**CHAPTER 1**

**INTRODUCTION**

**1.1 Domain Overview**

**1.1.1** **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.

**1.1.2** **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.

**1.1.3** **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, 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 around 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.

**1.1.4** **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 labeling to learn data has to be labeled 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.

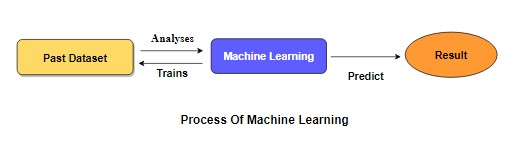


Fig 1.1.4.1 Process of Machine Learning

[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 labeled 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”.

**Preparing The Dataset:**

This dataset contains 316 records of features extracted from Kidney Stone data’s which were then classified into 2 classes

**CHAPTER 2**

**LITERATURE SURVEY:**

**General**

A literature review is a body of text that aims to review the critical points of current knowledge on and/or methodological approaches to a particular topic. It is secondary sources and discuss published information in a particular subject area and sometimes information in a particular subject area within a certain time period. Its ultimate goal is to bring the reader up to date with current literature on a topic and forms the basis for another goal, such as future research that may be needed in the area and precedes a research proposal and may be just a simple summary of sources. Usually, it has an organizational pattern and combines both summary and synthesis.

A summary is a recap of important information about the source, but a synthesis is a re-organization, reshuffling of information. It might give a new interpretation of old material or combine new with old interpretations or it might trace the intellectual progression of the field, including major debates. Depending on the situation, the literature review may evaluate the sources and advise the reader on the most pertinent or relevant of them.

**Review of Literature Survey**

**Title :** Application of Machine-learning models in Predicting Transfusion Among Complex Renal Stones Patients Receiving Percutaneous Nephrolithotomy: A Retrospective Study

**Author:** Yue Yang, Zhi Cao

**Year :** 2022

Kidney stone disease (urolithiasis) is a common urological disorder, affecting 6%-9% of the population of the entire world, and both prevalence and incidence continue to increase, with substantial variation according to age, sex, and geographic region. Complex kidney stones were defined as partial or complete staghorn stones or renal pelvic stones with the accompaniment of at least two calyceal stones regardless of their size and burden. Percutaneous nephrolithotomy (PCNL) is still the gold standard for the treatment of complex upper urinary tract stones based on the guidelines of the American Urological Association and the European Association of Urology, but perioperative bleeding is still a relatively frequent complication. Moreover, transfusion rate varies from 1–11%, and a higher transfusion rate, 2–53%, was determined in the complex kidney stones cases. Several studies have identified diabetes, hydronephrosis, stone size, number and size of tracts, staghorn calculi, and operation time as risk factors for peri-operation PCNL bleeding, but there is currently no quantitative index that can be used to predict bleeding of sufficient severity to require transfusion.

**Title :** KIDNEY STONE DETECTION USING MACHINE LEARNING ALGORITHM

**Author:** Surya v, P.V.Sumanth, U.Surendra, V.Chakradhar

**Year :** 2021

Stones in Kidneys is the major problem facing by the many of the people now a days. Many of the people is effected heavily by this problem due to delay in the detection of the stones in kidneys , Most of methods present in past days will detect the kidney stones when they are heavily effected. Now we came up with the better method called the back propagation network this process will detect the stones in kidneys in early days and it is mostly working on the machine learning algorithms. This processes is divided into multiple steps this will help the medical assistance to the human and capture the images of the stones in the kidneys, those images will be delivered with high pixels and provide the at most accuracy of the region of the stones.

**Title :** Analysis and Implementation of Kidney Stone Detection by Reaction Diffusion Level Set Segmentation Using Xilinx System Generator on FPGA

**Author:** Kalannagari Viswanath and Ramalingam Gunasundari

**Year :** 2015

Kidney stone disease is one of the major life threatening ailments persisting worldwide.The stone diseases remain unnoticed in the initial stage, which in turn damages the kidney as they develop. A majority of people are affected by kidney failure due to diabetes mellitus, hypertension, glomerulonephritis, and so forth. Since kidney malfunctioning can be menacing, diagnosis of the problem in the initial stages is advisable. Ultrasound (US) image is one of the currently available methods with noninvasive low cost and widely used imaging techniques for analyzing kidney diseases. Shock wave lithotripsy (SWL), percutaneous nephrolithotomy (PCNL), and relative super saturation (RSS) are the available practices to test urine. The Robertson Risk Factor Algorithms (RRFA) are open and are used for laparoscopic surgery; these algorithms are assigned for exceptional special cases. Hyaluronan is a large (>106 Da) linear glycosaminoglycan composed of repeating units of glucuronic acid (GlcUA) and N-acetyl glucosamine (GlcNAc) disaccharides. It has a significant role in a number of processes that can eventually lead to renal stone disease, including urine concentration, uric acid, salt form crystal, crystallization inhibition, crystal retention, magnesium ammonium phosphate, and amino acid.

# **Title :** Deep learning computer vision algorithm for detecting kidney stone composition

# **Author:** Kristian M. Black, Hei Law, Ali Aldoukhi, Jia Deng and Khurshid R. Ghani

**Year :** 2020

There is increasing interest on optimising holmium laser settings and techniques like ‘dusting’, as ureteroscopy (URS) with laser lithotripsy has become the predominant surgical treatment for urinary stones in North America. Currently, stones are fragmented by selecting pulse energy and frequency to break stones into either fine powder (dusting) or medium-sized fragments for extraction. Laser energy needed to ablate stones varies with stone composition and size. Today, surgeons manually choose laser settings based on a visual recognition of the stone type and its durability. However, if settings could be automatically calculated based on recognition of stone composition, this could improve the efficiency of lithotripsy. Furthermore, because stone samples are often extracted with baskets for composition analysis to guide management, an endoscopic visualisation system that reliably determines stone composition would have benefits in reducing operative time and surgical costs.

**Title :** Model for Predicting the Risk of Kidney Stone using Data Mining Techniques

**Author:** Oladeji F. A., Idowu P. A. and Egejuru N.

**Year :** 2019

This paper focused on the development of a predictive model for the classification of the risk of kidney stones in Nigerian using data mining techniques based on historical information elicited about the risk of kidney stones among Nigerians. Following the identification of the risk factors of kidney stone from experienced endocrinologists, structured questionnaires were used to collect information about the risk factors and the associated risk of kidney stones from selected respondents. The predictive model for the risk of kidney diseases was formulated using three (3) supervised machine learning algorithms (Decision Tree, Multi-layer perception and Genetic Algorithm) following the identification of relevant features. The predictive model was simulated using the Waikato Environment for Knowledge Analysis (WEKA) environment; and the model was validated using historical dataset of kidney stone risk via performance metrics: accuracy, true positive rate, precision and false positive rate.

**CHAPTER 3**

**3.1 Existing System:**

An accurate estimation of glomerular filtration rate (GFR) is clinically crucial for kidney disease diagnosis and predicting the prognosis of chronic kidney disease (CKD). Machine learning methodologies such as deep neural networks provide a potential avenue for increasing accuracy in GFR estimation. We developed a novel deep learning architecture, a deep and shallow neural network, to estimate GFR (dlGFR for short) and examined its com parative performance with estimated GFR from Modification of Diet in Renal Disease (MDRD) and Chronic Kidney Disease Epidemiology Collaboration (CKD-EPI) equations. In this work using data from studies housed in the NIDDK Central Database Repository, we built a new deep learning architecture to estimate GFR, and evaluated the comparative performance vs. established and commonly used estimating equations. Overall, our results suggest that the performance of dl- GFR evaluated by measures including bias, classification and accuracy were non-inferior to that of CKD-EPI and MDRD.

**3.1.1 Disadvantages:**

* Accuracy is low.
* Deployment is not done.

**3.2 Proposed System:**

The proposed method is to build a machine learning model for classification of Kidney stone. The process carries from data collection where the past data related to Kidney stone are collected. The Kidney stone if found before proper treatment can save lives. Machine learning is now applied and mostly used in health care where it reduces the manual effort and better model makes error less which leads in saving the life.

The data analysis is done on the dataset proper variable identification is done that is both the dependent variables and independent variables are found. The proper machine learning algorithm is applied on the dataset where the pattern of data is learned. After applying different algorithms a better algorithm is used for the prediction of the outcome.

Data Pre-processing

Training dataset

Model

Test dataset

Classification ML Algorithm

Kidney Stone Dataset

3.2.1.1 Architecture of Proposed model

**3.2.1 Advantages:**

* Accuracy will be improved.
* Project will be deployed in a web framework.

**ENVIRONMENTAL REQUIREMENTS:**

**3.3 Software Requirements:**

Operating System : Windows

Tool : Anaconda with Jupyter Notebook

**3.4 Hardware requirements:**

Processor : Pentium IV/III

Hard disk : minimum 80 GB

RAM : minimum 2 GB

**3.5** **SOFTWARE DESCRIPTION:**

Anaconda is a [free and open-source](https://en.wikipedia.org/wiki/Free_and_open-source) 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.

**CHAPTER 4**

**SYSTEM DESIGN**

#### **4.1** **Objectives**

The goal is to develop a machine learning model for kidney stone prediction, to potentially replace the updatable supervised machine learning classification models by predicting results in the form of best accuracy by comparing supervised algorithms.

**4.1.1** **Project Goals**

# **Exploration data analysis of variable identification**

* Loading the given dataset
* Import required libraries packages
* Analyze the general properties
* Find duplicate and missing values
* Checking unique and count values

# **Uni-variate data analysis**

* Rename, add data and drop the data
* To specify data type

# **Exploration data analysis of bi-variate and multi-variate**

* Plot diagram of pairplot, heatmap, bar chart and Histogram

# **Method of Outlier detection with feature engineering**

* Pre-processing the given dataset
* Splitting the test and training dataset
* Comparing the Naïve Bayes and KNN model and Logistic Regression etc.

# **Comparing algorithm to predict the result**

* Based on the best accuracy

**4.1.2** **Scope of the Project**

The scope of this paper is to implement and investigate how different supervised binary classification methods impact default prediction. The model evaluation techniques used in this project are limited to precision, sensitivity, F1-score. The main scope is to predict the kidney stone analysis.

**4.2**  **FEASIBILITY STUDY:**

## 4.2.1 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.

**4.2.2** **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 Machine learning algorithms are applied on the Training set and based on the test result accuracy, Test set prediction is done.

**4.2.3** **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.

**4.2.4** **Building the classification model**

The prediction of kidney stone, 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.

**4.2.5** **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 preprocess 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

4.2.5.1 Process of dataflow diagram

**LIST OF MODULES:**

* Data Pre-processing
* Exploratory Data Analysis of Visualization
* Implementing Algorithm 1
* Implementing Algorithm 2
* Implementing Algorithm 3
* Implementing Algorithm 4
* Deployment

**CHAPTER 5**

**PROJECT REQUIREMENTS**

**5.1 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

**5.1.1 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.

**Non-Functional Requirements:**

Process of functional steps,

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

**5.2 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)

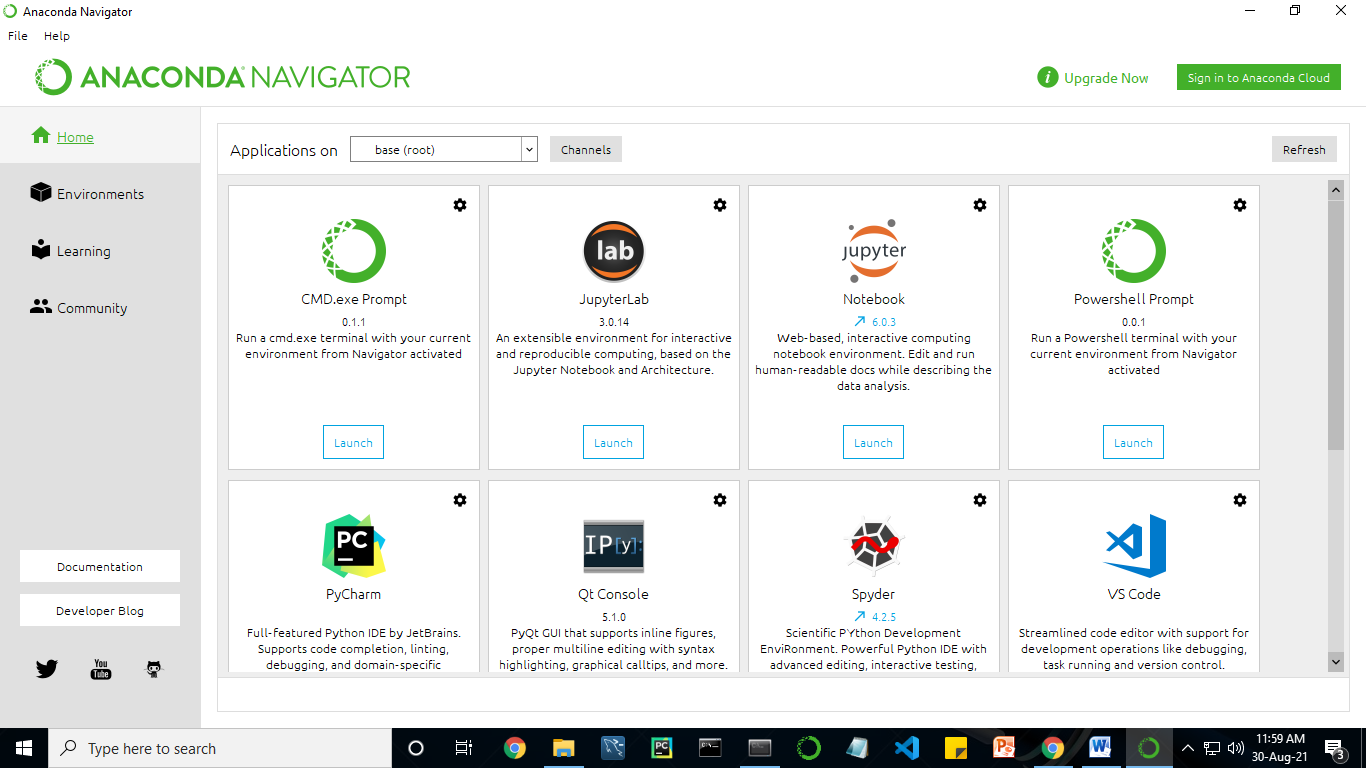


Fig 5.2.1 Anaconda Navigator Home

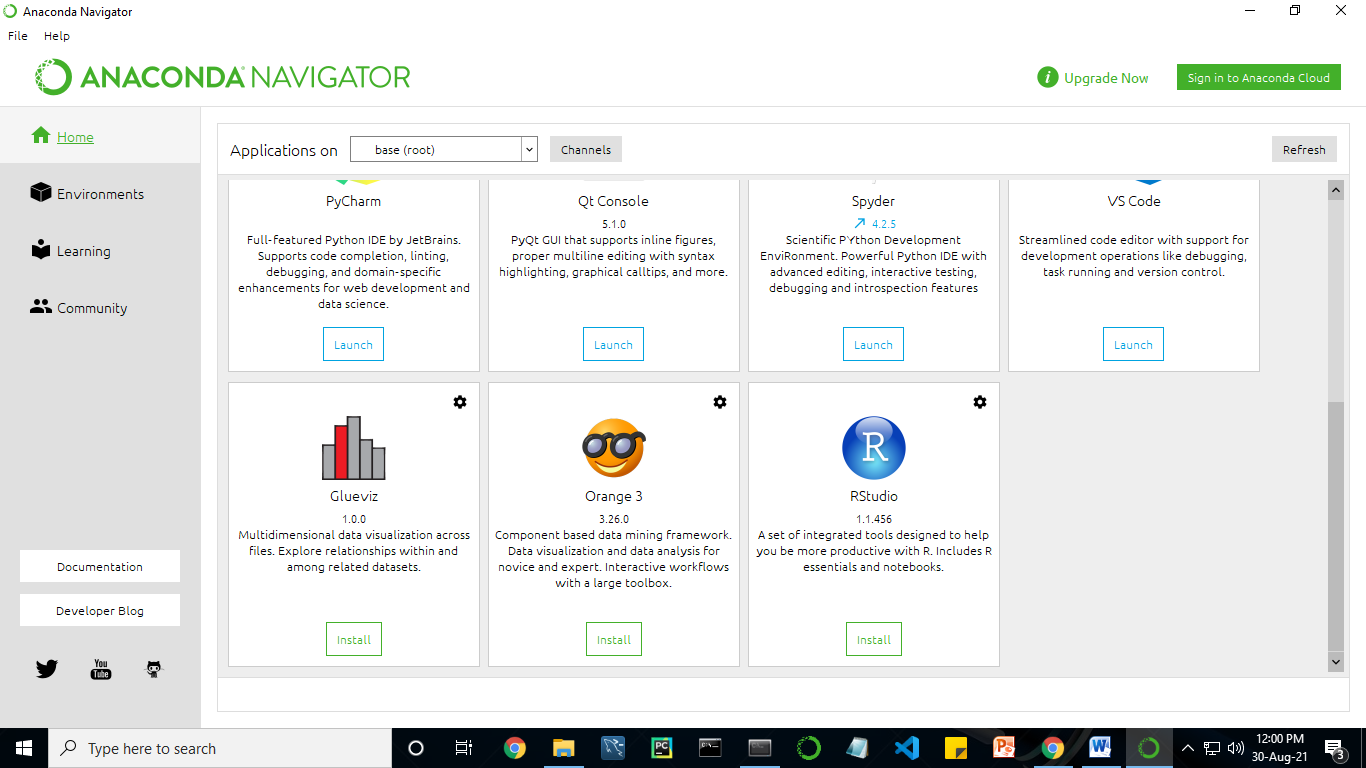


Fig 5.2.2 Anaconda Navigator Applications

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.

**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.

**5.3 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.

**5.4 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, 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 [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.

* [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 mypy supports compile-time type checking.

**5.5 SYSTEM ARCHITECTURE:**

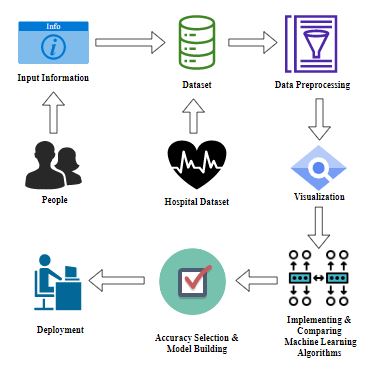


Fig 5.5.1.1 System Architecture

**5.5.1 WORK FLOW DIAGRAM:**

Source Data

Data Processing and Cleaning

Training Dataset

Testing Dataset

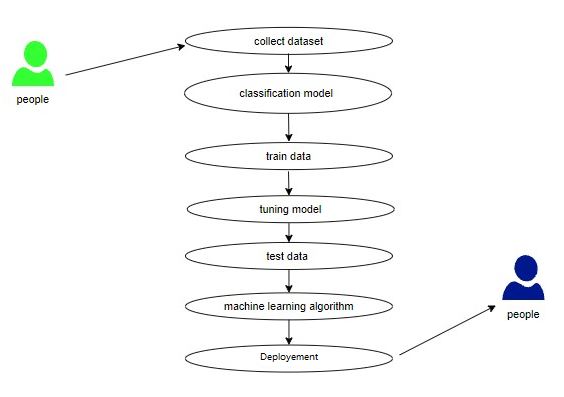
Best Model by Accuracy

Classification ML Algorithms

Finding Kidney Stone details

Fig 5.5.1.1.2 Workflow Diagram

**5.5.2 USE CASE DIAGRAM:**



**Fig 5.5.2.1.3 Use Case Diagram**

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed 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.

**5.5.3 CLASS DIAGRAM**:

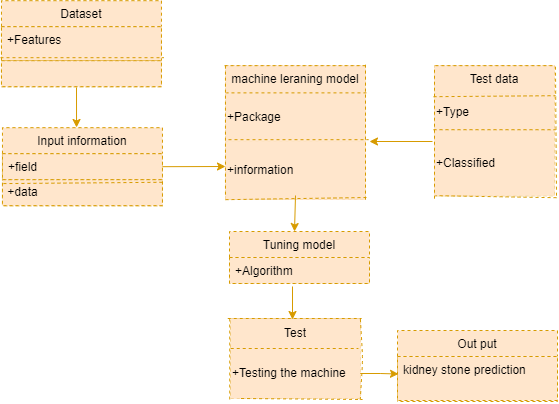


Fig 5.5.3.1.4 Class Diagram

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 a spect 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.

**5.5.4 ACTIVITY DIAGRAM**:

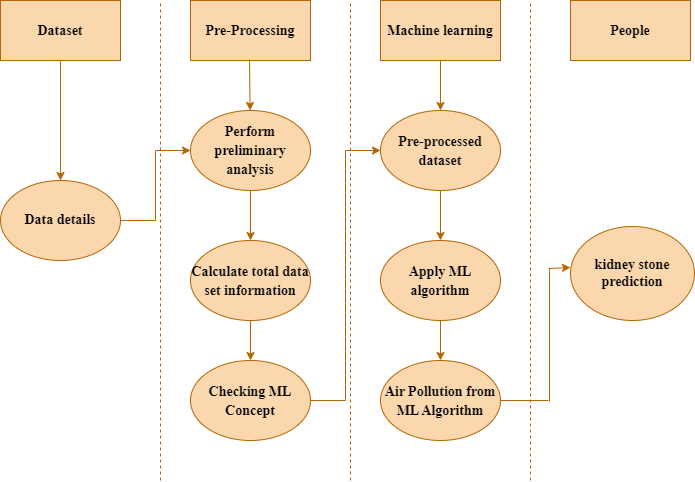


Fig 5.5.4.1.5 Activity Diagram

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.

**5.5.5 SEQUENCE DIAGRAM**:

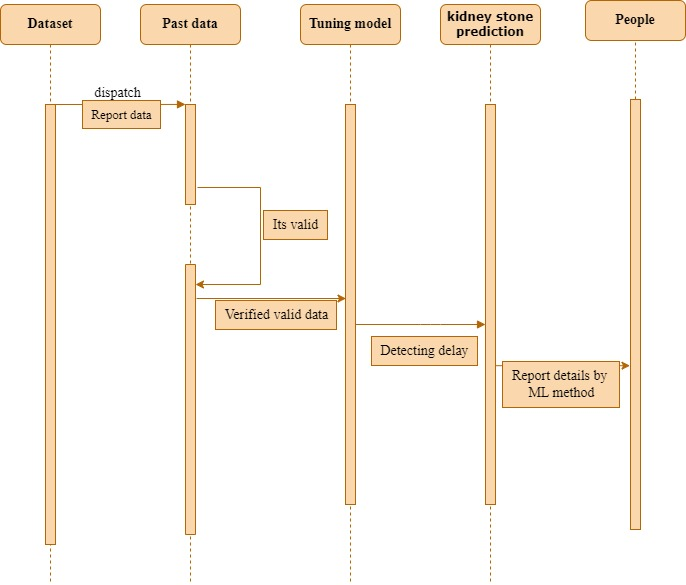


Fig 5.5.5.1.6 Sequence Diagram

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 modeling, which focuses on identifying the behavior within your system. Other dynamic modeling 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.

**5.5.6 ENTITY RELATIONSHIP DIAGRAM (ERD)**

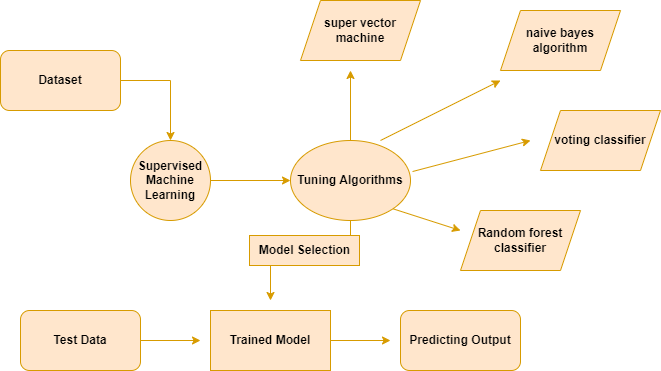


Fig 5.5.6.1.7 Entity Relationship Diagram(ERD)

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships 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.

**CHAPTER 6**

**MODULE DESCRIPTION**

**6.1 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

**6.2** **Data Validation/ Cleaning/Preparing Process**

Importing the library packages with loading given dataset. To analyzing the variable identification by data shape, data type and evaluating the missing values, duplicate values. A validation dataset is a sample of data held back from training your model that is used to give an estimate of model skill while tuning model's and procedures that you can use to make the best use of validation and test datasets when evaluating your models. Data cleaning / preparing by rename the given dataset and drop the column etc. to analyze the uni-variate, bi-variate and multi-variate process. The steps and techniques for data cleaning will vary from dataset to dataset. The primary goal of data cleaning is to detect and remove errors and anomalies to increase the value of data in analytics and decision making.

**6.3** **MODULE DIAGRAM**

**GIVEN INPUT EXPECTED OUTPUT**

input : data

output : removing noisy data

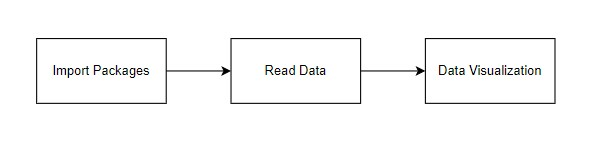
**6.3.1** **Exploration data analysis of visualization**

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

**6.3.2** **Comparing Algorithm with prediction in the form of best accuracy result**

It is important to compare the performance of multiple different machine learning algorithms consistently and it will discover to create a test harness to compare multiple different machine learning algorithms in Python with scikit-learn. It can use this test harness as a template on your own machine learning problems and add more and different algorithms to compare. Each model will have different performance characteristics. Using resampling methods like cross validation, you can get an estimate for how accurate each model may be on unseen data. It needs to be able to use these estimates to choose one or two best models from the suite of models that you have created. When have a new dataset, it is a good idea to visualize the data using different techniques in order to look at the data from different perspectives. The same idea applies to model selection. You should use a number of different ways of looking at the estimated accuracy of your machine learning algorithms in order to choose the one or two to finalize. A way to do this is to use different visualization methods to show the average accuracy, variance and other properties of the distribution of model accuracies.

In the next section you will discover exactly how you can do that in Python with scikit-learn. The key to a fair comparison of machine learning algorithms is ensuring that each algorithm is evaluated in the same way on the same data and it can achieve this by forcing each algorithm to be evaluated on a consistent test harness.

In the example below 4 different algorithms are compared:

* Support Vector Machine
* Naive Bayes algorithm
* Voting Classifier
* Random Forest Classifier

The K-fold cross validation procedure is used to evaluate each algorithm, importantly configured with the same random seed to ensure that the same splits to the training data are performed and that each algorithm is evaluated in precisely the same way. Before that comparing algorithm, Building a Machine Learning Model using install Scikit-Learn libraries. In this library package have to done preprocessing, linear model with logistic regression method, cross validating by KFold method, ensemble with random forest method and tree with decision tree classifier. Additionally, splitting the train set and test set. To predicting the result by comparing accuracy.

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. To achieving better results from the applied model in Machine Learning method of the data has to be in a proper manner. Some specified Machine Learning model needs information in a specified format, for example, Random Forest algorithm does not support null values. Therefore, to execute random forest algorithm null values have to be managed from the original raw data set. And another aspect is that data set should be formatted in such a way that more than one Machine Learning and Deep Learning algorithms are executed in given dataset.

**Prediction result by accuracy:**

Logistic regression algorithm also uses a linear equation with independent predictors to predict a value. The predicted value can be anywhere between negative infinity to positive infinity. It need the output of the algorithm to be classified variable data. Higher accuracy predicting result is logistic regression model by comparing the best accuracy.

**False Positives (FP):** A person who will pay predicted as defaulter. When actual class is no and predicted class is yes. E.g. if actual class says this passenger did not survive but predicted class tells you that this passenger will survive.

**False Negatives (FN):** A person who default predicted as payer. When actual class is yes but predicted class in no. E.g. if actual class value indicates that this passenger survived and predicted class tells you that passenger will die.

**True Positives (TP):** A person who will not pay predicted as defaulter. These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes. E.g. if actual class value indicates that this passenger survived and predicted class tells you the same thing.

**True Negatives (TN):** A person who default predicted as payer. These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no. E.g. if actual class says this passenger did not survive and predicted class tells you the same thing.

True Positive Rate(TPR) = TP / (TP + FN)

False Positive rate(FPR) = FP / (FP + TN)

**Accuracy:** The Proportion of the total number of predictions that is correct otherwise overall how often the model predicts correctly defaulters and non-defaulters.

**Accuracy calculation:**

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same.

**Precision:** The proportion of positive predictions that are actually correct.

Precision = TP / (TP + FP)

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. The question that this metric answer is of all passengers that labeled as survived, how many actually survived? High precision relates to the low false positive rate. We have got 0.788 precision which is pretty good.

**Recall:** The proportion of positive observed values correctly predicted. (The proportion of actual defaulters that the model will correctly predict)

Recall = TP / (TP + FN)

Recall(Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

**F1 Score** is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it’s better to look at both Precision and Recall.

**General Formula:**

F- Measure = 2TP / (2TP + FP + FN)

**F1-Score Formula:**

F1 Score = 2\*(Recall \* Precision) / (Recall + Precision)

**6.4** **ALGORITHM AND TECHNIQUES**

**6.4.1** **Algorithm Explanation**

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. In Supervised Learning, algorithms learn from labeled data. After understanding the data, the algorithm determines which label should be given to new data based on pattern and associating the patterns to the unlabeled new data.

**Used Python Packages:**

**sklearn:**

* + In python, sklearn is a machine learning package which include a lot of ML algorithms.
  + Here, we are using some of its modules like train\_test\_split, DecisionTreeClassifier or Logistic Regression and accuracy\_score.

**NumPy:**

* + It is a numeric python module which provides fast maths functions for calculations.
  + It is used to read data in numpy arrays and for manipulation purpose.

**Pandas:**

* + Used to read and write different files.
  + Data manipulation can be done easily with data frames.

**Matplotlib:**

* + Data visualization is a useful way to help with identify the patterns from given dataset.
  + Data manipulation can be done easily with data frames.

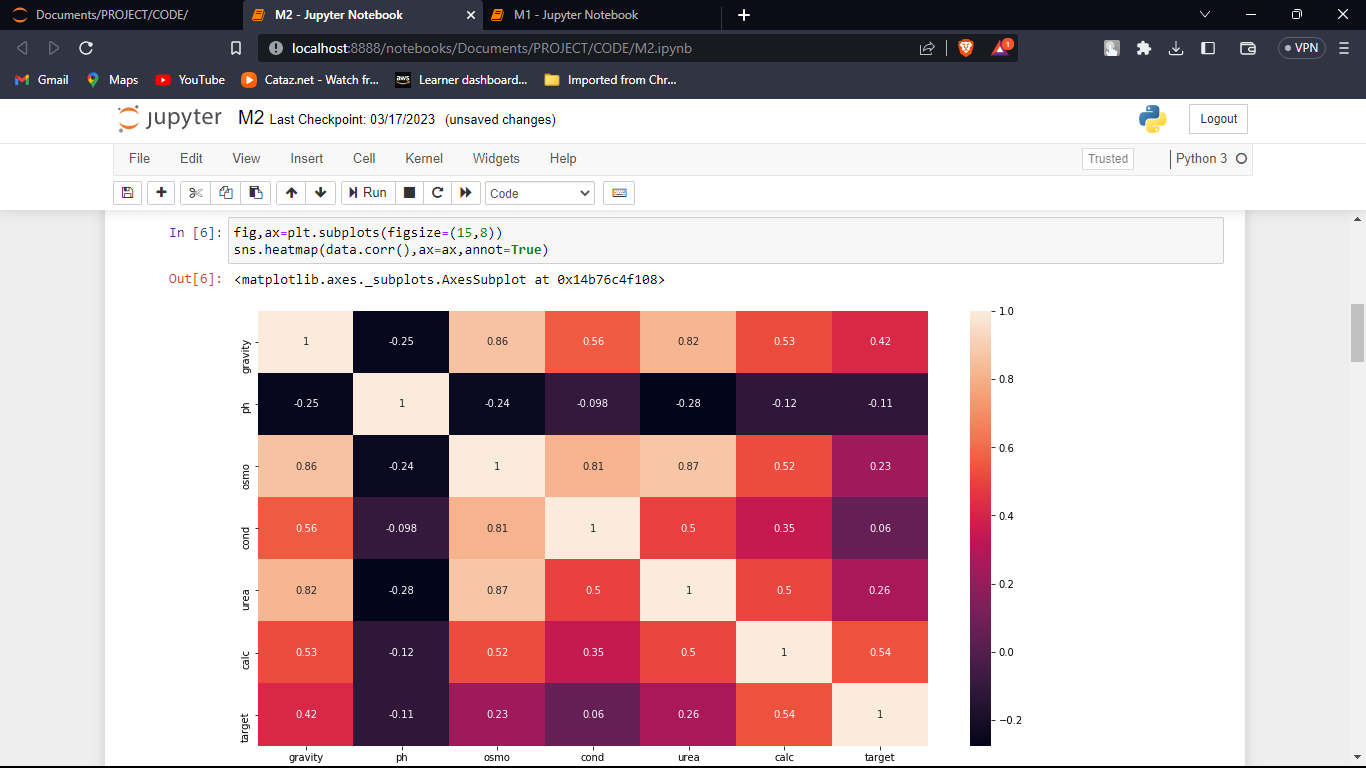


Fig 6.4.1.1.1 PH Value Content In kidney Stone

**6.4.1.1 Support Vector Machine:**

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.

**MODULE DIAGRAM**



**GIVEN INPUT EXPECTED OUTPUT**

input : data

output : getting accuracy

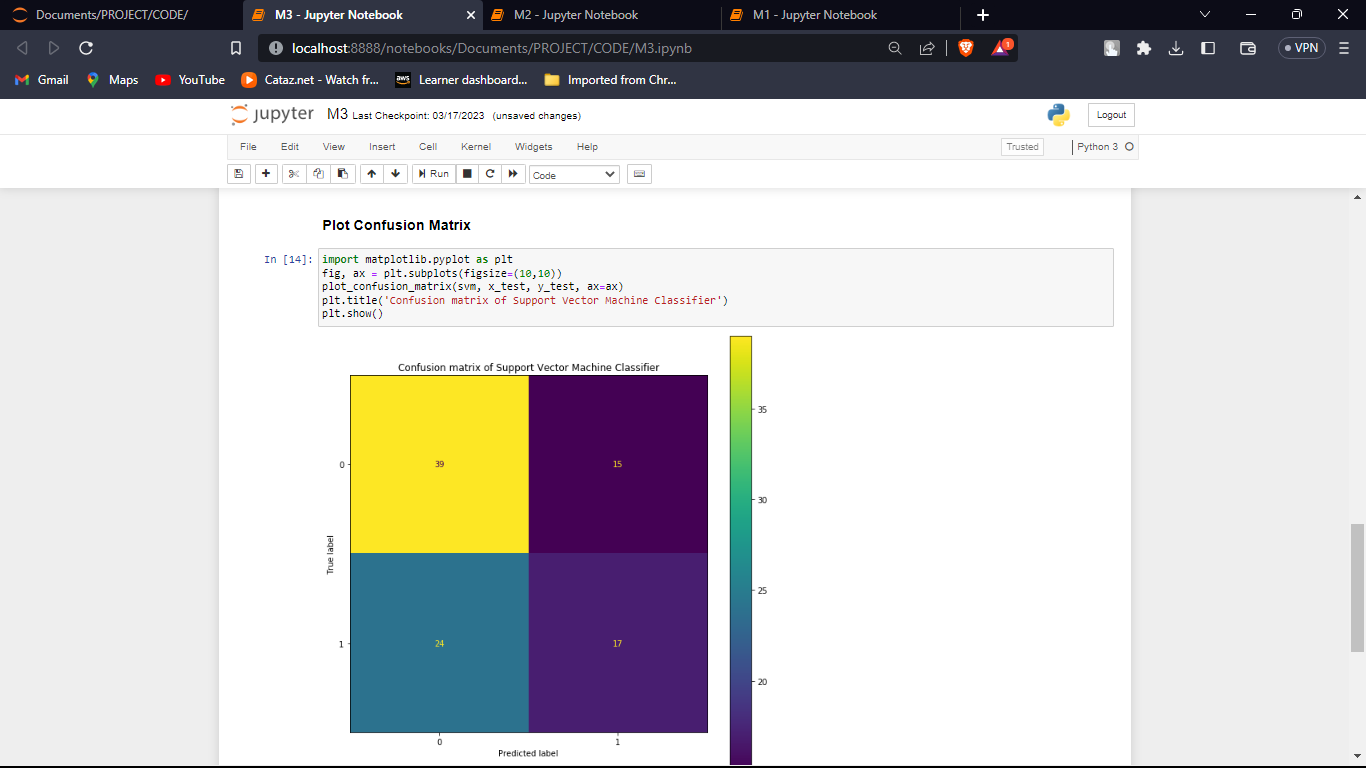


Fig 6.4.1.1.2 Plot confusion Matrix In SVM

**6.4.1.2 Naive Bayes algorithm**

* The Naive Bayes algorithm is an intuitive method that uses the probabilities of each attribute belonging to each class to make a prediction. It is the supervised learning approach you would come up with if you wanted to model a predictive modeling problem probabilistically.
* Naive bayes simplifies the calculation of probabilities by assuming that the probability of each attribute belonging to a given class value is independent of all other attributes. This is a strong assumption but results in a fast and effective method.
* The probability of a class value given a value of an attribute is called the conditional probability. By multiplying the conditional probabilities togeth er for each attribute for a given class value, we have a probability of a data instance belonging to that class. To make a prediction we can calculate probabilities of the instance belonging to each class and select the class value with the highest probability.
* Naive Bayes is a statistical classification technique based on Bayes Theorem. It is one of the simplest supervised learning algorithms.Naive Bayes classifier is the fast, accurate and reliable algorithm. Naive Bayes classifiers have high accuracy and speed on large datasets Naive Bayes classifier assumes that the effect of a particular feature in a class is independent of other features. For example, a loan applicant is desirable or not depending on his/her income, previous loan and transaction history, age, and location.
* Even if these features are interdependent, these features are still considered independently. This assumption simplifies computation, and that's why it is considered as naive. This assumption is called class conditional independence.

**MODULE DIAGRAM**



**GIVEN INPUT EXPECTED OUTPUT**

input : data

output : getting accuracy

**6.4.1.3 Voting Classifier:**

A Voting Classifier is a machine learning model that trains on an ensemble of numerous models and predicts an output (class) based on their highest probability of chosen class as the output.  
It simply aggregates the findings of each classifier passed into Voting Classifier and predicts the output class based on the highest majority of voting. The idea is instead of creating separate dedicated models and finding the accuracy for each them, we create a single model which trains by these models and predicts output based on their combined majority of voting for each output class.

Voting Classifier supports two types of votings.

1. **Hard Voting:** In hard voting, the predicted output class is a class with the highest majority of votes i.e the class which had the highest probability of being predicted by each of the classifiers. Suppose three classifiers predicted the output class(A, A, B), so here the majority predicted A as output. Hence A will be the final prediction.
2. **Soft Voting**: In soft voting, the output class is the prediction based on the average of probability given to that class. Suppose given some input to three models, the prediction probability for class A = (0.30, 0.47, 0.53) and B = (0.20, 0.32, 0.40). So the average for class A is 0.4333 and B is 0.3067, the winner is clearly class A because it had the highest probability averaged by each classifier.

**MODULE DIAGRAM**



**GIVEN INPUT EXPECTED OUTPUT**

input : data

output : getting accuracy

**6.4.1.4 Random Forest Classifier:**

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.*

As the name suggests, *"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."* Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

**MODULE DIAGRAM**



**GIVEN INPUT EXPECTED OUTPUT**

input : data

output : getting accuracy

**Deployment:**

Deployment is the mechanism through which applications, modules, updates, and patches are delivered from developers to users. The methods used by developers to build, test and deploy new code will impact how fast a product can respond to changes in customer preferences or requirements and the quality of each change. Deployment refers to assigning people to serve in various locations, especially soldiers and other military personnel. A deployment may include soldiers, as well as equipment and generals. Deployments might be part of a war or a peaceful mission. The deployment process flow consists of five steps: Planning, development, testing, deploying, and monitoring.

**CHAPTER 7**

**CONCLUSION**

The analytical process started from data cleaning and processing, missing values analysis, exploratory analysis and finally model building and evaluation. The best accuracy on public test set is higher accuracy score will be find out. This project can help to find the Prediction of Kidney Stone.

**CHAPTER 8**

**FUTURE WORK**

* Kidney Stone prediction to connect with Cloud.
* To optimize the work to implement in web development.

**CHAPTER 9**

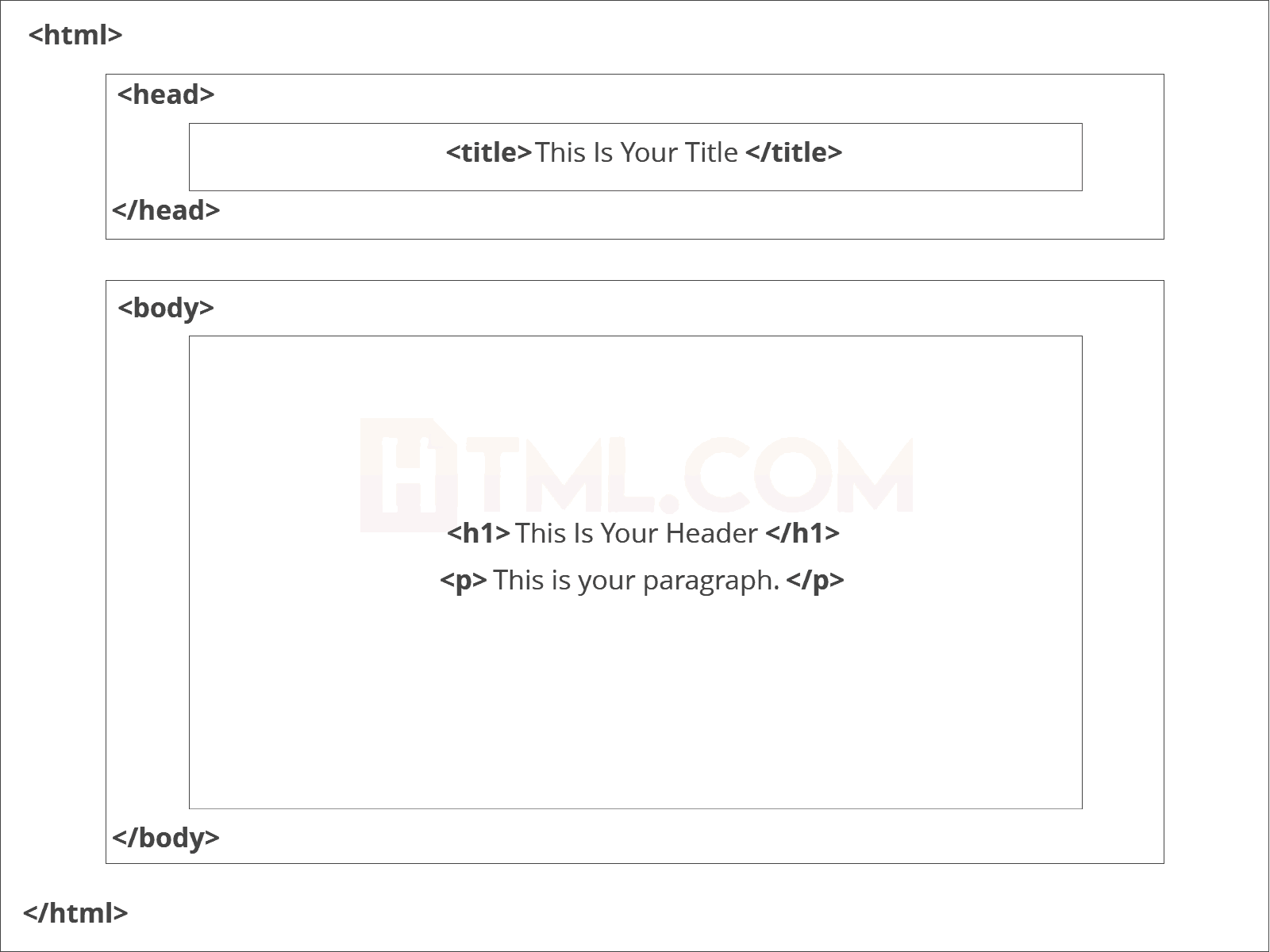
**APPENDIX 1 – SOURCE CODE**

**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 **metadata for the page** goes — stuff mostly meant for search engines and other computer programs.

<body> — This is where the**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>>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”>

**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

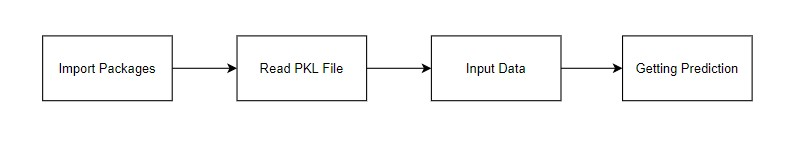
**CSS Background**

* 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

**Module Diagram**



**Given Input Expected Output**

input : data values

output : predicting output

**Module 1:**

**Coding:**

### Importing warnings to avoid any Warnings

In [ ]:

### Importing warnings to avoid any Warnings

In [ ]:

**import** warnings

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

### Importing Pandas and Numpy

In [ ]:

**import** pandas **as** pd

**import** numpy **as** np

### Loading CSV File

In [ ]:

data**=**pd**.**read\_csv('KS.csv')

### Size of our Dataset

In [ ]:

print(data**.**shape)

### Lets Know our Parameters

In [ ]:

data**.**columns

### Lets See is there any Null Values

In [ ]:

data**.**isnull()**.**sum()

In [ ]:

*# So There were no null values*

### Lets Check Unique Values

In [ ]:

data['gravity']**.**unique()

In [ ]:

data['ph']**.**unique()

In [ ]:

data['osmo']**.**unique()

In [ ]:

data['cond']**.**unique()

In [ ]:

data['urea']**.**unique()

In [ ]:

data['calc']**.**unique()

In [ ]:

data['target']**.**unique()

### Let we see first five rows of the data

In [ ]:

data**.**head()

### Let's locate our central Values in Each Column

In [ ]:

data**.**describe()

In [ ]:

data['target']**.**value\_counts()

In [ ]:

data**.**corr()

**Module** 2:

**import** warnings

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

### Importing Pandas and Numpy

In [ ]:

**import** pandas **as** pd

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

### Loading CSV File

In [ ]:

data**=**pd**.**read\_csv('KS.csv')

In [ ]:

data**.**columns

### Let's Seperate Features and Target

In [ ]:

features**=**['gravity', 'ph', 'osmo', 'cond', 'urea', 'calc']

target**=**['target']

*## So our target column is target and 0 means kidney stone was not formed*

*## Whereas 1 means kidney stone was formed*

### As there were no Duplicates and no null values we don't Need to do any alteration to the data

## Let's See Pictographs

### lets see Heatmap of the Data

In [ ]:

fig,ax**=**plt**.**subplots(figsize**=**(15,8))

sns**.**heatmap(data**.**corr(),ax**=**ax,annot**=True**)

In [ ]:

*### This is the relation between the columns. Let's make a graph with related columns*

### Distribution Of Calcium Concentration In The Tested People

In [ ]:

p**=**sns**.**factorplot('calc',data**=**data,kind**=**'count',hue**=**'target')

p**.**set\_xlabels('Calcium')

p**.**set\_ylabels('')

p**.**fig**.**set\_figheight(3)

p**.**fig**.**set\_figwidth(100)

print('Here on y axis we have calcium concentration in urine with target differentiation.')

print('We can observe that maximum affected group are lying at high calcium Concentrations.')

### Specific Gravity Relation With Kidney Stones Formation

In [ ]:

p**=**sns**.**factorplot('gravity',data**=**data,kind**=**'count',hue**=**'target')

p**.**set\_xlabels('Specific Gravity')

p**.**set\_ylabels('')

p**.**fig**.**set\_figheight(3)

p**.**fig**.**set\_figwidth(100)

print('Here on y axis we have Specific Gravity of the urine with target differentiation.')

print('We can observe that increase in specific gravity can cause increased risk of stones formation in the kidneys.')

In [ ]:

In [ ]:

plt**.**figure(figsize**=**(10,5))

plt**.**scatter(data['calc'],data['gravity'])

plt**.**title('Relation Between Specific gravity and Calcium conc')

plt**.**xlabel('Calcium Concentration')

plt**.**ylabel('Specific Gravity')

plt**.**show()

### Urea Concentration And Specific Gravity Relation

In [ ]:

plt**.**figure(figsize**=**(10,5))

plt**.**scatter(data['urea'],data['osmo'])

plt**.**title('Relation Between Specific gravity and Calcium conc')

plt**.**xlabel('Urea Concentration ')

plt**.**ylabel('Osmolarity')

plt**.**show()

print('We can see that osmolarity of the urine is gradually increased with increase in urea concentration.')

In [ ]:

In [ ]:

data["osmo"]**.**hist(figsize**=**(15,4), color **=** 'green')

plt**.**title("Osmolarity Distribution")

plt**.**xlabel('Osmolarity')

plt**.**show()

In [ ]:

*#Propagation by variable*

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

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

ax **=** dataframe\_pie**.**plot**.**pie(figsize**=**(7,7), autopct**=**'%1.2f%%', fontsize **=** 12)

ax**.**set\_title(variable **+** ' \n', fontsize **=** 15)

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

PropByVar(data, 'target')

In [ ]:

fig, ax **=** plt**.**subplots(figsize**=**(15,6))

sns**.**boxplot(data**.**calc, ax **=**ax)

plt**.**title("Calcium concentrations in tested data")

plt**.**show()

**Module** **3**:

## Support Vector Machine

In [ ]:

**import** warnings

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

In [ ]:

**import** pandas **as** pd

**import** numpy **as** np

**from** sklearn.svm **import** SVC

**from** sklearn.metrics **import** classification\_report, confusion\_matrix, plot\_confusion\_matrix, accuracy\_score

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

### Loading CSV File

In [ ]:

data**=**pd**.**read\_csv('KS.csv')

In [ ]:

pd**.**options**.**display**.**max\_rows**=**100

data**.**head(100)

### Let's Know the shape of our Dataset

In [ ]:

data**.**shape

### Let's Know our Columns

In [ ]:

data**.**columns

### Let's seperate data

In [ ]:

features **=** ['gravity', 'ph', 'osmo', 'cond', 'urea', 'calc']

target**=**['target']

In [ ]:

x**=**data[features]

y**=**data[target]

### Let's Split and Train Data

In [ ]:

x\_train,x\_test,y\_train,y\_test**=**train\_test\_split(x,y,test\_size**=**0.3,stratify**=**y, random\_state**=**32)

In [ ]:

svm**=** SVC()

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

In [ ]:

y\_prediction**=**svm**.**predict(x\_test)

### Accuracy Score

In [ ]:

Accuracy\_Score**=** accuracy\_score(y\_test,y\_prediction)

print('The Accuracy Score is :',Accuracy\_Score)

### Confusion matrix

In [ ]:

confusion\_matrix(y\_test,y\_prediction)

### Plot Confusion Matrix

In [ ]:

**import** matplotlib.pyplot **as** plt

fig, ax **=** plt**.**subplots(figsize**=**(10,10))

plot\_confusion\_matrix(svm, x\_test, y\_test, ax**=**ax)

plt**.**title('Confusion matrix of Support Vector Machine Classifier')

plt**.**show()

### Classification Report

In [ ]:

classification\_Report**=** classification\_report(y\_test,y\_prediction)

print(classification\_Report)

In [ ]:

In [ ]:

**Module** **4**:

## Gaussian Naive Bayes

In [ ]:

**import** warnings

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

In [ ]:

**import** pandas **as** pd

**import** numpy **as** np

**from** sklearn.naive\_bayes **import** GaussianNB

**from** sklearn.metrics **import** classification\_report, confusion\_matrix, plot\_confusion\_matrix, accuracy\_score

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

### Loading CSV File

In [ ]:

data**=**pd**.**read\_csv('KS.csv')

### Let's Know the shape of our Dataset

In [ ]:

data**.**shape

### Let's Know our Columns

In [ ]:

data**.**columns

### Let's seperate data

In [ ]:

features **=** ['gravity', 'ph', 'osmo', 'cond', 'urea', 'calc']

target**=**['target']

In [ ]:

x**=**data[features]

y**=**data[target]

### Let's Split and Train Data

In [ ]:

x\_train,x\_test,y\_train,y\_test**=**train\_test\_split(x,y,test\_size**=**0.3, stratify**=**y, random\_state**=**32)

In [ ]:

GNB**=** GaussianNB()

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

In [ ]:

y\_prediction**=**GNB**.**predict(x\_test)

### Accuracy Score

In [ ]:

Accuracy\_Score**=** accuracy\_score(y\_test,y\_prediction)

print('The Accuracy Score is :',Accuracy\_Score)

### Confusion matrix

In [ ]:

confusion\_matrix(y\_test,y\_prediction)

### Plot Confusion Matrix

In [ ]:

**import** matplotlib.pyplot **as** plt

fig, ax **=** plt**.**subplots(figsize**=**(10,10))

plot\_confusion\_matrix(GNB, x\_test, y\_test, ax**=**ax)

plt**.**title('Confusion matrix of Support Vector Machine Classifier')

plt**.**show()

### Classification Report

In [ ]:

classification\_Report**=** classification\_report(y\_test,y\_prediction)

print(classification\_Report)

In [ ]:

**Module** **5:**

## Voting Classifier

In [ ]:

**import** warnings

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

In [ ]:

**import** pandas **as** pd

**import** numpy **as** np

**from** sklearn.ensemble **import** VotingClassifier

**from** sklearn.metrics **import** classification\_report, confusion\_matrix, plot\_confusion\_matrix, accuracy\_score

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

### Loading CSV File

In [ ]:

data**=**pd**.**read\_csv('KS.csv')

### Let's Know the shape of our Dataset

In [ ]:

data**.**shape

### Let's Know our Columns

In [ ]:

data**.**columns

### Let's seperate data

In [ ]:

features **=** ['gravity', 'ph', 'osmo', 'cond', 'urea', 'calc']

target**=**['target']

In [ ]:

x**=**data[features]

y**=**data[target]

### Let's Split and Train Data

In [ ]:

x\_train,x\_test,y\_train,y\_test**=**train\_test\_split(x,y,test\_size**=**0.3,stratify**=**y)

In [ ]:

**from** xgboost **import** XGBClassifier

**from** sklearn.naive\_bayes **import** GaussianNB

**from** sklearn.svm **import** SVC

clf2 **=** XGBClassifier(random\_state**=**42)

clf3 **=** GaussianNB()

clf4 **=** SVC(probability**=True**, random\_state**=**42)

VC **=** VotingClassifier(estimators**=**[('XGB', clf2), ('GNB', clf3), ('svm', clf4)],

voting**=**'soft', weights**=**[1,1,1])

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

In [ ]:

y\_prediction**=**VC**.**predict(x\_test)

### Accuracy Score

In [ ]:

Accuracy\_Score**=** accuracy\_score(y\_test,y\_prediction)

print('The Accuracy Score is :',Accuracy\_Score)

### Confusion matrix

In [ ]:

confusion\_matrix(y\_test,y\_prediction)

### Plot Confusion Matrix

In [ ]:

**import** matplotlib.pyplot **as** plt

fig, ax **=** plt**.**subplots(figsize**=**(10,10))

plot\_confusion\_matrix(VC, x\_test, y\_test, ax**=**ax)

plt**.**title('Confusion matrix of Support Vector Machine Classifier')

plt**.**show()

### Classification Report

In [ ]:

classification\_Report**=** classification\_report(y\_test,y\_prediction)

print(classification\_Report)

In [ ]:

**Module** **6**:

## Random Forest Classifier

In [ ]:

**import** warnings

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

In [ ]:

**import** pandas **as** pd

In [ ]:

**import** numpy **as** np

In [ ]:

**from** sklearn.ensemble **import** RandomForestClassifier

In [ ]:

**from** sklearn.metrics **import** classification\_report, confusion\_matrix, plot\_confusion\_matrix, accuracy\_score

In [ ]:

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

### Loading CSV File

In [ ]:

data**=**pd**.**read\_csv('KS.csv')

### Let's Know the shape of our Dataset

In [ ]:

data**.**shape

### Let's Know our Columns

In [ ]:

data**.**columns

### Let's seperate data

In [ ]:

features **=** ['gravity', 'ph', 'osmo', 'cond', 'urea', 'calc']

target**=**['target']

In [ ]:

x**=**data[features]

y**=**data[target]

### Let's Split and Train Data

In [ ]:

x\_train,x\_test,y\_train,y\_test**=**train\_test\_split(x,y,test\_size**=**0.3,stratify**=**y)

In [ ]:

RFC**=**RandomForestClassifier()

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

In [ ]:

y\_prediction**=**RFC**.**predict(x\_test)

### Accuracy Score

In [ ]:

Accuracy\_Score**=** accuracy\_score(y\_test,y\_prediction)

print('The Accuracy Score is :',Accuracy\_Score)

### Confusion matrix

In [ ]:

confusion\_matrix(y\_test,y\_prediction)

### Plot Confusion Matrix

In [ ]:

**import** matplotlib.pyplot **as** plt

fig, ax **=** plt**.**subplots(figsize**=**(10,10))

plot\_confusion\_matrix(RFC, x\_test, y\_test, ax**=**ax)

plt**.**title('Confusion matrix of Support Vector Machine Classifier')

plt**.**show()

### Classification Report

In [ ]:

classification\_Report**=** classification\_report(y\_test,y\_prediction)

print(classification\_Report)

In [ ]:

**import** joblib

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

In [ ]:

Django DEPLOY:

*from* django.shortcuts *import* render, redirect

*from* django.contrib.auth.forms *import* UserCreationForm, AuthenticationForm

*from* django.contrib.auth *import* login *as* auth\_login, authenticate, logout

*from* . *import* forms

*from* . *import* models

*import* numpy *as* np

*import* joblib

model = joblib.load(

    "C:/Users/SPIRO/Desktop/sarthar/Sarthar/ITPML29/Deploy/latest/new/RFC.pkl"

)

*# Create your views here.*

def home\_view(*request*):

*if* *request*.method == "POST":

        username = *request*.POST["username"]

        print(username)

        password = *request*.POST["password"]

        print(password)

        name = *request*.POST["user"]

*if* name == "user":

            user = authenticate(*request*, *username*=username, *password*=password)

*# print(user)*

*if* user is not None:

                auth\_login(*request*, user)

*return* render(*request*, "new/index.html")

*else*:

                msg = "Invalid Credentials"

                form = AuthenticationForm(*request*.POST)

*return* render(

*request*, "new/user\_login.html", {"form": form, "msg": msg}

                )

*else*:

            user = authenticate(*request*, *username*=username, *password*=password)

*# print(user)*

*if* user is not None:

                auth\_login(*request*, user)

                model = models.UserPredictDataModel.objects.latest("id")

                form = forms.FeedForm(*request*.POST)

*# print(model)*

*return* render(*request*, "new/last.html", {"model": model, "form": form})

*else*:

                msg = "Invalid Credentials"

                form = AuthenticationForm(*request*.POST)

*return* render(

*request*, "new/user\_login.html", {"form": form, "msg": msg}

                )

*else*:

        form = AuthenticationForm()

*return* render(*request*, "new/user\_login.html", {"form": form})

def login(*request*):

    form = AuthenticationForm()

*return* render(*request*, "new/login.html", {"form": form})

def user\_register(*request*):

*if* *request*.method == "POST":

        form = UserCreationForm(*request*.POST)

*if* form.is\_valid():

*# print('saving')*

            form.save()

*return* render(

*request*,

                "new/user\_signup.html",

                {"msg": "Registered Successfully", "form": form},

            )

*else*:

        form = UserCreationForm()

*return* render(*request*, "new/user\_signup.html", {"form": form})

def register(*request*):

*if* *request*.method == "POST":

        form = UserCreationForm(*request*.POST)

*if* form.is\_valid():

*# print('saving')*

            form.save()

*return* render(

*request*,

                "new/user\_signup.html",

                {"msg": "Registered Successfully", "form": form},

            )

*else*:

        form = UserCreationForm()

*return* render(*request*, "new/signup.html", {"form": form})

def predict\_view(*request*):

*if* *request*.method == "POST":

        print("IF")

        fieldss = ["gravity", "ph", "osmo", "cond", "urea", "calc"]

        form = forms.UserPredictDataForm(*request*.POST)

        features = []

*for* i *in* fieldss:

            info = *request*.POST[i]

            features.append(info)

        final\_features = [np.array(features)]

*# print(final\_features)*

        prediction = model.predict(final\_features)

*# print(prediction)*

        output = prediction[0]

*if* output == 0:

            output = "You Are Not In A Risk"

*elif* output == 1:

            output = "You Are In A Risk. Please Visit A Nephrologist."

        print(features)

        print(output)

*if* form.is\_valid():

            print("saving")

            form.save()

        ob = models.UserPredictDataModel.objects.latest("id")

        ob.Attack\_type = output

        ob.save()

*return* render(

*request*, "new/index.html", {"prediction\_text": output, "form": form}

        )

*else*:

        print("ELSE")

        form = forms.UserPredictDataForm(*request*.POST)

*return* render(*request*, "new/index.html", {"form": form})

def view\_all(*request*):

    model = models.UserPredictDataModel.objects.all()

*# print(model)*

*return* render(*request*, "new/all.html", {"model": model})

def view\_last(*request*):

*if* *request*.method == "POST":

        form = forms.FeedForm(*request*.POST)

*# print('form',form)*

*if* form.is\_valid():

            form.save()

            model = models.UserPredictDataModel.objects.latest("id")

*# print(model)*

*return* render(

*request*, "new/last.html", {"model": model, "msg": "Feedback sent"}

            )

*else*:

            model = models.UserPredictDataModel.objects.latest("id")

*return* render(

*request*, "new/last.html", {"model": model, "msg": "Feedback Error"}

            )

*else*:

        form = forms.FeedForm()

        model = models.UserPredictDataModel.objects.latest("id")

*return* render(*request*, "new/last.html", {"model": model, "form": form})

def feedback(*request*):

    model = models.FeedModel.objects.latest("id")

*return* render(*request*, "new/feedback.html", {"model": model})

def apredict(*request*):

*return* render(*request*, "new/index.html")

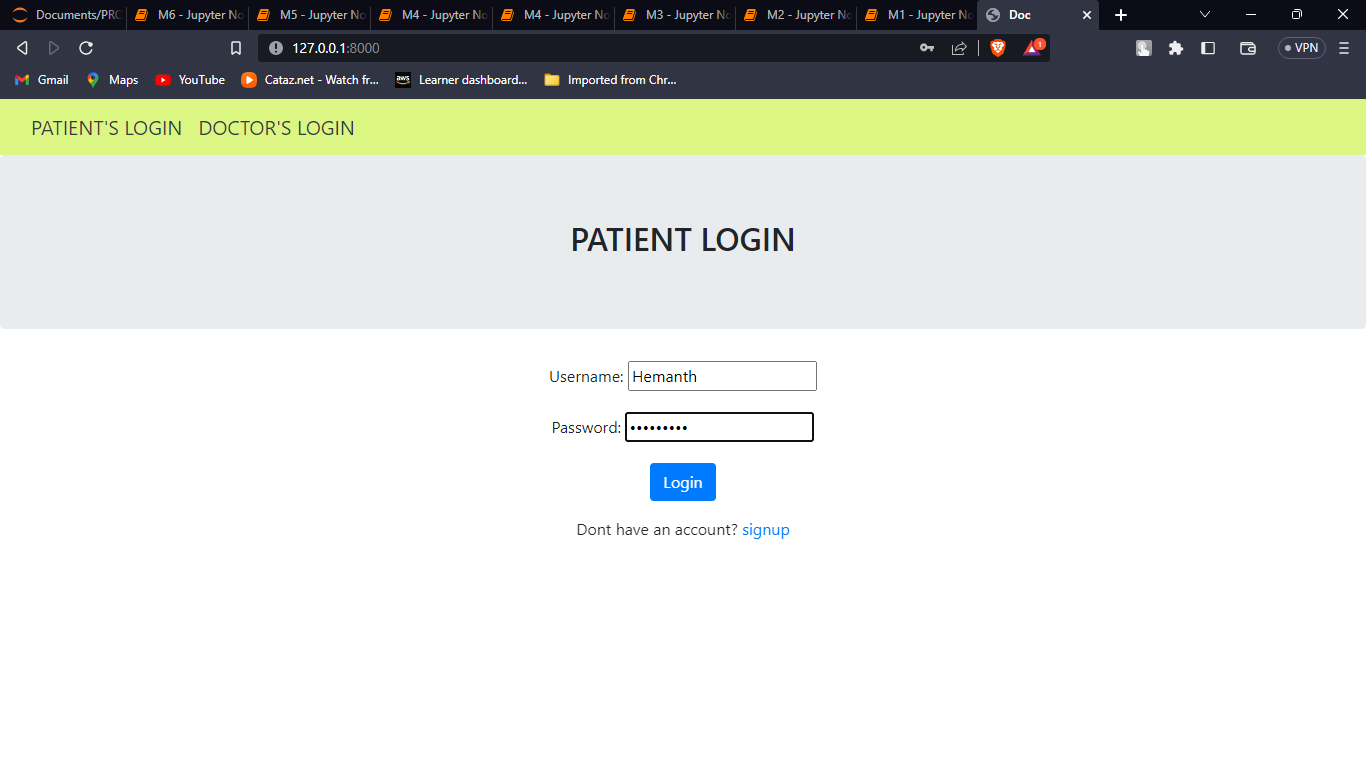
def logout\_view(*request*):

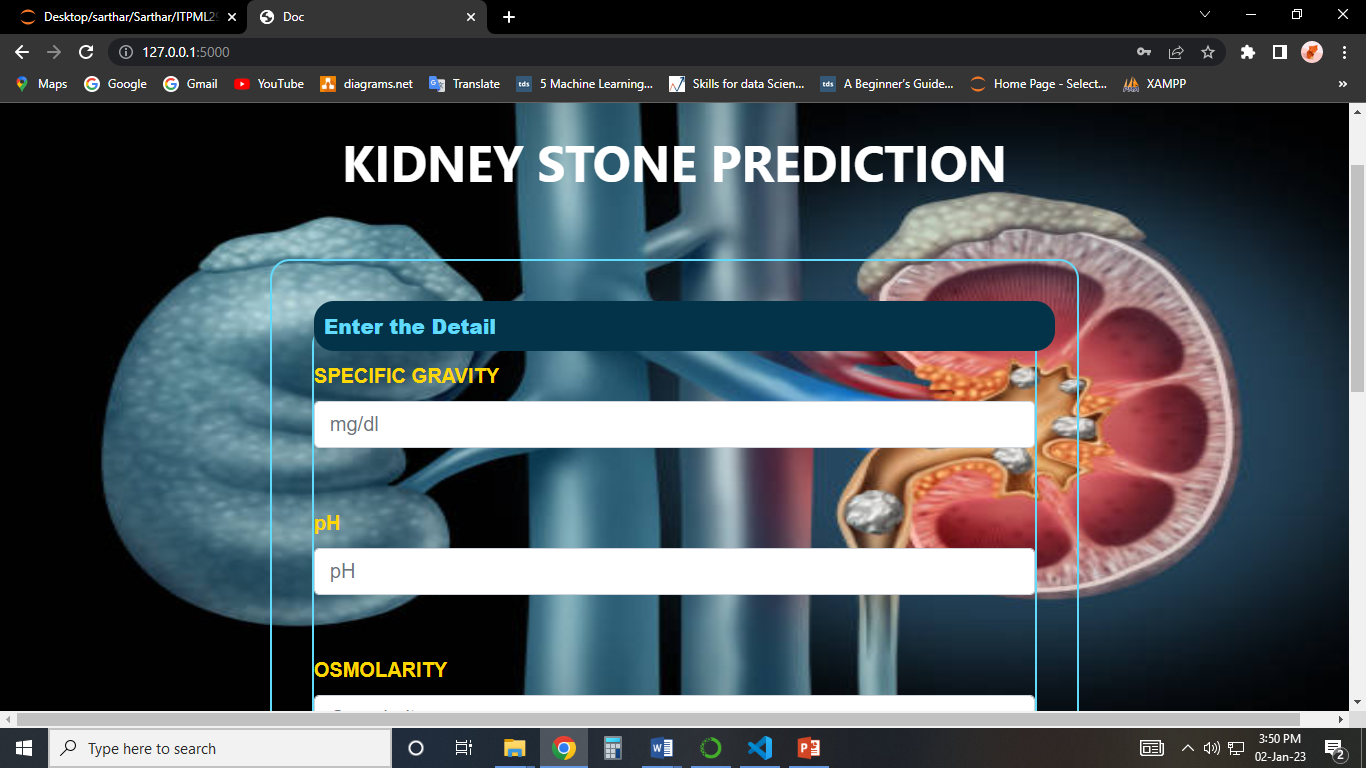
    logout(*request*)

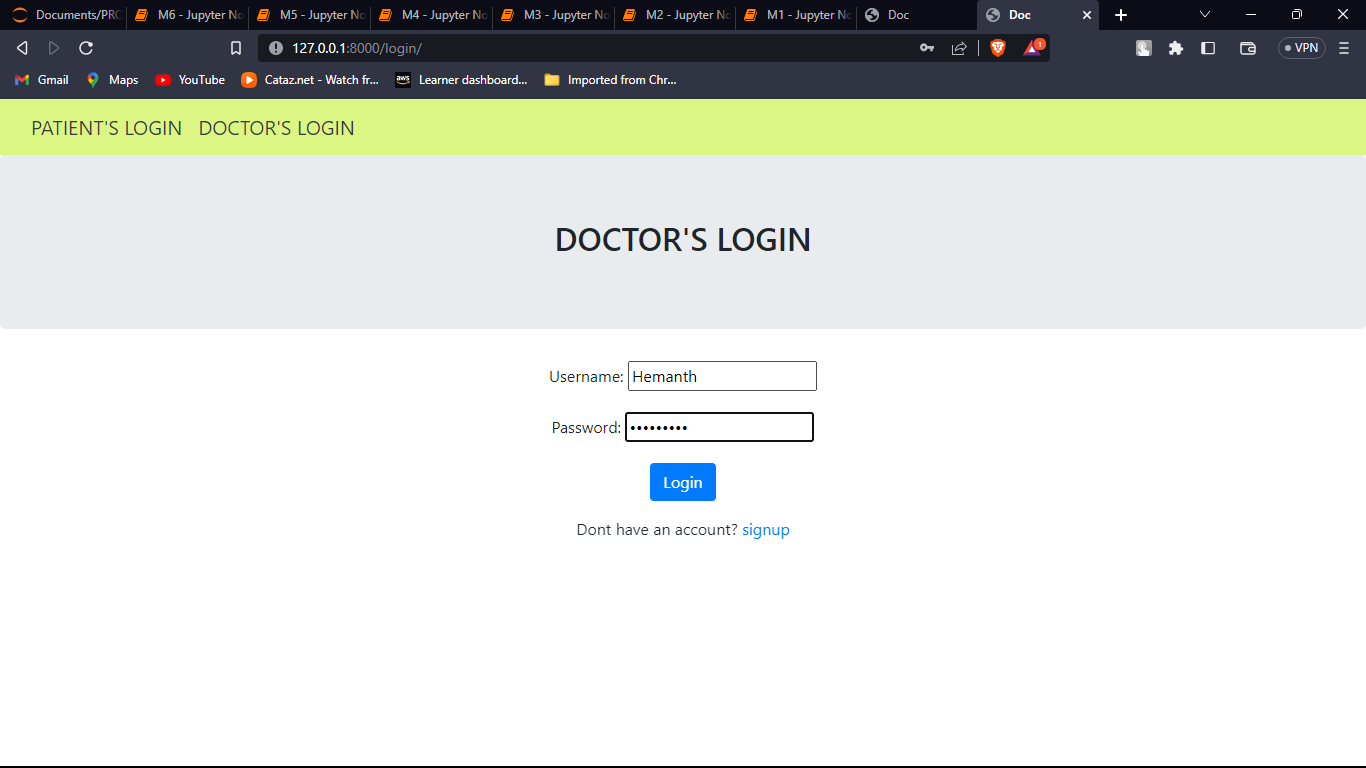
*return* redirect("home\_view")

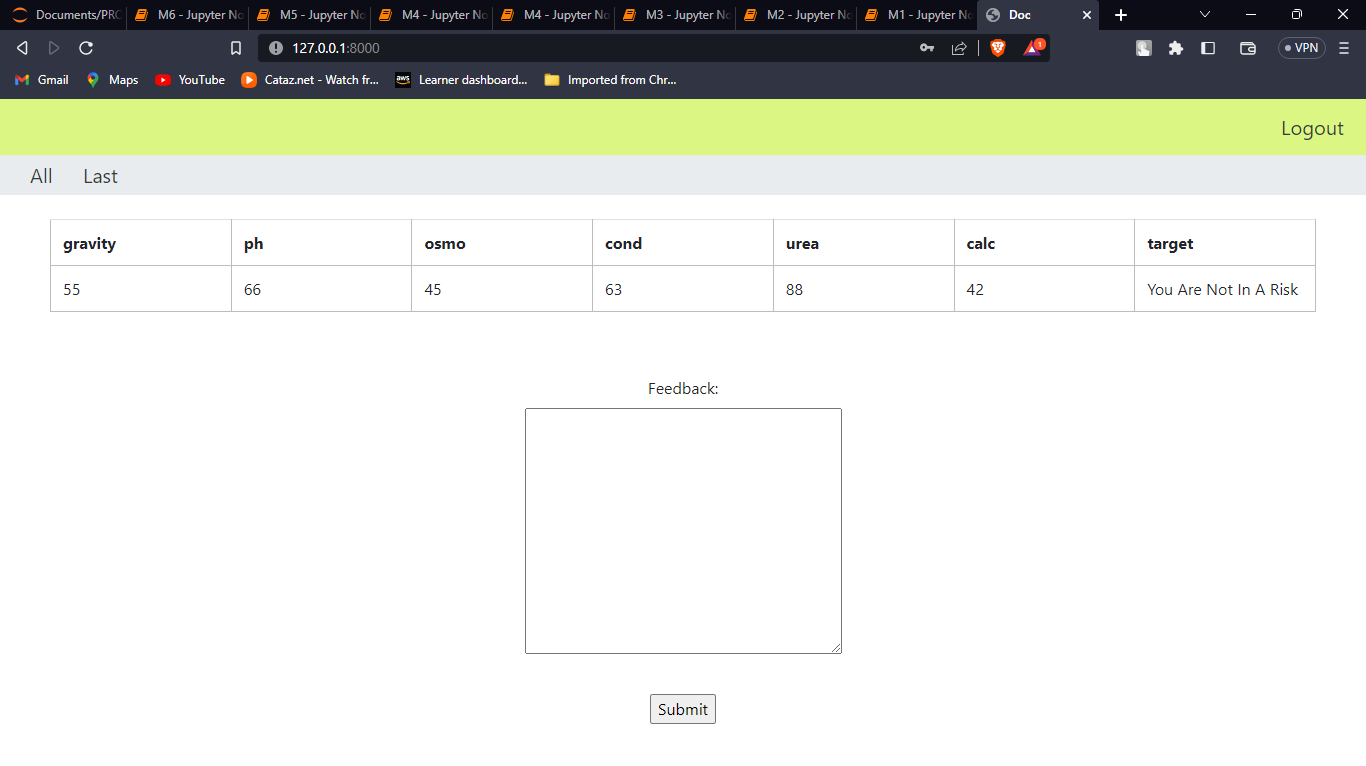
**APPENDIX 2**

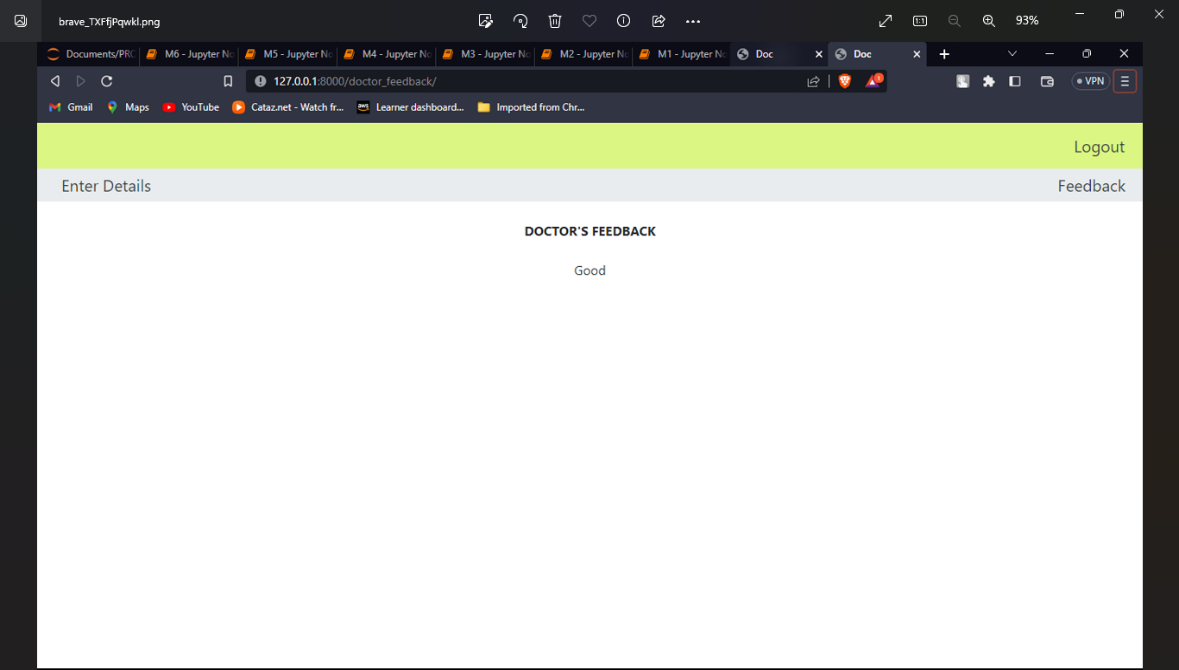
**OUTPUT SCREENSHOTS**





****





**CHAPTER 9**

**REFERENCES**

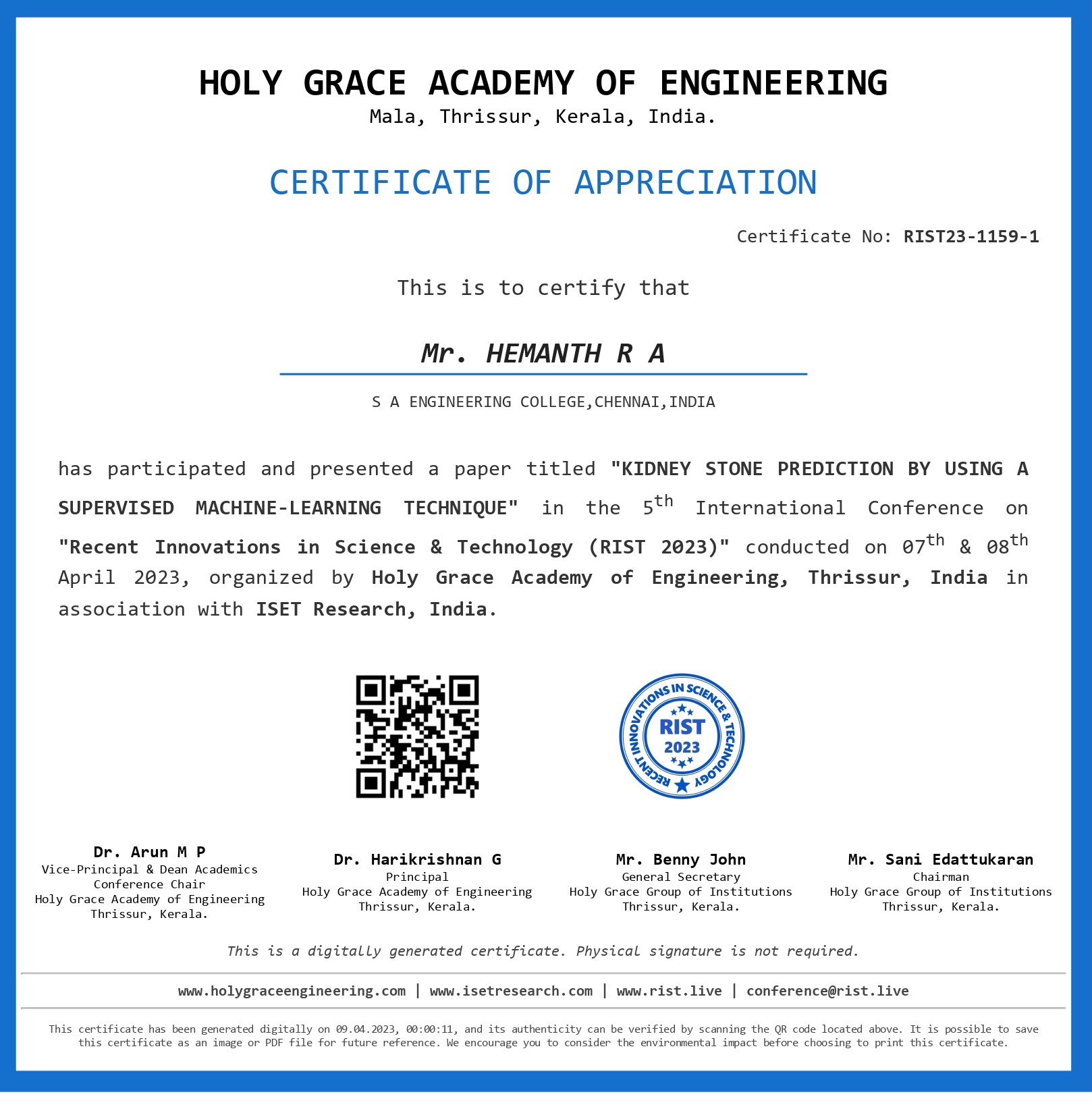
1. H. I. Feldman *et al.*, “The chronic renal insufficiency cohort (CRIC) study: Design and methods,” *J. Amer. Soc. Nephrol.*, vol. 14, no. 2, pp. S148–S153, 2003.
2. G. Beck *et al.*, “Design and statistical issues of the modification of diet in renal disease trial. The modification of diet in renal disease study group,” *Controlled Clin. Trials*, vol. 12, no. 5, pp. 566–586, 1991.
3. J. T. Wright *et al.*, “Design and baseline characteristics of partici- pants in the African American study of kidney disease and hyper- tension (AASK) pilot study,” *Controlled Clin. Trials*, vol. 17, no. 4, pp. S3–S16, Aug. 1996.
4. M. Sika *et al.*, “Baseline characteristics of participants in the African American study of kidney disease and hypertension (AASK) clinical trial and cohort study,” *Amer. J. Kidney Diseases*, vol. 50, no. 1, pp. 78–89, 2007.
5. D. M. Nathan *et al.*, “The effect of intensive treatment of diabetes on the development and progression of long-term complications in insulin- dependent diabetes mellitus,” *New England J. Med.*, vol. 329, no. 14, pp. 977–986, 1993.
6. D. M. Nathan *et al.*, “Intensive diabetes treatment and cardiovascular disease in patients with type 1 diabetes,” *New England J. Med.*, vol. 353, no. 25, pp. 2643–2653, 2005.
7. A. B. Chapman *et al.*, “Renal structure in early autosomal-dominant polycystic kidney disease (ADPKD): The consortium for radiologic imaging studies of polycystic kidney disease (CRISP) cohort,” *Kidney Int.*, vol. 64, no. 3, pp. 1035–1045, 2003.
8. B. L. Kasiske *et al.*, “A prospective controlled study of kidney donors: Baseline and 6-month follow-up,” *Amer. J. Kidney Diseases*, vol. 62, no. 3, pp. 577–586, Sep. 2013.
9. B. L. Kasiske *et al.*, “A prospective controlled study of living kidney donors: Three-year follow-up,” *Amer. J. Kidney Diseases*, vol. 66, no. 1, pp. 114–124, Jul. 2015.
10. P. Delanaye, “How measuring glomerular filtration rate? Comparison of reference methods,” in *Basic Nephrology Acute Kidney Injury*. IntechOpen, 2012.
11. L. A. Stevens and A. S. Levey, “Measured GFR as a confirmatory test for estimated GFR,” *J. Amer. Soc. Nephrol.*, vol. 20, no. 11, pp. 2305–2313, Nov. 2009.
12. F. Gaspari *et al.*, “Precision of plasma clearance of iohexol for estimation of GFR in patients with renal disease,” *J. Amer. Soc. Nephrol.*, vol. 9, no. 2, pp. 310–313, Feb. 1998.
13. A. H. Anderson *et al.*, “Estimating GFR among participants in the chronic renal insufficiency cohort (CRIC) study,” *Amer. J. Kidney Diseases*, vol. 60, no. 2, pp. 250–261, Aug. 2012.
14. L. A. Inker *et al.*, “Estimating glomerular filtration rate from serum creatinine and cystatin C,” *New England J. Med.*, vol. 367, no. 1, pp. 20–29, 2012.
15. A. S. Levey *et al.*, “Definition and classification of chronic kidney disease: A position statement from kidney disease: Improving global out- comes (KDIGO),” *Kidney Int.*, vol. 67, no. 6, pp. 2089–2100, Jun. 2005.
16. M. J. Pencina, R. B. D’Agostino, R. B. D’Agostino, and R. S. Vasan, “Evaluating the added predictive ability of a new marker: From area under the ROC curve to reclassification and beyond,” *Statist. Med.*, vol. 27, no. 2, pp. 157–172, Jan. 2008.
17. X. Liu *et al.*, “Improving precision of glomerular filtration rate estimat- ing model by ensemble learning,” *J. Transl. Med.*, vol. 15, no. 1, p. 231, Dec. 2017.
18. A. D. Rule, T. S. Larson, E. J. Bergstralh, J. M. Slezak, S. J. Jacobsen, and G. F. Cosio, “Using serum creatinine to estimate glomerular filtration rate: Accuracy in good health and in chronic kidney disease,” *Ann. Internal Med.*, vol. 141, no. 12, pp. 929–937, 2004.
19. L. A. Inker *et al.*, “Performance of glomerular filtration rate estimating equations in a community-based sample of blacks and whites: The multiethnic study of atherosclerosis,” *Nephrol. Dialysis Transplantation*, vol. 33, no. 3, pp. 417–425, 2017.
20. K. Murata, N. A. Baumann, A. K. Saenger, T. S. Larson, A. D. Rule, and J. C. Lieske, “Relative performance of the MDRD and CKD-EPI equations for estimating glomerular filtration rate among patients with varied clinical presentations,” *Clin. J. Amer. Soc. Nephrol.*, vol. 6, no. 8, pp. 1963–1972, Aug. 2011.
21. W. M. Michels, D. C. Grootendorst, M. Verduijn, E. G. Elliott,

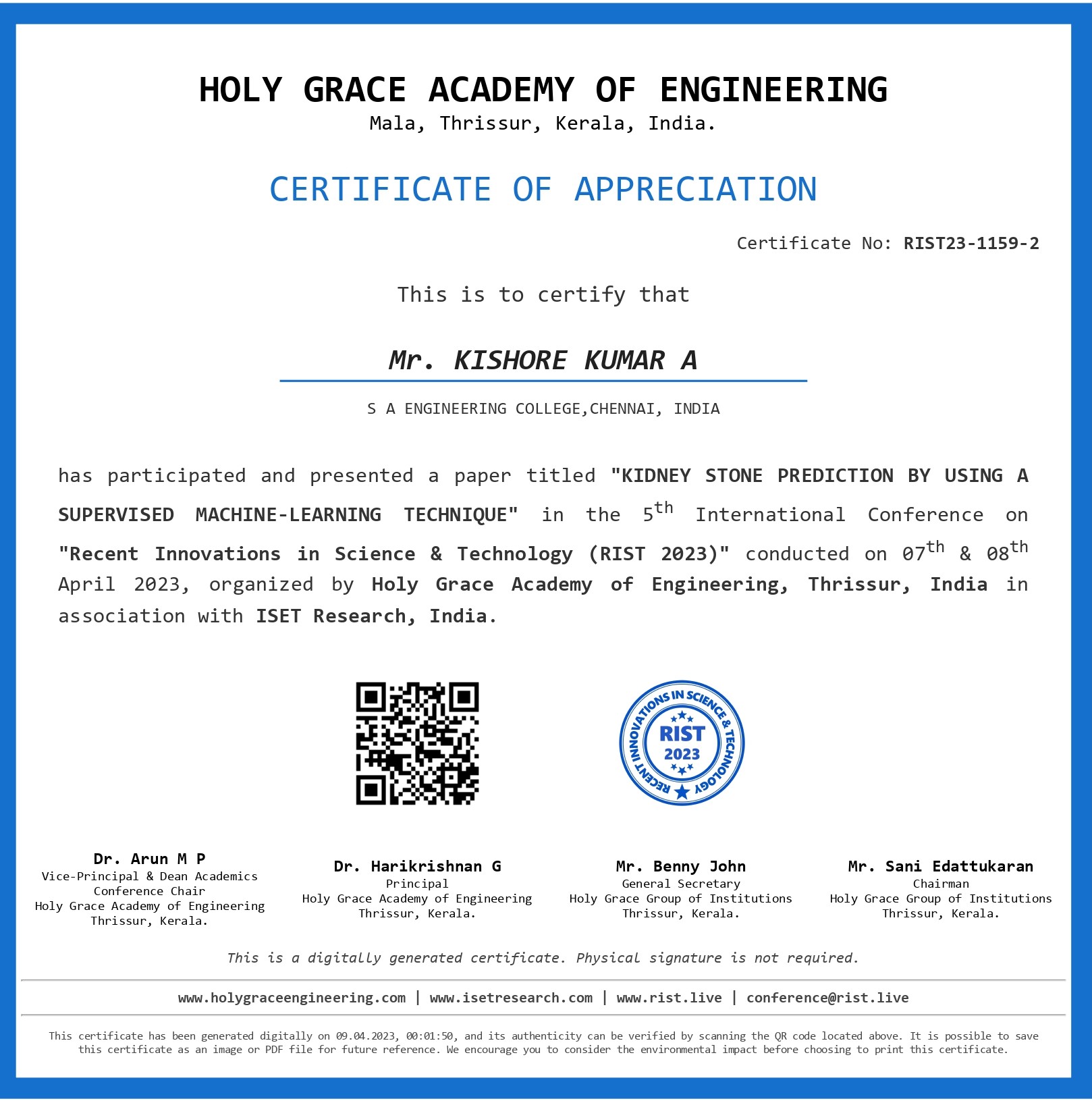
F. W. Dekker, and R. T. Krediet, “Performance of the cockcroft-gault, MDRD, and new CKD-EPI formulas in relation to GFR, age, and body size,” *Clin. J. Amer. Soc. Nephrol.*, vol. 5, no. 6, pp. 1003–1009, Jun. 2010.

1. N. R. Hill *et al.*, “Global prevalence of chronic kidney disease—A systematic review and meta-analysis,” *PLoS ONE*, vol. 11, no. 7, Jul. 2016, Art. no. e0158765.
2. L. A. Stevens, J. Coresh, T. Greene, and A. S. Levey, “Assessing kidney function—Measured and estimated glomerular filtration rate,” *New England J. Med.*, vol. 354, no. 23, pp. 2473–2483, Jun. 2006.
3. A. S. Levey, L. A. Inker, and J. Coresh, “GFR estimation: From physiology to public health,” *Amer. J. Kidney Diseases*, vol. 63, no. 5, pp. 820–834, May 2014.
4. A. S. Levey *et al.*, “A new equation to estimate glomerular filtration rate,” *Ann. Internal Med.*, vol. 150, no. 9, pp. 604–612, 2009.
5. A. S. Levey, J. P. Bosch, J. B. Lewis, T. Greene, N. Rogers, and

D. Roth, “A more accurate method to estimate glomerular filtration rate from serum creatinine: A new prediction equation,” *Ann. Internal Med.*, vol. 130, no. 6, pp. 461–470, 1999.

1. A. Levin *et al.*, “Kidney disease: Improving global outcomes (KDIGO) CKD work group. KDIGO 2012 clinical practice guideline for the evaluation and management of chronic kidney disease,” *Kidney Int. Suppl.*, vol. 3, no. 1, pp. 1–150, Jan. 2013.
2. R. Botev, J.-P. Mallié, J. F. M. Wetzels, C. Couchoud, and O. Schück, “The clinician and estimation of glomerular filtration rate by creatinine- based formulas: Current limitations and quo vadis,” *Clin. J. Amer. Soc. Nephrol.*, vol. 6, no. 4, pp. 937–950, Apr. 2011.
3. X. Liu *et al.*, “Improved glomerular filtration rate estimation by an artificial neural network,” *PLoS ONE*, vol. 8, no. 3, Mar. 2013, Art. no. e58242.
4. T. Song, A. Rodríguez-Patón, P. Zheng, and X. Zeng, “Spiking neural P systems with colored spikes,” *IEEE Trans. Cogn. Devel. Syst.*, vol. 10, no. 4, pp. 1106–1115, Dec. 2017.
5. H. Wang, Z. Cui, Y. Chen, M. Avidan, A. B. Abdallah, and A. Kro- nzer, “Predicting hospital readmission via cost-sensitive deep learning,” *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 15, no. 6, pp. 1968–1978, Dec. 2018.
6. T. Song, X. Zeng, P. Zheng, M. Jiang, and A. Rodríguez-Patón, “A par- allel workflow pattern modeling using spiking neural P systems with colored spikes,” *IEEE Trans. Nanobiosci.*, vol. 17, no. 4, pp. 474–484, Oct. 2018.
7. H. Wang and P. Avillach, “Diagnostic classification and prognostic prediction using common genetic variants in autism spectrum disorder: Genotype-based deep learning,” *JMIR Med. Informat.*, vol. 9, no. 4, Apr. 2021, Art. no. e24754.
8. A. Esteva *et al.*, “Dermatologist-level classification of skin cancer with deep neural networks,” *Nature*, vol. 542, no. 7639, p. 115, 2017.
9. H. Wang, L. Li, L. Chi, and Z. Zhao, “Autism screening using deep embedding representation,” in *Proc. Int. Conf. Comput. Sci.* Springer, 2019, pp. 160–173.
10. T. Song, P. Zheng, M. L. D. Wong, M. Jiang, and X. Zeng, “On the computational power of asynchronous axon membrane systems,” *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 4, no. 5, pp. 696–704, Oct. 2020.

****

****