

HEPATITIS C VIRUS PREDICTION AND ANALYSIS USING MACHINE LEARNING ALGORITHMS

Submitted in partial fulfillment of the requirements for the award of
Master of Science in Bioinformatics and Data Science

By

G.HARSHINI (42739013)



DEPARTMENT OF BIOINFORMATICS

SCHOOL OF BIO AND CHEMICAL ENGINEERING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC | 12B Status by UGC | Approved by AICTE

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600 119

APRIL – 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with “A” grade by NAAC | 12B Status by UGC | Approved by AICTE

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600119

www.sathyabama.ac.in

DEPARTMENT OF BIOINFORMATICS

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **G.Harshini (42739013)**, who carried out the final year project entitled “**HEPATITIS C VIRUS PREDICTION AND ANALYSIS OF MACHINE LEARNING ALGORITHMS**” under our supervision from January 2024 to April 2024.

Internal Guide

External Guide

Dr. DANIEL ALEX ANAND

Dr. DANIEL ALEX ANAND

Dr .H. JEMMY CHRISTY

Head of the Department

Submitted for Viva Voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **G.Harshini**, hereby declare that the Project Report entitled **Hepatitis C virus prediction and analysis of machine learning algorithms** done by me under the guidance of **Dr. DANIEL ALEX ANAND** at Sathyabama Institute of Science and Technology, Jeppiaar Nagar, Rajiv Gandhi Road, Chennai-600119, Tamil Nadu, India, is submitted in partial fulfillment of the requirements for the award of **Master of Science in Bioinformatics and Data Science**.

DATE:

PLACE:

SIGNATURE OF CANDIDATE

ACKNOWLEDGEMENTS

I am delighted to express my heartfelt acknowledgement to **the Board of Management of SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement and support throughout the duration of my project. Their guidance has been invaluable in ensuring the successful completion of this project, and I am sincerely grateful for their contribution.

I extend my sincere gratitude to **DR. REKHA CHAKRAVATHY**, dean of the school of **SCIENCE AND HUMANITIES**, and I am deeply grateful for her mentorship and leadership which have been a constant source of inspiration her dedication to fostering a conducive academic environment has been instrumental in the completion of this thesis

I sincerely thank **Dr. H. Jemmy Christy**, Head of the Department, Dept. of Bioinformatics, for his unwavering support and timely provision of necessary details during the progressive reviews. His support has been instrumental in helping me to overcome obstacles and achieve success in this project.

I am deeply indebted to my Project Guide, **Dr. Daniel Alex Anand**, for his invaluable guidance, suggestions, and constant encouragement throughout the project. His expertise and insights have been instrumental in shaping my work and ensuring its successful completion.

I also wish to extend my gratitude to my family for their support and assistance in various ways, which have contributed to the successful completion of this project.

ABSTRACT

This study presents significant advancements in leveraging machine learning for risk stratification and treatment optimization in Hepatitis C patients. Employing a diverse set of patient data, including clinical, demographic, and molecular information, our machine learning models demonstrate enhanced accuracy in predicting disease progression and treatment responses. Through comprehensive analysis, we identify key risk factors and biomarkers contributing to Hepatitis C severity. The developed models not only enable precise risk stratification but also facilitate personalized treatment recommendations, thereby optimizing therapeutic interventions. The integration of advanced algorithms enhances the predictive power of our models, fostering a more nuanced understanding of individual patient trajectories. This research contributes to the evolving landscape of precision medicine in Hepatitis C care, offering a promising avenue for tailoring interventions based on patient-specific characteristics. The findings underscore the potential of machine learning in refining risk assessment and treatment strategies, ultimately contributing to improved clinical outcomes in Hepatitis C management.

Keywords: Hepatitis C,disease,biomarkers,medicine,patient.

TABLE OF CONTENT

SL.NO	TITLE	PAGE.NO
	ABSTRACT	
	FIGURES	
1	INTRODUCTION	
1.1	HEPATITS C VIRUS	1
1.2	DATA SCIENCE	1
1.3	ARTIFICIAL INTELLIGENCE	2
1.3.1	LEARNING PRECESSES	4
1.3.2	REASONING PRECESSES	4
1.3.3	SELF-CORRECTION PRECESSES	4
2	LITERATURE REVIEW	
2.1	GENERAL	5
2.2	REVIEW OF THE LITERATURE SURVEY	5
3	AIM AND SCOPE	
3.1	AIM	8
3.2	SCOPE	8
4	MATERIALS AND METHODS USED	
4.1	PREPARING THE DATASET	9
4.1.1	DATA COLLECTION	9
4.1.2	DATA CLEANING	9
4.1.3	FEATURE ENGINEERING	9
4.1.4	DATA ENCODING	9
4.1.5	HANDLING IMBALANCE	9
4.1.6	DATA SPLITTING	10
4.2	NORMALIZATION	10
4.2.1	DATA PRIVACY AND SECURITY	10
4.2.2	DOCUMENTATION	10
4.3	PROPOSED SYSTEM	11
4.4	LIST OF MODULES	11
4.5	ENVIRONMENTAL REQUIREMENTS	12
4.6	PYTHON	16
4.7	DESIGN ARCHITECTURE	17
4.8	CLASS DIAGRAM	20
4.9	MODULE DISCRPTION	21
4.9.1	DATA PROCESSING	21
4.9.2	DATA VISUALIZATON	23
4.9.3	ALGORITHM IMPLEMENTATION	24

4.10	PERFORMANCE METRICS TO CALCULATE	25
4.11	KNN CLASSIFIER	27
4.12	CAT BOOST CLASSIFIER	28
4.13	RANDOM FOREST CLASSIFIER	29
4.14	DEPLOYMENT	29
4.15	HTML INTRODUCTION	32
4.16	CSS	36
4.17	DEPLOYING THE MODEL PREDICTING OUTPUT	41
5	RESULT AND DISCUSSION	
5.1	DATA COLLECTION	42
5.2	PREPROCESSING	42
5.3	DATA VISUALIZATION	42
5.4	FEATURE SELECTON	44
5.5	MODEL TRAINING	45
6	SUMMARY AND CONCLUSION	
6.1	SUMMARY	47
6.2	CONCLUSION	47
6.3	FUTURE SCOPE	48
7	REFERENCES	50
8	APPENDIX	53

LIST OF FIGURES

SL.NO	TITLE	PAGE.NO
01	SYSTEM ARCHITECTURE	19
02	WORKFLOW DIAGRAM	21
03	USECASE DIAGRAM	22
04	CLASS DIAGRAM	23
05	ACTIVITY DIAGRAM	46
06	SEQUENCE DIAGRAM	47
07	ER – DIAGRAM	48
08	MODULE DIAGRAM	49

CHAPTER 1

INTRODUCTION

1.1 HEPATITIS C VIRUS

In the dynamic landscape of Hepatitis C management, the integration of machine learning heralds a transformative era marked by personalized risk stratification and optimized treatment. As individuals afflicted with Hepatitis C present a diverse array of symptoms, ranging from mild fatigue to advanced liver disease, the need for tailored approaches is more pressing than ever. Machine learning, fuelled by its ability to process and decipher vast datasets, stands at the forefront of this healthcare revolution, analysing not only clinical manifestations but also genomic and demographic nuances to unravel intricate patterns in disease progression.(salama et al.,2019)

With symptoms serving as crucial indicators of both disease severity and treatment response, the application of machine learning techniques holds immense promise.(chang et al.,2010) By harnessing the power of artificial intelligence, healthcare professionals can not only predict the course of Hepatitis C with greater accuracy but also customize treatment plans that address the unique symptomatology and individual characteristics of patients. This symbiotic relationship between machine learning and symptom-based insights not only enhances diagnostic precision but also opens avenues for proactive interventions, ensuring that Hepatitis C patients receive targeted, timely, and optimized care based on their distinct symptom profiles(WAH,K.K et al.,2020)

1.2 DATA SCIENCE

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured data, and apply knowledge and actionable insights from data across a broad range of application domains.(salama et al.,2019)

The term "data science" has been traced back to 1974, when Peter Naur proposed it as an alternative name for computer science. In 1996, the International Federation of Classification Societies became the first conference to

specifically feature data science as a topic. However, the definition was still in flux.(gould et al.,2021)

The term “data science” was first coined in 2008 by D.J. Patil, and Jeff Hammerbacher, the pioneer leads of data and analytics efforts at LinkedIn and Facebook. In less than a decade, it has become one of the hottest and most trending professions in the market.(hassan et.al.,2012)

Data science is the field of study that combines domain expertise, programming skills, and knowledge of mathematics and statistics to extract meaningful insights from data.(Cornejo et al.,2012)

Data science can be defined as a blend of mathematics, business acumen, tools, algorithms and machine learning techniques, all of which help us in finding out the hidden insights or patterns from raw data which can be of major use in the formation of big business decisions.(bashhar et.al.,2017)

1.3 ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving. Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to the natural intelligence displayed by humans or animals.(deo et al.,2015) Leading AI textbooks define the field as the study of "intelligent agents" any system that perceives its environment and takes actions that maximize its chance of achieving its goals. Some popular accounts use the term "artificial intelligence" to describe machines that mimic "cognitive" functions that humans associate with the human mind, such as "learning" and "problem solving", however this definition is rejected by major AI researchers. Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, and speech recognition and machine vision.(khan et al.,2021)

AI applications include advanced web search engines, recommendation systems (used by Youtube, Amazon and Netflix), Understanding human

speech (such as Siri or Alexa), self-driving cars (e.g. Tesla), and competing at the highest level in strategic game systems (such as chess and Go), As machines become increasingly capable, tasks considered to require "intelligence" are often removed from the definition of AI, a phenomenon known as the AI effect. (sudars et al.,2020)

For instance, optical character recognition is frequently excluded from things considered to be AI, having become a routine technology. Artificial intelligence was founded as an academic discipline in 1956, and in the years since has experienced several waves of optimism, followed by disappointment and the loss of funding (known as an "AI winter"), followed by new approaches, success and renewed funding. AI research has tried and discarded many different approaches during its lifetime, including simulating the brain, modeling human problem solving, formal logic, large databases of knowledge and imitating animal behavior. In the first decades of the 21st century, highly mathematical statistical machine learning has dominated the field, and this technique has proved highly successful, helping to solve many challenging problems throughout industry and academia.(ozola et al.,2020) The various sub-fields of AI research are centered on particular goals and the use of particular tools. The traditional goals of AI research include reasoning, knowledge representation, planning, learning, natural language processing, perception and the ability to move and manipulate objects. General intelligence (the ability to solve an arbitrary problem) is among the field's long-term goals. To solve these problems, AI researchers use versions of search and mathematical optimization, formal logic, artificial neural networks, and methods based on statistics, probability and economics. AI also draws upon computer science, psychology, linguistics, philosophy, and many other fields.(dua et al.,2017)

The field was founded on the assumption that human intelligence "can be so precisely described that a machine can be made to simulate it". This raises philosophical arguments about the mind and the ethics of creating artificial beings endowed with human-like intelligence. (salam et al.,2019)These issues have been explored by myth, fiction and philosophy since antiquity. Science fiction and futurology have also suggested that, with its enormous potential and power, AI may become an existential risk to humanity. As the hype around AI has accelerated,

vendors have been scrambling to promote how their products and services use AI. Often what they refer to as AI is simply one component of AI, such as machine learning. AI requires a foundation of specialized hardware and software for writing and training machine learning algorithms. No one programming language is synonymous with AI, but a few, including Python, R and Java, are popular.(bowyer et al.,2002)

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

AI programming focuses on three cognitive skills: learning, reasoning and self-correction.(dong et al.,2021)

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

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

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

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

intelligence technologies are quickly evolving, primarily because AI processes large amounts of data much faster and makes predictions more accurately than humanly possible.(Sivakumar et al.2021)

CHAPTER 2

LITERATURE REVIEW

2.1 GENERAL

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

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

2.2 REVIEW OF LITERATURE SURVEY

Hepatitis C is a liver disease caused by infection with the hepatitis C virus (HCV), which is transmitted through the blood. The disease can lead to diseases ranging from a mild form to serious lifelong illness. Studies to detect the disease early and reduce its effect are continuing. This study proposes an effective support vector machine model supported by principal component analysis for detecting hepatitis c disease. The dataset consisted of twelve independent variables, each containing 582 samples, and these variables were used as inputs to the two classifiers, support vector machine (SVM) and artificial neural network (ANN). The accuracy, sensitivity, specificity, Matthew's correlation coefficient (MCC) and KAPPA were calculated using two classification models. In addition, performance comparisons of classifiers were made for the two cases with and without principal component

analysis (PCA) applied to the inputs. The highest accuracy (98.7%), sensitivity (99.1%), specificity (95.2%), MCC (92.3%) and Kappa (92.3%) in the binary class label were obtained with the SVM with PCA. In the four-class label, the highest accuracy was achieved with the same model with 95.7%. The results show that an SVM classifier model, in which PCA-reduced independent variables are applied to its inputs, may be a candidate for an accurate prediction model to predict hepatitis C disease.

Keywords: Artificial neural network; Dimensional reduction; Hepatitis C; Principal component analysis; Support vector machine(Gundogdu et.al.,2022)

Hepatitis C is a significant public health concern, resulting in substantial morbidity and mortality worldwide. Early diagnosis and effective treatment are essential to prevent the disease's progression to chronic liver disease. Machine learning algorithms have been increasingly used to develop predictive models for various diseases, including hepatitis C.(Nabot et.al.,2023) The study aims to evaluate the performance of several machine learning algorithms in diagnosing chronic liver disease, with a specific focus on hepatitis C, to improve the cost-effectiveness and efficiency of the diagnostic process. We collected a comprehensive dataset of 1801 patient records, each with 12 distinct features, from Jordan University Hospital. To assess the robustness and dependability of our proposed framework, we conducted two research scenarios, one with feature selection and one without. We also employed the Sequential Forward Selection (SFS) method to identify the most relevant features that can enhance the model's accuracy. Moreover, we investigated the effect of the synthetic minority oversampling technique (SMOTE) on the accuracy of the model's predictions. Our findings indicate that all machine learning models achieved an average accuracy of 83% when applied to the dataset. Furthermore, the use of SMOTE did not significantly affect the accuracy of the model's predictions. Despite the increasing use of machine learning models in medical diagnosis, there is a growing concern about their interpretability. As such, we addressed this issue by utilizing the Shapley Additive Explanations (SHAP) method to explain the predictions of our machine learning model, which was specifically developed for hepatitis C prediction in Jordan. This work provides a comprehensive evaluation of various machine learning algorithms in diagnosing chronic liver disease, with a particular emphasis on hepatitis C. The results provide

valuable insights into the cost-effectiveness and efficiency of the diagnostic process and highlight the importance of interpretability in medical diagnosis.

Keywords: hepatitis C; data augmentation; feature selection; classification algorithms; machine learning; SHA

This prognostic study included 48 151 patients with hepatitis C virus (HCV)–related cirrhosis in the national Veterans Health Administration who had at least 3 years of follow-up after the diagnosis of cirrhosis. Patients were identified by having at least 1 positive HCV RNA test between January 1, 2000, to January 1, 2016, and were followed up from the diagnosis of cirrhosis to January 1, 2019, for the development of incident HCC. A total of 3 models predicting HCC during a 3-year period were developed and compared, as follows: (1) logistic regression (LR) with cross-sectional inputs (cross-sectional LR); (2) LR with longitudinal inputs (longitudinal LR); and (3) RNN with longitudinal inputs. Data analysis was conducted from April 2018 to August 2020 (Tang et.al.,2020)

CHAPTER 3

AIM AND SCOPE

3.1 AIM

The aim of this study is to harness the power of machine learning in advancing the precision and efficacy of Hepatitis C management through refined risk stratification and treatment optimization strategies. By leveraging comprehensive datasets encompassing clinical, genomic, and demographic information, our objective is to develop predictive models capable of identifying nuanced patterns associated with disease progression, symptomatology, and treatment responses. Through these advancements, we aspire to pave the way for personalized healthcare interventions, tailoring Hepatitis C treatment plans to individual patient profiles, ultimately improving outcomes, minimizing adverse effects, and contributing to the evolution of patient-centric approaches in the realm of infectious disease management.

3.2 SCOPE

The scope of this project encompasses the comprehensive integration of machine learning methodologies into the domain of Hepatitis C management, with a focus on risk stratification and treatment optimization. By leveraging advanced analytics on diverse datasets, ranging from clinical records to genomic information, the project aims to develop models that can discern intricate patterns associated with disease progression and treatment outcomes. The scope extends to the creation of predictive algorithms capable of identifying high-risk individuals and tailoring treatment plans to individual patient profiles, contributing to the paradigm shift towards personalized medicine in Hepatitis C care. Additionally, the project aims to offer insights into the scalability and generalizability of the machine learning models, fostering their potential adoption in broader healthcare contexts and underscoring the significance of data-driven approaches in infectious disease management. The ultimate aspiration is to enhance the precision, efficacy, and patient-centric nature of Hepatitis C interventions, thereby contributing to advancements in the broader field of infectious disease research and healthcare practices.

CHAPTER 4

MATERIALS AND METHODS USED

4.1 PREPARING THE DATASET:

Dataset Link: <https://archive.ics.uci.edu/dataset/571/hcv+data>

Preparing a dataset for machine learning in the context of Hepatitis C involves several crucial steps to ensure the accuracy and effectiveness of the models. Here is a concise guide:

4.1.1 Data Collection:

Gather diverse and comprehensive datasets encompassing clinical records, laboratory results, genomic data, and patient demographics.

Include information on symptoms, disease progression, treatment history, and relevant outcomes.

4.1.2 Data Cleaning:

Identify and handle missing values, outliers, and inaccuracies in the dataset.

Standardize and normalize numerical features to ensure consistency.

4.1.3 Feature Engineering:

Select relevant features that contribute to risk stratification and treatment optimization, such as viral load, liver enzyme levels, and genetic markers.

Create new features if needed, like aggregating symptom severity scores or deriving composite indicators.

4.1.4 Data Encoding:

Convert categorical variables into numerical representations using techniques like one-hot encoding or label encoding.

Ensure uniformity in data types to facilitate machine learning algorithms.

4.1.5 Handling Imbalances:

Address imbalances in the dataset, especially if there are significantly more instances of one outcome (e.g., more patients with a specific symptom severity).

Consider techniques like oversampling, under sampling, or using synthetic data.

4.1.6 Data Splitting:

Divide the dataset into training, validation, and test sets to evaluate model performance accurately.

Ensure that each set is representative of the overall dataset in terms of the distribution of features and outcomes.

4.2 NORMALIZATION:

Normalize numerical values to a standard scale to prevent any single feature from dominating the model due to its scale.

Handling Time-series Data (if applicable):

If the dataset involves temporal information, consider time-based splitting and feature engineering.

Ensure that the order of data points is preserved to capture temporal dependencies.

4.2.1 Data Privacy and Security:

Implement necessary measures to protect patient privacy and adhere to data security regulations.

4.2.2 Documentation:

Maintain detailed documentation about the dataset, including its source, preprocessing steps, and any modifications made.

By meticulously preparing the dataset, you lay the foundation for robust machine learning models that can contribute significantly to risk stratification and treatment optimization for Hepatitis C

4.3 PROPOSED SYSTEM:

In the proposed system, we aim to implement advanced machine learning techniques for the risk stratification and treatment optimization of Hepatitis C patients. Leveraging large-scale patient data, our system will develop predictive models to identify individuals at higher risk of disease progression and complications. By analysing a diverse range of clinical, genetic, and demographic factors, the system will enable precise risk assessment, allowing healthcare providers to tailor interventions based on individual patient profiles. The proposed system also integrates real-time data updates, ensuring continuous learning and adaptation to evolving patient conditions. Through the optimization of treatment strategies using machine learning algorithms, healthcare professionals can make informed decisions regarding antiviral therapies, personalized dosages, and treatment durations. This innovative approach not only enhances the efficacy of interventions but also minimizes potential adverse effects. Furthermore, the system emphasizes interpretability, providing healthcare practitioners with transparent insights into the decision-making process of the machine learning models. This ensures a collaborative and trustful relationship between the automated system and healthcare providers. The proposed advancements hold the potential to significantly improve patient outcomes, reduce treatment costs, and contribute to the overall efficiency of Hepatitis C management strategies.

4.4 LIST OF MODULES:

4.4.1 Data Analysis and Exploration:

Exploring the characteristics of the collected datasets.

Analysing the distribution of features and outcomes.

4.4.2 Feature Engineering:

Analysing the relevance of different features for risk stratification and treatment optimization.

Identifying patterns or relationships between features.

4.4.3 Machine Learning Model Development:

Utilizing various machine learning algorithms to analyze and learn patterns from the data.

Adjusting model parameters and configurations to optimize performance.

4.4.4 Validation and Evaluation:

Analyzing model performance on validation and test sets.

Assessing the accuracy, precision, recall, and other relevant metrics to gauge model effectiveness.

4.4.5 Interpretability and Explainability:

Analyzing the interpretability and explainability of the machine learning models.

Understanding how the models make predictions and ensuring transparency in the decision-making process.

4.4.6 Scalability and Generalizability Assessment:

Analyzing how well the models can scale to larger datasets or be generalized to different populations.

Assessing the robustness of the models in diverse healthcare scenarios.

➤ Deployment

4.5 Environmental Requirements:

1. Software Requirements:

Operating System : Windows 10 or later

Tool : Anaconda with Jupyter Notebook

2. Hardware requirements:

Processor : Intel i3

Hard disk : minimum 10 GB

RAM : minimum 4 GB

4.5.1 SOFTWARE DESCRIPTION

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

4.5.2 ANACONDA NAVIGATOR

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

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

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

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

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

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

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook
- Spyder
- PyCharm
- VSCode
- Glueviz
- Orange 3 App
- RStudio
- Anaconda Prompt (Windows only)
- Anaconda PowerShell (Windows only)

4.5.3 JUPYTER NOTEBOOK :

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

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

Notebook documents are documents produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc...). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc.) as well as executable documents which can be run to perform data analysis.

4.5.4 Working Process:

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

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

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

- It will force you to install and start the Python interpreter (at the very least).

- It will give you a bird's eye view of how to step through a small project.
- It will give you confidence, maybe to go on to your own small projects.

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

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

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

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

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

4.6 PYTHON

Introduction:

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant

indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

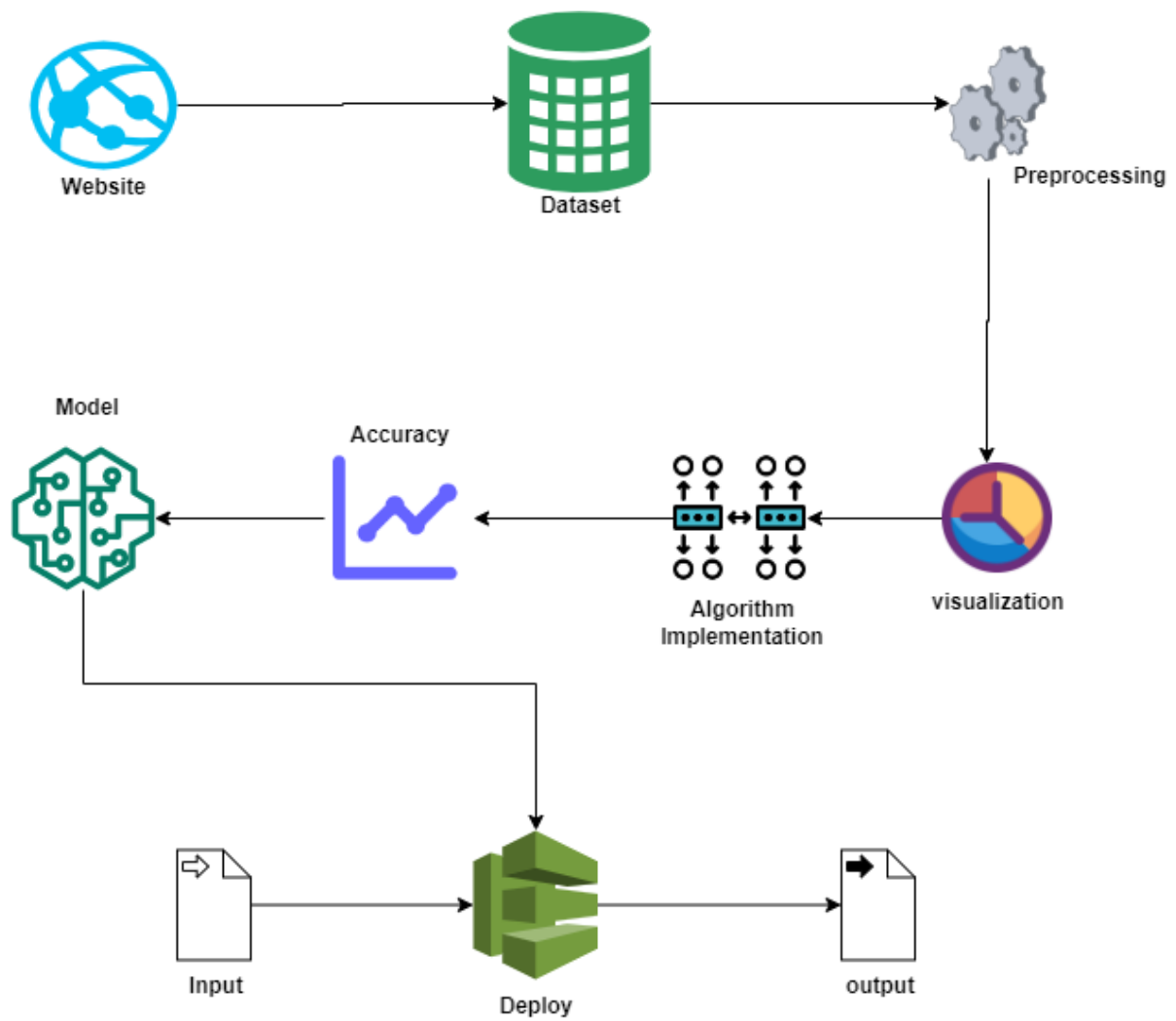
Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible. Python 2 was discontinued with version 2.7.18 in 2020.

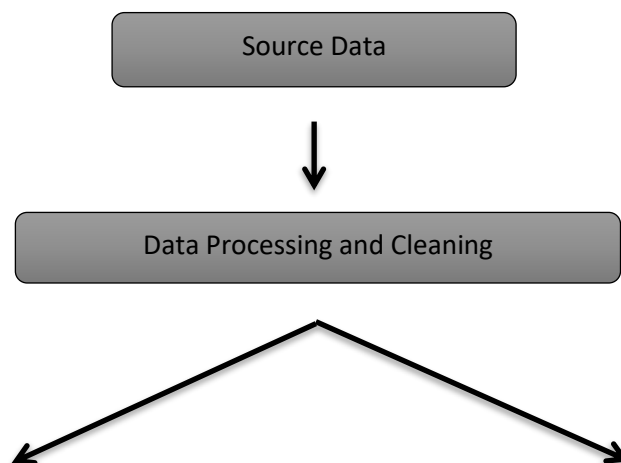
Python consistently ranks as one of the most popular programming languages

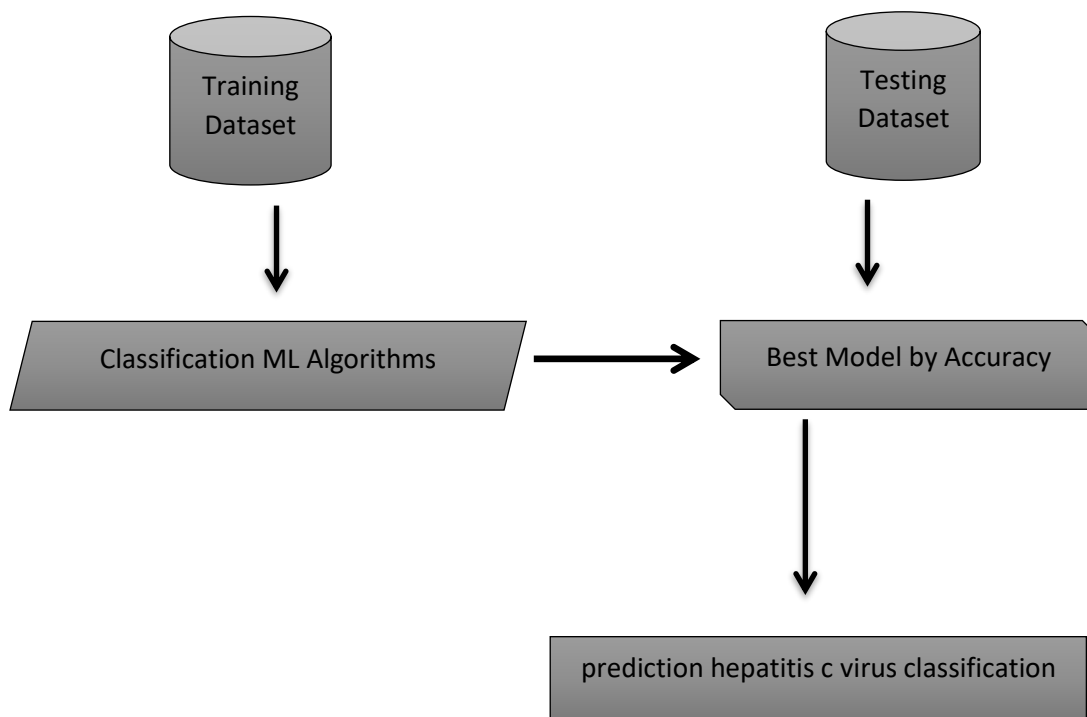
4.7 DESIGN ARCHITECTURE:

4.7.1 System Architecture:

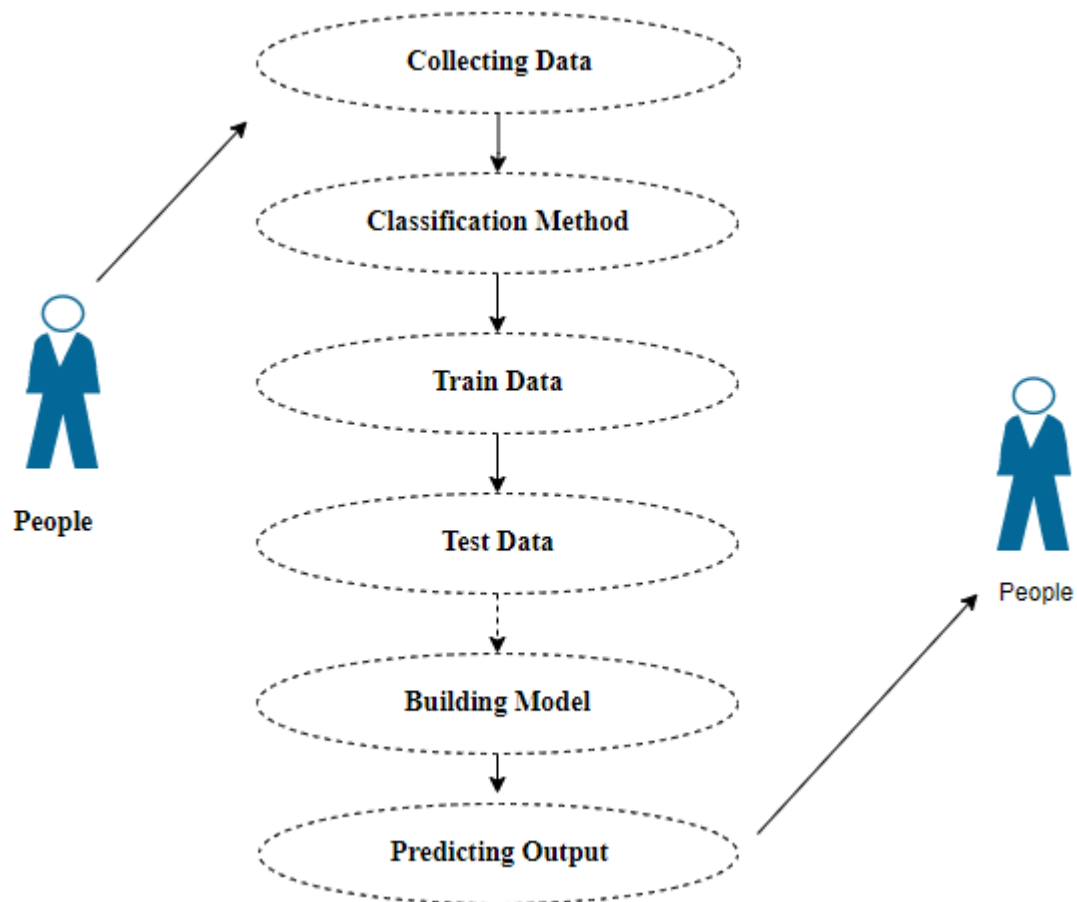


4.7.2 Work flow diagram



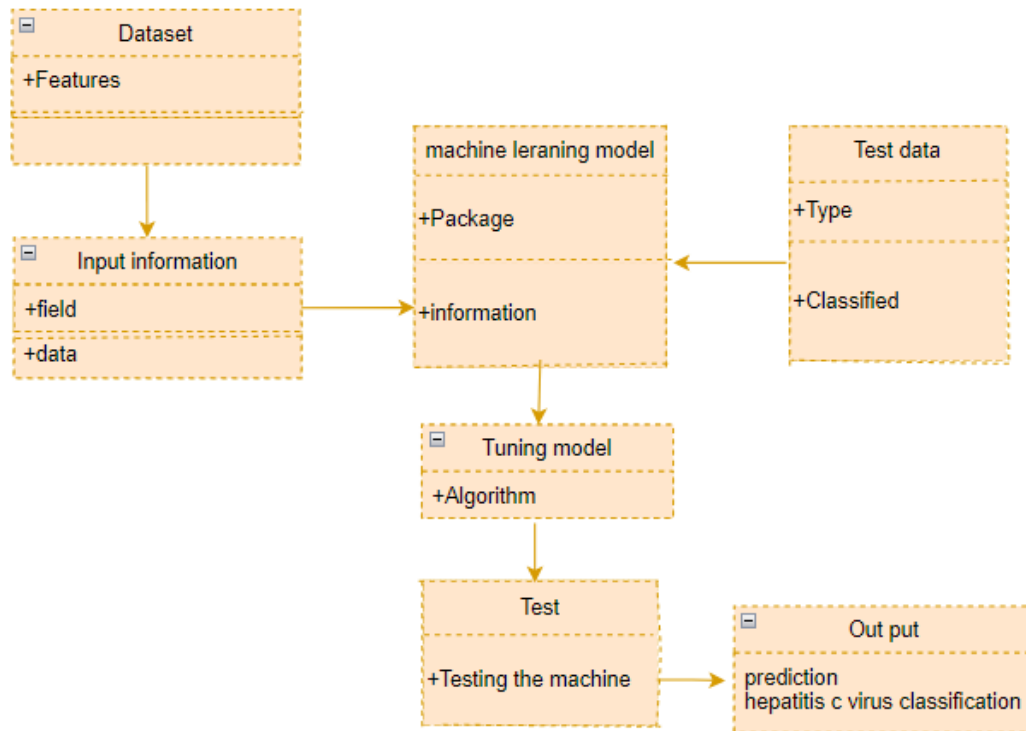


4.7.3 Use Case Diagram:



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

4.8 Class Diagram:



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

4.8.1 Entity Relationship Diagram (ERD):

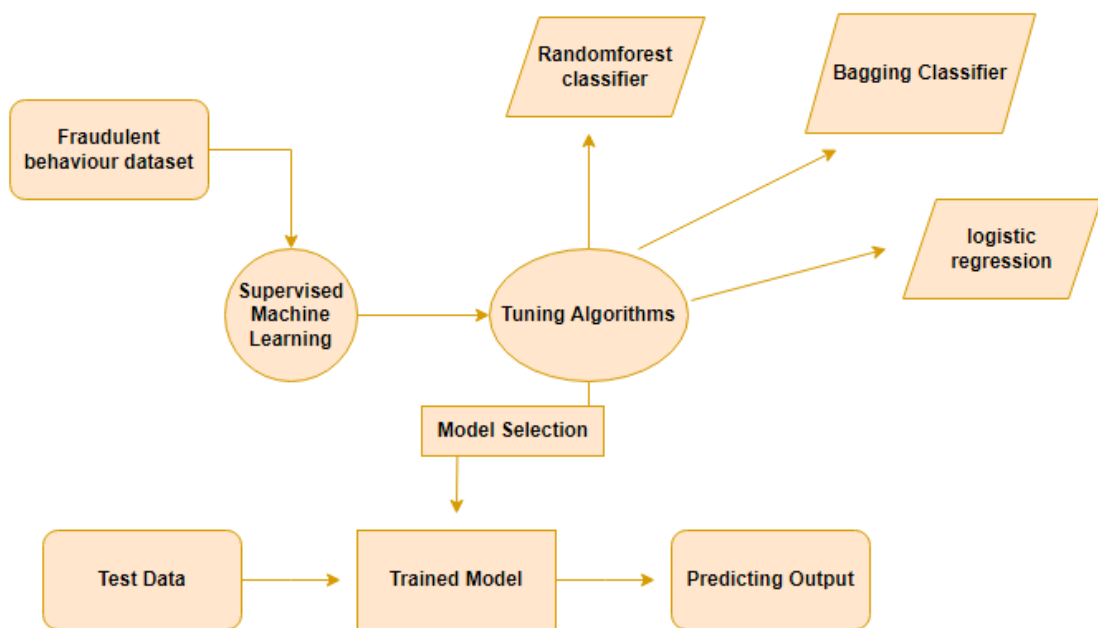


Fig 4.8.1 ERD

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

4.9 MODULE DESCRIPTION:

4.9.1 Data Pre-processing:

Validation techniques in machine learning are used to get the error rate of the Machine Learning (ML) model, which can be considered as close to the true error rate of the dataset. If the data volume is large enough to be representative of the population, you may not need the validation techniques. However, in real-world

scenarios, to work with samples of data that may not be a true representative of the population of given dataset. To finding the missing value, duplicate value and description of data type whether it is float variable or integer. The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyper parameters.

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

A number of different data cleaning tasks using Python's Pandas library and specifically, it focus on probably the biggest data cleaning task, missing values and it able to more quickly clean data. It wants to spend less time cleaning data, and more time exploring and modeling.

Some of these sources are just simple random mistakes. Other times, there can be a deeper reason why data is missing. It's important to understand these different types of missing data from a statistics point of view. The type of missing data will influence how to deal with filling in the missing values and to detect missing values, and do some basic imputation and detailed statistical approach for dealing with missing data. Before, joint into code, it's important to understand the sources of missing data. Here are some typical reasons why data is missing:

- User forgot to fill in a field.
- Data was lost while transferring manually from a legacy database.
- There was a programming error.

- Users chose not to fill out a field tied to their beliefs about how the results would be used or interpreted.

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

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

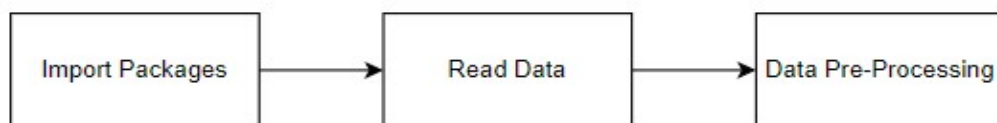


Fig 4.9.1 MODULE DIAGRAM

4.9.2 Data visualization:

Data visualization is an important skill in applied statistics and machine learning. Statistics does indeed focus on quantitative descriptions and estimations of data. Data

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

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

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

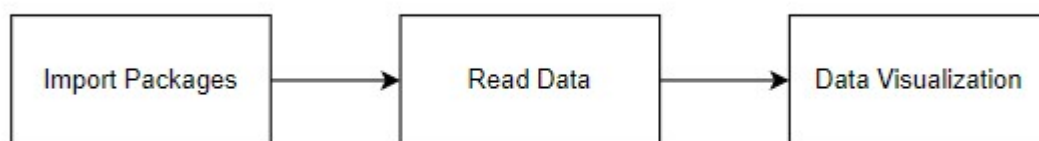


Fig 4.9.2 module diagram

4.9.3 Algorithm implementation:

It is important to compare the performance of multiple different machine learning algorithms consistently and it will discover to create a test harness to compare multiple different machine learning algorithms in Python with scikit-learn. It can use

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

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

4.10 Performance Metrics to calculate:

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

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

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

4.10.4 True Negatives (TN): A person who default predicted as payer. These are the correctly predicted negative values which means that the value of actual class is no

and value of predicted class is also no. E.g. if actual class says this passenger did not survive and predicted class tells you the same thing.

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

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

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

➤ **Accuracy calculation:**

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN)$$

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

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

$$\text{Precision} = TP / (TP + FP)$$

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

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

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

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

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

- **General Formula:**

$$\text{F-Measure} = 2\text{TP} / (2\text{TP} + \text{FP} + \text{FN})$$

- **F1-Score Formula:**

$$\text{F1 Score} = 2 * (\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$$

The below 4 different algorithms are compared:

- K- Nearest Neighbour's Classifier
- Cat Boost Classifier
- Random Forest Classifier

4.11 KNN CLASSIFIER

Ensemble learning, particularly through bagging classifier algorithms, has emerged as a powerful technique to enhance the robustness and accuracy of machine learning models. By constructing an ensemble of decision trees trained on different subsets of the data through bootstrapping, Random Forest mitigates overfitting and improves the generalization performance. The approach is especially effective in handling complex

and high-dimensional datasets, making Random Forest a versatile choice for classification tasks across various domains.

Random Forest, bagging can be applied to other base classifiers, such as Support Vector Machines (SVM) or K-Nearest Neighbors (KNN). Bagging SVM involves training multiple SVMs on different bootstrapped samples of the data and combining their predictions, providing a robust solution for both linear and non-linear classification problems. These bagging classifier algorithms showcase the effectiveness of ensemble methods in harnessing the collective wisdom of diverse models to achieve superior classification accuracy and generalization on challenging datasets.

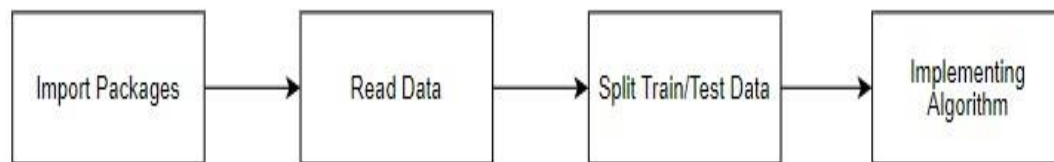


FIG 4.11 MODULE DIAGRAM

4.12 CAT BOOST CLASSIFIER:

Logistic Regression is a widely used statistical method and a fundamental algorithm in machine learning, particularly for binary classification problems. Unlike linear regression, which predicts continuous outcomes, logistic regression is specifically designed for predicting the probability of an instance belonging to a particular class.

The logistic function (sigmoid function) is applied to a linear combination of input features and their associated weights. The result of this transformation is constrained between 0 and 1, representing the probability of the instance belonging to the positive class. The logistic regression model is trained by adjusting the weights through iterative optimization methods, such as gradient descent, to minimize the difference between predicted probabilities and actual class labels.

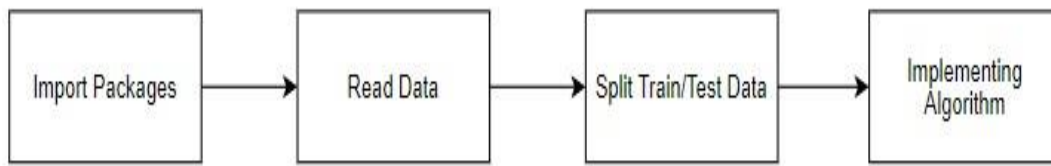


FIG 4.12 MODULE DIAGRAM

4.13 RANDOM FOREST CLASSIFIER:

Random Forest is an ensemble machine learning algorithm that combines the predictions of multiple decision trees, each created using different subsets of data and features. This approach enhances prediction accuracy, reduces overfitting, and provides valuable insights into feature importance. It's a versatile and widely used tool in machine learning for tasks such as classification and regression, particularly suited for handling complex datasets and improving overall model robustness.

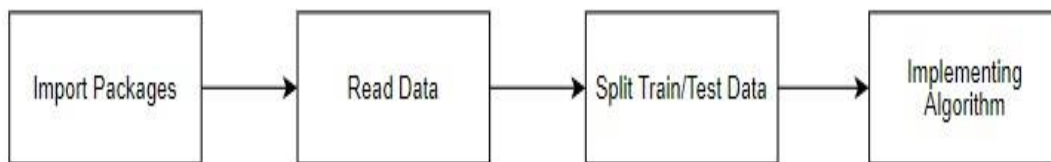


fig 4.13 module diagram

4.14 DEPLOYMENT:

4.14.1 DEPLOYING THE MODEL PREDICTING OUTPUT

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

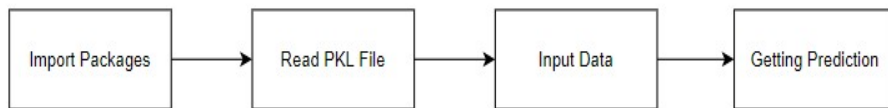


FIG 4.14.1 MODULE DIAGRAM

4.14.2 Django (Web FrameWork) :

Django is a micro web framework written in Python.

It is classified as a micro-framework because it does not require particular tools or libraries.

It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

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

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

Django was created by Armin Ronacher of Pocoo, an international group of Python enthusiasts formed in 2004. According to Ronacher, the idea was originally an April Fool's joke that was popular enough to make into a serious application. The name is a play on the earlier Bottle framework.

Django has become popular among Python enthusiasts. As of October 2020, it has second most stars on GitHub among Python web-development frameworks, only slightly behind Django, and was voted the most popular web framework in the Python Developers Survey 2018.

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

4.14.3 FEATURES:

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

Still the question remains why use Django as your web application framework if we have immensely powerful Django, Pyramid, and don't forget web mega-framework Turbo-gears? Those are supreme Python web frameworks BUT out-of-the-box Django is pretty impressive too with it's:

- Built-In Development server and Fast debugger
- integrated support for unit testing
- RESTful request dispatching
- Uses Jinja2 Templating
- support for secure cookies
- Unicode based
- Extensive Documentation
- Google App Engine Compatibility
- Extensions available to enhance features desired

4.14.4 START USING AN API

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

The `django` object implements a WSGI application and acts as the central object. It is passed the name of the module or package of the application. Once it is

created it will act as a central registry for the view functions, the URL rules, template configuration and much more.

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

4.15 HTML Introduction

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

4.15.1 Basic Construction of an HTML Page

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



Fig 4.15.1 HTML tag

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

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

`<head>` — This is where all the **metadata for the page** goes — stuff mostly meant for search engines and other computer programs.

`<body>` — This is where the **content of the page** goes.

4.15.2 FURTHER TAGS

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

`<title>`

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

<meta>

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

4.15.3 HEAD TAG

<head>

<title>My First Webpage</title>

<meta charset="UTF-8">

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

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

</head>

Adding Content

Next, we will make<body> tag.

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

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

4.15.4 ADD HTML HEADINGS TO WEB PAGE

In HTML, headings are written in the following elements:

- <h1>
- <h2>
- <h3>
- <h4>

- <h5>
- <h6>

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

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

Creating Your Heading

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

```
<h1> Welcome To My Page </h1>
```

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

4.15. 5 ADD TEXT IN HTML

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

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

4.15.6 ADD LINKS IN HTML

As you may have noticed, the internet is made up of lots of links.

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

Links are included in an attribute opened by the `<a>` tag. This element is the first that we've met which uses an attribute and so it **looks different to previously mentioned tags**.

```
<a href=http://www.google.com>Google</a>
```

4.15.7 IMAGE TAG

In today's modern digital world, images are everything. The `<img>` tag has everything you need to display images on your site. Much like the `<a>` anchor element, `<img>` also contains an attribute.

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

```

```

4.16 CSS

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

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

4.16.1 CSS Syntax

Selector {

Property 1 : value;

Property 2 : value;

Property 3 : value;

}

For example:

h1

{

Color: red;

Text-align: center;

}

#unique

{

color: green;

}

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

CSS Comment

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

CSS How-To

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

4.16.2 INLINE CSS

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

`<h3 style = "color:red"> Have a great day </h3>`

`<p style = "color:green"> I did this, I did that </p>`

4.16.3 INTERNAL CSS

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

```
<style>
```

```
H1{
```

```
Color:red;
```

```
}
```

```
</style>
```

```
<h3> Have a great day </h3>
```

4.16.4 EXTERNAL CSS

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

```
<head>
```

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

```
</head>
```

```
h1{
```

```
color:red;    //.css file
```

```
}
```

CSS Selectors

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

CSS Colors

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

4.16.5 CSS Background

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

- Attachment – It basically helps in controlling the mechanism of scrolling.

CSS BoxModel

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

4.17 DEPLOYING THE MODEL PREDICTING OUTPUT

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

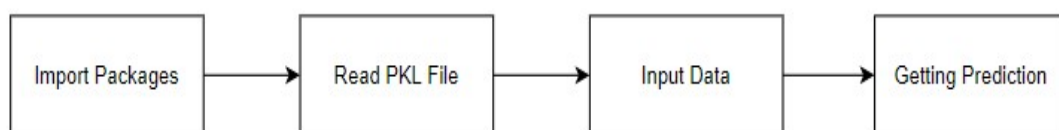


Fig 4.17 Module diagram

Chapter 5

RESULT AND DISCUSSION

5.1 DATA COLLECTION:

Gather diverse and comprehensive datasets encompassing clinical records, laboratory results, genomic data, and patient demographics.

Include information on symptoms, disease progression, treatment history, and relevant outcomes.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1		Category	Age	Sex	ALB	ALP	ALT	AST	BIL	CHE	CHOL	CREA	GGT	PROT					
2		1 0=Blood D	32 m		38.5	52.5	7.7	22.1	7.5	6.93	3.23	106	12.1	69					
3		2 0=Blood D	32 m		38.5	70.3	18	24.7	3.9	11.17	4.8	74	15.6	76.5					
4		3 0=Blood D	32 m		46.9	74.7	36.2	52.6	6.1	8.84	5.2	86	33.2	79.3					
5		4 0=Blood D	32 m		43.2	52	30.6	22.6	18.9	7.33	4.74	80	33.8	75.7					
6		5 0=Blood D	32 m		39.2	74.1	32.6	24.8	9.6	9.15	4.32	76	29.9	68.7					
7		6 0=Blood D	32 m		41.6	43.3	18.5	19.7	12.3	9.92	6.05	111	91	74					
8		7 0=Blood D	32 m		46.3	41.3	17.5	17.8	8.5	7.01	4.79	70	16.9	74.5					
9		8 0=Blood D	32 m		42.2	41.9	35.8	31.1	16.1	5.82	4.6	109	21.5	67.1					
10		9 0=Blood D	32 m		50.9	65.5	23.2	21.2	6.9	8.69	4.1	83	13.7	71.3					
11		10 0=Blood D	32 m		42.4	86.3	20.3	20	35.2	5.46	4.45	81	15.9	69.9					
12		11 0=Blood D	32 m		44.3	52.3	21.7	22.4	17.2	4.15	3.57	78	24.1	75.4					
13		12 0=Blood D	33 m		46.4	68.2	10.3	20	5.7	7.36	4.3	79	18.7	68.6					
14		13 0=Blood D	33 m		36.3	78.6	23.6	22	7	8.56	5.38	78	19.4	68.7					
15		14 0=Blood D	33 m		39	51.7	15.9	24	6.8	6.46	3.38	65	7	70.4					
16		15 0=Blood D	33 m		38.7	39.8	22.5	23	4.1	4.63	4.97	63	15.2	71.9					
17		16 0=Blood D	33 m		41.8	65	33.1	38	6.6	8.83	4.43	71	24	72.7					
18		17 0=Blood D	33 m		40.9	73	17.2	22.9	10	6.98	5.22	90	14.7	72.4					
19		18 0=Blood D	33 m		45.2	88.3	32.4	31.2	10.1	9.78	5.51	102	48.5	76.5					
20		19 0=Blood D	33 m		36.6	57.1	28.0	10.2	21.9	0.62	5.5	112	27.6	60.2					

5.2 PREPROCESSING:

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

to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyper parameters.

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

5.3 DATA VISUALIZATION

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

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

5.4 Feature Selection

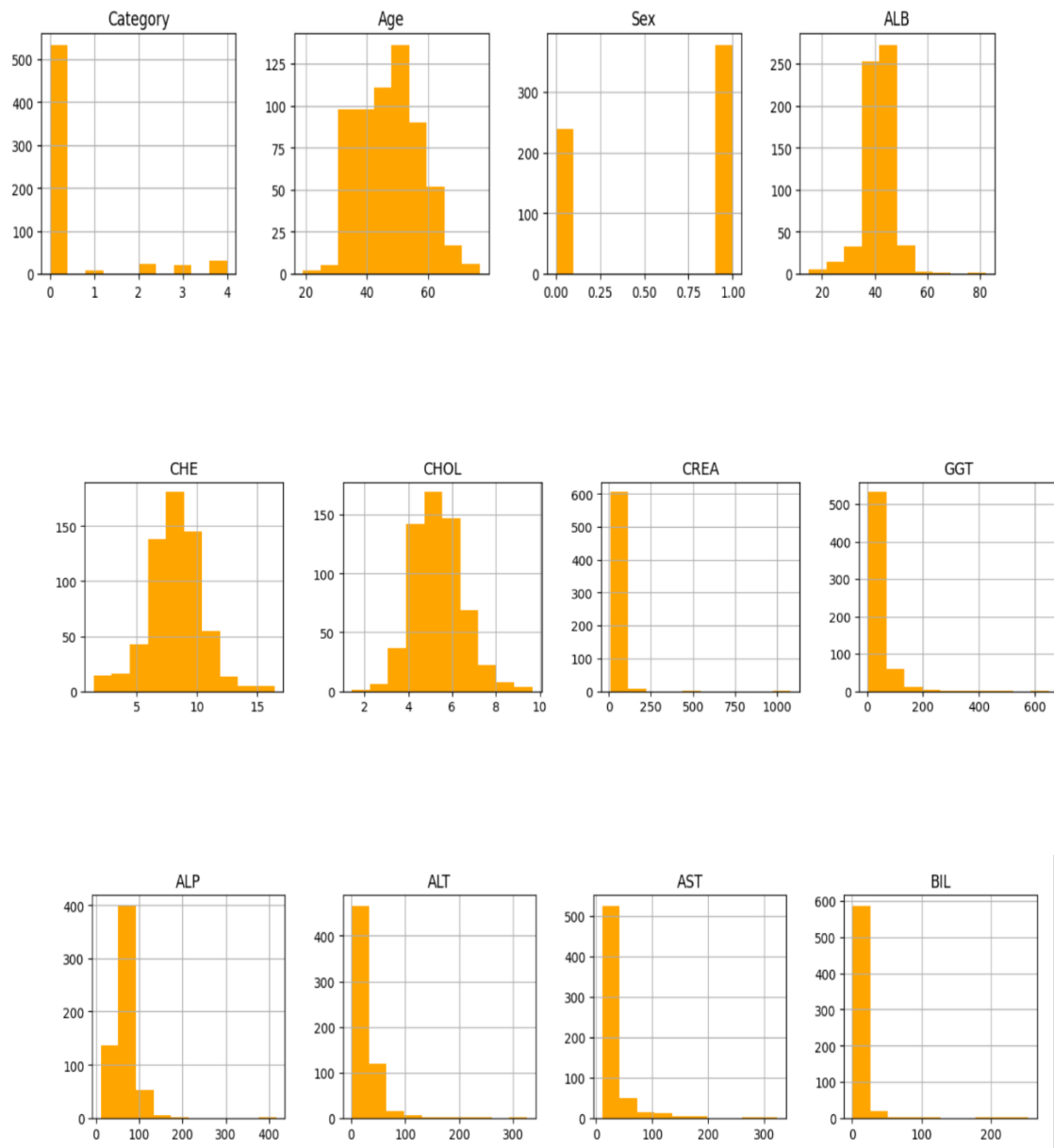


Fig 5.4 features selection

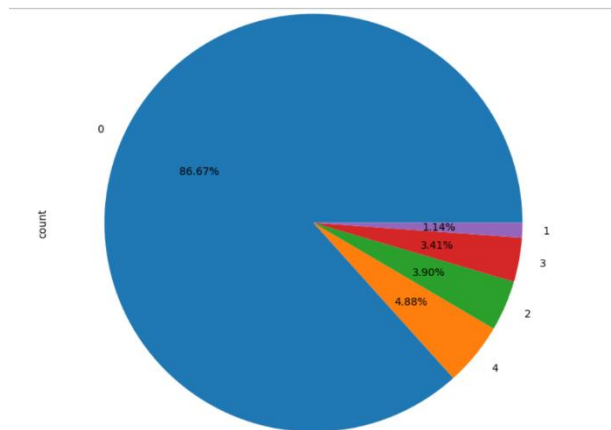


Fig 5.4 features selection

5.5 MODEL TRAINING:

Model training discussions in Hepatitis C virus prediction encompass a range of topics from data preparation and algorithm selection to hyperparameter tuning. Evaluation metrics interpretability ,and deployment consideration. Collaborative are essential for designing and deployment effective predictive models that improve patient outcomes and support clinical decision-making in intensive care settings.

SCREENSHOT:

```
from sklearn.metrics import accuracy_score
a = accuracy_score(y_test,predicted)
print("THE ACCURACY SCORE OF AdaBoostClassifier IS :",a*100)
```

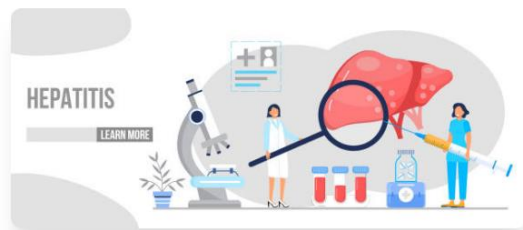
THE ACCURACY SCORE OF AdaBoostClassifier IS : 57.79467680608364

```
from sklearn.metrics import accuracy_score
a = accuracy_score(y_test,predicted)
print("THE ACCURACY SCORE OF LogisticRegression is :",a*100)
```

THE ACCURACY SCORE OF LogisticRegression is : 87.45247148288973

hepatitis C virus (HCV). Hepatitis C is a viral infection affecting the liver.

Early diagnosis and treatment can significantly improve outcomes and prevent the progression of liver disease.

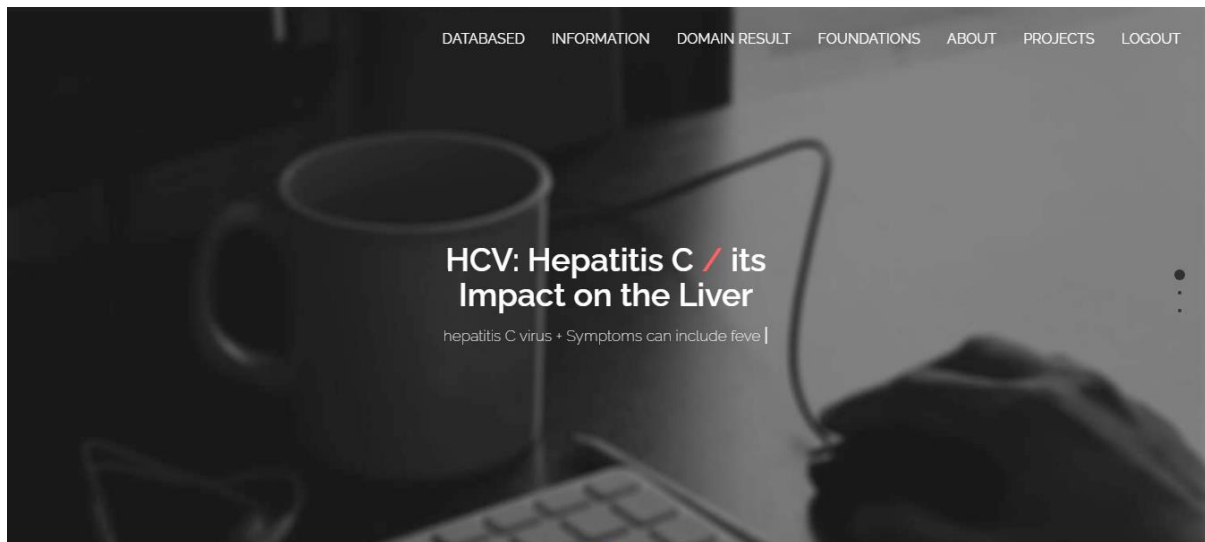


FOR
CONTINUE
PLEASE LOGIN

Sign In

Already Sing up [Sign in now](#)

SIGN IN



[← Home](#)

HEPATITIS C (HCV) PREDICTION USING SUPERVISED MACHINE LEARNING TECHNIQUES

Deployment in ML is the process of integrating a trained machine learning model into a production environment, enabling it to make real-time predictions or automate decision-making based on new data. It involves ensuring scalability, performance monitoring, and security while seamlessly integrating the model into existing systems.

AGE

AGE

GENDER

Male

CHEST_PAIN

CHAPTER 6

SUMMARY AND CONCLUSION

6.1 SUMMARY:

Hepatitis C virus prediction Medical decision support systems have been on the rise with technological advances and they have been the subject of many studies. By analyzing and visualizing the values in the data set, features that may be important for HCV was determined, and HCV estimation was made using various machine learning methods, pre-processing and feature extraction. According to the features obtained from this study, the estimation of HCV can be made automatically and can be a decision support system that helps the researchers and clinicians.

As we navigate this intersection of technology and healthcare, it becomes increasingly evident that machine learning is not just a tool but a catalyst for redefining standards in risk assessment and treatment strategies. The ongoing evolution of these methodologies offers hope for more effective, adaptive, and targeted interventions, emphasizing the continual progress in our quest for enhanced Hepatitis C care and the broader landscape of precision medicine.

symptoms serving as crucial indicators of both disease severity and treatment response, the application of machine learning techniques holds immense promise. By harnessing the power of artificial intelligence, healthcare professionals can not only predict the course of Hepatitis C with greater accuracy but also customize treatment plans that address the unique symptomatology and individual characteristics of patients.

6.2 CONCLUSION:

In conclusion, the integration of machine learning into the domain of Hepatitis C management marks a pivotal moment in healthcare, bringing forth substantial advancements in risk stratification and treatment optimization. By harnessing the analytical power of diverse datasets, ranging from clinical records to genomic information, machine learning models enable a nuanced understanding of disease

progression, symptomatology, and treatment responses. This transformative approach not only identifies high-risk individuals with unprecedented precision but also tailors treatment regimens based on individualized factors, ushering in an era of personalized medicine.

The impact of these advancements extends beyond mere predictive capabilities, fostering a paradigm shift towards patient-centric care in Hepatitis C. The ability to optimize treatment plans, minimize adverse effects, and enhance therapeutic outcomes underscores the potential of machine learning to significantly improve the quality of life for affected individuals. Moreover, the scalability and generalizability of these models hold promise for broader applications in infectious disease management.

6.3 FUTURE SCOPE :

The strides made in utilizing machine learning for risk stratification and treatment optimization in Hepatitis C open avenues for exciting future work and exploration. Here are potential directions for further research and development:

Integration of Real-time Data:

Incorporating real-time patient data streams to enable dynamic and continuous risk assessment, adapting treatment strategies based on evolving health conditions.

Exploration of Advanced Machine Learning Techniques:

Investigating the application of more advanced machine learning techniques, such as deep learning, to capture intricate patterns in the data and enhance predictive capabilities.

Longitudinal Studies and Outcome Prediction:

Conducting longitudinal studies to evaluate the long-term effectiveness of treatment plans and predicting sustained virological response (SVR) over extended periods.

Patient Engagement and Adherence Monitoring:

Developing tools to engage patients actively in their treatment plans and monitoring adherence through innovative solutions, potentially integrating wearable devices or mobile applications.

Incorporation of Multi-omics Data:

Expanding datasets to include multi-omics information, such as proteomics and metabolomics, for a more comprehensive understanding of the molecular basis of Hepatitis C and treatment response.

Explanatory Models for Clinical Decision Support:

Enhancing the interpretability of machine learning models to provide clear explanations for clinical decision support, fostering trust and understanding among healthcare professionals.

Validation in Diverse Patient Populations:

Conducting rigorous validation studies across diverse patient populations to ensure the generalizability and effectiveness of machine learning models in different healthcare settings.

Cost-effectiveness Analysis:

Conducting cost-effectiveness analyses to evaluate the economic impact of implementing machine learning-driven strategies in Hepatitis C management.

Collaboration with Healthcare Providers:

Establishing collaborative efforts with healthcare providers to seamlessly integrate machine learning models into existing clinical workflows, ensuring practical usability and adoption.

Ethical Considerations and Data Privacy:

Addressing ethical considerations and data privacy concerns associated with the use of machine learning in healthcare, ensuring compliance with regulations and safeguarding patient information.

By addressing these areas, future work can contribute to the continued evolution of machine learning applications in Hepatitis C care, advancing the field towards more personalized, adaptive, and effective interventions for patients.

CHAPTER 7

REFERENCES

Abd El-Salam, S. M., Ezz, M. M., Hashem, S., Elakel, W., Salama, R., ElMakhzangy, H., & ElHefnawi, M. (2019). Performance of machine learning approaches on prediction of esophageal varices for Egyptian chronic hepatitis C patients. *Informatics in Medicine Unlocked*, 17, 100267.

<https://doi.org/10.1016/j.imu.2019.100267>

Chang, P. C., Fan, C. Y., & Dzan, W. Y. (2010). A CBR-based fuzzy decision tree approach for database classification. *Expert Systems with Applications*, 37(1), 214-225. <https://doi.org/10.1016/j.eswa.2009.05.035>

Deo, R. C. (2015). Machine learning in medicine. *Circulation*, 132(20), 1920-1930. <https://doi.org/10.1161/CIRCULATIONAHA.115.001593>

Gould, R. (2021). Toward data-scientific thinking. *Teaching Statistics*, 43, S11-S22. <https://doi.org/10.1111/test.12334>

Gower, E., Estes, C., Blach, S., Razavi-Shearer, K., & Razavi, H. (2014). Global epidemiology and genotype distribution of the hepatitis C virus infection. *Journal of hepatology*, 61(1), S45-S57. <https://doi.org/10.1016/j.jhep.2014.07.027>

Chang, P. C., Fan, C. Y., & Dzan, W. Y. (2010). A CBR-based fuzzy decision tree approach for database classification. *Expert Systems with Applications*, 37(1), 214-225. <https://doi.org/10.1016/j.eswa.2009.05.035>

Deo, R. C. (2015). Machine learning in medicine. *Circulation*, 132(20), 1920-1930. <https://doi.org/10.1161/CIRCULATIONAHA.115.001593>

Gould, R. (2021). Toward data-scientific thinking. *Teaching Statistics*, 43, S11-S22. <https://doi.org/10.1111/test.12334>

Gower, E., Estes, C., Blach, S., Razavi-Shearer, K., & Razavi, H. (2014). Global epidemiology and genotype distribution of the hepatitis C virus infection. *Journal of hepatology*, 61(1), S45-S57. <https://doi.org/10.1016/j.jhep.2014.07.027>

Hashem, S., Esmat, G., Elakel, W., Habashy, S., Raouf, S. A., Elhefnawi, M., ... & ElHefnawi, M. (2017). Comparison of machine learning approaches for prediction of advanced liver fibrosis in chronic hepatitis C patients. *IEEE/ACM transactions on computational biology and bioinformatics*, 15(3), 861-868. <https://doi.org/10.1109/TCBB.2017.2690259>

Lin, J. H., & Haug, P. J. (2006). Data preparation framework for preprocessing clinical data in data mining. In *AMIA annual symposium proceedings* (Vol. 2006, p. 489). American Medical Informatics Association. <https://doi.org/10.1197/jamia.M1905>

Nilashi, M., Ibrahim, O., Ahmadi, H., & Shahmoradi, L. (2017). A knowledge-based system for breast cancer classification using fuzzy logic method. *Telematics and Informatics*, 34(4), 133-144. <https://doi.org/10.1016/j.tele.2016.10.006>

Seera, M., & Lim, C. P. (2014). A hybrid intelligent system for medical data classification. *Expert systems with applications*, 41(5), 2239-2249. <https://doi.org/10.1016/j.eswa.2013.09.045>

Sehume, O. L., & Markus, E. D. (2020). A critical analysis of medical robotic assistive systems for early diagnosis of common ailments in South Africa. *Int J Mech Eng Robot Res*, 9(10), 1451-1456. <https://doi.org/10.18178/ijmerr.9.10.1451-1456>

Simmonds, P., Bukh, J., Combet, C., Deléage, G., Enomoto, N., Feinstone, S., ... & Widell, A. (2005). Consensus proposals for a unified system of nomenclature of hepatitis C virus genotypes. *Hepatology*, 42(4), 962-973. <https://doi.org/10.1002/hep.20819>

Sudars, K., Jasko, J., Namatevs, I., Ozola, L., & Badaukis, N. (2020). Dataset of annotated food crops and weed images for robotic computer vision control. *Data in brief*, 31, 105833. <https://doi.org/10.1016/j.dib.2020.105833>

APPENDIX:

MODEL-1

DATA PREPROCESSING AND DATA CLEANING

```
import pandas as pd  
import numpy as np
```

```
import warnings  
warnings.filterwarnings('ignore')
```

```
df = pd.read_csv('HepatitisCdata.csv')  
df.head()
```

```
df.tail()  
del df['Unnamed: 0']  
df.describe()
```

```
df.shape
```

```
df.size
```

```
df.columns
```

```
df.isnull().sum()
```

```
df = df.dropna()
```

```
df.describe()
```

```
df.isnull().sum()
```

```
df.info()
```

```
pd.crosstab(df["CREA"], df["Category"])
```

```
df["Category"].value_counts()
```

```
pd.Categorical(df["PROT"]).describe()
```

```
df.duplicated()
```

```
sum(df.duplicated())
```

MODEL-2

DATA VISUALIZATION AND DATA ANALYSIS

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv('HepatitisCdata.csv')
df.head()
df.columns
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
var=['Category','Sex']
for i in var:
    df[i]=le.fit_transform(df[i])
del df['Unnamed: 0']
plt.figure(figsize=(12,7))
sns.countplot(x='Category',data=df)
plt.figure(figsize=(15,5))

plt.subplot(1,2,1)
plt.hist(df['GGT'],color='red')

plt.subplot(1,2,2)
plt.hist(df['ALB'],color='blue')
df.hist(figsize=(15,55),layout=(15,4), color='orange')
plt.show()
plt.boxplot(df["AST"])
    plt.scatter(df['BIL'],df["CHE"]) # scatter, plot, triplot, stackplot
df['ALB'].plot(kind='density')
```

```

fig, ax = plt.subplots(figsize=(20,15))
sns.heatmap(df.corr(),annot = True, fmt='0.2%',cmap = 'autumn',ax=ax)
def plot(df, variable):
    dataframe_pie = df[variable].value_counts()
    ax = dataframe_pie.plot.pie(figsize=(9,9), autopct='%1.2f%%', fontsize = 10)
    ax.set_title(variable + ' \n', fontsize = 10)
    return np.round(dataframe_pie/df.shape[0]*100,2)

```

```

plot(df, 'Category')

```

MODEL-3

BaggingClassifier Algorithm

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
#https://archive.ics.uci.edu/dataset/571/hcv+data
df = pd.read_csv('HepatitisCdata.csv')
df.head()
df.columns

from sklearn.preprocessing import LabelEncoder
lab = LabelEncoder()
var = ['Category', 'Sex']
for i in var:
    df[i] = lab.fit_transform(df[i])
df.info()
df.head()
x1 = df.drop(labels='Category', axis=1)
y1 = df.loc[:,'Category']
df['Category'].value_counts()

```

```

import imblearn
from imblearn.over_sampling import RandomOverSampler
from collections import Counter

ros =RandomOverSampler(random_state=42)
x,y=ros.fit_resample(x1,y1)
print("OUR DATASET COUNT      :", Counter(y1))
print("OVER SAMPLING DATA COUNT :", Counter(y))
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.30)
print("NUMBER OF TRAIN DATASET  :", len(x_train))
print("NUMBER OF TEST DATASET   :", len(x_test))
print("TOTAL NUMBER OF DATASET  :", len(x_train)+len(x_test))
print("NUMBER OF TRAIN DATASET  :", len(y_train))
print("NUMBER OF TEST DATASET   :", len(y_test))
print("TOTAL NUMBER OF DATASET  :", len(y_train)+len(y_test))
# df['Type'].unique()
from sklearn.ensemble import BaggingClassifier
BC = BaggingClassifier()
BC.fit(x_train,y_train)
predicted = BC.predict(x_test)
from sklearn.metrics import classification_report
cr = classification_report(y_test,predicted)
print('THE CLASSIFICATION REPORT OF BaggingClassifier:\n\n',cr)
from sklearn.metrics import accuracy_score
a = accuracy_score(y_test,predicted)
print("THE ACCURACY SCORE OF BaggingClassifier IS :",a*100)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,predicted)
print('THE CONFUSION MATRIX SCORE OF BaggingClassifier:\n\n\n',cm)
from sklearn.model_selection import cross_val_score
accuracy = cross_val_score(BC, x, y, scoring='accuracy')
print('THE CROSS VALIDATION TEST RESULT OF ACCURACY :\n\n\n',
accuracy*100)

```

```

from sklearn.metrics import hamming_loss
hl = hamming_loss(y_test,predicted)
print("THE HAMMING LOSS OF BaggingClassifier IS :",hl*100)
def plot_confusion_matrix(cm, title='THE CONFUSION MATRIX SCORE OF
BaggingClassifier\n\n', cmap=plt.cm.cool):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()

cm1=confusion_matrix(y_test, predicted)
print('THE CONFUSION MATRIX SCORE OF BaggingClassifier:\n\n')
print(cm)
plot_confusion_matrix(cm)
import matplotlib.pyplot as plt
df2 = pd.DataFrame()
df2["y_test"] = y_test
df2["predicted"] = predicted
df2.reset_index(inplace=True)
plt.figure(figsize=(20, 5))
plt.plot(df2["predicted"][:100], marker='x', linestyle='dashed', color='red')
plt.plot(df2["y_test"][:100], marker='o', linestyle='dashed', color='green')
plt.show()
import joblib
joblib.dump(BC,'BC.pkl')

```

MODEL-4

LogisticRegression Algorithm

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

```

```

import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
df = pd.read_csv('HepatitisCdata.csv')
df.head()
del df['Unnamed: 0']
df['Category'].unique()
df['Category'].value_counts()
from sklearn.preprocessing import LabelEncoder

lab = LabelEncoder()

var = ['Category', 'Sex']

for i in var:
    df[i] = lab.fit_transform(df[i])
df.head()
df.isnull().sum()
df=df.dropna()
df.head()
df.info()
x1 = df.drop(labels='Category', axis=1)
y1 = df.loc[:, 'Category']
x1
df.head()
df.isnull().sum()
import imblearn
from imblearn.over_sampling import RandomOverSampler
from collections import Counter

ros =RandomOverSampler(random_state=42)
x,y=ros.fit_resample(x1,y1)
print("OUR DATASET COUNT      : ", Counter(y1))

```

```

print("OVER SAMPLING DATA COUNT :", Counter(y))
x1
y1
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20,
random_state=42, stratify=y)
print("NUMBER OF TRAIN DATASET  :", len(x_train))
print("NUMBER OF TEST DATASET   :", len(x_test))
print("TOTAL NUMBER OF DATASET  :", len(x_train)+len(x_test))
print("NUMBER OF TRAIN DATASET  :", len(y_train))
print("NUMBER OF TEST DATASET   :", len(y_test))
print("TOTAL NUMBER OF DATASET  :", len(y_train)+len(y_test))
# df['Type'].unique()
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression()
LR.fit(x_train,y_train)
predicted = LR.predict(x_test)
from sklearn.metrics import classification_report
cr = classification_report(y_test,predicted)
print('THE CLASSIFICATION REPORT OF LogisticRegression:\n\n',cr)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,predicted)
print('THE CONFUSION MATRIX SCORE OF LogisticRegression:\n\n\n',cm)
from sklearn.model_selection import cross_val_score
accuracy = cross_val_score(LR, x, y, scoring='accuracy')
print('THE CROSS VALIDATION TEST RESULT OF ACCURACY :\n\n\n',
accuracy*100)
from sklearn.metrics import accuracy_score
a = accuracy_score(y_test,predicted)
print("THE ACCURACY SCORE OF LogisticRegression is :",a*100)
from sklearn.metrics import hamming_loss
hl = hamming_loss(y_test,predicted)
print("THE HAMMING LOSS OF LogisticRegression IS :",hl*100)

```

```

def plot_confusion_matrix(cm, title='THE CONFUSION MATRIX SCORE OF
RANDOM FOREST CLASSIFIER\n\n', cmap=plt.cm.cool):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()

cm1=confusion_matrix(y_test, predicted)
print('THE CONFUSION MATRIX SCORE OF LogisticRegression:\n\n')
print(cm)
plot_confusion_matrix(cm)
import matplotlib.pyplot as plt
df2 = pd.DataFrame()
df2["y_test"] = y_test
df2["predicted"] = predicted
df2.reset_index(inplace=True)
plt.figure(figsize=(20, 5))
plt.plot(df2["predicted"][:100], marker='x', linestyle='dashed', color='red')
plt.plot(df2["y_test"][:100], marker='o', linestyle='dashed', color='green')
plt.show()
import joblib
joblib.dump(LR,'LR.pkl')

```

MODEL-5

RANDOM FOREST CLASSIFIER ALGORITHM

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')

```

```

#https://archive.ics.uci.edu/dataset/571/hcv+data
df = pd.read_csv('HepatitisCdata.csv')
df.head()
df.columns

from sklearn.preprocessing import LabelEncoder
lab = LabelEncoder()
var = ['Category', 'Sex']
for i in var:
    df[i] = lab.fit_transform(df[i])
df.info()
df.head()
x1 = df.drop(labels='Category', axis=1)
y1 = df.loc[:, 'Category']
df['Category'].value_counts()
import imblearn
from imblearn.over_sampling import RandomOverSampler
from collections import Counter

ros =RandomOverSampler(random_state=42)
x,y=ros.fit_resample(x1,y1)
print("OUR DATASET COUNT      :", Counter(y1))
print("OVER SAMPLING DATA COUNT :", Counter(y))
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.30)
print("NUMBER OF TRAIN DATASET  :", len(x_train))
print("NUMBER OF TEST DATASET   :", len(x_test))
print("TOTAL NUMBER OF DATASET  :", len(x_train)+len(x_test))
print("NUMBER OF TRAIN DATASET  :", len(y_train))
print("NUMBER OF TEST DATASET   :", len(y_test))
print("TOTAL NUMBER OF DATASET  :", len(y_train)+len(y_test))
# df['Type'].unique()

from sklearn.ensemble import RandomForestClassifier
RFC = RandomForestClassifier()
RFC.fit(x_train,y_train)

```

```

predicted = RFC.predict(x_test)
from sklearn.metrics import classification_report
cr = classification_report(y_test,predicted)
print('THE CLASSIFICATION REPORT OF RANDOM FOREST
CLASSIFIER:\n\n',cr)
from sklearn.metrics import accuracy_score
a = accuracy_score(y_test,predicted)
print("THE ACCURACY SCORE OF RANDOM FOREST CLASSIFIER IS :",a*100)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,predicted)
print('THE CONFUSION MATRIX SCORE OF RANDOM FOREST
CLASSIFIER:\n\n\n',cm)
from sklearn.model_selection import cross_val_score
accuracy = cross_val_score(RFC, x, y, scoring='accuracy')
print('THE CROSS VALIDATION TEST RESULT OF ACCURACY :\n\n\n',
accuracy*100)
from sklearn.metrics import hamming_loss
hl = hamming_loss(y_test,predicted)
print("THE HAMMING LOSS OF RANDOM FOREST CLASSIFIER IS :",hl*100)
def plot_confusion_matrix(cm, title='THE CONFUSION MATRIX SCORE OF
RANDOM FOREST CLASSIFIER\n\n', cmap=plt.cm.cool):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()

cm1=confusion_matrix(y_test, predicted)
print('THE CONFUSION MATRIX SCORE OF RANDOM FOREST
CLASSIFIER:\n\n')
print(cm)
plot_confusion_matrix(cm)
import matplotlib.pyplot as plt
df2 = pd.DataFrame()
df2["y_test"] = y_test
df2["predicted"] = predicted

```

```
df2.reset_index(inplace=True)
plt.figure(figsize=(20, 5))
plt.plot(df2["predicted"][:100], marker='x', linestyle='dashed', color='red')
plt.plot(df2["y_test"][:100], marker='o', linestyle='dashed', color='green')
plt.show()
import joblib
joblib.dump(RFC, 'RFC.pkl')
```