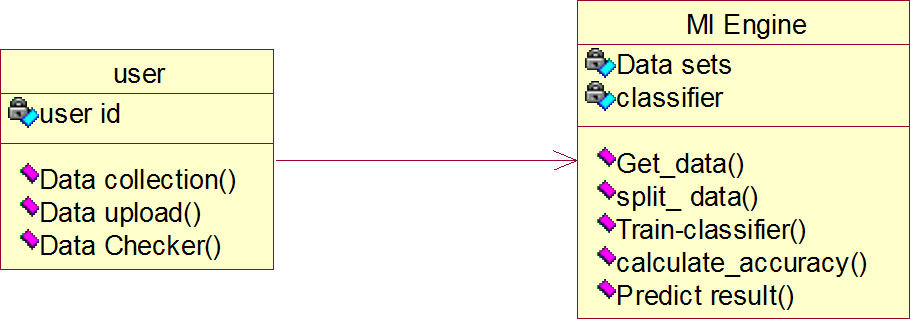
**Graphical Notation**: The elements on a Class diagram are classes and the relationships between them.

**Class:** Classes are the building blocks in object-oriented programming. A Class is depicted using a rectangle divided into three sections. The top section is the name of the Class. The middle section defines the properties of the class. The bottom section lists the methods of the class.



**Association:** An Association is a generic relationship between two classes and is modeled by a line connecting the two classes. This line can be qualified with the type of relationship, and can also feature multiplicity rule (e.g., one-to- one, one-to-many, many- to-many) for the relationship.





### Description:

The above diagram represents the use case diagram of the project, normalization of duplicate records from multiple sources. The above class diagram describes the relation between the classes such as admin, publisher, login, register and users.

## SEQUENCE DIAGRAM:

Sequence diagrams document the interactions between classes to achieve a result, such as a use case. These interactions between classes are known as messages since UML is made for object-oriented programming. The Sequence diagram models the evolution of these communications through time by listing objects horizontally

and vertically.

* + - * User: The user or healthcare professional initiates the hypertension prediction process by requesting a prediction.
      * Hypertension Predictor: The predictor receives the prediction request from the user and starts the prediction process.
      * Preprocess Data: The hypertension predictor preprocesses the received data, performing necessary data cleaning, normalization, and feature engineering.
      * Train Prediction Model: The predictor trains a prediction model using the preprocessed data, applying machine learning techniques such as logistic regression, decision trees, or neural networks.
      * Validate Model: The predictor validates the trained model to assess its performance and accuracy, using appropriate validation techniques.
      * Send Validation Result: The predictor sends the validation result back to the user/hcp, indicating the accuracy and reliability of the prediction model.

### Graphical Notation:

In a Sequence diagram, classes and actors are listed as columns, with vertical lifelines indicating the lifetime of the object over time.

Object Objects are instances of classes and are arranged

horizontally. The pictorial representation for an Object 

is a class (a rectangle) with the name prefixed by the object name (optional) and a semi-colon.

Lifeline The Lifeline identifies the existence of the object over time. The notation 2for a Lifeline is a vertical dotted line extending from an object.

Activation Activations, modeled as rectangular boxes on the lifeline, indicate when the object is performing an action.

Message Messages, modeled as horizontal arrows Message between Activations, indicate the communications between objects.

1: Upload dataset()

ML Engine

User

2: Preprocessing()

3: Feature Extraction()

4: Split data()

5: Build Model()

6: calculate Accuracy()

7: Predict result()

8: Return result()

Sequence diagram for System

### Description:

The above diagram represents the sequence diagram of the project, normalization of duplicate records from multiple sources. In the above sequence diagram represents publisher, admin server and users are acts as object.

### Collaboration Diagram:

Like the other Beh.avioral diagrams, Collaboration diagrams model the interactions between classes. This type of diagram is a cross between an class diagram and a sequence diagram. Unlike the Sequence diagram, which models the interaction in a column and row type format, the Collaboration diagram uses the free-form arrangement of class as found in an Class diagram. This makes it easier to see all interactions involving a particular class.

* + - * User: The user or healthcare professional interacts with the hypertension prediction system, providing input and receiving output.
      * Hypertension Predictor: This component represents the main functionality responsible for coordinating the prediction process.
      * Data Source: The data source component provides the necessary patient data for hypertension prediction, such as electronic health records (EHRs) or data from wearable devices.
      * Preprocessing Component: This component preprocesses the raw data, handling missing values, outliers, normalization, and feature engineering.
      * Prediction Model: The prediction model component employs machine learning algorithms (e.g., logistic regression, decision trees) to predict hypertension based on

the pre-processed data.

* + - * Prediction Result: The prediction result component represents the output of the system, such as the predicted risk score or probability of developing hypertension.

Object Objects are instances of classes, and are arranged horizontally. The pictorial representation for an object is a class (a rectangle) with the name prefixed.

 Actor Actors can also communicate with Objects, so they too can be listed on Collaboration Diagrams. An actor is depicted by a stick figure.

Message Messages, modeled as horizontal arrows between Activations, indicate the communications

between objects.

1: Upload dataset()

|  |  |  |
| --- | --- | --- |
| User |  | ML Engine |
|  |  |

8: Return result()

2: Preprocessing()

3: Feature Extraction()

4: Split data()

5: Build Model()

6: calculate Accuracy()

7: Predict result()

Collaboration Diagram for system

##### Description:

The above diagram represents the collaboration diagram of the project, normalization of duplicate records from multiple sources. In the above sequence diagram represents publisher, admin server and users are acts as objects.

### Activity Diagram:

This shows the flow of events within the dataset. The activities that occur within a use case or within a class behaviour typically occur in a sequence. An activity diagram is designed to be simplified by looking at what happens during an operation or a process.

Each activity is represented by a rounded rectangle. The processing within an activity goes to compilation and then an automatic transmission to the next activity occurs. An arrow represents the transition from one activity to the next. An activity diagram describes a system in terms of activities. Activities are the state that represents the execution of a set of operations. These are like flow chart diagram and dataflow.

**Initial state:** which state is starting the process?

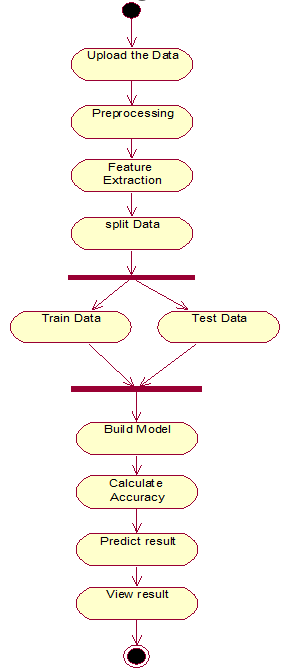
**Action State:** An action state represents the execution of an atomic action, typically the invocation of an operation. An action state is a simple state with an entry action whose only exit transition is triggered by the implicit event of completing the execution of the entry action.

Activity1

**Transition:** A transition is a directed relationship between a source state vertex and a target state vertex. It may be part of a compound transition, which takes the static machine from one static configuration to another, representing the complete response of the static machine to a particular event instance.

**Final state:** A final state represents the last or "final" state of the enclosing composite state. There may be more than one final state at any level signifying that the composite state can end in different ways or conditions. When a final state is reached and there are no other enclosing states it means that the entire state machine has completed its transitions and no more transitions can occur.

**Decision:** A state diagram (and by derivation an activity diagram) expresses decision when guard conditions are used to indicate different possible transitions that depend on Boolean conditions of the owning object.



### Description:

Activity Diagram

The above diagram represents the use case diagram of the project, normalization of duplicate records from multiple sources. The above activity diagram describes the activity of all the actors of the project.

# TECHNOLOGY DESCRIPTION

## INTRODUCTION TO PYTHON:

Guido van Rossum is the author of Python, a high-level object-oriented programming language. It is also known as a general-purpose programming language because it is employed in practically every industry we can imagine, as shown in the examples below:

* Web Development
* Software Development
* Game Development
* AI & ML
* Data Analytics

Every Programming language serves some purpose or use-case according to a domain. for e.g., JavaScript is the most popular language amongst web developers as it gives the developer the power to handle applications via different frameworks like react, Vue, angular which are used to build beautiful User Interfaces. Similarly, they have pros and cons at the same time. so if we consider python it is general-purpose which means it is widely used in every domain the reason is it’s very simple to understand, scalable because of which the speed of development is so fast. Now you get the idea why besides learning python it doesn’t require any programming background so that’s why it’s popular amongst developers as well.

Python has simpler syntax similar to the English language and also the syntax allows developers to write programs with fewer lines of code. Since it is open source there are many libraries available that make developers’ jobs easy ultimately resulting in high productivity. They can easily focus on business logic and its demanding skills in the digital era where information is available in large data sets.

Python is commonly used for developing websites and software, task automation, data analysis, and data visualization. Since it’s relatively easy to learn, Python has been adopted by many non-programmers such as accountants and scientists, for a variety of everyday tasks, like organizing finances.

Writing programs is a very creative and rewarding activity,” says University of Michigan and Coursera instructor Charles R Severance in his book Python for

Everybody. “You can write programs for many reasons, ranging from making your living to solving a difficult data analysis problem to having fun helping someone else solve a problem.

Python is a popular and in-demand skill to learn. But what is python programming used for? We’ve already briefly touched on some of the areas it can be applied to, and we’ve expanded on these and more Python examples below.

* + AI
* Machine Learning
* Data Analytics
* Data Visualization

Python is often used to develop the back end of a website or application—the parts that a user doesn’t see. Python’s role in web development can include sending data to and from servers, processing data and communicating with databases, URL routing, and ensuring security. Python offers several frameworks for web development. Commonly used ones include Django and Flask. Some web development jobs that use Python include back-end engineers, full stack engineers, Python developers, software engineers, and DevOps engineers. In software development, Python can aid in tasks like build control, bug tracking, and testing. With Python, software developers can automate testing for new products or features. Some Python tools used for software testing include Green and Requestion. Python isn't only for programmers and data scientists. Learning Python can open new possibilities for those in less data-heavy professions, like journalists, small business owners, or social media marketers. Python can also enable non- programmers to simplify certain tasks in their lives. Here are just a few of the tasks you could automate with Python:

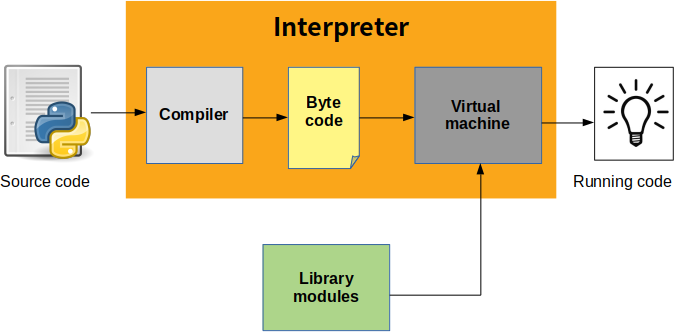
* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte code.
* Building large applications.
* It provides very high-level dynamic date types and supports dynamic checking.
* It supports automatic garbage collection.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

### History of Python:

Python was developed by Guido van Rossum 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, Unix shell, and other scripting languages.

At the time when he began implementing Python, Guido van Rossum was also reading the published scripts from "Monty Python's Flying Circus" (a BBC comedy series from the seventies, in the unlikely case you didn't know). It occurred to him that he needed a name that was short, unique, and slightly mysterious, so he decided to call the language Python.



### How To Install Python:

* Python is a widely used high-level programming language. To write and execute code in python, we first need to install Python on our system.
* Installing Python on Windows takes a series of easy steps.

### Step 1 − Select Version of Python to Install

Python has various versions available with differences between the syntax and working of different versions of the language. We need to choose the version which we want to use or need. There are different versions of Python 2 and Python 3 available.

### Step 2 − Download Python Executable Installer

On the web browser, on the official site of python (www.python.org), move to the Download for Windows section.

All the available versions of Python will be listed. Select the version required by you and click on Download. Let’s suppose, we chose the Python 3.11.1 version.

On clicking download, various available executable installers shall be visible with different operating system specifications. Choose the installer which suits your system operating system and download the installer. Let’s suppose we select the Windows installer (64 bits). The download size is less than 30MB.

### Step 3 − Run Executable Installer

We downloaded the Python 3.11.1 Windows 64-bit installer.

Run the installer. Make sure to select both the checkboxes at the bottom and then click Install New.



On Clicking the Install Now, the installation process starts.

The installation process will take a few minutes to complete and once the installation is successful, the following screen is displayed.

### Step 4 − Verify Python is installed on Windows:

To ensure Python is successfully installed on your system. Follow the given steps −

* Open the command prompt.
* Type ‘python’ and press enter.
* The version of the python which you have installed will be displayed if the python is successfully installed on your windows.

### Step 5 − Verify Pip was installed:

Pip is a powerful package management system for Python software packages. Thus, make sure that you have it installed.

To verify if pip was installed, follow the given steps −

* Open the command prompt.
* Enter pip –V to check if pip was installed.
* The following output appears if pip is installed successfully.

### Python Features:

**Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax.

**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-maintain.

### Need of Python Programming:

**Support libraries:**

Python comes with a large collection of prebuilt and portable functionality, known as the standard library. This library supports an array of application-level programming tasks, from text pattern matching to network scripting. It can be extended with both home - grown libraries and a vast collection of third- party application support software.

### It's Object-Oriented:

Python is an object-oriented language, from the ground up. Its class model supports advanced notions such as polymorphism, operator overloading, and multiple inheritance, yet in the context of Python's dynamic typing, object-oriented programming. Python's OOP nature makes it ideal as a scripting tool for object- oriented systems languages such as C++ and Java.

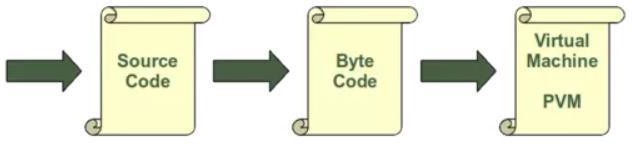
### It's Free:

Python is freeware something which has lately been come to be called opensource software. As with Tel and Perl, you can get the entire system for free over the Internet. On the contrary, the Python online community responds to user queries with a speed that most commercial software vendors would do well to notice.

### It's Portable:

Python is written in portable ANSI C, and compiles and runs on virtually every major platform in use today. For example, it runs on UNIX systems, Linux.

Python code is translated into intermediate code, which has to be executed by a virtual machine, known as the PVM, the Python Virtual Machine. This is a similar approach to the one taken by Java. There is even a way of translating Python programs into Java byte code for the Java Virtual Machine (JVM). This can be achieved with Python.



The compilation is hidden from the user for a good reason. Some newbies wonder sometimes where these ominous files with the .pyc suffix might come from. If Python has write-access for the directory where the Python program resides, it will store the compiled byte code in a file that ends with a .pyc suffix. If Python has no write access, the program will work anyway. The byte code will be produced but discarded when the program exists. Whenever a Python program is called, Python will check if a compiled version with the .pyc suffix exists. This file has to be newer than the file with the .py suffix. If such a file exists, Python will load the byte code, which will speed up the startup time of the script. If there is no byte code version, Python will create the byte code before it starts the execution of the program. Execution of a Python program means execution of the byte code on the Python.

Below are some facts about Python Programming Language:

* + Python is currently the most widely used multi-purpose, high-level programming language.
  + Python allows programming in Object-Oriented and Procedural paradigms.
  + Python programs generally are smaller than other programming languages like Java. Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.
  + Python language is being used by almost all tech-giant companies like Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.
  + The biggest strength of Python is huge collection of standard libraries which can be used for the following:
    - [Machine Learning](https://www.geeksforgeeks.org/machine-learning/)
    - GUI Applications (like [Kivy,](https://www.geeksforgeeks.org/kivy-tutorial/) Tkinter, PyQt etc.)
    - Web frameworks like [Django](https://www.geeksforgeeks.org/django-tutorial/) (used by YouTube, Instagram, Dropbox)
    - Image processing (like [OpenCV,](https://www.geeksforgeeks.org/opencv-python-tutorial/) Pillow)
    - Web scraping (like Scrapy, Beautiful Soup, Selenium)
    - Test frameworks
    - Multimedia
    - Scientific computing
    - Text processing and many more

## INTRODUCTION TO MACHINE LEARNING:

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy. IBM has a rich history with machine learning. One of its own, Arthur Samuel, is credited for coining the term, “machine learning” with his research (PDF, 481 KB) (link resides outside IBM) around the game of checkers. Robert Neeley, the self-proclaimed checkers master, played the game on an IBM 7094 computer in 1962, and he lost to the computer. Machine learning is an important component of the growing field of data science. Through the use of statistical methods, algorithms are trained to make classifications or predictions, uncovering key insights within data mining projects. These insights subsequently drive decision-making within applications and businesses, ideally impacting key growth metrics. As big data continues to expand and grow, the market demand for data scientists will increase, requiring them to assist in the identification of the most relevant business questions and subsequently the data to answer them. The learning system of a machine learning algorithm is into three main parts.

### A Decision Process:

In general, machine learning algorithms are used to make predictions or classification. Based on some input data, which can be labeled or unlabeled, your algorithm will produce an estimate of a pattern in the data.

### An Error Function:

An error function serves to evaluate the prediction of the model. If there are known examples, an error function can make a comparison to assess the accuracy of the model.

### A Model Optimization Process:

If the model can fit better to the data points in the training set, then weights are adjusted to reduce the discrepancy between the known example and the model estimate. The algorithm will repeat this evaluation and optimize the process, updating weights autonomously until a threshold of accuracy has been met. Machine learning classifiers fall into three primary categories.

### Supervised machine learning:

Supervised learning, also known as supervised machine learning, is defined by its use of labeled datasets to train algorithms to classify data or predict outcomes accurately. As input data is fed into the model, it adjusts its weights until the model has been fitted appropriately. This occurs as part of the cross- validation process to ensure that the model avoids overfitting or underfitting. Supervised learning helps organizations solve a variety of real-world problems at scale, such as classifying spam in a separate folder from your inbox. Some methods used in supervised learning include neural networks, naïve bayes, linear regression, logistic regression, random forest, support vector machine (SVM), and more.

### Unsupervised machine learning:

Unsupervised learning, also known as unsupervised machine learning, uses machine learning algorithms to analyze and cluster unlabeled datasets. These algorithms discover hidden patterns or data groupings without the need for human intervention. Its ability to discover similarities and differences in information makes it the ideal solution for exploratory data analysis, cross-selling strategies, customer segmentation, image and pattern recognition.

### Semi-supervised learning:

Semi-supervised learning offers a happy medium between supervised and unsupervised learning. During training, it uses a smaller labeled data set to guide classification and feature extraction from a larger, unlabeled data set. Semi-supervised learning can solve the problem of not having enough labeled data to train a supervised

learning algorithm.

### Reinforcement Learning:

Reinforcement learning involves an agent that learns to interact with an environment by taking actions and receiving feedback or rewards. The agent learns to maximize its cumulative reward through trial and error.

### Advantages of Machine Learning:

##### Automation and Efficiency:

Machine learning allows for automation of complex tasks that would otherwise require manual effort. It can process vast amounts of data quickly and efficiently, enabling faster decision-making and increased productivity.

##### Accurate Predictions and Insights:

Machine learning models can analyze large datasets and identify patterns, trends, and correlations that humans may not be able to detect. This capability enables accurate predictions, valuable insights, and data-driven decision-making.

##### Adaptability and Generalization:

Machine learning algorithms have the ability to learn from new data and adapt their models accordingly. They can generalize patterns and make predictions on unseen data, which is particularly useful in handling real-world scenarios and evolving environments.

##### Handling Complex and Large-Scale Data:

Machine learning techniques are well-suited for handling complex and high- dimensional data. They can efficiently process and extract useful information from massive datasets, including text, images, videos, and sensor data.

##### Improved Accuracy:

Machine learning algorithms can learn from labelled data or historical patterns to make predictions or classifications with a high degree of accuracy. They can detect subtle patterns and relationships in the data that may not be apparent to humans, leading to improved decision-making and outcomes.

##### Adaptability and Generalization:

Machine learning models have the ability to adapt and generalize to new, unseen data. Once trained, they can make predictions or decisions on new instances based on the patterns and knowledge gained from the training data. This adaptability

makes machine learning useful in scenarios where the problem or data distribution may change over time.

### Disadvantages of Machine Learning:

##### Data Dependency:

Machine learning models heavily rely on high-quality, relevant, and representative data for training. If the training data is biased, incomplete, or of poor quality, it can lead to biased or inaccurate predictions. Data preprocessing and data quality assurance are crucial steps in machine learning to mitigate these issues.

##### Need for Expertise and Resources:

Implementing machine learning requires expertise in data science, statistics, and programming. Developing, training, and fine-tuning machine learning models can be resource-intensive in terms of computational power, storage, and time. Additionally, organizations need skilled personnel to manage and interpret the results of machine learning algorithms.

##### Interpretability and Explainability:

Some machine learning models, especially deep learning models, are considered "black boxes" because their decision-making process is not easily interpretable by humans. This lack of interpretability can be a concern, especially in critical domains where the reasoning behind decisions needs to be explained.

##### Overfitting and Generalization Issues:

Machine learning models can sometimes overfit the training data, meaning they become overly specialized to the training examples and fail to generalize well to new, unseen data. Balancing model complexity, regularization techniques, and having sufficient diverse and representative data are crucial to address this issue.

##### Computational Requirements:

Training complex machine learning models can be computationally intensive and require significant computational resources, including powerful hardware and specialized infrastructure. Implementing and deploying such models can be expensive and may require expertise in distributed computing and optimization techniques.

It's important to weigh these advantages and disadvantages when considering the application of machine learning in specific contexts and to implement appropriate safeguards to mitigate potential risks.

# SAMPLE CODE

import pandas as pd **(#importing library in dataset)**

import csv as csv **(# imports dataset)**

import numpy as np **(#imports numerical values)**

df = pd.read\_csv('smoking.csv') **(#Reads the dataset)**

df df.head()

### Adding proper column names

column\_names = ['id', 'pseudo\_psu', 'pseudo\_stratum', 'stat\_weight', 'age', 'sex', 'race', 'body\_weight', 'height', 'avg\_systolic\_bp', 'avg\_diastolic\_bp', 'smoked\_alot', 'currently\_smokes', 'smoking', 'serum\_cholesterol', 'hbp'] df.columns = column\_names

df.head()

df.drop(df.index[-1], inplace=True)

df = df.apply(pd.to\_numeric, args=('coerce',)) df.shape

df.isnull().sum()

for i in df: print(i,np.round(df[i].isnull().mean(),4),'%')

for i in df: print(i,np.round(df[i].isnull().mean()\*100,4)>20)

df.info()

for column in df: print(column + ': ')

print(np.unique(df[column].values))

for column in df: print(column + ': ')

print(df[column].isnull().sum())

**Finding missing values percentage** missing\_value=pd.DataFrame(df.isnull().sum()) missing\_value=(missing\_value/len(df)\*100) missing\_value.reset\_index() missing\_value=missing\_value.rename(columns={'index': 'Variables', 0: 'Missing\_percentage'})

missing\_value=missing\_value.sort\_values('Missing\_percentage', ascending=False)

missing\_value.to\_csv("Missing\_perc.csv", index=False) missing\_value

df.drop(['currently\_smokes'],axis=1) **(#dropping the highest missing column)**

import matplotlib.pyplot as plt import seaborn as sns f,ax=plt.subplots(figsize=(15,12)) corr=df.corr()

sns.heatmap(corr,mask=np.zeros\_like(corr,dtype=np.bool),annot=True,cmap=sns.di verging\_palette(220,10,as\_cmap=True),square=True,ax=ax)

missing = ((~df['race'].isnull())&(~df['height'].isnull())&(~df['avg\_systolic\_bp'].isnull())&(~df ['avg\_diastolic\_bp'].isnull())

&(~df['smoking'].isnull())&(~df['serum\_cholesterol'].isnull())&(~df['age'].isnull())& (~df['sex'].isnull()))

data = df[missing][~df['body\_weight'].isnull()] data.head()

total\_data = data[['race', 'age', 'sex', 'height', 'avg\_systolic\_bp', 'avg\_diastolic\_bp', 'smoking', 'serum\_cholesterol',

'body\_weight']].values

from sklearn.model\_selection import train\_test\_split

x\_train, x\_test, y\_train, y\_test = train\_test\_split(total\_data[:,:-1], total\_data[:,-1], test\_size=.33, random\_state=0)

from sklearn import linear\_model

regr = linear\_model.LinearRegression() regr.fit(x\_train, y\_train) print(regr.score(x\_test, y\_test))

replace = df[missing][df['body\_weight'].isnull()][['race', 'age', 'sex', 'height', 'avg\_systolic\_bp', 'avg\_diastolic\_bp', 'smoking', 'serum\_cholesterol']].values new\_values = regr.predict(replace)

new\_values

for idx, val in enumerate(df[missing][df['body\_weight'].isnull()].index): df.loc[val, 'body\_weight'] = new\_values[idx]

missing = ((~df['race'].isnull())&(~df['age'].isnull())&(~df['sex'].isnull())) data = df[missing][~df['body\_weight'].isnull()]

total\_data = data[['race', 'age', 'sex', 'hbp', 'body\_weight']].values

x\_train, x\_test, y\_train, y\_test = train\_test\_split(total\_data[:,:-1], total\_data[:,-1], test\_size=.33, random\_state=0)

regr = linear\_model.LinearRegression() regr.fit(x\_train, y\_train)

replace = df[missing][df['body\_weight'].isnull()][['race', 'age', 'sex', 'hbp']].values new\_values = regr.predict(replace)

for idx, val in enumerate(df[missing][df['body\_weight'].isnull()].index): df.loc[val, 'body\_weight'] = new\_values[idx]

missing = ((~df['race'].isnull())&(~df['age'].isnull())&(~df['sex'].isnull())&(~df['body\_weight']. isnull()))

data = df[missing][~df['height'].isnull()]

total\_data = data[['race', 'age', 'sex', 'body\_weight', 'height']].values

x\_train, x\_test, y\_train, y\_test = train\_test\_split(total\_data[:,:-1], total\_data[:,-1], test\_size=.33, random\_state=0)

regr = linear\_model.LinearRegression() regr.fit(x\_train, y\_train) print(regr.score(x\_test, y\_test))

replace = df[missing][df['height'].isnull()][['race', 'age', 'sex', 'body\_weight']].values new\_values = regr.predict(replace)

for idx, val in enumerate(df[missing][df['height'].isnull()].index): df.loc[val, 'height'] = new\_values[idx]

data = df[~df['race'].isnull()] **(#Predict Race)**

total\_data = data[['age', 'sex', 'body\_weight', 'height', 'smoking', 'race']].values x\_train, x\_test, y\_train, y\_test = train\_test\_split(total\_data[:,:-1], total\_data[:,-1], test\_size=.33, random\_state=0)

regr = linear\_model.LinearRegression() regr.fit(x\_train, y\_train)

replace = df[df['race'].isnull()][['age', 'sex', 'body\_weight', 'height', 'smoking']].values

new\_values = regr.predict(replace)

for idx, val in enumerate(df[df['race'].isnull()].index): df.loc[val, 'race'] = new\_values[idx]

data = df[~df['smoking'].isnull()] **(#Predicting Smoke)**

total\_data = data[['age', 'sex', 'body\_weight', 'height', 'race', 'smoked\_alot', 'smoking']].values

x\_train, x\_test, y\_train, y\_test = train\_test\_split(total\_data[:,:-1], total\_data[:,-1], test\_size=.33, random\_state=0)

from sklearn.naive\_bayes import GaussianNB gnb = GaussianNB()

gnb = gnb.fit(x\_train, y\_train) y\_pred = gnb.predict(x\_test) from sklearn import metrics

print(metrics.accuracy\_score(y\_test, y\_pred))

replace = df[df['smoking'].isnull()][['age', 'sex', 'body\_weight', 'height', 'race', 'smoked\_alot']].values

new\_values = gnb.predict(replace)

for idx, val in enumerate(df[df['smoking'].isnull()].index): df.loc[val, 'smoking'] = new\_values[idx]

data=df[~df['serum\_cholesterol'].isnull()] **(#Predict Cholesterol)** total\_data=data[['age', 'sex', 'body\_weight', 'height', 'smoking', 'race', 'serum\_cholesterol']].values

x\_train, x\_test, y\_train, y\_test = train\_test\_split(total\_data[:,:-1], total\_data[:,-1], test\_size=.33, random\_state=0)

regr = linear\_model.LinearRegression() regr.fit(x\_train, y\_train)

replace= df[df['serum\_cholesterol'].isnull()][['age', 'sex', 'body\_weight', 'height', 'smoking', 'race']].values

new\_values = regr.predict(replace)

for idx, val in enumerate(df[df['serum\_cholesterol'].isnull()].index): df.loc[val, 'serum\_cholesterol'] = new\_values[idx]

data = df[~df['avg\_systolic\_bp'].isnull()] **(#Predict Systolic bp)** total\_data = data[['age', 'sex', 'body\_weight', 'height', 'smoking', 'race', 'serum\_cholesterol', 'avg\_systolic\_bp']].values

x\_train, x\_test, y\_train, y\_test = train\_test\_split(total\_data[:,:-1], total\_data[:,-1], test\_size=.33, random\_state=0)

regr = linear\_model.LinearRegression() regr.fit(x\_train, y\_train)

replace = df[df['avg\_systolic\_bp'].isnull()][['age', 'sex', 'body\_weight', 'height', 'smoking', 'race', 'serum\_cholesterol']].values

new\_values = regr.predict(replace)

for idx, val in enumerate(df[df['avg\_systolic\_bp'].isnull()].index): df.loc[val, 'avg\_systolic\_bp'] = new\_values[idx]

import numpy as np **(#Predict Diastolic bp)**

np.isnan(data.any()) np.isfinite(data.all())

data = df[~df['avg\_diastolic\_bp'].isnull()]

total\_data = data[['age', 'sex', 'body\_weight', 'height', 'smoking', 'race', 'serum\_cholesterol', 'avg\_systolic\_bp',

'avg\_diastolic\_bp']].values

x\_train, x\_test, y\_train, y\_test = train\_test\_split(total\_data[:,:-1], total\_data[:,-1], test\_size=.33, random\_state=0)

regr = linear\_model.LinearRegression() regr.fit(x\_train, y\_train)

replace = df[df['avg\_diastolic\_bp'].isnull()][['age', 'sex', 'body\_weight', 'height', 'smoking', 'race', 'serum\_cholesterol', 'avg\_systolic\_bp']].values

new\_values = regr.predict(replace)

for idx, val in enumerate(df[df['avg\_diastolic\_bp'].isnull()].index):

df.loc[val, 'avg\_diastolic\_bp'] = new\_values[idx]

df['true\_hbp'] = df['hbp']

df.loc[(df['hbp'] == 0)&(df['avg\_diastolic\_bp'] >= 90), 'true\_hbp'] = 1 df.head()

df['BMI'] = (df['body\_weight'] / (df['height'] \*\* 2)) \* 703 df.head()

### (#Predictions on data)

total\_data = df[['age', 'sex', 'body\_weight', 'height', 'race', 'smoking', 'serum\_cholesterol', 'hbp']].values

x\_train, x\_test, y\_train, y\_test = train\_test\_split(total\_data[:, :-1], total\_data[:, -1], test\_size=.33, random\_state=0)

### (#importing the graphs)

%matplotlib inline

import matplotlib.pyplot as plt

plt.scatter(total\_data[:100, 0], total\_data[:100, -1]) plt.title('Age vs HBP')

plt.ylabel('HBP') plt.xlabel('Age') plt.show()

x\_train[:, 0]

plt.scatter(total\_data[:100, 2], total\_data[:100, -1]) plt.title('Body Weight vs HBP')

plt.ylabel('HBP') plt.xlabel('Body Weight') plt.show()

plt.scatter(total\_data[:200, 6], total\_data[:200, -1]) plt.title('Cholesterol vs HBP')

plt.ylabel('HBP') plt.xlabel('Cholesterol') plt.show()

plt.scatter(total\_data[:100, 2], total\_data[:100, -1]) plt.title('Height vs HBP')

plt.ylabel('HBP') plt.xlabel('Height vs HBP') plt.show() df[df['avg\_diastolic\_bp'] > 90]

total\_data[:, 1]

### (#Gaussian Naïve Bayesian Algorithm)

gnb=GaussianNB()

y\_pred=gnb.fit(x\_train, y\_train).predict(x\_test)

print(metrics.accuracy\_score(y\_test, y\_pred)) from sklearn.metrics import classification\_report target\_names = ['NO HBP', 'HBP']

classification\_report(y\_test, y\_pred, target\_names=target\_names)

### (#Logistic Regression Algorithm)

from sklearn.linear\_model import LogisticRegression

y\_pred = LogisticRegression().fit(x\_train, y\_train).predict(x\_test)

print(metrics.accuracy\_score(y\_test, y\_pred)) classification\_report(y\_test, y\_pred, target\_names=target\_names)

### (#Support Vector Machine Algorithm)

from sklearn.svm import SVC

clf=SVC(C=1.0, cache\_size=200, class\_weight=None,coef0=0.0, decision\_function\_shape='ovr', degree=3, gamma='auto', kernel='rbf', max\_iter=-1, probability=False, random\_state=None, shrinking=True, tol=0.001, verbose=False)

clf.fit(x\_train, y\_train) y\_pred=clf.predict(x\_test)

print(metrics.accuracy\_score(y\_test, y\_pred)) classification\_report(y\_test, y\_pred, target\_names=target\_names)

df.iloc[0:1,:](**#PREDICTS THE RESULT)**

y\_pred=clf.predict([[30,0,200,1,180,3,1]]) y\_pred[0]

df[['age', 'sex', 'body\_weight', 'height', 'race', 'smoking', 'serum\_cholesterol', 'hbp']].values

## INTRODUCTION:

# TESTING

In general, software engineers distinguish software faults from software failures. In case of a failure, the software does not do what the user expects. A fault is a programming error that may or may not actually manifest as a failure. A fault can also be described as an error in the correctness of the semantics of a computer program. A fault will become a failure if the exact computation conditions are met, one of them being that the faulty portion of computer software executes on the CPU. A fault can also turn into a failure when the software is ported to a different hardware platform or a different compiler, or when the software gets extended. Software testing is the technical investigation of the product under test to provide stakeholders with quality related information.

### System Testing and Implementation:

The purpose is to exercise the different parts of the module code to detect coding errors. After this the modules are gradually integrated into subsystems, which are then integrated themselves too eventually forming the entire system. During integration of module integration testing is performed. The goal of this is to detect design errors, while focusing the interconnection between modules. After the system was put together, system testing is performed. Here the system is tested against the system requirements to see if all requirements were met, and the system performs as specified by the requirements. Finally accepting testing is performed to demonstrate to the client the operation of the system. For the testing to be successful, proper selection of the test case is essential. There are two different approaches for selecting test cases. The software or the module to be tested is treated as a black box, and the test cases are decided based on the specifications of the system or module. For this reason, this form of testing is also called “black box testing”. The focus here is on testing the external behavior of the system. In structural testing the test cases are decided based on the logic of the module to be tested. A common approach here is to achieve some type of coverage of the statements in the code. The two forms of testing are

complementary: one tests the external behavior, the other tests the internal structure.

Testing is an extremely critical and time-consuming activity. It requires proper planning of the overall testing process. Frequently the testing process starts with the test plan. This plan identifies all testing related activities that must be performed and specifies the schedule, allocates the resources, and specifies guidelines for testing. The test plan specifies conditions that should be tested; different units to be tested, and the manner in which the module will be integrated together. Then for different test units, a test case specification document is produced, which lists all the different test cases, together with the expected outputs, that will be used for testing.

During the testing of the unit the specified test cases are executed and the actual results are compared with the expected outputs. The final output of the testing phase is the testing report and the error report, or a set of such reports. Each test report contains a set of test cases and the result of executing the code with the test cases. The error report describes the errors encountered and the action taken to remove the error. **Testing Techniques:**

Testing is a process which reveals errors in the program. It is the major quality measure employed during software development. During testing, the program is executed with a set of conditions known as test cases and the output is evaluated to determine whether the program is performing as expected. In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development are:

### Black Box Testing:

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been uses to find errors in the following categories:

* Incorrect or missing functions.
* Interface errors.
* Errors in data structure or external database access.
* Performance errors
* Initialization and termination errors.

In this testing only the output is checked for correctness. The logical flow of the data is not checked.

### White Box Testing:

In this testing, the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been uses to generate the test cases in the following cases:

* Guarantee that all independent paths have been executed.
* Execute all logical decisions on their true and false sides.
* Execute all loops at their boundaries and within their operational.

### Testing strategy Unit:

**Testing:**

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 andexpected results.

This System consists of 3 modules. Those are Reputation module, route discovery module, audit module. Each module is taken as a unit and tested. Identified errors are corrected and executable units are obtained.

### Integration Testing:

Integration tests are designed to test integrated software components to determine if they 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.

### System Testing:

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.

### Functional Testing:

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

Functional testing is centered on the following items.

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

: identified classes of invalid input must be rejected. Functions

: identified functions must be exercised.

Output : identified classes of application outputs must be

exercised.

Procedures : interfacing systems or procedures must be invoked.

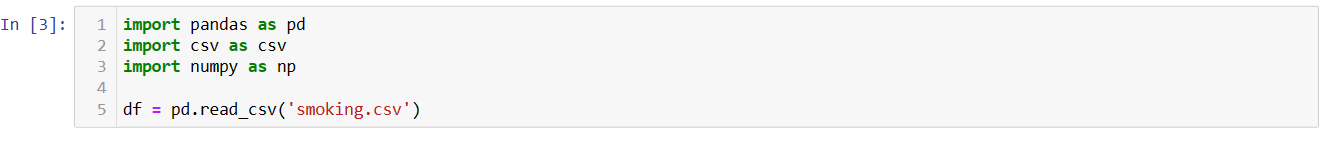
## SAMPLE TEST CASE SPECIFICATION:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case id** | **Test case name** | **Input** | **Expected Output** | **Observed**  **Output** | **Result** |
| T1 | Upload Data Set | Enter valid Path of dataset | Dataset should load successful. | Dataset loaded Successfully. | Pass |
| T2 | Upload Data Set | Enter invalid  Path of dataset | Dataset should upload successfully. | Dataset not loaded successfully. | Pass |
|  |
|  |

Table 7.2.1: Sample Test Cases Specifications

**Testing Screens:**

**Test case 1:**



**Description:**  The above screen tells that the dataset has uploaded successfully.

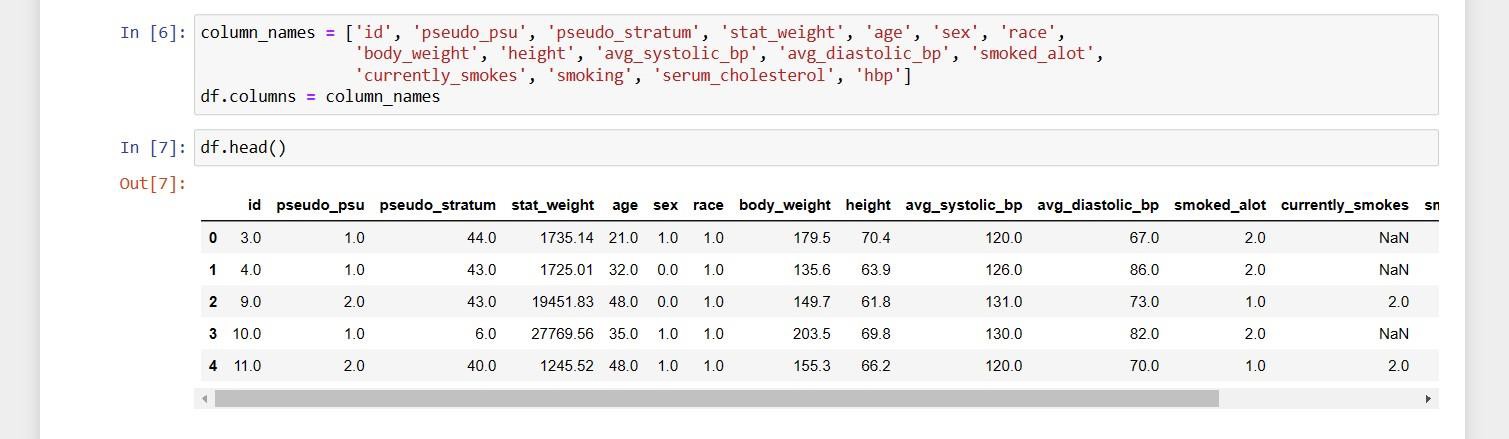
**Test case 2:**



**Description:** The above screen tells that the dataset not loaded successfully.

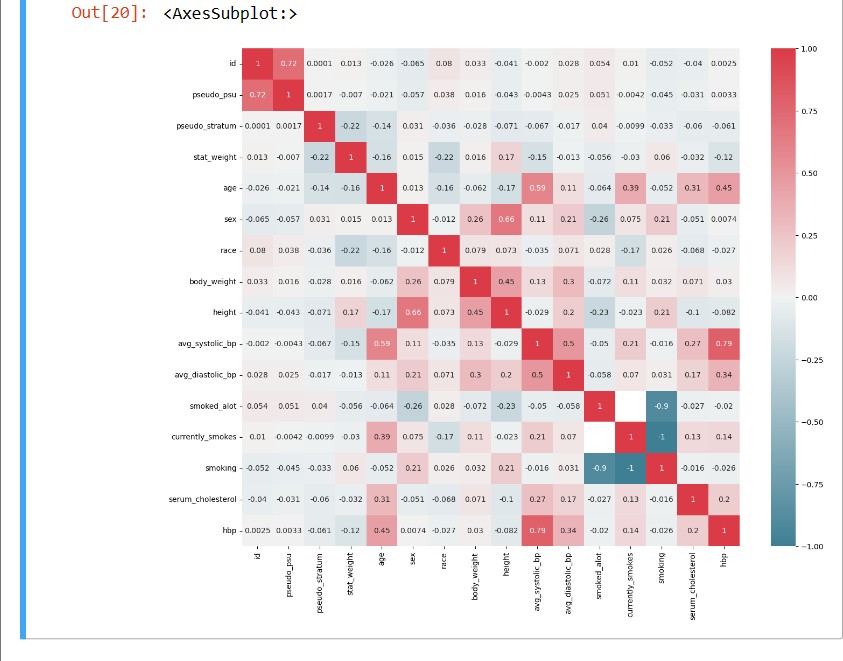
# SCREENSHOTS

### Screenshot for System Execution using dataset:



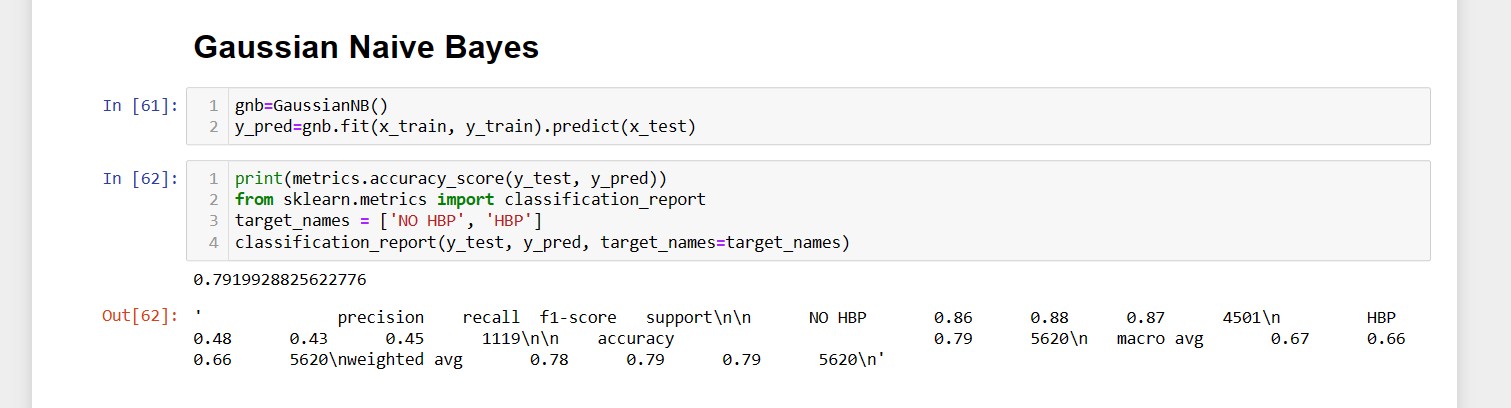
**Description:** We can represent the column names by giving the new and simple names to the respective columns for identifying.

### Screenshot based on the confusion matrix:



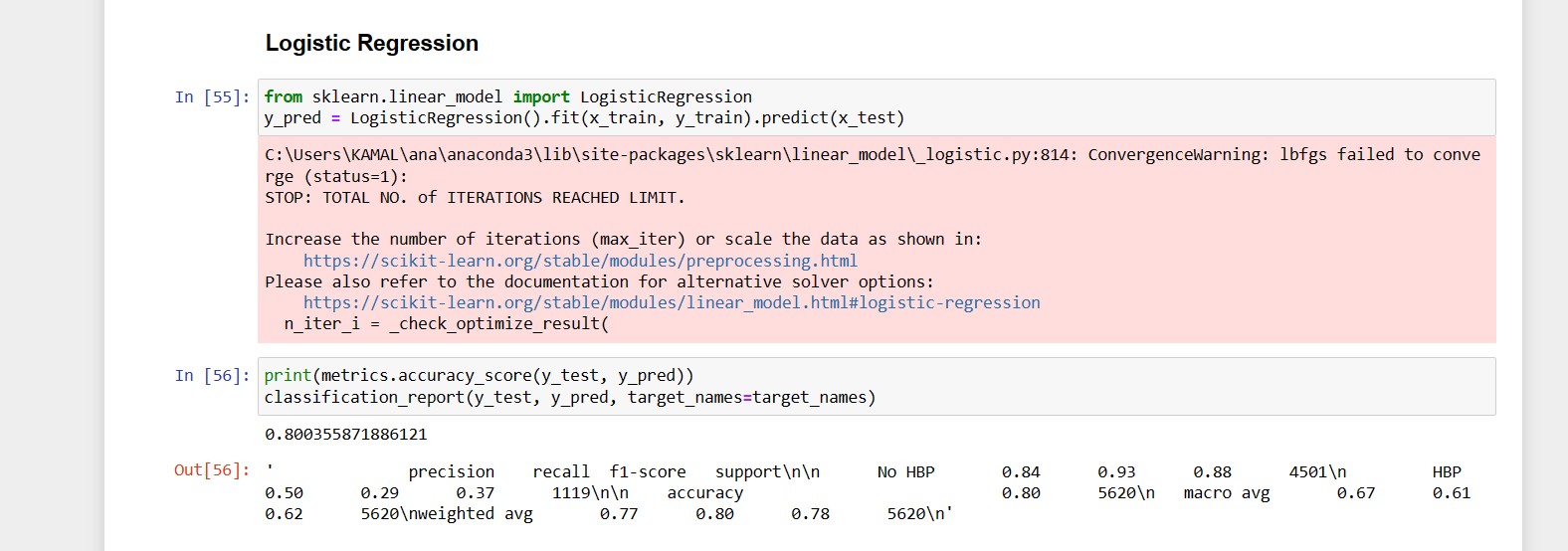
**Description:** In this, it shows the outcome values of both actual and predicted values of each column in it. As it ranges from -1 to 1. The confusion matrix is used to show the values and difference between the outcomes.

### Screenshot for Execution using Gaussian Naïve Bayes Algorithm:



**Description:** Here we are testing and predicting the values for accuracy which are dependent on target variables based on the final values. As it shows the final accuracy with 79.1%.

### Screenshot for Logistic Regression Algorithm:



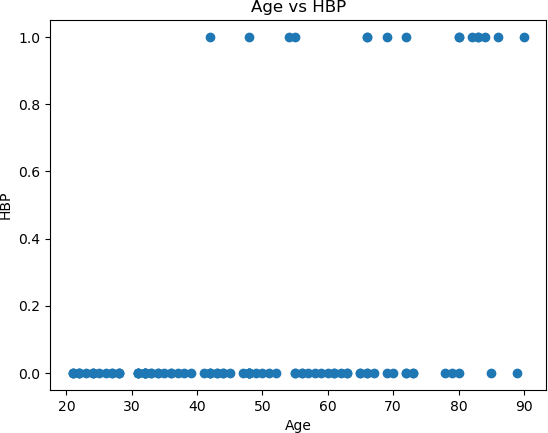
**Description:** Here we are testing and predicting the values for accuracy which are dependent on target variables based on the final values. As it shows the final accuracy with 80.0%.

### Screenshot for Support Vector Machine Algorithm:



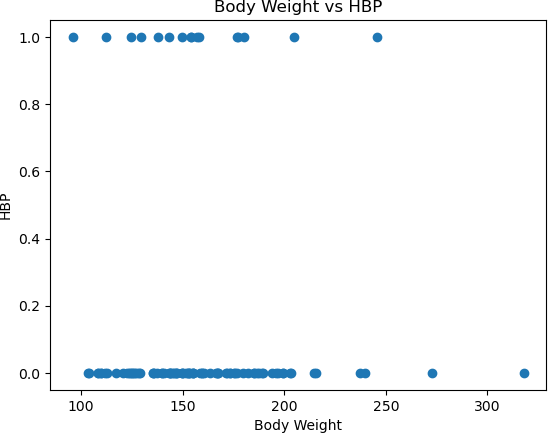
**Description:** Here we are testing and predicting the values for accuracy which are dependent on target variables based on the final values. As it shows the final accuracy with 80.10%. As from the above one we can see that it has more accuracy when compared to the other ones.

### Screenshot for Distribution of Age Vs HBP:



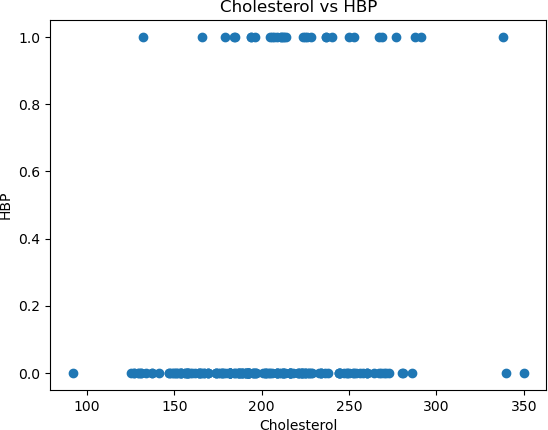
**Description:** It compares Age Vs HBP.

### Screenshot for Body Weight Vs HBP:



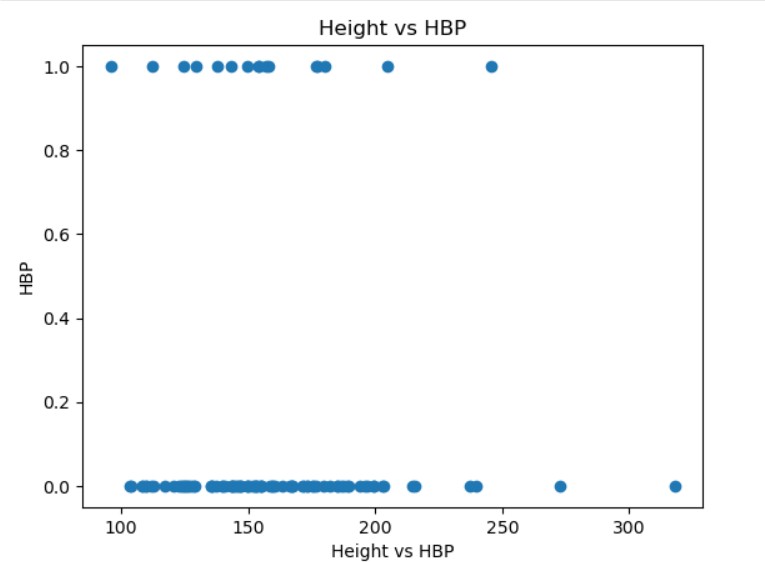
**Description:** It compares Body Weight Vs HBP.

### Screenshot for Cholesterol Vs HBP:



**Description:** It compares cholesterol Vs HBP.

### Screenshot for Distribution of Height Vs HBP:



**Description:** It compares height vs hbp.

# CONCLUSION

The conclusion of this project is that it prevents hypertension by checking their blood pressure on a website that has been created in the form of web application interface that is used for giving their respective details of their health details. After predicting, it results in whether they had hypertension or not. Predictive models for hypertension can also help in deciding on the level of interventions needed within the community and thus, assuring a positive impact. Future research should consider improving the predictive accuracy of models by using algorithms in larger general populations to avoid the healthy population effect. By using Naïve Bayesian Classifier (NBC), Logistic Regression (LR) and Support Vector Machine (SVM) algorithms we can find the accuracy and manage the respective hypertensive disease.

**REFFERENCES:**

# BIBLIOGRAPHY

1. Predicting the Risk of Hypertension Based on Several Easy-to-Collect Risk Factors: A Machine Learning Method by Huanhuan, Yang xu, [10.3389/fpubh.2021.619429](https://doi.org/10.3389/fpubh.2021.619429)
2. Predicting hypertension using machine learning: Findings from Qatar Biobank Study: Latifa A. AlKaabi, Lina S. Ahmed, [10.1371/journal.pone.0240370](https://doi.org/10.1371/journal.pone.0240370)
3. Development and validation of a hypertension risk prediction model and construction of a risk score in a Canadian population:[Mohammad Ziaul Islam Chowdhury](https://www.nature.com/articles/s41598-022-16904-x#auth-Mohammad_Ziaul_Islam-Chowdhury), [Alexander A. Leung](https://www.nature.com/articles/s41598-022-16904-x#auth-Alexander_A_-Leung),10.1038/s41598-022-16904-x

## WEB SITES:

1. <https://medium.com/mlpoint/numpy-for-machine-learning-211a3e58b574>
2. <https://machinelearningmastery.com/function-optimization-with-scipy/>
3. https://medium.com/mlpoint/pandas-for-machine-learning-53846bc9a98b

## TEXTBOOKS:

1. Introduction to Machine Learning with Python: A Guide for Data Scientists, Andreas, Muller & Sarah Guido, Orielly Publications,2019.
2. Machine Learning, Tom. Mitchell, Mc Graw-Hill publication,2017.
3. Programming and problem solving with python, Ashok Namdev Kamthane, Amit Ashok TMH,2019.