**CHRONIC KIDNEY DISEASE PREDICTION SYSTEM USING MACHINE LEARNING**

**A PROJECT REPORT**

***Submitted by***

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

***In***

**COMPUTER SCIENCE AND ENGINEERING**

****

**ANNA UNIVERSITY : CHENNAI 600 025**

**APRIL 2020**

**ABSTRACT**

Chronic kidney disease, also called chronic kidney failure, describes the gradual loss of kidney function. Your kidneys filter wastes and excess fluids from your blood, which are then excreted in your urine. When chronic kidney disease reaches an advanced stage, dangerous levels of fluid, electrolytes and wastes can build up in your body. Treatment for chronic kidney disease focuses on slowing the progression of the kidney damage, usually by controlling the underlying cause. Chronic kidney disease can progress to end-stage kidney failure, which is fatal without artificial filtering (dialysis) or a kidney transplant.Chronic Kidney Disease (CKD) need to be diagnosed earlier before kidneys fail to work.In order to help doctors or medical experts in prediction of CKD among patients easily, this paper has developed an intelligent system named Chronic Kidney Disease Prediction System (CKDPS) that can predict CKD among patients.

The proposed system predict the CKD with minimal feature input instead of dumping all the features which may not relevant to predict the disease.To achieve this we have planned to approach by three feature selection algorithm with combination of two feature Extraction algorithm.After performing feature selection and Feature Extraction, those features will be trained with different Machine Learning algorithm. The accuracy of best combination algorithm will be implemented for predicting the CKD.Finally, Random Forest algorithm is chosen to implement CKDPS as it gives 95% accuracy, precision and recall results.

**LIST OF FIGURES**

**Figure No. Figure Name Page No.**

1. Architecture Diagram 10
2. Uml Diagram 11
3. Use Case Diagram 12
4. Collabration Uml Diagram 13
5. Activity Diagram 14
6. DFD Level 0 15
7. DFD Level 1 15
8. DFD Level 2 16
9. Preprocessing Diagram 25
10. Regression Module 27
11. SVM 33
12. Decision Tree 37
13. K-Nearest Neighbour 40
14. Linear Regression 42
15. Class Diagram 43
16. Random Forest 46

**LIST OF ABBREVIATIONS**

**ABBREVIATIONS EXPANSION**

ML Machine Learning.

CKD Chronic Kidney Disease.

CKDPS Chronic Kidney Disease Prediction System.

KNN K-Nearest Neighbour.

SVM Support Vector Machine.

RF Random Forest.

NB Naïve Bayes.

DT Decision Tree.

ESRD End Stage Renal Disease.

GUI Graphical User Interface.

DFD Data Flow Diagram.

**TABLE OF CONTENTS**

**CHAPTER TITLE PAGE NO NO ABSTRACT**

**LIST OF FIGURES**

**LIST OF ABBREVATION**

1. **INTRODUCTION**

* 1. Domain introduction 1
  2. Project introduction 2

2. **LITERATURE REVIEW**

2.1 General 4

2.2 Literature Survey 4

3. **SYSTEM ANALYSIS AND DESIGN**

3.1 Introduction 8

3.2 Existing system 8

3.3 Proposed system 9

3.4 System design 10

3.4.1 Architectural diagram 10

3.5 UML diagram 11

3.5.1 usecase diagram 11

3.5.2 collabration uml diagram 12

3.5.3 Activity diagram 13

3.6 Data Flow diagram 15

4. **SYSTEM SPECIFICATIONS**

4.1 Requirement analysis 17

4.2 Hardware specifications 17

4.3 Software specification 18

4.3.1 Anaconda 19

4.3.2 Spyder 19

4.3.3 Django 21

4.3.4 Hyper Text Markup Language(HTML) 22

4.3.5 Python 23

5. **MODULE EXPLAINATION**

5.1 list of modules 25

5.1.1 user interface module 25

5.1.2 Preprocessing module 26

5.1.3 Classification module 28

5.1.4 Feature Selection module 29

5.1.5 Feature Extraction module 29

5.1.6 prediction module 30

5.1.7 Output module 31

5.2 Algorithms used

5.2.1 Support Vector Machine Algorithm 31

5.2.2 Decision Tree Algorithm 34

5.2.3 K-Nearest Neighbour Algorithm 38

5.3.4 Naïve Bayes Algorithm 41

5.3.5 Linear Regression Algorithm 43

5.3.6 Random Forest Algorithm 45

6. **CONCLUSION AND FUTURE ENHANCEMENT**

6.1 conclusion 48

6.2 Future enhancement 48

**APPENDIX A(coding) 49**

**APPENDIX B(output) 76**

**JOURNAL PAPER PUBLICATION CERTIFICATES**

**REFERENCES 82**

**CHAPTER 1**

**INTRODUCTION**

* 1. **DOMAIN INTRODUCTION:**

**Machine learning**, sometimes called ML, is a cutting-edge field in computer science that seeks to get computers to carry out tasks without being explicitly programmed to carry out a given task. Machine learning uses many techniques to create algorithm to learn and make predictions from data sets. It is used in data mining which is a technique to discover patterns and models in data sets where relationships are previously unknown. Machine learning is used in search engines, optimization problems, computer vision, and more. These concepts have been applied in Google’s self-driving car and in recommendation engines on sites like Amazon and Netflix.

Machine learning algorithms work by building a model from a **training set**. A training set is a data set that is input into an algorithm where the correct outputs are already known. The ML algorithm builds the model as it reads the training set, reading the next input, predicting the output, then checking its prediction to the actual output, and adjusting accordingly.

Machine learning is sometimes thought of as purely data mining, but data mining is a subfield of machine learning that uses unsupervised learning.

Supervised learning is the machine learning task of determining a function from labeled data. For example, in a machine learning algorithm that detects if a post is spam or not, the training set would include posts labeled as "spam" and posts labeled as "not spam" to help teach the algorithm how to recognize the difference. Supervised learning algorithms infer a function from labeled data and use this function on new examples. Since the algorithm receives a data set as input that already has a correct answer associated with it, the algorithm will learn by comparing its output with the correct answer, and if it finds errors, it adjusts the model accordingly. The training continues until the algorithm outputs information within the desired accuracy range.Unsupervised learning uses input data that is unlabeled. Specifically, because the data is unlabeled, there is no error or reward to let the algorithm know if it is close or far away from the proper solution. Unsupervised learning is very important when using machine learning on problems where the answer is not known. The goal of unsupervised learning is to take in data and explore it to find some structure within the data. A model is created from unsupervised learning by deducing structures and patterns in the input data.

Regression is a statistics method used to estimate relationships among variables. Usually, regression is concerned with how the typical value of the dependent variable changes when an independent variable is altered. Regression analysis estimates the **conditional expectation** of the dependent variable given the independent variables. In other words, it estimates the average value of the dependent variable when the independent variables are fixed.

* 1. **PROJECT INTRODUCTION:**

In today’s world chronic kidney disease (CKD) becomes one of the serious public health problems. CKD is the damaging condition of kidneys that can be worse over time. If the kidneys are damaged very badly then they fail to work. This is called kidney failure, or end-stage renal disease (ESRD). As per report in India 6000 renal transplants are done annually. It is also reported that, CKD among people is increasing rapidly all over the world. Diabetes, heart disease, heredity and high blood pressure are the most common causes of kidney disease. However, it has been observed that if CKD is diagnosed in early stage then kidney failure can be prevented with proper medical treatment. Medical experts or doctors diagnose CKD by patient’s symptoms, physical exam, urine test and blood tests. So, if an expert system is able to predict the kidney disease in patients, it will be very helpful for the medical experts or doctors to give proper treatment to the patients in time. In the recent world, many researches are done in the medical ﬁeld. Nowadays, in our society machine learning algorithms are used widely for effective prediction of various diseases. In this paper, to help medical experts or doctors in prediction of CKD, Chronic Kidney Disease Prediction System (CKDPS) using Random Forest algorithm has been developed. Random Forest algorithm is one of the machine learning techniques. An application of artiﬁcial intelligence (AI) is machine learning that provides the able systems to learn automatically and improve from experience without being explicitly programmed. In medical ﬁeld machine learning analyzes massive quantities of medical data and delivers accurate results in order to identify the disease. In this paper, the dataset used to develop CKDPS is taken from the Kaggle machine learning database. The machine learning algorithms such as, k-Nearest Neighbors (KNN), Linear Regression (LR), Decision Tree (DT), Random Forest (RF), Naïve Bayes (NB), Support Vector Machine (SVM)algorithm are applied on the dataset. Experimental result shows that only Random Forest Classiﬁer algorithm gives better classiﬁcation performance with 95% accuracy, precision and recall results. So, here, only Random Forest Classiﬁer has been used to predict CKD in CKDPS. As a screening tool the graphical user interface(GUI) of CKDPS has been developed with the help of python so that doctors or medical experts diagnose CKD among patients very easily.

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 GENERAL**

Literature [survey](http://www.blurtit.com/q876299.html) is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy n company strength. Once these things r satisfied, ten next step is to determine which operating system and language can be used for developing the tool. Once the [programmers](http://www.blurtit.com/q876299.html) start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from [book](http://www.blurtit.com/q876299.html) or from websites. Before building the system the above consideration are taken into account for developing the proposed system.

**2.2 REVIEW OF LITERATURE**

**SURVEY-1:**

**Title:** Identifying important attributes for early detection of Chronic Kidney Disease.

**Author:** Anandanadarajah Nishanth and Tharmarajah Thiruvaran.

**Year:**2017.

**Description:** They proposed the If the important attributes that could help to detect CKD is known then even people who are not diagnosed CKD also may get a clue of the condition of their kidney from the medical test that they took for some other purposes. Then they may proceed to properly check for CKD. A weighting vector based on CSP filter and LDA analysis and then classification analysis using LDA and KNN classifiers were used to identify the dominant attributes. It is found that hemoglobin, albumin, specific gravity, serum creatinine, hypertension and diabetes mellitus are the most important attributes in detecting CKD.

**SURVEY-2:**

**Title:** A Deep Learning-based System for Automated Sensing of Chronic Kidney Disease.

**Author:** Navaneeth Bhaskar and Suchetha M.

**Year:**2019.

**Description:** They have proposed and investigated a new sensing model for prompt and accurate diagnosis of CKD. The raw sensor signal is directly given to the deep learning algorithm for predictive decision making. The proposed 1-D CNN-SVM algorithm extracted features directly from the raw signal and successfully classified the samples with an accuracy of 98.04%. The proposed sensing approach is tested and validated by the physician.They have performed the statistical analysis to determine how well the proposed sensing method values and traditional urea estimation values are correlated. A positive correlation is observed between the two values with r and R2 values of 0.9898 and 0.9799 respectively. The experimental evaluation results show that the proposed sensing module can be successfully used with the capabilities of deep learning techniques for detecting CKD more effectively than traditional methods.

**SURVEY-3:**

**Title:** In vivo detection of chronic kidney disease using tissue deformation fields from dynamic MR imaging.

**Author:** Erlend Hodneland1, Eirik Keilegavlen, Erik A. Hanson, Erling Andersen, Jan Ankar Monssen, Jarle Rørvik.

**Year:**2018.

**Description:** They proposed a framework for detecting pathological changes in stiffness using image registration. Our findings are based on a small cohort, and need to be verified in a larger clinical study. Still, initial results indicate that the absolute deformation, normalized volume changes, as well as pressure gradients demonstrate a statistical correlation to arteriosclerosis in the kidneys, and hence could be used as a proxy for arteriosclerosis grade. The simulation study indicated that image registration has a borderline sensitivity close to the stiffness changes seen in mild CKD.

**SURVEY-4:**

**Title:** Erythropoietin Dose Optimization for Anemia in Chronic Kidney Disease Using Recursive Zone Model Predictive Control.

**Author:** Jayson McAllister, Zukui Li, Jinfeng Liu and Ulrich Simonsmeier.

**Year:**2018.

**Description:** They proposes a novel modeling and control algorithm, based on recursive weighted constrained ARX modeling and ZMPC, for the regulation of hemoglobin concentration in patients with CKD undergoing ESA treatment. This paper compares the performance of several different modeling techniques on actual clinical data, showing that the chosen modeling method provides sufficient modeling accuracy, while being less complex and able to be solved with a more robust optimization solver.

**SURVEY-5:**

**Title:** Mechanical Anisotropy Assessment in Kidney Cortex Using ARFI Peak Displacement: Preclinical Validation and Pilot *In Vivo* Clinical Results in Kidney Allografts.

**Author:** Jayson McAllister, Zukui Li, Jinfeng Liu and Ulrich Simonsmeier.

**Year:**2018.

**Description:** The clinical feasibility of ARFI-PD ratio as a measure of mechanical anisotropy in the allografts of renal transplant patients. In healthy allografts, PD ratios were stable over two, four, and six months post-transplantations. Preclinical validation studies in pig kidneys showed that PD-ratios linearly correlated with SWEI-derived shear moduli ratios *ex vivo* as well as *in vivo* at baseline, with arterial ligation, and with venous ligation. Further, mechanical anisotropy was exploited by using an asymmetrical ARF, but a symmetric ARF generally obviated anisotropy and enabled angle independent mechanical property assessment in the anisotropic cortex. Thus, when mechanical anisotropy is of interest as a renal biomarker, an asymmetric ARF should be implemented.

**CHAPTER 3**

**SYSTEM ANALYSIS AND DESIGN**

**3.1 INTRODUCTION:**

* Chronic kidney disease, also called chronic kidney failure, describes the gradual loss of kidney function. Your kidneys filter wastes and excess fluids from your blood, which are then excreted in your urine. When chronic kidney disease reaches an advanced stage, dangerous levels of fluid, electrolytes and wastes can build up in your body.
* In the early stages of chronic kidney disease, you may have few signs or symptoms. Chronic kidney disease may not become apparent until your kidney function is significantly impaired.
* Treatment for chronic kidney disease focuses on slowing the progression of the kidney damage, usually by controlling the underlying cause. Chronic kidney disease can progress to end-stage kidney failure, which is fatal without artificial filtering (dialysis) or a kidney transplant.
* we can predict the early stage of Chronic kidney Disease using machine learning techniques.

**3.2 EXISTING SYSTEM:**

Existing have used neural network for Colombian Population dataset.The result was obtained by training a neuronal network with 5 layers: an input layer with 7,492 neurons, corresponding to the variables or characteristics of the model, 3 hidden layers with 500, 100 and 50 neurons respectively, and an output layer with a single neuron representing the class of the binary classification problem.

The model achieves 95% accuracy in the test data set, making its application for disease prognosis feasible.

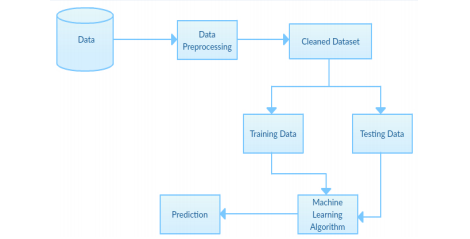
**3.3 PROPOSED SYSTEM:**

* The proposed system, uses machine learning algorithm to predict CKD for Indian set.
* By using Feature selection and feature extraction minimal number of feature will be required for the prediction.
* To achieve this we have planned to approach by three feature selection algorithm with combination of two feature Extraction algorithm.
* After performing feature selection and Feature Extraction, those features will be trained with different Machine Learning algorithm.
* The accuracy of best combination algorithm will be implemented for predicting the CKD.
* The dataset is collected from Kaggle machine learning database, obtained by the survey of CKD in India that contains laboratory results of both positive and negative cases of CKD. It contains cases of 400 patients with 25 attributes (eg, red blood cell count, white blood cell count, etc.).
* Input attributes are used to take input from user in CKDPS and output attribute is for diagnosis result.
* To classify that patient has CKD or Not CKD according to the values of input attributes the output attribute “Classiﬁcation” is set as the target class.
* Except the target class column, the entire dataset is divided into two sets in 7:3 ratios. 70% of which is used for training the machine learning algorithms and the 30% is used to test their accuracy, precision and recall.

**3.4 SYSTEM DESIGN:**

**3.4.1 ARCHITECTURAL DIAGRAM:**

An architecture diagram is a graphical representation of a set of concepts, that are part of an architecture, including their principles, elements and components. There are many kinds of architecture diagrams, like a software architecture diagram, system architecture diagram, application architecture diagram, security architecture diagram, etc. A **system architecture** or **systems architecture** is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behavior of the system.

****

**Fig 1. Architecture Diagram**

**3.5 UML DIAGRAM:**

The **Unified Modeling Language** (**UML**) is a general-purpose, developmental, modeling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system.

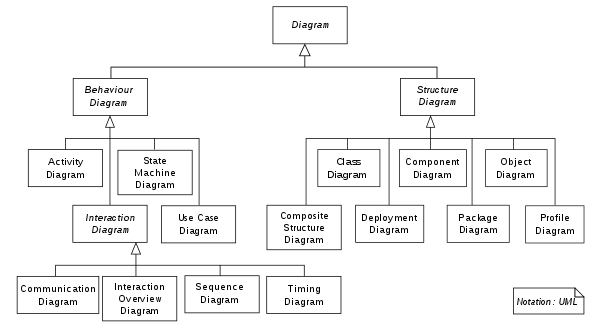


Fig 2 . Uml Diagram.

**3.5.1 USE CASE DIAGRAM:**

A **use case diagram** at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.

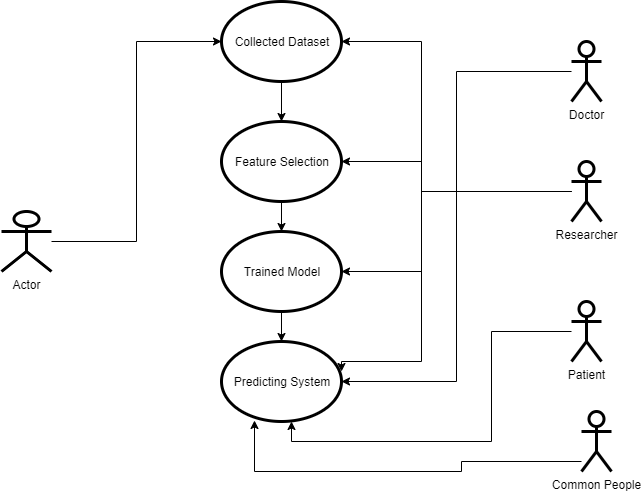
****

Fig 3. Use Case Diagram

**3.5.2 COLLABRATION UML DIAGRAM:**

A Collabration Diagram is a diagram that shows object interactions organized around the objects and their links to each other. Unlike a sequence Diagram, a Collabration diagram shows the relationships among the objects. Sequence diagrams and the collabration diagram s express similar information, but show it in different ways.

It represents the collabration, which is a set of objects roles related in a particular context, and in the interaction it is the set of messages exchanged among the objects to achieve an operation or result.

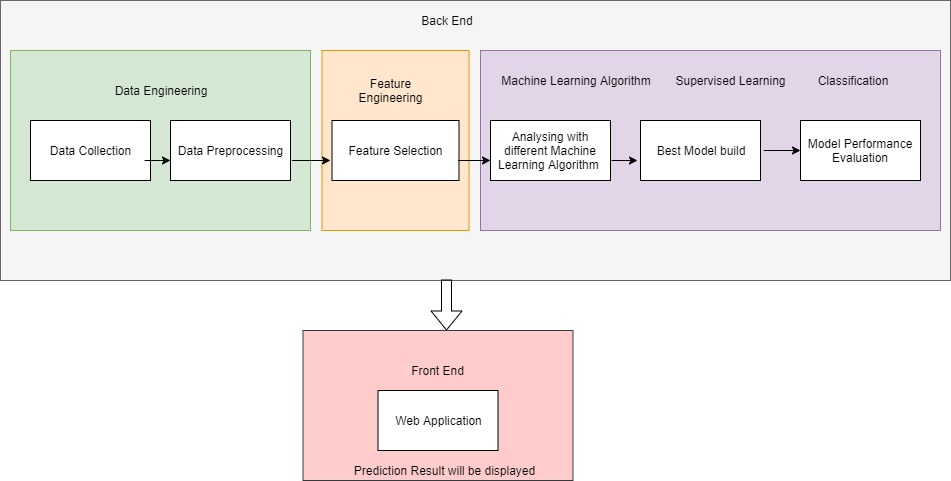


Fig 4. Collabration Uml Diagram

**3.5.3 ACTIVITY DIAGRAM:**

**Activity diagrams** are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e., workflows), as well as the data flows intersecting with the related activities. Although activity diagrams primarily show the overall flow of control, they can also include elements showing the flow of data between activities through one or more data stores.

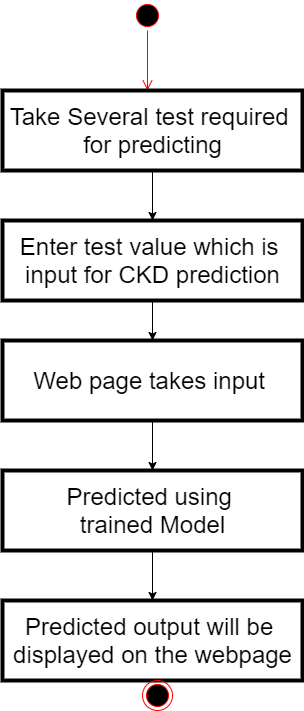
****

Fig 5. Activity Diagram.

**3.6 Data Flow Diagram:**

Data Flow diagram (DFD) is a way of representing a flow of a data of a process or a system (usually an information system).The DFD also provides information about the output and input of each entity and process itself .A data flow diagram has no control flow ,there are no decision rules and no loops. Specific operations based on the data can be represented by a flowchart.

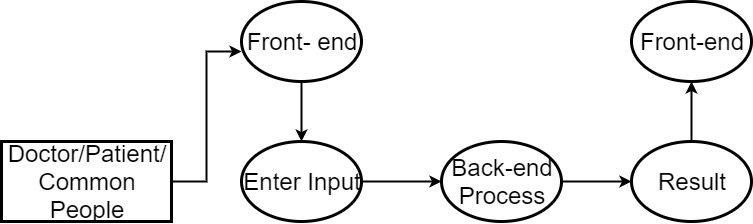
****

Fig 6. DFD Level 0.

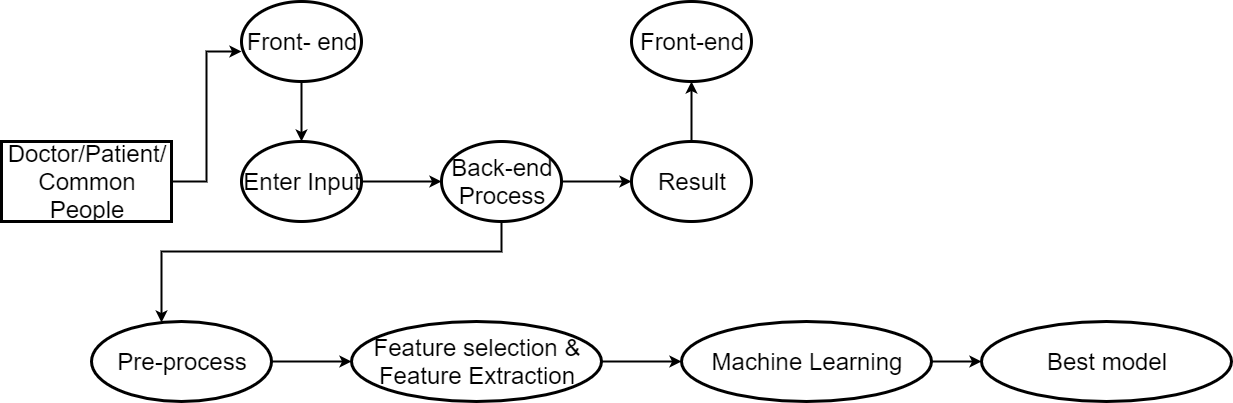


Fig 7. DFD Level 1.

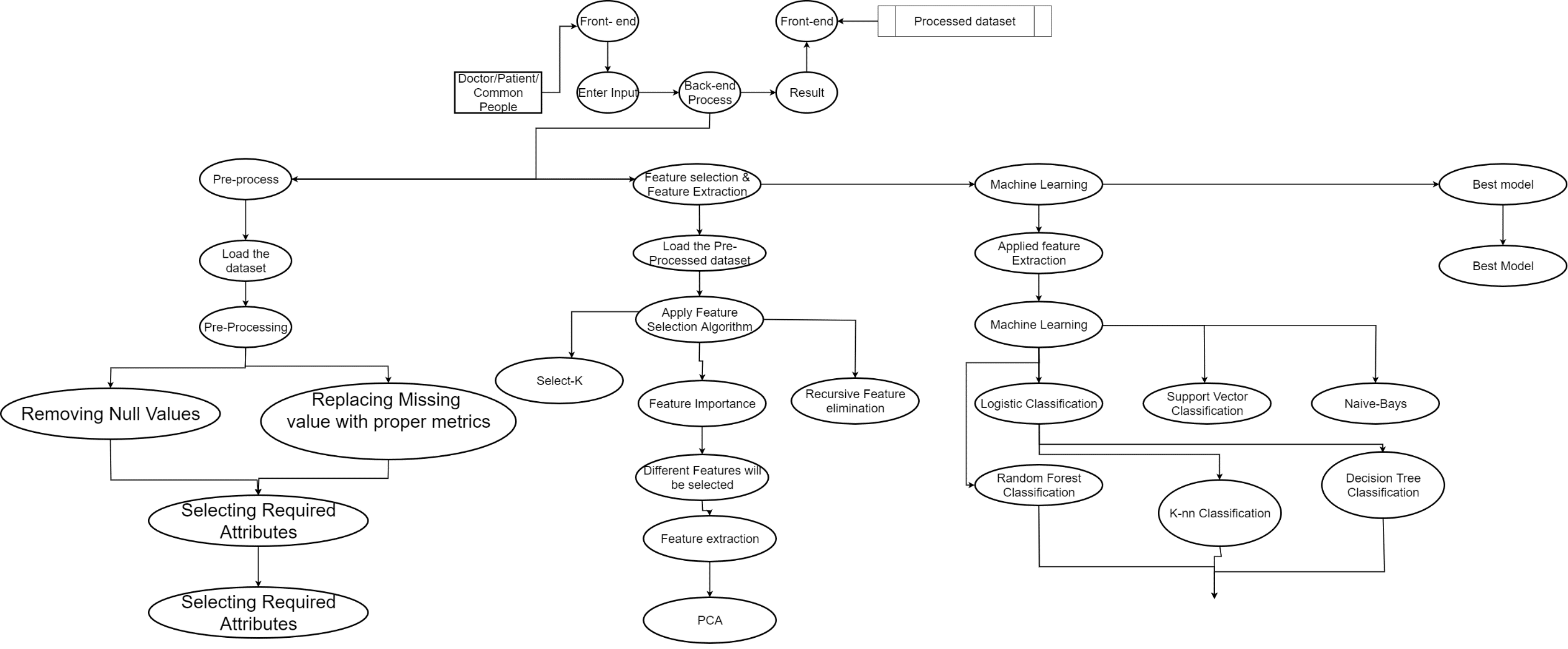


Fig 8. DFD Level 2.

**CHAPTER 4**

**SYSTEM SPECIFICATIONS:**

System specifications is a structured collection of information that embodies the requirements of a system. The System Specification describes the functional and non-functional requirements posed on a system element (system, Enabling System or segment). In order to prepare the System Specification, the requirements will be derived from the specifications of higher system elements or from the Overall System Specification.

**4.1. REQUIREMENT ANALYSIS:**

In Systems engineering and software engineering, requirements analysis focuses on the tasks that determine the needs or conditions to meet the new or altered product or project, taking account of the possibly conflicting requirements of the various stakeholders, analyzing, documenting, validating and managing software or system requirements.

Requirements analysis is critical to the success or failure of a systems or software project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

**4.2. HARDWARE SPECIFICATIONS:**

Hardware requirements is the most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems.

* Processor: AMD pro, Intel i3,i5,i7.
* RAM: 4 gigabyte (GB) for 64-bit(minimum) or 8 GB for 64-bit.
* Hard disk space: 250 GB for 64-bit OS.
* Display: 1200x800.

**4.3. SOFTWARE SPECIFICATIONS:**

[Software requirements](https://en.wikipedia.org/wiki/Software_requirements) deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed.

Specification involves representing and storing the collected requirements knowledge in a persistent and well-organized fashion that facilitates effective communication and change management. Use cases, user stories, functional requirements, and visual analysis models are popular choices for requirements specification.

* OPERATING SYSTEM: Windows 8,8.1,10.
* PLATFORM : Anaconda.
* TEXT EDITOR : Spyder.
* FRAMEWORK : Django.
* LANGUAGES : Python,HTML,CSS,Bootstrap.

**4.3.1. ANACONDA:**

**Anaconda** is a free and open source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analysis, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system *conda.* The Anaconda distribution includes data-science packages suitable for Windows, Linux, and MacOS.

Anaconda Navigator is a desktop graphical user interface(GUI) included in Anaconda distribution that allows users to launch applications and 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, install them in an environment, run the packages and update them. It is available for Windows, macOS and Linux.

Anaconda Cloud is a package management service by Anaconda where you can find, access, store and share public and private notebooks, environments, and conda and PyPI packages.Cloud hosts useful Python packages, notebooks and environments for a wide variety of applications. You do not need to log in or to have a Cloud account, to search for public packages, download and install them.

You can build new packages using the Anaconda Client command line interface (CLI), then manually or automatically upload the packages to Cloud.

**4.3.2. SPYDER:**

**Spyder** is an open source cross-platform integrated development environment (IDE) for scientific programming in the Python Language. Spyder integrates with a number of prominent packages in including NumPy, SciPy, Matplotlib, Pandas, Ipython, Sympy and Cy-thon, as well as other open source software. It is released under the MIT License.

Initially created and developed by Pierre Raybaut in 2009, since 2012 Spyder has been maintained and continuously improved by a team of scientific Python developers and the community.

Spyder is extensible with first- and third-party plugins, includes support for interactive tools for data inspection and embeds Python-specific code quality assurance and introspection instruments, such as Pyflakes, Pylint and Rope. It is available cross-platform through Anaconda, on Windows, on macOS through Macports, and on major Linux distributions such as ArchLinux, Debin, Fedora,Gento Linux, openSUSE and Ubuntu.

Spyder uses Qt for its GUI, and is designed to use either of the PyQt or PySide Python bindings. QtPy, a thin abstraction layer developed by the Spyder project and later adopted by multiple other packages, provides the flexibility to use either backend.

**Features:**

* An editor with syntax highlighting, introspection, code completion.
* Support for multiple Ipython consoles.
* The ability to explore and edit variables from a GUI.
* A Help pane able to retrieve and render rich text documentation on functions, classes and methods automatically or on-demand.
* A debugger linked to IPdb, for step-by-step execution.
* Static code analysis, powered by Pylint.
* A run-time Profiler, to benchmark code.
* Project support, allowing work on multiple development efforts simultaneously.
* A built-in file explorer, for interacting with the filesystem and managing projects.
* A "Find in Files" feature, allowing full regular expression search over a specified scope.
* An online help browser, allowing users to search and view Python and package documentation inside the IDE.
* A history log, recording every user command entered in each console.
* An internal console, allowing for introspection and control over Spyder's own operation.

**4.3.3. DJANGO:**

Django was created in the fall of 2003, when the web programmers at the Lawrence Journal-World newspaper, Adrian Holovaty and Simon Willison, began using Python to build applications. Jacob Kaplan-Moss was hired early in Django's development shortly before Simon Willison's internship ended. It was released publicly under a BSD license in July 2005. The framework was named after guitarist Django Reinhardt.

In June 2008, it was announced that a newly formed Django Software Foundation (DSF) would maintain Django in the future.

In July 2015, Revolution Systems, a software consulting company connected to some Django co-founders and developers, hosted 10th anniversary events in Lawrence

**Features:**

Despite having its own nomenclature, such as naming the callable objects generating the HTTP responses "views",the core Django framework can be seen as an MVC architecture. It consists of an object-relational mapper (ORM) that mediates between data models (defined as Python classes) and a relational database ("**M**odel"), a system for processing HTTP requests with a web templating system ("**V**iew"), and a regular-expression-based URL dispatcher ("**C**ontroller").

Also included in the core framework are:

* a lightweight and standalone web server for development and testing
* a form serialization and validation system that can translate between HTML forms and values suitable for storage in the database
* a template system that utilizes the concept of inheritance borrowed from object-oriented programming
* a caching framework that can use any of several cache methods
* support for middleware classes that can intervene at various stages of request processing and carry out custom functions
* an internal dispatcher system that allows components of an application to communicate events to each other via pre-defined signals
* an internationalization system, including translations of Django's own components into a variety of languages
* a serialization system that can produce and read XML and/or JSON representations of Django model instances
* a system for extending the capabilities of the template engine
* an interface to Python's built-in unit test framework
* Django REST framework is a powerful and flexible toolkit for building Web APIs.

**4.3.4 HYPER TEXT MARKUP LANGUAGE(HTML):**

* Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.
* Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document. HTML can embed programs written in a [scripting language](https://en.wikipedia.org/wiki/Scripting_language) such as [JavaScript](https://en.wikipedia.org/wiki/JavaScript), which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content.
* HTML markup consists of several key components, including those called tags (and their attributes), character-based datatypes, character references and entity references. HTML tags most commonly come in pairs like <**h1**> and </**h1**>, although some represent empty elements and so are unpaired, for example <**img**>. The first tag in such a pair is the start tag, and the second is the end tag (they are also called opening tags and closing tags).
* Another important component is the HTML document type declaration, which triggers standards mode rendering.

**4.3.4 PYTHON:**

Python is interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. It is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. Python  [interpreters](https://en.wikipedia.org/wiki/Interpreter_(computing)) are available for many [operating systems](https://en.wikipedia.org/wiki/Operating_system). A global community of programmers develops and maintains [CPython](https://en.wikipedia.org/wiki/CPython" \o "CPython), an [open source](https://en.wikipedia.org/wiki/Open-source_software) [reference implementation](https://en.wikipedia.org/wiki/Reference_implementation).

**Elements:**

HTML documents imply a structure of nested HTML elements. These are indicated in the document by HTML *tags*, enclosed in angle brackets thus: <**p**>.

In the simple, general case, the extent of an element is indicated by a pair of tags: a "start tag" <**p**> and "end tag" </**p**>. The text content of the element, if any, is placed between these tags.

Tags may also enclose further tag markup between the start and end, including a mixture of tags and text. This indicates further (nested) elements, as children of the parent element.

The start tag may also include attributes within the tag. These indicate other information, such as identifiers for sections within the document, identifiers used to bind style information to the presentation of the document, and for some tags such as the <**img**> used to embed images, the reference to the image resource.

Some elements, such as the line break <**br**>, do not permit any embedded content, either text or further tags. These require only a single empty tag (akin to a start tag) and do not use an end tag.

Many tags, particularly the closing end tag for the very commonly used paragraph element <**p**>, are optional. An HTML browser or other agent can infer the closure for the end of an element from the context and the structural rules defined by the HTML standard. These rules are complex and not widely understood by most HTML coders.

<**tag** attribute1="value1" attribute2="value2">''content''</**tag**>. Some HTML elements are defined as empty elements and take the form <**tag** attribute1="value1" attribute2="value2">. Empty elements may enclose no content, for instance, the <**br**> tag or the inline <**img**> tag. The name of an HTML element is the name used in the tags. Note that the end tag's name is preceded by a slash character, /, and that in empty elements the end tag is neither required nor allowed. If attributes are not mentioned, default values are used in each case.

**CHAPTER 5**

**MODULE EXPLAINATION**

**5.1 LIST OF MODULES:**

5.1.1 user interface module

5.1.2 Preprocessing module

5.1.3 Classification module

5.1.4 prediction module

5.1.5 Output module

**5.1.1 USER INTERFACE MODULE:**

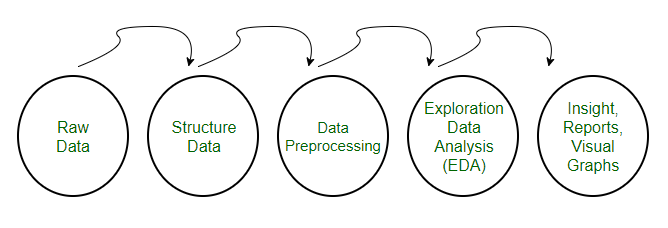
The **user interface** (**UI**), in the industrial design field of human-computer interaction, is the space where interactions between humans and machines occur. The goal of this interaction is to allow effective operation and control of the machine from the human end, whilst the machine simultaneously feeds back information that aids the operators' decision-making process. Examples of this broad concept of user interfaces include the interactive aspects of computer operating systems, hand tools, heavy machinery operator controls, and process controls. The design considerations applicable when creating user interfaces are related to or involve such disciplines as ergonomics and psychology.

Generally, the goal of user interface design is to produce a user interface which makes it easy, efficient, and enjoyable (user-friendly) to operate a machine in the way which produces the desired result. This generally means that the operator needs to provide minimal input to achieve the desired output, and also that the machine minimizes undesired outputs to the human.

User interfaces are composed of one or more layers including a **human-machine interface** (**HMI**) interfaces machines with physical input hardware such as keyboards, mice, game pads and output hardware such as computer monitors, speakers, and printers. A device that implements an HMI is called a human interface device (HID). Other terms for human-machine interfaces are **man-machine interface** (**MMI**) and when the machine in question is a computer **human-computer interface**. Additional UI layers may interact with one or more human sense, including: tactile UI (touch), visual UI (sight), auditory UI (sound), olfactory UI (smell), equilibrial UI (balance), and gustatory UI (taste).

## **5.1.2 PREPROCESSING MODULE:**

• 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.



**Fig 9. Preprocessing Diagram.**

**Need of Data Preprocessing:**  
• For achieving better results from the applied model in Machine Learning projects the format 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.  
• 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 one data set, and best out of them is chosen.

1. **Rescale Data:**  
   • When our data is comprised of attributes with varying scales, many machine learning algorithms can benefit from rescaling the attributes to all have the same scale.  
   • This is useful for optimization algorithms in used in the core of machine learning algorithms like gradient descent.  
   • It is also useful for algorithms that weight inputs like regression and neural networks and algorithms that use distance measures like K-Nearest Neighbors.  
   • We can rescale your data using scikit-learn using the MinMax Scalar class.
2. **Binarize Data (Make Binary):**  
   • We can transform our data using a binary threshold. All values above the threshold are marked 1 and all equal to or below are marked as 0.  
   • This is called binarizing your data or threshold your data. It can be useful when you have probabilities that you want to make crisp values. It is also useful when feature engineering and you want to add new features that indicate something meaningful.  
   • We can create new binary attributes in Python using scikit-learn with the Binarizer class.
3. **Standardize Data:**  
   • Standardization is a useful technique to transform attributes with a Gaussian distribution and differing means and standard deviations to a standard Gaussian distribution with a mean of 0 and a standard deviation of 1.  
   • We can standardize data using scikit-learn with the Standard Scalar class.

**5.1.3 CLASSIFICATION/REGRESSION MODULE:**

A regression problem is when the output variable is a real or continuous value, such as “salary” or “weight”. Many different models can be used, the simplest is the linear regression. It tries to fit data with the best hyper-plane which goes through the points.

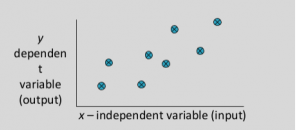


Fig 10. Regression Module.

**For Examples:**  
**Which of the following is a regression task?**

* Predicting age of a person
* Predicting nationality of a person
* Predicting whether stock price of a company will increase tomorrow
* Predicting whether a document is related to sighting of UFOs?

**Solution :**Predicting age of a person (because it is a real value, predicting nationality is categorical, whether stock price will increase is discrete-yes/no answer, predicting whether a document is related to UFO is again discrete- a yes/no answer).

**5.1.4 FEATURE SELECTION MODULE:**

A feature selection algorithm can be seen as the combination of a search technique for proposing new feature subsets, along with an evaluation measure which scores the different feature subsets. The simplest algorithm is to test each possible subset of features finding the one which minimizes the error rate. This is an exhaustive search of the space, and is computationally intractable for all but the smallest of feature sets. The choice of evaluation metric heavily influences the algorithm, and it is these evaluation metrics which distinguish between the three main categories of feature selection algorithms: wrappers, filters and embedded methods.

Univariate feature selection works by selecting the best features based on univariate statistical tests. It can be seen as a preprocessing step to an estimator. Scikit-learn exposes feature selection routines as objects that implement the transform method:

* [**SelectKBest**](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html#sklearn.feature_selection.SelectKBest) removes all but the k highest scoring features.

**5.1.5 FEATURE EXTRACTION MODULE:**

In machine learning, pattern recognition and in image processing, **feature extraction** starts from an initial set of measured data and builds derived values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations. Feature extraction is related to dimensionality reduction.

Understanding the math behind Principal Component Analysis (PCA) without a solid linear algebra foundation is challenging. When I taught Data Science at General Assembly in San Francisco, I found that helping students visualize the transformation between features and principal components greatly enhanced their understanding.**PCA is a dimensionality reduction technique that has four main parts: feature covariance, eigende composition, principal component transformation, and choosing components in terms of explained variance.**

**5.1.6 PREDICTION MODULE:**

Predictive modeling is a process that uses Data Mining and probability to forecast outcomes. Each model is made up of a number of predictors, which are variables that are likely to influence future results. Once data has been collected for relevant predictors, a statistical model is formulated. The model may employ a simple linear equation, or it may be a complex neural network, mapped out by sophisticated software. As additional data becomes available, the statistical analysis model is validated or revised.

### Modeling methods

Although it may be tempting to think that big data makes predictive models more accurate, statistical theorems show that, after a certain point, feeding more data into a predictive analytics model does not improve accuracy. Analyzing representative portions of the available information -- sampling -- can help speed development time on models and enable them to be deployed more quickly.

Once data scientists gather this sample data, they must select the right model. Linear regressions are among the simplest types of predictive models. Linear models essentially take two variables that are correlated -- one independent and the other dependent -- and plot one on the x-axis and one on the y-axis. The model applies a best fit line to the resulting data points. Data scientists can use this to predict future occurrences of the dependent variable.

**5.1.7 OUTPUT MODULE:**

Output for the given model is predicted by matching the given input and the preprocessed data. Preprocessed data consist of both training set and testing set. Given input is loaded according to the preprocessed data. Based on the comparision, output is predicted along with its accuracy and loss. This process is carried out in both the RF Algorithm, SVM Algorithm, KNN Algorithm, NB Algorithm, DT Algorithm thus it results in ROC curve (diagrammatic representation).ROC curve compares both the results and contrast its accuracy.

**5.2 ALGORITHMS USED:**

# **5.2.1 Support Vector Machine Algorithm(SVM):-**

Support Vector Machines (SVM) have recently gained prominence in the field of machine learning and pattern classification. Classification is achieved by realizing a linear or non-linear separation surface in the input space. The DirectSVM is an intuitively appealing algorithm, which builds the Support Vector set incrementally . Recently it has been proved that the closest pair of points of the opposite class are always Support Vectors. DirectSVM starts off with this pair of points in the candidate Support Vector set. The Geometric SVM proposed by us improves the scaling behavior of the DirectSVM by using an optimization based approach to add points to the candidate Support Vector set.

Three main ideas:

1. Define what an optimal hyperplane is (taking into account that it needs to be computed efficiently): maximize margin

2. Generalize to non-linearly separable problems: have a penalty term for misclassifications

3. Map data to high dimensional space where it is easier to classify with linear decision surfaces: reformulate problem so that data are mapped implicitly to this space.

Soft vs Hard margin in SVM

Soft-Margin always has a solution • Soft-Margin is more robust to outliers – Smoother surfaces (in the non-linear case) • Hard-Margin does not require to guess the cost parameter (requires no parameters at all)

• SVMs that perform regression and also clustering .

• Support Vector Machines: maximize margin while bounding the number of margin errors.

• Leave One Out Machines: minimize the bound of the leave-one-out error.

• SVM formulations that allow different cost of misclassification for different classes.

• Kernels suitable for sequences of strings, or other specialized kernels.

• Very successful.

• Other formulations exist where minimizing the number of variables is folded into the optimization problem.

• Similar algorithms for non-linear SVMs.

• Quite successful.

SVMs formulate learning as a mathematical program taking advantage of the rich theory in optimization.

• SVM uses kernels to map indirectly to extremely high dimensional spaces.

• SVMs are extremely successful, robust, efficient, and versatile, and have a good theoretical basis.

## **SVM ALGORITHM:-**

candidateSV = { closest pair from opposite classes }

**while** there are violating points **do**

Find a violator

candidateSV = candidateSV U violator

if any ap < 0 due to addition of c to S **then**

candidateSV = candidateSV \ p

repeat till all such points are pruned

**end** if

**end while**

Recursive Feature Elimination –

Train a linear SVM – Remove the x% of variables with the lowest weights (those variables affect classification the least) – Retrain the SVM with remaining variables and repeat until classification quality is reduced.

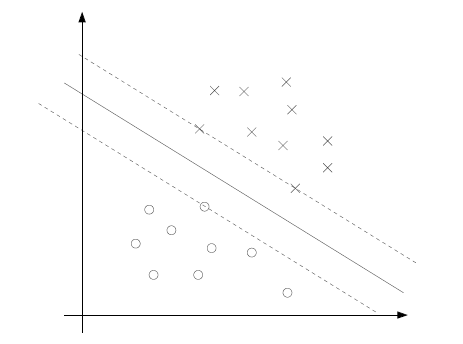


Fig 11. SVM

**5.2.2 Decision Tree Algorithm(DT):**

A decision tree is a classifier expressed as a recursive partition of the instance space. The decision tree consists of nodes that form a rooted tree, meaning it is a directed tree with a node called “root” that has no incoming edges. All other nodes have exactly one incoming edge. A node with outgoing edges is called an internal or test node. All other nodes are called leaves (also known as terminal or decision nodes). In a decision tree, each internal node splits the instance space into two or more sub-spaces according to a certain discrete function of the input attributes values. In the simplest and most free each test considers a single attribute, such that the instance space is partitioned according to the attribute’s value. In the case of numeric attributes, the condition refers to a range. Each leaf is assigned to one class representing the most appropriate target value. Alternatively, the leaf may hold a probability vector indicating the probability of the target attribute having a certain value. Instances are classified by navigating them from the root of the tree down to a leaf, according to the outcome of the tests along the path. Describes a decision tree that reasons whether or not a potential customer will respond to a direct mailing. Internal nodes are represented as circles, whereas leaves are denoted as triangles. Note that this decision tree incorporates both nominal and numeric attributes. Given this classifier, the analyst can predict the response of a potential customer (by sorting it down the tree), and understand the behavioral characteristics of the entire potential customers population regarding direct mailing. Each node is labeled with the attribute it tests, and its branches are labeled with its corresponding values.

Algorithmic framework for top–down inducing of a decision tree using growing and pruning. Note that these algorithms are greedy by nature and construct the decision tree in a top–down, recursive manner (also known as “divide and conquer“). In each iteration, the algorithm considers the partition of the training set using the outcome of a discrete function of the input attributes. The selection of the most appropriate function is made according to some splitting measures. After the selection of an appropriate split, each node further subdivides the training set into smaller subsets, until no split gains sufficient splitting measure or a stopping criteria is satisfied

## **Decision Tree Algorithm:-**

TreeGrowing (S,A,y)

Where: S - Training Set

A - Input Feature Set

y - Target Feature

Create a new tree T with a single root node.

IF One of the Stopping Criteria is fulfilled THEN

Mark the root node in T as a leaf with the most

common value of y in S as a label.

ELSE

Find a discrete function f(A) of the input

attributes values such that splitting S

according to f(A)’s outcomes (v1,...,vn) gains

the best splitting metric.

IF best splitting metric > treshold THEN

Label t with f(A)

FOR each outcome vi of f(A):

Set Subtreei= TreeGrowing (σf(A)=viS,A,y).

Connect the root node of tT to Subtreei with

an edge that is labelled as vi

END FOR

ELSE

Mark the root node in T as a leaf with the most

common value of y in S as a label.

END IF

END IF

RETURN T

TreePruning (S,T,y)

Where:

S - Training Set

y - Target Feature

T - The tree to be pruned

DO

Select a node t in T such that pruning it

maximally improve some evaluation criteria

IF t6=Ø THEN T=pruned(T,t)

UNTIL t=Ø

RETURN T

Information gain is an impurity-based criterion that uses the entropy measure (origin from information theory) as the impurity measure:

(InformationGain(ai , S) = Entropy(y, S) − P vi,j∈dom(ai) |σai=vi,j S|/ |S| · Entropy(y, σai=vi,jS)

**Gini Index :-**

Gini index is an impurity-based criterion that measures the divergences between the probability distributions of the target attribute’s values. The Gini index has been used in various works such as and it is defined as:

Gini(y, S) = 1 − X cj∈dom(y) Ã¯ ¯σy=cjS ¯ ¯/ |S| !2

**Gain Ratio**

The gain ratio “normalizes” the information gain as follows :

GainRatio(ai , S) = InformationGain(ai , S)/ Entropy(ai , S).

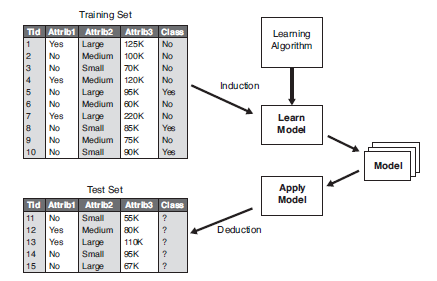


Fig 12. Decision Tree

# **5.2.3 K- Nearest Neighbor Algorithm(KNN):**

K Nearest Neighbor (kNN) algorithm based on sparse learning, so as to overcome the drawbacks of the previous kNN algorithm, such as the ﬁxed k value for each test sample and the neglect of the correlation of samples. Speciﬁcally, reconstructs test samples by training samples to learn the optimal k value for each test sample, and then uses kNN algorithm with the learnt k value to conduct all kinds of tasks, such as classiﬁcation, regression, and missing value imputation. The k Nearest Neighbors algorithm (kNN for short) is an instance-based, or a lazy learning method. It has been regarded as one of the simplest of all machine learning algorithms [4][13]. The rational of kNN is that similar samples belonging to the same class have high probability, while the key idea of kNN algorithm is to ﬁrst select k nearest neighbors for each test sample, followed by using the learnt k nearest neighbors to predict this test sample.Therefore, Knn algorithm was often thought as an algorithm, in which no explicit training step is required.

## **K- Nearest Neighbor Algorithm:-**

Input: η(0) = 0.01,α(1) = 1, γ =0 .002, ρ1, ρ2;

Output: W;

Initialize t =1 ;

Initialize W(1) as a random diagonal matrix;

repeat

while L(W(t)) >G η(t−1)(πη(t−1)(W(t)),W(t)) do

Set η(t−1) = γη(t−1);

end

Set η(t)=η(t−1);

Compute W(t + 1) = argmin W Gη(t)(W,V(t));

Compute α(t + 1) = 1+√1+4α(t)2 2 ;

Compute ;

until converges;

Input: X, Y;

Output:

switch task do

case 1

Class labels;

endsw

case 2

Predicted value;

endsw

case 3

Imputation value;

endsw

endsw

Normalizing X and Y (When Y is class labels without normalization);

Optimizing Eq. (7) to obtain the optimal solution W;

Obtaining the optimal k value for test samples based on W;

switch task do

case 1

Obtaining class labels via majority rule;

endsw

case 2

Obtaining prediction value ;

endsw

case 3

Obtaining imputation value ;

endsw

endsw

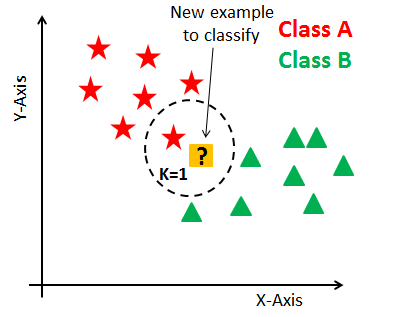


Fig 13. K-Nearest Neighbour.

# **5.3.4 Naive Bayes Algorithm(NB):**

Naïve Bayes is a subset of Bayesian decision theory. It’s called naive because the formulation makes some naïve assumptions. Python’s text-processing abilities which split up a document into a vector are used. This can be used to classify text. Classifies may put into human-readable form. It is a popular classification method in addition to conditional independence, overfitting , and Bayesian methods. In the face of the simplicity of Naive Bayes , it can classify documents surprisingly well. Instinctively a potential justification for the conditional independence assumption is that if the document is about politics, this is a good evidence of the kinds of other words found in the document. Naive Bayes is a reasonable classifier in this sense and has minimal storage and fast training, it is applied to time-storage critical applications, such as automatically classifying web pages into types and spam filtering. Naive Bayes is a classification algorithm which is based on Bayes theorem with strong and naïve independence assumptions. It simplifies learning by assuming that features are independent of given class. This paper surveys about naïve Bayes algorithm, which describes its concept, hidden naïve Bayes, text classification, traditional naïve Bayes and machine learning. Considering a set of objects, each of which belongs to a known class, and each of which has a known vector of variables, the aim is to create a rule which enables to allocate future objects to a class, given just the vectors of variables marking out the future objects. These problems are known as supervised classification problem‖, are worldwide, and most of the methods for constructing such rules have been developed. It is very easy to establish, and no need any complicated repetitive parameter estimation schemes. This means it should be applied to huge data sets. It is easy to interpret, so unskilled users in classifier technology can make out the reason for it is making the classification it makes. Finally, it often does surprisingly well: it should not be the best possible classifier in any particular application, but it can usually be relied on to be robust and to do well. Classification is a fundamental issue in machine learning and data mining. In classification, the goal of a learning algorithm is to construct a classifier given a set of training examples with class labels. Regularly example E is represented by attribute values by a tuple(x1,x2,···,xn), where xiis the value of attribute Xi. Let C represent the classification variable, and let c be the value of C. There are only two classes here: +(the positive class) or (the negative class).

## **Naive bayes Algorithm:-**

Derivation: D : Set of tuples

Each Tuple is an ‘n’ dimensional attribute vector

X : (x1,x2,x3,…. xn) Let there be ‘m’ Classes : C1,C2,C3…Cm Naïve Bayes classifier predicts X belongs to Class Ci iff

P (Ci/X) > P(Cj/X) for 1<= j <= m , j <> i Maximum Posteriori Hypothesis

P(Ci/X) = P(X/Ci) P(Ci) / P(X)

Maximize P(X/Ci) P(Ci) as P(X) is constant

With many attributes, it is computationally expensive to evaluate P(X/Ci).

Naïve Assumption of “class conditional independence” ∏ = = n k k xCiCiPPX 1 /))(/.( P(X/Ci) = P(x1/Ci) \* P(x2/Ci) \*…\* P(xn/ Ci)

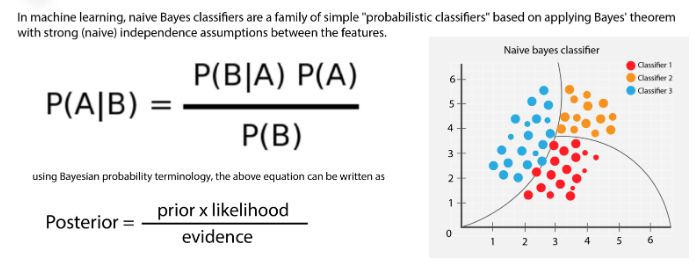


Fig 14. Naïve Bayes.

# **5.3.5 Linear regression Algorithm(LR):**

Linear regression analysis can be divided into simple linear regression and multiple linear regression. The paper will mainly analyze simple linear regression model that is the analysis method of studying the relations between independent variable and dependent variable. We will set the model of dependent variable y and the independent variable (i=1,2,3……) that will influence the variable y and the predict the development trend of y ,Simple linear regression model will be expressed as followed: a ax e y is the dependent variable and x is the independent+ +=0 1 y variable. 0 a , the constant term, is the intercept of the regression line on the vertical axis and 1 a is regression coefficient that is the slope of the regression line. e is the random error which will be used to express the effect of random factors on dependent variable.

Linear regression assumes a linear or straight line relationship between the input variables (X) and the single output variable (y).More specifically, that output (y) can be calculated from a linear combination of the input variables (X). When there is a single input variable, the method is referred to as a simple linear regression.In simple linear regression we can use statistics on the training data to estimate the coefficients required by the model to make predictions on new data.

The line for a simple linear regression model can be written as:

y = b0 + b1 \* x

where b0 and b1 are the coefficients we must estimate from the training data.Once the coefficients are known, we can use this equation to estimate output values for y given new input examples of x.

## Linear regression Algorithm:-

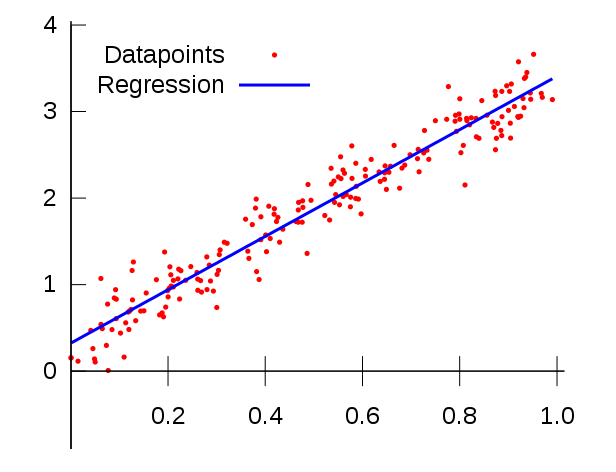


Fig 15. Linear Regression.

# **5.3.6 Random Forest Algorithm(RF):**

Random Forests were introduced by Leo Breiman who was inspired by earlier work by Amit and Geman. Although not obvious from the description in , Random Forests are an extension of Breiman’s bagging idea and were developed as a competitor to boosting. Random Forests can be used for either a categorical response variable, referred to in as “classiﬁcation”, or a continuous response, re- ferred to as “regression”. Similarly, the predictor variables can be either categorical or continuous. From a computational standpoint,

Random Forests are appealing because they

• naturally handle both regression and (multiclass) classiﬁcation;

• are relatively fast to train and to predict;

• depend only on one or two tuning parameters;

• have a built in estimate of generalization error;

• can be used directly for high-dimensional problems;

• can easily be implemented in parallel.

Statistically, Random Forests are appealing because of the additional features they provide, such as

• measures of variable importance;

• differential class weighting;

• missing value imputation;

• visualization;

• outlier detection;

• unsupervised learning.

## Random forest Algorithm:-

Algorithm Random Forests

Let D = {(x1,y1),...,(xN,yN)} denote the training data, with xi = (xi,1,...,xi,p)T.

For j = 1 to J:

1. Take a bootstrap sample Dj of size N from D.

2. Using the bootstrap sample Dj as the training data, ﬁt a tree using binary recursive partitioning :

a. Start with all observations in a single node.

b. Repeat the following steps recursively for each unsplit node until the stopping criterion is met:

i. Select m predictors at random from the p available predictors.

ii. Find the best binary split among all binary splits on the m predictors from step i. iii. Split the node into two descendant nodes using the split from step ii.

To make a prediction at a new point x,

• ˆ f(x) = 1 J ∑J j=1 ˆ hj(x) for regression

• ˆ f(x) = argmaxy∑J j=1 I(ˆ hj(x) = y) for classiﬁcation

where ˆ hj(x) is the prediction of the response variable at x using the jth tree.

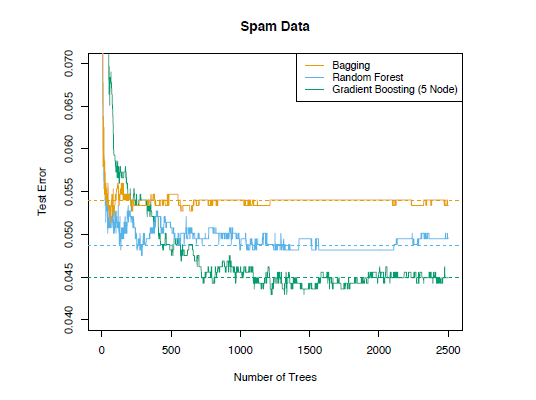


Fig 16. Random Forest.

**CHAPTER-6**

6. **CONCLUSION AND FUTURE ENHANCEMENT**

**6.1. CONCLUSION:**

Chronic Kidney Disease (CKD) should be diagnosed earlier before kidneys fail to work. To help doctors or medical experts in prediction of CKD among patients easily, this paper has developed a Chronic Kidney Disease Prediction System (CKDPS) using Random Forest Algorithm. Random Forest Algorithm is a machine learning algorithm that combines decision trees to get more accurate and stable prediction.

**6.2. FUTURE ENHANCEMENTS:**

Research and analysis show that the proposed approach is a valuable diagnosing system for the doctors and medical experts to detect the Chronic Kidney Disease. In future work, it would be interesting to include additional feature information with the dietary plans based on the stages of the disease. Another future line would be the detection of various kidney related diseases.

**APPENDIX-A (CODING)**

**FULLPROCESS.py**

import pandas as pd

dataset1=pd.read\_csv("prep.csv",index\_col=None)

df2=dataset1

from sklearn.tree import export\_graphviz #plot tree

from sklearn.metrics import roc\_curve, auc #for model evaluation

from sklearn.metrics import classification\_report #for model evaluation

from sklearn.metrics import confusion\_matrix #for model evaluation

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

#X\_train, X\_test, y\_train, y\_test = train\_test\_split(df2.drop('classification\_yes', 1), df2['classification\_yes'], test\_size = .2, random\_state=10)

import time

import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler

from sklearn.feature\_selection import SelectKBest

from sklearn.feature\_selection import chi2

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

from sklearn.decomposition import PCA

from sklearn.feature\_selection import RFE

from sklearn.linear\_model import LogisticRegression

import pickle

import matplotlib.pyplot as plt

df2 = pd.get\_dummies(df2, drop\_first=True)

def selectkbest(indep\_X,dep\_Y):

test = SelectKBest(score\_func=chi2, k=5)

fit1= test.fit(indep\_X,dep\_Y)

# summarize scores

features = indep\_X.columns.values.tolist()

np.set\_printoptions(precision=2)

print(features)

print(fit1.scores\_)

#plt.figure(figsize=(12,3))

#plt.bar(fit1.scores\_,height=0.6)

feature\_series = pd.Series(data=fit1.scores\_,index=features)

feature\_series.plot.bar()

selectk\_features = fit1.transform(indep\_X)

return selectk\_features

def rfeFeature(indep\_X,dep\_Y):

model = LogisticRegression(solver='lbfgs')

rfe = RFE(model, 5)

fit3 = rfe.fit(indep\_X, dep\_Y)

rfe\_feature=fit3.transform(indep\_X)

features = indep\_X.columns.values.tolist()

#feature\_series = pd.Series(data=rfe\_feature,index=features)

#feature\_series.plot.bar()

return rfe\_feature

def pca(features,dep\_Y):

pca = PCA(n\_components=5)

fit2 = pca.fit(features)

pca\_feature=fit2.transform(features)

return pca\_feature

def svm(features,indep\_X,dep\_Y):

X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, dep\_Y, test\_size = 0.25, random\_state = 0)

#X\_train, X\_test, y\_train, y\_test = train\_test\_split(rfe\_feature, dep\_Y, test\_size = 0.25, random\_state = 0)

#X\_train, X\_test, y\_train, y\_test = train\_test\_split(pca\_feature, dep\_Y, test\_size = 0.25, random\_state = 0)

#X\_train, X\_test, y\_train, y\_test = train\_test\_split(feature\_import, dep\_Y, test\_size = 0.25, random\_state = 0)

#X\_train, X\_test, y\_train, y\_test = train\_test\_split(indep\_X,dep\_Y, test\_size = 0.25, random\_state = 0)

#Feature Scaling

#from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

# Fitting K-NN to the Training set

from sklearn.svm import SVC

classifier = SVC(kernel = 'rbf', random\_state = 0)

classifier.fit(X\_train, y\_train)

# Predicting the Test set results

y\_pred = classifier.predict(X\_test)

# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix

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

from sklearn.metrics import accuracy\_score

from sklearn.metrics import classification\_report

#from sklearn.metrics import confusion\_matrix

#cm = confusion\_matrix(y\_test, y\_pred)

Accuracy=accuracy\_score(y\_test, y\_pred )

report=classification\_report(y\_test, y\_pred)

return classifier,Accuracy,report,X\_test,y\_test,cm

def naives(features,indep\_X,dep\_Y):

#X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, dep\_Y, test\_size = 0.25, random\_state = 0)

#X\_train, X\_test, y\_train, y\_test = train\_test\_split(rfe\_feature, dep\_Y, test\_size = 0.25, random\_state = 0)

#X\_train, X\_test, y\_train, y\_test = train\_test\_split(pca\_feature, dep\_Y, test\_size = 0.25, random\_state = 0)

#X\_train, X\_test, y\_train, y\_test = train\_test\_split(feature\_import, dep\_Y, test\_size = 0.25, random\_state = 0)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(indep\_X,dep\_Y, test\_size = 0.25, random\_state = 0)

#Feature Scaling

#from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

# Fitting K-NN to the Training set

from sklearn.naive\_bayes import GaussianNB

classifier = GaussianNB()

classifier.fit(X\_train, y\_train)

# Predicting the Test set results

y\_pred = classifier.predict(X\_test)

# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix

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

from sklearn.metrics import accuracy\_score

from sklearn.metrics import classification\_report

#from sklearn.metrics import confusion\_matrix

#cm = confusion\_matrix(y\_test, y\_pred)

Accuracy=accuracy\_score(y\_test, y\_pred )

report=classification\_report(y\_test, y\_pred)

return classifier,Accuracy,report,X\_test,y\_test,cm

def Decision(features,indep\_X,dep\_Y):

X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, dep\_Y, test\_size = 0.25, random\_state = 0)

#X\_train, X\_test, y\_train, y\_test = train\_test\_split(rfe\_feature, dep\_Y, test\_size = 0.25, random\_state = 0)

#X\_train, X\_test, y\_train, y\_test = train\_test\_split(pca\_feature, dep\_Y, test\_size = 0.25, random\_state = 0)

#X\_train, X\_test, y\_train, y\_test = train\_test\_split(feature\_import, dep\_Y, test\_size = 0.25, random\_state = 0)

#X\_train, X\_test, y\_train, y\_test = train\_test\_split(indep\_X,dep\_Y, test\_size = 0.25, random\_state = 0)

#Feature Scaling

#from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

# Fitting K-NN to the Training set

from sklearn.linear\_model import LogisticRegression

classifier = LogisticRegression(random\_state = 0)

classifier.fit(X\_train, y\_train)

# Predicting the Test set results

y\_pred = classifier.predict(X\_test)

# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix

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

from sklearn.metrics import accuracy\_score

from sklearn.metrics import classification\_report

#from sklearn.metrics import confusion\_matrix

#cm = confusion\_matrix(y\_test, y\_pred)

Accuracy=accuracy\_score(y\_test, y\_pred )

report=classification\_report(y\_test, y\_pred)

return classifier,Accuracy,report,X\_test,y\_test,cm

def knn(features,indep\_X,dep\_Y):

X\_train1, X\_test1, y\_train, y\_test = train\_test\_split(features, dep\_Y, test\_size = 0.25, random\_state = 0)

#X\_train, X\_test, y\_train, y\_test = train\_test\_split(rfe\_feature, dep\_Y, test\_size = 0.25, random\_state = 0)

#X\_train, X\_test, y\_train, y\_test = train\_test\_split(pca\_feature, dep\_Y, test\_size = 0.25, random\_state = 0)

#X\_train, X\_test, y\_train, y\_test = train\_test\_split(feature\_import, dep\_Y, test\_size = 0.25, random\_state = 0)

#X\_train, X\_test, y\_train, y\_test = train\_test\_split(indep\_X,dep\_Y, test\_size = 0.25, random\_state = 0)

#Feature Scaling

#from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train1)

X\_test = sc.transform(X\_test1)

# Fitting K-NN to the Training set

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n\_neighbors = 5, metric = 'minkowski', p = 2)

classifier.fit(X\_train, y\_train)

# Predicting the Test set results

y\_pred = classifier.predict(X\_test)

# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix

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

from sklearn.metrics import accuracy\_score

from sklearn.metrics import classification\_report

#from sklearn.metrics import confusion\_matrix

#cm = confusion\_matrix(y\_test, y\_pred)

Accuracy=accuracy\_score(y\_test, y\_pred )

report=classification\_report(y\_test, y\_pred)

return classifier,Accuracy,report,X\_test,y\_test,cm

def random(features,indep\_X,dep\_Y):

X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, dep\_Y, test\_size = 0.25, random\_state = 0)

#X\_train, X\_test, y\_train, y\_test = train\_test\_split(rfe\_feature, dep\_Y, test\_size = 0.25, random\_state = 0)

#X\_train, X\_test, y\_train, y\_test = train\_test\_split(pca\_feature, dep\_Y, test\_size = 0.25, random\_state = 0)

#X\_train, X\_test, y\_train, y\_test = train\_test\_split(feature\_import, dep\_Y, test\_size = 0.25, random\_state = 0)

#X\_train, X\_test, y\_train, y\_test = train\_test\_split(indep\_X,dep\_Y, test\_size = 0.25, random\_state = 0)

#Feature Scaling

#from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

# Fitting K-NN to the Training set

from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(n\_estimators = 10, criterion = 'entropy', random\_state = 0)

classifier.fit(X\_train, y\_train)

# Predicting the Test set results

y\_pred = classifier.predict(X\_test)

# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix

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

from sklearn.metrics import accuracy\_score

from sklearn.metrics import classification\_report

#from sklearn.metrics import confusion\_matrix

#cm = confusion\_matrix(y\_test, y\_pred)

Accuracy=accuracy\_score(y\_test, y\_pred )

report=classification\_report(y\_test, y\_pred)

return classifier,Accuracy,report,X\_test,y\_test,cm

indep\_X=df2.drop('classification\_yes', 1)

dep\_Y=df2['classification\_yes']

selectk\_feature=selectkbest(indep\_X,dep\_Y)

rfe\_feature=rfeFeature(indep\_X,dep\_Y)

selectk\_pca=pca(selectk\_feature,dep\_Y)

rfe\_pca=pca(rfe\_feature,dep\_Y)

"""SVM"""

classifier,Accuracy,report,X\_test,y\_test,cm=svm(selectk\_feature,indep\_X,dep\_Y)

classifier,Accuracy,report,X\_test,y\_test,cm=svm(rfe\_feature,indep\_X,dep\_Y)

classifier,Accuracy,report,X\_test,y\_test,cm=svm(selectk\_pca,indep\_X,dep\_Y)

classifier,Accuracy,report,X\_test,y\_test,cm=svm(rfe\_pca,indep\_X,dep\_Y)

"""Random Forest"""

classifier,Accuracy,report,X\_test,y\_test,cm=random(selectk\_feature,indep\_X,dep\_Y)

classifier,Accuracy,report,X\_test,y\_test,cm=random(rfe\_feature,indep\_X,dep\_Y)

classifier,Accuracy,report,X\_test,y\_test,cm=random(selectk\_pca,indep\_X,dep\_Y)

classifier,Accuracy,report,X\_test,y\_test,cm=random(rfe\_pca,indep\_X,dep\_Y)

"""Decision Tree"""

classifier,Accuracy,report,X\_test,y\_test,cm=Decision(selectk\_feature,indep\_X,dep\_Y)

classifier,Accuracy,report,X\_test,y\_test,cm=Decision(rfe\_feature,indep\_X,dep\_Y)

classifier,Accuracy,report,X\_test,y\_test,cm=Decision(selectk\_pca,indep\_X,dep\_Y)

classifier,Accuracy,report,X\_test,y\_test,cm=Decision(rfe\_pca,indep\_X,dep\_Y)

"""Navies bay"""

classifier,Accuracy,report,X\_test,y\_test,cm=naives(selectk\_feature,indep\_X,dep\_Y)

classifier,Accuracy,report,X\_test,y\_test,cm=naives(rfe\_feature,indep\_X,dep\_Y)

classifier,Accuracy,report,X\_test,y\_test,cm=naives(selectk\_pca,indep\_X,dep\_Y)

classifier,Accuracy,report,X\_test,y\_test,cm=naives(rfe\_pca,indep\_X,dep\_Y)

"""knn"""

classifier,Accuracy,report,X\_test,y\_test,cm=knn(selectk\_feature,indep\_X,dep\_Y)

classifier,Accuracy,report,X\_test,y\_test,cm=knn(rfe\_feature,indep\_X,dep\_Y)

classifier,Accuracy,report,X\_test,y\_test,cm=knn(selectk\_pca,indep\_X,dep\_Y)

classifier,Accuracy,report,X\_test,y\_test,cm=knn(rfe\_pca,indep\_X,dep\_Y)

**Figgy.py**

import pandas as pd

from sklearn.tree import export\_graphviz #plot tree

from sklearn.metrics import roc\_curve, auc #for model evaluation

from sklearn.metrics import classification\_report #for model evaluation

from sklearn.metrics import confusion\_matrix #for model evaluation

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

#X\_train, X\_test, y\_train, y\_test = train\_test\_split(df2.drop('classification\_yes', 1), df2['classification\_yes'], test\_size = .2, random\_state=10)

import time

import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler

from sklearn.feature\_selection import SelectKBest

from sklearn.feature\_selection import chi2

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

from sklearn.decomposition import PCA

from sklearn.feature\_selection import RFE

from sklearn.linear\_model import LogisticRegression

import pickle

import matplotlib.pyplot as plt

class ckd:

def \_\_init\_\_(self):

pass

def selectkbest(self,indep\_X,dep\_Y):

test = SelectKBest(score\_func=chi2, k=5)

fit1= test.fit(indep\_X,dep\_Y)

# summarize scores

features = indep\_X.columns.values.tolist()

np.set\_printoptions(precision=2)

print(features)

print(fit1.scores\_)

#plt.figure(figsize=(12,3))

#plt.bar(fit1.scores\_,height=0.6)

feature\_series = pd.Series(data=fit1.scores\_,index=features)

feature\_series.plot.bar()

selectk\_features = fit1.transform(indep\_X)

return selectk\_features

def pca(self,features,dep\_Y):

pca = PCA(n\_components=5)

fit2 = pca.fit(features)

pca\_feature=fit2.transform(features)

return pca\_feature

def svm(self,features,indep\_X,dep\_Y):

X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, dep\_Y, test\_size = 0.25, random\_state = 0)

#X\_train, X\_test, y\_train, y\_test = train\_test\_split(rfe\_feature, dep\_Y, test\_size = 0.25, random\_state = 0)

#X\_train, X\_test, y\_train, y\_test = train\_test\_split(pca\_feature, dep\_Y, test\_size = 0.25, random\_state = 0)

#X\_train, X\_test, y\_train, y\_test = train\_test\_split(feature\_import, dep\_Y, test\_size = 0.25, random\_state = 0)

#X\_train, X\_test, y\_train, y\_test = train\_test\_split(indep\_X,dep\_Y, test\_size = 0.25, random\_state = 0)

#Feature Scaling

#from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

# Fitting K-NN to the Training set

from sklearn.svm import SVC

classifier = SVC(kernel = 'rbf', random\_state = 0)

classifier.fit(X\_train, y\_train)

# Predicting the Test set results

y\_pred = classifier.predict(X\_test)

# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix

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

from sklearn.metrics import accuracy\_score

from sklearn.metrics import classification\_report

#from sklearn.metrics import confusion\_matrix

#cm = confusion\_matrix(y\_test, y\_pred)

Accuracy=accuracy\_score(y\_test, y\_pred )

report=classification\_report(y\_test, y\_pred)

return classifier,Accuracy,report,X\_test,y\_test,cm

def random(self,features,indep\_X,dep\_Y):

X\_train, X\_test1, y\_train, y\_test = train\_test\_split(features, dep\_Y, test\_size = 0.25, random\_state = 0)

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test1)

from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(n\_estimators = 10, criterion = 'entropy', random\_state = 0,max\_depth=5)

classifier.fit(X\_train, y\_train)

# Predicting the Test set results

y\_pred = classifier.predict(X\_test)

# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix

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

from sklearn.metrics import accuracy\_score

from sklearn.metrics import classification\_report

#from sklearn.metrics import confusion\_matrix

#cm = confusion\_matrix(y\_test, y\_pred)

Accuracy=accuracy\_score(y\_test, y\_pred )

report=classification\_report(y\_test, y\_pred)

return classifier,Accuracy,report,X\_test,y\_test,cm,X\_test1

dataset1=pd.read\_csv("prep.csv",index\_col=None)

dicc={'yes':1,'no':0}

dataset1['classification']=dataset1['classification'].replace(dicc)

df2=dataset1

df2=df2.drop('classification', 1)

df2 = pd.get\_dummies(df2, drop\_first=True)

indep\_X=df2

dep\_Y=dataset1['classification']

obj=ckd()

selectk\_feature=obj.selectkbest(indep\_X,dep\_Y)

columns=['bgr','bu','sc','pcv','wbc']

selectk\_feature=pd.DataFrame(selectk\_feature)

selectk\_feature.columns=columns

#selectk\_pca=obj.pca(selectk\_feature,dep\_Y)

"""Random Forest"""

classifier,Accuracy,report,X\_test,y\_test,cm,X\_test1=obj.random(selectk\_feature,indep\_X,dep\_Y)

#classifier,Accuracy,report,X\_test,y\_test,cm=obj.svm(selectk\_feature,indep\_X,dep\_Y)

out=classifier.predict([[153,76,3.3,38.8689,8408.19]])

import eli5 #for purmutation importance

from eli5.sklearn import PermutationImportance

import shap #for SHAP values

from pdpbox import pdp, info\_plots #for partial plots

np.random.seed(123) #ensure reproduc

perm = PermutationImportance(classifier, random\_state=1).fit(X\_test, y\_test)

print(eli5.format\_as\_text(eli5.explain\_weights(perm)))

eli5.show\_weights(perm, feature\_names = X\_test1.columns.tolist())

def blockbox(model, patient):

explainer = shap.TreeExplainer(model)

shap\_values = explainer.shap\_values(patient)

shap.initjs()

return shap.force\_plot(explainer.expected\_value[1], shap\_values[1], patient,matplotlib=True,show=False)

#Xtestt=pd.DataFrame(X\_test1)

plt.clf()

data = np.array([140,23,0.6,48,5800])

sc = StandardScaler()

data = sc.fit\_transform(data.reshape(-1,1))

# providing an index

ser = pd.DataFrame(data, index =['bgr','bu','sc','pcv','wbc'])

ss=ser.T.squeeze()

#data\_for\_prediction = X\_test1.iloc[0,:].astype(float)

#data\_for\_prediction =obj.pca(np.array(data\_for\_prediction),y\_test)

blockbox(classifier, ss)

plt.savefig("force\_plot.png",dpi=150, bbox\_inches='tight')

**Base.html**

{% load static %}

<!DOCTYPE html>

<html lang="en" dir="ltr">

<head>

<meta charset="utf-8">

<title></title>

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">

<link rel="stylesheet" href="/css/style.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>

<style>

body {

background-image: url("{% static 'matrix.PNG' %}");

background-repeat: no-repeat;

/\* background-attachment: ;\*/

/\* margin-top:300px;\*/

background-size: 40% 100%;

}

.navbar-default {

background-color: #F9CD05;

border-color: #F9CD05;

}

</style>

</head>

<body>

<nav class="navbar navbar-expand-sm navbar-default navbar-dark">

<!-- Brand -->

<a class="navbar-brand" href="{% url 'ckdApp:ckd' %}">Home</a>

<!-- Links -->

<ul class="navbar-nav">

<li class="nav-item">

<a class="nav-link" href="{% url 'ckdApp:ckd' %}">CKD</a>

</li>

</ul>

</nav>

<div class="container">

{% block body\_block %}

{% endblock %}

</div>

</body>

</html>

**Create.html**

{% extends 'base.html' %}

{% load crispy\_forms\_tags %}

{% block body\_block %}

<style type="text/css">

.jumbotron

{

background-color: #fff;

margin-left: 400px;

}

</style>

<div class="jumbotron">

<center>

<h1>Chronic Kidney Disease</h1>

<head>

Please enter repective fields

</head>

</center>

<form method = "post" enctype="multipart/form-data">

{% csrf\_token %}

{{ form|crispy }}

<center> <button type="submit" class="btn btn-primary">PREDICT</button></center>

</form>

</div>

{% endblock %}

**Success.html**

{% extends 'base.html' %}

{% block body\_block %}

{% load static %}

<style type="text/css">

.jumbotron

{

background-color: #fff0;

margin-left: 450px;

}

.fieldWrapper

{

font-size:20px;

}

</style>

<div class="jumbotron">

<center>

<h1>Chronic Kidney Disease </h1>

</center>

<div class="fieldWrapper">

<center> <h2>For the values</h2></center>

<p>Blood Glucose Random:{{data\_bgr}}</p>

<p>Blood Urea:{{data\_bu}}</p>

<p>Serum Creatine:{{data\_sc}}</p>

<p>Packed cell volume:{{data\_pcv}}</p>

<p>White blood count:{{data\_wc}}</p>

<br>

{% if out == 1 %}

<p>You <b>will have</b> chronic Kidney Disease in the future:<b>Positive</b></p>

{% else %}

<p>You <b>will not</b> have chronic Kidney Disease in the future: <b>Negative</b></p>

{% endif %}

<br>

<br>

<img class="mx-auto d-block" style="width: 900px; height: 450px" alt="posneg" src= "{% static 'force\_plot.png' %}"/>

<br>

<br>

<a href="{% url 'ckdApp:ckd' %}">CKD Predicition</a>

<br>

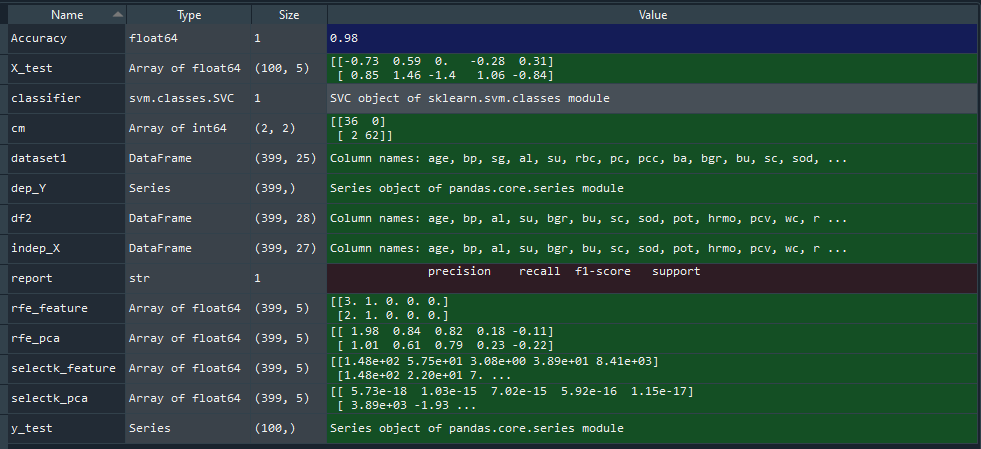
</div>

{% endblock %}

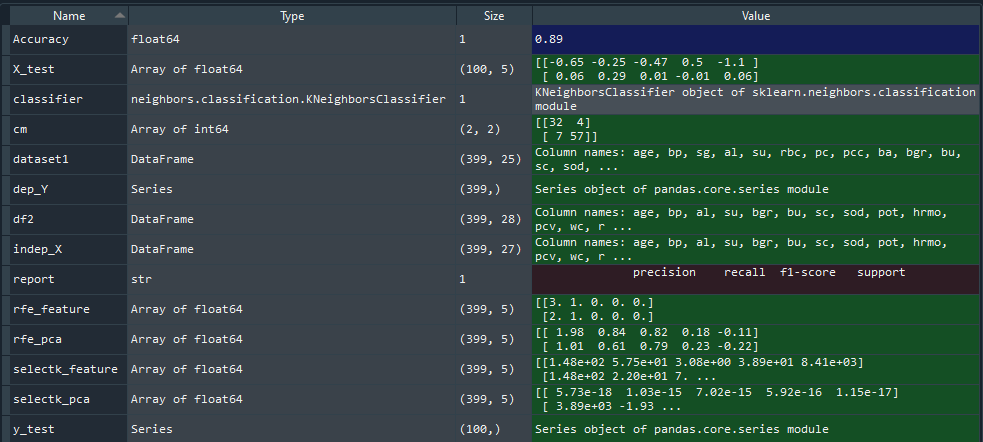
**APPENDIX-B(OUTPUT)**

**ACCURACY COMPARISION**

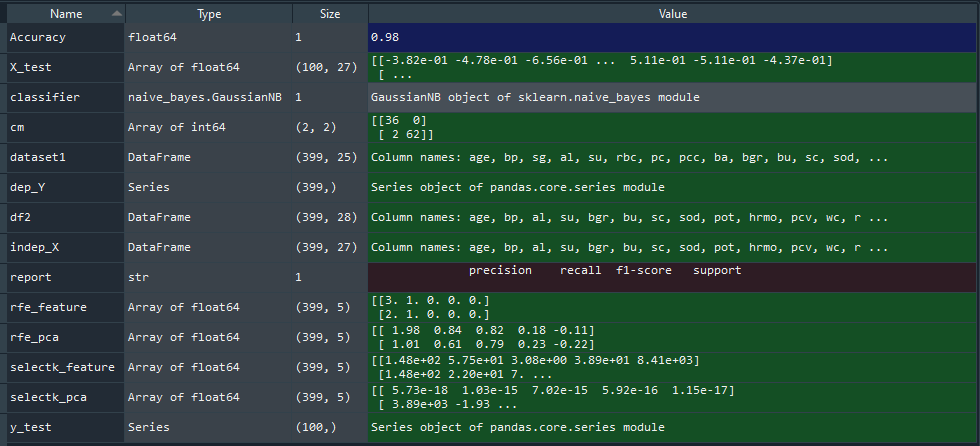
**SVM:**



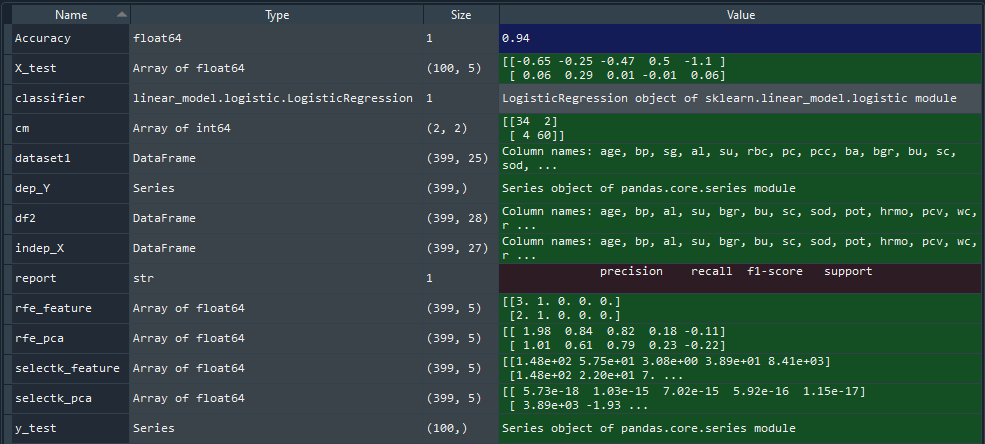
**KNN:**



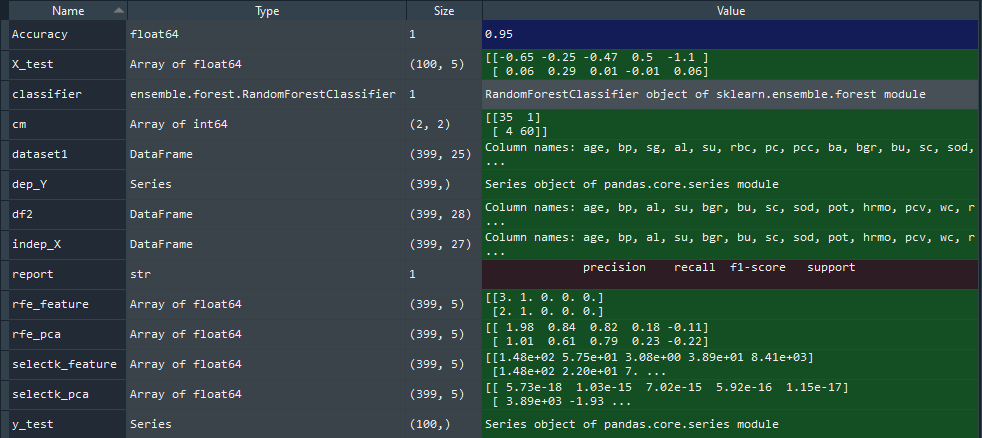
**NB:**



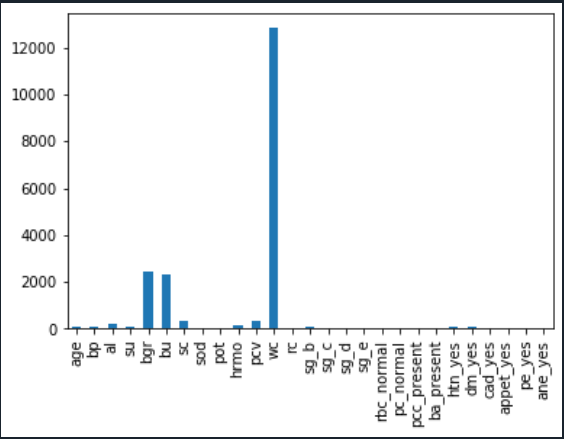
**DT:**



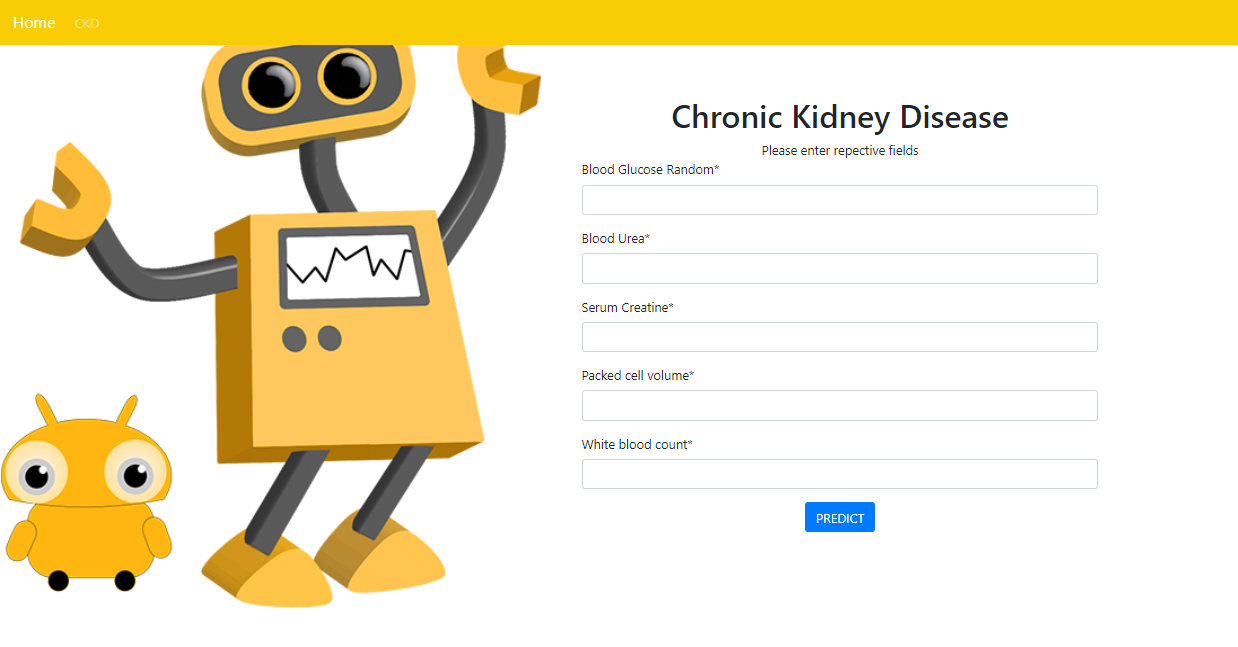
**RF:**

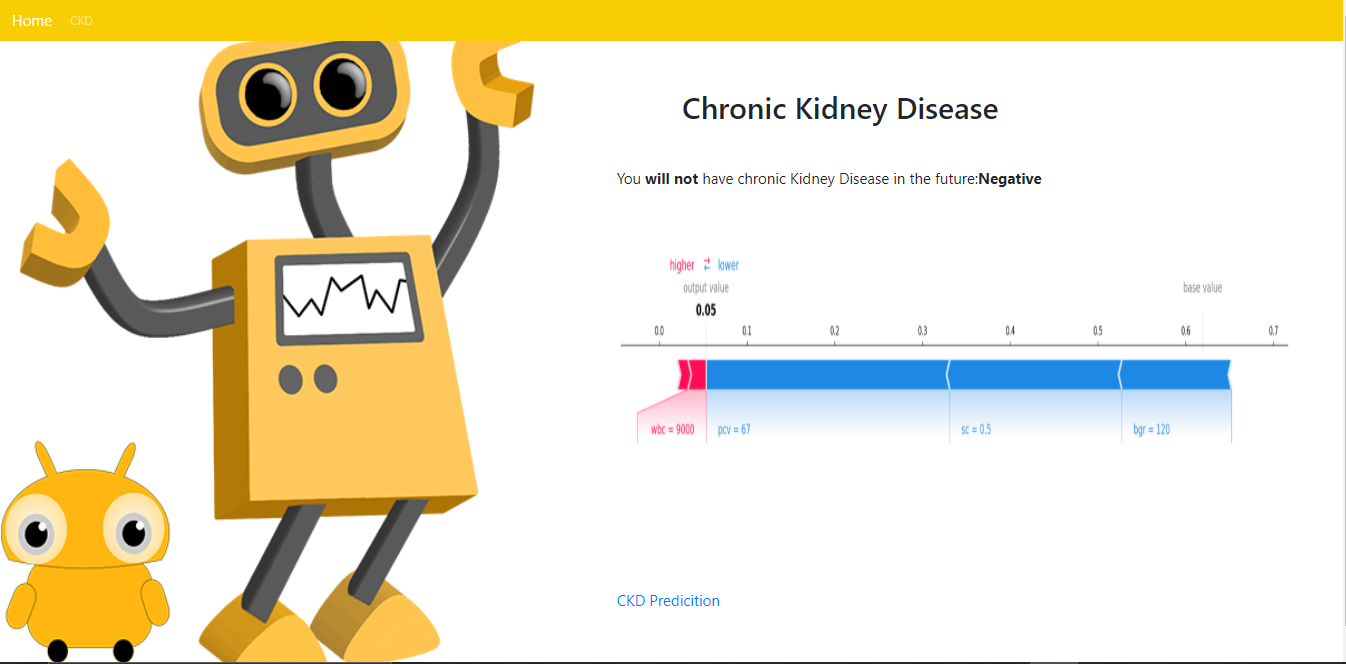


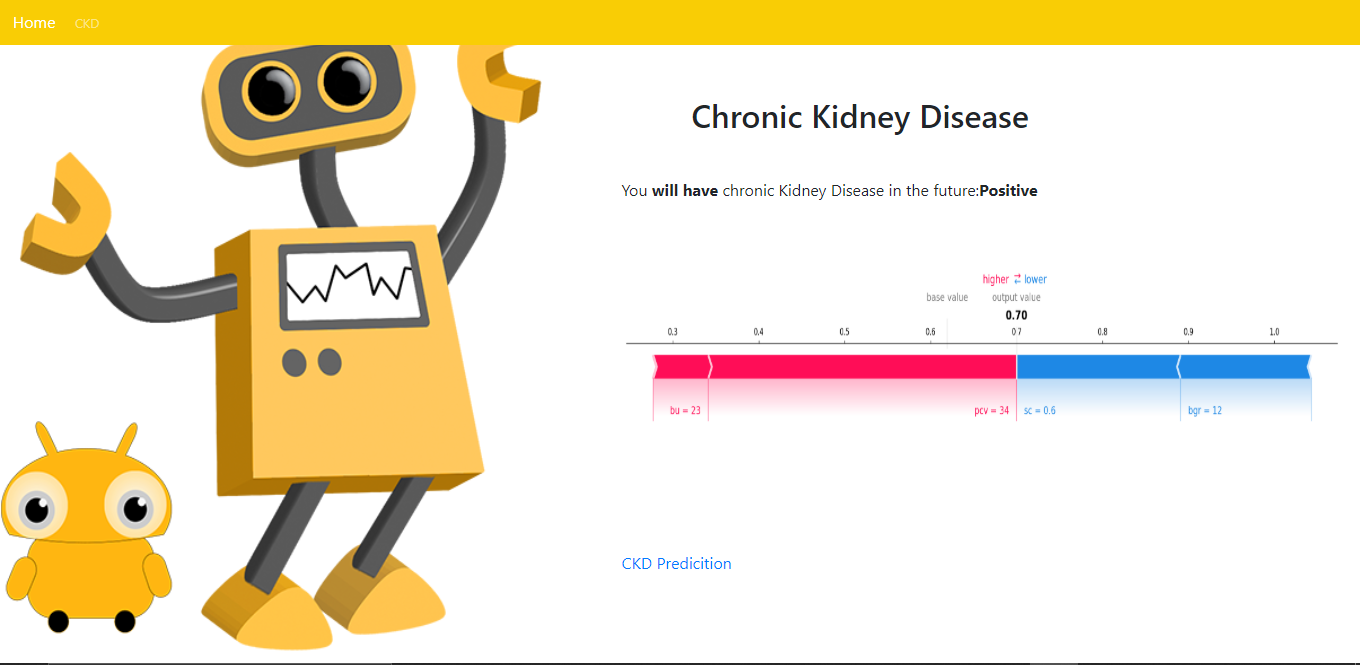
**Variable Plot:**



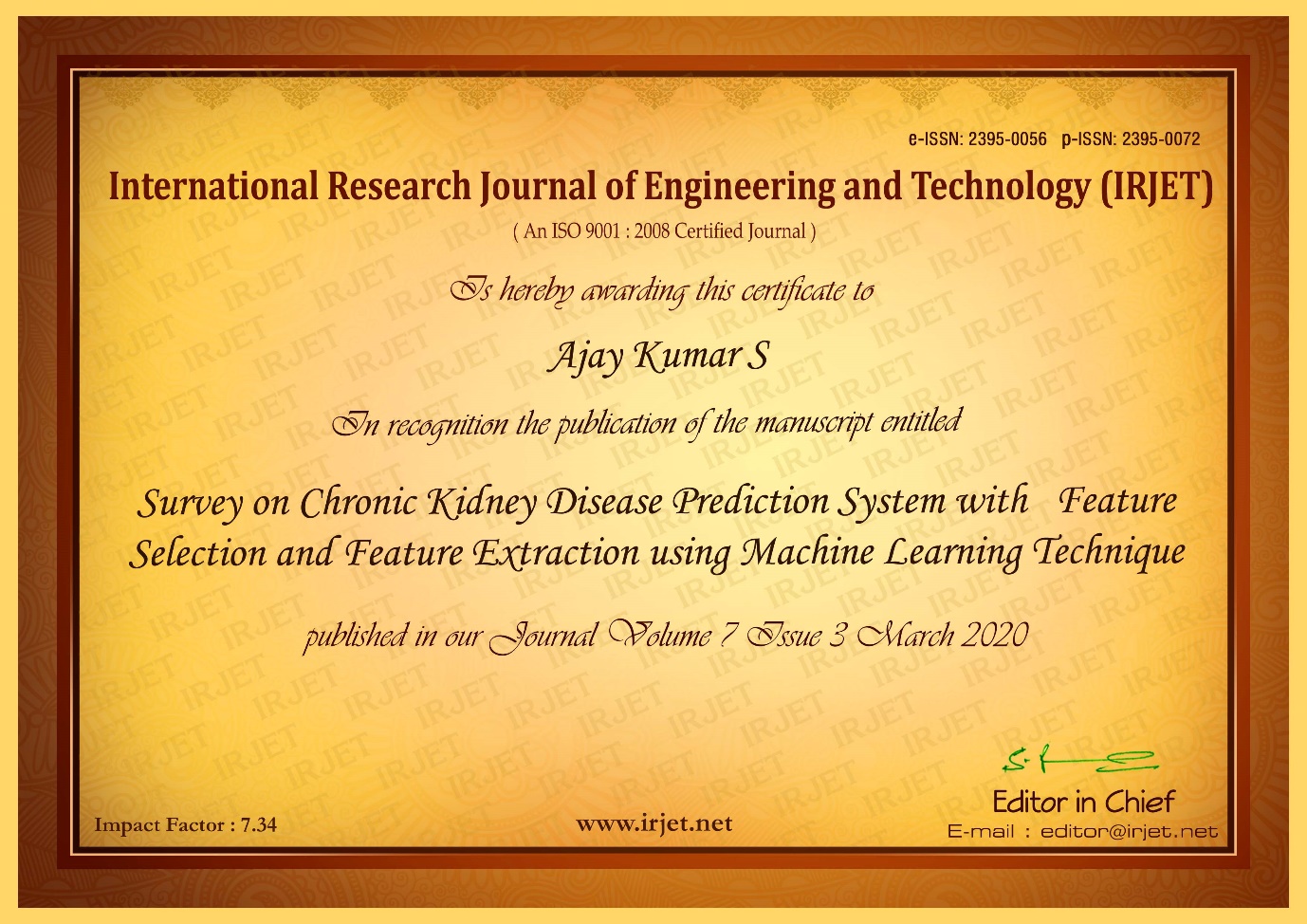
**Output**

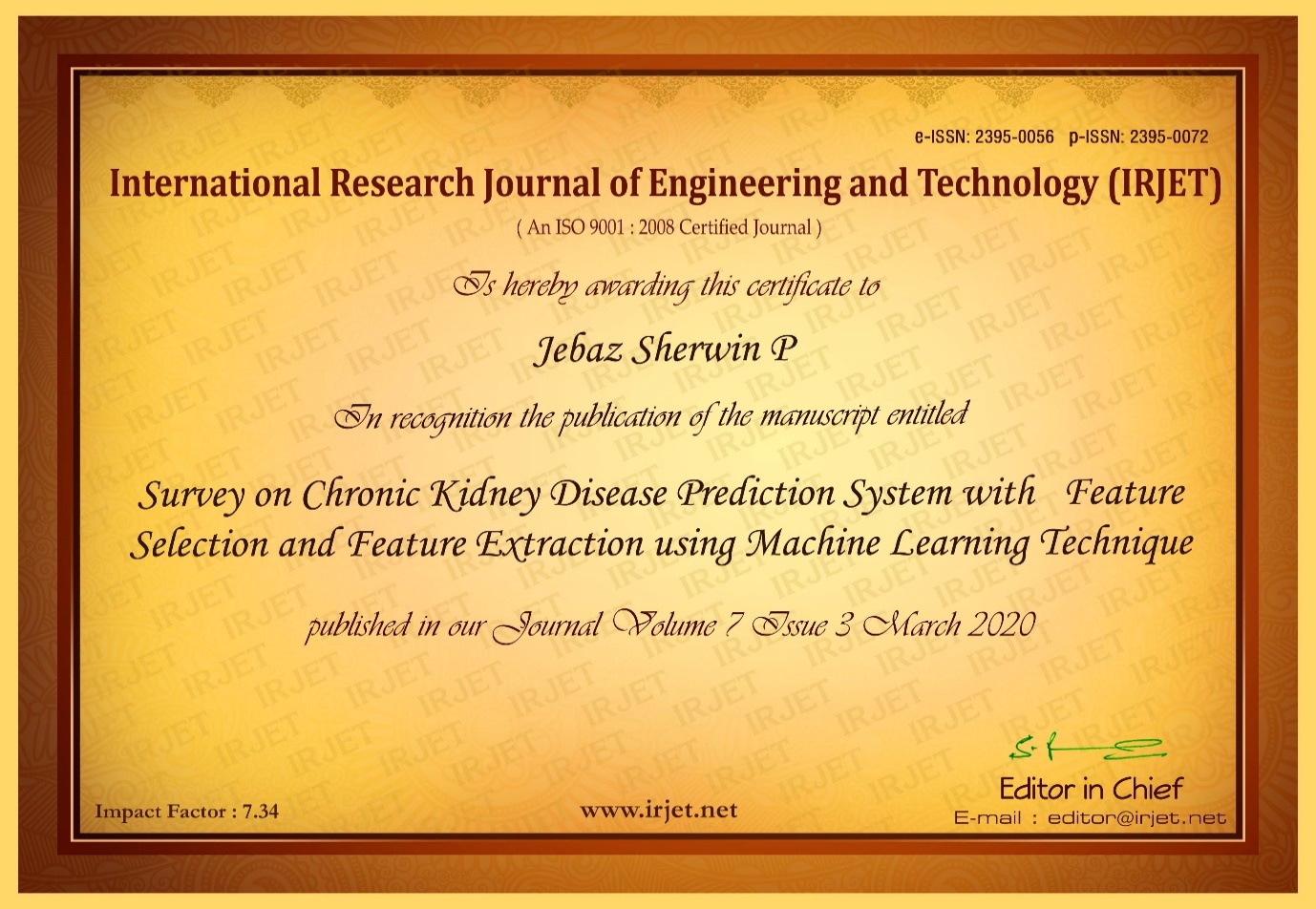


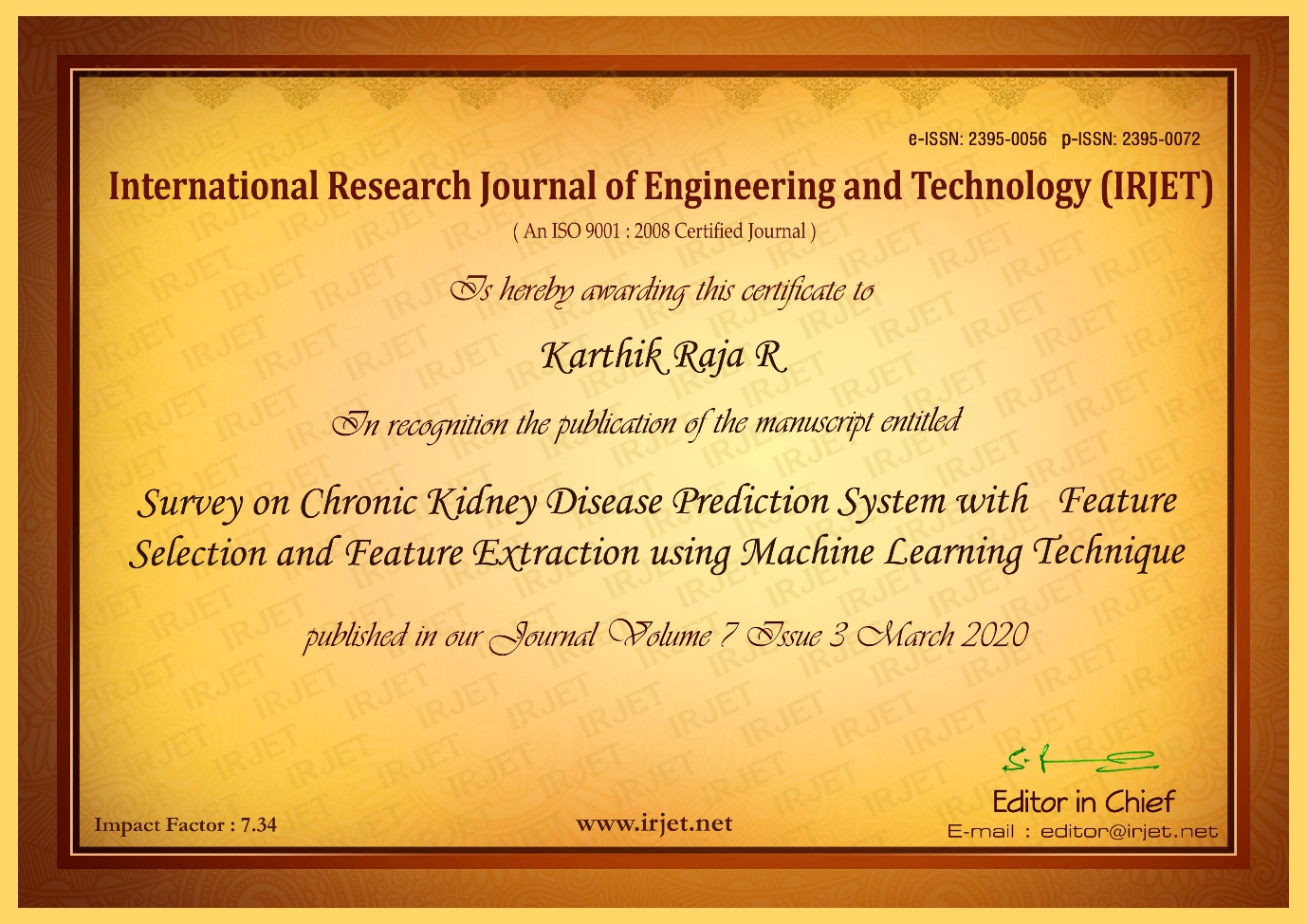




**JOURNAL PAPER PUBLICATION CERTIFICATES**









**REFERENCES**

1. Agarwal, S.K.: Chronic kidney disease in India - magnitude and issues involved (2009)
2. Vinitha, S., Sweetlin, S., Vinusha, H., Sajini, S.: Disease prediction using machine learning over big data. Comput. Sci. Eng. Int. J. (CSEIJ) 8, 1–8 (2018).
3. Subas, A., Alickovic, E., Kevric, J.: Diagnosis of chronic kidney disease by using random forest, pp. 589–594 (2017).
4. Imran, A.A., Amin, M.N., Johora, F.T.: Classiﬁcation of chronic kidney disease using logistic regression, feedforward neural network and wide and deep learning. In: 2018 International Conference on Innovation in Engineering and Technology (ICIET), pp. 1–6 (2018).
5. Wibawa, H.A., Malik, I., Bahtiar, N.: Evaluation of kernel-based extreme learning machine performance for prediction of chronic kidney disease. In: 2018 2nd International Conference on Informatics and Computational Sciences (ICICoS), pp. 1–4 (2018).
6. Radha, N., Ramya, S.: Performance analysis of machine learning algorithms for predicting chronic kidney disease. Int. J. Comput. Sci. Eng. Open Access 3, 72–76 (2015).
7. Ramya, S., Radha, N.: Diagnosis of chronic kidney disease using machine learning algorithms. Int. J. Innov. Res. Comput. Commun. Eng. 4, 812–820 (2016).
8. Charleonnan, A., Fufaung, T., Niyomwong, T., Chokchueypattanakit, W., Suwannawach, S., Ninchawee, N.: Predictive analytics for chronic kidney disease using machine learning.
9. Gunarathne, W.H.S.D., Perera, K.D.M., Kahandawaarachchi, K.A.D.C.P.: Performance evaluation on machine learning classiﬁcation techniques for disease classiﬁcation and forecasting through data analytics for chronic kidney disease (CKD). In: 2017 IEEE 17th International Conference on Bioinformatics and Bioengineering (BIBE), pp. 291–296 (2017).

10.Basar, M.D., Akan, A.: Detection of chronic kidney disease by using ensemble classiﬁers. In: 2017 10th International Conference on Electrical and Electronics Engineering (ELECO), pp. 544–547 (2017).

11.Tekale, S., Shingavi, P., Wandhekar, S., Chatorikar, A.: Prediction of chronic kidney disease using machine learning algorithm. Int. J. Adv. Res. Comput. Commun. Eng. 7, 92–96 (2018).

12.Ogunleye, A., Wang, Q.: Enhanced XGBoost-based automatic diagnosis system for chronic kidney disease. In: 2018 IEEE 14th International Conference on Control and Automation (ICCA), pp. 805–810 (2018).

13.Sisodia, D.S., Verma, A.: Prediction performance of individual and ensemble learners for chronic kidney disease. In: 2017 International Conference on Inventive Computing and Informatics (ICICI), pp. 1027–1031 (2017).

14.Jadhav, S.D., Channe, H.P.: Comparative study of K-NN, Naive Bayes and decision tree classiﬁcation techniques. Int. J. Sci. Res. (IJSR) 5, 1842–1845.