

A Major Project Report
on
“A MACHINE LEARNING METHODOLOGY FOR DIAGNOSING
CHRONIC KIDNEY DISEASE”

Submitted to the
JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY
HYDERABAD

In partial fulfilment of the requirement for the award of the degree of
BACHELOR OF TECHNOLOGY

IN
Computer Science and Engineering
BY

P. AISHWARYA SRI(17WJ1A05N0)

Under the Esteemed Guidance of
Mr. V. Deva Sekhar
Associate Professor, HOD Of CSE, GNITC



GURU NANAK INSTITUTIONS TECHNICAL CAMPUS (AUTONOMOUS)

School of Engineering and Technology

Ibrahimpattanam, R.R District 501506

2020-2021



Department of Computer Science and Engineering

CERTIFICATE

This is to certify that this project report entitled “**A MACHINE LEARNING METHODOLOGY FOR DIAGNOSING CHRONIC KIDNEY DISEASE**” by **PATI AISHWARYA SRI (17WJ1A05N0)** submitted in partial fulfilment the requirements for the degree of **Bachelor of Technology** in **Computer Science and Engineering** of the **Jawaharlal Nehru Technological University Hyderabad** during the academic year 2020-2021, is a bonafide record of work carried out under our guidance and supervision.

INTERNAL GUIDE
Mr.V. Deva Sekhar

PROJECT CO-ORDINATOR
Mrs.V. Swathi

HOD CSE
Mr.V. Deva Sekhar

EXTERNAL EXAMINER

**RAM Innovative Infotech**

M : +91 9581 012 012
E : raminnovativeinfotech@gmail.com

Flat No.#309, Amrutha Ville,
Opp: Yashoda Hospital, Somajiguda,
Hyderabad-82, Telangana, India
www.raminnovativeinfotech.webs.com

PROJECT COMPLETION CERTIFICATE

This is to certify that the following student of final year B.Tech, Department of Computer science and engineering - Guru Nanak Institutions Technical Campus (GNITC) has completed her training and project at GNITC successfully.

STUDENT NAME:

Pati Aishwarya Sri

ROLL NO:

17WJ1A05N0

The training was conducted on Machine learning Technology for the completion of the project titled A Machine Learning Methodology For Diagnosing Chronic Kidney Disease in RAM Innovative Infotech. The project has been completed in all aspects.


Signature

ACKNOWLEDGEMENT

I wish to express our sincere thanks to **Dr. Rishi Sayal**, Associate Director, GNITC for providing us the conducive environment for carrying through our academic schedule and project with ease.

I have been truly blessed to have a wonderful adviser **Mr.V.Deva Sekhar**, Associate Professor & Head, Dept of CSE, GNITC for guiding us to explore the ramification of our work and we express my sincere gratitude towards him for leading me through the completion of project.

I thank Project Coordinator **Mrs. V. Swathi**, Assistant Professor, CSE, GNITC for providing seamless support and right suggestions that are given in the development of the project.

I especially thank our internal guide **Mr.V.Deva Sekhar**, Associate Professor & Head, Dept of CSE , GNITC for her constant guidance in every stage of the project. We would also like to thank all our lecturers for helping us in every possible way whenever the need arose.

On a more personal note, we thank our beloved parents and friends for their moral support during the course of our project.

PATI AISHWARYA SRI (17WJ1A05N0)

TITLE:

**A Machine Learning Methodology for Diagnosing
Chronic Kidney Disease**

TABLE OF CONTENTS

TITLE	PAGE NO.
ABSTRACT.....	i
LIST OF FIGURES.....	ii
LIST OF SYMBOLS.....	iii-iv
LIST OF ABBREVIATION.....	v
CHAPTER 1 : INTRODUCTION.....	1-38
1.1 General	
1.2 Scope of The Project	
1.3 Objective	
1.4 Problem Statement	
1.5 Existing System	
1.5.1 Existing System Disadvantages	
1.5.2 Literature Survey	
1.6 Proposed System	
1.6.1 Proposed System Advantages	
CHAPTER 2 : PROJECT DESCRIPTION.....	39-44
2.1 General	
2.2 Methodologies	
2.2.1 Modules Name	
2.2.2 Modules Explanation	
2.2.3 Given Input and Expected Output	
CHAPTER 3 : REQUIREMENTS.....	45-47
3.1 General	
3.2 Hardware Requirements	
3.3 Software Requirements	
3.4 Functional Specification	
3.5 Non-Functional Specification	

CHAPTER 4 : SYSTEM DESIGN.....48-54

4.1 General

4.1.1 Use Case Diagram

4.1.2 Class Diagram

4.1.3 Sequenced Diagram

4.1.4 Activity Diagram

4.1.5 Data Flow Diagram

4.2 System Architecture

CHAPTER 5 : SOFTWARE SPECIFICATION.....55-67

5.1 General

CHAPTER 6 : IMPLEMENTATION.....68-69

6.1 General

6.2 Implementation

CHAPTER 7 : SNAPSHOTS.....70-76

7.1 General

7.2 Various Snapshots

CHAPTER 8 : SOFTWARE TESTING.....77-80

8.1 General

8.2 Developing Methodologies

8.3 Types of Testing

8.3.1 Unit Testing

8.3.2 Functional Test

8.3.3 System Test

8.3.4 Performance Test

8.3.5 Integration Testing

8.3.6 Acceptance Testing

8.3.7 Build the Test Plan

8.4 Unit Testing

8.5 Integration Test

8.6 Acceptance Testing

CHAPTER 9 : APPLICATIONS AND FUTURE ENHANCEMENT.....82

9.1 General

9.2 Future Enhancements

CHAPTER 10 : CONCLUSION AND REFERENCES.....83-85

10.1 Conclusion

10.2 References

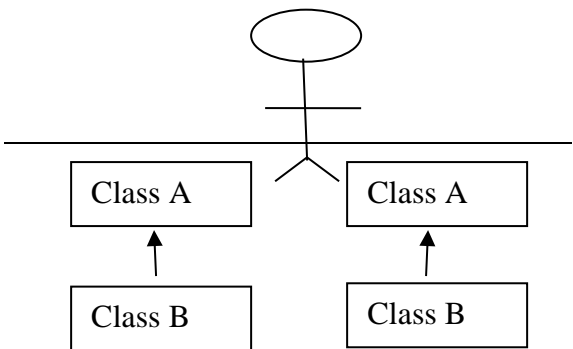

ABSTRACT



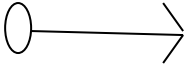
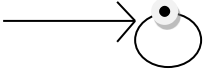

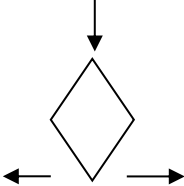
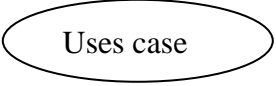
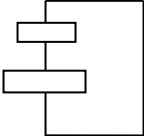
Chronic kidney disease (CKD) is a global health problem with high morbidity and mortality rate, and it induces other diseases. Since there are no obvious symptoms during the early stages of CKD, patients often fail to notice the disease. Early detection of CKD enables patients to receive timely treatment to ameliorate the progression of this disease. Machine learning models can effectively aid clinicians achieve this goal due to their fast and accurate recognition performance. In this study, we propose a machine learning methodology for diagnosing CKD. The CKD data set was obtained from the University of California Irvine (UCI) machine learning repository, which has a large number of missing values. KNN imputation was used to fill in the missing values, which selects several complete samples with the most similar measurements to process the missing data for each incomplete sample. Missing values are usually seen in real-life medical situations because patients may miss some measurements for various reasons. After effectively filling out the incomplete data set, six machine learning algorithms (logistic regression, random forest, support vector machine, k-nearest neighbour, naive Bayes classifier and feed forward neural network) were used to establish models. Among these machine learning models, random forest achieved the best performance with 99.75% diagnosis accuracy. By analyzing the misjudgements generated by the established models, we proposed an integrated model that combines logistic regression and random forest by using perceptron, which could achieve an average accuracy of 99.83% after ten times of simulation. Hence, we speculated that this methodology could be applicable to more complicated clinical data for disease diagnosis.

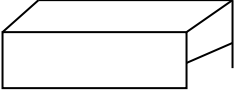
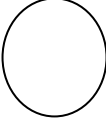


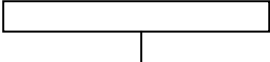
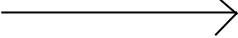
LIST OF FIGURES

FIGURE NO	NAME OF THE FIGURE	PAGE NO.
4.1.1	Use case Diagram	49
4.1.2	Class Diagram	50
4.1.6	Sequence Diagram	51
4.1.9	Activity Diagram	52
4.1.10	Data flow Diagram	53
4.1.12	System Architecture	54

LIST OF NOTATIONS

S.NO	NAME	NOTATION	DESCRIPTION
1.	Class	<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="border: 1px solid black; padding: 5px; text-align: center;"> <i>+ public</i> <i>- private</i> <i># protected</i> </div> <div style="border: 1px solid black; padding: 5px; text-align: center;"> <i>Class Name</i> </div> </div> <div style="display: flex; justify-content: space-around; align-items: center; margin-top: 10px;"> <div style="border: 1px solid black; padding: 5px; text-align: center;"> <i>- attribute</i> <i>- attribute</i> </div> <div style="border: 1px solid black; padding: 5px; text-align: center;"> <i>+ operation</i> <i>+ operation</i> <i>+ operation</i> </div> </div>	Represents a collection of similar entities grouped together.
2.	Association	<div style="display: flex; justify-content: center; align-items: center; margin-bottom: 20px;"> <div style="border: 1px solid black; padding: 5px; margin: 0 10px;">Class A</div> <div style="border-bottom: 1px solid black; width: 50px; margin: 0 5px;"></div> <div style="border: 1px solid black; padding: 5px; margin: 0 10px;">Class B</div> </div> <div style="display: flex; justify-content: center; align-items: center;"> <div style="border: 1px solid black; padding: 5px; margin: 0 10px;">Class A</div> <div style="border-bottom: 1px solid black; width: 50px; margin: 0 5px;"></div> <div style="border: 1px solid black; padding: 5px; margin: 0 10px;">Class B</div> </div>	Associations represents static relationships between classes. Roles represents the way the two classes see each other.
3.	Aor		It aggregates several classes into a single classes
5.	Aggregation		Interaction between the system and external environment
5.	Relation (uses)	Uses	Used for additional process communication.
6.	Relation (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 of the process.
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 action.
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.

LIST OF ABBREVIATION

S.NO	ABBREVIATION	EXPANSION
1.	DB	Data Base
2.	ML	Machine Learning
3.	SL	Supervised Learning
4.	AI	Artificial intelligence
5.	ANNs	Artificial Neural Networks
6.	RF	Random Forest

CHAPTER 1

INTRODUCTION

1.1 GENERAL

Chronic kidney disease (CKD) is a global public health problem affecting approximately 10% of the world's population [1], [2]. The percentage of prevalence of CKD in China is 10.8% [3], and the range of prevalence is 10%-15% in the United States [4]. According to another study, this percentage has reached 14.7% in the Mexican adult general population [5]. This disease is characterised by a slow deterioration in renal function, which eventually causes a complete loss of renal function. CKD does not show obvious symptoms in its early stages. Therefore, the disease may not be detected until the kidney loses about 25% of its function [6]. In addition, CKD has high morbidity and mortality, with a global impact on the human body [7]. It can induce the occurrence of cardiovascular disease [8], [9]. CKD is a progressive and irreversible pathologic syndrome [10]. Hence, the prediction and diagnosis of CKD in its early stages is quite essential, it may be able to enable patients to receive timely treatment to ameliorate the progression of the disease

What is Machine Learning?

Machine Learning is a system of computer algorithms that can learn from example through self-improvement without being explicitly coded by a programmer. Machine learning is a part of artificial Intelligence which combines data with statistical tools to predict an output which can be used to make actionable insights.

The breakthrough comes with the idea that a machine can singularly learn from the data (i.e., example) to produce accurate results. Machine learning is closely related to data mining and Bayesian predictive modeling. The machine receives data as input and uses an algorithm to formulate answers.

A typical machine learning tasks are to provide a recommendation. For those who have a Netflix account, all recommendations of movies or series are based on the user's historical data. Tech companies are using unsupervised learning to improve the user experience with personalizing recommendation.

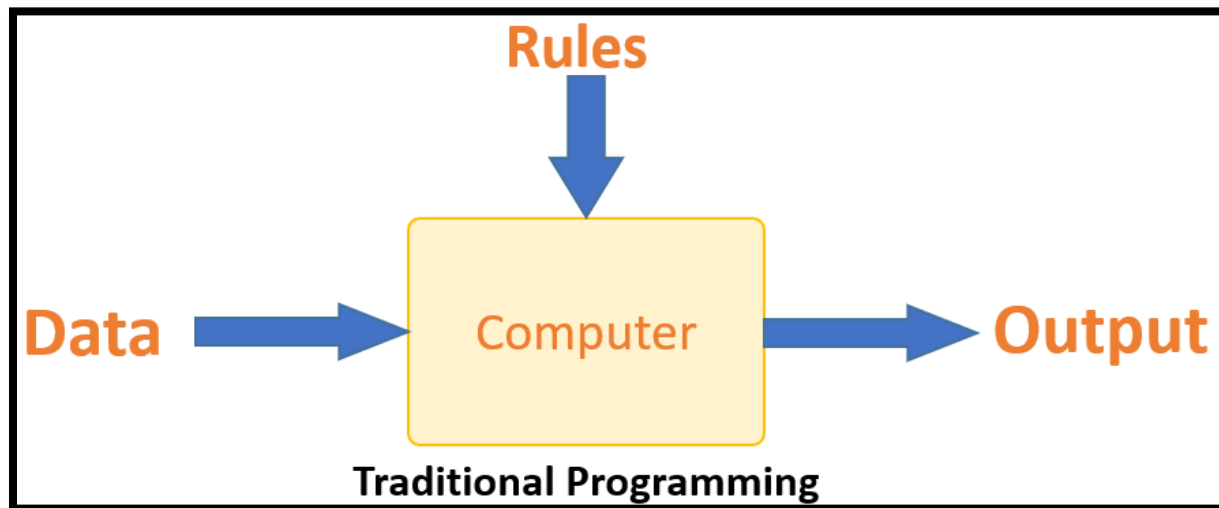
Machine learning is also used for a variety of tasks like fraud detection, predictive maintenance, portfolio optimization, automatize task and so on.

Machine Learning vs. Traditional Programming

Traditional programming differs significantly from machine learning. In traditional programming, a programmer code all the rules in consultation with an expert in the industry for which software is being developed. Each rule is based on a logical foundation; the machine will execute an output following the logical statement. When the system grows complex, more rules need to be written. It can quickly become unsustainable to maintain.

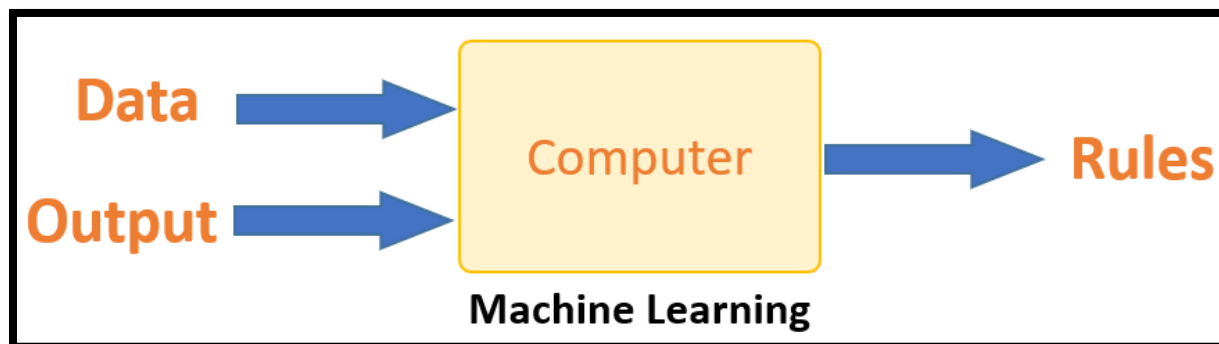
Traditional programming differs significantly from machine learning. In traditional programming, a programmer code all the rules in consultation with an expert in the industry for which software is being developed. Each rule is based on a logical foundation; the machine will execute an output

following the logical statement. When the system grows complex, more rules need to be written. It can quickly become unsustainable to maintain.



Traditional Programming

Machine learning is supposed to overcome this issue. The machine learns how the input and output data are correlated and it writes a rule. The programmers do not need to write new rules each time there is new data. The algorithms adapt in response to new data and experiences to improve efficacy over time.



Machine Learning

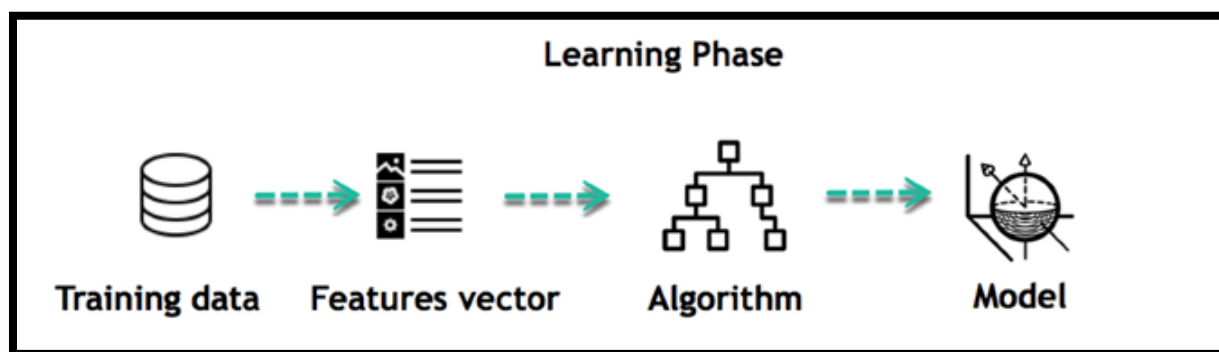
How does Machine Learning Work?

Machine learning is the brain where all the learning takes place. The way the machine learns is similar to the human being. Humans learn from experience. The more we know, the more easily we can predict. By analogy, when we face an unknown situation, the likelihood of success is lower than the known situation. Machines are trained the same. To make an accurate prediction, the machine sees an example. When we give the machine a similar example, it can figure out the

outcome. However, like a human, if its feed a previously unseen example, the machine has difficulties to predict.

The core objective of machine learning is the **learning** and **inference**. First of all, the machine learns through the discovery of patterns. This discovery is made thanks to the **data**. One crucial part of the data scientist is to choose carefully which data to provide to the machine. The list of attributes used to solve a problem is called a **feature vector**. You can think of a feature vector as a subset of data that is used to tackle a problem.

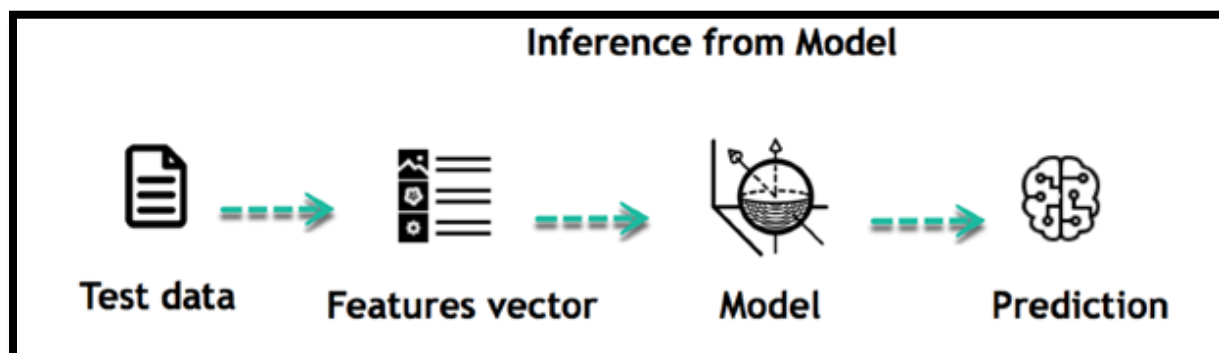
The machine uses some fancy algorithms to simplify the reality and transform this discovery into a **model**. Therefore, the learning stage is used to describe the data and summarize it into a model.



For instance, the machine is trying to understand the relationship between the wage of an individual and the likelihood to go to a fancy restaurant. It turns out the machine finds a positive relationship between wage and going to a high-end restaurant: This is the model

Inferring

When the model is built, it is possible to test how powerful it is on never-seen-before data. The new data are transformed into a features vector, go through the model and give a prediction. This is all the beautiful part of machine learning. There is no need to update the rules or train again the model. You can use the model previously trained to make inference on new data.

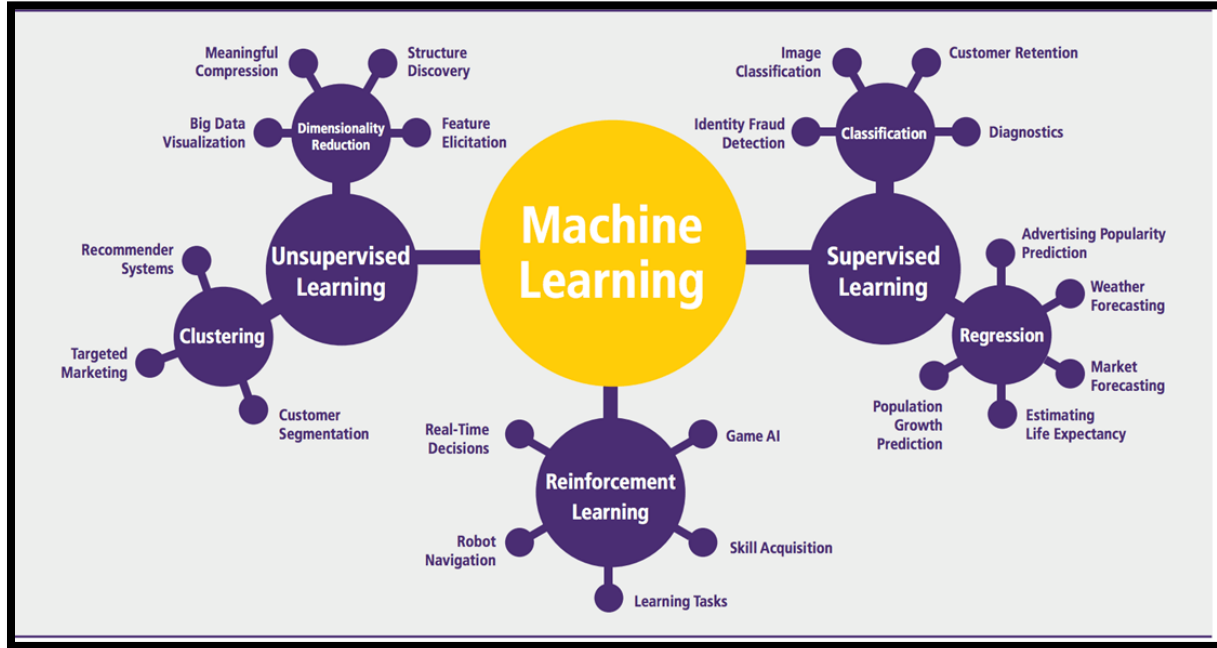


The life of Machine Learning programs is straightforward and can be summarized in the following points:

1. Define a question
2. Collect data
3. Visualize data
4. Train algorithm
5. Test the Algorithm
6. Collect feedback
7. Refine the algorithm
8. Loop 4-7 until the results are satisfying
9. Use the model to make a prediction

Once the algorithm gets good at drawing the right conclusions, it applies that knowledge to new sets of data.

Machine Learning Algorithms and Where they are Used?



Machine learning Algorithms

Machine learning can be grouped into two broad learning tasks: Supervised and Unsupervised. There are many other algorithms

Supervised learning

An algorithm uses training data and feedback from humans to learn the relationship of given inputs to a given output. For instance, a practitioner can use marketing expense and weather forecast as input data to predict the sales of cans.

You can use supervised learning when the output data is known. The algorithm will predict new data.

There are two categories of supervised learning:

- Classification task
- Regression task

Classification

Imagine you want to predict the gender of a customer for a commercial. You will start gathering data on the height, weight, job, salary, purchasing basket, etc. from your customer database. You know the gender of each of your customer, it can only be male or female. The objective of the classifier will be to assign a probability of being a male or a female (i.e., the label) based on the information (i.e., features you have collected). When the model learned how to recognize male or female, you can use new data to make a prediction. For instance, you just got new information from an unknown customer, and you want to know if it is a male or female. If the classifier predicts male = 70%, it means the algorithm is sure at 70% that this customer is a male, and 30% it is a female.

The label can be of two or more classes. The above Machine learning example has only two classes, but if a classifier needs to predict object, it has dozens of classes (e.g., glass, table, shoes, etc. each object represents a class)

Regression

When the output is a continuous value, the task is a regression. For instance, a financial analyst may need to forecast the value of a stock based on a range of feature like equity, previous stock performances, macroeconomics index. The system will be trained to estimate the price of the stocks with the lowest possible error.

Algorithm Name	Description	Type
Linear regression	Finds a way to correlate each feature to the output to help predict future values.	Regression
Logistic regression	Extension of linear regression that's used for classification tasks. The output variable is binary (e.g., only black or white) rather than continuous (e.g., an infinite list of potential colors)	Classification
Decision tree	Highly interpretable classification or regression model that splits data feature values into branches at decision nodes (e.g., if a feature is color, each possible color becomes a new branch) until a final decision output is made	Regression Classification
Naive Bayes	The Bayesian method is a classification method that makes use of the Bayesian theorem. The theorem updates the prior knowledge of an event with the independent probability of each feature that can affect the event.	Regression Classification
Support vector machine	Support Vector Machine, or SVM, is typically used for both classification and regression tasks. SVM algorithm finds a hyperplane that optimally divides the classes. It is best used with a non-linear solver.	Regression (not very common) Classification
Random forest	The algorithm is built upon a decision tree to improve the accuracy drastically. Random forest generates many times simple decision tree and uses the 'majority vote' method to decide on which label to return. For the classification task, the final prediction will be the one with the most vote; while for the regression task, the average prediction of all the trees is the final prediction.	Regression Classification
AdaBoost	Classification or regression technique that uses a multitude of models to come up with a decision but weighs them based on their accuracy in predicting the outcome	Regression Classification
Gradient-boosting trees	Gradient-boosting trees is a state-of-the-art classification/regression technique. It is focusing on the error committed by the previous tree and tries to correct it.	Regression Classification

Unsupervised learning

In unsupervised learning, an algorithm explores input data without being given an explicit output variable (e.g., explores customer demographic data to identify patterns)

You can use it when you do not know how to classify the data, and you want the algorithm to find patterns and classify the data for you

Algorithm	Description	Type
K-means clustering	Puts data into some groups (k) that each contains data with similar characteristics (as determined by the model, not in advance by humans)	Clustering
Gaussian mixture model	A generalization of k-means clustering that provides more flexibility in the size and shape of groups (clusters)	Clustering
Hierarchical clustering	Splits clusters along a hierarchical tree to form a classification system. Can be used for Cluster loyalty-card customer	Clustering
Recommender system	Help to define the relevant data for making a recommendation.	Clustering
PCA/T-SNE	Mostly used to decrease the dimensionality of the data. The algorithms reduce the number of features to 3 or 4 vectors with the highest variances. Reduction	Dimension

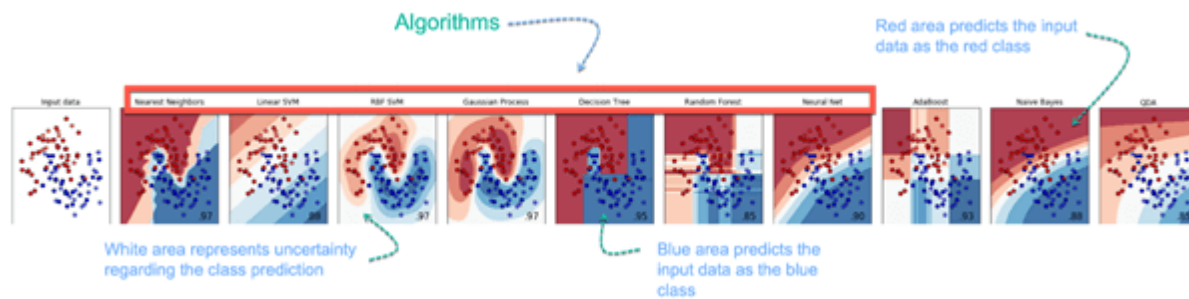
How to Choose Machine Learning Algorithm

Machine Learning (ML) algorithm:

There are plenty of machine learning algorithms. The choice of the algorithm is based on the objective.

In the Machine learning example below, the task is to predict the type of flower among the three varieties. The predictions are based on the length and the width of the petal. The picture depicts the results of ten different algorithms. The picture on the top left is the dataset. The data is classified into three categories: red, light blue and dark blue. There are some groupings. For instance, from the second image, everything in the upper left belongs to the red category, in the middle part, there

is a mixture of uncertainty and light blue while the bottom corresponds to the dark category. The other images show different algorithms and how they try to classify the data.



Challenges and Limitations of Machine Learning

The primary challenge of machine learning is the lack of data or the diversity in the dataset. A machine cannot learn if there is no data available. Besides, a dataset with a lack of diversity gives the machine a hard time. A machine needs to have heterogeneity to learn meaningful insight. It is rare that an algorithm can extract information when there are no or few variations. It is recommended to have at least 20 observations per group to help the machine learn. This constraint leads to poor evaluation and prediction.

Application of Machine Learning

Augmentation:

- Machine learning, which assists humans with their day-to-day tasks, personally or commercially without having complete control of the output. Such machine learning is used in different ways such as Virtual Assistant, Data analysis, software solutions. The primary user is to reduce errors due to human bias.

Automation:

- Machine learning, which works entirely autonomously in any field without the need for any human intervention. For example, robots performing the essential process steps in manufacturing plants.

Finance Industry

- Machine learning is growing in popularity in the finance industry. Banks are mainly using ML to find patterns inside the data but also to prevent fraud.

Government organization

- The government makes use of ML to manage public safety and utilities. Take the example of China with the massive face recognition. The government uses Artificial intelligence to prevent jaywalker.

Healthcare industry

- Healthcare was one of the first industry to use machine learning with image detection.

Marketing

- Broad use of AI is done in marketing thanks to abundant access to data. Before the age of mass data, researchers develop advanced mathematical tools like Bayesian analysis to estimate the value of a customer. With the boom of data, marketing department relies on AI to optimize the customer relationship and marketing campaign.

Example of application of Machine Learning in Supply Chain

Machine learning gives terrific results for visual pattern recognition, opening up many potential applications in physical inspection and maintenance across the entire supply chain network.

Unsupervised learning can quickly search for comparable patterns in the diverse dataset. In turn, the machine can perform quality inspection throughout the logistics hub, shipment with damage and wear.

For instance, IBM's Watson platform can determine shipping container damage. Watson combines visual and systems-based data to track, report and make recommendations in real-time.

In past year stock manager relies extensively on the primary method to evaluate and forecast the inventory. When combining big data and machine learning, better forecasting techniques have been implemented (an improvement of 20 to 30 % over traditional forecasting tools). In term of sales, it means an increase of 2 to 3 % due to the potential reduction in inventory costs.

Example of Machine Learning Google Car

For example, everybody knows the Google car. The car is full of lasers on the roof which are telling it where it is regarding the surrounding area. It has radar in the front, which is informing the car of the speed and motion of all the cars around it. It uses all of that data to figure out not only how to drive the car but also to figure out and predict what potential drivers around the car are going to do. What's impressive is that the car is processing almost a gigabyte a second of data.

Why is Machine Learning Important?

Machine learning is the best tool so far to analyze, understand and identify a pattern in the data. One of the main ideas behind machine learning is that the computer can be trained to automate tasks that would be exhaustive or impossible for a human being. The clear breach from the traditional analysis is that machine learning can take decisions with minimal human intervention. Take the following example for this ML tutorial; a retail agent can estimate the price of a house based on his own experience and his knowledge of the market.

A machine can be trained to translate the knowledge of an expert into features. The features are all the characteristics of a house, neighborhood, economic environment, etc. that make the price difference. For the expert, it took him probably some years to master the art of estimate the price of a house. His expertise is getting better and better after each sale.

For the machine, it takes millions of data, (i.e., example) to master this art. At the very beginning of its learning, the machine makes a mistake, somehow like the junior salesman. Once the machine sees all the example, it got enough knowledge to make its estimation. At the same time, with incredible accuracy. The machine is also able to adjust its mistake accordingly.

Most of the big company have understood the value of machine learning and holding data. McKinsey have estimated that the value of analytics ranges from \$9.5 trillion to \$15.4 trillion while \$5 to 7 trillion can be attributed to the most advanced AI techniques.

Machine learning (ML) is the study of computer algorithms that improve automatically through experience. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

A subset of machine learning is closely related to computational statistics, which focuses on making predictions using computers; but not all machine learning is statistical learning. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis

through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics.

Overview

Machine learning involves computers discovering how they can perform tasks without being explicitly programmed to do so. It involves computers learning from data provided so that they carry out certain tasks. For simple tasks assigned to computers, it is possible to program algorithms telling the machine how to execute all steps required to solve the problem at hand; on the computer's part, no learning is needed. For more advanced tasks, it can be challenging for a human to manually create the needed algorithms. In practice, it can turn out to be more effective to help the machine develop its own algorithm, rather than having human programmers specify every needed step.

The discipline of machine learning employs various approaches to teach computers to accomplish tasks where no fully satisfactory algorithm is available. In cases where vast numbers of potential answers exist, one approach is to label some of the correct answers as valid. This can then be used as training data for the computer to improve the algorithm(s) it uses to determine correct answers. For example, to train a system for the task of digital character recognition, the MNIST dataset of handwritten digits has often been used.

Machine learning approaches

Machine learning approaches are traditionally divided into three broad categories, depending on the nature of the "signal" or "feedback" available to the learning system:

Supervised learning: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.

Unsupervised learning: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).

Reinforcement learning: A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an

opponent). As it navigates its problem space, the program is provided feedback that's analogous to rewards, which it tries to maximize.

Other approaches have been developed which don't fit neatly into this three-fold categorisation, and sometimes more than one is used by the same machine learning system. For example topic modeling, dimensionality reduction or meta learning.

As of 2020, deep learning has become the dominant approach for much ongoing work in the field of machine learning.

History and relationships to other fields

The term machine learning was coined in 1959 by Arthur Samuel, an American IBMer and pioneer in the field of computer gaming and artificial intelligence. A representative book of the machine learning research during the 1960s was the Nilsson's book on Learning Machines, dealing mostly with machine learning for pattern classification. Interest related to pattern recognition continued into the 1970s, as described by Duda and Hart in 1973. In 1981 a report was given on using teaching strategies so that a neural network learns to recognize 40 characters (26 letters, 10 digits, and 4 special symbols) from a computer terminal.

Tom M. Mitchell provided a widely quoted, more formal definition of the algorithms studied in the machine learning field: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T , as measured by P , improves with experience E ." This definition of the tasks in which machine learning is concerned offers a fundamentally operational definition rather than defining the field in cognitive terms. This follows Alan Turing's proposal in his paper "Computing Machinery and Intelligence", in which the question "Can machines think?" is replaced with the question "Can machines do what we (as thinking entities) can do?".

Modern day machine learning has two objectives, one is to classify data based on models which have been developed, the other purpose is to make predictions for future outcomes based on these models. A hypothetical algorithm specific to classifying data may use computer vision of moles coupled with supervised learning in order to train it to classify the cancerous moles. Where as, a machine learning algorithm for stock trading may inform the trader of future potential predictions.

Artificial intelligence

Machine Learning as subfield of AI

Part of Machine Learning as subfield of AI or part of AI as subfield of Machine Learning

As a scientific endeavor, machine learning grew out of the quest for artificial intelligence. In the early days of AI as an academic discipline, some researchers were interested in having machines learn from data. They attempted to approach the problem with various symbolic methods, as well as what was then termed "neural networks"; these were mostly perceptrons and other models that were later found to be reinventions of the generalized linear models of statistics. Probabilistic reasoning was also employed, especially in automated medical diagnosis.

However, an increasing emphasis on the logical, knowledge-based approach caused a rift between AI and machine learning. Probabilistic systems were plagued by theoretical and practical problems of data acquisition and representation. By 1980, expert systems had come to dominate AI, and statistics was out of favor. Work on symbolic/knowledge-based learning did continue within AI, leading to inductive logic programming, but the more statistical line of research was now outside the field of AI proper, in pattern recognition and information retrieval. Neural networks research had been abandoned by AI and computer science around the same time. This line, too, was continued outside the AI/CS field, as "connectionism", by researchers from other disciplines including Hopfield, Rumelhart and Hinton. Their main success came in the mid-1980s with the reinvention of backpropagation.

Machine learning (ML), reorganized as a separate field, started to flourish in the 1990s. The field changed its goal from achieving artificial intelligence to tackling solvable problems of a practical nature. It shifted focus away from the symbolic approaches it had inherited from AI, and toward methods and models borrowed from statistics and probability theory.

As of 2020, many sources continue to assert that machine learning remains a subfield of AI. The main disagreement is whether all of ML is part of AI, as this would mean that anyone using ML could claim they are using AI. Others have the view that not all of ML is part of AI where only an 'intelligent' subset of ML is part of AI.

The question to what is the difference between ML and AI is answered by Judea Pearl in The Book of Why. Accordingly ML learns and predicts based on passive observations, whereas AI implies an agent interacting with the environment to learn and take actions that maximize its chance of successfully achieving its goals.

Data mining

Machine learning and data mining often employ the same methods and overlap significantly, but while machine learning focuses on prediction, based on known properties learned from the training data, data mining focuses on the discovery of (previously) unknown properties in the data (this is the analysis step of knowledge discovery in databases). Data mining uses many machine learning methods, but with different goals; on the other hand, machine learning also employs data mining methods as "unsupervised learning" or as a preprocessing step to improve learner accuracy. Much of the confusion between these two research communities (which do often have separate conferences and separate journals, ECML PKDD being a major exception) comes from the basic assumptions they work with: in machine learning, performance is usually evaluated with respect to the ability to reproduce known knowledge, while in knowledge discovery and data mining (KDD) the key task is the discovery of previously unknown knowledge. Evaluated with respect to known knowledge, an uninformed (unsupervised) method will easily be outperformed by other supervised methods, while in a typical KDD task, supervised methods cannot be used due to the unavailability of training data.

Optimization

Machine learning also has intimate ties to optimization: many learning problems are formulated as minimization of some loss function on a training set of examples. Loss functions express the discrepancy between the predictions of the model being trained and the actual problem instances (for example, in classification, one wants to assign a label to instances, and models are trained to correctly predict the pre-assigned labels of a set of examples).

Generalization

The difference between optimization and machine learning arises from the goal of generalization: while optimization algorithms can minimize the loss on a training set, machine learning is concerned with minimizing the loss on unseen samples. Characterizing the generalization of various learning algorithms is an active topic of current research, especially for deep learning algorithms.

Statistics

Machine learning and statistics are closely related fields in terms of methods, but distinct in their principal goal: statistics draws population inferences from a sample, while machine learning finds generalizable predictive patterns. According to Michael I. Jordan, the ideas of machine learning, from methodological principles to theoretical tools, have had a long pre-history in statistics. He also suggested the term data science as a placeholder to call the overall field.

Leo Breiman distinguished two statistical modeling paradigms: data model and algorithmic model, wherein "algorithmic model" means more or less the machine learning algorithms like Random forest.

Some statisticians have adopted methods from machine learning, leading to a combined field that they call statistical learning.

Theory

A core objective of a learner is to generalize from its experience. Generalization in this context is the ability of a learning machine to perform accurately on new, unseen examples/tasks after having experienced a learning data set. The training examples come from some generally unknown probability distribution (considered representative of the space of occurrences) and the learner has to build a general model about this space that enables it to produce sufficiently accurate predictions in new cases.

The computational analysis of machine learning algorithms and their performance is a branch of theoretical computer science known as computational learning theory. Because training sets are finite and the future is uncertain, learning theory usually does not yield guarantees of the performance of algorithms. Instead, probabilistic bounds on the performance are quite common. The bias–variance decomposition is one way to quantify generalization error.

For the best performance in the context of generalization, the complexity of the hypothesis should match the complexity of the function underlying the data. If the hypothesis is less complex than the function, then the model has under fitted the data. If the complexity of the model is increased in response, then the training error decreases. But if the hypothesis is too complex, then the model is subject to overfitting and generalization will be poorer.

In addition to performance bounds, learning theorists study the time complexity and feasibility of learning. In computational learning theory, a computation is considered feasible if it can be done in polynomial time. There are two kinds of time complexity results. Positive results show that a certain class of functions can be learned in polynomial time. Negative results show that certain classes cannot be learned in polynomial time.

Approaches

Types of learning algorithms

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve.

Supervised learning

A support vector machine is a supervised learning model that divides the data into regions separated by a linear boundary. Here, the linear boundary divides the black circles from the white. Supervised learning algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs. The data is known as training data, and consists of a set of training examples. Each training example has one or more inputs and the desired output, also known as a

supervisory signal. In the mathematical model, each training example is represented by an array or vector, sometimes called a feature vector, and the training data is represented by a matrix. Through iterative optimization of an objective function, supervised learning algorithms learn a function that can be used to predict the output associated with new inputs. An optimal function will allow the algorithm to correctly determine the output for inputs that were not a part of the training data. An algorithm that improves the accuracy of its outputs or predictions over time is said to have learned to perform that task.

Types of supervised learning algorithms include active learning, classification and regression. Classification algorithms are used when the outputs are restricted to a limited set of values, and regression algorithms are used when the outputs may have any numerical value within a range. As an example, for a classification algorithm that filters emails, the input would be an incoming email, and the output would be the name of the folder in which to file the email.

Similarity learning is an area of supervised machine learning closely related to regression and classification, but the goal is to learn from examples using a similarity function that measures how similar or related two objects are. It has applications in ranking, recommendation systems, visual identity tracking, face verification, and speaker verification.

Unsupervised learning

Unsupervised learning algorithms take a set of data that contains only inputs, and find structure in the data, like grouping or clustering of data points. The algorithms, therefore, learn from test data that has not been labeled, classified or categorized. Instead of responding to feedback, unsupervised learning algorithms identify commonalities in the data and react based on the presence or absence of such commonalities in each new piece of data. A central application of unsupervised learning is in the field of density estimation in statistics, such as finding the probability density function. Though unsupervised learning encompasses other domains involving summarizing and explaining data features.

Cluster analysis is the assignment of a set of observations into subsets (called clusters) so that observations within the same cluster are similar according to one or more predesignated criteria,

while observations drawn from different clusters are dissimilar. Different clustering techniques make different assumptions on the structure of the data, often defined by some similarity metric and evaluated, for example, by internal compactness, or the similarity between members of the same cluster, and separation, the difference between clusters. Other methods are based on estimated density and graph connectivity.

Semi-supervised learning

Semi-supervised learning falls between unsupervised learning (without any labeled training data) and supervised learning (with completely labeled training data). Some of the training examples are missing training labels, yet many machine-learning researchers have found that unlabeled data, when used in conjunction with a small amount of labeled data, can produce a considerable improvement in learning accuracy.

In weakly supervised learning, the training labels are noisy, limited, or imprecise; however, these labels are often cheaper to obtain, resulting in larger effective training sets.

Reinforcement learning

Reinforcement learning is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. Due to its generality, the field is studied in many other disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, statistics and genetic algorithms. In machine learning, the environment is typically represented as a Markov decision process (MDP). Many reinforcement learning algorithms use dynamic programming techniques. Reinforcement learning algorithms do not assume knowledge of an exact mathematical model of the MDP, and are used when exact models are infeasible. Reinforcement learning algorithms are used in autonomous vehicles or in learning to play a game against a human opponent.

Self learning

Self-learning as a machine learning paradigm was introduced in 1982 along with a neural network capable of self-learning named crossbar adaptive array (CAA). It is a learning with no external rewards and no external teacher advice. The CAA self-learning algorithm computes, in a crossbar fashion, both decisions about actions and emotions (feelings) about consequence situations. The system is driven by the interaction between cognition and emotion. The self-learning algorithm updates a memory matrix $W = ||w(a,s)||$ such that in each iteration executes the following machine learning routine:

In situation s perform an action a ;
 Receive consequence situation s' ;
 Compute emotion of being in consequence situation $v(s')$;
 Update crossbar memory $w'(a,s) = w(a,s) + v(s')$.

It is a system with only one input, situation s , and only one output, action (or behavior) a . There is neither a separate reinforcement input nor an advice input from the environment. The backpropagated value (secondary reinforcement) is the emotion toward the consequence situation. The CAA exists in two environments, one is the behavioral environment where it behaves, and the other is the genetic environment, wherefrom it initially and only once receives initial emotions about situations to be encountered in the behavioral environment. After receiving the genome (species) vector from the genetic environment, the CAA learns a goal-seeking behavior, in an environment that contains both desirable and undesirable situations.

Feature learning

Several learning algorithms aim at discovering better representations of the inputs provided during training. Classic examples include principal components analysis and cluster analysis. Feature learning algorithms, also called representation learning algorithms, often attempt to preserve the information in their input but also transform it in a way that makes it useful, often as a pre-processing step before performing classification or predictions. This technique allows reconstruction of the inputs coming from the unknown data-generating distribution, while not being necessarily faithful to configurations that are implausible under that distribution. This replaces manual feature engineering, and allows a machine to both learn the features and use them to perform a specific task.

Feature learning can be either supervised or unsupervised. In supervised feature learning, features are learned using labeled input data. Examples include artificial neural networks, multilayer perceptrons, and supervised dictionary learning. In unsupervised feature learning, features are learned with unlabeled input data. Examples include dictionary learning, independent component analysis, autoencoders, matrix factorization and various forms of clustering.

Manifold learning algorithms attempt to do so under the constraint that the learned representation is low-dimensional. Sparse coding algorithms attempt to do so under the constraint that the learned representation is sparse, meaning that the mathematical model has many zeros. Multilinear subspace learning algorithms aim to learn low-dimensional representations directly from tensor representations for multidimensional data, without reshaping them into higher-dimensional vectors. Deep learning algorithms discover multiple levels of representation, or a hierarchy of features, with higher-level, more abstract features defined in terms of (or generating) lower-level features. It has been argued that an intelligent machine is one that learns a representation that disentangles the underlying factors of variation that explain the observed data.

Feature learning is motivated by the fact that machine learning tasks such as classification often require input that is mathematically and computationally convenient to process. However, real-world data such as images, video, and sensory data has not yielded to attempts to algorithmically define specific features. An alternative is to discover such features or representations thorough examination, without relying on explicit algorithms.

Sparse dictionary learning

Sparse dictionary learning is a feature learning method where a training example is represented as a linear combination of basis functions, and is assumed to be a sparse matrix. The method is strongly NP-hard and difficult to solve approximately. A popular heuristic method for sparse dictionary learning is the K-SVD algorithm. Sparse dictionary learning has been applied in several contexts. In classification, the problem is to determine the class to which a previously unseen training example belongs. For a dictionary where each class has already been built, a new training example is associated with the class that is best sparsely represented by the corresponding

dictionary. Sparse dictionary learning has also been applied in image de-noising. The key idea is that a clean image patch can be sparsely represented by an image dictionary, but the noise cannot.

Anomaly detection

In data mining, anomaly detection, also known as outlier detection, is the identification of rare items, events or observations which raise suspicions by differing significantly from the majority of the data. Typically, the anomalous items represent an issue such as bank fraud, a structural defect, medical problems or errors in a text. Anomalies are referred to as outliers, novelties, noise, deviations and exceptions.

In particular, in the context of abuse and network intrusion detection, the interesting objects are often not rare objects, but unexpected bursts of inactivity. This pattern does not adhere to the common statistical definition of an outlier as a rare object, and many outlier detection methods (in particular, unsupervised algorithms) will fail on such data unless it has been aggregated appropriately. Instead, a cluster analysis algorithm may be able to detect the micro-clusters formed by these patterns.

Three broad categories of anomaly detection techniques exist. Unsupervised anomaly detection techniques detect anomalies in an unlabeled test data set under the assumption that the majority of the instances in the data set are normal, by looking for instances that seem to fit least to the remainder of the data set. Supervised anomaly detection techniques require a data set that has been labeled as "normal" and "abnormal" and involves training a classifier (the key difference to many other statistical classification problems is the inherently unbalanced nature of outlier detection). Semi-supervised anomaly detection techniques construct a model representing normal behavior from a given normal training data set and then test the likelihood of a test instance to be generated by the model.

Robot learning

In developmental robotics, robot learning algorithms generate their own sequences of learning experiences, also known as a curriculum, to cumulatively acquire new skills through self-guided exploration and social interaction with humans. These robots use guidance mechanisms such as active learning, maturation, motor synergies and imitation.

Association rules

Association rule learning is a rule-based machine learning method for discovering relationships between variables in large databases. It is intended to identify strong rules discovered in databases using some measure of "interestingness".

Rule-based machine learning is a general term for any machine learning method that identifies, learns, or evolves "rules" to store, manipulate or apply knowledge. The defining characteristic of a rule-based machine learning algorithm is the identification and utilization of a set of relational rules that collectively represent the knowledge captured by the system. This is in contrast to other machine learning algorithms that commonly identify a singular model that can be universally applied to any instance in order to make a prediction. Rule-based machine learning approaches include learning classifier systems, association rule learning, and artificial immune systems.

Based on the concept of strong rules, Rakesh Agrawal, Tomasz Imieliński and Arun Swami introduced association rules for discovering regularities between products in large-scale transaction data recorded by point-of-sale (POS) systems in supermarkets. For example, the rule
$$\{\mathrm{onions, potatoes}\} \rightarrow \{\mathrm{burger}\} \implies \{\mathrm{onions, potatoes}\} \rightarrow \{\mathrm{burger}\}$$
 found in the sales data of a supermarket would indicate that if a customer buys onions and potatoes together, they are likely to also buy hamburger meat. Such information can be used as the basis for decisions about marketing activities such as promotional pricing or product placements. In addition to market basket analysis, association rules are employed today in application areas including Web usage mining, intrusion detection, continuous production, and bioinformatics. In contrast with sequence mining, association rule learning typically does not consider the order of items either within a transaction or across transactions.

Learning classifier systems (LCS) are a family of rule-based machine learning algorithms that combine a discovery component, typically a genetic algorithm, with a learning component, performing either supervised learning, reinforcement learning, or unsupervised learning. They seek to identify a set of context-dependent rules that collectively store and apply knowledge in a piecewise manner in order to make predictions.

Inductive logic programming (ILP) is an approach to rule-learning using logic programming as a uniform representation for input examples, background knowledge, and hypotheses. Given an encoding of the known background knowledge and a set of examples represented as a logical database of facts, an ILP system will derive a hypothesized logic program that entails all positive and no negative examples. Inductive programming is a related field that considers any kind of programming language for representing hypotheses (and not only logic programming), such as functional programs.

Inductive logic programming is particularly useful in bioinformatics and natural language processing. Gordon Plotkin and Ehud Shapiro laid the initial theoretical foundation for inductive machine learning in a logical setting. Shapiro built their first implementation (Model Inference System) in 1981: a Prolog program that inductively inferred logic programs from positive and negative examples. The term inductive here refers to philosophical induction, suggesting a theory to explain observed facts, rather than mathematical induction, proving a property for all members of a well-ordered set.

Models

Performing machine learning involves creating a model, which is trained on some training data and then can process additional data to make predictions. Various types of models have been used and researched for machine learning systems.

Artificial neural networks

An artificial neural network is an interconnected group of nodes, akin to the vast network of neurons in a brain. Here, each circular node represents an artificial neuron and an arrow represents a connection from the output of one artificial neuron to the input of another.

Artificial neural networks (ANNs), or connectionist systems, are computing systems vaguely inspired by the biological neural networks that constitute animal brains. Such systems "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules.

An ANN is a model based on a collection of connected units or nodes called "artificial neurons", which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit information, a "signal", from one artificial neuron to another. An artificial neuron that receives a signal can process it and then signal additional artificial neurons connected to it. In common ANN implementations, the signal at a connection between artificial neurons is a real number, and the output of each artificial neuron is computed by some non-linear function of the sum of its inputs. The connections between artificial neurons are called "edges". Artificial neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Artificial neurons may have a threshold such that the signal is only sent if the aggregate signal crosses that threshold. Typically, artificial neurons are aggregated into layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first layer (the input layer) to the last layer (the output layer), possibly after traversing the layers multiple times.

The original goal of the ANN approach was to solve problems in the same way that a human brain would. However, over time, attention moved to performing specific tasks, leading to deviations from biology. Artificial neural networks have been used on a variety of tasks, including computer vision, speech recognition, machine translation, social network filtering, playing board and video games and medical diagnosis.

Deep learning consists of multiple hidden layers in an artificial neural network. This approach tries to model the way the human brain processes light and sound into vision and hearing. Some successful applications of deep learning are computer vision and speech recognition.

Decision trees

Decision tree learning uses a decision tree as a predictive model to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). It is one of the predictive modeling approaches used in statistics, data mining, and machine learning. Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. In data mining, a decision tree describes data, but the resulting classification tree can be an input for decision making.

Support vector machines

Support vector machines (SVMs), also known as support vector networks, are a set of related supervised learning methods used for classification and regression. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that predicts whether a new example falls into one category or the other. An SVM training algorithm is a non-probabilistic, binary, linear classifier, although methods such as Platt scaling exist to use SVM in a probabilistic classification setting. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

Regression analysis

Regression analysis encompasses a large variety of statistical methods to estimate the relationship between input variables and their associated features. Its most common form is linear regression, where a single line is drawn to best fit the given data according to a mathematical criterion such as ordinary least squares. The latter is often extended by regularization (mathematics) methods to mitigate overfitting and bias, as in ridge regression. When dealing with non-linear problems, go-to models include polynomial regression (for example, used for trendline fitting in Microsoft

Excel, logistic regression (often used in statistical classification) or even kernel regression, which introduces non-linearity by taking advantage of the kernel trick to implicitly map input variables to higher-dimensional space.

Bayesian networks

A simple Bayesian network. Rain influences whether the sprinkler is activated, and both rain and the sprinkler influence whether the grass is wet.

A Bayesian network, belief network, or directed acyclic graphical model is a probabilistic graphical model that represents a set of random variables and their conditional independence with a directed acyclic graph (DAG). For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases. Efficient algorithms exist that perform inference and learning. Bayesian networks that model sequences of variables, like speech signals or protein sequences, are called dynamic Bayesian networks. Generalizations of Bayesian networks that can represent and solve decision problems under uncertainty are called influence diagrams.

Genetic algorithms

A genetic algorithm (GA) is a search algorithm and heuristic technique that mimics the process of natural selection, using methods such as mutation and crossover to generate new genotypes in the hope of finding good solutions to a given problem. In machine learning, genetic algorithms were used in the 1980s and 1990s. Conversely, machine learning techniques have been used to improve the performance of genetic and evolutionary algorithms.

Training models

Usually, machine learning models require a lot of data in order for them to perform well. Usually, when training a machine learning model, one needs to collect a large, representative sample of data from a training set. Data from the training set can be as varied as a corpus of text, a collection of images, and data collected from individual users of a service. Overfitting is something to watch out for when training a machine learning model. Trained models derived from biased data can result in skewed or undesired predictions. Algorithmic bias is a potential result from data not fully prepared for training.

Federated learning

Federated learning is an adapted form of distributed artificial intelligence to training machine learning models that decentralizes the training process, allowing for users' privacy to be maintained by not needing to send their data to a centralized server. This also increases efficiency by decentralizing the training process to many devices. For example, Gboard uses federated machine learning to train search query prediction models on users' mobile phones without having to send individual searches back to Google.

1.2 SCOPE OF THE PROJECT

We propose a machine learning methodology for diagnosing CKD. The CKD data set was obtained from the University of California Irvine (UCI) machine learning repository, which has a large number of missing values. KNN imputation was used to fill in the missing values, which selects several complete samples with the most similar measurements to process the missing data for each incomplete sample. Missing values are usually seen in real-life medical situations because patients may miss some measurements for various reasons. After effectively filling out the incomplete data set, six machine learning algorithms (logistic regression, random forest, support vector machine, k-nearest neighbor, naive Bayes classifier and feed forward neural network) were used to establish models. Among these machine learning models, random forest achieved the best performance with 99.75% diagnosis accuracy.

1.3 OBJECTIVE

By analyzing the misjudgments generated by the established models, we proposed an integrated model that combines logistic regression and random forest by using perceptron, which could achieve an average accuracy of 99.83% after ten times of simulation. Hence, we speculated that this methodology could be applicable to more complicated clinical data for disease diagnosis.

1.4 PROBLEM STATEMENT

Chen et al. used k-nearest neighbor (KNN), support vector machine (SVM) and soft independent modeling of class analogy to diagnose CKD, KNN and SVM achieved the highest accuracy of 99.7%. In addition, they used fuzzy rule-building expert system, fuzzy optimal associative memory and partial least squares discriminant analysis to diagnose CKD, and the range of accuracy in those models was 95.5%-99.6%. Their studies have achieved good results in the diagnosis of CKD. KNN imputation is used to fill in the missing values. To our knowledge, this is the first time that KNN imputation has been used for the diagnosis of CKD. In addition, building an integrated model is also a good way to improve the performance of separate individual models. The proposed methodology might effectively deal with the scene where patients are missing certain measurements before being diagnosed.

1.5 EXISTING SYSTEM

- Hodneland et al. utilized image registration to detect renal morphologic changes.
- Vasquez-Morales et al. established a classifier based on neural network using large-scale CKD data, and the accuracy of the model on their test data was 95%. In addition, most of the previous studies utilized the CKD data set that was obtained from the UCI machine learning repository.
- Chen et al. used k-nearest neighbor (KNN), support vector machine (SVM) and soft independent modeling of class analogy to diagnose CKD, KNN and SVM achieved the highest accuracy of 99.7%. In addition, they used fuzzy rule-building expert system, fuzzy optimal associative memory and partial least squares discriminant analysis to diagnose CKD, and the range of accuracy in those models was 95.5%-99.6%. Their studies have achieved good results in the diagnosis of CKD.

1.5.1 EXISTING SYSTEM DISADVANTAGES

- Most of them suffering from either the method used to impute missing values has a limited application range or relatively low accuracy.
- In the above models, the mean imputation is used to fill in the missing values and it depends on the diagnostic categories of the samples. As a result, their method could not be used when the diagnostic results of the samples are unknown. In reality, patients might miss some measurements for various reasons before diagnosing.
- In addition, for missing values in categorical variables, data obtained using mean imputation might have a large deviation from the actual values.

1.5.2 LITERATURE SURVEY

Title: Diagnosis of patients with chronic kidney disease by using two fuzzy classifiers

Author: Z. Chen, Z. Zhang, R. Zhu, Y. Xiang, and P. B. Harrington

Year:

Description:

The feasibility of two in-house fuzzy classifiers, fuzzy rule-building expert system (FuRES) and fuzzy optimal associative memory (FOAM), for diagnosis of patients with chronic kidney disease (CKD) was investigated. A linear classifier, partial least squares discriminant analysis (PLS-DA), was used for comparison. The CKD data used in this work were taken from the UCI Machine Learning Repository. Composite datasets were created by adding different levels of proportional noise to evaluate the robustness of the two fuzzy approaches. Firstly, 11 levels of proportional noises were added to each numeric attribute of the training and prediction sets one after another, and then these simulated training and prediction sets were combined in pairs. Thus, a grid with 121 groups of simulated data was generated, and classification rates for these 121 pairs were compared. Secondly, the performances of two fuzzy classifiers using the simulated datasets, in which 11 levels of noise were randomly distributed to each numeric attribute, were compared and the average prediction rates of FuRES and FOAM were $98.1 \pm 0.5\%$ and $97.2 \pm 1.2\%$, respectively, with 200 bootstrap Latin partitions. The PLS-DA can give $94.3 \pm 0.8\%$ with the identical evaluation. Confluent datasets comprised of the original and modified datasets were also used to evaluate FuRES, FOAM, and PLS-DA classification models. The average prediction rates of FuRES and FOAM obtained from 200 bootstrapped evaluations were $99.2 \pm 0.3\%$ and $99.0 \pm 0.3\%$. PLS-DA yields slightly worse accuracy with $95.9 \pm 0.6\%$. The results demonstrate that both FuRES and FOAM perform well on the identification of CKD patients, while FuRES is more robust than FOAM. These two fuzzy classifiers are useful tools for the diagnosis of CKD patients with satisfactory robustness, and can also be used for other kinds of patients.

Title: Diagnosis of chronic kidney disease by using random forest

Author: A. Subasi, E. Alickovic, and J. Kevric

Year:

Description:

Chronic kidney disease (CKD) is a global public health problem, affecting approximately 10% of the population worldwide. Yet, there is little direct evidence on how CKD can be diagnosed in a systematic and automatic manner. This paper investigates how CKD can be diagnosed by using machine learning (ML) techniques. ML algorithms have been a driving force in detection of abnormalities in different physiological data, and are, with a great success, employed in different classification tasks. In the present study, a number of different ML classifiers are experimentally validated to a real data set, taken from the UCI Machine Learning Repository, and our findings are compared with the findings reported in the recent literature. The results are quantitatively and qualitatively discussed and our findings reveal that the random forest (RF) classifier achieves the near-optimal performances on the identification of CKD subjects. Hence, we show that ML algorithms serve important function in diagnosis of CKD, with satisfactory robustness, and our findings suggest that RF can also be utilized for the diagnosis of similar diseases.

Title: Prevalence of chronic kidney disease in China: A cross sectional survey

Author: L. Zhang

Year:

Description:

Background: The prevalence of chronic kidney disease is high in developing countries. However, no national survey of chronic kidney disease has been done incorporating both estimated glomerular filtration rate (eGFR) and albuminuria in a developing country with the economic diversity of China. We aimed to measure the prevalence of chronic kidney disease in China with such a survey.

Methods: We did a cross-sectional survey of a nationally representative sample of Chinese adults. Chronic kidney disease was defined as eGFR less than 60 mL/min per 1.73 m² or the presence of albuminuria. Participants completed a lifestyle and medical history questionnaire and had their blood pressure measured, and blood and urine samples taken. Serum creatinine was measured and used to estimate glomerular filtration rate. Urinary albumin and creatinine were tested to assess albuminuria. The crude and adjusted prevalence of indicators of kidney damage were calculated and factors associated with the presence of chronic kidney disease analysed by logistic regression.

Title: Incorporating temporal EHR data in predictive models for risk stratification of renal function deterioration

Author: A. Singh, G. Nadkarni, O. Gottesman, S. B. Ellis, E. P. Bottinger, and J. V. Guttag

Year:

Description:

Predictive models built using temporal data in electronic health records (EHRs) can potentially play a major role in improving management of chronic diseases. However, these data present a multitude of technical challenges, including irregular sampling of data and varying length of available patient history. In this paper, we describe and evaluate three different approaches that use machine learning to build predictive models using temporal EHR data of a patient. The first approach is a commonly used non-temporal approach that aggregates values of the predictors in the patient's medical history. The other two approaches exploit the temporal dynamics of the data. The two temporal approaches vary in how they model temporal information and handle missing data. Using data from the EHR of Mount Sinai Medical Center, we learned and evaluated the models in the context of predicting loss of estimated glomerular filtration rate (eGFR), the most common assessment of kidney function. Our results show that incorporating temporal information in patient's medical history can lead to better prediction of loss of kidney function. They also demonstrate that exactly how this information is incorporated is important. In particular, our results demonstrate that the relative importance of different predictors varies over time, and that using multi-task learning to account for this is an appropriate way to robustly capture the temporal dynamics in EHR data. Using a case study, we also demonstrate how the multi-task learning based model can yield predictive models with better performance for identifying patients at high risk of short-term loss of kidney function.

Title: Prevalence of chronic kidney disease in an adult population

Author: A. M. Cueto-Manzano, L. Cortés-Sanabria, H. R. Martínez-Ramírez,
E. Rojas-Campos, B. Gómez-Navarro, and M. Castellero-Manzano

Year:

Description:

Background and aims: One strategy to prevent and manage chronic kidney disease (CKD) is to offer screening programs. The aim of this study was to determine the percentage prevalence and risk factors of CKD in a screening program performed in an adult general population.

Methods: This is a cross-sectional study. Six-hundred ten adults (73% women, age 51 ± 14 years) without previously known CKD were evaluated. Participants were subjected to a questionnaire, blood pressure measurement and anthropometry. Glomerular filtration rate estimated by CKD-EPI formula and urine tested with albuminuria dipstick.

Results: More than 50% of subjects reported family antecedents of diabetes mellitus (DM), hypertension and obesity, and 30% of CKD. DM was self-reported in 19% and hypertension in 29%. During screening, overweight/obesity was found in 75%; women had a higher frequency of obesity (41 vs. 34%) and high-risk abdominal waist circumference (87 vs. 75%) than men. Hypertension (both self-reported and diagnosed in screening) was more frequent in men (49%) than in women (38%). CKD was found in 14.7%: G1, 5.9%; G2, 4.5%; G3a, 2.6%; G3b, 1.1%, G4, 0.3%; and G5, 0.3%. Glomerular filtration rate was mildly/moderately reduced in 2.6%, moderately/severely reduced in 1.1%, and severely reduced in <1%. Abnormal albuminuria was found in 13%. CKD was predicted by DM, hypertension and male gender.

1.6 PROPOSED SYSTEM

- KNN imputation is used to fill in the missing values. To our knowledge, this is the first time that KNN imputation has been used for the diagnosis of CKD. In addition, building an integrated model is also a good way to improve the performance of separate individual models. The proposed methodology might effectively deal with the scene where patients are missing certain measurements before being diagnosed.
- In addition, the resulting integrated model shows a higher accuracy. Therefore, it is speculated that this methodology might be applicable to the clinical data in the actual medical diagnosis.

1.6.1 PROPOSED SYSTEM ADVANTAGES

- We propose a methodology to extend application range of the CKD diagnostic models.
- At the same time, the accuracy of the model is further improved.

CHAPTER 2

PROJECT DESCRIPTION

2.1 GENERAL

The implemented chatbot will solve queries of the users, provide information to users as they require, improve quality of service time and make customers happy by providing smart solutions. It also improves productivity by providing 24/7 service, reduces crowd at help desk and also reduces human efforts.

2.2 METHODOLOGIES

2.2.1 MODULES NAME:

This project having the following five modules:

- Data Collection
- Dataset
- Data Preparation
- Model Selection
- Analyze and Prediction
- Accuracy on test set
- Saving the Trained Model

2.2.2 MODULES EXPLANATION AND DIAGRAM

➤ Data Collection:

This is the first real step towards the real development of a machine learning model, collecting data. This is a critical step that will cascade in how good the model will be, the more and better data that we get, the better our model will perform.

There are several techniques to collect the data, like web scraping, manual interventions and etc.

The dataset used in this Chronic kidney disease dataset taken from UCI:https://archive.ics.uci.edu/ml/datasets/chronic_kidney_disease

➤ Dataset:

The dataset consists of 400 individual data. There are 26 columns in the dataset, which are described below.

age	-	age
bp	-	blood pressure
sg	-	specific gravity
al	-	albumin
su	-	sugar
rbc	-	red blood cells
pc	-	pus cell
pcc	-	pus cell clumps
ba	-	bacteria
bgr	-	blood glucose random
bu	-	blood urea
sc	-	serum creatinine
sod	-	sodium
pot	-	potassium
hemo	-	hemoglobin
pcv	-	packed cell volume
wc	-	white blood cell count
rc	-	red blood cell count

htn	-	hypertension
dm	-	diabetes mellitus
cad	-	coronary artery disease
appet	-	appetite
pe	-	pedal edema
ane	-	anemia
class	-	classification

➤ **Data Preparation:**

we will transform the data. By getting rid of missing data and removing some columns. First, we will create a list of column names that we want to keep or retain.

Next, we drop or remove all columns except for the columns that we want to retain.

Finally, we drop or remove the rows that have missing values from the data set.

Split into training and evaluation sets

➤ **Model Selection:**

It is a supervised learning algorithm that includes more dependent variables. The response of this algorithm is in the binary form. Logistics regression can provide the continuous outcome of a specific data. This algorithm consists of statistical model with binary variables.

➤ **Analyze and Prediction:**

In the actual dataset, we chose only 19 features :

1. Age(numerical) --> age in years
2. Blood Pressure(numerical) bp in mm/Hg
3. Specific Gravity(nominal) sg - (1.005,1.010,1.015,1.020,1.025)
4. Albumin(nominal)al - (0,1,2,3,4,5)

5. Sugar(nominal) su - (0,1,2,3,4,5)
6. Blood Glucose Random(numerical) bgr in mgs/dl
7. Blood Urea(numerical) bu in mgs/dl
8. Serum Creatinine(numerical) sc in mgs/dl
9. Sodium(numerical) sod in mEq/L
10. Potassium(numerical) pot in mEq/L
11. Haemoglobin(numerical) hemo in gms
12. Packed Cell Volume(numerical)
13. White Blood Cell Count(numerical) wc in cells/cumm
14. Hypertension(nominal) htn - (yes,no)
15. Diabetes Mellitus(nominal) dm - (yes,no)
16. Coronary Artery Disease(nominal) cad - (yes,no)
17. Appetite(nominal) ppet - (good,poor)
18. Pedal Edema(nominal) pe - (yes,no)
19. Anemia(nominal)ane - (yes,no)
20. Chronic_kidney_disease: Displays whether the individual is suffering from kidney disease or not

➤ **Accuracy on test set:**

We got a accuracy of 92.7% on test set.

➤ **Saving the Trained Model:**

Once you're confident enough to take your trained and tested model into the production-ready environment, the first step is to save it into a .h5 or. pkl file using a library like pickle.

Make sure you have pickle installed in your environment.

Next, let's import the module and dump the model into. pkl file

2.2.3 GIVEN INPUT EXPECTED OUTPUT:

➤ INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

➤ OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

➤ OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system

through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

CHAPTER 3

REQUIREMENTS ENGINEERING

3.1 GENERAL

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources.

3.2 HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It should state what the system do and not how it should be implemented.

HARDWARE

- Processor : Pentium Iv 2.6 Ghz, Intel Core 2 Duo.
- Ram : 512 Mb Dd Ram
- Monitor : 15” Color
- Hard Disk : 40 GB

3.3 SOFTWARE REQUIREMENTS

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team's progress throughout the development activity.

SOFTWARE

- Front End : HTML,CSS3,JAVASCRIPT
- Back End : MY SQL 5.5
- Operating System : Windows 7
- IDE : Eclipse

3.4 FUNCTIONAL REQUIREMENTS

A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behaviour, and outputs.

The contributions of the proposed work are as follows.

- 1) we used KNN imputation to fill in the missing values in the data set, which could be applied to the data set with the diagnostic categories are unknown.
- 2) Logistic regression (LOG), RF, SVM, KNN, naive Bayes classifier (NB) and feed forward neural network (FNN) were used to establish CKD diagnostic models on the complete CKD data sets. The models with better performance were extracted for misjudgment analysis.
- 3) An integrated model that combines LOG and RF by using perceptron was established and it improved the performance of the component models in CKD diagnosis after the missing values were filled by KNN imputation.

3.5 NON-FUNCTIONAL REQUIREMENTS

The major non-functional Requirements of the system are as follows

➤ **Usability**

The system is designed with completely automated process hence there is no or less user intervention.

➤ **Reliability**

The system is more reliable because of the qualities that are inherited from the chosen platform java. The code built by using java is more reliable.

➤ **Performance**

This system is developing in the high level languages and using the advanced front-end and back-end technologies it will give response to the end user on client system with in very less time.

➤ **Supportability**

The system is designed to be the cross platform supportable. The system is supported on a wide range of hardware and any software platform, which is having JVM, built into the system.

➤ **Implementation**

The system is implemented in web environment using struts framework. The apache tomcat is used as the web server and windows xp professional is used as the platform. Interface the user interface is based on Struts provides HTML Tag

CHAPTER 4

DESIGN ENGINEERING

4.1 GENERAL

Design Engineering deals with the various UML [Unified Modelling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into finished product.

4.1.1 Use Case Diagram

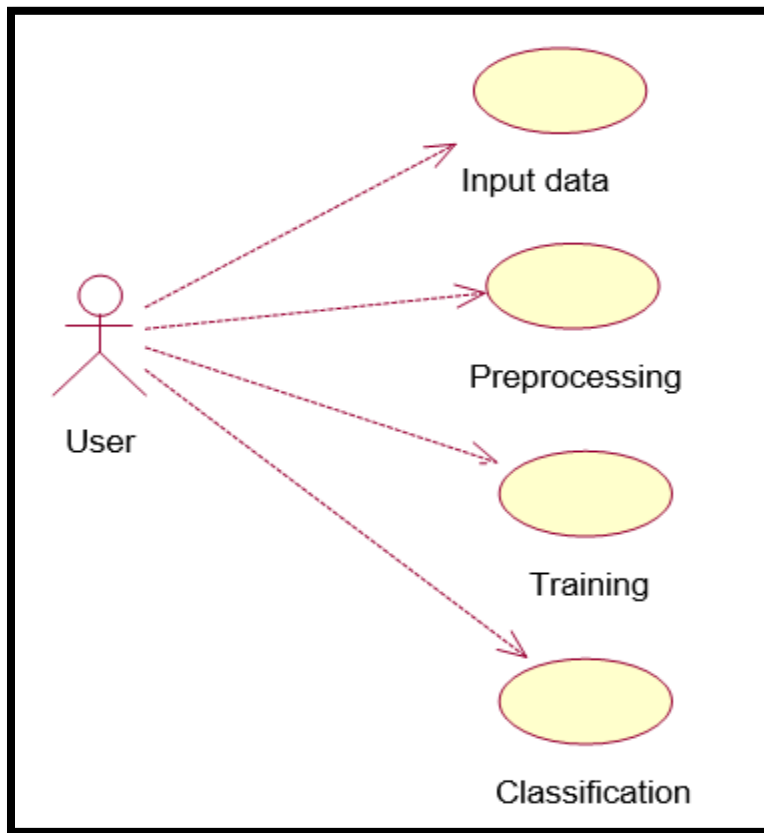


Figure 4.1.1 -Use case Diagram for Diagnosing Chronic Kidney Disease

EXPLANATION:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

4.1.2 Class Diagram

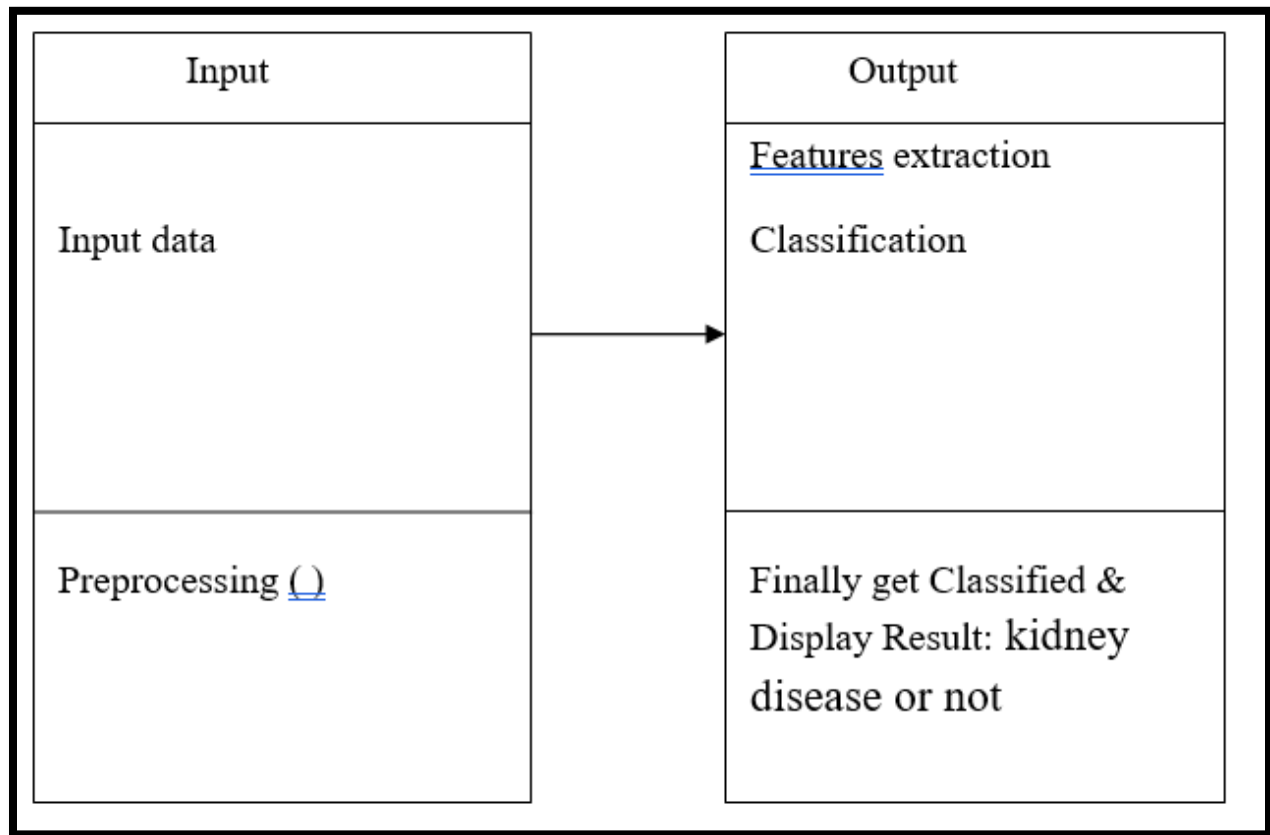


Figure 4.1.2 -Class Diagram for Diagnosing Chronic Kidney Disease

EXPLANATION

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

4.1.3 Sequence Diagram

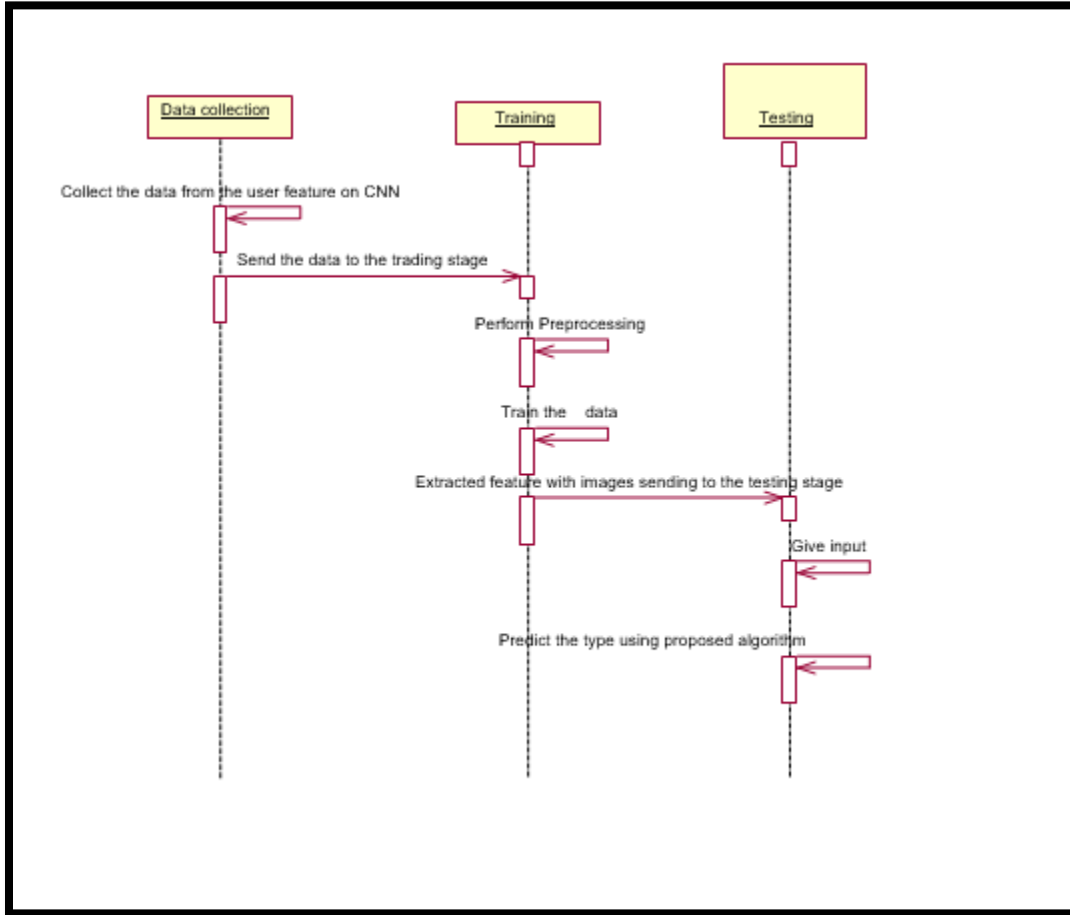


Figure 4.1.3 -Sequence Diagram for Diagnosing Chronic Kidney Disease

EXPLANATION:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

4.1.4 Activity Diagram

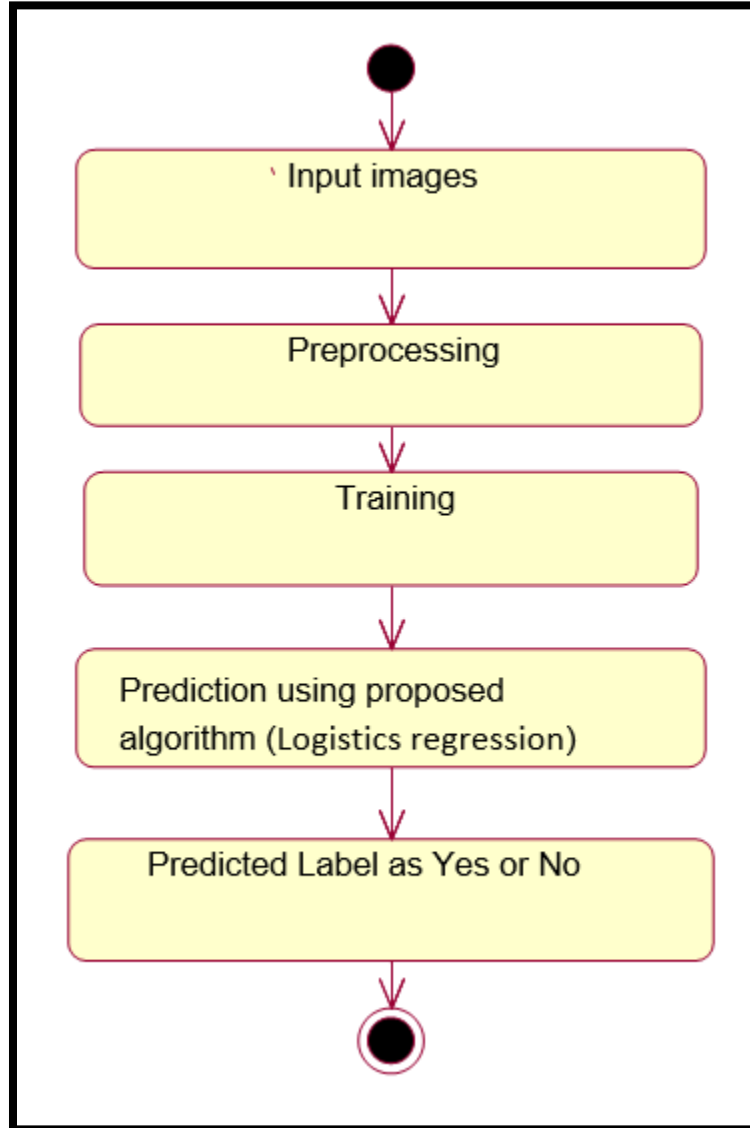


Figure 4.1.4 -Activity Diagram for Diagnosing Chronic Kidney Disease

EXPLANATION:

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 can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

4.1.5 Data Flow Diagram:

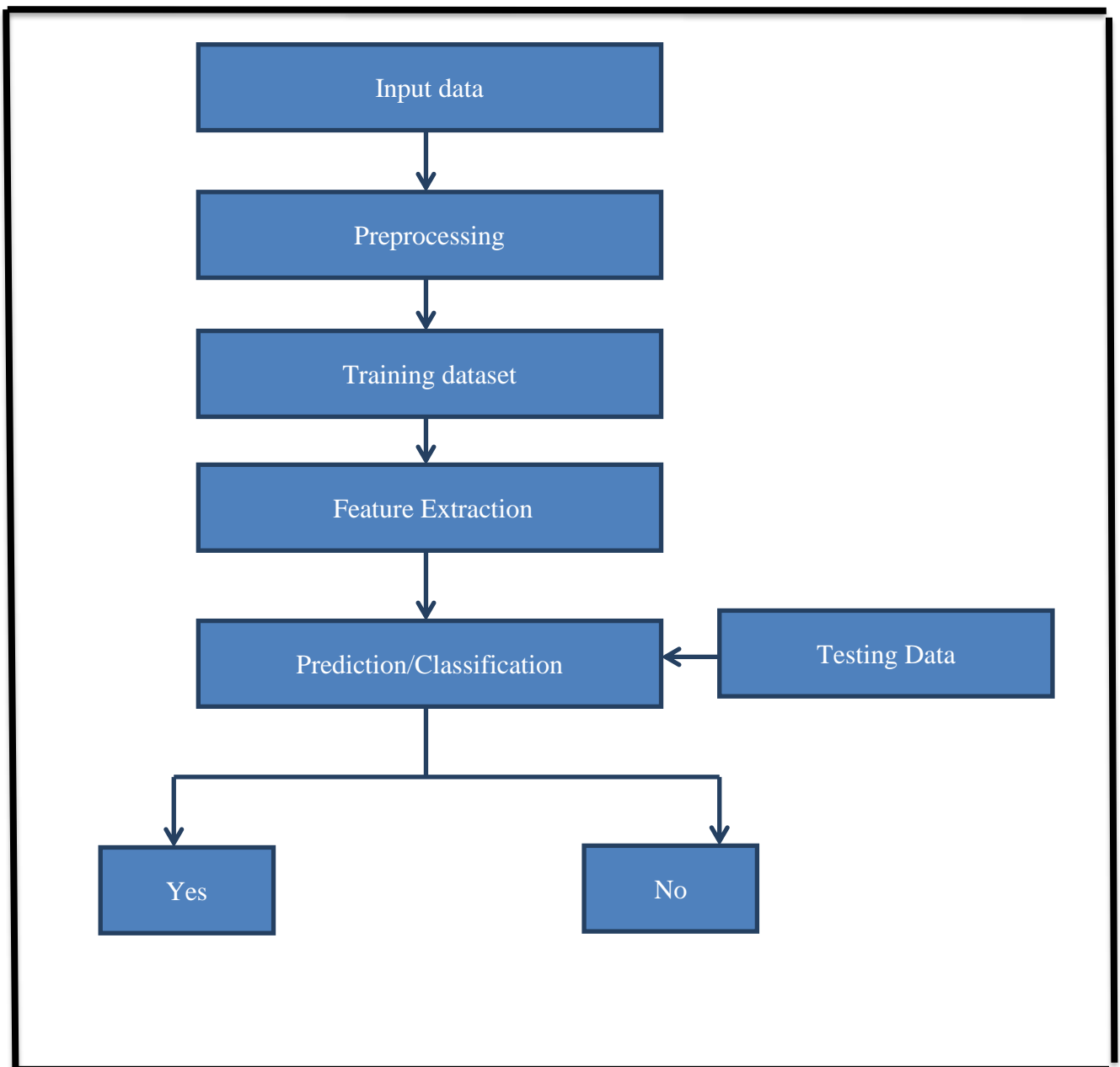


Figure 4.1.5- Data Flow Diagram for Diagnosing Chronic Kidney Disease

EXPLANATION:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

4.2 System Architecture

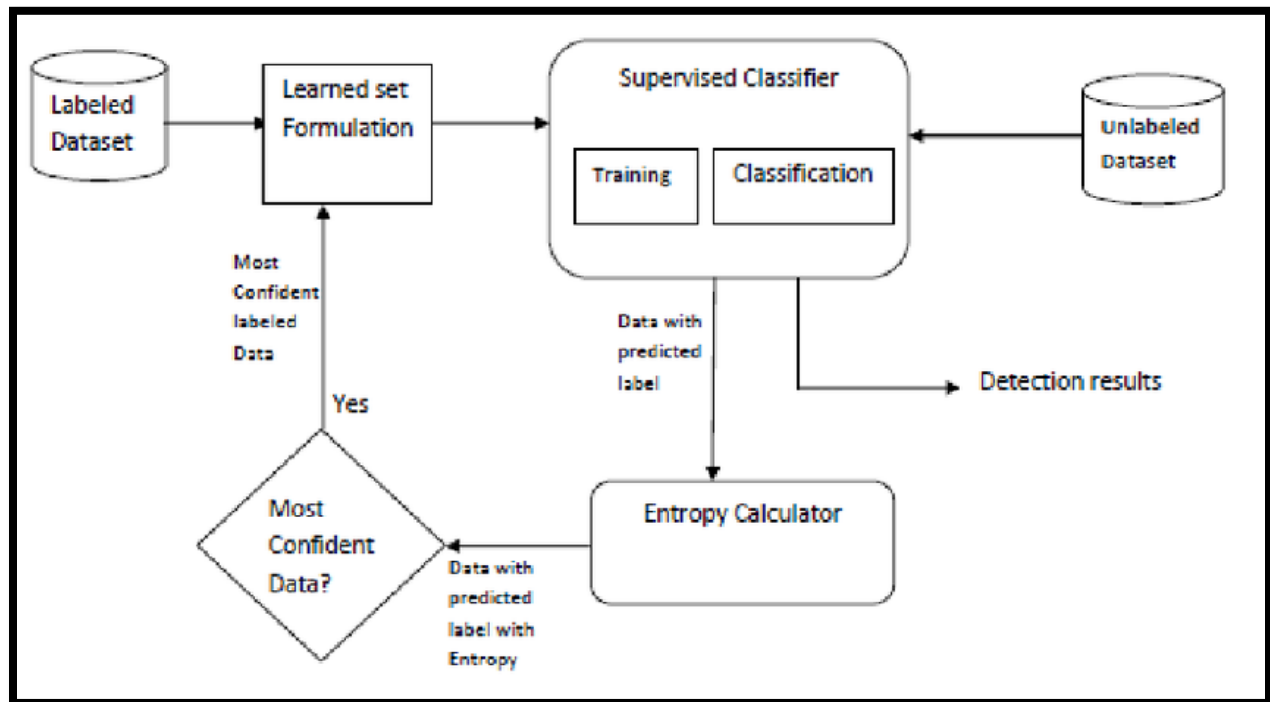


Figure 4.2- System Architecture for Diagnosing Chronic Kidney Disease

Now-a-days data is playing a major role. There is large amount of data from various sector. So the first important step is data collection. The data should be collected for future use. As data comes from various sources it may consists of errors. So data pre-processing is the next step. In this whatever data collected will be processed by removing unnecessary data and corrects errors in data. Once the data is pre-processed the required features need to be extracted. Machine Learning Algorithms are used to classify the data based on required models like k-nearest neighbour, Decision trees, SVM, etc. The data is classified as Testing data (70%) and Training data (30%). The testing data is compared with training data by using machine learning models. Machine learning Algorithms is classified as Supervised learning, Unsupervised Learning and Reinforcement. Finally the output can be visualized as graphs, charts etc which helps us to understand the situation in easy manner. We can also visualize in form of dashboard which helps to compare and see the efficiency of different algorithms on a screen.

CHAPTER 5

DEVELOPMENT TOOLS

5.1 GENERAL

Python:

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** – You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** – Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** – Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

Python's features include –

- **Easy-to-learn** – Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** – Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** – Python's source code is fairly easy-to-maintain.

- **A broad standard library** – Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** – Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** – Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** – You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** – Python provides interfaces to all major commercial databases.
- **GUI Programming** – Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** – Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below –

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

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

Getting Python

The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python <https://www.python.org>.

Windows Installation

Here are the steps to install Python on Windows machine.

- Open a Web browser and go to <https://www.python.org/downloads/>.

- Follow the link for the Windows installer python-XYZ.msifile where XYZ is the version you need to install.
- To use this installer python-XYZ.msi, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local machine and then run it to find out if your machine supports MSI.
- Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done.

The Python language has many similarities to Perl, C, and Java. However, there are some definite differences between the languages.

First Python Program

Let us execute programs in different modes of programming.

Interactive Mode Programming

Invoking the interpreter without passing a script file as a parameter brings up the following prompt –

```
$ python
Python2.4.3(#1,Nov112010,13:34:43)
[GCC 4.1.220080704(RedHat4.1.2-48)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Type the following text at the Python prompt and press the Enter –

```
>>> print "Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in **print ("Hello, Python!")**; However in Python version 2.4.3, this produces the following result –

```
Hello, Python!
```


Script Mode Programming

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension **.py**. Type the following source code in a test.py file –

```
print"Hello, Python!"
```

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows –

```
$ python test.py
```

This produces the following result –

```
Hello, Python!
```

Flask Framework:

Flask is a web application framework written in Python. Armin Ronacher, who leads an international group of Python enthusiasts named Pocco, develops it. Flask is based on Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.

Http protocol is the foundation of data communication in world wide web. Different methods of data retrieval from specified URL are defined in this protocol.

The following table summarizes different http methods –

Sr.No	Methods & Description
1	GET Sends data in unencrypted form to the server. Most common method.

2	HEAD Same as GET, but without response body
3	POST Used to send HTML form data to server. Data received by POST method is not cached by server.
4	PUT Replaces all current representations of the target resource with the uploaded content.
5	DELETE Removes all current representations of the target resource given by a URL

By default, the Flask route responds to the **GET** requests. However, this preference can be altered by providing methods argument to **route()** decorator.

In order to demonstrate the use of **POST** method in URL routing, first let us create an HTML form and use the **POST** method to send form data to a URL.

Save the following script as login.html

```
<html>

<body>

<formaction="http://localhost:5000/login"method="post">

<p>Enter Name:</p>

<p><inputtype="text"name="nm"/></p>

<p><inputtype="submit"value="submit"/></p>
```

```
</form>
```

```
</body>
```

```
</html>
```

Now enter the following script in Python shell.

```
from flask import Flask, redirect, url_for, request

app = Flask(__name__)

@app.route('/success/<name>')

def success(name):

    return 'welcome %s' % name

@app.route('/login', methods=['POST', 'GET'])

def login():

    if request.method == 'POST':

        user = request.form['nm']

        return redirect(url_for('success', name= user))

    else:

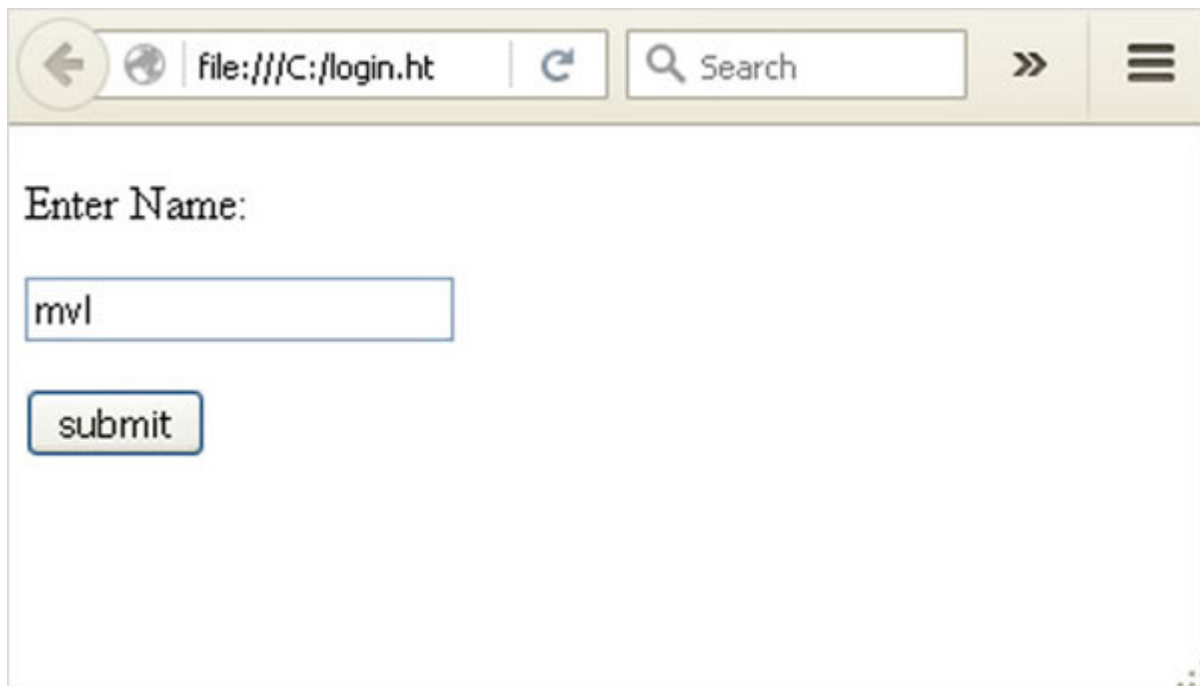
        user = request.args.get('nm')

        return redirect(url_for('success', name= user))

if __name__ == '__main__':

    app.run(debug = True)
```

After the development server starts running, open **login.html** in the browser, enter name in the text field and click **Submit**.

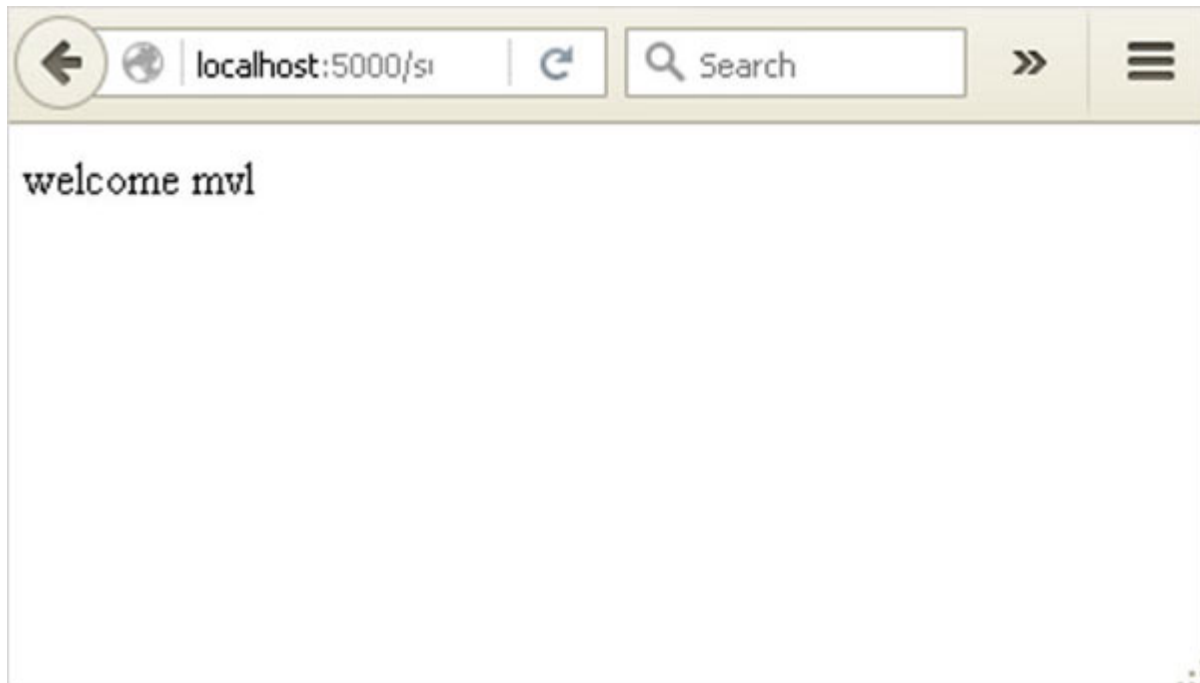
A screenshot of a web browser window. The address bar shows the file path 'file:///C:/login.ht'. The page content includes the text 'Enter Name:', a text input field containing 'mvl', and a 'submit' button.

Form data is POSTed to the URL in action clause of form tag.

http://localhost/login is mapped to the **login()** function. Since the server has received data by **POST** method, value of 'nm' parameter obtained from the form data is obtained by –

```
user = request.form['nm']
```

It is passed to **‘/success’** URL as variable part. The browser displays a **welcome** message in the window.



Change the method parameter to '**GET**' in **login.html** and open it again in the browser. The data received on server is by the **GET** method. The value of 'nm' parameter is now obtained by –

```
User = request.args.get('nm')
```

Here, **args** is dictionary object containing a list of pairs of form parameter and its corresponding value. The value corresponding to 'nm' parameter is passed on to '/success' URL as before.

What is Python?

Python is a popular programming language. It was created in 1991 by Guido van Rossum.

It is used for:

- web development (server-side),

- software development,
- mathematics,
- system scripting.

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.

Good to know

- The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

Python Syntax compared to other programming languages

- Python was designed to for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.

- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Python Install

Many PCs and Macs will have python already installed.

To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

```
C:\Users\Your Name>python --version
```

To check if you have python installed on a Linux or Mac, then on linux open the command line or on Mac open the Terminal and type:

```
python --version
```

If you find that you do not have python installed on your computer, then you can download it for free from the following website: <https://www.python.org/>

Python Quickstart

Python is an interpreted programming language, this means that as a developer you write Python (.py) files in a text editor and then put those files into the python interpreter to be executed.

The way to run a python file is like this on the command line:

```
C:\Users\Your Name>python helloworld.py
```

Where "helloworld.py" is the name of your python file.

Let's write our first Python file, called helloworld.py, which can be done in any text editor.

```
helloworld.py
```

```
print("Hello, World!")
```

Simple as that. Save your file. Open your command line, navigate to the directory where you saved your file, and run:

```
C:\Users\Your Name>python helloworld.py
```

The output should read:

```
Hello, World!
```

Congratulations, you have written and executed your first Python program.

The Python Command Line

To test a short amount of code in python sometimes it is quickest and easiest not to write the code in a file. This is made possible because Python can be run as a command line itself.

Type the following on the Windows, Mac or Linux command line:

```
C:\Users\Your Name>python
```

From there you can write any python, including our hello world example from earlier in the tutorial:

```
C:\Users\Your Name>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, World!")
```

Which will write "Hello, World!" in the command line:


```
C:\Users\Your Name>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, World!")
Hello, World!
```

Whenever you are done in the python command line, you can simply type the following to quit the python command line interface:

```
exit()
```

Execute Python Syntax

As we learned in the previous page, Python syntax can be executed by writing directly in the Command Line:

```
>>> print("Hello, World!")
Hello, World!
```

Or by creating a python file on the server, using the .py file extension, and running it in the Command Line:

```
C:\Users\Your Name>python myfile.py
```

Python Indentations

Where in other programming languages the indentation in code is for readability only, in Python the indentation is very important.

Python uses indentation to indicate a block of code.

Example

```
if 5 > 2:
    print("Five is greater than two!")
```

Python will give you an error if you skip the indentation:

Example

```
if 5 > 2:
print("Five is greater than two!")
```

Comments

Python has commenting capability for the purpose of in-code documentation.

Comments start with a #, and Python will render the rest of the line as a comment:

Example

Comments in Python:

```
#This is a comment.  
print("Hello, World!")
```

Docstrings

Python also has extended documentation capability, called docstrings.

Docstrings can be one line, or multiline.

Python uses triple quotes at the beginning and end of the docstring:

Example

Docstrings are also comments:

```
"""This is a  
multiline docstring."""  
print("Hello, World!")
```

CHAPTER 6

IMPLEMENTATION

6.1 GENERAL

The Implementation is nothing but sores code of project.

6.2 IMPLEMENTATION

Coding:

```
import numpy as np
from flask import Flask, request, render_template
from flask_cors import CORS, cross_origin
import pickle
import pandas as pd

app = Flask(__name__)
cors = CORS(app)
app.config['CORS_HEADERS'] = 'Content-Type'
model = pickle.load(open('kid1.pkl', 'rb'))
#json_file = open('model.json', 'r')

#loaded_model_json = json_file.read()
#json_file.close()

#loaded_model = model_from_json(loaded_model_json)

#loaded_model.load_weights("model9954.h5")
#print("Loaded model from disk")

@app.route('/')
@app.route('/first')
def first():
    return render_template('first.html')
@app.route('/index')
def index():
    return render_template('index.html')
@app.route('/abstract')
def abstract():
    return render_template('abstract.html')
@app.route('/future')
def future():
```

```

        return render_template('future.html')

@app.route('/chart')
def chart():
    return render_template('chart.html')
@app.route('/pie')
def pie():
    return render_template('pie.html')
@app.route('/upload')
def upload():
    return render_template('upload.html')
@app.route('/preview', methods=["POST"])
def preview():
    if request.method == 'POST':
        dataset = request.files['datasetfile']
        df = pd.read_csv(dataset, encoding = 'unicode_escape')
        df.set_index('Id', inplace=True)
        return render_template("preview.html", df_view = df)

@app.route('/home')
def home():
    return render_template('index2.html')

@app.route('/predict', methods=['POST'])
@cross_origin()
def predict():
    int_features = [float(x) for x in request.form.values()]
    final_features = [np.array(int_features)]

    # prediction = model.predict_proba(final_features)
    prediction = model.predict(final_features)

    output = prediction[0]
    if output<0.5:
        output1='Normal'
    elif output>0.5:
        output1='Chronic Kidney Disease'

    return render_template('index2.html', prediction_text='The
Patient has {}'.format(output1))

if __name__ == "__main__":
    app.run(debug=True)

```

CHAPTER 7

SNAPSHOTS

7.1 GENERAL

This project implements a web application using Python and the Server process is maintained using the SOCKET & SERVERSOCKET and the Design part is played by Cascading Style Sheet.

7.2 VARIOUS SNAPSHOTS



ABSTRACT

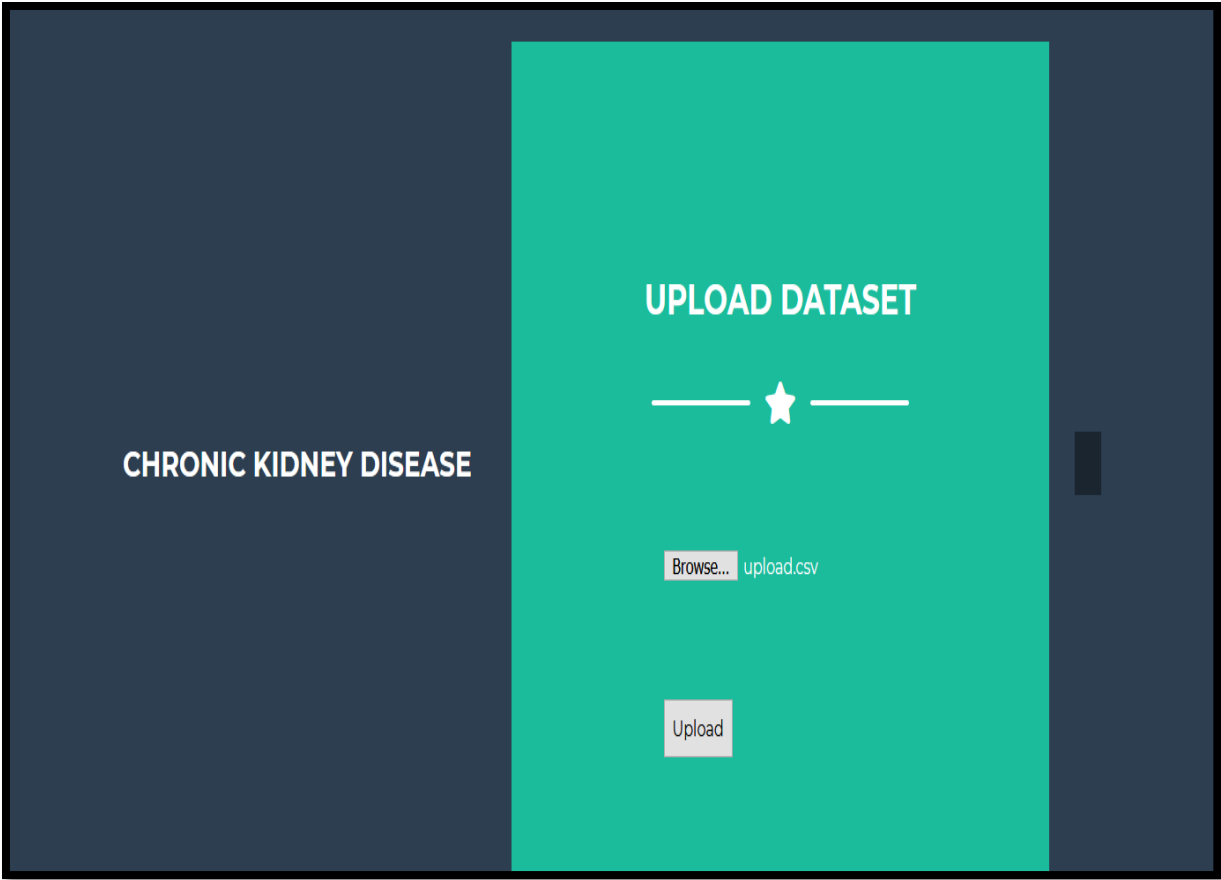
Chronic kidney disease (CKD) is a global health problem with high morbidity and mortality rate, and it induces other diseases. Since there are no obvious symptoms during the early stages of CKD, patients often fail to notice the disease. Early detection of CKD enables patients to receive timely treatment to ameliorate the progression of this disease. Machine learning models can effectively aid clinicians achieve this goal due to their fast and accurate recognition performance. In this study, we propose a machine learning methodology for diagnosing CKD. The CKD data set was obtained from the University of California Irvine (UCI) machine learning repository, which has a large number of missing values. KNN imputation was used to fill in the missing values, which selects several complete samples with the most similar measurements to process the missing data for each incomplete sample. Missing values are usually seen in real-life medical situations because patients may miss some measurements for various reasons. After effectively filling out the incomplete data set, six machine learning algorithms (logistic regression, random forest, support vector machine, k-nearest neighbor, naive Bayes classifier and feed forward neural network) were used to establish models. Among these machine learning models, random forest achieved the best performance with 99.75% diagnosis accuracy. By analyzing the misjudgments generated by the established models, we proposed an integrated model that combines logistic regression and random forest by using perceptron, which could achieve an average accuracy of 99.83% after ten times of simulation. Hence, we speculated that this methodology could be applicable to more complicated clinical data for disease diagnosis.

NEXT

ACCOUNT LOGIN

Username	Password
admin	*****
SIGN IN	

[Forgot password?](#)



CHRONIC KIDNEY DISEASE

PREVIEW



	age	bp	sg	al	su	bgr	bu	sc	sod	pot	hemo	pcv
Id												
1	48.000000	80.000000	1.020000	1.000000	0.000000	121.000000	36.000000	1.200000	135.297553	3.000617	15.400000	44.000000
2	7.000000	50.000000	1.020000	4.000000	0.000000	134.383588	18.000000	0.800000	131.564702	4.401237	11.300000	38.000000
3	62.000000	80.000000	1.010000	2.000000	3.000000	423.000000	53.000000	1.800000	134.149877	6.809881	9.600000	31.000000

CHRONIC KIDNEY DISEASE

398	12.000000	80.000000	1.020000	0.000000	0.000000	100.000000	26.000000	0.600000	137.000000	4.400000	15.800000	49.000000
399	17.000000	60.000000	1.025000	0.000000	0.000000	114.000000	50.000000	1.000000	135.000000	4.900000	14.200000	51.000000
400	58.000000	80.000000	1.025000	0.000000	0.000000	131.000000	18.000000	1.100000	141.000000	3.500000	15.800000	53.000000

[Click to Train | Test](#)



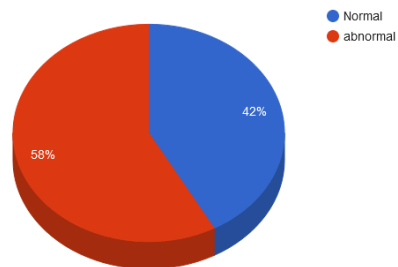
Kidney Disease Diagnosis



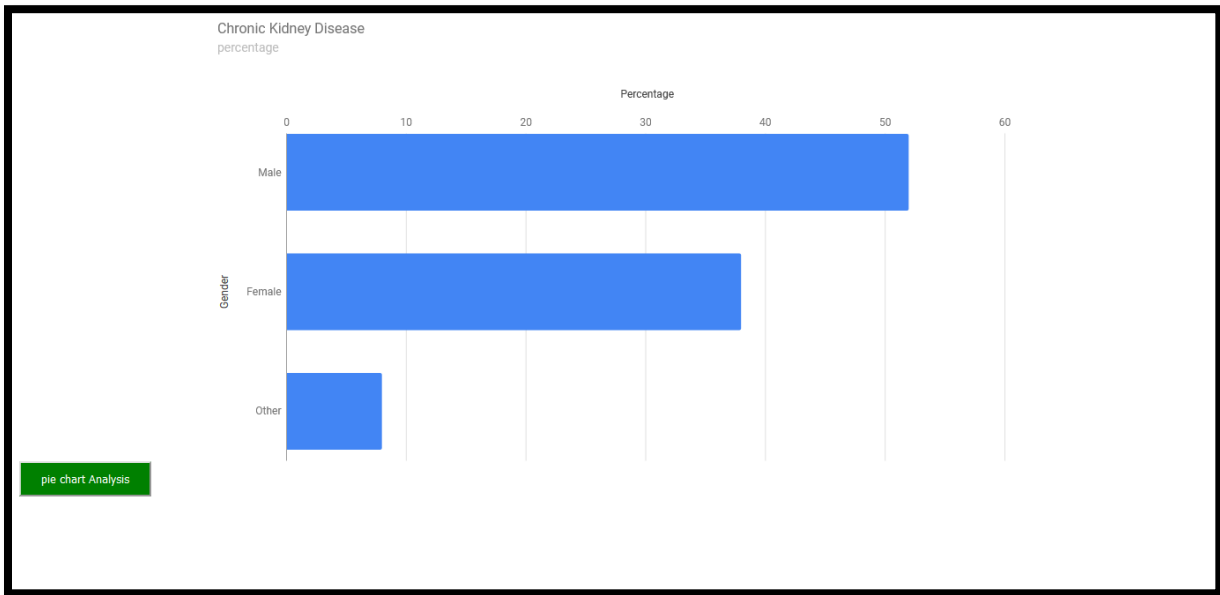
Kidney Disease Diagnosis

48
80
1.02
1
0
121
36
1.2
135.2975
3.006
15.4
44

Chronic Kidney Disease (pie chart analysis)



Future



CONCLUSION AND FUTURE WORK

The proposed CKD diagnostic methodology is feasible in terms of data imputation and samples diagnosis. After unsupervised imputation of missing values in the data set by using KNN imputation, the integrated model could achieve a satisfactory accuracy. Hence, we speculate that applying this methodology to the practical diagnosis of CKD would achieve a desirable effect. In addition, this methodology might be applicable to the clinical data of the other diseases in actual medical diagnosis. However, in the process of establishing the model, due to the limitations of the conditions, the available data samples are relatively small, including only 400 samples. Therefore, the generalization performance of the model might be limited. In addition, due to there are only two categories (ckd and notckd) of data samples in the data set, the model can not diagnose the severity of CKD. In the future, a large number of more complex and representative data will be collected to train the model to improve the generalization performance while enabling it to detect the severity of the disease. We believe that this model will be more and more perfect by the increase of size and quality of the data.

HOME

CHAPTER 8

SOFTWARE TESTING

8.1 GENERAL

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

8.2 DEVELOPING METHODOLOGIES

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used. The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

8.3 Types of Tests

8.3.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system

configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

8.3.2 Functional test

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

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.
- Systems/Procedures : interfacing systems or procedures must be invoked.

8.3.3 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

8.3.4 Performance Test

The Performance test ensures that the output be produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

8.3.5 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

8.3.6 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Acceptance testing for Data Synchronization:

- The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
- The Route add operation is done only when there is a Route request in need
- The Status of Nodes information is done automatically in the Cache Updation process

8.3.7 Build the test plan

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identify the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

8.4 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

8.5 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

8.6 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER 9

APPLICATION

9.1 GENERAL

Our results show the feasibility of the proposed methodology. By the use of KNN imputation, LOG, RF, SVM and FNN could achieve better performance than the cases when the random imputation and mean and mode imputation were used. KNN imputation could fill in the missing values in the data set for the cases wherein the diagnostic categories are unknown, which is closer to the real-life medical situation. Through the misjudgments analysis, LOG and RF were selected as the component models. The LOG achieved an accuracy of around 98.75%, which indicates most samples in the data set are linearly separable. The RF achieved better performance compared with the LOG with the accuracy was around 99.75%. From Tables 7 and 8, the misjudgments produced by LOG and RF are different in almost all cases, and the corresponding calculation speeds are relatively fast. Therefore, an integrated model combining LOG and RF was established to improve the performance of the component models. From the simulation result, the method of integrating several different classifiers is feasible and effective.

9.2 FUTURE ENHANCEMENT

In the future, a large number of more complex and representative data will be collected to train the model to improve the generalization performance while enabling it to detect the severity of the disease. We believe that this model will be more and more perfect by the increase of size and quality of the data.

CHAPTER 10

CONCLUSION & REFERENCE

10.1 CONCLUSION

The proposed CKD diagnostic methodology is feasible in terms of data imputation and samples diagnosis. After unsupervised imputation of missing values in the data set by using KNN imputation, the integrated model could achieve a satisfactory accuracy. Hence, we speculate that applying this methodology to the practical diagnosis of CKD would achieve a desirable effect. In addition, this methodology might be applicable to the clinical data of the other diseases in actual medical diagnosis. However, in the process of establishing the model, due to the limitations of the conditions, the available data samples are relatively small, including only 400 samples. Therefore, the generalization performance of the model might be limited. In addition, due to there are only two categories (ckd and notckd) of data samples in the data set, the model can not diagnose the severity of CKD.

10.2 REFERENCE

- [1] Z. Chen, Z. Zhang, R. Zhu, Y. Xiang, and P. B. Harrington, "Diagnosis of patients with chronic kidney disease by using two fuzzy classifiers," *Chemometrics Intell. Lab. Syst.*, vol. 153, pp. 140145, Apr. 2016.
- [2] A. Subasi, E. Alickovic, and J. Kevric, "Diagnosis of chronic kidney disease by using random forest," in *Proc. Int. Conf. Med. Biol. Eng.*, Mar. 2017, pp. 589594.
- [3] L. Zhang, "Prevalence of chronic kidney disease in China: A crosssectional survey," *Lancet*, vol. 379, pp. 815822, Mar. 2012.
- [4] A. Singh, G. Nadkarni, O. Gottesman, S. B. Ellis, E. P. Bottinger, and J. V. Guttag, "Incorporating temporal EHR data in predictive models for risk stratification of renal function deterioration," *J. Biomed. Informat.*, vol. 53, pp. 220228, Feb. 2015.
- [5] A. M. Cueto-Manzano, L. Cortés-Sanabria, H. R. Martínez-Ramírez, E. Rojas-Campos, B. Gómez-Navarro, and M. Castellero-Manzano, "Prevalence of chronic kidney disease in an adult population," *Arch. Med. Res.*, vol. 45, no. 6, pp. 507513, Aug. 2014.
- [6] H. Polat, H. D. Mehr, and A. Cetin, "Diagnosis of chronic kidney disease based on support vector machine by feature selection methods," *J. Med. Syst.*, vol. 41, no. 4, p. 55, Apr. 2017.
- [7] C. Barbieri, F. Mari, A. Stopper, E. Gatti, P. Escandell-Montero, J. M. Martínez-Martínez, and J. D. Martín-Guerrero, "A new machine learning approach for predicting the response to anemia treatment in a large cohort of end stage renal disease patients undergoing dialysis," *Comput. Biol. Med.*, vol. 61, pp. 5661, Jun. 2015.
- [8] V. Papademetriou, E. S. Nylen, M. Doumas, J. Probsteld, J. F. Mann, R. E. Gilbert, and H. C. Gerstein, "Chronic kidney disease, basal insulin glargine, and health outcomes in people with dysglycemia: The ORIGIN Study," *Amer. J. Med.*, vol. 130, no. 12, pp. 1465.e271465.e39, Dec. 2017.
- [9] N. R. Hill, "Global prevalence of chronic kidney diseaseA systematic review and meta-analysis," *PLoS ONE*, vol. 11, no. 7, Jul. 2016, Art. no. e0158765.
- [10] M. M. Hossain, R. K. Detwiler, E. H. Chang, M. C. Caughey, M.W. Fisher, T. C. Nichols, E. P. Merricks, R. A. Raymer, M. Whitford, D. A. Bellinger, L. E. Wimsey, and C. M. Gallippi, "Mechanical anisotropy assessment in kidney cortex using ARFI peak displacement: Preclinical validation and pilot in vivo clinical results in kidney allografts," *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 66, no. 3, pp. 551562, Mar. 2019.

- [11] M. Alloghani, D. Al-Jumeily, T. Baker, A. Hussain, J. Mustana, and A. J. Aljaaf, "Applications of machine learning techniques for software engineering learning and early prediction of students' performance," in Proc. Int. Conf. Soft Comput. Data Sci., Dec. 2018, pp. 246258.
- [12] D. Gupta, S. Khare, and A. Aggarwal, "A method to predict diagnostic codes for chronic diseases using machine learning techniques," in Proc. Int. Conf. Comput., Commun. Autom. (ICCCA), Apr. 2016, pp. 281287.
- [13] L. Du, C. Xia, Z. Deng, G. Lu, S. Xia, and J. Ma, "A machine learning based approach to identify protected health information in Chinese clinical text," Int. J. Med. Informat., vol. 116, pp. 2432, Aug. 2018.
- [14] R. Abbas, A. J. Hussain, D. Al-Jumeily, T. Baker, and A. Khattak, "Classification of foetal distress and hypoxia using machine learning approaches," in Proc. Int. Conf. Intell. Comput., Jul. 2018, pp. 767776.
- [15] M. Mahyoub, M. Randles, T. Baker, and P. Yang, "Comparison analysis of machine learning algorithms to rank alzheimer's disease risk factors by importance," in Proc. 11th Int. Conf. Develop. eSyst. Eng. (DeSE), Sep. 2018, pp. 111.
- [16] E. Alickovic and A. Subasi, "Medical decision support system for diagnosis of heart arrhythmia using DWT and random forests classifier," J. Med. Syst., vol. 40, no. 4, Apr. 2016.
- [17] Z. Masetic and A. Subasi, "Congestive heart failure detection using random forest classifier," Comput. Methods Programs Biomed., vol. 130, pp. 5464, Jul. 2016.
- [18] Q. Zou, "Predicting diabetes mellitus with machine learning techniques," Frontiers Genet., vol. 9, p. 515, Nov. 2018.
- [19] Z. Gao, J. Li, J. Guo, Y. Chen, Z. Yi, and J. Zhong, "Diagnosis of diabetic retinopathy using deep neural networks," IEEE Access, vol. 7, pp. 33603370, 2019.
- [20] R. J. Kate, R. M. Perez, D. Mazumdar, K. S. Pasupathy, and V. Nilakantan, "Prediction and detection models for acute kidney injury in hospitalized older adults," BMC Med. Inform. Decis. Making, vol. 16, p. 39, Mar. 2016.
- [21] N. Park, E. Kang, M. Park, H. Lee, H.-G. Kang, H.-J. Yoon, and U. Kang, "Predicting acute kidney injury in cancer patients using heterogeneous and irregular data," PLoS ONE, vol. 13, no. 7, Jul. 2018, Art. no. e0199839.

- [22] M. Patricio, "Using Resistin, glucose, age and BMI to predict the presence of breast cancer," *BMC Cancer*, vol. 18, p. 29, Jan. 2018.
- [23] X. Wang, Z. Wang, J. Weng, C. Wen, H. Chen, and X. Wang, "A new effective machine learning framework for sepsis diagnosis," *IEEE Access*, vol. 6, pp. 4830048310, 2018.
- [24] Y. Chen, Y. Luo, W. Huang, D. Hu, R.-Q. Zheng, S.-Z. Cong, F.-K. Meng, H. Yang, H.-J. Lin, Y. Sun, X.-Y. Wang, T. Wu, J. Ren, S.-F. Pei, Y. Zheng, Y. He, Y. Hu, N. Yang, and H. Yan, "Machine-learning-based classification of real-time tissue elastography for hepatic fibrosis in patients with chronic hepatitis B," *Comput. Biol. Med.*, vol. 89, pp. 1823, Oct. 2017.
- [25] E. Hodneland, E. Keilegavlen, E. A. Hanson, E. Andersen, J. A. Monssen, J. Rorvik, S. Leh, H.-P. Marti, A. Lundervold, E. Svarstad, and J. M. Nordbotten, "In Vivo detection of chronic kidney disease using tissue deformation fields from dynamic MR imaging," *IEEE Trans. Biomed. Eng.*, vol. 66, no. 6, pp. 17791790, Jun. 2019.
- [26] G. R. Vasquez-Morales, S. M. Martinez-Monterrubio, P. Moreno-Ger, and J. A. Recio-Garcia, "Explainable prediction of chronic renal disease in the colombian population using neural networks and case-based reasoning," *IEEE Access*, vol. 7, pp. 152900152910, 2019.
- [27] Z. Chen, X. Zhang, and Z. Zhang, "Clinical risk assessment of patients with chronic kidney disease by using clinical data and multivariate models," *Int. Urol. Nephrol.*, vol. 48, no. 12, pp. 20692075, Dec. 2016.
- [28] A. J. Aljaaf, D. Al-Jumeily, H. M. Haglan, M. Alloghani, T. Baker, A. J. Hussain, and J. Mustana, "Early prediction of chronic kidney disease using machine learning supported by predictive analytics," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2018, pp. 19.
- [29] B. Boukenze, A. Haqiq, and H. Mousannif, "Predicting chronic kidney failure disease using data mining techniques," in *Proc. Int. Symp. Ubiquitous Netw.*, Nov. 2016, pp. 701712.
- [30] N. A. Almansour, H. F. Syed, N. R. Khayat, R. K. Altheeb, R. E. Juri, J. Alhiya, S. Alrashed, and S. O. Olatunji, "Neural network and support vector machine for the prediction of chronic kidney disease: A comparative study," *Comput. Biol. Med.*, vol. 109, pp. 101111, Jun. 2019.
- [31] W. Gunarathne, K. Perera, and K. Kahandawaarachchi, "Performance evaluation on machine learning classification techniques for disease classification and forecasting through data analytics for chronic kidney disease (CKD)," in *Proc. IEEE 17th Int. Conf. Bioinf. Bioeng. (BIBE)*, Oct. 2017, pp. 291296.

- [32] D. Dua and C. Graff, ``UCI machine learning repository," Ph.D. dissertation, School Inf. Comput. Sci., Univ. California, Oakland, CA, USA, 2017. [Online]. Available: <http://archive.ics.uci.edu/ml>.
- [33] D. Ichikawa, T. Saito, W. Ujita, and H. Oyama, ``How can machine learning methods assist in virtual screening for hyperuricemia? A healthcare machine-learning approach," J. Biomed. Informat., vol. 64, pp. 2024, Dec. 2016.
- [34] L. N. Sanchez-Pinto, L. R. Venable, J. Fahrenbach, and M. M. Churpek, ``Comparison of variable selection methods for clinical predictive modeling," Int. J. Med. Informant., vol. 116, pp. 1017, Aug. 2018.
- [35] G. Feng, G.-B. Huang, Q. Lin, and R. Gay, ``Error minimized extreme learning machine with growth of hidden nodes and incremental learning," IEEE Trans. Neural Netw., vol. 20, no. 8, pp. 13521357, Aug. 2009.
- [36] G. Ciaburro and B. Venkateswaran, Neural Networks With R. Beijing, China: China Machine Press, 2018, pp. 93106.
- [37] M. Hummel, D. Edelmann, and A. Kopp-Schneider, ``Clustering of samples and variables with mixed-type data," PLoS ONE, vol. 12, no. 11, Nov. 2017, Art. no. e0188274.