**ADVANCED MACHINE LEARNING MODELS FOR PREDICTIVE ANALYSIS OF PATIENT RISK LEVELS IN CLINICAL DECISION SUPPORT SYSTEMS**

**TABLE OF CONTENT**

|  |  |  |
| --- | --- | --- |
| **SL.NO** | **TITLE** | **PAGE.NO** |
| **01** | **ABSTRACT** |  |
| **02** | **EXISTING SYSTEM**  2.1 DRAWBACKS |  |
| **03** | **INTRODUCTION**  3.1 DATA SCIENCE  3.2 ARTIFICIAL INTELLIGENCE |  |
| **04** | **MACHINE LEARNING** |  |
| **05** | **PREPARING DATASET** |  |
| **06** | **PROPOSED SYSTEM**  6.1 ADVANTAGES |  |
| **07** | **LITERATURE SURVEY** |  |
| **08** | **SYSTEM STUDY**  8.1 OBJECTIVES  8.2 PROJECT GOAL  8.3 SCOPE OF THE PROJECT |  |
| **09** | **FEASIBILITY STUDY** |  |
| **10** | **LIST OF MODULES** |  |
| **11** | **PROJECT REQUIREMENTS**  11.1 FUNCTIONAL REQUIREMENTS  11.2 NON-FUNCTIONAL REQUIREMENTS |  |
| **12** | **ENVIRONMENT REQUIREMENT** |  |
| **13** | **SOFTWARE DESCRIPTION**  13.1 ANACONDA NAVIGATOR  13.2 JUPYTER NOTEBOOK |  |
| **14** | **PYTHON** |  |
| **15** | **SYSTEM ARCHITECTURE** |  |
| **16** | **WORKFLOW DIAGRAM** |  |
| **17** | **USECASE DIAGRAM** |  |
| **18** | **CLASS DIAGRAM** |  |
| **19** | **ACTIVITY DIAGRAM** |  |
| **20** | **SEQUENCE DIAGRAM** |  |
| **21** | **ER – DIAGRAM** |  |
| **22** | **MODULE DESCRIPTION**  22.1 MODULE DIAGRAM  22.2 MODULE GIVEN INPUT EXPECTED OUTPUT |  |
| **23** | **DEPLOYMENT** |  |
| **24** | **HTML** |  |
| **25** | **CSS** |  |
| **26** | **CODING** |  |
| **27** | **CONCLUSION** |  |
| **28** | **FUTURE WORK** |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **SL.NO** | **TITLE** | **PAGE.NO** |
| **01** | **SYSTEM ARCHITECTURE** |  |
| **02** | **WORKFLOW DIAGRAM** |  |
| **03** | **USECASE DIAGRAM** |  |
| **04** | **CLASS DIAGRAM** |  |
| **05** | **ACTIVITY DIAGRAM** |  |
| **06** | **SEQUENCE DIAGRAM** |  |
| **07** | **ER – DIAGRAM** |  |
| **08** | **MODULE DIAGRAM** |  |

**LIST OF SYMBOLS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **NOTATION**  **NAME** | **NOTATION** | **DESCRIPTION** |
| 1. | Class | *Class Name*  *-attribute*  *-attribute*  *+operation*  *+operation*  *+operation*  *+ public*  *-private*  *# protected* | Represents a collection of similar entities grouped together. |
| 2. | Association | nAME  Class A  Class B    Class A  Class B | Associations represents static relationships between classes. Roles representsthe way the two classes see each other. |
| 3. | Actor | Class A  Class A  Class B  Class B | It aggregates several classes into a single classes. |
| 4. | Aggregation | Interaction between the system and external environment |

|  |  |  |  |
| --- | --- | --- | --- |
| 5. | Relation  (uses) | uses | Used for additional process communication. |
| 6. | Relation  (extends) | EXTENDS | Extends relationship is used when one use case is similar to another use case but does a bit more. |
| 7. | Communication |  | Communication between various use cases. |
| 8. | State | State | State of the processs. |
| 9. | Initial State |  | Initial state of the object |
| 10. | Final state |  | Final state of the object |
| 11. | Control flow |  | Represents various control flow between the states. |
| 12. | Decision box |  | Represents decision making process from a constraint |
| 13. | Usecase |  | Interact ion between the system and external environment. |

|  |  |  |  |
| --- | --- | --- | --- |
| 14. | Component |  | Represents physical modules which is a collection of components. |
| 15. | Node |  | Represents physical modules which are a collection of components. |
| 16. | Data Process/State |  | A circle in DFD represents a state or process which has been triggered due to some event or acion. |
| 17. | External entity |  | Represents external entities such as keyboard,sensors,etc. |
| 18. | Transition |  | Represents communication that occurs between processes. |
| 19. | Object Lifeline |  | Represents the vertical dimensions that the object communications. |
| 20. | Message | Message | Represents the message exchanged. |

**ADVANCED MACHINE LEARNING MODELS FOR PREDICTIVE ANALYSIS OF PATIENT RISK LEVELS IN CLINICAL DECISION SUPPORT SYSTEMS**

**ABSTRACT:**

The integration of advanced machine learning models into clinical decision support systems (CDSS) offers a transformative approach to predicting patient risk levels with greater accuracy and efficiency. This project leverages supervised learning algorithms, to analyze patient health records and vital parameters for early risk stratification. Implemented using the Django framework, the system facilitates risk level predictions through a secure, web-based interface designed for healthcare professionals. By automating the detection of high-risk patients, the platform enhances clinical decision-making, supports timely interventions, and ultimately contributes to improved patient outcomes. The model is trained and validated on curated medical datasets, ensuring robustness and scalability for deployment in hospital management systems.

**Keywords:**  
Machine Learning, Patient Risk Prediction, Clinical Decision Support System, Django Framework, Healthcare Analytics, Risk Stratification, Supervised Learning, Medical Diagnosis.

**02. EXISTING SYSTEM:**

Physiological reports have confirmed that there are differences in speech signals between depressed and healthy individuals. Therefore, as an application in the field of affective computing, automatic depression level prediction through speech signals has received the attention of researchers, which often estimate the depression severity of individuals by the Fourier or Mel spectrograms of speech signals. However, some studies on speech emotion recognition suggest that directly modeling the raw speech signal is more helpful for extracting emotion-related information. Inspired by this fact, we develop a WavDepressionNet to model raw speech signals for the improvement of prediction accuracy. In our method, a representation block is proposed to find a set of basis vectors to construct the optimal transformation space and generate the transformation result (named Depression Feature Map, DFM) of speech signal for facilitating the perception of depression cues. We further propose an assessment block, which cannot only use the designed spatiotemporal self-calibration mechanism to calibrate the DFM and highlight the useful elements, but also aggregates the calibrated DFM across various temporal ranges with the dilated convolution. Experimental results on the AVEC 2013 and AVEC 2014 depression databases demonstrate the effectiveness of our approach over previous works.

**Disadvantages:**

* there are not utilizing deployment part
* there not utilizing machine learning technologies
* Focus on Raw Speech
* Computational Cost
* Data Dependence

**3. INTRODUCTION**

The integration of advanced machine learning models into Clinical Decision Support Systems (CDSS) is revolutionizing the healthcare landscape by enabling more accurate and timely predictions of patient risk levels. This project leverages sophisticated algorithms to analyze diverse patient data and identify potential health risks. By embedding these predictive models within the Django web framework, the system ensures seamless interaction between data processing, model inference, and user interfaces for clinicians. This not only enhances decision-making but also supports early intervention strategies, ultimately improving patient outcomes and optimizing healthcare delivery.

**3.1 Data Science**

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured data, and apply knowledge and actionable insights from data across a broad range of application domains.

The term "data science" has been traced back to 1974, when Peter Naur proposed it as an alternative name for computer science. In 1996, the International Federation of Classification Societies became the first conference to specifically feature data science as a topic. However, the definition was still in flux.

The term “data science” was first coined in 2008 by D.J. Patil, and Jeff Hammerbacher, the pioneer leads of data and analytics efforts at LinkedIn and Facebook. In less than a decade, it has become one of the hottest and most trending professions in the market.

Data science is the field of study that combines domain expertise, programming skills, and knowledge of mathematics and statistics to extract meaningful insights from data.

Data science can be defined as a blend of mathematics, business acumen, tools, algorithms and machine learning techniques, all of which help us in finding out the hidden insights or patterns from raw data which can be of major use in the formation of big business decisions.

**Data Scientist:**

Data scientists examine which questions need answering and where to find the related data. They have business acumen and analytical skills as well as the ability to mine, clean, and present data. Businesses use data scientists to source, manage, and analyze large amounts of unstructured data.

**Required Skills for a Data Scientist:**

* **Programming**: Python, SQL, Scala, Java, R, MATLAB.
* **Machine Learning**: Natural Language Processing, Classification, Clustering.
* **Data Visualization**: Tableau, SAS, D3.js, Python, Java, R libraries.
* **Big data platforms**: MongoDB, Oracle, Microsoft Azure, Cloudera.

**3.2 ARTIFICIAL INTELLIGENCE**

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving.

Artificial intelligence (AI) is [intelligence](https://en.wikipedia.org/wiki/Intelligence) demonstrated by [machines](https://en.wikipedia.org/wiki/Machine), as opposed to the natural intelligence [displayed by humans](https://en.wikipedia.org/wiki/Human_intelligence) or [animals](https://en.wikipedia.org/wiki/Animal_cognition). Leading AI textbooks define the field as the study of "[intelligent agents](https://en.wikipedia.org/wiki/Intelligent_agent)" any system that perceives its environment and takes actions that maximize its chance of achieving its goals. Some popular accounts use the term "artificial intelligence" to describe machines that mimic "cognitive" functions that humans associate with the [human mind](https://en.wikipedia.org/wiki/Human_mind), such as "learning" and "problem solving", however this definition is rejected by major AI researchers.

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, and speech recognition and machine vision.

AI applications include advanced web search engines, recommendation systems (used by Youtube, Amazon and Netflix), Understanding human speech (such as Siri or Alexa), self-driving cars (e.g. Tesla), and competing at the highest level in strategic game systems (such as chess and Go), As machines become increasingly capable, tasks considered to require "intelligence" are often removed from the definition of AI, a phenomenon known as the AI effect. For instance, optical character recognition is frequently excluded from things considered to be AI, having become a routine technology.

Artificial intelligence was founded as an academic discipline in 1956, and in the years since has experienced several waves of optimism, followed by disappointment and the loss of funding (known as an "AI winter"), followed by new approaches, success and renewed funding. AI research has tried and discarded many different approaches during its lifetime, including simulating the brain, modeling human problem solving, formal logic, large databases of knowledge and imitating animal behavior. In the first decades of the 21st century, highly mathematical statistical machine learning has dominated the field, and this technique has proved highly successful, helping to solve many challenging problems throughout industry and academia.

The various sub-fields of AI research are centered on particular goals and the use of particular tools. The traditional goals of AI research include reasoning, knowledge representation, planning, learning, natural language processing, perception and the ability to move and manipulate objects. General intelligence (the ability to solve an arbitrary problem) is among the field's long-term goals. To solve these problems, AI researchers use versions of search and mathematical optimization, formal logic, artificial neural networks, and methods based on statistics, probability and economics. AI also draws upon computer science, psychology, linguistics, philosophy, and many other fields.

The field was founded on the assumption that human intelligence "can be so precisely described that a machine can be made to simulate it". This raises philosophical arguments about the mind and the ethics of creating artificial beings endowed with human-like intelligence. These issues have been explored by myth, fiction and philosophy since antiquity. Science fiction and futurology have also suggested that, with its enormous potential and power, AI may become an existential risk to humanity.

As the hype around AI has accelerated, vendors have been scrambling to promote how their products and services use AI. Often what they refer to as AI is simply one component of AI, such as machine learning. AI requires a foundation of specialized hardware and software for writing and training machine learning algorithms. No one programming language is synonymous with AI, but a few, including Python, R and Java, are popular.

In general, AI systems work by ingesting large amounts of labeled training data, analyzing the data for correlations and patterns, and using these patterns to make predictions about future states. In this way, a chatbot that is fed examples of text chats can learn to produce life like exchanges with people, or an image recognition tool can learn to identify and describe objects in images by reviewing millions of examples.

AI programming focuses on three cognitive skills: learning, reasoning and self-correction.

**Learning processes.** This aspect of AI programming focuses on acquiring data and creating rules for how to turn the data into actionable information. The rules, which are called algorithms, provide computing devices with step-by-step instructions for how to complete a specific task.

**Reasoning processes.** This aspect of AI programming focuses on choosing the right algorithm to reach a desired outcome.

**Self-correction processes.** This aspect of AI programming is designed to continually fine-tune algorithms and ensure they provide the most accurate results possible.

AI is important because it can give enterprises insights into their operations that they may not have been aware of previously and because, in some cases, AI can perform tasks better than humans. Particularly when it comes to repetitive, detail-oriented tasks like analyzing large numbers of legal documents to ensure relevant fields are filled in properly, AI tools often complete jobs quickly and with relatively few errors.

Artificial neural networks and deep learning artificial intelligence technologies are quickly evolving, primarily because AI processes large amounts of data much faster and makes predictions more accurately than humanly possible.

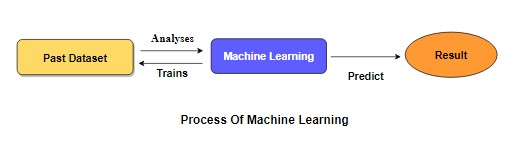
**Natural Language Processing (NLP):**

[Natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing) (NLP) allows machines to read and [understand](https://en.wikipedia.org/wiki/Natural-language_understanding) human language. A sufficiently powerful natural language processing system would enable [natural-language user interfaces](https://en.wikipedia.org/wiki/Natural-language_user_interface) and the acquisition of knowledge directly from human-written sources, such as newswire texts. Some straightforward applications of natural language processing include [information retrieval](https://en.wikipedia.org/wiki/Information_retrieval), [text mining](https://en.wikipedia.org/wiki/Text_mining), [question answering](https://en.wikipedia.org/wiki/Question_answering) and [machine translation](https://en.wikipedia.org/wiki/Machine_translation). Many current approaches use word co-occurrence frequencies to construct syntactic representations of text. "Keyword spotting" strategies for search are popular and scalable but dumb; a search query for "dog" might only match documents with the literal word "dog" and miss a document with the word "poodle". "Lexical affinity" strategies use the occurrence of words such as "accident" to [assess the sentiment](https://en.wikipedia.org/wiki/Sentiment_analysis) of a document. Modern statistical NLP approaches can combine all these strategies as well as others, and often achieve acceptable accuracy at the page or paragraph level. Beyond semantic NLP, the ultimate goal of "narrative" NLP is to embody a full understanding of common sense reasoning. By 2019, [transformer](https://en.wikipedia.org/wiki/Transformer_(machine_learning_model))-based deep learning architectures could generate coherent text.

**4. MACHINE LEARNING**

Machine learning is to predict the future from past data. Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data and the basics of Machine Learning, implementation of a simple machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms. It feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. Machine learning can be roughly separated in to three categories. There are supervised learning, unsupervised learning and reinforcement learning. Supervised learning program is both given the input data and the corresponding labelling to learn data has to be labelled by a human being beforehand. Unsupervised learning is no labels. It provided to the learning algorithm. This algorithm has to figure out the clustering of the input data. Finally, Reinforcement learning dynamically interacts with its environment and it receives positive or negative feedback to improve its performance.

Data scientists use many different kinds of machine learning algorithms to discover patterns in python that lead to actionable insights. At a high level, these different algorithms can be classified into two groups based on the way they “learn” about data to make predictions: supervised and unsupervised learning. Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function from input variables(X) to discrete output variables(y). In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc.



[Supervised Machine Learning](https://www.geeksforgeeks.org/supervised-unsupervised-learning/) **is the** majority of practical machine learning uses supervised learning. Supervised learning is where have input variables (X) and an output variable (y) and use an algorithm to learn the mapping function from the input to the output**is y = f(X).** The goal is to approximate the mapping function so well that when you have new input data (X) that you can predict the output variables (y) for that data. Techniques of Supervised Machine Learning algorithms include **logistic regression**, **multi-class classification**, **Decision Trees** and **support vector machines etc**. Supervised learning requires that the data used to train the algorithm is already labelled with correct answers. Supervised learning problems can be further grouped into **Classification** problems. This problem has as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for categorical for classification. A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. A classification problem is when the output variable is a category, such as “red” or “blue”.

**06. PROPOSED SYSTEM:**

The proposed system aims to revolutionize dairy farming through advanced machine learning techniques tailored for precise milk quality predictions. By integrating data from various sources such as cow health records, environmental conditions, and feed composition, the system will employ supervised learning algorithms to forecast milk quality metrics like fat content, protein levels, and somatic cell counts. Utilizing historical data, the model will continuously improve its accuracy, enabling dairy farmers to preemptively address potential issues that affect milk quality, such as diet adjustments or health interventions for cows. Real-time monitoring and predictive analytics will empower farmers to optimize their operations, enhance herd management practices, and ensure consistent high-quality milk production, thereby promoting sustainability and profitability in the dairy industry.

**6.1ADVANTAGES:**

* Early Detection
* Improved Decision-Making
* Consistent Quality Assurance
* Sustainability and Efficiency
* High scalability and Adaptability
* Empowerment of Farmers
* Optimized Resource Allocation

**Work Flow Diagram**

Dataset

Test dataset

Data Processing

Training dataset

Classification ML Algorithm

Model

**07.LITERATURE REVIEW:**

**General**

In the present world the farming is neglected and everyone is moving into modernization. No one don’t know how to do farming

**Review of Literature Survey**

**Title :** MEDIFORECAST: MULTIPLE DISEASE PREDICTION

**Author:** Vanita Gadekar, Tejas Vishnu Gund,,

**Year :2024**

Early disease detection is paramount for effective healthcare management. In this research, titled "MediForecast: Multiple Disease Prediction," we address this critical challenge by harnessing the power of machine learning, specifically employing the Support Vector Machine (SVM) classifier algorithm. Focusing on heart disease, Parkinson's disease, and diabetes, we explore innovative approaches to predict these conditions accurately. Our methodology involves meticulous data collection, preprocessing, and feature selection tailored for each disease. We employ the SVM classifier to create robust prediction models. Our implementation demonstrates the practical application of these models, showcasing their effectiveness in diagnosing the aforementioned diseases. The results reveal promising outcomes, indicating high accuracy, sensitivity, and specificity in disease prediction. By empowering medical professionals with timely and precise predictive capabilities, our research contributes significantly to the advancement of healthcare practices. We highlight the transformative potential of machine learning, particularly the SVM classifier, in revolutionizing disease diagnosis, paving the way for a healthier future.

**Title :** MATERNAL RISK LEVEL PREDICTION USING ENSEMBLE MODEL .

**Author:**  Nirmala, Rekha S Kambli

**Year : 2023**

In the context of Bangladesh, this work has created a system for accurately monitoring and forecasting a pregnant woman's risk level. Pregnant women's health information and risk factors will be examined by this method to determine the risk intensity level. By 2030, the United Nations wants to lower mother and infant deaths and improve maternal health, but the rate is not declining as quickly as it should. This study evaluated the risk level based on risk factors in pregnancy using the relevant analytical tools and machine learning algorithms. Data on maternal health was acquired for this study from the UCI machine learning library. Risk has been examined using categorization and classification techniques based on risk level. The Random Forest Algorithm provides the highest accuracy of 96% for training data, when it comes to classification and prediction of the risk level, according to a comparison of certain groups of machine learning algorithms.

**Title :** Automated Pregnancy Risk Level Prediction using Advanced Machine Learning and Deep Learning Algorithm .

**Author:** Proloy Karmakar.

**Year :** 2025

Md Sazzad Hossain Department of Computer Mechanical Engineering, Sonargaon University, Dhaka, Bangladesh In this study we analyzed different well-established machine learning (ML) and deep learning (DL) supervised models to enable the risk prediction of maternal health, thus offering a viable and systematic technique to automatically identify pregnancy risk. The Maternal Health Risk Data Set, which covers various critical attributes such as age, blood pressure, blood sugar, body temperature, heart rate, and risk level, was applied [8]. Data pretreatment methods, including deleting missing data (if any) and conducting feature scaling and selection, were incorporated to create the model. Different ML models were created and tested, including but not limited to Support Vector Machine (SVM), Random Forest (RF), and Gradient Boosting (GB), as well as deep learning architectures such as Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Long Short-Term Memory (LSTM). Model performances were evaluated using metric measures, including accuracy and F1 scores. Of them, CNN showed the highest accuracy (98.58), exceeding alternative models by its capacity to uncover spatial correlations crucial for successful risk prediction. CNN shows great accuracy, which indicates that its real-life clinical application in predicting high-risk pregnancies would result in a considerable improvement in maternal care. Adding AI-driven models to existing healthcare settings could assist in the faster and more accurate evaluation of pregnancy risk, particularly in low resource settings, boosting focused preventative therapy and evidence-based clinical decision-making. The expanding presence of AI has the potential to revolutionize healthcare, taking us closer to scalable automated solutions for maternal health that correspond with global healthcare development goals, with implications from this study.

**Title :** MODEL FOR PREDICTING RISK LEVELS IN MATERNAL HEALTHCARE.

**Author:**  Rekha S Kambli1, Nirmala,

**Year : 2022**

In the context of Bangladesh, a system has been built in this study for efficiently tracking and forecasting a pregnant woman's risk level. To determine the risk intensity level, this system will analyze the health information and risk factors of pregnant women. By 2030, the United Nations wants to improve maternal health and reduce mother and infant mortality; however, the rate is not decreasing as quickly as it should be. This study aimed to determine the risk level based on risk factors in pregnancy using appropriate analytical tools and machine learning algorithms. Data on maternal health were retrieved from the UCI machine learning repository for this study. According to the level of risk, categorization and classification methods have been utilized for the examination of risk variables. The AdaBoost Algorithm provides the highest AUC of 96%, when it comes to classification and prediction of the risk level, according to a comparison of certain groups of machine learning algorithms.

**Title :** Uncertainty-Aware Pre-Trained Foundation Models for Patient Risk Prediction via Gaussian Process.

**Author:** Jiaying Lu.

**Year :** 2022

Patient risk prediction models are crucial as they enable health care providers to proactively identify and address potential health risks. Large pre-trained foundation models offer remarkable per formance in risk prediction tasks by analyzing multimodal patient data. However, a notable limitation of pre-trained foundation mod els lies in their deterministic predictions (i.e., lacking the ability to acknowledge uncertainty). We propose Gaussian Process-based foundation models to enable the generation of accurate predictions with instance-level uncertainty quantification, thus allowing health care professionals to make more informed and cautious decisions. Our proposed approach is principled and architecture-agnostic. Experimental results show that our proposed approach achieves competitive performance on classical classification metrics. More over, we observe that the accuracy of certain predictions is much higher than that of the uncertain ones, which validates the uncer tainty awareness of our proposed method. Therefore, healthcare providers can trust low-uncertainty predictions and conduct more GP Pre-trained Foundation Model NEG POS comprehensive investigations on high-uncertainty predictions, ul timately enhancing patient outcomes with less expert intervention.

**8.SYSTEM STUDY**

**8.1. AIM:**

To develop and deploy an intelligent Clinical Decision Support System (CDSS) using advanced machine learning models integrated with the Django framework to accurately predict patient risk levels, enhance early diagnosis, and support healthcare professionals in making timely and data-driven clinical decisions.

### **OBJECTIVES:**

1. **To collect and preprocess diverse patient health records and clinical datasets for training predictive models.**
2. **To implement and compare advanced machine learning algorithms (e.g.,**
3. **To evaluate model performance using metrics such as accuracy, precision, recall, F1-score, and AUC-ROC curve.**
4. **To integrate the best-performing model into a Django-based web application for patient risk assessment.**
5. **To design a user-friendly interface for healthcare professionals to input patient data and receive risk predictions and clinical recommendations.**
6. **To ensure system scalability, data security, and compliance with healthcare data privacy regulations.**
7. **To validate the proposed system with clinical experts and assess its effectiveness in supporting decision-making in real-world healthcare environments.**

**8.3.Scope of the Project**

The scope of this project encompasses the development and integration of advanced machine learning models for the predictive analysis of patient risk levels within Clinical Decision Support Systems (CDSS), utilizing the Django framework for seamless web-based deployment. The system aims to process and analyze historical and patient health data, including vital signs, medical history, and lab reports, to classify patients into various risk categories such as low, medium, or high. By leveraging powerful algorithms like the project supports healthcare professionals in making accurate, timely, and data-driven decisions. The Django framework facilitates secure data handling, user-friendly interfaces, and integration with electronic health record systems, ultimately enhancing the quality of patient care and reducing the likelihood of clinical errors.

**09.Feasibility study:**

## **Data Wrangling**

## In this section of the report will load in the data, check for cleanliness, and then trim and clean given dataset for analysis. Make sure that the document steps carefully and justify for cleaning decisions.

**Data collection**

The data set collected for predicting given data is split into Training set and Test set. Generally, 7:3 ratios are applied to split the Training set and Test set. The Data Model which was created using Random Forest, Naive Bayes, Decision Tree and Logistic Regression are applied on the Training set and based on the test result accuracy, Test set prediction is done.

**Preprocessing**

The data which was collected might contain missing values that may lead to inconsistency. To gain better results data need to be preprocessed so as to improve the efficiency of the algorithm. The outliers have to be removed and also variable conversion need to be done.

**Building the classification model**

The prediction of Cirrhosis in a high accuracy prediction model is effective because of the following reasons: It provides better results in classification problem.

* It is strong in preprocessing outliers, irrelevant variables, and a mix of continuous, categorical and discrete variables.
* It produces out of bag estimate error which has proven to be unbiased in many tests and it is relatively easy to tune with.

**Construction of a Predictive Model**

Machine learning needs data gathering have lot of past data’s. Data gathering have sufficient historical data and raw data. Before data pre-processing, raw data can’t be used directly. It’s used to pre-process then, what kind of algorithm with model. Training and testing this model working and predicting correctly with minimum errors. Tuned model involved by tuned time to time with improving the accuracy.

Data Pre-Processing

Data Gathering

Choose model

Train model

Test model

Tune model

Prediction

Process of dataflow diagram

**10.LIST OF MODULES:**

* Data Pre-processing
* Data Analysis of Visualization
* SUPPORT VECTOR MACHINE
* ADABOOST CLASSIFER
* RANDOM FOREST CLASSIFER
* Deployment Using DJANGO

**11.Project Requirements**

**General:**

Requirements are the basic constrains that are required to develop a system. Requirements are collected while designing the system. The following are the requirements that are to be discussed.

1. Functional requirements

2. Non-Functional requirements

3. Environment requirements

A. Hardware requirements

B. software requirements

**11.1. Functional requirements:**

The software requirements specification is a technical specification of requirements for the software product. It is the first step in the requirements analysis process. It lists requirements of a particular software system. The following details to follow the special libraries like sk-learn, pandas, numpy, matplotlib and seaborn.

**11.2. Non-Functional Requirements:**

Process of functional steps,

1. Problem define
2. Preparing data
3. Evaluating algorithms
4. Improving results
5. Prediction the result

**12.Environmental Requirements:**

1. Software Requirements:

Operating System : Windows 10 or later

Tool : Anaconda with Jupyter Notebook

2. Hardware requirements:

Processor : Intel i3

Hard disk : minimum 10 GB

RAM : minimum 4 GB

**13. SOFTWARE DESCRIPTION**

Anaconda is a freehand distribution of the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) and [R](https://en.wikipedia.org/wiki/R_(programming_language)) programming languages for [scientific computing](https://en.wikipedia.org/wiki/Scientific_computing) ([data science](https://en.wikipedia.org/wiki/Data_science), [machine learning](https://en.wikipedia.org/wiki/Machine_learning) applications, large-scale data processing, [predictive analytics](https://en.wikipedia.org/wiki/Predictive_analytics), etc.), that aims to simplify [package management](https://en.wikipedia.org/wiki/Package_management) and deployment. Package versions are managed by the [package management system](https://en.wikipedia.org/wiki/Package_manager) “Conda”. The Anaconda distribution is used by over 12 million users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS. So, Anaconda distribution comes with more than 1,400 packages as well as the [Conda](https://en.wikipedia.org/wiki/Conda_(package_manager)) package and virtual environment manager called Anaconda Navigator and it eliminates the need to learn to install each library independently. The open source packages can be individually installed from the Anaconda repository with the conda install command or using the pip install command that is installed with Anaconda. [Pip packages](https://en.wikipedia.org/wiki/Pip_(package_manager)) provide many of the features of conda packages and in most cases they can work together. Custom packages can be made using the conda build command, and can be shared with others by uploading them to Anaconda Cloud, [PyPI](https://en.wikipedia.org/wiki/Python_Package_Index) or other repositories. The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, you can create new environments that include any version of Python packaged with conda.

**13.1 ANACONDA NAVIGATOR**

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository.

Anaconda. Now, if you are primarily doing data science work, Anaconda is also a great option. Anaconda is created by Continuum Analytics, and it is a Python distribution that comes preinstalled with lots of useful python libraries for data science.

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.

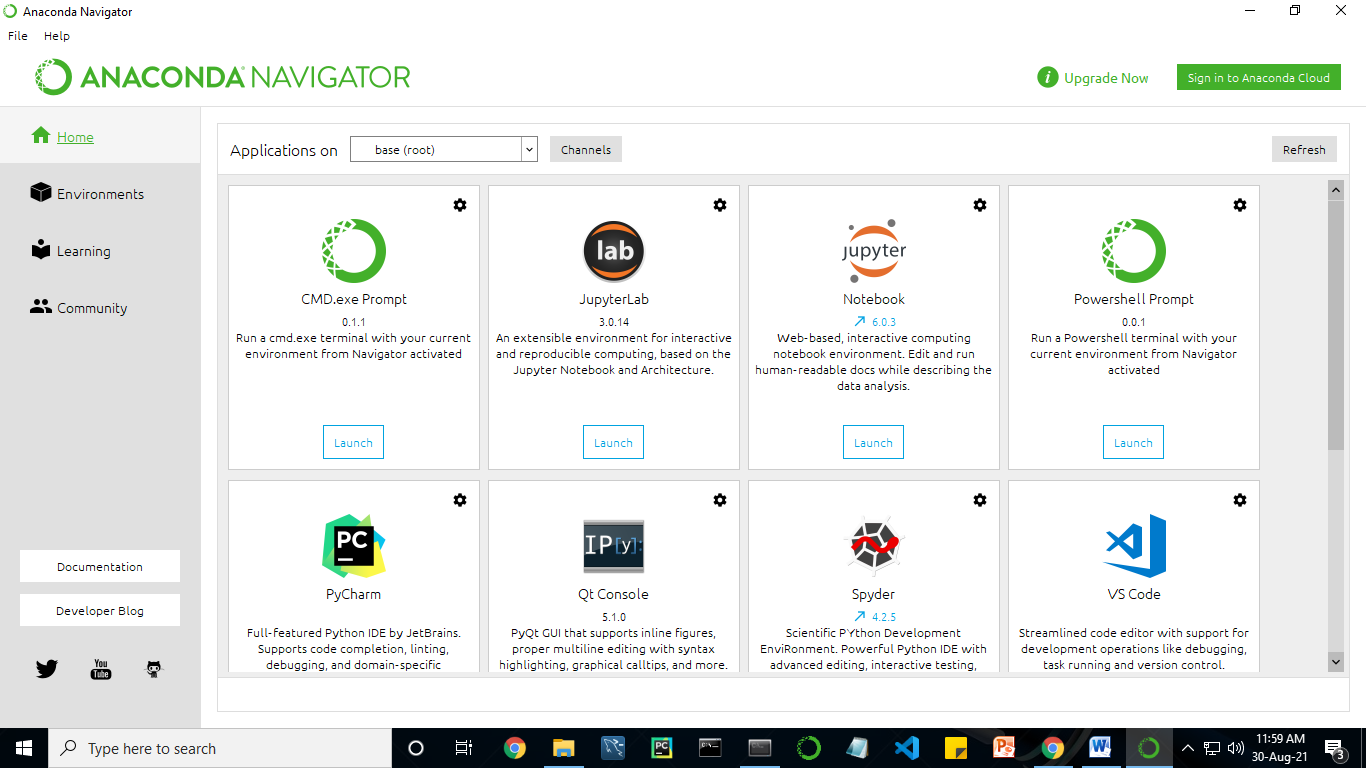
In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.

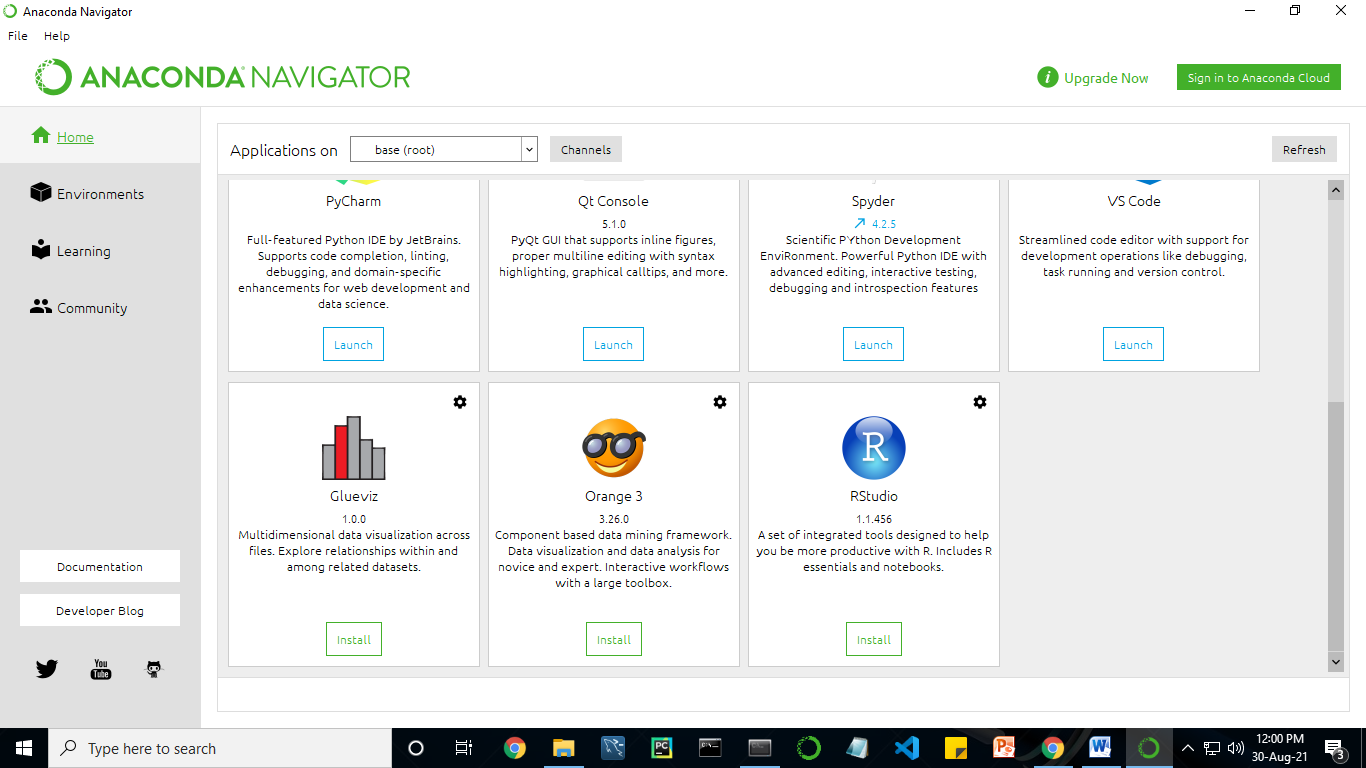
The command-line program conda is both a package manager and an environment manager. This helps data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages, and update them – all inside Navigator.

The following applications are available by default in Navigator:

* [JupyterLab](https://jupyterlab.readthedocs.io/en/stable/)
* [Jupyter Notebook](https://jupyter.readthedocs.io/en/latest/)
* [Spyder](https://www.spyder-ide.org/)
* [PyCharm](https://www.jetbrains.com/pycharm/documentation/)
* [VSCode](https://code.visualstudio.com/docs)
* [Glueviz](http://glueviz.org/en/stable/)
* [Orange 3 App](http://orange.biolab.si/docs/)
* [RStudio](http://docs.rstudio.com/)
* Anaconda Prompt (Windows only)
* Anaconda PowerShell (Windows only)





Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution.

Navigator allows you to launch common Python programs and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository.

Anaconda comes with many built-in packages that you can easily find with conda list on your anaconda prompt. As it has lots of packages (many of which are rarely used), it requires lots of space and time as well. If you have enough space, time and do not want to burden yourself to install small utilities like JSON, YAML, you better go for Anaconda.

**Conda :**

Conda is an open source, cross-platform, language-agnostic package manager and environment management systemthat installs, runs, and updates packages and their dependencies. It was created for Python programs, but it can package and distribute software for any language (e.g., R), including multi-language projects. The conda package and environment manager is included in all versions of Anaconda, Miniconda, and Anaconda Repository.

Anaconda is freely available, open source distribution of python and R programming languages which is used for scientific computations. If you are doing any machine learning or deep learning project then this is the best place for you. It consists of many softwares which will help you to build your machine learning project and deep learning project. These softwares have great graphical user interface and these will make your work easy to do. You can also use it to run your python script. These are the software carried by anaconda navigator.

**13.2 JUPYTER NOTEBOOK**

This website acts as “meta” documentation for the Jupyter ecosystem. It has a collection of resources to navigate the tools and communities in this ecosystem, and to help you get started.

Project Jupyter is a project and community whose goal is to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages". It was spun off from IPython in 2014 by Fernando Perez.

Notebook documents are documents produced by the [Jupyter Notebook App](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#notebook-app), which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc…). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc.) as well as executable documents which can be run to perform data analysis.

## Installation:The easiest way to install the Jupyter Notebook App is installing a scientific python distribution which also includes scientific python packages. The most common distribution is called **Anaconda**

# Running the Jupyter Notebook

## Launching Jupyter Notebook App: The [Jupyter Notebook App](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#notebook-app) can be launched by clicking on the Jupyter Notebook icon installed by Anaconda in the start menu (Windows) or by typing in a terminal (cmd on Windows): “jupyter notebook”

## This will launch a new browser window (or a new tab) showing the [Notebook Dashboard](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#dashboard), a sort of control panel that allows (among other things) to select which notebook to open.

## When started, the [Jupyter Notebook App](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#notebook-app) can access only files within its start-up folder (including any sub-folder). No configuration is necessary if you place your notebooks in your home folder or subfolders. Otherwise, you need to choose a [Jupyter Notebook App](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#notebook-app) start-up folder which will contain all the notebooks.

## Save notebooks:Modifications to the notebooks are automatically saved every few minutes. To avoid modifying the original notebook, make a copy of the notebook document (menu file -> make a copy…) and save the modifications on the copy.

## Executing a notebook:Download the notebook you want to execute and put it in your notebook folder (or a sub-folder of it).

* Launch the jupyter notebook app
* In the [Notebook Dashboard](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#dashboard) navigate to find the notebook: clicking on its name will open it in a new browser tab.
* Click on the menu Help -> User Interface Tour for an overview of the [Jupyter Notebook App](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#notebook-app) user interface.
* You can run the notebook document step-by-step (one cell a time) by pressing shift + enter.
* You can run the whole notebook in a single step by clicking on the menu Cell -> Run All.
* To restart the [kernel](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#kernel) (i.e. the computational engine), click on the menu Kernel -> Restart. This can be useful to start over a computation from scratch (e.g. variables are deleted, open files are closed, etc…).

[**Purpose**](https://www.google.com/search?q=project+jupyter+purpose&sa=X&ved=2ahUKEwin49vtmdjyAhXx4zgGHXSOCuwQ6BMoADAkegQINxAC&cshid=1630307847256010)**:** To support [interactive](https://www.google.com/search?q=interactive&stick=H4sIAAAAAAAAAONgVuLUz9U3MM0uyYpfxMqdmVeSWpSYXJJZlgoApkTFPhsAAAA&sa=X&ved=2ahUKEwin49vtmdjyAhXx4zgGHXSOCuwQmxMoATAkegQINxAD&cshid=1630307847256010) data science and scientific computing across all programming languages.

**File Extension:** An **IPYNB** file is a notebook document created by Jupyter Notebook, an interactive computational environment that helps scientists manipulate and analyze data using Python.

**13.2 JUPYTER NOTEBOOK**

This website acts as “meta” documentation for the Jupyter ecosystem. It has a collection of resources to navigate the tools and communities in this ecosystem, and to help you get started.

Project Jupyter is a project and community whose goal is to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages". It was spun off from IPython in 2014 by Fernando Perez.

Notebook documents are documents produced by the [Jupyter Notebook App](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#notebook-app), which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc…). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc.) as well as executable documents which can be run to perform data analysis.

## Installation:The easiest way to install the Jupyter Notebook App is installing a scientific python distribution which also includes scientific python packages. The most common distribution is called **Anaconda**

# Running the Jupyter Notebook

## Launching Jupyter Notebook App: The [Jupyter Notebook App](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#notebook-app) can be launched by clicking on the Jupyter Notebook icon installed by Anaconda in the start menu (Windows) or by typing in a terminal (cmd on Windows): “jupyter notebook”

## This will launch a new browser window (or a new tab) showing the [Notebook Dashboard](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#dashboard), a sort of control panel that allows (among other things) to select which notebook to open.

## When started, the [Jupyter Notebook App](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#notebook-app) can access only files within its start-up folder (including any sub-folder). No configuration is necessary if you place your notebooks in your home folder or subfolders. Otherwise, you need to choose a [Jupyter Notebook App](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#notebook-app) start-up folder which will contain all the notebooks.

## Save notebooks:Modifications to the notebooks are automatically saved every few minutes. To avoid modifying the original notebook, make a copy of the notebook document (menu file -> make a copy…) and save the modifications on the copy.

## Executing a notebook:Download the notebook you want to execute and put it in your notebook folder (or a sub-folder of it).

* Launch the jupyter notebook app
* In the [Notebook Dashboard](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#dashboard) navigate to find the notebook: clicking on its name will open it in a new browser tab.
* Click on the menu Help -> User Interface Tour for an overview of the [Jupyter Notebook App](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#notebook-app) user interface.
* You can run the notebook document step-by-step (one cell a time) by pressing shift + enter.
* You can run the whole notebook in a single step by clicking on the menu Cell -> Run All.
* To restart the [kernel](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#kernel) (i.e. the computational engine), click on the menu Kernel -> Restart. This can be useful to start over a computation from scratch (e.g. variables are deleted, open files are closed, etc…).

[**Purpose**](https://www.google.com/search?q=project+jupyter+purpose&sa=X&ved=2ahUKEwin49vtmdjyAhXx4zgGHXSOCuwQ6BMoADAkegQINxAC&cshid=1630307847256010)**:** To support [interactive](https://www.google.com/search?q=interactive&stick=H4sIAAAAAAAAAONgVuLUz9U3MM0uyYpfxMqdmVeSWpSYXJJZlgoApkTFPhsAAAA&sa=X&ved=2ahUKEwin49vtmdjyAhXx4zgGHXSOCuwQmxMoATAkegQINxAD&cshid=1630307847256010) data science and scientific computing across all programming languages.

**File Extension:** An **IPYNB** file is a notebook document created by Jupyter Notebook, an interactive computational environment that helps scientists manipulate and analyze data using Python.

**JUPYTER Notebook App:**

The Jupyter Notebook App is a server-client application that allows editing and running [notebook documents](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#notebook-document) via a web browser.

The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.

In addition to displaying/editing/running notebook documents, the Jupyter Notebook App has a “Dashboard” ([Notebook Dashboard](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#dashboard)), a “control panel” showing local files and allowing to open notebook documents or shutting down their [kernels](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#kernel).

## [kernel](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#id7): A notebook kernel is a “computational engine” that executes the code contained in a [Notebook document](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#notebook-document). The ipython kernel, referenced in this guide, executes python code. Kernels for many other languages exist ([official kernels](http://jupyter.readthedocs.org/en/latest/#kernels)).

When you open a [Notebook document](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#notebook-document), the associated kernel is automatically launched. When the notebook is executed (either cell-by-cell or with menu Cell -> Run All), the kernel performs the computation and produces the results.

Depending on the type of computations, the kernel may consume significant CPU and RAM. Note that the RAM is not released until the kernel is shut-down

## [Notebook Dashboard](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#id8):The Notebook Dashboard is the component which is shown first when you launch [Jupyter Notebook App](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#notebook-app). The Notebook Dashboard is mainly used to open [notebook documents](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#notebook-document), and to manage the running [kernels](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#kernel) (visualize and shutdown).

The Notebook Dashboard has other features similar to a file manager, namely navigating folders and renaming/deleting files

**Working Process:**

* Download and install anaconda and get the most useful package for machine learning in Python.
* Load a dataset and understand its structure using statistical summaries and data visualization.
* Machine learning models, pick the best and build confidence that the accuracy is reliable.

Python is a popular and powerful interpreted language. Unlike R, Python is a complete language and platform that you can use for both research and development and developing production systems. There are also a lot of modules and libraries to choose from, providing multiple ways to do each task. It can feel overwhelming.

The best way to get started using Python for machine learning is to complete a project.

* It will force you to install and start the Python interpreter (at the very least).
* It will give you a bird’s eye view of how to step through a small project.
* It will give you confidence, maybe to go on to your own small projects.

When you are applying machine learning to your own datasets, you are working on a project. A machine learning project may not be linear, but it has a number of well-known steps:

* Define Problem.
* Prepare Data.
* Evaluate Algorithms.
* Improve Results.
* Present Results.

The best way to really come to terms with a new platform or tool is to work through a machine learning project end-to-end and cover the key steps. Namely, from loading data, summarizing data, evaluating algorithms and making some predictions.

Here is an overview of what we are going to cover:

1. Installing the Python anaconda platform.
2. Loading the dataset.
3. Summarizing the dataset.
4. Visualizing the dataset.
5. Evaluating some algorithms.
6. Making some predictions.

**14. PYTHON**

**Introduction:**

**Python** is an [interpreted](https://en.wikipedia.org/wiki/Interpreted_language) [high-level](https://en.wikipedia.org/wiki/High-level_programming_language) [general-purpose programming language](https://en.wikipedia.org/wiki/General-purpose_programming_language). Its design philosophy emphasizes [code readability](https://en.wikipedia.org/wiki/Code_readability) with its use of [significant indentation](https://en.wikipedia.org/wiki/Off-side_rule). Its [language constructs](https://en.wikipedia.org/wiki/Language_construct) as well as its [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) approach aim to help [programmers](https://en.wikipedia.org/wiki/Programmers) write clear, logical code for small and large-scale projects.

Python is [dynamically-typed](https://en.wikipedia.org/wiki/Type_system#DYNAMIC) and [garbage-collected](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigms), including [structured](https://en.wikipedia.org/wiki/Structured_programming) (particularly, [procedural](https://en.wikipedia.org/wiki/Procedural_programming)), object-oriented and [functional programming](https://en.wikipedia.org/wiki/Functional_programming). It is often described as a "batteries included" language due to its comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library).

[Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) began working on Python in the late 1980s, as a successor to the [ABC programming language](https://en.wikipedia.org/wiki/ABC_(programming_language)), and first released it in 1991 as Python 0.9.0.[Python 2.0 was released in 2000 and introduced new features, such as](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-32)[list comprehensions](https://en.wikipedia.org/wiki/List_comprehension) and a garbage collection system using [reference counting](https://en.wikipedia.org/wiki/Reference_counting). Python 3.0 was released in 2008 and was a major revision of the language that is not completely [backward-compatible](https://en.wikipedia.org/wiki/Backward_compatibility). Python 2 was discontinued with version 2.7.18 in 2020.

Python consistently ranks as one of the most popular programming languages

**History:**

Python was conceived in the late 1980s by [Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) at [Centrum Wiskunde & Informatica](https://en.wikipedia.org/wiki/Centrum_Wiskunde_%26_Informatica) (CWI) in the [Netherlands](https://en.wikipedia.org/wiki/Netherlands) as a successor to [ABC programming language](https://en.wikipedia.org/wiki/ABC_(programming_language)), which was inspired by [SETL](https://en.wikipedia.org/wiki/SETL),  capable of [exception handling](https://en.wikipedia.org/wiki/Exception_handling) and interfacing with the [Amoeba](https://en.wikipedia.org/wiki/Amoeba_(operating_system)) operating system. Its implementation began in December 1989.  Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's [Benevolent Dictator For Life](https://en.wikipedia.org/wiki/Benevolent_Dictator_For_Life), a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. In January 2019, active Python core developers elected a 5-member "Steering Council" to lead the project.  As of 2021, the current members of this council are Barry Warsaw, Brett Cannon, Carol Willing, Thomas Wouters, and Pablo Galindo Salgado.

Python 2.0 was released on 16 October 2000, with many major new features, including a [cycle-detecting](https://en.wikipedia.org/wiki/Cycle_detection) [garbage collector](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)) and support for [Unicode](https://en.wikipedia.org/wiki/Unicode).

Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely [backward-compatible](https://en.wikipedia.org/wiki/Backward_compatibility). Many of its major features were [backported](https://en.wikipedia.org/wiki/Backporting) to Python 2.6.x and 2.7.x version series. Releases of Python 3 include the 2 to 3 utility, which automates (at least partially) the translation of Python 2 code to Python 3.

Python 2.7's [end-of-life](https://en.wikipedia.org/wiki/End-of-life_(product)) date was initially set at 2015 then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3. No more security patches or other improvements will be released for it. With Python 2's [end-of-life](https://en.wikipedia.org/wiki/End-of-life_(product)), only Python 3.6.x  and later are supported.

Python 3.9.2 and 3.8.8 were expeditedas all versions of Python (including 2.7) had security issues, leading to possible [remote code execution](https://en.wikipedia.org/wiki/Remote_code_execution) and [web cache poisoning](https://en.wikipedia.org/wiki/Cache_poisoning).

**Design Philosophy & Feature**

Python is a [multi-paradigm programming language](https://en.wikipedia.org/wiki/Multi-paradigm_programming_language). [Object-oriented programming](https://en.wikipedia.org/wiki/Object-oriented_programming) and [structured programming](https://en.wikipedia.org/wiki/Structured_programming) are fully supported, and many of its features support functional programming and [aspect-oriented programming](https://en.wikipedia.org/wiki/Aspect-oriented_programming) (including by [meta-programming](https://en.wikipedia.org/wiki/Metaprogramming) and [meta-objects](https://en.wikipedia.org/wiki/Metaobject) (magic methods)). Many other paradigms are supported via extensions, including [design by contract](https://en.wikipedia.org/wiki/Design_by_contract) and [logic programming](https://en.wikipedia.org/wiki/Logic_programming).

Python uses [dynamic typing](https://en.wikipedia.org/wiki/Dynamic_typing) and a combination of [reference counting](https://en.wikipedia.org/wiki/Reference_counting) and a cycle-detecting garbage collector for [memory management](https://en.wikipedia.org/wiki/Memory_management). It also features dynamic [name resolution](https://en.wikipedia.org/wiki/Name_resolution_(programming_languages)) ([late binding](https://en.wikipedia.org/wiki/Late_binding)), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the [Lisp](https://en.wikipedia.org/wiki/Lisp_(programming_language)) tradition. It has filter, map and reduce functions;  [list comprehensions](https://en.wikipedia.org/wiki/List_comprehension), [dictionaries](https://en.wikipedia.org/wiki/Associative_array), sets, and [generator](https://en.wikipedia.org/wiki/Generator_(computer_programming)) expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from [Haskell](https://en.wikipedia.org/wiki/Haskell_(programming_language)) and [Standard ML](https://en.wikipedia.org/wiki/Standard_ML).

The language's core philosophy is summarized in the document The [Zen of Python](https://en.wikipedia.org/wiki/Zen_of_Python) (PEP 20), which includes [aphorisms](https://en.wikipedia.org/wiki/Aphorism) such as:

* Beautiful is better than ugly.
* Explicit is better than implicit.
* Simple is better than complex.
* Complex is better than complicated.
* Readability counts.

Rather than having all of its functionality built into its core, Python was designed to be highly [extensible](https://en.wikipedia.org/wiki/Extensibility) (with modules). This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with [ABC](https://en.wikipedia.org/wiki/ABC_(programming_language)), which espoused the opposite approach.

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to [Perl](https://en.wikipedia.org/wiki/Perl)'s "[there is more than one way to do it](https://en.wikipedia.org/wiki/There_is_more_than_one_way_to_do_it)" motto, Python embraces a "there should be one— and preferably only one —obvious way to do it" design philosophy. [Alex Martelli](https://en.wikipedia.org/wiki/Alex_Martelli), a [Fellow](https://en.wikipedia.org/wiki/Fellow) at the [Python Software Foundation](https://en.wikipedia.org/wiki/Python_Software_Foundation) and Python book author, writes that "To describe something as 'clever' is not considered a compliment in the Python culture."

Python's developers strive to avoid [premature optimization](https://en.wikipedia.org/wiki/Premature_optimization), and reject patches to non-critical parts of the [C-Python](https://en.wikipedia.org/wiki/CPython) reference implementation that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use [PyPy](https://en.wikipedia.org/wiki/PyPy), a [just-in-time compiler](https://en.wikipedia.org/wiki/Just-in-time_compilation). [Cython](https://en.wikipedia.org/wiki/Cython) is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

Python's developers aim to keep the language fun to use. This is reflected in its name a tribute to the British comedy group [Monty Python](https://en.wikipedia.org/wiki/Monty_Python) and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (a reference to a [Monty Python sketch](https://en.wikipedia.org/wiki/Spam_(Monty_Python))) instead of the standard [foo and bar](https://en.wikipedia.org/wiki/Foobar).

A common [neologism](https://en.wikipedia.org/wiki/Neologism) in the Python community is pythonic, which can have a wide range of meanings related to program style. To say that code is pythonic is to say that it uses Python idioms well, that it is natural or shows fluency in the language, that it conforms with Python's minimalist philosophy and emphasis on readability. In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called unpythonic.

Users and admirers of Python, especially those considered knowledgeable or experienced, are often referred to as Pythonistas

**Syntax and Semantics :**

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use [curly brackets](https://en.wikipedia.org/wiki/Curly_bracket_programming_language) to delimit blocks, and semicolons after statements are allowed but are rarely, if ever, used. It has fewer syntactic exceptions and special cases than [C](https://en.wikipedia.org/wiki/C_(programming_language)) or [Pascal](https://en.wikipedia.org/wiki/Pascal_(programming_language)).

**Indentation :**

Main article: [Python syntax and semantics & Indentation](https://en.wikipedia.org/wiki/Python_syntax_and_semantics#Indentation)

Python uses [whitespace](https://en.wikipedia.org/wiki/Whitespace_character) indentation, rather than [curly brackets](https://en.wikipedia.org/wiki/Curly_bracket_programming_language) or keywords, to delimit [blocks](https://en.wikipedia.org/wiki/Block_(programming)). An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. Thus, the program's visual structure accurately represents the program's semantic structure. This feature is sometimes termed the [off-side rule](https://en.wikipedia.org/wiki/Off-side_rule), which some other languages share, but in most languages indentation does not have any semantic meaning. The recommended indent size is four spaces.

**Statements and control flow :**

Python's [statements](https://en.wikipedia.org/wiki/Statement_(computer_science)) include:

* The [assignment](https://en.wikipedia.org/wiki/Assignment_(computer_science)) statement, using a single equals sign =.
* The if statement, which conditionally executes a block of code, along with else and elif (a contraction of else-if).
* The for statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block.
* The while statement, which executes a block of code as long as its condition is true.
* The Try statement, which allows exceptions raised in its attached code block to be caught and handled by except clauses; it also ensures that clean-up code in a finally block will always be run regardless of how the block exits.
* The raise statement, used to raise a specified exception or re-raise a caught exception.
* The class statement, which executes a block of code and attaches its local namespace to a [class](https://en.wikipedia.org/wiki/Class_(computer_science)), for use in object-oriented programming.
* The def statement, which defines a [function](https://en.wikipedia.org/wiki/Function_(computing)) or [method](https://en.wikipedia.org/wiki/Method_(computing)).
* The with statement, which encloses a code block within a context manager (for example, acquiring a [lock](https://en.wikipedia.org/wiki/Lock_(computer_science)) before the block of code is run and releasing the lock afterwards, or opening a [file](https://en.wikipedia.org/wiki/Computer_file) and then closing it), allowing [resource-acquisition-is-initialization](https://en.wikipedia.org/wiki/Resource_acquisition_is_initialization) (RAII) - like behavior and replaces a common try/finally idiom.
* The break statement, exits from a loop.
* The continue statement, skips this iteration and continues with the next item.
* The del statement, removes a variable, which means the reference from the name to the value is deleted and trying to use that variable will cause an error. A deleted variable can be reassigned.
* The pass statement, which serves as a [NOP](https://en.wikipedia.org/wiki/NOP_(code)). It is syntactically needed to create an empty code block.
* The assert statement, used during debugging to check for conditions that should apply.
* The yield statement, which returns a value from a [generator](https://en.wikipedia.org/wiki/Generator_(computer_programming)#Python) function and yield is also an operator. This form is used to implement [co-routines](https://en.wikipedia.org/wiki/Coroutine).
* The return statement, used to return a value from a function.
* The import statement, which is used to import modules whose functions or variables can be used in the current program.

The assignment statement (=) operates by binding a name as a [reference](https://en.wikipedia.org/wiki/Pointer_(computer_programming)) to a separate, dynamically-allocated [object](https://en.wikipedia.org/wiki/Object_(computer_science)). Variables may be subsequently rebound at any time to any object. In Python, a variable name is a generic reference holder and does not have a fixed [data type](https://en.wikipedia.org/wiki/Type_system) associated with it. However, at a given time, a variable will refer to some object, which will have a type. This is referred to as [dynamic typing](https://en.wikipedia.org/wiki/Dynamic_type) and is contrasted with [statically-typed](https://en.wikipedia.org/wiki/Statically-typed) programming languages, where each variable may only contain values of a certain type.

Python does not support [tail call](https://en.wikipedia.org/wiki/Tail_call) optimization or [first-class continuations](https://en.wikipedia.org/wiki/First-class_continuations), and, according to Guido van Rossum, it never will.[[80]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-AutoNT-55-80)[[81]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-AutoNT-56-81) However, better support for [co-routine](https://en.wikipedia.org/wiki/Coroutine)-like functionality is provided, by extending Python's [generators](https://en.wikipedia.org/wiki/Generator_(computer_programming)). Before 2.5, generators were [lazy](https://en.wikipedia.org/wiki/Lazy_evaluation) [iterators](https://en.wikipedia.org/wiki/Iterator); information was passed uni-directionally out of the generator. From Python 2.5, it is possible to pass information back into a generator function, and from Python 3.3, the information can be passed through multiple stack levels.

**Expressions** :

Some Python [expressions](https://en.wikipedia.org/wiki/Expression_(computer_science)) are similar to those found in languages such as C and [Java](https://en.wikipedia.org/wiki/Java_(programming_language)), while some are not:

* Addition, subtraction, and multiplication are the same, but the behavior of division differs. There are two types of divisions in Python. They are floor division (or integer division) // and floating-point/division. Python also uses the \*\* operator for exponentiation.
* From Python 3.5, the new @ infix operator was introduced. It is intended to be used by libraries such as [NumPy](https://en.wikipedia.org/wiki/NumPy) for [matrix multiplication](https://en.wikipedia.org/wiki/Matrix_multiplication).
* From Python 3.8, the syntax :=, called the 'walrus operator' was introduced. It assigns values to variables as part of a larger expression.
* In Python, == compares by value, versus Java, which compares numerics by value and objects by reference. (Value comparisons in Java on objects can be performed with the equals() method.) Python's is operator may be used to compare object identities (comparison by reference). In Python, comparisons may be chained, for example A<=B<=C.
* Python uses the words and, or, not for or its boolean operators rather than the symbolic &&, ||, ! used in Java and C.
* Python has a type of expression termed a [list comprehension](https://en.wikipedia.org/wiki/List_comprehension#Python) as well as a more general expression termed a [generator](https://en.wikipedia.org/wiki/Generator_(computer_programming)) expression.
* [Anonymous functions](https://en.wikipedia.org/wiki/Anonymous_function) are implemented using [lambda expressions](https://en.wikipedia.org/wiki/Lambda_(programming)); however, these are limited in that the body can only be one expression.
* Conditional expressions in Python are written as x if c else y (different in order of operands from the c ? x : y operator common to many other languages).
* Python makes a distinction between [lists](https://en.wikipedia.org/wiki/List_(computer_science)) and [tuples](https://en.wikipedia.org/wiki/Tuple). Lists are written as [1, 2, 3], are mutable, and cannot be used as the keys of dictionaries (dictionary keys must be [immutable](https://en.wikipedia.org/wiki/Immutable) in Python). Tuples are written as (1, 2, 3), are immutable and thus can be used as the keys of dictionaries, provided all elements of the tuple are immutable. The + operator can be used to concatenate two tuples, which does not directly modify their contents, but rather produces a new tuple containing the elements of both provided tuples. Thus, given the variable t initially equal to (1, 2, 3), executing t = t + (4, 5) first evaluates t + (4, 5), which yields (1, 2, 3, 4, 5), which is then assigned back to t, thereby effectively "modifying the contents" of t, while conforming to the immutable nature of tuple objects. Parentheses are optional for tuples in unambiguous contexts.
* Python features sequence unpacking wherein multiple expressions, each evaluating to anything that can be assigned to (a variable, a writable property, etc.), are associated in an identical manner to that forming tuple literals and, as a whole, are put on the left-hand side of the equal sign in an assignment statement. The statement expects an iterable object on the right-hand side of the equal sign that produces the same number of values as the provided writable expressions when iterated through and will iterate through it, assigning each of the produced values to the corresponding expression on the left.
* Python has a "string format" operator %. This functions analogously ton printf format strings in C, e.g. “spam=%s eggs=%d” % (“blah”,2) evaluates to “spam=blah eggs=2”. In Python 3 and 2.6+, this was supplemented by the format() method of the str class, e.g. “spam={0} eggs={1}”.format(“blah”,2). Python 3.6 added "f-strings": blah = “blah”; eggs = 2; f‘spam={blah} eggs={eggs}’
* Strings in Python can be [concatenated](https://en.wikipedia.org/wiki/Concatenation), by "adding" them (same operator as for adding integers and floats). E.g. “spam” + “eggs” returns “spameggs”. Even if your strings contain numbers, they are still added as strings rather than integers. E.g. “2” + “2” returns “2”.
* Python has various kinds of [string literals](https://en.wikipedia.org/wiki/String_literal):
  + Strings delimited by single or double quote marks. Unlike in [Unix shells](https://en.wikipedia.org/wiki/Unix_shell), [Perl](https://en.wikipedia.org/wiki/Perl) and Perl-influenced languages, single quote marks and double quote marks function identically. Both kinds of string use the backslash (\) as an [escape character](https://en.wikipedia.org/wiki/Escape_character). [String interpolation](https://en.wikipedia.org/wiki/String_interpolation) became available in Python 3.6 as "formatted string literals".
  + Triple-quoted strings, which begin and end with a series of three single or double quote marks. They may span multiple lines and function like [here documents](https://en.wikipedia.org/wiki/Here_document) in shells, Perl and [Ruby](https://en.wikipedia.org/wiki/Ruby_(programming_language)).
  + [Raw string](https://en.wikipedia.org/wiki/Raw_string) varieties, denoted by prefixing the string literal with an r. Escape sequences are not interpreted; hence raw strings are useful where literal backslashes are common, such as [regular expressions](https://en.wikipedia.org/wiki/Regular_expression) and [Windows](https://en.wikipedia.org/wiki/Microsoft_Windows)-style paths. Compare "@-quoting" in [C#](https://en.wikipedia.org/wiki/C_Sharp_(programming_language)).
* Python has [array index](https://en.wikipedia.org/wiki/Array_index) and [array slicing](https://en.wikipedia.org/wiki/Array_slicing) expressions on lists, denoted as a[Key], a[start:stop] or a[start:stop:step]. Indexes are [zero-based](https://en.wikipedia.org/wiki/Zero-based_numbering), and negative indexes are relative to the end. Slices take elements from the start index up to, but not including, the stop index. The third slice parameter, called step or stride, allows elements to be skipped and reversed. Slice indexes may be omitted, for example a[:] returns a copy of the entire list. Each element of a slice is a [shallow copy](https://en.wikipedia.org/wiki/Shallow_copy).

In Python, a distinction between expressions and statements is rigidly enforced, in contrast to languages such as [Common Lisp](https://en.wikipedia.org/wiki/Common_Lisp), [Scheme](https://en.wikipedia.org/wiki/Scheme_(programming_language)), or [Ruby](https://en.wikipedia.org/wiki/Ruby_(programming_language)). This leads to duplicating some functionality. For example:

* [List comprehensions](https://en.wikipedia.org/wiki/List_comprehensions) vs. for-loops
* [Conditional](https://en.wikipedia.org/wiki/Conditional_(programming)) expressions vs. if blocks
* The eval() vs. exec() built-in functions (in Python 2, exec is a statement); the former is for expressions, the latter is for statements.

Statements cannot be a part of an expression, so list and other comprehensions or [lambda expressions](https://en.wikipedia.org/wiki/Lambda_(programming)), all being expressions, cannot contain statements. A particular case of this is that an assignment statement such as a=1 cannot form part of the conditional expression of a conditional statement. This has the advantage of avoiding a classic C error of mistaking an assignment operator = for an equality operator == in conditions: if (c==1) {…} is syntactically valid (but probably unintended) C code but if c=1: … causes a syntax error in Python.

**Methods** :

[Methods](https://en.wikipedia.org/wiki/Method_(programming)) on objects are [functions](https://en.wikipedia.org/wiki/Function_(programming)) attached to the object's class; the syntax instance.method(argument) is, for normal methods and functions, [syntactic sugar](https://en.wikipedia.org/wiki/Syntactic_sugar) for Class.method(instance, argument). Python methods have an explicit self parameter access [instance data](https://en.wikipedia.org/wiki/Instance_data), in contrast to the implicit self (or this) in some other object-oriented programming languages (e.g., [C++](https://en.wikipedia.org/wiki/C%2B%2B), Java, [Objective-C](https://en.wikipedia.org/wiki/Objective-C), or [Ruby](https://en.wikipedia.org/wiki/Ruby_(programming_language))). Apart from this Python also provides methods, sometimes called d-under methods due to their names beginning and ending with double-underscores, to extend the functionality of custom class to support native functions such as print, length, comparison, support for arithmetic operations, type conversion, and many more.

### Typing :

Python uses duck typing and has typed objects but untyped variable names. Type constraints are not checked at compile time; rather, operations on an object may fail, signifying that the given object is not of a suitable type. Despite being dynamically-typed, Python is strongly-typed, forbidding operations that are not well-defined (for example, adding a number to a string) rather than silently attempting to make sense of them.

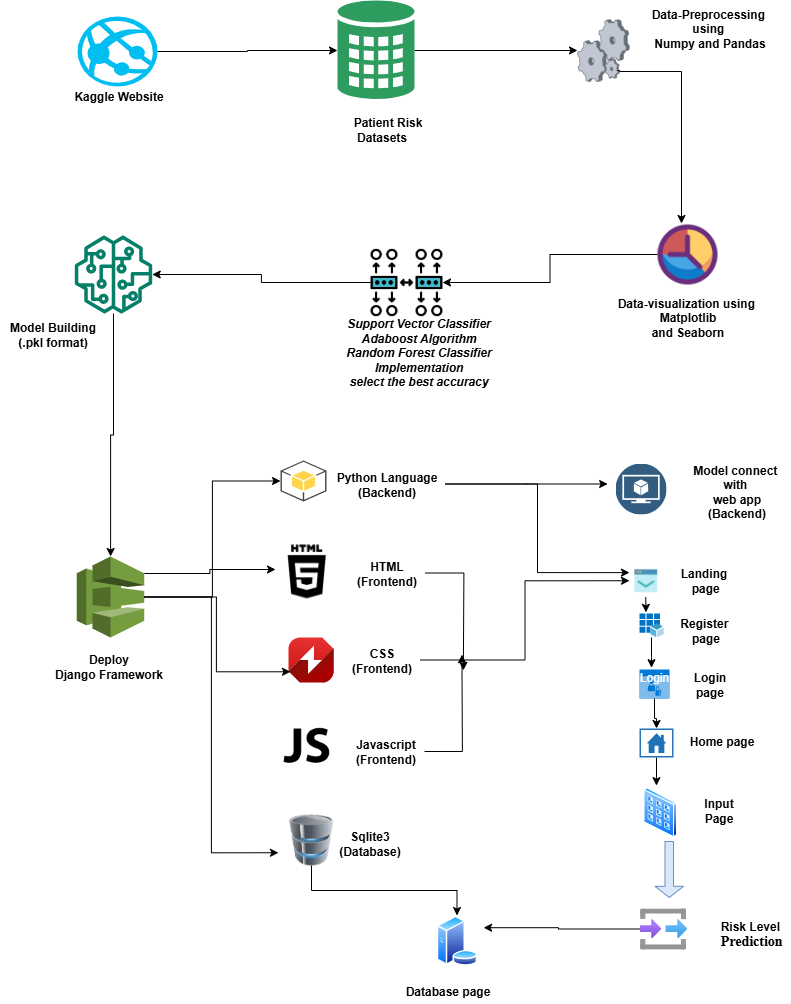
Python allows programmers to define their own types using [classes](https://en.wikipedia.org/wiki/Class_(computer_science)), which are most often used for object-oriented programming. New instances of classes are constructed by calling the class (for example, SpamClass() or EggsClass()), and the classes are instances of the metaclass type (itself an instance of itself), allowing meta-programming and reflection.

Before version 3.0, Python had two kinds of classes: old-style and new-style.The syntax of both styles is the same, the difference being whether the class object is inherited from, directly or indirectly (all new-style classes inherit from object and are instances of type). In versions of Python 2 from Python 2.2 onwards, both kinds of classes can be used. Old-style classes were eliminated in Python 3.0.

The long-term plan is to support gradual typing and from Python 3.5, the syntax of the language allows specifying static types but they are not checked in the default implementation, CPython[. An experimental optional static type checker named](https://en.wikipedia.org/wiki/CPython)mypy supports compile-time type checking.

**15. DESIGN ARCHITECTURE:**

**System Architecture:**

****

**16.Work flow diagram**

Source Data

Data Processing and Cleaning

Training Dataset

Testing Dataset

Best Model by Accuracy

Classification ML Algorithms

Patient Risk Level Prediction

A workflow diagram in the context of machine learning (ML) represents the sequence of steps or processes involved in a machine learning project. It outlines the high-level structure of how data is collected, prepared, processed, and evaluated to build and deploy machine learning models. Below is a description of the key components and steps typically found in a machine learning workflow diagram:

Data Collection:

The workflow begins with data collection. This step involves gathering relevant data from various sources, which may include databases, external datasets, APIs, or sensors.

Data collection may also encompass the annotation of data, such as labelling images or categorizing text.

Data Pre-processing:

After data collection, the raw data often needs to be cleaned and pre-processed. This step includes handling missing values, removing duplicates, and addressing outliers.

Data pre-processing also involves data normalization, scaling, and feature engineering to make the data suitable for modelling.

Data Splitting:

The dataset is typically divided into training, validation, and testing sets. This separation is crucial for training and evaluating the machine learning model effectively.

Feature Selection and Engineering:

Feature selection involves identifying and selecting the most relevant features from the dataset, which can improve the model's performance and reduce overfitting.

Feature engineering includes creating new features or transforming existing ones to capture meaningful information for the model.

Model Building:

In this step, machine learning models are designed, implemented, and trained using the pre-processed data. This may involve selecting a suitable algorithm, architecture, or framework.

Hyper parameter tuning, cross-validation, and other techniques are applied to optimize the model's performance.

Model Evaluation:

The trained models are evaluated using the validation dataset to assess their performance and make necessary improvements. Common evaluation metrics include accuracy, precision, recall, F1 score, and more, depending on the problem type (classification, regression, etc.).

Model Testing:

After successful validation, the models are tested on the separate testing dataset to assess their generalization ability and ensure they perform well on unseen data.

Model Deployment:

Once a satisfactory model is obtained, it can be deployed into a production environment where it can make predictions on new, real-world data.

Monitoring and Maintenance:

Continuous monitoring of the deployed model is crucial to ensure that it performs as expected over time. This includes tracking data drift and model drift.

Maintenance involves updating the model and retraining it as new data becomes available or as the model's performance degrades.

Feedback Loop:A feedback loop is an integral part of a machine learning workflow. It allows for the continuous improvement of models based on feedback from real-world usage.

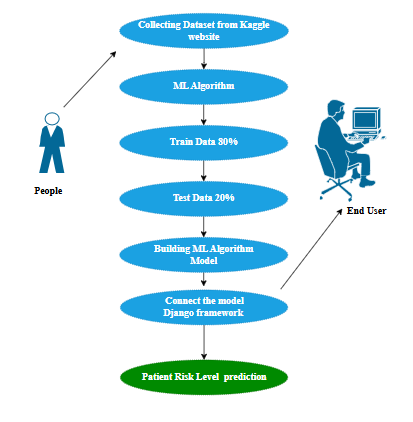
Documentation and Reporting:

Throughout the workflow, documentation is essential for keeping track of the entire process, including data sources, pre-processing steps, model configurations, and results.

Regular reporting of model performance and updates may be necessary to inform stakeholders and decision-makers.

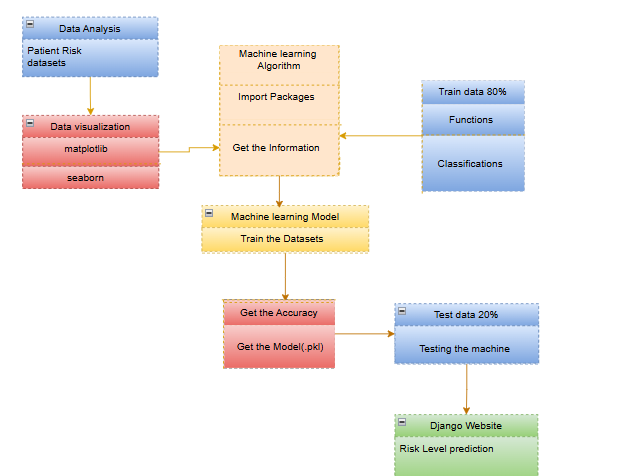
A machine learning workflow diagram provides a visual representation of these steps, helping teams and stakeholders understand the process and dependencies. It serves as a guide for the entire ML project, from data collection to model deployment and maintenance, ensuring a systematic and organized approach to building and managing machine learning systems.

**17.Use Case Diagram:**

****

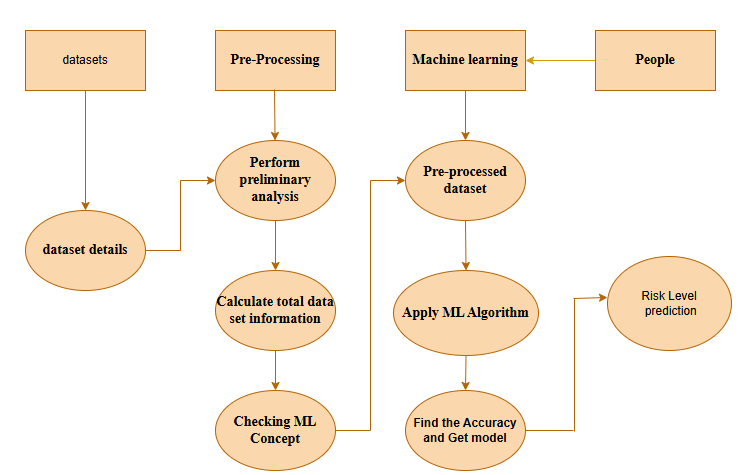
Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analysed the functionalities are captured in use cases. So, it can say that uses cases are nothing but the system functionalities written in an organized manner.

**18. Class Diagram**:



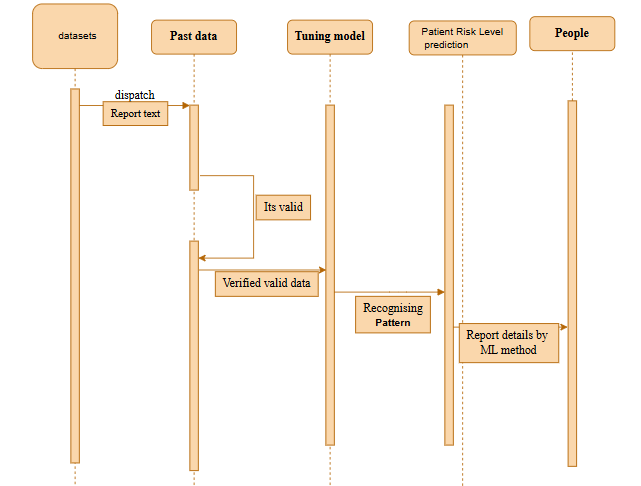
Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of class diagrams represent the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element and their relationships should be identified in advance Responsibility (attributes and methods) of each class should be clearly identified for each class minimum number of properties should be specified and because, unnecessary properties will make the diagram complicated. Use notes whenever required to describe some aspect of the diagram and at the end of the drawing it should be understandable to the developer/coder. Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

**19. Activity Diagram**:

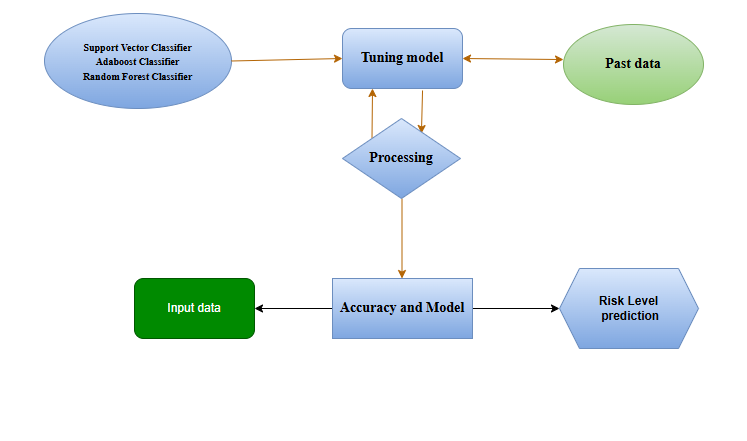


Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is some time considered as the flow chart. Although the diagrams looks like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single.

**20. Sequence Diagram**:



Sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modelling, which focuses on identifying the behaviour within your system. Other dynamic modelling techniques include [activity diagramming](http://agilemodeling.com/artifacts/activityDiagram.htm), [communication diagramming](http://agilemodeling.com/artifacts/communicationDiagram.htm), [timing diagramming](http://agilemodeling.com/artifacts/timingDiagram.htm), and [interaction overview diagramming](http://agilemodeling.com/artifacts/interactionOverviewDiagram.htm). Sequence diagrams, along with [class diagrams](http://agilemodeling.com/artifacts/classDiagram.htm) and [physical data models](http://agiledata.org/essays/dataModeling101.html) are in my opinion the most important design-level models for modern business application development.

**21. Entity Relationship Diagram (ERD): **

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationship among people, objects, places, concepts or events within that system. An ERD is a [data modeling](https://searchdatamanagement.techtarget.com/definition/data-modeling) technique that can help define business processes and be used as the foundation for a [relational database](https://searchdatamanagement.techtarget.com/definition/relational-database). Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After a relational database is rolled out, an ERD can still serve as a referral point, should any debugging or business process re-engineering be needed later.

**22. MODULE DESCRIPTION:**

**22.1 Data Pre-processing:**

Validation techniques in machine learning are used to get the error rate of the Machine Learning (ML) model, which can be considered as close to the true error rate of the dataset. If the data volume is large enough to be representative of the population, you may not need the validation techniques. However, in real-world scenarios, to work with samples of data that may not be a true representative of the population of given dataset. To finding the missing value, duplicate value and description of data type whether it is float variable or integer. The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyper parameters.

The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set is used to evaluate a given model, but this is for frequent evaluation. It as machine learning engineers use this data to fine-tune the model hyper parameters. Data collection, data analysis, and the process of addressing data content, quality, and structure can add up to a time-consuming to-do list. During the process of data identification, it helps to understand your data and its properties; this knowledge will help you choose which algorithm to use to build your model.

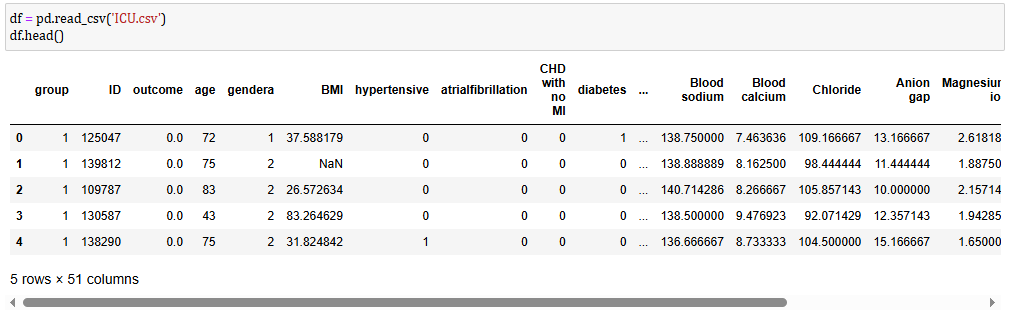
A number of different **data cleaning** tasks using Python’s [Pandas library](https://pandas.pydata.org/) and specifically, it focus on probably the biggest data cleaning task, **missing values** and it able to **more**[**quickly clean data**](https://www.dataoptimal.com/data-cleaning-with-python-2018/). It wants to **spend less time cleaning data**, and more time exploring and modeling.

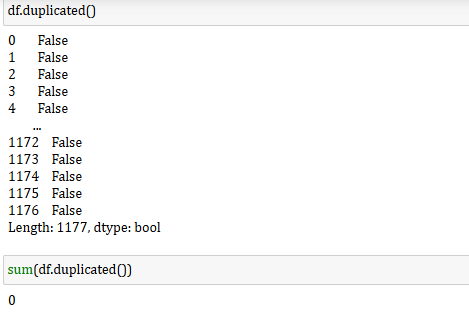
Some of these sources are just simple random mistakes. Other times, there can be a deeper reason why data is missing. It’s important to understand these [different types of missing data](https://en.wikipedia.org/wiki/Missing_data) from a statistics point of view. The type of missing data will influence how to deal with filling in the missing values and to detect missing values, and do some basic imputation and detailed statistical approach for [dealing with missing data](https://github.com/matthewbrems/ODSC-missing-data-may-18/blob/master/Analysis%20with%20Missing%20Data.pdf). Before, joint into code, it’s important to understand the sources of missing data. Here are some typical reasons why data is missing:

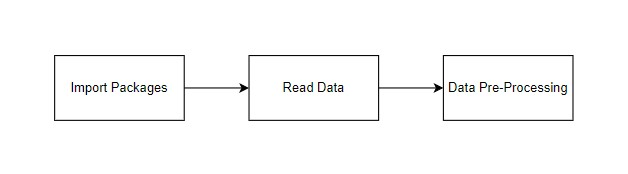
* User forgot to fill in a field.
* Data was lost while transferring manually from a legacy database.
* There was a programming error.
* Users chose not to fill out a field tied to their beliefs about how the results would be used or interpreted.

Variable identification with Uni-variate, Bi-variate and Multi-variate analysis:

* import libraries for access and functional purpose and read the given dataset
* General Properties of Analyzing the given dataset
* Display the given dataset in the form of data frame
* show columns
* shape of the data frame
* To describe the data frame
* Checking data type and information about dataset
* Checking for duplicate data
* Checking Missing values of data frame
* Checking unique values of data frame
* Checking count values of data frame
* Rename and drop the given data frame
* To specify the type of values
* To create extra columns





MODULE DIAGRAM

GIVEN INPUT EXPECTED OUTPUT

input : data

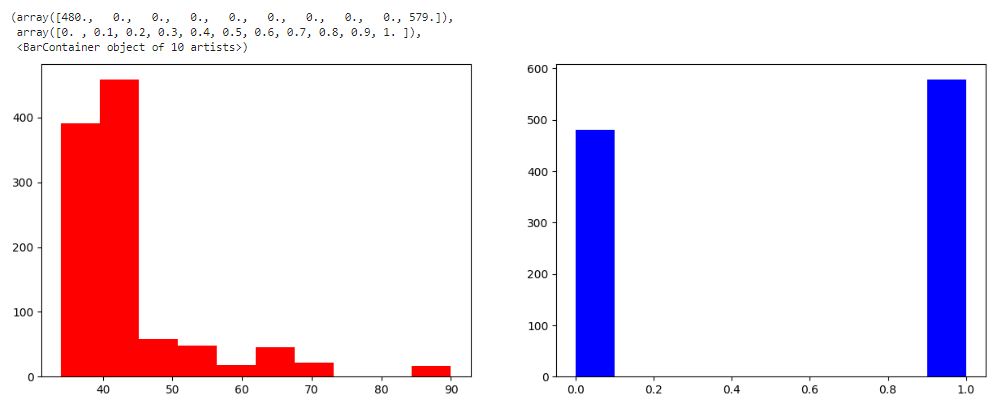
output : removing noisy data

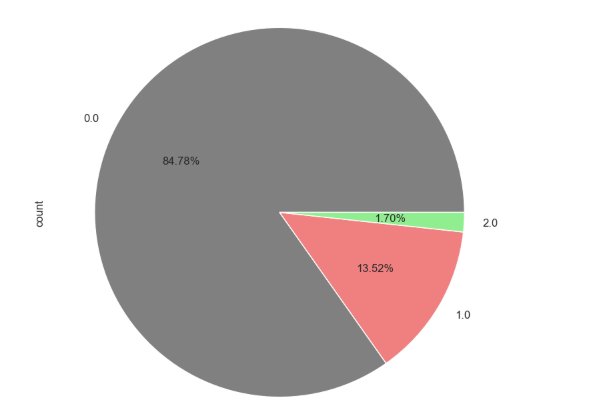
**22.2 Data visualization:**

Data visualization is an important skill in applied statistics and machine learning. Statistics does indeed focus on quantitative descriptions and estimations of data. Data visualization provides an important suite of tools for gaining a qualitative understanding. This can be helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupt data, outliers, and much more. With a little domain knowledge, data visualizations can be used to express and demonstrate key relationships in plots and charts that are more visceral and stakeholders than measures of association or significance. Data visualization and exploratory data analysis are whole fields themselves and it will recommend a deeper dive into some the books mentioned at the end.

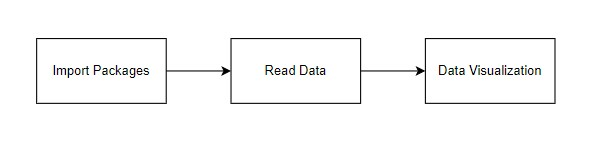
Sometimes data does not make sense until it can look at in a visual form, such as with charts and plots. Being able to quickly visualize of data samples and others is an important skill both in applied statistics and in applied machine learning. It will discover the many types of plots that you will need to know when visualizing data in Python and how to use them to better understand your own data.

* How to chart time series data with line plots and categorical quantities with bar charts.
* How to summarize data distributions with histograms and box plots.

****

****

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

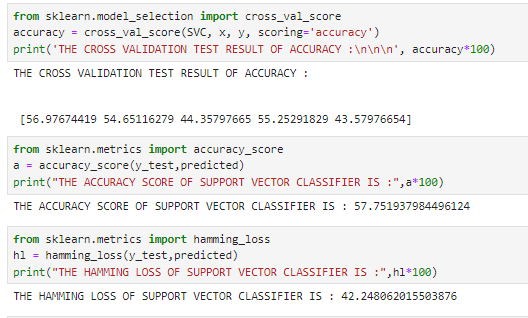
output : visualized data

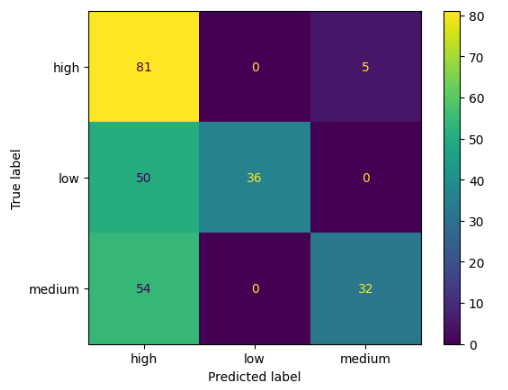
**Support Vector Machine or SVM:**

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

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 as 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





MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

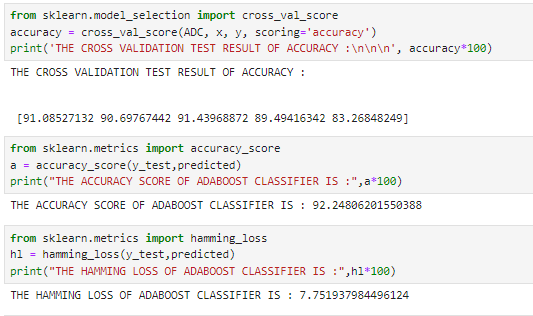
output : getting accuracy

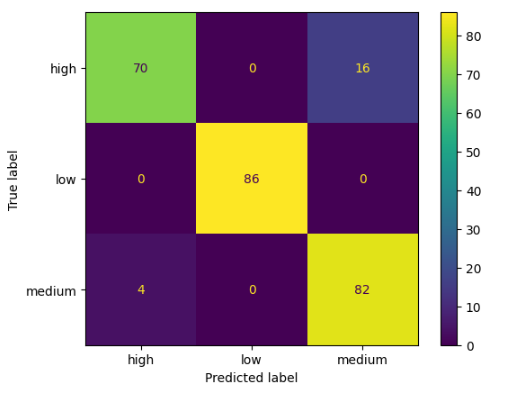
**Adaboost Classifier:**

An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

AdaBoost can be used to boost the performance of any machine learning algorithm. It is best used with weak learners. These are models that achieve accuracy just above random chance on a classification problem. The most suited and therefore most common algorithm used with AdaBoost are decision trees with one level. How does the AdaBoost algorithm work explain?

It works on the principle of learners growing sequentially. Except for the first, each subsequent learner is grown from previously grown learners. In simple words, weak learners are converted into strong ones. The AdaBoost algorithm works on the same principle as boosting with a slight difference.





MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

**Random Forest Classifier:**

A Random Forest classifier is a popular ensemble learning algorithm in machine learning that is primarily used for classification tasks. It is based on the idea of creating multiple decision trees during training and then combining their predictions to make more accurate and robust predictions. Random Forests are known for their versatility and ability to handle a wide range of data types and complexities.

Here's a detailed explanation of how the Random Forest classifier works:

Ensemble Learning: The term "ensemble" in machine learning refers to the practice of combining the predictions of multiple models to improve the overall accuracy and reliability. Random Forest is an ensemble method because it combines the predictions of multiple decision trees.

Decision Trees: A decision tree is a simple yet powerful machine learning model that can be used for both classification and regression tasks. It makes decisions by recursively splitting the dataset into subsets based on the values of input features until a stopping condition is met. Each split is determined by selecting the feature that best separates the data according to a certain criterion, typically Gini impurity or information gain for classification tasks.

Randomization: The "random" aspect of Random Forests comes from two main sources of randomness:

a. Bootstrapping: During training, a random subset of the original training data is selected with replacement. This means that some data points may appear multiple times in a subset, while others may not appear at all. This process is known as bootstrapping, and it helps create diverse training sets for each tree.

b. Feature Randomization: When building each decision tree, a random subset of features (columns) is considered at each split point. This ensures that the individual trees have different views of the data and prevents any single feature from dominating the decision-making process.

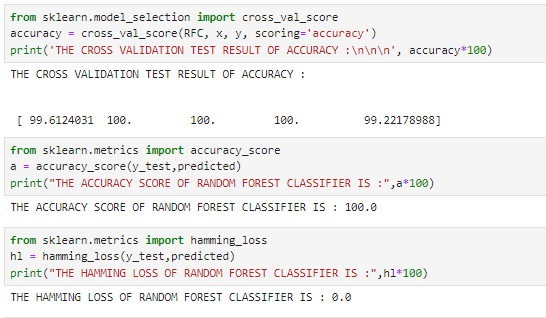
Training Multiple Decision Trees: Random Forest trains a predefined number of decision trees (an ensemble). Each tree is built independently using a different bootstrap sample and feature subset.

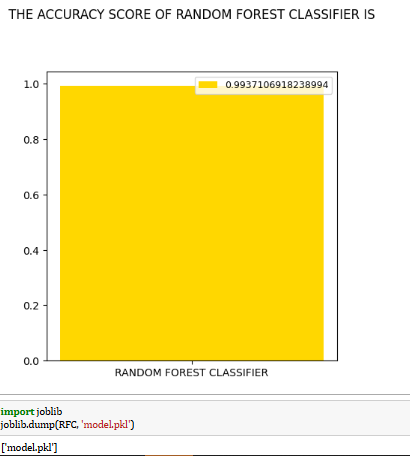
Voting or Averaging: For classification tasks, each tree in the forest makes a prediction. The most common strategy for combining these predictions is majority voting: the class that receives the most votes from the individual trees is considered the final prediction. For regression tasks, the individual tree predictions are typically averaged to produce the final prediction.

Robustness and Generalization: Random Forests are known for their robustness against over fitting, which is a common issue in decision trees. By aggregating the predictions of multiple trees and introducing randomness into the model-building process, Random Forests reduce the risk of over fitting and provide better generalization to unseen data.

Tuning Hyper parameters: There are some important hyper parameters to consider when using Random Forests, such as the number of trees in the forest, the maximum depth of each tree, and the size of the feature subsets. Tuning these hyper parameters can have a significant impact on the performance of the model.

Random Forests have many advantages, including their ability to handle high-dimensional data, handle missing values, and provide feature importance’s. They are widely used in various applications, including image classification, text classification, and medical diagnosis, among others, due to their robustness and high predictive accuracy.





MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

**23. Deployment**

**Deploying the model in Django Framework and predicting output**

In this module the trained deep learning model is converted into hierarchical data format file (.h5 file) which is then deployed in our django framework for providing better user interface and predicting the output whether the given image is CKD / Not CKD.

**Deployment:**

**Django (Web FrameWork) :**

Django is a micro web framework written in Python.It is classified as a micro-framework because it does not require particular tools or libraries.It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

However, Django supports extensions that can add application features as if they were implemented in Django itself.

Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

Django was created by [Armin Ronacher](https://en.wikipedia.org/wiki/Armin_Ronacher) of Pocoo, an international group of Python enthusiasts formed in 2004. According to Ronacher, the idea was originally an [April Fool’s](https://en.wikipedia.org/wiki/April_Fool's) joke that was popular enough to make into a serious application. The name is a play on the earlier [Bottle](https://en.wikipedia.org/wiki/Bottle_(web_framework)) framework.

When Ronacher and Georg Brand created a bulletin board system written in Python, the Pocoo projects Werkzeug and [Jinja](https://en.wikipedia.org/wiki/Jinja_(template_engine)) were developed.

In April 2016, the Pocoo team was disbanded and development of Django and related libraries passed to the newly formed Pallets project.

Django has become popular among Python enthusiasts. As of October 2020, it has second most stars on [GitHub](https://en.wikipedia.org/wiki/GitHub) among Python web-development frameworks, only slightly behind Django, and was voted the most popular web framework in the Python Developers Survey 2018.

The micro-framework Django is part of the Pallets Projects, and based on several others of them.

Django **is** based on Werkzeug, [Jinja2](http://quintagroup.com/cms/python/jinja2) and inspired by Sinatra Ruby framework, available under BSD licence. It was developed at pocoo by Armin Ronacher. Although Django is rather young compared to most [Python](https://quintagroup.com/services/python) frameworks, it holds a great promise and has already gained popularity among Python web developers. Let’s take a closer look into Django, so-called “micro” framework for Python.

**FEATURES:**

Django was designed to be **easy to use and extend**.  The idea behind Django is to build a solid foundation for web applications of different complexity. From then on you are free to**plug in any extensions** you think you need. Also you are free to build your own modules. Django is great for all kinds of projects.  It's especially good for prototyping. Django depends on two external libraries: the Jinja2 template engine and the Werkzeug WSGI toolkit.

Still the question remains why use Django as your web application framework if we have immensely powerful [Django](https://quintagroup.com/services/python/django), [Pyramid,](https://quintagroup.com/cms/python/pyramid) and don’t forget web mega-framework [Turbo-gears](https://quintagroup.com/cms/python/turbogears)? Those are supreme[Python web frameworks](https://quintagroup.com/services/python/python-web-development.png) BUT out-of-the-box Django is pretty impressive too with it’s:

* Built-In Development server and Fast debugger
* integrated support for unit testing
* RESTful request dispatching
* Uses [Jinja2](https://quintagroup.com/cms/python/jinja2) Templating
* support for secure cookies
* Unicode based
* Extensive Documentation
* Google App Engine Compatibility
* Extensions available to enhance features desired

Plus Django gives you so much more **CONTROL** on the development stage of **your project**. It follows the principles of minimalism and let you decide  how you will build your application.

* Django has a lightweight and modular design, so it easy to transform it to the web framework you need with a few extensions without weighing it down
* ORM-agnostic: you can plug in your favourite ORM e.g. [SQLAlchemy](https://quintagroup.com/cms/python/sqlalchemy).
* Basic foundation API is nicely shaped and coherent.
* Django documentation is comprehensive, full of examples and well structured. You can even try out some sample application to really get a feel of Django.
* It is super easy to deploy Django in production (Django is 100%WSGI 1.0 compliant”)
* HTTP request handling functionality
* High Flexibility

The configuration is even more flexible than that of Django, giving you plenty of solution for every production need.

To sum up, Django is one of the most polished and feature-rich micro frameworks, available. Still young, Django has a thriving community, first-class extensions, and an **elegant API**.  Django comes with all the benefits of fast templates, strong WSGI features, **thorough unit testability** at the web application and library level, **extensive documentation**. So next time you are starting a new project where you need some good features and a vast number of extensions, definitely check out Django.

Django is an API of Python that allows us to build up web-applications. It was developed by Armin Ronacher. Django's framework is more explicit than Django framework and is also easier to learn because it has less base code to implement a simple web-Application

Django is a micro web framework written in Python. It is classified as a micro-framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

Overview of Python Django Framework Web apps are developed to generate content based on retrieved data that changes based on a user’s interaction with the site. The server is responsible for querying, retrieving, and updating data. This makes web applications to be slower and more complicated to deploy than static websites for simple applications.

Django is an excellent web development framework for REST API creation. It is built on top of Python which makes it powerful to use all the python features.

Django is used for the backend, but it makes use of a templating language called Jinja2 which is used to create HTML, XML or other markup formats that are returned to the user via an HTTP request.

Django is considered to be more popular because it provides many out of box features and reduces time to build complex applications. Django is a good start if you are getting into web development. Django is a simple, un-opinionated framework; it doesn't decide what your application should look like developers do.

Django is a web framework. This means Django provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, and a wiki or go as big as a web-based calendar application or a commercial website.

**Advantages of Django:**

* Higher compatibility with latest technologies.
* Technical experimentation.
* Easier to use for simple cases.
* Codebase size is relatively smaller.
* High scalability for simple applications.
* Easy to build a quick prototype.
* Routing URL is easy.
* Easy to develop and maintain applications.

Framework Django is a web framework from Python language. Django provides a library and a collection of codes that can be used to build websites, without the need to do everything from scratch. But Framework Django still doesn't use the Model View Controller (MVC) method.

Django-RESTful is an extension for Django that provides additional support for building REST APIs. You will never be disappointed with the time it takes to develop an API. Django-Restful is a lightweight abstraction that works with the existing ORM/libraries. Django-RESTful encourages best practices with minimal setup.

Django Restful is an extension for Django that adds support for building REST APIs in Python using Django as the back-end. It encourages best practices and is very easy to set up. Django restful is very easy to pick up if you're already familiar with Django.

Django is a web framework for Python, meaning that it provides functionality for building web applications, including managing HTTP requests and rendering templates and also we can add to this application to create our API.

**Start Using an API**

1. Most APIs require an API key. ...
2. The easiest way to start using an API is by finding an HTTP client online, like REST-Client, Postman, or Paw.
3. The next best way to pull data from an API is by building a URL from existing API documentation.

The Django object implements a WSGI application and acts as the central object. It is passed the name of the module or package of the application. Once it is created it will act as a central registry for the view functions, the URL rules, template configuration and much more.

The name of the package is used to resolve resources from inside the package or the folder the module is contained in depending on if the package parameter resolves to an actual python package (a folder with an \_\_init\_\_.py file inside) or a standard module (just a .py file).

For more information about resource loading, see [open resource()](https://flask.palletsprojects.com/en/2.0.x/api/#flask.Flask.open_resource).

Usually you create a [Django](https://flask.palletsprojects.com/en/2.0.x/api/#flask.Flask) instance in your main module or in the \_\_init\_\_.py file of your package.

**Parameters**

* **rule** ([str](https://docs.python.org/3/library/stdtypes.html#str)) – The URL rule string.
* **endpoint** (Optional[[str](https://docs.python.org/3/library/stdtypes.html#str)]) – The endpoint name to associate with the rule and view function. Used when routing and building URLs. Defaults to view\_func.\_\_name\_\_.
* **view\_func** (Optional[Callable]) – The view function to associate with the endpoint name.
* **provide\_automatic\_options** (Optional[bool]) – Add the OPTIONS method and respond to OPTIONS requests automatically.
* **options** (Any) – Extra options passed to the [Rule](https://werkzeug.palletsprojects.com/en/2.0.x/routing/#werkzeug.routing.Rule) object.

Return type -- [None](https://docs.python.org/3/library/constants.html#None)

After\_Request(f)

Register a function to run after each request to this object.

The function is called with the response object, and must return a response object. This allows the functions to modify or replace the response before it is sent.

If a function raises an exception, any remaining after request functions will not be called. Therefore, this should not be used for actions that must execute, such as to close resources. Use [teardown\_request()](https://flask.palletsprojects.com/en/2.0.x/api/#flask.Flask.teardown_request) for that.

**Parameters:**

**f** (Callable[[[Response](https://flask.palletsprojects.com/en/2.0.x/api/#flask.Response)], [Response](https://flask.palletsprojects.com/en/2.0.x/api/#flask.Response)])

Return type

Callable[[[Response](https://flask.palletsprojects.com/en/2.0.x/api/#flask.Response)], [Response](https://flask.palletsprojects.com/en/2.0.x/api/#flask.Response)]

after\_request\_funcs: t.Dict[AppOrBlueprintKey,

t.List[AfterRequestCallable]]

A data structure of functions to call at the end of each request, in the format {scope: [functions]}. The scope  key is the name of a blueprint the functions are active for, or None for all requests.

To register a function, use the [after\_request()](https://flask.palletsprojects.com/en/2.0.x/api/#flask.Flask.after_request) decorator.

This data structure is internal. It should not be modified directly and its format may change at any time.

app\_context()

Create an [AppContext](https://flask.palletsprojects.com/en/2.0.x/api/#flask.ctx.AppContext). Use as a with block to push the context, which will make [current\_app](https://flask.palletsprojects.com/en/2.0.x/api/#flask.current_app) point at this application.

An application context is automatically pushed by [RequestContext.push()](https://flask.palletsprojects.com/en/2.0.x/api/#flask.ctx.RequestContext.push) when handling a request, and when running a CLI command. Use this to manually create a context outside of these situations.

With app.app\_context():

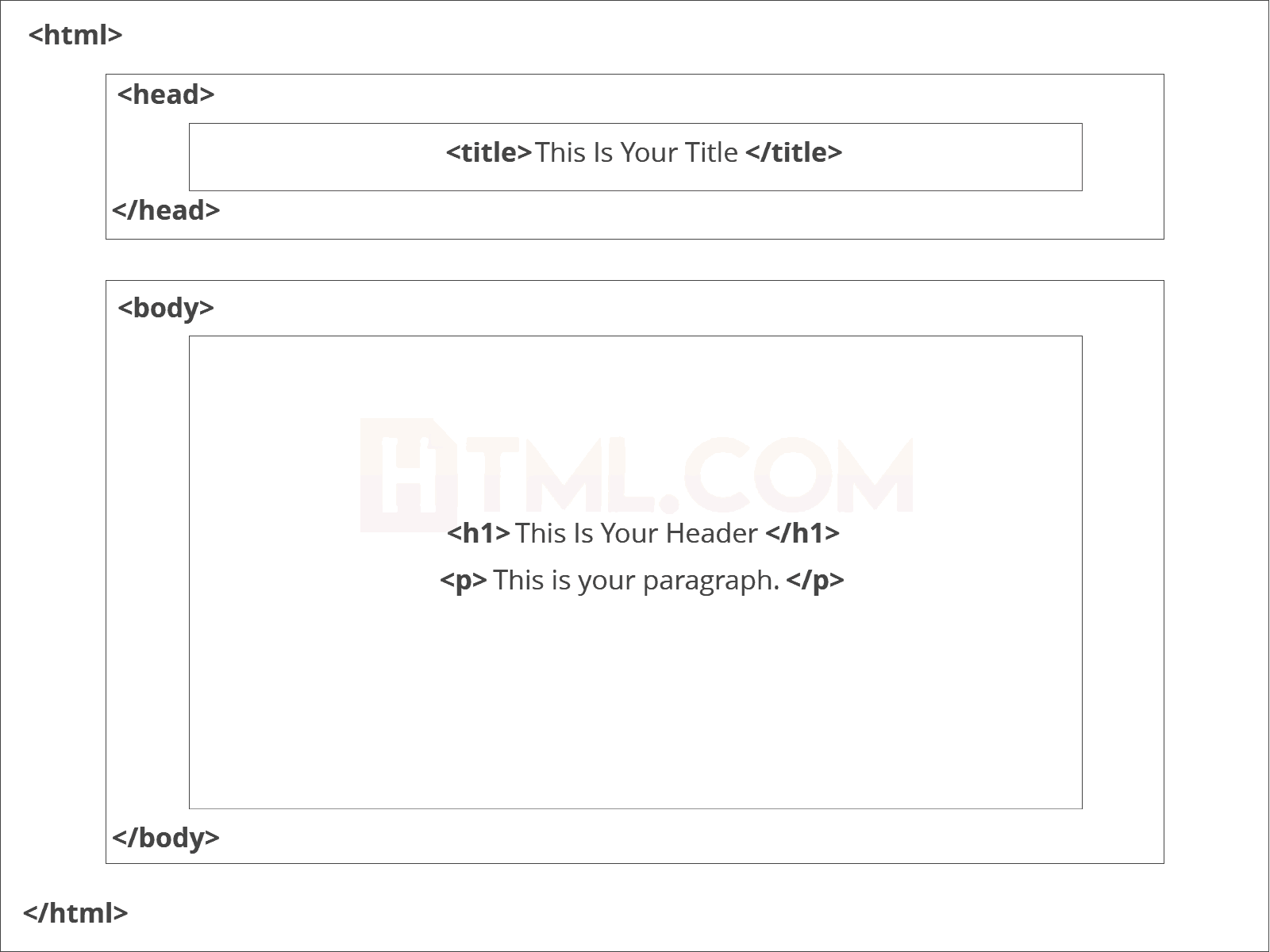
Init\_db()

**24. HTML Introduction**

HTML stands for Hyper Text Markup Language. It is used to design web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. A markup language is used to define the text document within tag which defines the structure of web pages. This language is used to annotate (make notes for the computer) text so that a machine can understand it and manipulate text accordingly. Most markup languages (e.g. HTML) are human-readable. The language uses tags to define what manipulation has to be done on the text.

#### Basic Construction of an HTML Page

These tags should be placed underneath each other **at the top of every HTML page** that you create.



<!DOCTYPE html> — This tag**specifies the language** you will write on the page. In this case, the language is HTML 5.

<html> — This tag signals that from here on we are going to write in HTML code.

<head>[— This is where all the](https://html.com/tags/head/)**metadata for the page** goes — stuff mostly meant for search engines and other computer programs.

<body>[— This is where the](https://html.com/tags/body/)**content of the page** goes.

#### Further Tags

Inside the <head> tag, there is one tag that is always included: <title>, but there are others that are just as important:

<title>

This is where we**insert the page name** as it will appear at the top of the browser window or tab.

<meta>

This is where information about the document is stored: character encoding, name (page context), description.

**Head Tag**  
<head>

<title>My First Webpage</title>

<meta charset="UTF-8">

<meta name="description" content="This field contains information about your page. It is usually around two sentences long.">.

<meta name="author" content="Conor Sheils">

</header>

### Adding Content

Next, we will make<body> tag.

The HTML <body> is where we add the content which is designed for viewing by human eyes.

This includes **text, images, tables, forms**and everything else that we see on the internet each day.

#### Add HTML Headings To Web Page

In HTML, [headings](https://html.com/tags/heading/) are written in the following elements:

* <h1>
* <h2>
* <h3>
* <h4>
* <h5>
* <h6>

As you might have guessed <h1> and <h2> should be used for the most important titles, while the remaining tags should be used for sub-headings and less important text.

**Search engine bots use this order**when deciphering which information is most important on a page.

##### Creating Your Heading

Let’s try it out. On a new line in the HTML editor, type:

<h1> Welcome To My Page </h1>

And hit save. We will save this file as “index.html” in a new folder called “my webpage.”

**Add Text In HTML**

Adding text to our HTML page is simple using an element opened with the tag <p> which **creates a new paragraph**. We place all of our regular text inside the element <p>.

When we write text in HTML, we also have a number of other elements we can use **to control the text or make it appear in a certain way.**

#### Add Links In HTML

As you may have noticed, the internet is made up of lots of [links](https://html.com/anchors-links/).

Almost everything you click on while surfing the web is a link **takes you to another page** within the website you are visiting or to an external site.

Links are included in an attribute opened by the [**<a>**](https://html.com/tags/a/) tag. This element is the first that we’ve met which uses an attribute and so it**looks different to previously mentioned tags.**

<a href=[http://www.google.com](http://www.google.com/)>Google</a>

**Image Tag**

In today’s modern digital world, [images](https://html.com/blog/100-legal-sources-free-stock-images/) are everything. The [**<**img**>**](https://html.com/tags/img/) tag has everything you need to display images on your site. Much like the <a> anchor element, <img> also contains an attribute.

The attribute features information for your computer regarding the source, height, width and alt text of the image

<img src=”yourimage.jpg” alt=”Describe the image” height=“X” width=“X”>

**25. CSS**

CSS stands for Cascading Style Sheets. It is the language for describing the presentation of Web pages, including colours, layout, and fonts, thus making our web pages presentable to the users.CSS is designed to make style sheets for the web. It is independent of HTML and can be used with any XML-based markup language. Now let’s try to break the acronym:

* Cascading: Falling of Styles
* Style: Adding designs/Styling our HTML tags
* Sheets: Writing our style in different documents

## **CSS Syntax**

Selector {

Property 1 : value;

Property 2 : value;

Property 3 : value;

}

For example:

h1

{

Color: red;

Text-align: center;

}

#unique

{

color: green;

}

* Selector: selects the element you want to target
* Always remains the same whether we apply internal or external styling
* There are few basic selectors like tags, id’s, and classes
* All forms this key-value pair
* Keys: properties(attributes) like color, font-size, background, width, height,etc
* Value: values associated with these properties

## **CSS Comment**

* Comments don’t render on the browser
* Helps to understand our code better and makes it readable.
* Helps to debug our code
* Two ways to  comment:
  + Single line

## **CSS How-To**

* There are 3 ways to write CSS in our HTML file.
  + Inline CSS
  + Internal CSS
  + External CSS
* Priority order
  + Inline > Internal > External

**Inline CSS**

* Before CSS this was the only way to apply styles
* Not an efficient way to write as it has a lot of redundancy
* Self-contained
* Uniquely applied on each element
* The idea of separation of concerns was lost
* Example:

<h3 style = “color:red”> Have a great day </h3>

<p style = “color:green”> I did this, I did that </p>

**Internal CSS**

* With the help of style tag, we can apply styles within the HTML file
* Redundancy is removed
* But the idea of separation of concerns still lost
* Uniquely applied on a single document
* Example:

<style>

H1{

Color:red;

}

</style>

<h3> Have a great day </h3>

**External CSS**

* With the help of <link> tag in the head tag, we can apply styles
* Reference is added
* File saved with .css extension
* Redundancy is removed
* The idea of separation of concerns is maintained
* Uniquely applied to each document
* Example:

<head>

<link rel= “stylesheet” type= “text/css” href= “name of the CSS file”>

</head>

h1{

color:red; //.css file

}

## **CSS Selectors**

* The selector is used to target elements and apply CSS
* Three simple selectors
  + Element Selector
  + Id Selector
  + Class Selector
* Priority of Selectors

## **CSS Colors**

* There are different colouring schemes in CSS
* **RGB**-This starts with RGB and takes 3 parameter
* **HEX**-Hex code starts with # and comprises of 6 numbers which are further divided into 3 sets
* **RGBA**-This starts with RGB and takes 4 parameter

**CSS Background**

* There are different ways by which CSS can have an effect on HTML elements
* Few of them are as follows:
  + Color – used to set the color of the background
  + Repeat – used to determine if the image has to repeat or not and if it is repeating then how it should do that
  + Image – used to set an image as the background
  + Position – used to determine the position of the image
  + Attachment – It basically helps in controlling the mechanism of scrolling.

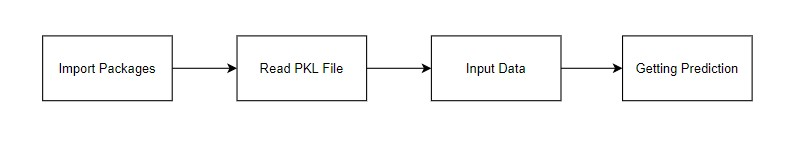
## **CSS BoxModel**

* Every element in CSS can be represented using the BOX model
* It allows us to add a border and define space between the content
* It helps the developer to develop and manipulate the elements
* It consists of 4 edges
  + Content edge – It comprises of the actual content
  + Padding edge – It lies in between content and border edge
  + Border edge – Padding is followed by the border edge
  + Margin edge – It is an outside border and controls the margin of the element

**Deploying the model predicting output**

In this module the trained machine learning model is converted into pickle data format file (.pkl file) which is then deployed for providing better user interface and predicting the output of intrusion.

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data values

output : predicting output

**CODING:**

**MODULE 1:**

**## DATA PREPROCESSING AND DATA CLEANING**

**import pandas as pd**

**import numpy as np**

**import warnings**

**warnings.filterwarnings('ignore')**

**df = pd.read\_csv('DATA.csv')**

**df.head()**

**df.tail()**

**df.shape**

**df.size**

**df.columns**

**df.isnull()**

**df = df.dropna()**

**df['Grade'].unique()**

**df.describe()**

**from sklearn.preprocessing import LabelEncoder**

**le = LabelEncoder()**

**var = ['Grade']**

**for i in var:**

**df[i] = le.fit\_transform(df[i]).astype(int)**

**df.corr()**

**df.info()**

**pd.crosstab(df["Turbidity"], df["Fat "])**

**df.groupby(["Odor","Temprature"]).groups**

**df["Grade"].value\_counts()**

**pd.Categorical(df["Grade"]).describe()**

**df.duplicated()**

**sum(df.duplicated())**

**df = df.drop\_duplicates()**

**sum(df.duplicated())**

**MODULE 2:**

**## DATA VISUALIZATION AND DATA ANALYSIS**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**df = pd.read\_csv('DATA.csv')**

**df.head()**

**df.columns**

**from sklearn.preprocessing import LabelEncoder**

**le = LabelEncoder()**

**var = ['Grade']**

**for i in var:**

**df[i] = le.fit\_transform(df[i]).astype(int)**

**plt.figure(figsize=(12,7))**

**sns.countplot(x='pH',data=df)**

**plt.figure(figsize=(15,5))**

**plt.subplot(1,2,1)**

**plt.hist(df['Temprature'],color='red')**

**plt.subplot(1,2,2)**

**plt.hist(df['Taste'],color='blue')**

**df.hist(figsize=(15,55),layout=(15,4), color='green')**

**plt.show()**

**df['Odor'].hist(figsize=(10,5),color='yellow')**

**sns.lineplot(df['Fat '], color='brown') # scatter, plot, triplot, stackplot**

**sns.violinplot(df['Turbidity'], color='purple')**

**df['Colour'].plot(kind='density')**

**sns.displot(df['Temprature'], color='purple')**

**# barplot, boxenplot, boxplot, countplot, displot, distplot, ecdfplot, histplot, kdeplot, pointplot, violinplot, stripplot**

**sns.displot(df['Temprature'], color='coral') # residplot, scatterplot**

**fig, ax = plt.subplots(figsize=(20,15))**

**sns.heatmap(df.corr(),annot = True, fmt='0.2%',cmap = 'autumn',ax=ax)**

**def plot(df, variable):**

**dataframe\_pie = df[variable].value\_counts()**

**ax = dataframe\_pie.plot.pie(figsize=(9,9), autopct='%1.2f%%', fontsize = 10)**

**ax.set\_title(variable + ' \n', fontsize = 10)**

**return np.round(dataframe\_pie/df.shape[0]\*100,2)**

**plot(df, 'Grade')**

**MODULE 3:**

**# SUPPORT VECTOR CLASSIFIER ALGORITHEM**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**import warnings**

**warnings.filterwarnings('ignore')**

**df = pd.read\_csv('DATA.csv')**

**df.head()**

**df.columns**

**df=df.dropna()**

**df.columns**

**df.tail()**

**x1 = df.drop(labels='Grade', axis=1)**

**y1 = df.loc[:,'Grade']**

**import imblearn**

**from imblearn.over\_sampling import RandomOverSampler**

**from collections import Counter**

**ros =RandomOverSampler(random\_state=42)**

**x,y=ros.fit\_resample(x1,y1)**

**print("OUR DATASET COUNT : ", Counter(y1))**

**print("OVER SAMPLING DATA COUNT : ", Counter(y))**

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

**x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.20, random\_state=6, stratify=y)**

**print("NUMBER OF TRAIN DATASET : ", len(x\_train))**

**print("NUMBER OF TEST DATASET : ", len(x\_test))**

**print("TOTAL NUMBER OF DATASET : ", len(x\_train)+len(x\_test))**

**print("NUMBER OF TRAIN DATASET : ", len(y\_train))**

**print("NUMBER OF TEST DATASET : ", len(y\_test))**

**print("TOTAL NUMBER OF DATASET : ", len(y\_train)+len(y\_test))**

**from sklearn.svm import SVC**

**SVC = SVC()**

**SVC.fit(x\_train,y\_train)**

**predicted = SVC.predict(x\_test)**

**from sklearn.metrics import confusion\_matrix**

**cm = confusion\_matrix(y\_test,predicted)**

**print('THE CONFUSION MATRIX SCORE OF SUPPORT VECTOR CLASSIFIER:\n\n\n',cm)**

**from sklearn.model\_selection import cross\_val\_score**

**accuracy = cross\_val\_score(SVC, x, y, scoring='accuracy')**

**print('THE CROSS VALIDATION TEST RESULT OF ACCURACY :\n\n\n', accuracy\*100)**

**from sklearn.metrics import accuracy\_score**

**a = accuracy\_score(y\_test,predicted)**

**print("THE ACCURACY SCORE OF SUPPORT VECTOR CLASSIFIER IS :",a\*100)**

**from sklearn.metrics import hamming\_loss**

**hl = hamming\_loss(y\_test,predicted)**

**print("THE HAMMING LOSS OF SUPPORT VECTOR CLASSIFIER IS :",hl\*100)**

**from sklearn.metrics import classification\_report**

**cl = classification\_report(y\_test,predicted)**

**print("THE CLASSIFICATION REPORT OF SUPPORT VECTOR CLASSIFIER IS :\n\n",cl)**

**from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay**

**CMD = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=SVC.classes\_)**

**CMD.plot()**

**plt.show()**

**def graph():**

**import matplotlib.pyplot as plt**

**data=[a]**

**alg="SUPPORT VECTOR CLASSIFIER"**

**plt.figure(figsize=(5,5))**

**b=plt.bar(alg,data,color=("BLUE"))**

**plt.title("THE ACCURACY SCORE OF SUPPORT VECTOR CLASSIFIER IS\n\n\n")**

**plt.legend(b,data,fontsize=9)**

**graph()**

**import matplotlib.pyplot as plt**

**df2 = pd.DataFrame()**

**df2["y\_test"] = y\_test**

**df2["predicted"] = predicted**

**df2.reset\_index(inplace=True)**

**plt.figure(figsize=(20, 5))**

**plt.plot(df2["predicted"][:100], marker='x', linestyle='dashed', color='red')**

**plt.plot(df2["y\_test"][:100], marker='o', linestyle='dashed', color='green')**

**plt.show()**

**MODULE 4:**

**# ADABOOST CLASSIFIER ALGORITHEM**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**import warnings**

**warnings.filterwarnings('ignore')**

**df = pd.read\_csv('DATA.csv')**

**df.head()**

**df.columns**

**df=df.dropna()**

**df.columns**

**df.tail()**

**x1 = df.drop(labels='Grade', axis=1)**

**y1 = df.loc[:,'Grade']**

**import imblearn**

**from imblearn.over\_sampling import RandomOverSampler**

**from collections import Counter**

**ros =RandomOverSampler(random\_state=42)**

**x,y=ros.fit\_resample(x1,y1)**

**print("OUR DATASET COUNT : ", Counter(y1))**

**print("OVER SAMPLING DATA COUNT : ", Counter(y))**

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

**x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.20, random\_state=6, stratify=y)**

**print("NUMBER OF TRAIN DATASET : ", len(x\_train))**

**print("NUMBER OF TEST DATASET : ", len(x\_test))**

**print("TOTAL NUMBER OF DATASET : ", len(x\_train)+len(x\_test))**

**print("NUMBER OF TRAIN DATASET : ", len(y\_train))**

**print("NUMBER OF TEST DATASET : ", len(y\_test))**

**print("TOTAL NUMBER OF DATASET : ", len(y\_train)+len(y\_test))**

**from sklearn.ensemble import AdaBoostClassifier**

**ADC = AdaBoostClassifier(random\_state=42)**

**ADC.fit(x\_train,y\_train)**

**predicted = ADC.predict(x\_test)**

**from sklearn.metrics import confusion\_matrix**

**cm = confusion\_matrix(y\_test,predicted)**

**print('THE CONFUSION MATRIX SCORE OF ADABOOST CLASSIFIER :\n\n\n',cm)**

**from sklearn.model\_selection import cross\_val\_score**

**accuracy = cross\_val\_score(ADC, x, y, scoring='accuracy')**

**print('THE CROSS VALIDATION TEST RESULT OF ACCURACY :\n\n\n', accuracy\*100)**

**from sklearn.metrics import accuracy\_score**

**a = accuracy\_score(y\_test,predicted)**

**print("THE ACCURACY SCORE OF ADABOOST CLASSIFIER IS :",a\*100)**

**from sklearn.metrics import hamming\_loss**

**hl = hamming\_loss(y\_test,predicted)**

**print("THE HAMMING LOSS OF ADABOOST CLASSIFIER IS :",hl\*100)**

**from sklearn.metrics import classification\_report**

**cl = classification\_report(y\_test,predicted)**

**print("THE CLASSIFICATION REPORT OF ADABOOST CLASSIFIER IS :\n\n",cl)**

**from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay**

**CMD = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=ADC.classes\_)**

**CMD.plot()**

**plt.show()**

**def graph():**

**import matplotlib.pyplot as plt**

**data=[a]**

**alg="ADABOOST CLASSIFIER"**

**plt.figure(figsize=(5,5))**

**b=plt.bar(alg,data,color=("GREEN"))**

**plt.title("THE ACCURACY SCORE OF ADABOOST CLASSIFIER IS\n\n\n")**

**plt.legend(b,data,fontsize=9)**

**graph()**

**import matplotlib.pyplot as plt**

**df2 = pd.DataFrame()**

**df2["y\_test"] = y\_test**

**df2["predicted"] = predicted**

**df2.reset\_index(inplace=True)**

**plt.figure(figsize=(20, 5))**

**plt.plot(df2["predicted"][:100], marker='x', linestyle='dashed', color='red')**

**plt.plot(df2["y\_test"][:100], marker='o', linestyle='dashed', color='green')**

**plt.show()**

**MODULE 5:**

**# RANDOM FOREST CLASSIFIER ALGORITHEM**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**import warnings**

**warnings.filterwarnings('ignore')**

**df = pd.read\_csv('DATA.csv')**

**df.head()**

**df.columns**

**df=df.dropna()**

**df.columns**

**from sklearn.preprocessing import LabelEncoder**

**le = LabelEncoder()**

**var = ['Grade']**

**for i in var:**

**df[i] = le.fit\_transform(df[i]).astype(int)**

**df.tail()**

**x1 = df.drop(labels='Grade', axis=1)**

**y1 = df.loc[:,'Grade']**

**import imblearn**

**from imblearn.over\_sampling import RandomOverSampler**

**from collections import Counter**

**ros =RandomOverSampler(random\_state=42)**

**x,y=ros.fit\_resample(x1,y1)**

**print("OUR DATASET COUNT : ", Counter(y1))**

**print("OVER SAMPLING DATA COUNT : ", Counter(y))**

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

**x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.20, random\_state=6, stratify=y)**

**print("NUMBER OF TRAIN DATASET : ", len(x\_train))**

**print("NUMBER OF TEST DATASET : ", len(x\_test))**

**print("TOTAL NUMBER OF DATASET : ", len(x\_train)+len(x\_test))**

**print("NUMBER OF TRAIN DATASET : ", len(y\_train))**

**print("NUMBER OF TEST DATASET : ", len(y\_test))**

**print("TOTAL NUMBER OF DATASET : ", len(y\_train)+len(y\_test))**

**from sklearn.ensemble import RandomForestClassifier**

**RFC = RandomForestClassifier(random\_state=42)**

**RFC.fit(x\_train,y\_train)**

**predicted = RFC.predict(x\_test)**

**from sklearn.metrics import confusion\_matrix**

**cm = confusion\_matrix(y\_test,predicted)**

**print('THE CONFUSION MATRIX SCORE OF RANDOM FOREST CLASSIFIER:\n\n\n',cm)**

**from sklearn.model\_selection import cross\_val\_score**

**accuracy = cross\_val\_score(RFC, x, y, scoring='accuracy')**

**print('THE CROSS VALIDATION TEST RESULT OF ACCURACY :\n\n\n', accuracy\*100)**

**from sklearn.metrics import accuracy\_score**

**a = accuracy\_score(y\_test,predicted)**

**print("THE ACCURACY SCORE OF RANDOM FOREST CLASSIFIER IS :",a\*100)**

**from sklearn.metrics import hamming\_loss**

**hl = hamming\_loss(y\_test,predicted)**

**print("THE HAMMING LOSS OF RANDOM FOREST CLASSIFIER IS :",hl\*100)**

**from sklearn.metrics import classification\_report**

**cl = classification\_report(y\_test,predicted)**

**print("THE CLASSIFICATION REPORT OF RANDOM FOREST CLASSIFIER IS :\n\n",cl)**

**from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay**

**CMD = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=RFC.classes\_)**

**CMD.plot()**

**plt.show()**

**def graph():**

**import matplotlib.pyplot as plt**

**data=[a]**

**alg="RANDOM FOREST CLASSIFIER"**

**plt.figure(figsize=(5,5))**

**b=plt.bar(alg,data,color=("gold"))**

**plt.title("THE ACCURACY SCORE OF RANDOM FOREST CLASSIFIER IS\n\n\n")**

**plt.legend(b,data,fontsize=9)**

**graph()**

**import matplotlib.pyplot as plt**

**df2 = pd.DataFrame()**

**df2["y\_test"] = y\_test**

**df2["predicted"] = predicted**

**df2.reset\_index(inplace=True)**

**plt.figure(figsize=(20, 5))**

**plt.plot(df2["predicted"][:100], marker='x', linestyle='dashed', color='red')**

**plt.plot(df2["y\_test"][:100], marker='o', linestyle='dashed', color='green')**

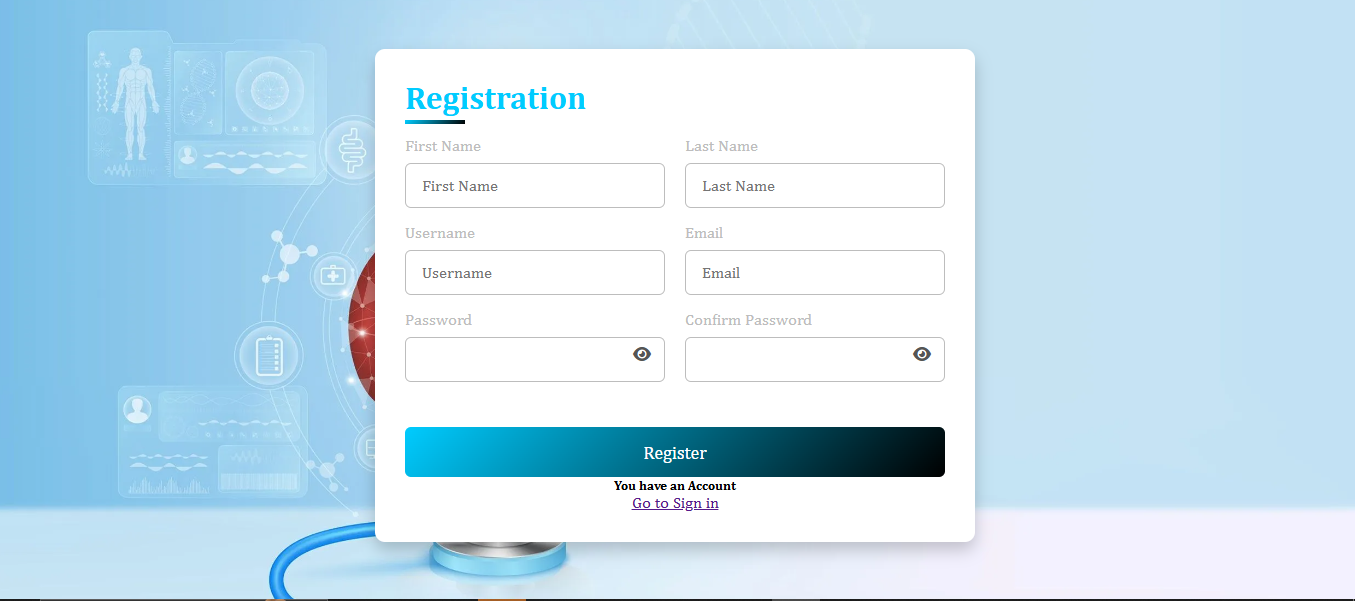
**plt.show()**

**import joblib**

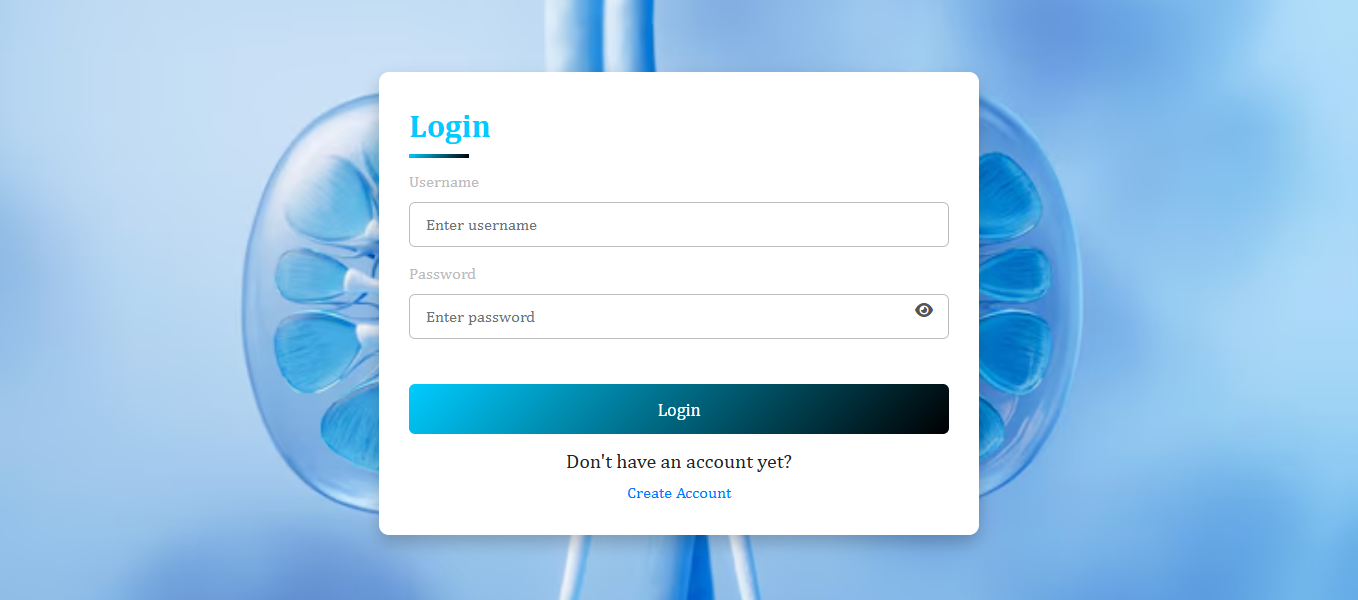
**joblib.dump(RFC, 'RFC.pkl')**

**SCREEN SHOT:**

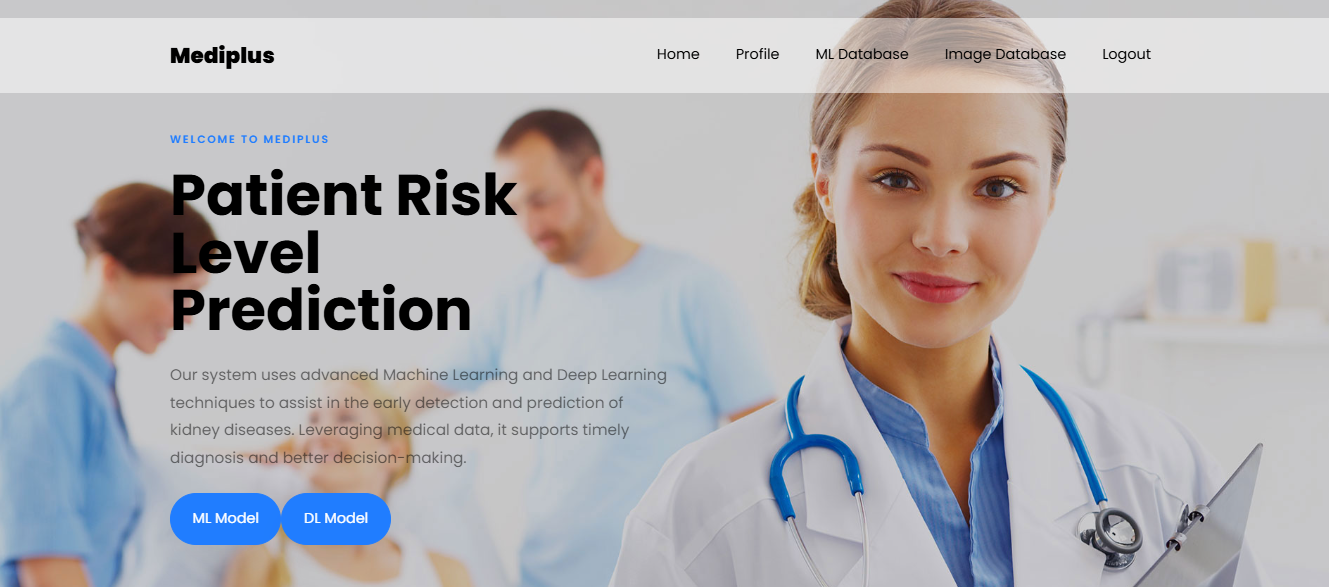
**Register Page:**

****

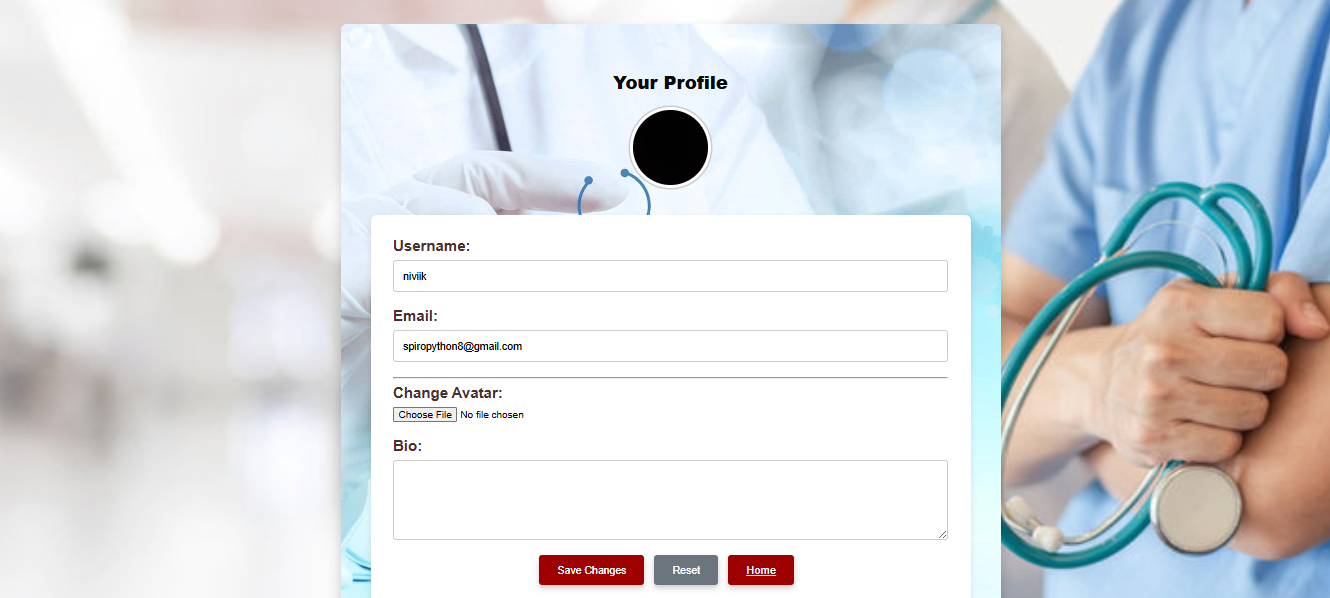
**Login Page:**

****

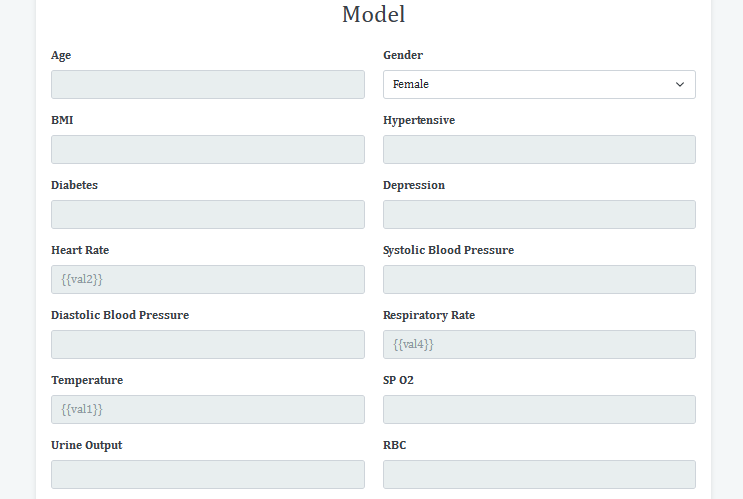
**Home Page:**

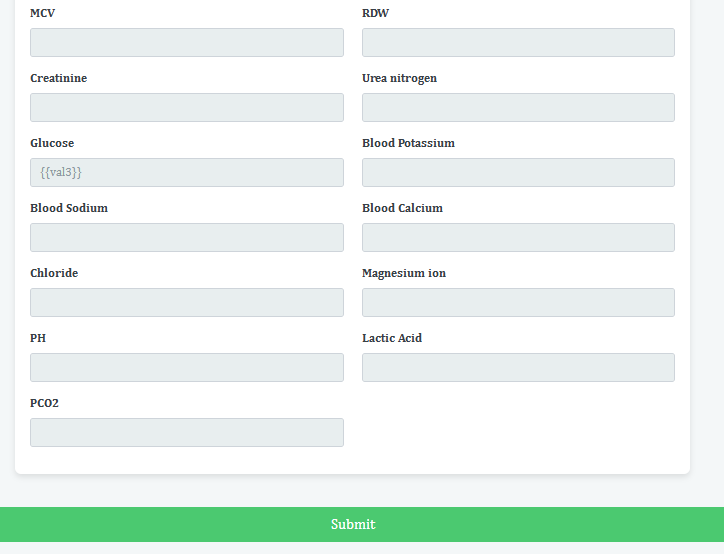
****

**Profile Page:**

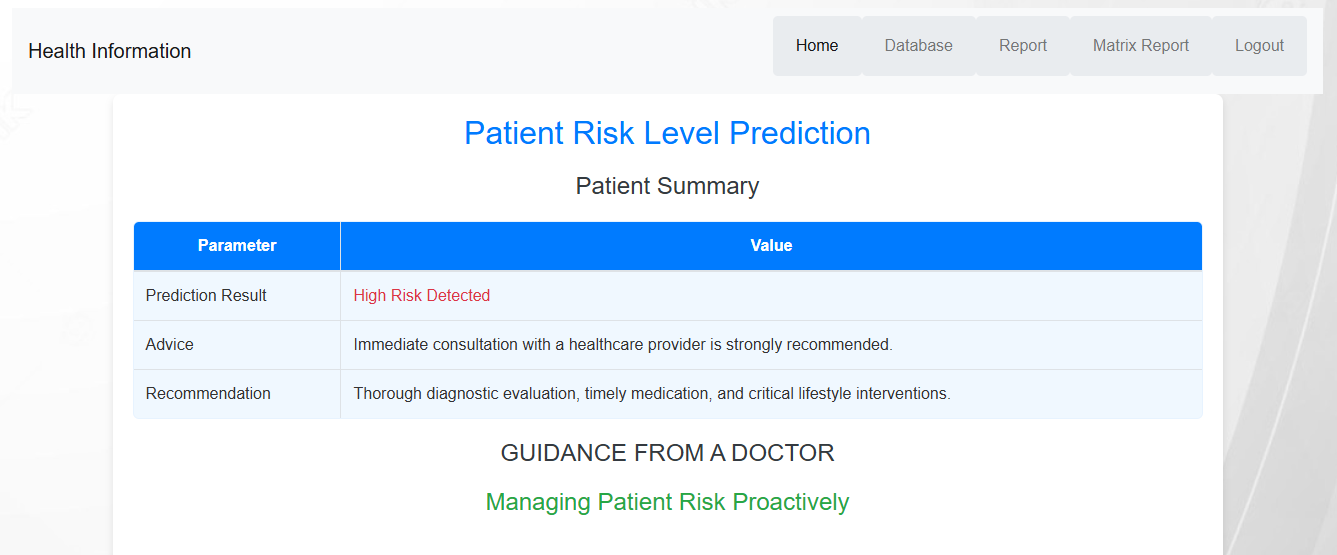
****

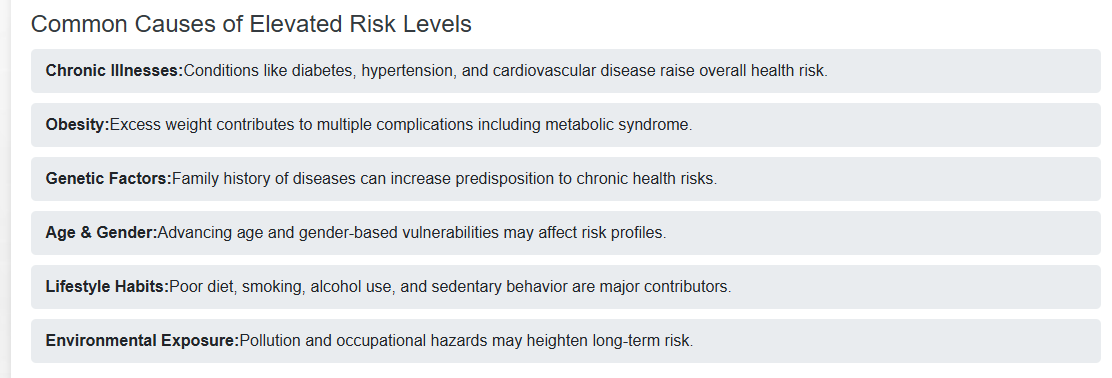
**Model Page:**

****

****

**OutPut Page:**

****

****

****

**27. Conclusion:**

In conclusion, the integration of advanced machine learning models into clinical decision support systems significantly enhances the predictive accuracy and efficiency of patient risk level assessment. By leveraging Django as the development framework, the system ensures robust backend support, scalable deployment, and seamless interaction with medical datasets. The predictive capabilities enable healthcare professionals to make informed decisions in ultimately improving patient outcomes, optimizing resource allocation, and reducing the likelihood of critical health events. This approach represents a transformative step toward intelligent, data-driven healthcare solutions.

**28. Future Work:**

* Prediction methods to connect with cloud model.
* To optimize the work to implement in IOT System.