**CLASSIFYING KIDNEY DISEASE WITH AI REVOLUTIONARY APROACH TO EARLY DETECTION**

**PREDICTION OF RENEWABLE ENERGY USING MACHINE LEARNING TECHNIQUES.**

**TABLE OF CONTENT**

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

**LIST OF FIGURES**

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

**LIST OF SYMBOLS**

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

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

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

1. **Abstract:**

Early prediction of kidney disease is crucial for effective treatment and management. This project focuses on leveraging Artificial Intelligence (AI) to classify and detect kidney diseases at an early stage. By integrating data preprocessing, visualization, and machine learning algorithms with the Django framework, the system aims to enhance diagnostic accuracy and healthcare workflows. The project involves preprocessing patient data to ensure quality and consistency, followed by visualizing the data to uncover patterns and insights. Advanced machine learning algorithms, such as employed to classify kidney diseases based on medical imaging and patient records. The Django framework serves as the backbone for developing a user-friendly interface, allowing healthcare professionals to interact with the system efficiently. This approach not only improves early detection but also supports informed decision-making in patient care.

**Keywords:** Kidney Disease Classification, Artificial Intelligence, Early Detection, Data Preprocessing, Data Visualization, Machine Learning Algorithms, Django Framework, Healthcare Analytics.

1. **Existing System:**

Histotripsy is a focused ultrasound therapy that ablates tissue via the action of bubble clouds. It is under investigation to treat a number of ailments, including renal tumors. Ultrasound imaging is used to monitor histotripsy, though there remains a lack of definitive imaging metrics to confirm successful treatment outcomes. In this study, a convolutional neural network (CNN) was developed to segment ablation on ultrasound images. Methods: A transfer learning approach was used to replace classification layers of the residual network ResNet-18. Inputs to the classification layers were based on ultrasound images of ablated red blood cell phantoms. Digital photographs served as the ground truth. The efficacy of the CNN was compared to subtraction imaging, and manual segmentation of images by two board-certified radiologists. Results: The CNN had a similar performance to manual segmentation, though was improved relative to segmentation with subtraction imaging. Predictions of the network improved over the course of treatment, with the Dice similarity coefficient less than 20% for fewer than 500 applied pulses, but 85% for more than 750 applied pulses. The network was also applied to ultrasound images of ex vivo kidney exposed to histotripsy, which indicated a morphological shift in the treatment profile relative to the phantoms. These findings were consistent with histology that confirmed ablation of the targeted tissue. Conclusion: Overall, the CNN showed promise as a rapid means to assess outcomes of histotripsy and automate treatment. Significance: Data collected in this study indicate integration of CNN image segmentation to gauge outcomes for histotripsy ablation holds promise for automating treatment procedures.

**DISADVANTAGES:**

 **Limited Generalization**: The CNN was trained on ablated red blood cell phantoms, which may not fully represent the variability in real renal tumors, potentially limiting the model's generalizability.

 **Dependency on Image Quality**: The effectiveness of the CNN might be compromised by variations in ultrasound image quality or artifacts, impacting its accuracy in different clinical settings.

 **Interpretation of Morphological Shifts**: While the CNN showed a morphological shift in the treatment profile, interpreting these changes might require additional validation and could be subject to inconsistencies between phantoms and actual tissue.

1. **INTRODUCTION**

Classifying kidney disease using AI represents a revolutionary approach to early detection, significantly enhancing diagnostic accuracy and patient outcomes. Machine learning and deep learning, particularly Convolutional Neural Networks (CNN), are at the forefront of this innovation. By analyzing medical images, lab results, and patient data, these advanced algorithms can detect subtle patterns and biomarkers that may be overlooked by traditional methods. This technology enables early identification of chronic kidney diseases (CKD), cysts, stones, and other conditions, offering timely intervention to prevent further complications. With AI-powered systems, healthcare providers can improve diagnostic precision, reduce human error, and deliver personalized treatment plans that can potentially save lives.

**3.1 Data Science**

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

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

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

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

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

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

**3.2ARTIFICIAL 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.

**Natural Language Processing (NLP):**

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

1. **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.

Analyses Predicts

Machine Learning

Past Dataset

Trains

Fig: 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”.

1. **Proposed System:**

To revolutionize the early prediction of kidney disease using AI, the proposed system will integrate several key components. First, it will implement advanced preprocessing techniques to clean and normalize medical data, ensuring high-quality input for machine learning algorithms. This will be followed by sophisticated data visualization tools to provide intuitive insights into patient data and disease patterns. Cutting-edge machine learning algorithms, ensemble methods, will be employed to enhance the accuracy and efficiency of disease classification. Finally, the Django framework will serve as the backbone for the system, facilitating integration of the preprocessing, visualization, and algorithmic components through a user-friendly web interface that enables predictions and interactive data exploration.

**ADVANTAGES:**

 **Enhanced Accuracy**: Advanced preprocessing and ensemble machine learning algorithms improve the precision of disease classification, leading to more reliable early predictions.

 **Intuitive Insights**: Sophisticated data visualization tools make it easier for healthcare professionals to understand patient data and disease patterns, facilitating better decision-making.

 **Seamless Integration**: The Django framework provides a user-friendly web interface, integrating all components efficiently and allowing for interactive data exploration and predictions.

Kidney dataset

Data Processing

Test dataset

Training dataset

Classification ML Algorithm

Model

1. **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. Loan default trends have been long studied from a socio-economic stand point. Most economics surveys believe in empirical modeling of these complex systems in order to be able to predict the loan default rate for a particular individual. The use of machine learning for such tasks is a trend which it is observing now. Some of the survey’s to understand the past and present perspective of loan approval or not.

**Review of Literature Survey**

**Title :** A study of the visualization of artificial intelligence applications in chronic kidney disease in the literature over the last 20 years .

**Author:** Yudi Li, Ying Ding, Yan Xu , Haoji Meng , Hongji Wu, Donglin Li, Yibo Hu.

**Year** : 2024

Chronic kidney disease (CKD) is a global public health problem characterized by persistent kidney damage or loss of kidney function. Previously, the diagnosis of CKD has mainly relied on serum creatinine and estimation of the glomerular filtration rate. However, with the development and progress of artificial intelligence (AI), AI has played different roles in various fields, such as early diagnosis, progression prediction, prediction of associated risk factors, and drug safety and efficacy evaluation. Therefore, research related to the application of AI in the field of CKD has become a hot topic at present. Therefore, this study adopts a bibliometric approach to study and analyze the development and evolution patterns and research hotspots of AI-CKD. English publications related to the field between January 1, 2004, and June 27, 2024, were extracted from the Web of Science Core Collection database. The research hotspots and trends of AI-CKD were analyzed at multiple levels, including publication trends, authors, institutions, countries, references and keywords, using VOSviewer and CiteSpace. The results showed that a total of 203 publications on AI-CKD were included in the study, of which Barbieri Carlo from the University of Milan, Italy, had the highest number of publications (NP=5) and had a high academic impact (H-Index=5), while the USA and its institution, the Mayo Clinic, were the publications. The USA and its Mayo Clinic are the countries and institutions with the highest number of publications, and China is the country with the second highest number of publications, with three institutions attributed to China among the top five institutions. Germany's institution, Fresenius Medical Care, has the highest academic impact (H-index=6). Keyword analysis yielded artificial intelligence, chronic kidney disease, machine learning, prediction model, risk, deep learning, and other keywords with high frequency, and cluster analysis based on the timeline yielded a total of 8 machine learning, deep learning, retinal microvascular abnormality, renal failure, Bayesian network, anemia, bone disease, and allograft nephropathology clusters. This study provides a comprehensive overview of the current state of research and global frontiers of AI-CKD through bibliometric analysis. These findings can provide a valuable reference and guidance for researchers.

**Title :** Chronic Kidney Disease Prediction Using Machine Learning.

**Author:** hrinidhI.

**Year** : 2023

Millions of people worldwide are afflicted with the degenerative condition known as chronic kidney disease (CKD). Improved patient outcomes and the prevention of future renal damage are possible with early detection of CKD. In this work, CKD was predicted using patient data using machine learning (ML) methods. Data from patients with CKD were included in the dataset for this investigation, including demographic, clinical, and laboratory information. In order to predict CKD, four ML algorithms—Naive Bayes, Decision Trees, K-Nearest Neighbours (KNNs), and Neural Networks—were applied. The study's findings demonstrated that all four machine learning algorithms were highly accurate at predicting CKD. With an AUC of 0.92, Neural Networks had the highest accuracy, followed by Decision Trees, KNNs, Naive Bayes, and KNNs with AUCs of 0.87, 0.85, 0.85, and 0.85, respectively. The study underscores the value of early detection for better patient outcomes and shows the potential of ML algorithms to predict CKD.

**Title:** Chronic Kidney Disease Prediction Using Machine Learning.

**Author:** Chamandeep Kaur , M. Sunil Kumar , Afsana Anjum , M. B. Binda , Maheswara Reddy Mallu .

**Year** : 2023

The occurrence of Chronic Renal Disease (CRD), is also referred to as Chronic Kidney Disease (CKD). It depicts a medical condition that harms the kidneys and has an impact on a person’s overall health. End-stage renal disease and the patient’s eventual mortality can result from improper disease diagnosis and treatment. In the field of medical science, Machine Learning (ML) techniques have become a valuable tool and play a significant role in disease prediction. The development and validation of a predictive model for the prognosis of chronic renal disease is the aim of the proposed study. A dataset on chronic kidney disease with 400 samples was taken from the UCI Machine Learning Repository. Three machine learning classifiers— Logistic Regression (LR), Decision Tree (DT), and Support Vector Machine (SVM)—were used for analysis, and the bagging ensemble method was used to enhance the model’s performance. The machine learning classifiers were trained using the clusters of the dataset for chronic renal disease. The Kidney Disease Collection is then compiled using nonlinear features and categories. The decision tree produces the best results, with an accuracy of 95%. Finally, we achieve the greatest accuracy of 97% by using the bagging ensemble approach.

**Title:** Predicting Chronic Kidney Disease using ML algorithms and XAI

**Author:** Sudip Raj Khadka, Saphal Subedi, Bikash Kumar Aidy, Lok Nath Regmi, Ashish Parajuli and Mohan Bhandari.

**Year** : 2023

Kidney disease is a significant health concern that is currently affecting individuals of all age groups. To predict the occurrence of chronic renal disease, a large number of scholars have employed machine learning and deep learning techniques. However, the efficacy of these methods is often hampered by a lack of transparency, which is a major issue in the application of artificial intelligence in healthcare and medical analysis. As such, the lack of clarity has prompted concern. To interpret the results of predictive models, the present study proposes the deployment of four machine learning algorithms, including Decision Tree, Logistic Regression, Multi-layer Perceptron Classifier, and Support Vector Machine, in combination with explainable AI (XAI) interface, leveraging the local interpretable model-agnostic explanation (LIME) and shapely additive explanation shapely additive values (SHAP). The proposed models are intended to facilitate effective decision-making in clinical research and therapeutic practices

**Title:** Role of Artificial Intelligence in Kidney Disease.

**Author:** Qiongjing Yuan, Haixia Zhang, Tianci Deng, Shumei Tang, Xiangning Yuan, Wenbin Tang.

**Year** : 2020

Artificial intelligence (AI), as an advanced science technology, has been widely used in medical fields to promote medical development, mainly applied to early detections, disease diagnoses, and management. Owing to the huge number of patients, kidney disease remains a global health problem. Challenges remain in its diagnosis and treatment. AI could take individual conditions into account, produce suitable decisions and promise to make great strides in kidney disease management. Here, we review the current studies of AI applications in kidney disease in alerting systems, diagnostic assistance, guiding treatment and evaluating prognosis. Although the number of studies related to AI applications in kidney disease is small, the potential of AI in the management of kidney disease is well recognized by clinicians; AI will greatly enhance clinicians’ capacity in their clinical practice in the future.

1. **SYSTEM STUDY**

**8.1 Overview of the system**

The "Classifying Kidney Disease with AI: Revolutionary Approach to Early Detection" system aims to transform kidney disease diagnosis through advanced artificial intelligence techniques. This innovative system utilizes deep learning algorithms, particularly Convolutional Neural Networks (CNNs), to analyze medical imaging data such as ultrasounds, CT scans, and MRIs. By integrating these AI models with comprehensive clinical data—including patient demographics, lab results, and medical history—the system enables precise early detection of kidney abnormalities. The AI-driven approach enhances diagnostic accuracy and efficiency, allowing for timely intervention and personalized treatment plans. This system not only improves patient outcomes by catching diseases at their earliest stages but also streamlines clinical workflows, offering a cutting-edge solution in the realm of kidney health management.

**Aim**

The aim is to develop an advanced AI system for early kidney disease detection by leveraging cutting-edge machine learning techniques. This approach seeks to enhance diagnostic accuracy and efficiency by analyzing medical images and clinical data. The goal is to enable timely and precise identification of kidney abnormalities, improving patient outcomes and facilitating proactive healthcare interventions.

#### **8.3 Objectives**

Develop an AI-driven system for the early and accurate classification of kidney diseases using advanced machine learning techniques. Enhance diagnostic precision by integrating medical imaging and clinical data for comprehensive disease analysis. Implement real-time analysis capabilities to facilitate timely intervention and personalized treatment plans. Ensure the system is scalable and adaptable to various healthcare environments, improving patient outcomes and healthcare efficiency..

**8.4 Problem Description/ Problem Statements**

Despite advances in medical technology, the early detection and accurate classification of kidney diseases remain significant challenges in healthcare. Traditional diagnostic methods often rely on subjective interpretations of medical images and clinical data, which can lead to delayed diagnoses and suboptimal patient outcomes. There is a pressing need for an innovative approach that leverages artificial intelligence (AI) to enhance the precision and timeliness of kidney disease detection. By employing AI-driven techniques, such as Convolutional Neural Networks (CNNs) and advanced data analytics, this project aims to develop a revolutionary system that can automatically analyze medical images and clinical records to detect kidney abnormalities at an early stage. The goal is to improve diagnostic accuracy, streamline the detection process, and ultimately lead to better management and treatment of kidney diseases, thereby addressing a critical gap in current healthcare practices.

**8.5 Scope**

The project on "Classifying Kidney Disease with AI: Revolutionary Approach to Early Detection" aims to harness the power of artificial intelligence to transform the early detection and classification of kidney diseases. By integrating advanced AI techniques such as Convolutional Neural Networks (CNNs) for medical imaging and machine learning algorithms for clinical and genetic data analysis, this project seeks to create a comprehensive and highly accurate diagnostic tool. The scope includes developing a robust AI model capable of analyzing various forms of medical imaging—such as ultrasound, CT scans, and MRIs—alongside clinical data to identify and classify kidney conditions at an early stage. The project will involve the collection and preprocessing of diverse datasets, training and validating the model, and deploying it within clinical settings to assist healthcare professionals in making more informed decisions. Additionally, the project addresses ethical considerations, including data privacy and fairness, ensuring that the AI system is both effective and equitable. Ultimately, this innovative approach aims to significantly enhance early diagnosis, improve patient outcomes, and set new standards in kidney disease management.

1. **Feasibility study:**

## **Data Wrangling**

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

**Data collection**

The data set collected for predicting given data is split into Training set and Test set. Generally, 7:3 ratios are applied to split the Training set and Test set. The Data Model which was created using machine learning algorithms are applied on the Training set and based on the test result accuracy, Test set prediction is done.

**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 Gathering

Data Pre-Processing

Choose model

Train model

Test model

Tune model

Prediction

Process of dataflow diagram

1. **List of Modules:**

* Data Pre-processing
* Data Analysis of Visualization
* Implementing GRB Algorithm
* Implementing MN Architecture
* MobileNet Architecture
* Deployment Using Django

1. **Environmental Requirements:**

**1. Software Requirements:**

Operating System : Windows

Tool : Anaconda with Jupyter Notebook

**2. Hardware requirements:**

Processor : Pentium IV/III

Hard disk : minimum 80 GB

RAM : minimum 2 GB

1. **SOFTWARE DESCRIPTION**

The software for classifying kidney disease using an AI-driven approach represents a groundbreaking advancement in early medical diagnostics. This innovative solution leverages cutting-edge artificial intelligence techniques, particularly deep learning algorithms, to analyze medical imaging and clinical data with exceptional precision. By integrating Convolutional Neural Networks (CNNs) for image analysis and advanced machine learning models for clinical data, the software can accurately identify and classify various stages of kidney disease from images such as ultrasound, CT scans, or MRIs. It processes vast amounts of data, including patient demographics, lab results, and medical history, to provide early and reliable detection of kidney abnormalities. The system's ability to continuously learn and adapt from new data ensures that it remains at the forefront of diagnostic accuracy. With an intuitive user interface designed for healthcare professionals, this software enhances decision-making, streamlines diagnostic workflows, and ultimately contributes to better patient outcomes through timely and precise interventions.

**13.1 ANACONDA NAVIGATOR**

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

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

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

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

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

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

**The following applications are available by default in Navigator:**

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

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

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

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

**Conda:**

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

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

**13.2 JUPYTER NOTEBOOK**

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

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

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**

1. **PYTHON**

**Introduction:**

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

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

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

Python consistently ranks as one of the most popular programming languages

**History:**

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

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

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

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

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

**Methods :**

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

### **Typing :**

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

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

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

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

**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.

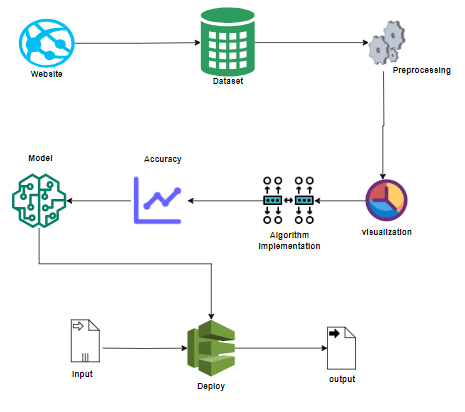
When you are applying machine learning to your own datasets, you are working on a project. A machine learning project may not be linear, but it has a number of well-known steps:

* 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.
7. **SYSTEM ARCHITECTURE:**



1. **Work flow diagram**

Source Data

Data Processing and Cleaning

Testing Dataset

Training Dataset

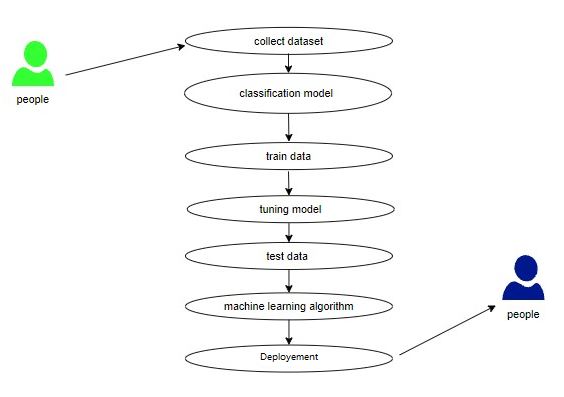
Best Model by Accuracy

Classification ML Algorithms

Finding kidney disease

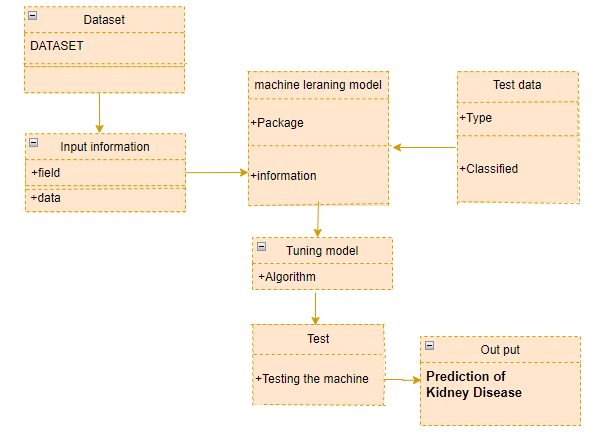
Workflow Diagram

1. **Use Case Diagram**



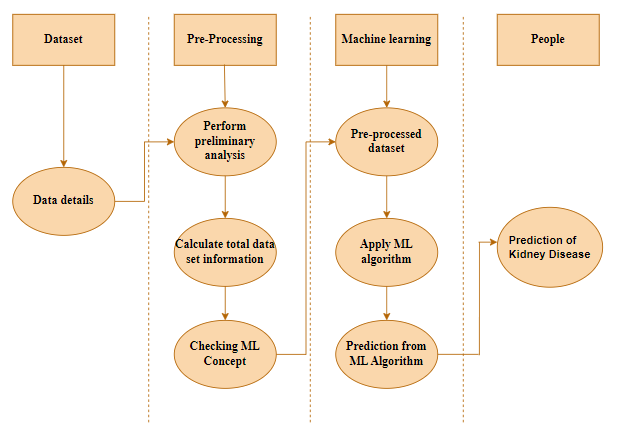
Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases. So, it can say that uses cases are nothing but the system functionalities written in an organized manner.

1. **Class Diagram:**



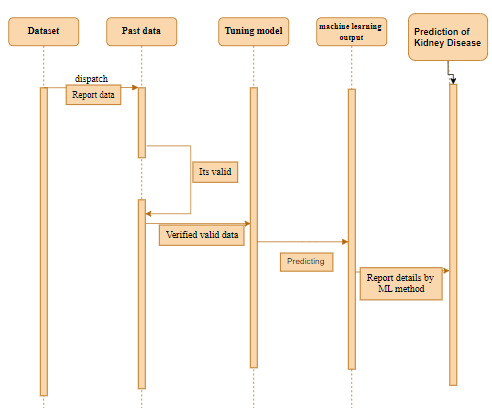
Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of class diagrams represent the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element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

1. **Activity Diagram**



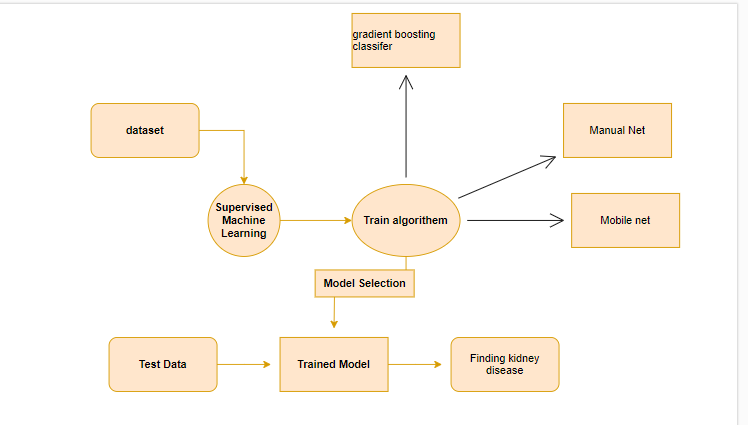
Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part. 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.

1. **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.

1. **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.

1. **Module description:**

**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.

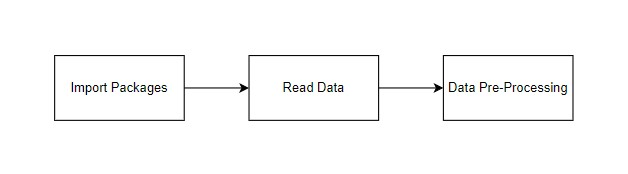
Variable identification with Uni-variate, Bi-variate and Multi-variate analysis:

* 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

**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.

**MODULE DIAGRAM**



GIVEN INPUT EXPECTED OUTPUT

input : data

output : removing noisy data

**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.

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.

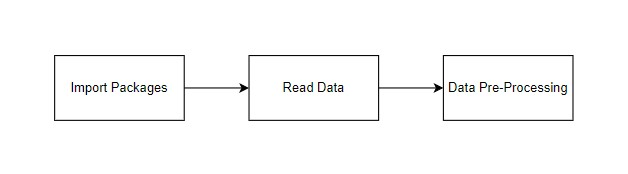
**False Positives (FP):** A person who will pay predicted as defaulter. When actual class is no and predicted class is yes. E.g. if actual class says this passenger did not survive but predicted class tells you that this passenger will survive.

**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.

**MODULE DIAGRAM**



**GIVEN INPUT EXPECTED OUTPUT**

input : data

output : visualized data

**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 algorithms are:

* GRBoosting
* MN Architecture
* MobileNet Architechture

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.

**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.

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. (When the model predicts default: how often is 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)

**ALGORITHM AND TECHNIQUES**

**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.

**GRBoosting Classifier:**

Machine learning is one of the most popular technologies to build predictive models for various complex regression and classification tasks. **Gradient Boosting Machine** (GBM) is considered one of the most powerful boosting algorithms.

Although, there are so many algorithms used in machine learning, boosting algorithms has become mainstream in the machine learning community across the world. Boosting technique follows the concept of ensemble learning, and hence it combines multiple simple models (weak learners or base estimators) to generate the final output. GBM is also used as an ensemble method in machine learning which converts the weak learners into strong learners. In this topic, **"GBM in Machine Learning"** we will discuss gradient machine learning algorithms, various boosting algorithms in machine learning, the history of GBM, how it works, various terminologies used in GBM, etc. But before starting, first, understand the boosting concept and various boosting algorithms in machine learning.

**1. Ensemble Learning:**

* Gradient Boosting is an ensemble learning technique where multiple weak learners (typically decision trees) are combined to create a strong learner.

**2. Weak Learners (Decision Trees):**

* The base learners, often decision trees, are called weak learners because they perform slightly better than random chance.

**3. Boosting Concept:**

* Gradient Boosting builds trees sequentially, each one correcting errors made by the previous trees.
* At each stage, a new tree is trained on the residuals (the differences between the actual values and the predictions made by the existing ensemble).

**4. Objective Function:**

* The algorithm minimizes a predefined loss function, which measures the difference between the predicted values and the actual values.
* Common loss functions include mean squared error for regression problems and deviance (logistic loss) for classification problems.

**5. Gradient Descent:**

* The "Gradient" in Gradient Boosting refers to the gradient descent optimization used to minimize the loss function.
* At each iteration, the algorithm calculates the negative gradient of the loss function with respect to the current ensemble's predictions.

**6. Learning Rate:**

* A learning rate parameter controls the step size during the gradient descent optimization.
* A lower learning rate makes the algorithm more robust, but it requires more iterations. A higher learning rate speeds up convergence but may lead to overshooting.

**7. Trees and Weak Learners:**

* Trees are typically shallow, and each new tree addresses the errors of the combined ensemble.
* Shallow trees reduce overfitting and contribute to the model's interpretability.

**8. Regularization:**

* Gradient Boosting includes regularization techniques, such as tree pruning and feature subsampling, to prevent overfitting.

**9. Prediction:**

* The final prediction is the sum of the predictions from all the weak learners, each multiplied by its associated learning rate.

**10. Libraries Implementation:**

* Popular libraries that implement Gradient Boosting include Scikit-learn (with GradientBoostingClassifier for classification tasks) and XGBoost, LightGBM, and CatBoost, which are specialized libraries optimized for performance.

**Advantages:**

* Good predictive performance.
* Handles both numerical and categorical data.
* Can capture complex relationships in the data.

**Challenges:**

* May require tuning of hyperparameters.
* Prone to overfitting if hyperparameters are not properly set.

Gradient Boosting has proven to be highly effective in various machine learning tasks and is widely used in practice. Its ability to build strong models by sequentially improving weak learners makes it a valuable tool in the data scientist's toolbox.

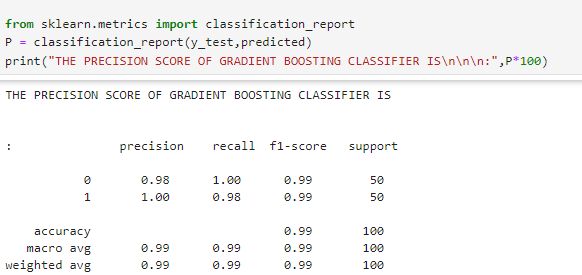
**MODULE DIAGRAM**

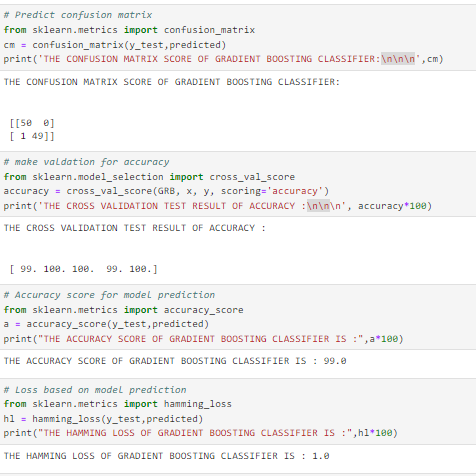


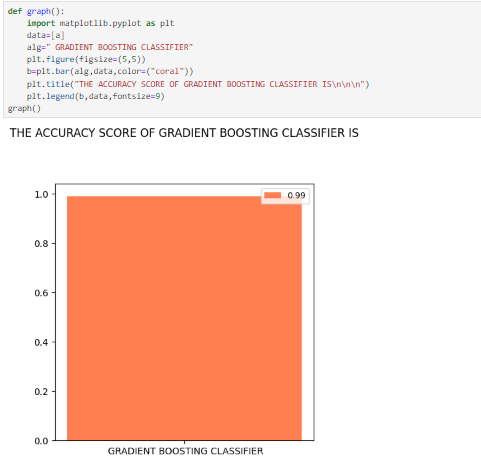
**GIVEN INPUT EXPECTED OUTPUT**

input : data

output : getting accuracy





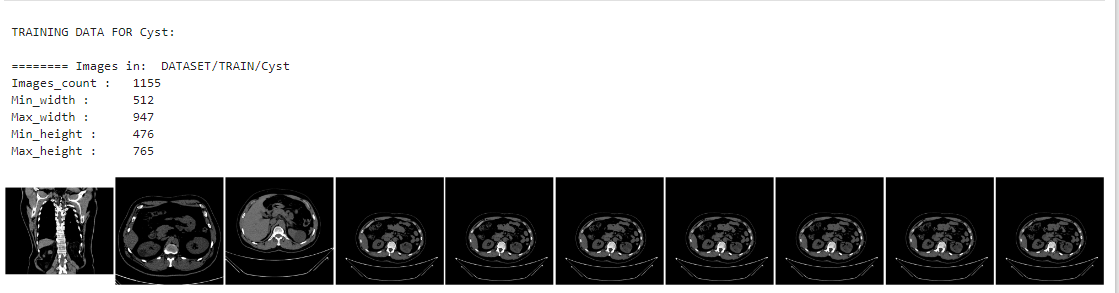


**MANUAL ARCHITECHTURE:**

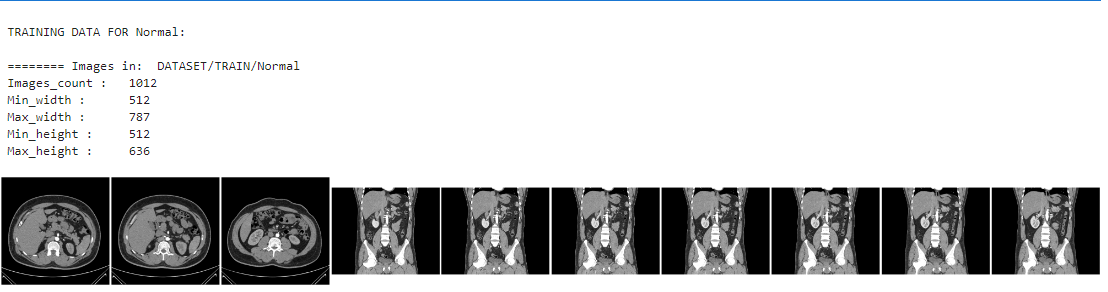
Manual architecture definition in deep learning refers to the process of designing neural network models by explicitly specifying the structure and parameters of the network. This involves selecting the number of layers, types of layers (e.g., convolutional, recurrent, fully connected), and their configurations such as kernel sizes, activation functions, and layer dimensions. Unlike automated approaches, where algorithms optimize the architecture based on performance metrics, manual architecture definition requires a deep understanding of the problem domain and the network's potential behavior. It necessitates iterative experimentation and fine-tuning to balance model complexity and computational efficiency, aiming to create a tailored architecture that maximizes performance for specific tasks. While this approach offers precise control over the model design, it can be time-consuming and demands significant expertise in both the application area and deep learning principles.

**CLASSIFICATIONS:**

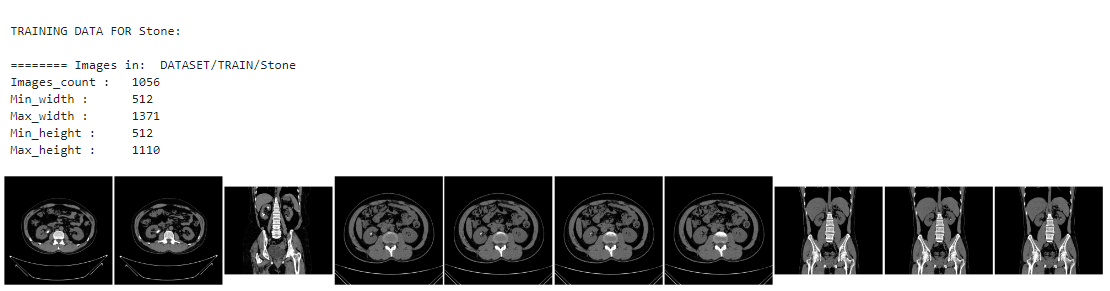
**CYST:**



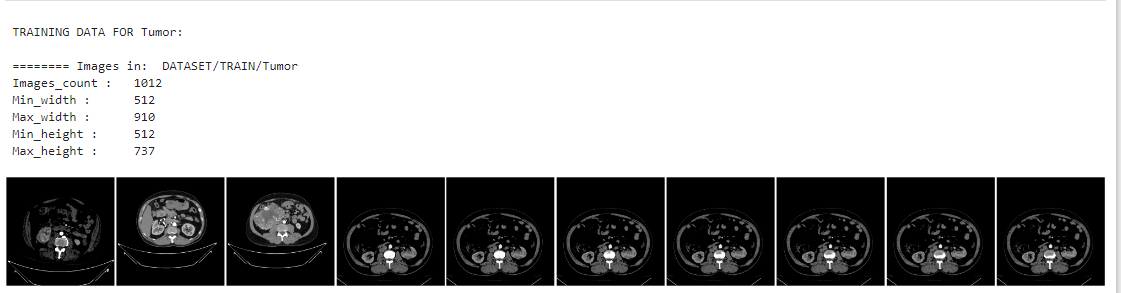
**NORMAL:**

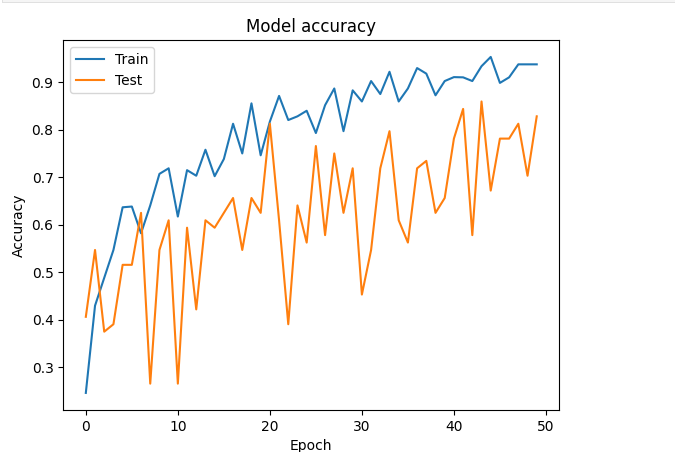


**STONE:**

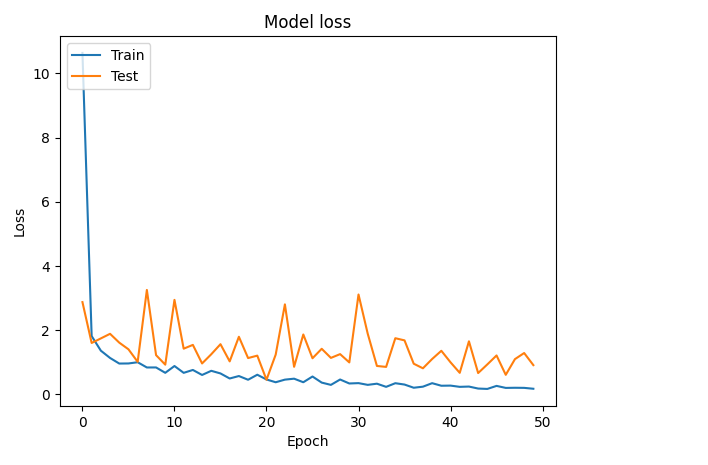


**TUMOR:**





E.g. Model Accuracy



E.g. Model Accuracy

**MOBILENET Architecture:**

MobileNet is a family of deep learning architectures designed for efficient on-device image classification and computer vision tasks. It was developed by Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam from Google in 2017.

The key motivation behind MobileNet is to create neural network architectures that are computationally lightweight, allowing them to run efficiently on mobile devices with limited resources (e.g., smartphones) and embedded systems. The focus is on achieving a good trade-off between model accuracy and model size or computational complexity.

MobileNet achieves this efficiency through the use of two main techniques:

Depthwise Separable Convolutions: Similar to the Xception architecture, MobileNet utilizes depthwise separable convolutions. This factorizes a standard convolution into two separate operations: depthwise convolution and pointwise convolution.

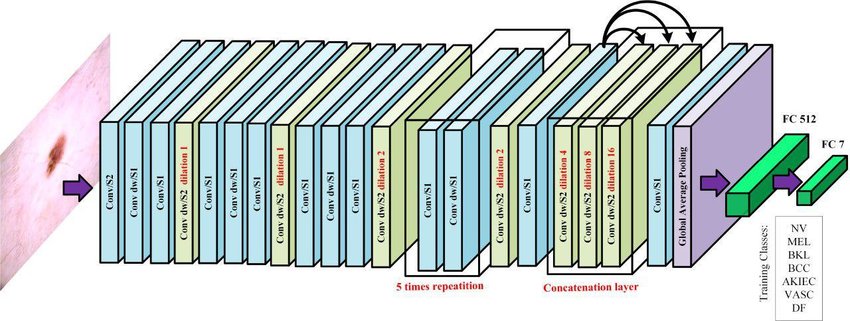
Depthwise Convolution: In this step, each channel of the input feature map is convolved with its own set of filters independently, just like in Xception. This reduces the computational cost significantly as it requires fewer operations compared to a standard convolution.

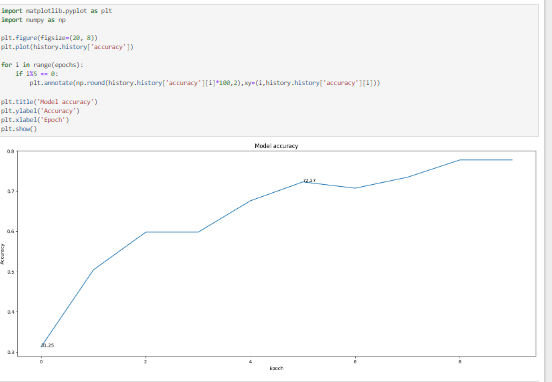
Pointwise Convolution: Following the depthwise convolution, a 1x1 pointwise convolution is applied to combine the output channels from the depthwise convolution. This helps capture cross-channel interactions and create the final output feature map.

Width Multiplier: MobileNet introduces a hyperparameter called the "width multiplier" (usually denoted by the Greek letter "α"). The width multiplier scales the number of channels in each layer of the network. By reducing the number of channels in each layer, the model becomes more compact and computationally efficient. The width multiplier allows for easy control over the model's size and computational cost, trading off accuracy for efficiency.

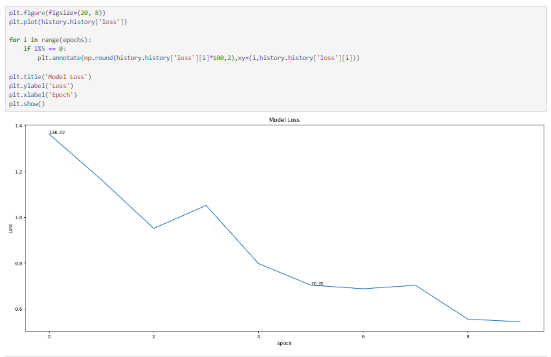
MobileNet comes in different versions, denoted as MobileNetV1, MobileNetV2, and MobileNetV3, each introducing improvements over the previous version. MobileNetV2, for example, introduced inverted residual blocks and linear bottlenecks to improve performance. MobileNetV3 further optimized the architecture with the use of h-swish activation and other refinements.

MobileNet has been widely adopted in various applications, such as real-time image classification on mobile devices, object detection, and semantic segmentation. Its efficiency and effectiveness have made it a popular choice for deploying deep learning models on resource-constrained devices and platforms.

****



e.g. Model Accuracy



e.g accuracy loss

**Django:**

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Ridiculously fast.

Django was designed to help developers take applications from concept to completion as quickly as possible.

Reassuringly secure.

Django takes security seriously and helps developers avoid many common security mistakes.

Exceedingly scalable.

Some of the busiest sites on the web leverage Django’s ability to quickly and flexibly scale.

With Django, you can take web applications from concept to launch in a matter of hours. Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Fully loaded.

Django includes dozens of extras you can use to handle common web development tasks. Django takes care of user authentication, content administration, site maps, RSS feeds, and many more tasks — right out of the box.

Reassuringly secure.

Django takes security seriously and helps developers avoid many common security mistakes, such as SQL injection, cross-site scripting, cross-site request forgery and clickjacking. Its user authentication system provides a secure way to manage user accounts and passwords.

Exceedingly scalable.

Some of the busiest sites on the planet use Django’s ability to quickly and flexibly scale to meet the heaviest traffic demands.

Incredibly versatile.

Companies, organizations and governments have used Django to build all sorts of things — from content management systems to social networks to scientific computing platforms.

1. **Model-View-Template (MVT) Architecture:**

* Django follows a variation of the traditional Model-View-Controller (MVC) architecture known as Model-View-Template (MVT).
* **Model:** Represents the data structure and business logic, defining how data is stored and retrieved.
* **View:** Handles user interface and presentation logic, processing user requests and returning appropriate responses.
* **Template:** Manages the generation of HTML dynamically, separating the presentation layer from the business logic.

2. **Object-Relational Mapping (ORM):**

* Django features a powerful Object-Relational Mapping (ORM) system that enables developers to interact with databases using Python objects instead of raw SQL queries.
* Models in Django define the data structure, including fields, relationships, and constraints.

3. **URL Routing:**

* Django uses a URL dispatcher to map URLs to views. This is done through a urls.py file, which contains patterns or regular expressions to route requests to the appropriate views.

4. **Views and Controllers:**

* In Django, views are responsible for processing user requests and returning appropriate responses. They contain the business logic of the application.
* While Django follows the MVT pattern, views often take on the responsibilities of both controllers and views in the traditional MVC pattern.

5. **Templates and Presentation Logic:**

* Templates in Django are responsible for generating dynamic HTML. They use a simple syntax to insert variables, control structures, and template tags.
* Templates separate the presentation logic from the business logic, promoting a clean and modular design.

6. **Middleware:**

* Django middleware is a way to process requests globally before they reach the view or after the view has processed the request.
* Middleware components can perform tasks such as authentication, logging, or modifying the response before it is sent to the client.

7. **Forms and User Input Handling:**

* Django provides a robust form handling system for managing user input. Forms can be used to validate and process data submitted by users.
* Forms simplify the process of handling HTML forms and help in maintaining a secure and consistent approach to data validation.

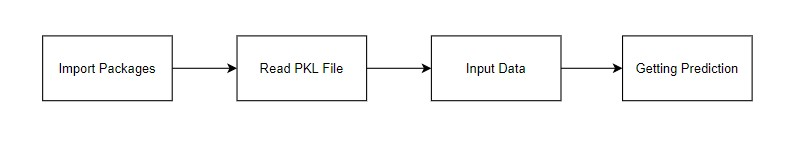
8. **Admin Interface:**

* Django includes a built-in admin interface that allows developers to manage application data through a web-based interface.
* The admin interface is automatically generated based on the models defined in the application.

9. **Static Files and Media:**

* Django provides mechanisms for handling static files (e.g., CSS, JavaScript, images) and media files (e.g., user uploads).
* The {% static %} template tag is used to reference static files, while the MEDIA\_ROOT and MEDIA\_URL settings manage media files.

**MODULE DIAGRAM**



**GIVEN INPUT EXPECTED OUTPUT**

input : data values

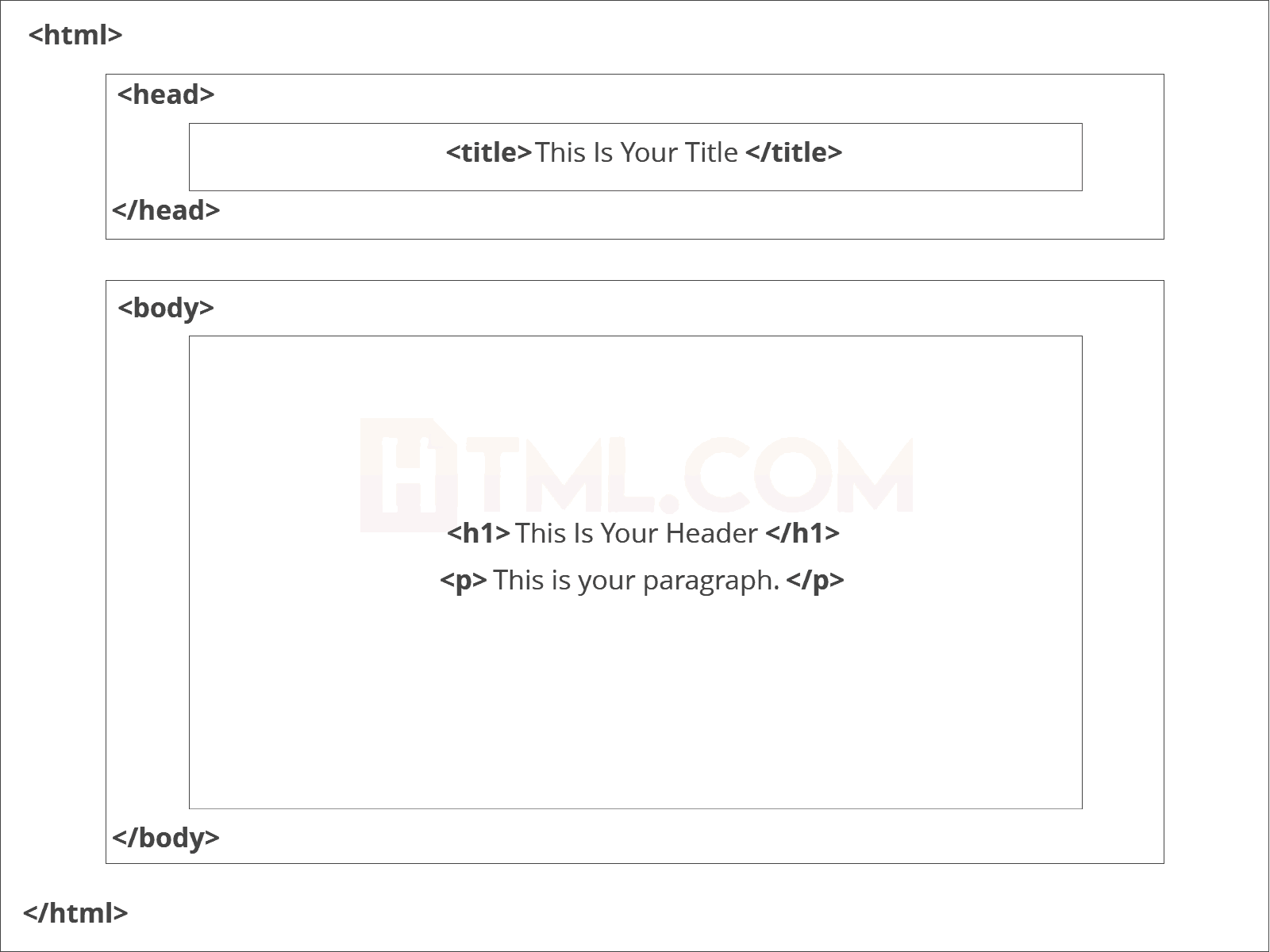
output : predicting output

**24. HTML Introduction**

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

#### Basic Construction of an HTML Page

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



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

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

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

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

#### Further Tags

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

<title>

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

<meta>

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

**Head Tag**  
<head>

<title>My First Webpage</title>

<meta charset="UTF-8">

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

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

</header>

### Adding Content

Next, we will make<body> tag.

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

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

#### Add HTML Headings To Web Page

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

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

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

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

##### Creating Your Heading

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

<h1> Welcome To My Page </h1>

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

**Add Text In HTML**

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

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

#### Add Links In HTML

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

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

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

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

**Image Tag**

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

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

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

**25. CSS**

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

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

## **CSS Syntax**

Selector {

Property 1 : value;

Property 2 : value;

Property 3 : value;

}

For example:

h1

{

Color: red;

Text-align: center;

}

#unique

{

color: green;

}

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

## **CSS Comment**

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

## **CSS How-To**

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

**Inline CSS**

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

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

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

**Internal CSS**

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

<style>

H1{

Color:red;

}

</style>

<h3> Have a great day </h3>

**External CSS**

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

<head>

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

</head>

h1{

color:red; //.css file

}

## **CSS Selectors**

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

## **CSS Colors**

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

**CSS Background**

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

## **CSS BoxModel**

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

**CODING:**

**MODULES 1**

## DATA PREPROCESSING AND DATA CLEANING

import pandas as pd

import numpy as np

import warnings

warnings.filterwarnings('ignore')

# Read csv file

df = pd.read\_csv('KIDNEY.csv')

df.head()

df.tail()

df.shape

df.size

df.columns

df.isnull()

df = df.dropna()

df['Rbc'].unique()

df.describe()

df.corr()

df.info()

pd.crosstab(df["Rbc"], df["Sod"])

df.groupby(["Rbc","Sod"]).groups

df["Rbc"].value\_counts()

pd.Categorical(df["Rbc"]).describe()

df.duplicated()

sum(df.duplicated())

df=df.drop\_duplicates()

sum(df.duplicated())

**MODULE 2:**

## DATA VISUALIZATION AND DATA ANALYSIS

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

# Read csv file

df = pd.read\_csv('KIDNEY.csv')

df.head()

df.columns

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

sns.countplot(x='Class',data=df)

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

plt.subplot(1,2,1)

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

plt.subplot(1,2,2)

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

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

plt.show()

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

sns.ecdfplot(df['Sc'], color='brown') # scatter, plot, triplot, stackplot

sns.kdeplot(df['Sc'], color='purple')

df['Rbc'].plot(kind='density')

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

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

sns.ecdfplot(df['Rbc'], color='coral') # residplot, scatterplot

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

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

def plot(df, variable):

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

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

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

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

plot(df, 'Class')

**Module 3:**

# GRADIENT BOOSTING CLASSIFIER ALGORITHEM

# Import required libaries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings('ignore')

# Read csv file

df = pd.read\_csv('KIDNEY.csv')

df.head()

# Drop Missing values

df=df.dropna()

# Display last five rows in the table

df.tail()

# split the dataset into dependent and independent variables

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

y1 = df.loc[:,'Class']

# Check for imbalanced dataset

import imblearn

from imblearn.over\_sampling import RandomOverSampler

from collections import Counter

ros =RandomOverSampler(random\_state=42)

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

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

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

# Specific grouping for testing and training dataset

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

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.20, random\_state=42, stratify=y)

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

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

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

# Display the dataset after seperation

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

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

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

# import gradient boosting classifier

from sklearn.ensemble import GradientBoostingClassifier

# Fit the datasets for prediction

GRB = GradientBoostingClassifier(random\_state=42)

GRB.fit(x\_train,y\_train)

# Prediction for the datasets

predicted = GRB.predict(x\_test)

# Predict confusion matrix

from sklearn.metrics import confusion\_matrix

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

print('THE CONFUSION MATRIX SCORE OF GRADIENT BOOSTING CLASSIFIER:\n\n\n',cm)

# make valdation for accuracy

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

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

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

# Accuracy score for model prediction

from sklearn.metrics import accuracy\_score

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

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

# Loss based on model prediction

from sklearn.metrics import hamming\_loss

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

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

# Prcision based on model result

from sklearn.metrics import precision\_score

P = precision\_score(y\_test,predicted)

print("THE PRECISION SCORE OF GRADIENT BOOSTING CLASSIFIER IS :",P\*100)

# Recall score for model prediction

from sklearn.metrics import recall\_score

R = recall\_score(y\_test,predicted)

print("THE RECALL SCORE OF GRADIENT BOOSTING CLASSIFIER IS :",R\*100)

from sklearn.metrics import f1\_score

f1 = f1\_score(y\_test,predicted)

print("THE PRECISION SCORE OF GRADIENT BOOSTING CLASSIFIER IS :",f1\*100)

def plot\_confusion\_matrix(cm, title='THE CONFUSION MATRIX SCORE OF GRADIENT BOOSTING CLASSIFIER\n\n', cmap=plt.cm.Blues):

target\_names=['']

plt.imshow(cm, interpolation='nearest', cmap=cmap)

plt.title(title)

plt.colorbar()

tick\_marks = np.arange(len(target\_names))

plt.xticks(tick\_marks, target\_names, rotation=45)

plt.yticks(tick\_marks, target\_names)

plt.tight\_layout()

plt.ylabel('True label')

plt.xlabel('Predicted label')

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

print('THE CONFUSION MATRIX SCORE OF GRADIENT BOOSTING CLASSIFIER:\n\n')

print(cm)

sns.heatmap(cm/np.sum(cm), annot=True, cmap = 'Blues', annot\_kws={"size": 16},fmt='.2%')

plt.show()

def graph():

import matplotlib.pyplot as plt

data=[a]

alg=" GRADIENT BOOSTING CLASSIFIER"

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

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

plt.title("THE ACCURACY SCORE OF GRADIENT BOOSTING CLASSIFIER IS\n\n\n")

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

graph()

import joblib

joblib.dump(GRB, 'KIDNEY1.pkl')

**MODULE 4:**

# MANUAL NET ARCHITECTURE

import warnings

warnings.filterwarnings('ignore')

import os

import glob

import numpy as np

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.models import Sequential

from PIL import Image

from tensorflow.keras.layers import Convolution2D

from tensorflow.keras.layers import MaxPooling2D

from tensorflow.keras.layers import Flatten

from tensorflow.keras.layers import Dense

from tensorflow.keras.layers import Activation

from keras.callbacks import ModelCheckpoint

import matplotlib.pyplot as plt

Cyst = 'DATASET/TRAIN/Cyst'

Normal = 'DATASET/TRAIN/Normal'

Stone = 'DATASET/TRAIN/Stone'

Tumor = 'DATASET/TRAIN/Tumor'

def plot\_images(item\_dir, n=6):

all\_item\_dir = os.listdir(item\_dir)

item\_files = [os.path.join(item\_dir, file) for file in all\_item\_dir][:n]

plt.figure(figsize=(80, 40))

for idx, img\_path in enumerate(item\_files):

plt.subplot(3, n, idx+1)

img = plt.imread(img\_path)

plt.imshow(img, cmap='gray')

plt.axis('off')

plt.tight\_layout()

def image\_details\_print(data,path):

print('======== Images in: ', path)

for key,values in data.items():

print(key,':\t', values)

def images\_details(path):

files=[f for f in glob.glob(path + "\*\*/\*.\*", recursive=True)]

data={}

data['Images\_count']=len(files)

data['Min\_width']=10\*\*100

data['Max\_width']=0

data['Min\_height']=10\*\*100

data['Max\_height']=0

for f in files:

img=Image.open(f)

width,height=img.size

data['Min\_width']=min(width,data['Min\_width'])

data['Max\_width']=max(width, data['Max\_width'])

data['Min\_height']=min(height, data['Min\_height'])

data['Max\_height']=max(height, data['Max\_height'])

image\_details\_print(data,path)

print("")

print("TRAINING DATA FOR Cyst:")

print("")

images\_details(Cyst)

print("")

plot\_images(Cyst, 10)

print("")

print("TRAINING DATA FOR Normal:")

print("")

images\_details(Normal)

print("")

plot\_images(Normal, 10)

print("")

print("TRAINING DATA FOR Stone:")

print("")

images\_details(Stone)

print("")

plot\_images(Stone, 10)

print("")

print("TRAINING DATA FOR Tumor:")

print("")

images\_details(Tumor)

print("")

plot\_images(Tumor, 10)

train\_datagen=ImageDataGenerator(rescale=1./255,shear\_range=0.2,zoom\_range=0.2,horizontal\_flip=True)

training\_set=train\_datagen.flow\_from\_directory('dataset/train',target\_size=(224,224),batch\_size=32,class\_mode='categorical')

test\_datagen=ImageDataGenerator(rescale=1./255)

test\_set=test\_datagen.flow\_from\_directory('dataset/test',target\_size=(224,224),batch\_size=32,class\_mode='categorical')

Classifier=Sequential()

Classifier.add(Convolution2D(32,(3,3),input\_shape=(224,224,3),activation='relu'))

Classifier.add(MaxPooling2D(pool\_size=(2,2)))

Classifier.add(Flatten())

Classifier.add(Dense(38, activation='relu'))

Classifier.add(Dense(4, activation='softmax'))

Classifier.compile(optimizer='rmsprop',loss='categorical\_crossentropy',metrics=['accuracy'])

model\_path = "MANUAL.h5"

callbacks = [

ModelCheckpoint(model\_path, monitor='accuracy', verbose=1, save\_best\_only=True)

]

epochs = 50

batch\_size = 512

#### Fitting the model

history = Classifier.fit(

training\_set, steps\_per\_epoch=training\_set.samples // batch\_size,

epochs=epochs,

validation\_data=test\_set,validation\_steps=test\_set.samples // batch\_size,

callbacks=callbacks)

import matplotlib.pyplot as plt

def graph():

#Plot training & validation accuracy values

plt.plot(history.history['accuracy'])

plt.plot(history.history['val\_accuracy'])

plt.title('Model accuracy')

plt.ylabel('Accuracy')

plt.xlabel('Epoch')

plt.legend(['Train', 'Test'], loc='upper left')

plt.show()

graph()

import matplotlib.pyplot as plt

def graph():

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'])

plt.title('Model loss')

plt.ylabel('Loss')

plt.xlabel('Epoch')

plt.legend(['Train', 'Test'], loc='upper left')

plt.show()

graph()

**MODULES 5:**

## MOBILENET ARCHITECTURE

import warnings

warnings.filterwarnings('ignore')

import tensorflow

import tensorflow as tf

print(tf.\_\_version\_\_)

import keras

import keras.backend as K

from keras.models import Model

from keras.layers import Input, Dense, Conv2D, Conv3D, DepthwiseConv2D, SeparableConv2D, Conv3DTranspose

from keras.layers import Flatten, MaxPool2D, AvgPool2D, GlobalAvgPool2D, UpSampling2D, BatchNormalization

from keras.layers import Concatenate, Add, Dropout, ReLU, Lambda, Activation, LeakyReLU, PReLU

from time import time

import numpy as np

from keras.callbacks import ModelCheckpoint

from tensorflow.keras.callbacks import EarlyStopping

import warnings

warnings.filterwarnings('ignore')

from tensorflow.keras.preprocessing.image import ImageDataGenerator

train=ImageDataGenerator(rescale=1./255,shear\_range=0.2,zoom\_range=0.2,horizontal\_flip=True,validation\_split = 0.2)

train\_data=train.flow\_from\_directory(directory = 'DATASET/TRAIN',target\_size=(224,224),

batch\_size=32,class\_mode='categorical')

test=ImageDataGenerator(rescale=1./255)

test\_data=test.flow\_from\_directory(directory = 'DATASET/TEST',target\_size=(224,224),

batch\_size=32,class\_mode='categorical')

def mobilenet(input\_shape, n\_classes):

def mobilenet\_block(x, f, s=1):

x = DepthwiseConv2D(3, strides=s, padding='same')(x)

x = BatchNormalization()(x)

x = ReLU()(x)

x = Conv2D(f, 1, strides=1, padding='same')(x)

x = BatchNormalization()(x)

x = ReLU()(x)

return x

input = Input(input\_shape)

x = Conv2D(32, 3, strides=2, padding='same')(input)

x = BatchNormalization()(x)

x = ReLU()(x)

x = mobilenet\_block(x, 64)

x = mobilenet\_block(x, 128, 2)

x = mobilenet\_block(x, 128)

x = mobilenet\_block(x, 256, 2)

x = mobilenet\_block(x, 256)

x = mobilenet\_block(x, 512, 2)

for \_ in range(5):

x = mobilenet\_block(x, 512)

x = mobilenet\_block(x, 1024, 2)

x = mobilenet\_block(x, 1024)

x = GlobalAvgPool2D()(x)

output = Dense(n\_classes, activation='softmax')(x)

model = Model(input, output)

model.compile(optimizer='Adam',loss='categorical\_crossentropy',metrics=['accuracy',tensorflow.keras.metrics.Precision()])

return model

input\_shape = 224, 224, 3

n\_classes = 4

K.clear\_session()

model = mobilenet(input\_shape, n\_classes)

model.summary()

model\_path = "kerash\_model..h5"

from keras.callbacks import ModelCheckpoint

M = ModelCheckpoint(model\_path, monitor='accuracy', verbose=1, save\_best\_only=True)

epochs = 10

batch\_size = 512

#### Fitting the model

history = model.fit(

train\_data, steps\_per\_epoch=train\_data.samples // batch\_size,

epochs=epochs,

validation\_data=test\_data,validation\_steps=test\_data.samples // batch\_size,

callbacks=[M])

history.history.keys()

import matplotlib.pyplot as plt

import numpy as np

plt.figure(figsize=(20, 8))

plt.plot(history.history['accuracy'])

for i in range(epochs):

if i%5 == 0:

plt.annotate(np.round(history.history['accuracy'][i]\*100,2),xy=(i,history.history['accuracy'][i]))

plt.title('Model accuracy')

plt.ylabel('Accuracy')

plt.xlabel('Epoch')

plt.show()

plt.figure(figsize=(20, 8))

plt.plot(history.history['loss'])

for i in range(epochs):

if i%5 == 0:

plt.annotate(np.round(history.history['loss'][i]\*100,2),xy=(i,history.history['loss'][i]))

plt.title('Model Loss')

plt.ylabel('Loss')

plt.xlabel('Epoch')

plt.show()

**DEPLOY:**

from django.shortcuts import render, redirect

from . forms import  UserRegisterForm,Patient\_info\_Form

from django.contrib.auth import authenticate, login,logout

from django.contrib import messages

import numpy as np

import joblib

from .models import UserPredictModel,Patient\_info

from tensorflow import keras

from PIL import Image, ImageOps

from . import forms

def Landing\_1(request):

    return render(request, '1\_Landing.html')

def Register\_2(request):

    form = UserRegisterForm()

    if request.method =='POST':

        form = UserRegisterForm(request.POST)

        if form.is\_valid():

            form.save()

            print("data passed")

            user = form.cleaned\_data.get('username')

            messages.success(request, 'Account was successfully created. ' + user)

            return redirect('Login\_3')

    context = {'form':form}

    return render(request, '2\_Register.html', context)

def Login\_3(request):

    if request.method =='POST':

        username = request.POST.get('username')

        password = request.POST.get('password')

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

        if user is not None:

            login(request, user)

            return redirect('Home\_4')

        else:

            messages.info(request, 'Username OR Password incorrect')

    context = {}

    return render(request,'3\_Login.html', context)

def Home\_4(request):

    return render(request, '4\_Home.html')

def Teamates\_5(request):

    return render(request,'5\_Teamates.html')

def report(request):

    return render(request,'report.html')

import pyttsx3

def Deploy\_10(request):

    print("HI")

    if request.method == "POST":

        form = forms.UserPredictForm(files=request.FILES)

        if form.is\_valid():

            print('HIFORM')

            form.save()

        obj = form.instance

        result1 = UserPredictModel.objects.latest('id')

        models = keras.models.load\_model('C:/Users/SPIRO-PYTHON1/Desktop/UK/ITPML32/Deploy/PROJECT/APP/keras\_model.h5')

        data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)

        image = Image.open("C:/Users/SPIRO-PYTHON1/Desktop/UK/ITPML32/Deploy/PROJECT/media/" + str(result1)).convert("RGB")

        size = (224, 224)

        image = ImageOps.fit(image, size, Image.ANTIALIAS)

        image\_array = np.asarray(image)

        normalized\_image\_array = (image\_array.astype(np.float32) / 127.0) - 1

        data[0] = normalized\_image\_array

        classes = ['Cyst','Normal','Stone','Tumor']

        prediction = models.predict(data)

        idd = np.argmax(prediction)

        a = (classes[idd])

        if a == 'Cyst':

            a = 'THE KIDNEY CANCER TYPE OF CYST AFFECTED'

            engine = pyttsx3.init()

            engine.say(a)

            engine.runAndWait()

        elif a == 'Normal':

            a = 'NORMAL'

            engine = pyttsx3.init()

            engine.say(a)

            engine.runAndWait()

        elif a == 'Stone':

            a = 'THE KIDNEY CANCER TYPE OF A STONE AFFECTED'

            engine = pyttsx3.init()

            engine.say(a)

            engine.runAndWait()

        elif a == 'Tumor':

            a = 'THE KIDNEY CANCER TYPE OF A TUMOR AFFECTED'

            engine = pyttsx3.init()

            engine.say(a)

            engine.runAndWait()

        else:

            a = 'WRONG INPUT'

        data = UserPredictModel.objects.latest('id')

        data.label = a

        data.save()

        return render(request, 'result.html',{'form':form,'obj':obj,'predict':a})

    else:

        form = forms.UserPredictForm()

    return render(request, '10\_Deploy.html',{'form':form})

Model1 = joblib.load('C:/Users/SPIRO-PYTHON1/Desktop/UK/ITPML32/Deploy/PROJECT/APP/KIDNEY1.pkl')

def Deploy\_9(request):

    if request.method == 'POST':

        form = Patient\_info\_Form(request.POST)

        if form.is\_valid():

            # Extract cleaned data from form

            Bp = form.cleaned\_data['Bp']

            Sg = form.cleaned\_data['Sg']

            Al = form.cleaned\_data['Al']

            Su = form.cleaned\_data['Su']

            Rbc = form.cleaned\_data['Rbc']

            Bu = form.cleaned\_data['Bu']

            Sc = form.cleaned\_data['Sc']

            Sod = form.cleaned\_data['Sod']

            Pot = form.cleaned\_data['Pot']

            Hemo = form.cleaned\_data['Hemo']

            Wbcc = form.cleaned\_data['Wbcc']

            Rbcc = form.cleaned\_data['Rbcc']

            Htn = form.cleaned\_data['Htn']

            # Prepare features for prediction

            features = np.array([[Bp, Sg, Al, Su, Rbc, Bu, Sc, Sod, Pot, Hemo, Wbcc, Rbcc, Htn]])

            print(features)

            # Predict using the loaded model

            prediction = Model1.predict(features)

            prediction = prediction[0]

            print(prediction)

            if prediction == 0:

                a="This conditions is No Kidney Disease predict"

            elif prediction == 1:

                a="This conditions is Kidney Disease predict"

            # Save data to database

            instance = form.save(commit=False)

            instance.Class = a

            instance.save()

            # Render output page with prediction result

            return render(request, '5\_Teamates.html', {'prediction\_text': a})

    else:

        form = Patient\_info\_Form()

    return render(request, '9\_Deploy.html', {'form': form})

def res(request):

    return render(request,'result.html')

def Logout(request):

    logout(request)

    return redirect('Login\_3')

from .models import Patient\_info

def patient\_list(request):

    patients = Patient\_info.objects.all()

    return render(request, 'patient\_list.html', {'patients': patients})

def database(request):

    models = UserPredictModel.objects.all()

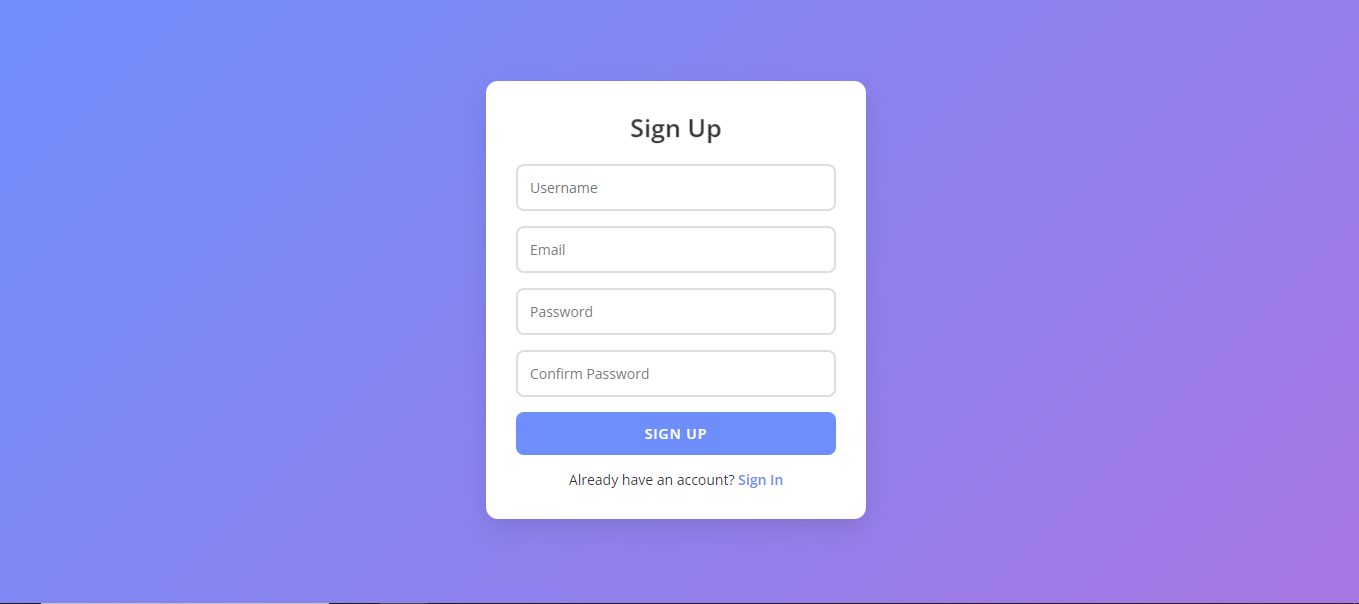
    return render(request, 'img\_database.html', {'models':models})

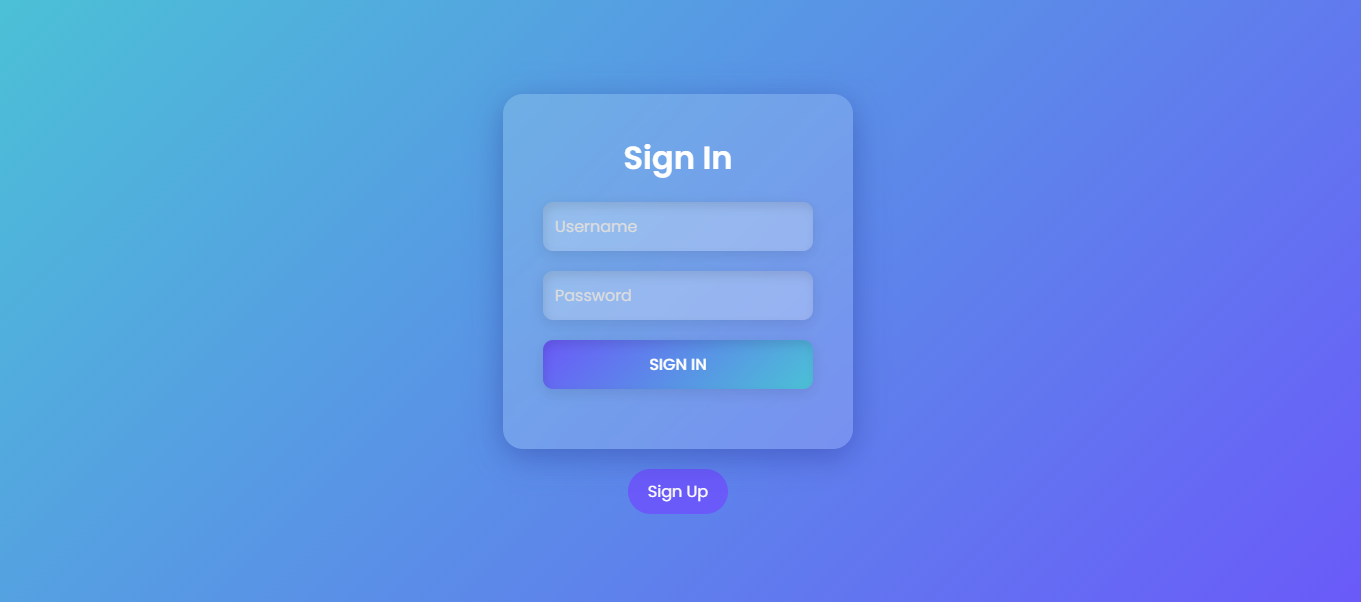
def matrix(request):

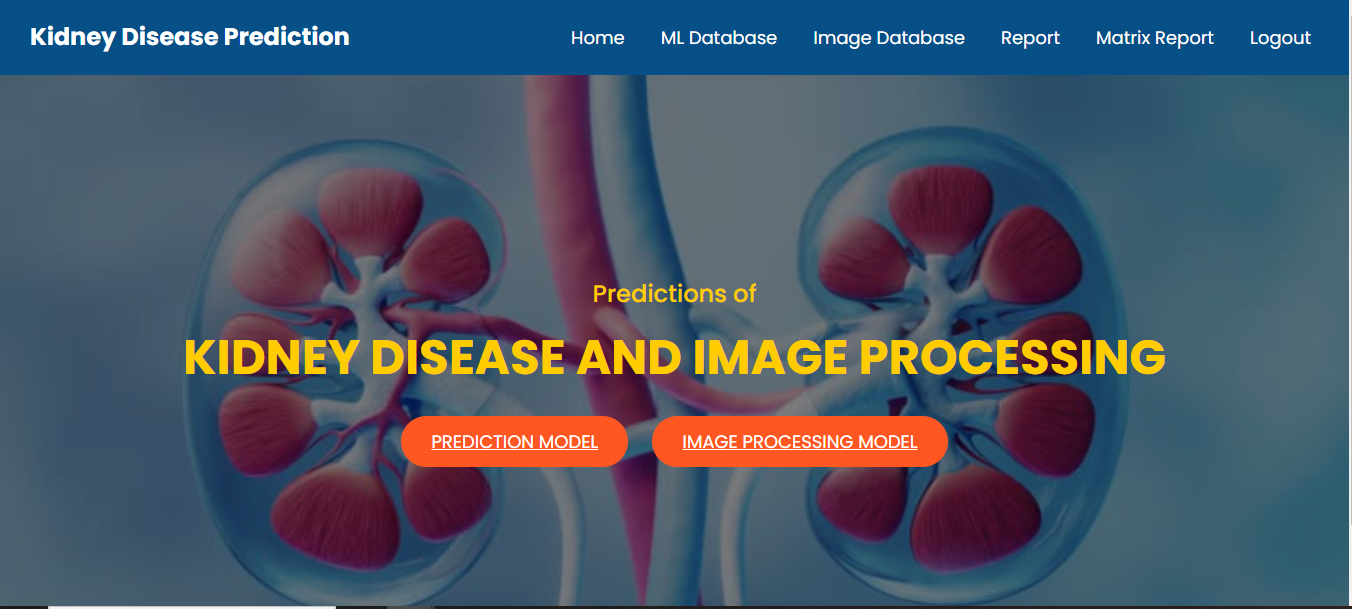
    return render(request,'matrix.html')

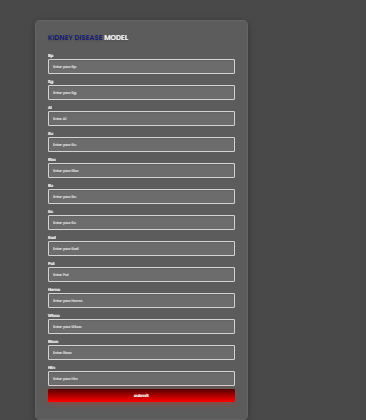
**OUTPUT SCREENHSOT:**

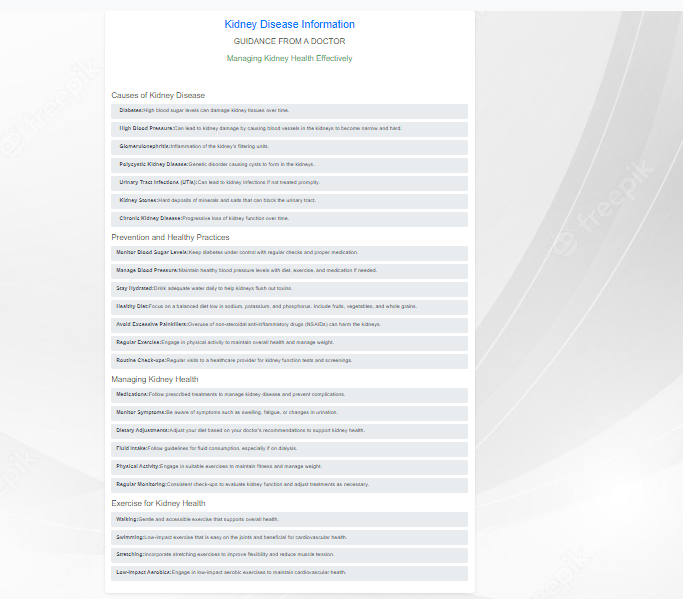
****

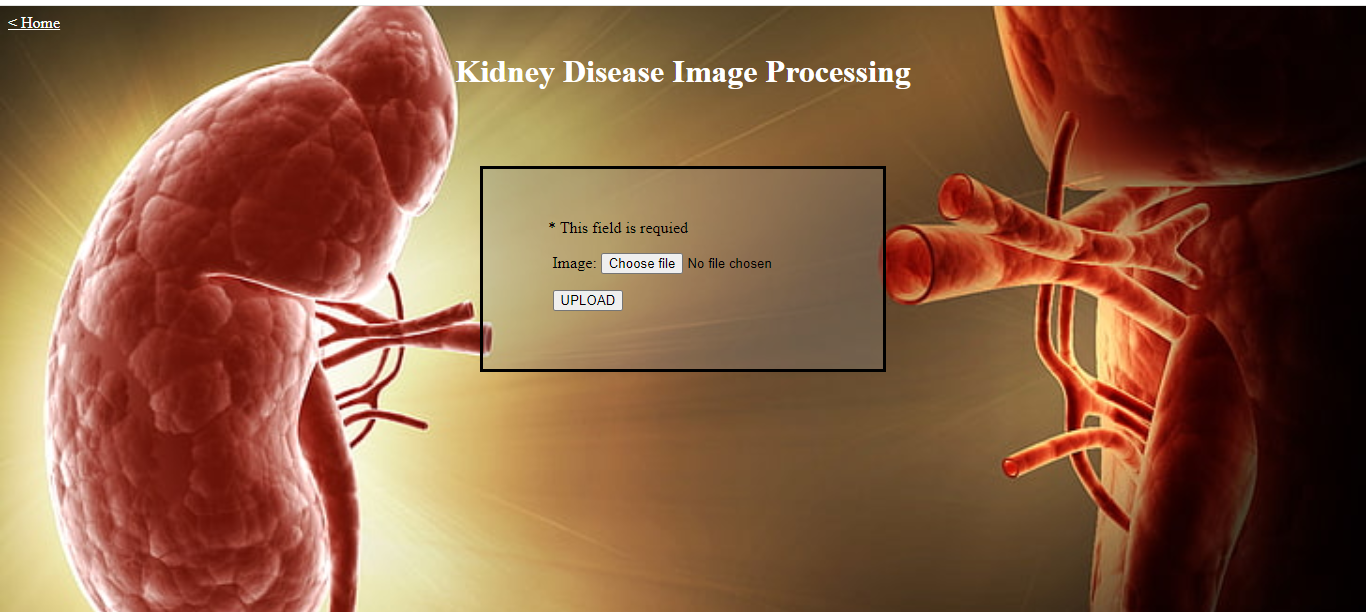
****

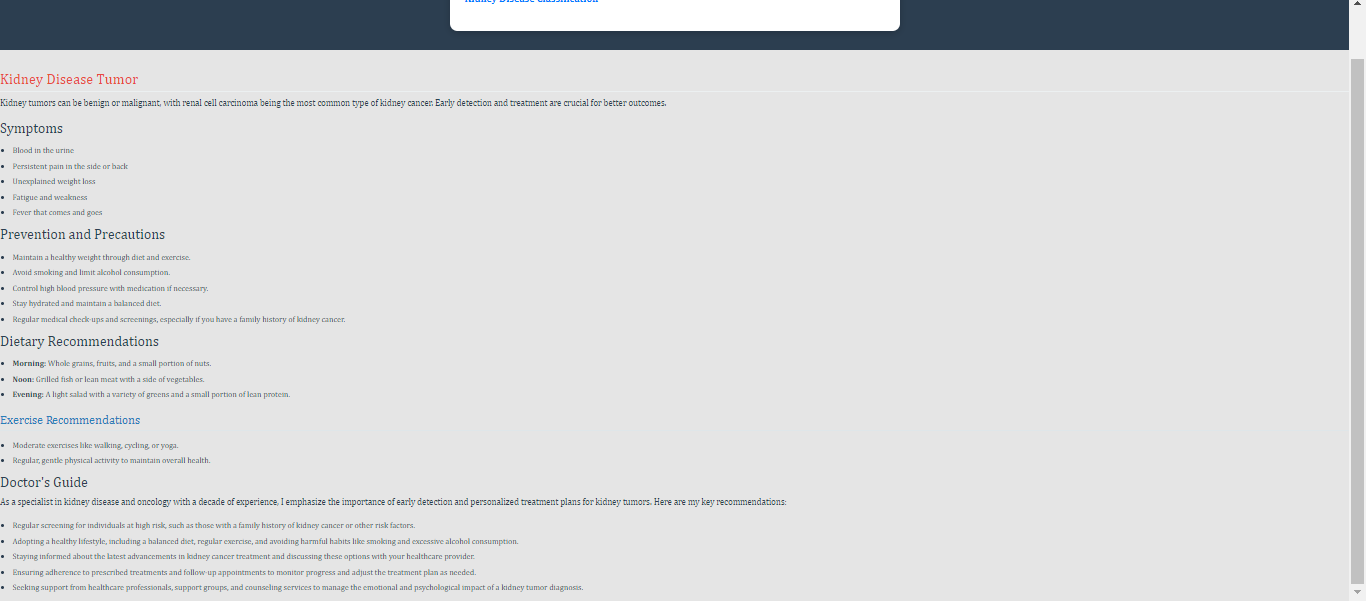
****

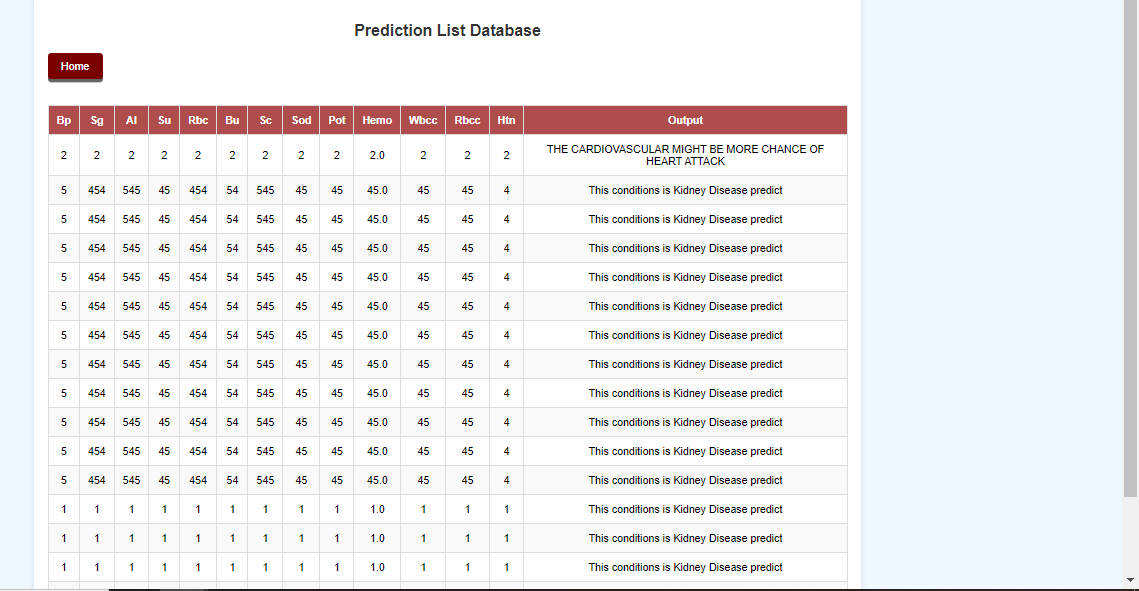
****

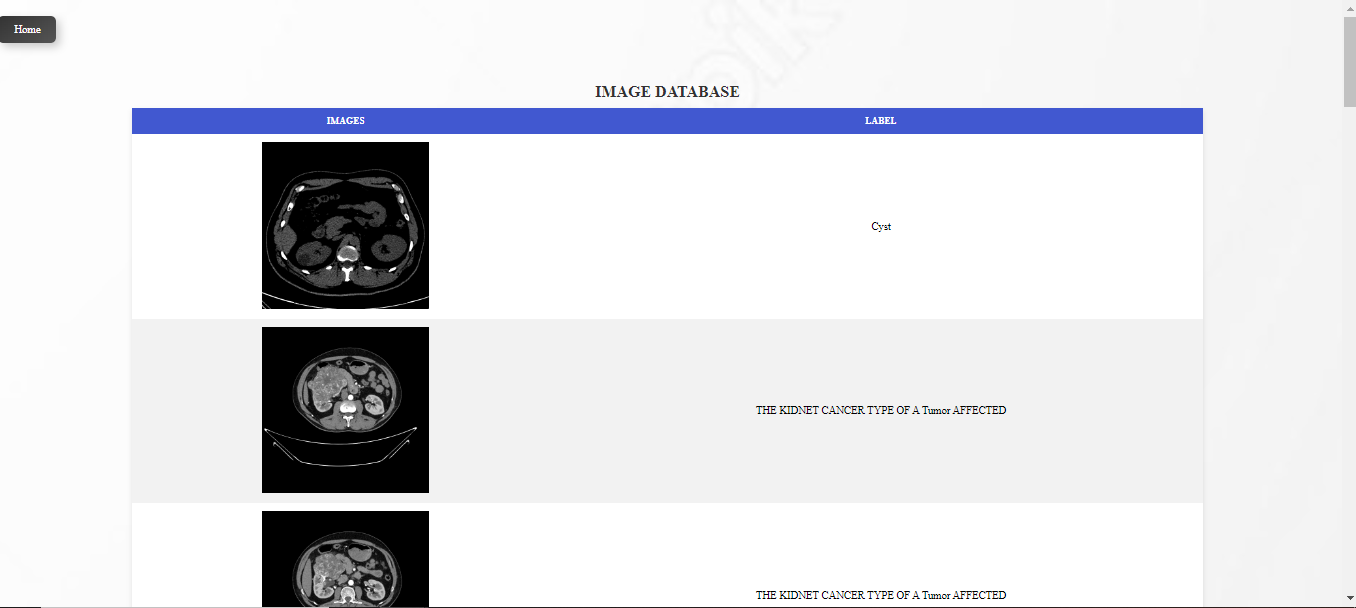
****

****

****

****

****

****

**Conclusion**

In conclusion, the integration of AI into kidney disease classification represents a revolutionary approach to early detection and diagnosis. By leveraging advanced machine learning techniques such as Convolutional Neural Networks (CNNs) and sophisticated data processing methods, this approach significantly enhances the accuracy and efficiency of identifying kidney abnormalities. The AI-powered system not only improves diagnostic precision by analyzing medical images and clinical data but also enables early intervention, which is crucial for better patient outcomes. With continuous advancements in AI technology and data collection, this methodology promises to transform kidney disease management, making it possible to detect conditions at their earliest stages and thereby improve treatment efficacy and patient quality of life.

**Future work**

* To automate this process by show the prediction result in web application or desktop application.
* We can deploy this model in any cloud based system.