

A WEB BASED APPLICATION FOR PREDICTING LIVER DISEASE USING MACHINE LEARNING

A PROJECT REPORT

Submitted by

SWETHA M

SRI HARINI G J

VARNAMALYA G

VIJAY RAJ K

In partial fulfillment for the award of the degree

of

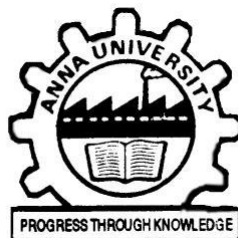
BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

PSNA COLLEGE OF ENGINEERING AND TECHNOLOGY,

DINDIGUL – 624622.



ANNA UNIVERSITY CHENNAI - 600 025

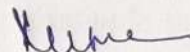
APRIL 2021

ANNA UNIVERSITY CHENNAI - 600 025

BONAFIDE CERTIFICATE

Certified that this project report "A WEB BASED APPLICATION FOR PREDICTING LIVER DISEASE USING MACHINE LEARNING" is the bonafide work of SWETHA M (921317104219), SRI HARINI G J (921317104212), VARNAMALYA G (921317104234), VIJAY RAJ K (921317104241) who carried out the project under my supervision.


SIGNATURE


SIGNATURE

Dr. D. Shanthi, M.E., Ph.D.,

Dr. N. Uma Maheswari, M.E., Ph.D.,

HEAD OF THE DEPARTMENT

SUPERVISOR

PROFESSOR

PROFESSOR

DEPARTMENT OF CSE,

DEPARTMENT OF CSE,

**PSNA COLLEGE OF
ENGINEERING
AND TECHNOLOGY**

**PSNA COLLEGE OF
ENGINEERING
AND TECHNOLOGY**

DINDIGUL-624622.

DINDIGUL-624622.

Submitted for the university project viva-voce held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

With warm hearts and immense pleasure, we thank the Almighty for his grace and blessings which drove me to the successful completion of the project. We would like to express our gratitude towards our parents for their kind co-operation and encouragement which help me in completion of this project.

We take this opportunity to express our sincere thanks to the respected **Chairperson Tmt. K. Dhanalakshmi Ammal**, who is the guiding light for all the activities in my college. We would like to express our deep gratitude to our **Pro-Chairman Rtn. MPH F R.S.K. Raguraam, D.A.E M.Com** for their continuous support towards the development of the students.

We would like to thank our Principal **Dr. D. Vasudevan., M.E., Ph.D.** for being a beacon in guiding every one of us and infusing us the strength and enthusiasm to work over successfully.

We express our sincere thanks and heartfelt gratitude to **Dr. D. Shanthi, M.E., Ph.D.**, Professor and Head, Department of Computer Science and Engineering for her valuable suggestions and continuous encouragement in the completion of the project work.

This project would not have been possible without the motivation and guidance of our project coordinator **Dr. R. Karthikeyan, M.E., Ph.D.**, Professor and project guide **Dr. N. Uma Maheswari, M.E., Ph.D.**, Professor of the Department of Computer Science and Engineering.

ABSTRACT

Today's Health care is a very important aspect for every human being, hence there is a need to provide medical services that are easily available to everyone. The liver is the largest organ of the body and it is essential for digesting food and releasing the toxic element of the body. Liver disorders have increased rapidly and it is considered to be a very fatal disease in many countries.

Since the lifestyle of the human changes, their cuisines and eating habit also changes. But this change is not easily accepted by the internal organs. Especially digestion is important process which gets affected. The liver regulates most chemical levels in the blood and excretes a product called bile. All the blood leaving the stomach and intestines passes through the liver. Liver helps in excretion of cholesterol, hormone, bilirubin and drugs. They also do metabolism activity which involves protein, carbohydrate and fats. Yearly 2 million people dies due to the Liver disease according to National Library of Medicine, USA. So, it is necessary to predict the possibility of the disease before it gets vigorous.

The main focus of the proposed project is to predict the liver disease based on Machine Learning approach using classification algorithms. Machine learning has made a significant impact on the biomedical field for liver disease prediction and diagnosis. Machine learning offers a guarantee for improving the detection and prediction of disease that has been made an interest in the biomedical field and they also increase the objectivity of the decision-making process. The main aspect is to predict the results more efficiently and reduce the cost of diagnosis in the medical sector. Therefore, we used different classification techniques for the classification of patients have liver disease

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	iv
	LIST OF TABLES	viii
	LIST OF FIGURES	ix
	LIST OF ABBREVIATIONS	ix
1	INTRODUCTION	1
	1.1 PROJECT OVERVIEW	1
	1.2 AIM	4
	1.3 OBJECTIVE	4
2	SYSTEM ANALYSIS	5
	2.1 LITERATURE SURVEY	5
	2.2 EXISTING SYSTEM	9
	2.2.1 Drawback of Existing System	9
	2.3 NEED FOR PROPOSED SYSTEM	9
	2.4 FEATURES OF PROPOSED SYSTEM	9
	2.4.1 Feasibility Study	9
	2.5 REQUIREMENT ANALYSIS	11
	2.5.1 Hardware Requirements	11
	2.5.2 Software Requirements	11
	2.5.2.1 Tools	11

	2.5.2.2 Programming language	13
	2.5.2.3 Database	16
	2.6 DATASET DESCRIPTION	17
	2.6.1 CSV	17
	2.6.2 Data Collection	18
	2.6.3 Info About Dataset	19
3	SYSTEM DESIGN	20
	3.1 UML DIAGRAMS	20
	3.1.1 Use Case Diagram	20
	3.1.2 Class Diagram	21
	3.1.3 Sequence Diagram	22
	3.1.4 Activity Diagram	23
	3.1.5 ER Diagram	24
	3.1.6 System Architecture Diagram	25
	3.2 DATABASE DESIGN	27
	3.2.1 Biological Data	27
4	IMPLEMENTATION AND TESTING	28
	4.1 MODULE DESCRIPTION	28
	4.1.1 0 LEVEL	28
	4.1.2 1 LEVEL	28

4.1.3	2 LEVEL	29
4.1.4	Module Split-up	30
4.2	IMPLEMENTATION	32
4.3	TESTING	37
4.3.1	Unit Testing	38
4.3.2	Integration Testing	38
4.3.3	Validation Testing	38
4.3.4	Functional Testing	38
4.3.5	Structural Testing	39
4.4	DRIVING TEST CASES	39
5	CONCLUSION AND FUTURE WORKS	41
5.1	CONCLUSION	42
5.2	FUTURE WORKS	43
	REFERENCES	44
	APPENDIX	x

LIST OF TABLES

S.NO	TABLE NO	TABLE NAME	PAGE NO
1	3.1	Biological Data	27
2	4.1	Data Preprocessing Technique	34
3	4.2	Comparison Table	36
4	4.3	Driving Test case	39

LIST OF FIGURES

S.NO	FIGURE NO	FIGURE NAME	PAGE NO
1	1.1	Machine Learning	1
2	1.2	Classification Algorithm	2
3	2.1	Dataset from Kaggle	19
4	3.1	Use case diagram	21
5	3.2	Class Diagram	22
6	3.3	Sequence Diagram	23
7	3.4	Activity Diagram	24
8	3.5	ER Diagram	25
9	3.6	System Architecture Diagram	26
10	4.1	0 LEVEL Implementation	28
11	4.2	1 LEVEL Implementation	28
12	4.3	2 LEVEL Implementation	29
13	4.4	Creating a model	31
14	4.5	Data pre-processing	32

15	4.6	Null value count	33
16	4.7	After pre-processing	35
17	4.8	Feature selection	35
18	4.9	Input UI	36
19	4.10	Output UI	37

LIST OF ABBREVIATIONS

S.NO	ACRONYMS	ABBREVIATIONS
1	HTML	H ypertext M ark-up L anguage
2	CSS	C ascade S tyle S heet
3	SVM	S upport V ector M achine
4	CSV	C omma S eparated V alue
5	DB	D atabase
6	UML	U nified M odelling L anguage
7	KNN	K Nearest Neighbor
8	NB	N aive B ayes
9	SDK	S oftware D evelopment K it
10	HDL	H igh-level D ata L ink

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves.

Machine learning algorithms are often categorized as supervised or unsupervised.

Supervised machine learning algorithms can apply what has been learned in the past to new data using labeled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.

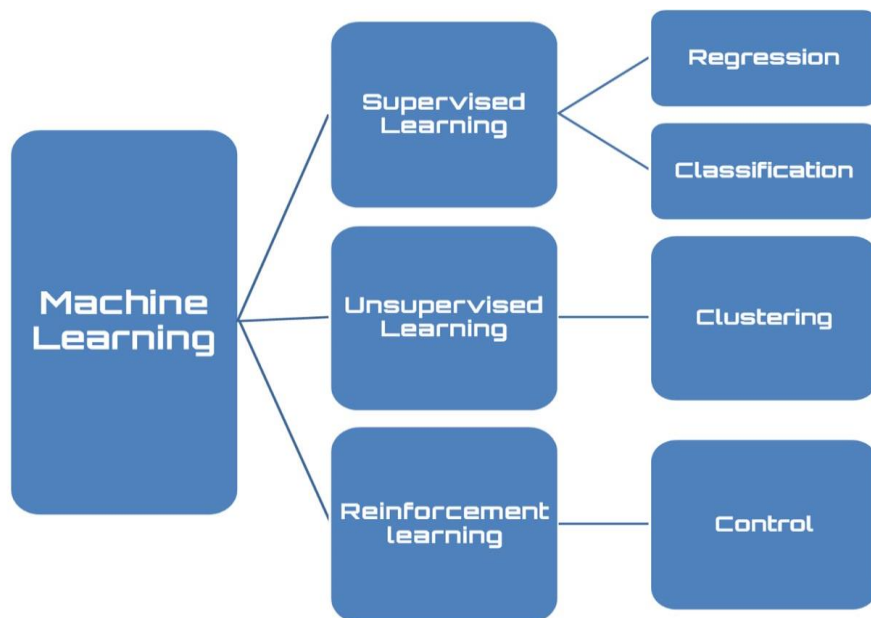


Fig 1.1 Machine Learning

In contrast, unsupervised machine learning algorithms are used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data. The system doesn't figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.

Naive Bayes Algorithm

Naïve Bayes algorithm may be a supervised learning algorithm, which is predicated on Bayes theorem and used for solving classification problems. It's not one algorithm but a family of algorithms where all of them share a standard principle, i.e. every pair of features being classified is independent of every other. Naive Bayes Classifier is one among the straightforward and best Classification algorithms which helps in building the fast machine learning models which will make quick predictions. Naive Bayes is one of the powerful machine learning algorithms that is used for classification.

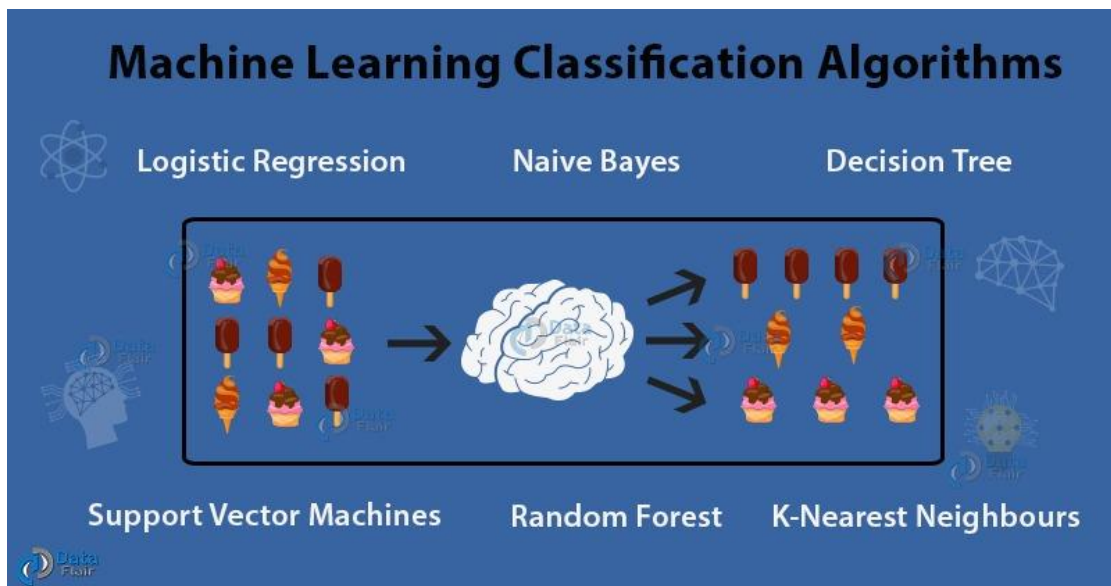


Fig 1.2 Classification Algorithm

K-Nearest Neighbors Algorithm

K-nearest neighbors is one of the most basic yet important classification algorithms in machine learning. KNNs belong to the supervised learning domain and have several applications in pattern recognition, data mining, and intrusion detection. These KNNs are used in real-life scenarios where non-parametric algorithms are required. These algorithms do not make any assumptions about how the data is distributed.

Support Vector Machine Algorithm

Support Vector Machines are a type of supervised machine learning algorithm that provides analysis of data for classification and regression analysis. While they can be used for regression, SVM is mostly used for classification. We carry out plotting in the n-dimensional space. The value of each feature is also the value of the specified coordinate. Then, we find the ideal hyperplane that differentiates between the two classes. These support vectors are the coordinate representations of individual observation. It is a frontier method for segregating the two classes.

1.2 AIM

To host a website which allows user to enter their biological data related to symptoms of liver disease, which is then compared with developed machine learning model and classify the user as prone to disease or not.

1.3 OBJECTIVE

- Liver disease (also called hepatic disease) is a type of damage to or disease of the liver. The liver is the largest organ in the body.
- Amongst its 500+ roles, the liver is responsible for food processing, energy storage, blood filtration, and immune response. Specifically, the liver contributes by secreting bile for lipid breakdown, storing excess glucose as

glycogen, and removing bacteria and toxins from the blood. Whenever the course of the problem lasts long, chronic liver disease ensues.

- Therefore, this makes the way to develop a system that helps in prediction of liver disease.
- Proposed model should able to get the data from user and compare that with existing model and should show the risk of affecting disease should be shown.
- With help of the prediction, the patient should be able to advance their medication in order to avoid getting disease at severe level

CHAPTER 2

SYSTEM ANALYSIS

System analysis is a problem-solving technique that decomposes a system in to its component pieces for the purpose of studying how well those component parts work and interact to accomplish their purposes. System analysis is the process of studying a procedure or business in order to identify its goals and purposes and create systems and procedures that will achieve them in an efficient way. Analysis and synthesis, as a scientific method, always go hand in hand; they complement one another. Every synthesis is built up on the results of a preceding analysis, and every analysis requires a subsequent synthesis in order to verify and correct its results.

2.1 LITERATURE SURVEY

Performance Assessment of Classification Algorithms on Early Detection of Liver Syndrome

Year: 2020

Author: Rashid Naseem, Bilal Khan, M. A. Shah, Karzan Wakil, Atif Khan, Wael Alosaimi, M. I. Uddin, Badar Alouffi

Methodology: In the recent era, a liver syndrome that causes any damage in life capacity is exceptionally normal everywhere throughout the world. It has been found that liver disease is exposed more in young people as a comparison with other aged people. At the point when liver capacity ends up, life endures just up to 1 or 2 days scarcely, and it is very hard to predict such illness in the early stage. Researchers are trying to project a model for early prediction of liver disease utilizing various machine learning approaches. However, this study compares ten classifiers including A1DE, NB, MLP, SVM, KNN, CHIRP, CDT, Forest-PA, J48, and RF to find the optimal solution for early and accurate prediction of liver disease. The datasets utilized in this study are taken from the UCI ML repository and the GitHub repository. The outcomes are assessed via RMSE, RRSE, recall, specificity,

precision, G-measure, F-measure, MCC, and accuracy. The exploratory outcomes show a better consequence of RF utilizing the UCI dataset. Assessing RF using RMSE and RRSE, the outcomes are 0.4328 and 87.6766, while the accuracy of RF is 72.1739% that is also better than other employed classifiers. However, utilizing the GitHub dataset, SVM beats other employed techniques in terms of increasing accuracy up to 71.3551%.

Publication: Journal of HealthCare Engineering vol. 2020, Article ID 6680002, 13 pages, 2020.

External URL: <https://www.hindawi.com/journals/jhe/2020/6680002/>

Diagnosis of Liver Disease using Machine Learning Model

Year: 2020

Author: A. Sivasangari; Baddigam Jaya Krishna Reddy; Annamareddy Kiran; P. Ajitha

Methodology: Liver is one of the most important organs in the human body but due to unhealthy lifestyle and excessive alcohol intake, liver disease has been increasing at an alarming rate globally hence it calls for an immediate attention to predict the disease before it is too late. However, medical data is often associated to be imbalanced and complex. Liver disease is one of the key causes of high numbers of deaths in the country and is considered a life-threatening disease, not just anywhere, but worldwide. Liver disease can also impact peoples early in their life. More than 2.4 per cent of annual Indian deaths are due to liver disorders. It is also difficult to detect liver disease due to mild symptoms in the early stages. If it is too late the signs always come to light. Thus liver-related disease poses more problems for people living and is more important nowadays to recognize the causes, and identification phase. So, for early detection of liver disease, an automated program is needed to build with more accuracy and reliability. Specific machine learning models are

developed for this purpose to predict the disease. In this paper, the methods of Support Vector Machines (SVM), Decision Tree (DT) and Random Forest (RF) is proposed to predict liver disease with better precision, accuracy and reliability. Hence, the aim of this project is to investigate the data mining algorithm to predict liver disease on imbalanced data through random sampling. Results are compared and analysed based on accuracy and ROC index. K-Nearest Neighbour (k-NN) outperforms the other algorithms such as Logistic Regression, AutoNeural and Random Forest with the accuracy of 99.794%.

Publication: 2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)

External URL: <https://ieeexplore.ieee.org/document/9243375/authors#authors>

Software-based Prediction of Liver Disease with Feature Selection and Classification Techniques

Year: 2019

Author: Jagdeep Singha , Sachin Baggab , Ranjodh Kaur

Methodology: In the human body, the liver organ plays a very important role in many functions like decomposition of red blood cells, etc. The liver is the largest body organ and located in the upper right position of our abdomen. Today's health care is very important aspect for every human, so there is a need to provide medical services that are easily available to everyone. In this paper, the main focus is to predict the liver disease based on a software engineering approach using classification and feature selection technique. The implementation of proposed work is done on Indian Liver Patient Dataset (ILPD) from the University of California, Irvine database. Proposed work focuses on the development of the software that will help in the prediction of the liver diseases based upon the various symptoms.

Publication: International Conference on Computational Intelligence and Data Science (ICCIDS 2019)

External URL: <https://pdf.sciencedirectassets.com>

Strategic Analysis in Prediction of Liver Disease Using Different Classification Algorithms

Year: 2019

Author: Binish Khan, Piyush Kumar Shukla , Manish Kumar Ahirwar

Methodology: Liver diseases averts the normal function of the liver. Mainly due to the large amount of alcohol consumption liver disease arises. Early prediction of liver disease using classification algorithms is an efficacious task that can help the doctors to diagnose the disease within a short duration of time. Discovering the existence of liver disease at an early stage is a complex task for the doctors. The main objective of this paper is to analyse the parameters of various classification algorithms and compare their predictive accuracies so as to find out the best classifier for determining the liver disease. This paper focuses on the related works of various authors on liver disease such that algorithms were implemented using Weka tool that is a machine learning software written in Java. This paper compares various classification algorithms such as Random Forest, Logistic Regression and Separation Algorithm with an aim to identify the best technique. Based on this study, Random Forest with the highest accuracy outperformed the other algorithms and can be further utilised in the prediction of liver disease.

Publication: International Journal of Computer Sciences and Engineering Open Access Research Paper Vol.-7, Issue-7, July 2019

External URL: <https://doi.org/10.26438/ijcse/v7i7.7176>

2.2 EXISTING SYSTEM

A lot of researchers have used exclusive statistics and mining techniques for predicting liver disease in medical field. They have explored some techniques and carried out various algorithms specifically decision making tree, various regression methods and finally they have also given few ideas and suggestions.

2.2.1 Drawbacks of Existing System

1. The Existing system does not use feature selection and uses lot of features for developing a model.
2. Use of a lot attributes reduces the time and cost efficiency.
3. The existing system shows reduced accuracy as compared to the proposed system.
4. The existing system does not use any database to store the user's data

2.3 NEED FOR PROPOSED SYSTEM

The proposed system will predict the risk factor of the HCC affected patients using Machine Learning.

2.4 FEATURES OF PROPOSED SYSTEM

- Logistic Classification will be used to classify the processed dataset.
- Pre-processing and correct replacement of missing value with different attribute type will improve the model.
- Feature selection is done for better and faster prediction.

2.4.1 Feasibility Study

The feasibility of the project is analyzed in this phase and business proposal is but forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This

is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

1. Economic Feasibility
2. Technical Feasibility
3. Social Feasibility

Economic Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of funds a company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the development system as well within the budget and this was achieved because most of the technologies are freely available. +

Technical Feasibility

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 being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

Social Feasibility

The aspect of study is to check the level of acceptance of the system by user. This includes the process of training the user to the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods employed to educate the user about the system and to make him familiar with it. His level of confidence

must be raised but also able to make some constructive criticism, which is welcomed, as the final user of the system.

2.5 REQUIREMENT ANALYSIS

2.5.1. HARDWARE REQUIREMENTS

Processor	: i3 processor and above
RAM	: 4GB or more
Hard Disk	: 20 GB or more
Internet Bandwidth	: At least 5 Mbps

2.5.2. SOFTWARE REQUIREMENTS

Operating System	: Windows 10
Language used	: Python, Django framework
Design Interface	: Bootstrap, HTML, CSS
Browser	: Mozilla, Google chrome
Software and IDE	: Spyder IDE, Pandas, Numpy
Database	: Sqlite3

2.5.2.1 TOOLS:

SPYDER IDE

Spyder is an open-source cross-platform integrated development environment (IDE) for scientific programming in the Python language. Spyder integrates with a number of prominent packages in the scientific Python stack, including NumPy, SciPy, Matplotlib, pandas, IPython, SymPy and Cython, as well as other open-source software. It is released under the MIT license.

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

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

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

ANACONDA

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

Conda analyses the current environment including everything currently installed, and together with any version limitations specified (e.g. the user may wish to have Tensorflow version 2.0 or higher), works out how to install a compatible set of dependencies, and shows a warning if this cannot be done.

DJANGO FRAMEWORK

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

application, and is suitable for both frontend and backend. Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and "pluggability" of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings, files, and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.

2.4.2.2 PROGRAMING LANGUAGES

PYTHON

Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs and 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. Python is often described as a "batteries included" language due to its comprehensive standard library.

HTML

HTML (Hypertext Markup Language) is the set of markup symbols or codes inserted in a file intended for display on a World Web browser page. The markup tells the Web browser how to display a Web page's words and images for the users. Each individual markup code is referred to as an element (but many people also refer to it as a tag).Some elements come in pairs that indicate when some display effect is to begin and when it is to end.

HTML markup consists of several key components, including those called tags (and their attributes), character-based data types, character references and entity references. HTML tags most commonly come in pairs like `<h1>` and `</h1>` although some represent empty elements and so are unpaired, for example ``. The first tag in such a pair is the start tag, and the second is the end tag (they are also called opening and closing tags). Another important component is the HTML document style type declaration, which triggers standards mode rendering.

HTML is a formal Recommendation by the World Wide Web consortium (W3C) and is generally adhered to by the major browsers, Microsoft Internet Explorer and Netscape Navigator, which also provide some additional non-standard codes. The current version of HTML is HTML 4.0. However, both Internet Explorer and Netscape implement some features differently and provide non-standard extensions.

Hyper Text Markup Language, commonly referred to as HTML, is the standard markup language used to create web pages. Along with CSS and JavaScript, HTML is a cornerstone technology used to create web pages, as well as to create user interfaces for mobile and web applications.

CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language. Although most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document, including plain XML, SVG and XUL, and is applicable to rendering in speech, or on other media. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging Webpages, user interfaces for web applications, and user interfaces for many mobile applications.

CSS was first proposed by Håkon Wium Lie on October 10, 1994. At the time, Lie was working with Tim Berners-Lee at CERN. Several other style sheet

languages for the web were proposed around the same time, and discussions on public mailing lists and inside World Wide Web Consortium resulted in the first W3C CSS Recommendation (CSS1) being released in 1996. In particular, a proposal by Bert Bos was influential; he became co-author of CSS1, and is regarded as co-creator of CSS.

BOOTSTRAP

Bootstrap is a combination of HTML, CSS, and Javascript designed as an open-source framework for building clean user interfaces. Bootstrap was created by, and is still actively maintained by the Twitter team.

Bootstrap has become one of the most popular front-end frameworks and open-source projects in the world.

Bootstrap was originally made to create front end websites. Meanwhile, it is, however, also used to build the backend of websites and to create web applications. Bootstrap is often associated with the term “responsive design” because with this Twitter framework, user interfaces can be designed dynamically so that they always conform to the format and the display size of the device being used in each case, whether computer, netbook, tablet, or smartphone.

In order to be able to use bootstrap, it must first be downloaded and installed. At the official download site, you can decide if you want to load bootstrap completely or only parts of this framework. The downloaded zip file contains one CSS and one JavaScript file. These must be integrated into the head section of an HTML page. You can subsequently use the bootstrap specifications.

Bootstrap makes it possible to reduce the loading time of websites, since only those components must be installed which are really needed. This can positively impact a website’s ranking, since page speed plays an important role. Moreover,

Bootstrap simplifies the generation of clean code and improves the user experience on a website.

2.4.2.3 DATABASE

Sqllite3

SQLite is a relational database management system (RDBMS) contained in a C library. In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program.

SQLite is a popular choice as embedded database software for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems (such as mobile phones), among others. [\[8\]](#) SQLite has bindings to many programming languages.

Unlike client–server database management systems, the SQLite engine has no standalone processes with which the application program communicates. Instead, the SQLite library is linked in and thus becomes an integral part of the application program. Linking may be static or dynamic. The application program uses SQLite's functionality through simple function calls, which reduce latency in database access: function calls within a single process are more efficient than inter-process communication.

SQLite stores the entire database (definitions, tables, indices, and the data itself) as a single cross-platform file on a host machine. It implements this simple design by locking the entire database file during writing. SQLite read operations can be multitasked, though writes can only be performed sequentially.

Due to the server-less design, SQLite applications require less configuration than client–server databases. SQLite is called zero-conf because it does not require

service management (such as start-up scripts) or access control based on GRANT and passwords. Access control is handled by means of file-system permissions given to the database file itself. Databases in client–server systems use file-system permissions that give access to the database files only to the daemon process.

Another implication of the serverless design is that several processes may not be able to write to the database file. In server-based databases, several writers will all connect to the same daemon, which is able to handle its locks internally. SQLite, on the other hand, has to rely on file-system locks. It has less knowledge of the other processes that are accessing the database at the same time. Therefore, SQLite is not the preferred choice for write-intensive deployments. However, for simple queries with little concurrency, SQLite performance profits from avoiding the overhead of passing its data to another process.

2.6 DATASET DESCRIPTION:

A data set (or dataset) is a collection of data. In the case of tabular data, a data set corresponds to one or more database tables, where every column of a table represents a particular variable, and each row corresponds to a given record of the data set in question. The data set lists values for each of the variables, such as height and weight of an object, for each member of the data set. Each value is known as a datum. Data sets can also consist of a collection of documents or files. Here, data in the dataset is collected in CSV format.

2.6.1 CSV

A Comma Separated Values (CSV) file is a plain text file that contains a list of data. These files are often used for exchanging data between different applications. For example, databases and contact managers often support CSV files.

These files may sometimes be called Character Separated Values or Comma Delimited files. They mostly use the comma character to separate (or delimit) data, but sometimes use other characters, like semicolons. The idea is that you can export complex data from one application to a CSV file, and then import the data in that CSV file into another application.

2.6.2 Data Collection

Data is collected in CSV format from Kaggle.

Kaggle

Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges.

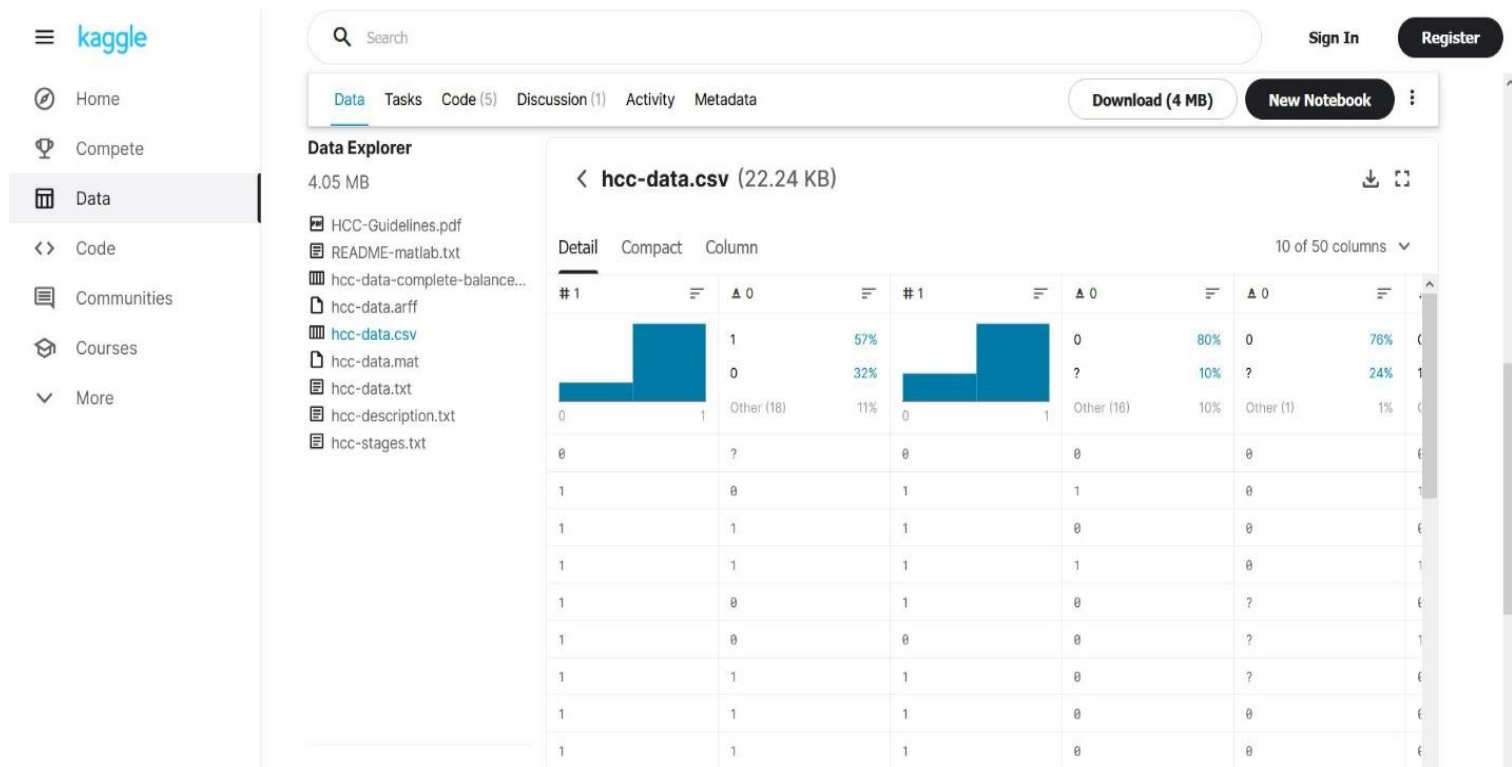


Fig 2.1 Dataset from Kaggle

2.6.3 Info about Dataset

- Last updated on 03.01.2019.
- Consist of 49 attribute and 1 class attribute
- Consist of 165 patient's data

CHAPTER 3

SYSTEM DESIGN

3.1 UML DIAGRAMS

The Unified Modelling Language is a standard language for specifying, visualizing, constructing, and documenting the artifacts of the software systems, as well as for business modelling and other non-software systems. The UML represents a collection of the best engineering practices that have proven successful in modelling of large and complex systems.

3.1.1 Use Case Diagram

A use case is a set of scenarios that describing an interaction between a user and system. A use case diagram displays the relationship among the actors and use cases. The two main components of a use case diagram are use cases and actors. Here Fig 3.1 describes the use case diagram for showing the involvement of different actors for various steps involved in prediction. Actor along with researcher collects a data preprocess it, add feature selection algorithm and train a model to create an accurate model. This is model is then deployed in web, where the doctor, patient and common people use it.

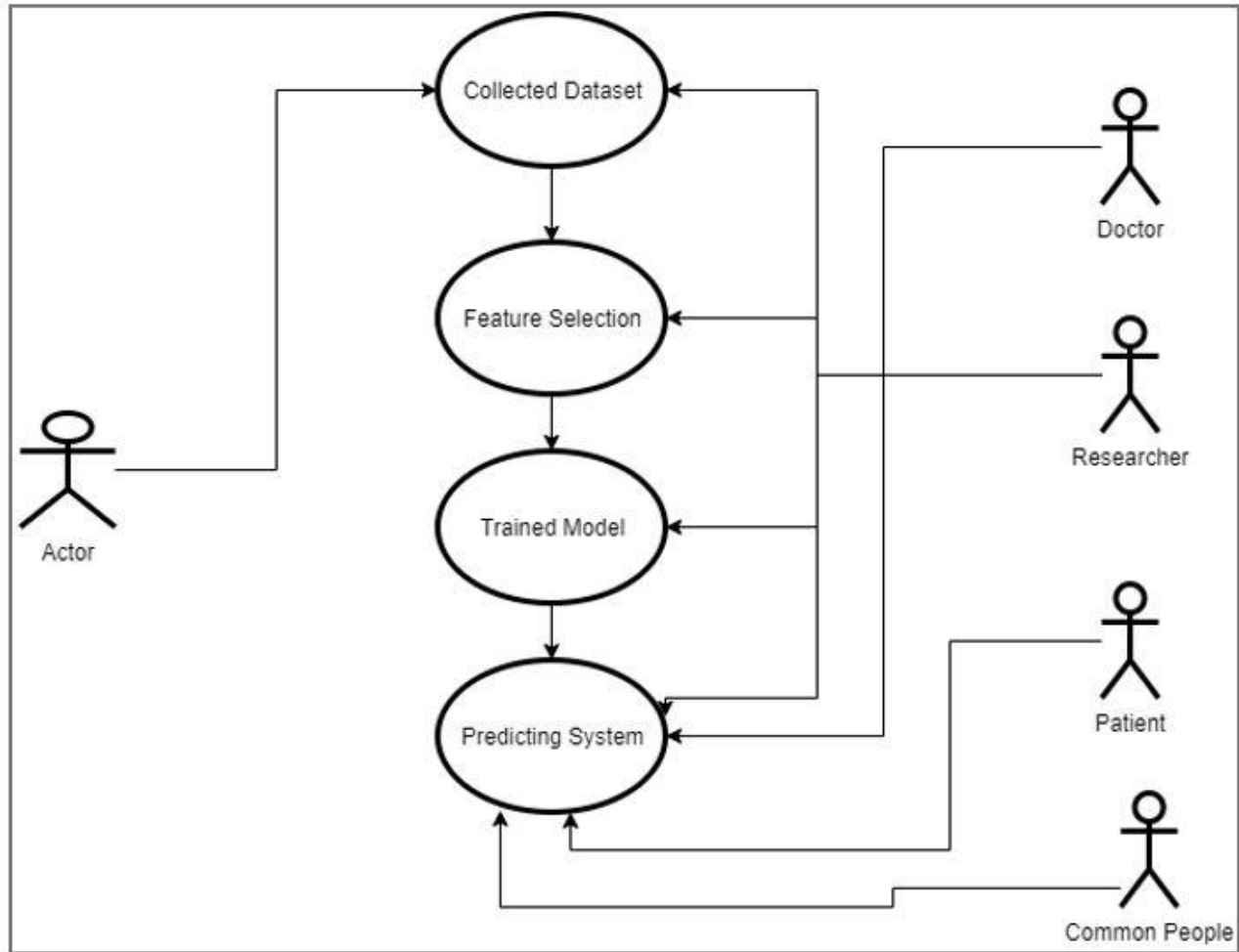


Fig 3.1 Use Case Diagram for liver prediction system

3.1.2 Class Diagram

Class diagrams are the most common diagrams used in UML. Class diagram consists of classes, interfaces, associations and collaborations. Class diagrams are basically representing the object-oriented view of the system which is static in nature. Active class is used in the class diagram to represent the concurrency of the system. This is the most widely used diagram at the time of system construction. In Fig 3.2 describes the class diagram for home page. Here we have set of attributes in homepage where the data are fetched and sends to Prediction System. System will process it and displays the output.

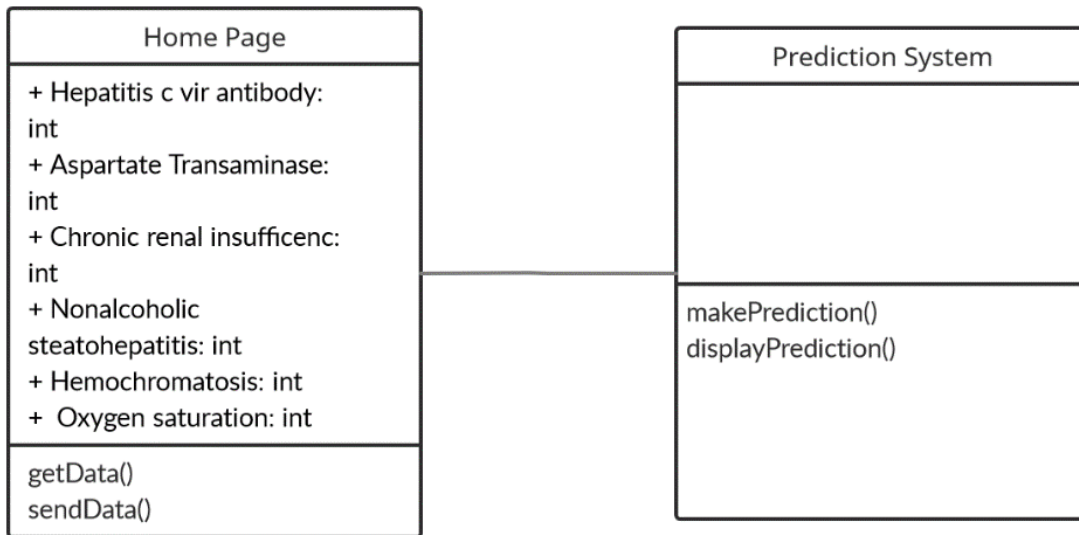


Fig 3.2 Class diagram for liver prediction syst

3.1.3 Sequence Diagram

A sequence diagram is an interaction diagram. Sequence diagram is used to visualize the sequence of calls in a system to perform a specific functionality. In Fig 3.3 represents the sequence of steps. Here the user will insert data in home page, data are fetched and sent to the prediction where prediction system will gives the appropriate predicted output to the user.

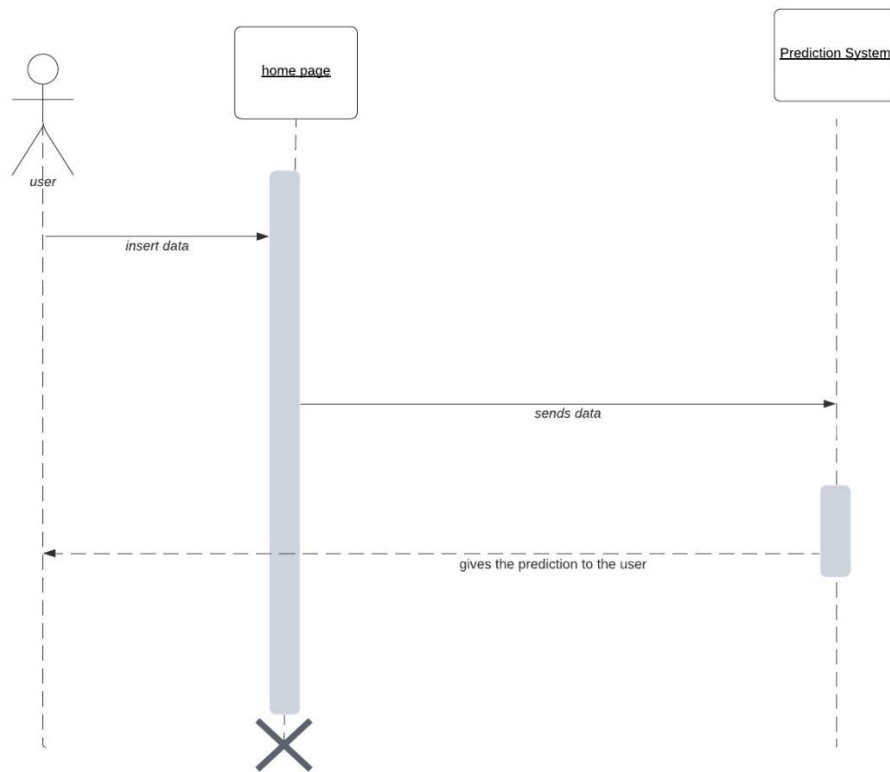


Fig 3.3 Sequence diagram for liver prediction system

3.1.4 Activity Diagram

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. This diagram is basically a flow chart to represent the flow from one activity to another activity. The basic purposes of activity diagrams are similar to other four diagrams. It captures the dynamic behavior of the system. Activity is the particular operation of the system. Fig 3.4 denotes the activities that are performed by the user for predicting the liver disease. Here the several tests are taken to get data for uploading, once it is received it is entered into the form on the home page, then the data is taken to predicting system which will give predicted output and this is shown to user through webpage.

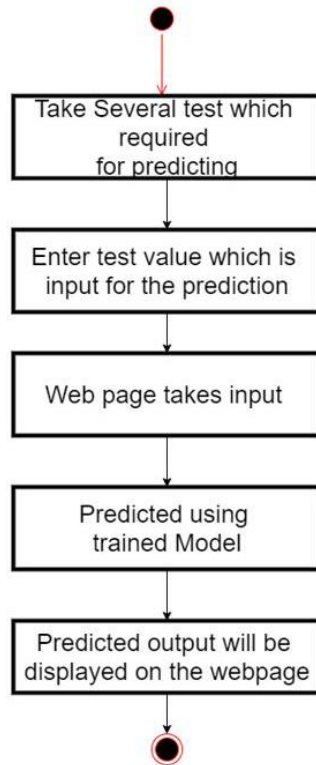


Fig 3.4 Activity diagram for liver prediction system

3.1.5 ER Diagram

An entity–relationship model (ER model for short) describes interrelated things of interest in a specific domain of knowledge. A basic ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between instances of those entity types. An ER model for the project is described in this Fig 3.5 which consist of all entities with their relationships. Here home page has different attributes as entries which are send to prediction system where prediction displays the output.

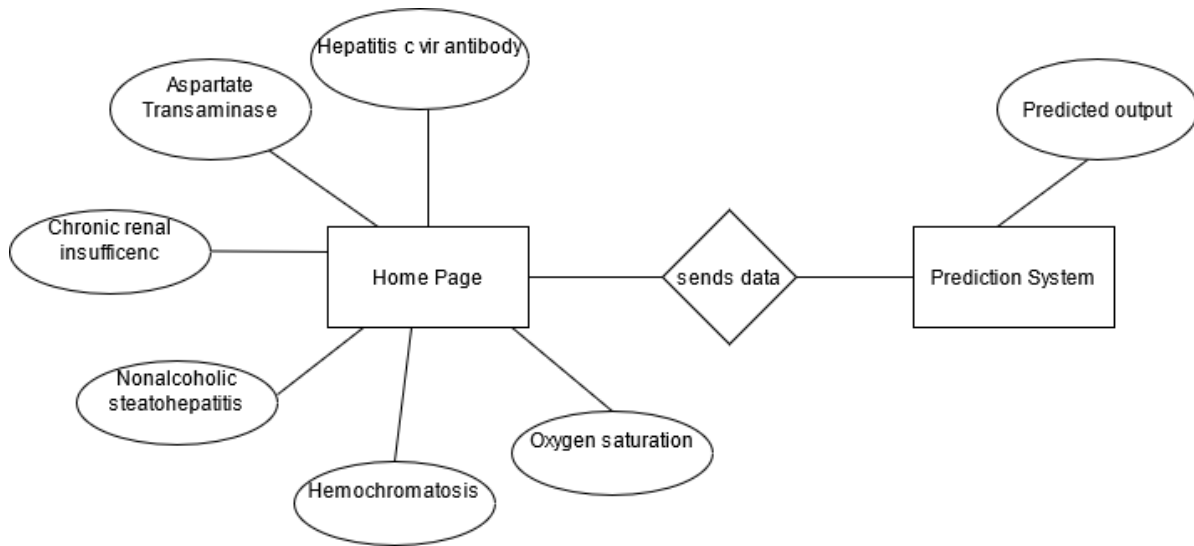


Fig 3.5 ER diagram for liver prediction system

3.1.6 System Architecture Diagram

This diagram represents the overall structure of the system. It also shows the behavior of the system in an abstracted way. The System architecture diagram for public transport management system which clearly describes the process of how the user and prediction system is shown in Fig 3.6 Researcher and developer work together in collecting dataset to create a best model for predicting liver disease. Once dataset is collected, it is preprocessed and feature selection is done with help of selectKbest and Rfe algorithm. Once feature are selected is used for developing model using KNN, Naïve Bayes and SVM. Out of this which have best accuracy is taken (SVM) and deployed in web application using Django framework. Then the user/doctor/patient uses this system to enter the data and get predicted output.

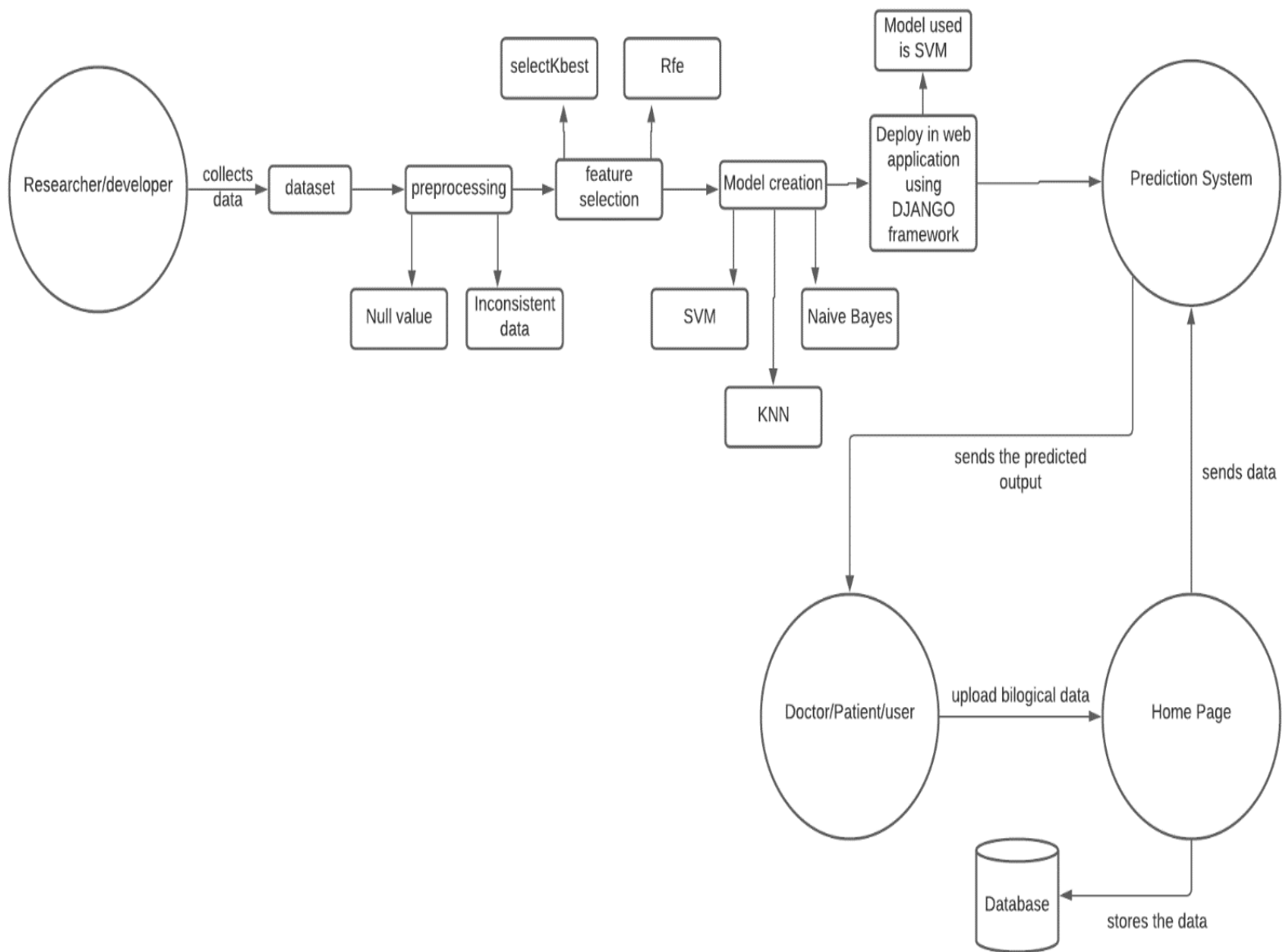


Fig 3.6 System architecture diagram for liver disease prediction

3.2 DATABASE DESIGN

3.2.1 Biological Data

S.NO.	FIELDS	DATATYPE
1	Hepatitis c vir antibody	NUMBER
2	Aspartate Transaminase	NUMBER
3	Chronic renal insufficenc	NUMBER
4	Nonalcoholic steatohepatitis	NUMBER
5	Hemochromatosis	NUMBER
6	Oxygen saturation	NUMBER

Table 3.1 Biological Data

CHAPTER 4

IMPLEMENTATION AND TESTING

4.1 MODULE DESCRIPTION

4.1.1 0 LEVEL

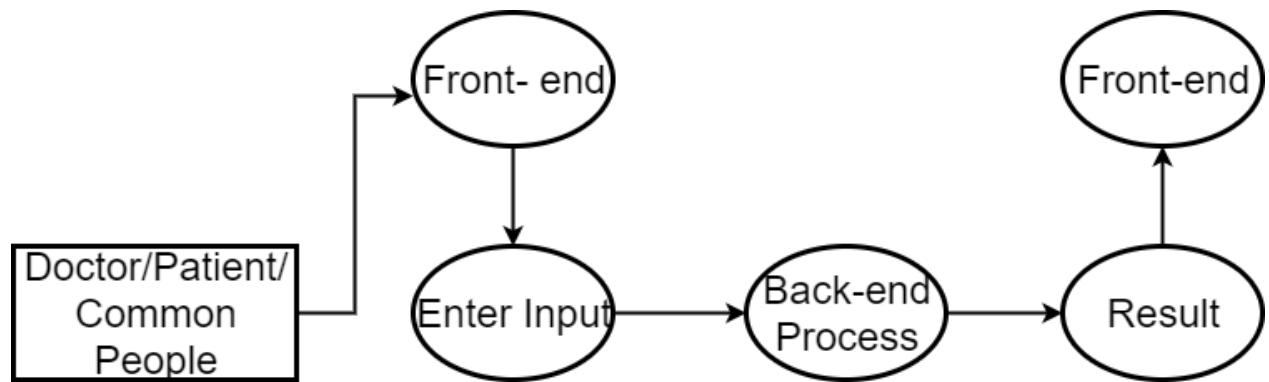


Fig 4.1 0 Level implementation of Liver disease prediction

0 level diagram is the more abstract level, which explains the flow of the data from the user to predicting system. In this flow, user enters the data and data is processed with predicting system and output is displayed.

4.1.2 1 LEVEL

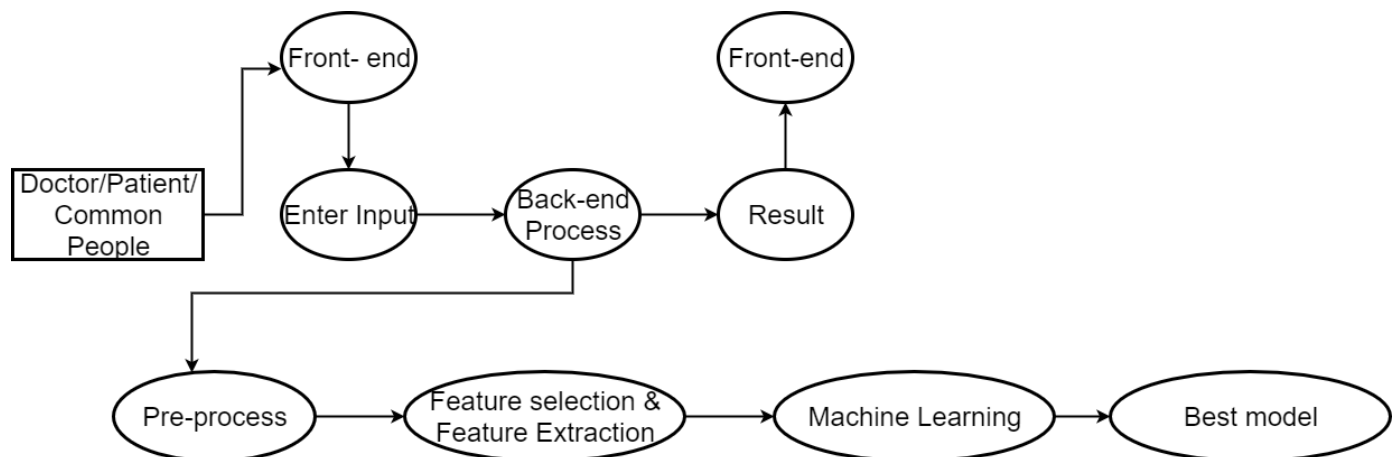


Fig 4.2 1 Level implementation of Liver disease prediction

1 level diagram is abstract level of back-end process, which explains what are the process involved in backend that are needed to get the best model and better accuracy.

4.1.3 2 LEVEL

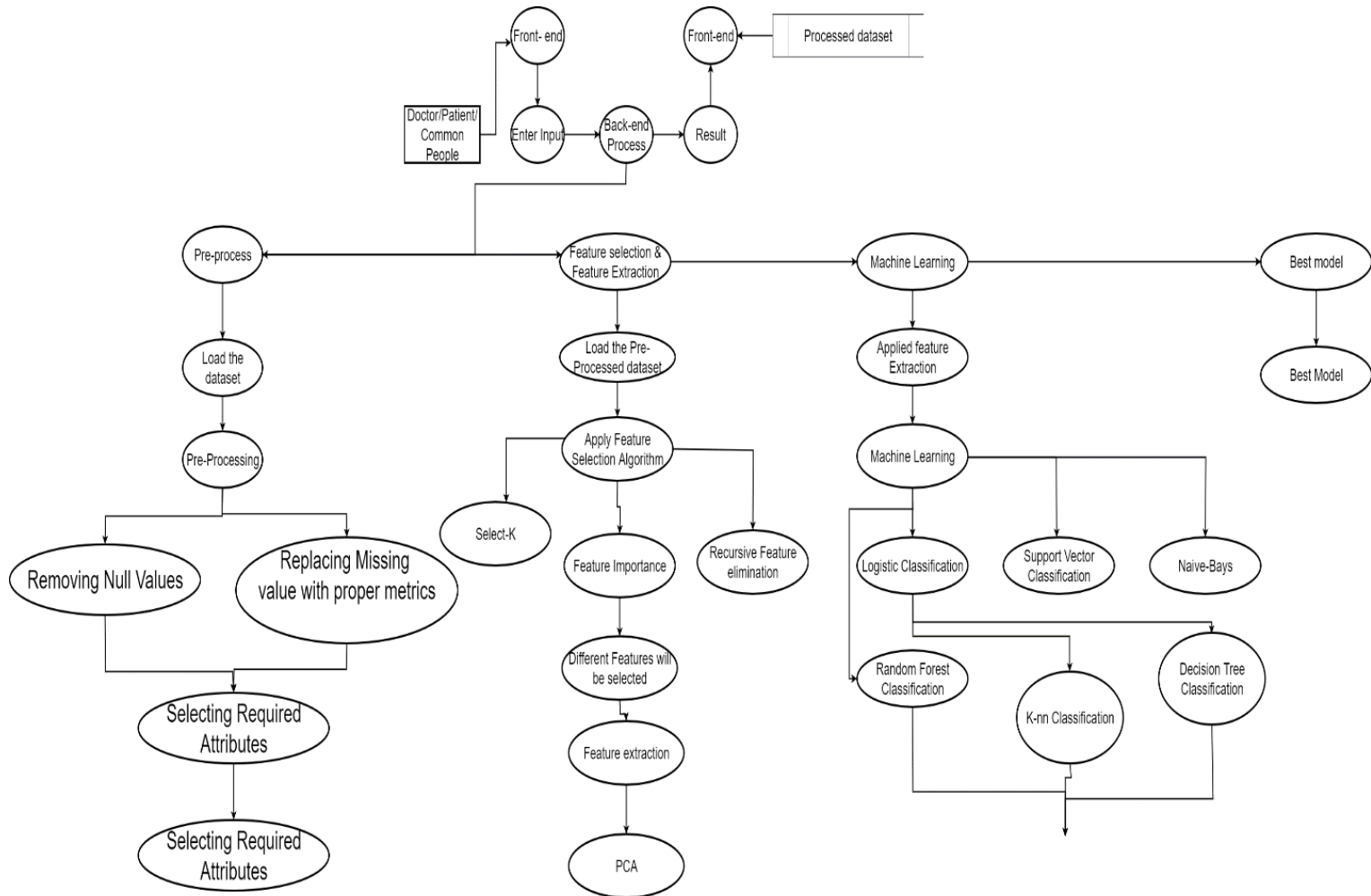


Fig 4.3 2 Level implementation of Liver disease prediction

2 level diagram describes the actual detailed representation of back-end process. It tells the preprocessing steps involved to remove inconsistent data and null valued data. It also explains the feature selection algorithm involved to select the best feature for the model creation i.e., selectKbest and RFE. And It tells which machine learning algorithm are involved to select the best model i.e., naïve bayes, SVM and KNN.

4.1.4 MODULE SPLIT-UP:

Module 1: Data Collection

To achieve the goal, Data Engineering is the first step. Data Engineering consists of two processes, they are Data Collection and Data Pre-processing. Data Collection will be collected with meaningful parameters like age, blood test and so on.

Module 2: Data Pre-processing

Collected data will be pre-processed which means encoding the categorical information in the data. Dropping unwanted parameters, scaling the parameter values to achieve normal distribution (Zero mean and Standard Deviation as one), handling missing values and so on.

Module 3: Feature Engineering

After the Data Engineering process, Feature Engineering will be done. Feature Engineering is an important step to predict our output. The advantage of Feature Engineering is minimizing the parameter. For example, if our whole dataset contains 10 parameters, after feature engineering only three-parameter enough to predict the output with high efficiency. Feature Engineering based on correlation, co-variance, co-linearity and etc. Feature Engineering has many algorithms to predict correct correlated parameters.

Module 5: Selection Of best Model

The final model will be optimized by selecting the best accuracy of the algorithm. This model helps to find the future prediction of employee resignation.

Module 6: Web Development

The trained model will be saved and loaded for web development. With the help of a built model and with a selected feature we can able to predict the employee

resignation. Web development will have an input variable of selected features, by submitting the answer of the selected feature, the prediction will be done.

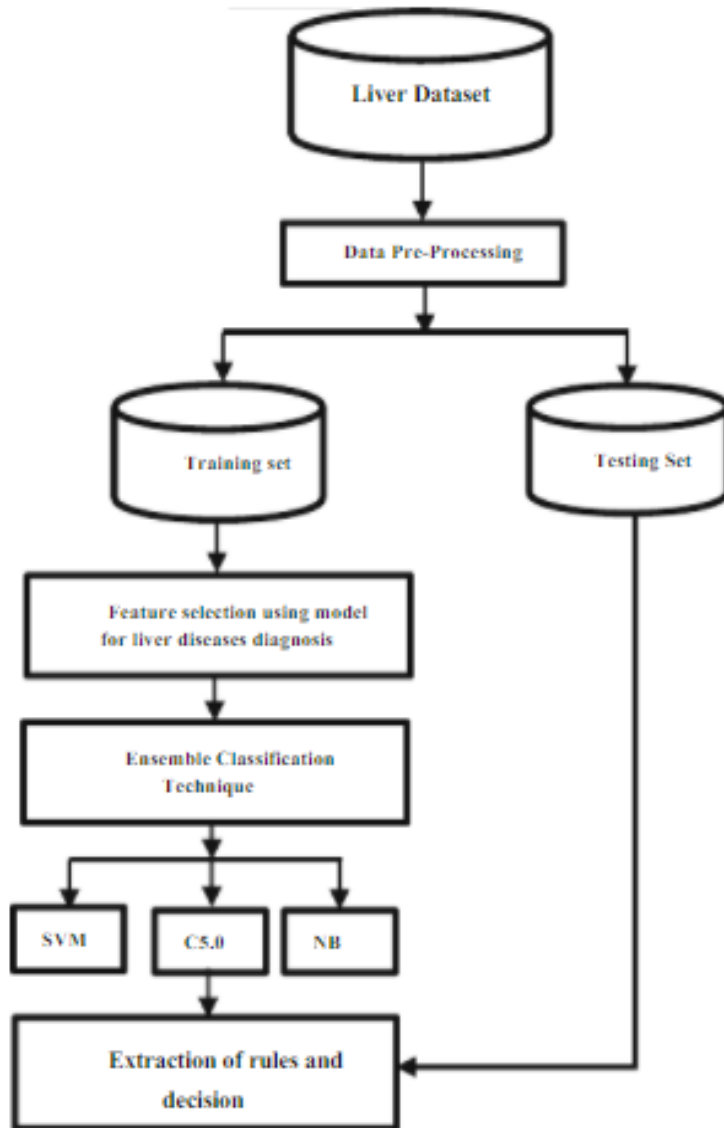


Fig 4.4 Creating a Model

4.2 IMPLEMENTATION

Data Pre-Processing: Raw Data

df - DataFrame

Index	gender	symptom	alcohol	b_surface	itis_b_eai	b_core_e	c_virus_e	cirrhosis	mic_cour	smoking	diabetes	obesity	ochroma	al_hyperte	renal_insu	numodef	plc_steat	hageal_v	lenomeg	l_hyperte	vein_thro	r_metast	ogical_ha	age	f_alcohol	c
0	1	0	1	0	0	0	0	1	0	1	1	nan	1	0	0	0	0	1	0	0	0	0	1	67	137	15
1	0	nan	0	0	0	0	1	1	nan	nan	1	0	0	1	0	0	0	1	0	0	0	0	1	62	0	nan
2	1	0	1	1	0	1	0	1	0	1	0	0	0	1	1	0	0	0	0	1	0	1	1	78	50	50
3	1	1	1	0	0	0	0	1	0	1	1	0	0	1	0	0	0	0	0	0	0	1	1	77	40	30
4	1	1	1	1	0	1	0	1	0	1	0	0	0	1	1	0	0	0	0	0	0	0	1	76	100	30
5	1	0	1	0	nan	0	0	1	0	nan	0	1	0	0	0	0	0	1	1	1	0	0	1	75	nan	nan
6	1	0	0	0	nan	1	1	1	0	0	1	0	nan	0	0	0	0	0	0	0	0	0	1	49	0	0
7	1	1	1	0	nan	0	0	1	0	1	1	nan	0	0	0	0	0	0	1	1	1	0	1	61	nan	20
8	1	1	1	0	0	0	0	1	0	1	1	0	0	1	0	0	0	nan	1	1	0	0	1	50	100	32
9	1	1	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	43	100	0
10	1	0	1	0	0	0	1	1	nan	nan	0	0	0	0	0	0	0	nan	1	1	0	0	1	41	nan	nan
11	1	0	1	0	0	0	1	1	1	0	0	0	0	1	0	0	0	nan	0	1	0	0	1	74	nan	0
12	1	0	1	0	0	0	0	1	0	1	1	0	0	1	0	0	nan	nan	1	1	1	0	0	66	nan	30
13	1	nan	0	0	0	0	1	1	nan	nan	0	0	0	0	0	0	0	0	0	0	0	0	1	56	0	nan
14	1	0	1	0	0	0	0	1	0	nan	1	0	0	1	0	0	nan	1	1	1	0	0	1	63	nan	nan
15	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	41	100	0
16	1	0	1	0	0	0	0	1	nan	1	1	nan	1	0	0	0	nan	nan	1	1	1	0	1	72	nan	nan
17	1	1	1	0	0	0	0	1	0	1	0	0	nan	0	0	0	nan	1	1	1	1	1	1	60	100	60
18	1	nan	1	0	0	0	0	1	nan	1	0	nan	0	1	1	0	0	1	0	0	0	0	1	64	200	78
19	1	1	1	0	0	0	0	1	nan	nan	0	0	0	1	0	0	0	1	1	1	1	0	1	75	500	nan
20	1	0	1	0	0	0	0	1	nan	1	0	1	0	0	0	0	0	1	1	1	0	0	0	71	200	60

Fig 4.5 Data Preprocessing

Data Pre-Processing: Null value count in each feature input

```
In [3]: print("Null values colunt for each column")
...:
...: data.isnull().sum(axis=0)
Null values colunt for each column
Out[3]:
gender                                0
symptoms                             18
alcohol                               0
hepatitis_b_surface_antigen          17
hepatitis_b_e_antigen                39
hepatitis_b_core_antibody            24
hepatitis_c_virus_antibody           9
cirrhosis                            0
endemic_countries                    39
smoking                              41
diabetes                             3
obesity                              10
hemochromatosis                      23
arterial_hypertension                 3
chronic_renal_insufficiency           2
human_immunodeficiency_virus         14
nonalcoholic_steatohepatitis         22
esophageal_varices                   52
splenomegaly                         15
portal_hypertension                   11
portal_vein_thrombosis                3
liver_metastasis                     4
radiological_hallmark                 2
age                                   0
grams_of_alcohol_per_day             48
packs_of_cigarets_per_year           53
performance_status                    0
encephalopathy_degree                1
ascites_degree                       2
international_normalised_ratio        4
```

```
Console 1/A
age                                0
grams_of_alcohol_per_day          48
packs_of_cigarets_per_year        53
performance_status                 0
encephalopathy_degree             1
ascites_degree                     2
international_normalised_ratio     4
alpha-fetoprotein                  8
haemoglobin                        3
mean_corpuscular_volume            3
leukocytes                         3
platelets                          3
albumin                            6
total_bilirubin                    5
alanine_transaminase               4
aspartate_transaminase             3
gamma_glutamyl_transferase         3
alkaline_phosphatase               3
total_proteins                     11
creatinine                         7
number_of_nodules                  2
major_dimension_of_nodule_cm       20
direct_bilirubin_mg/dL             44
iron                               79
oxygen_saturation_%                80
ferritin                           80
class_attribute                    0
dtype: int64
```

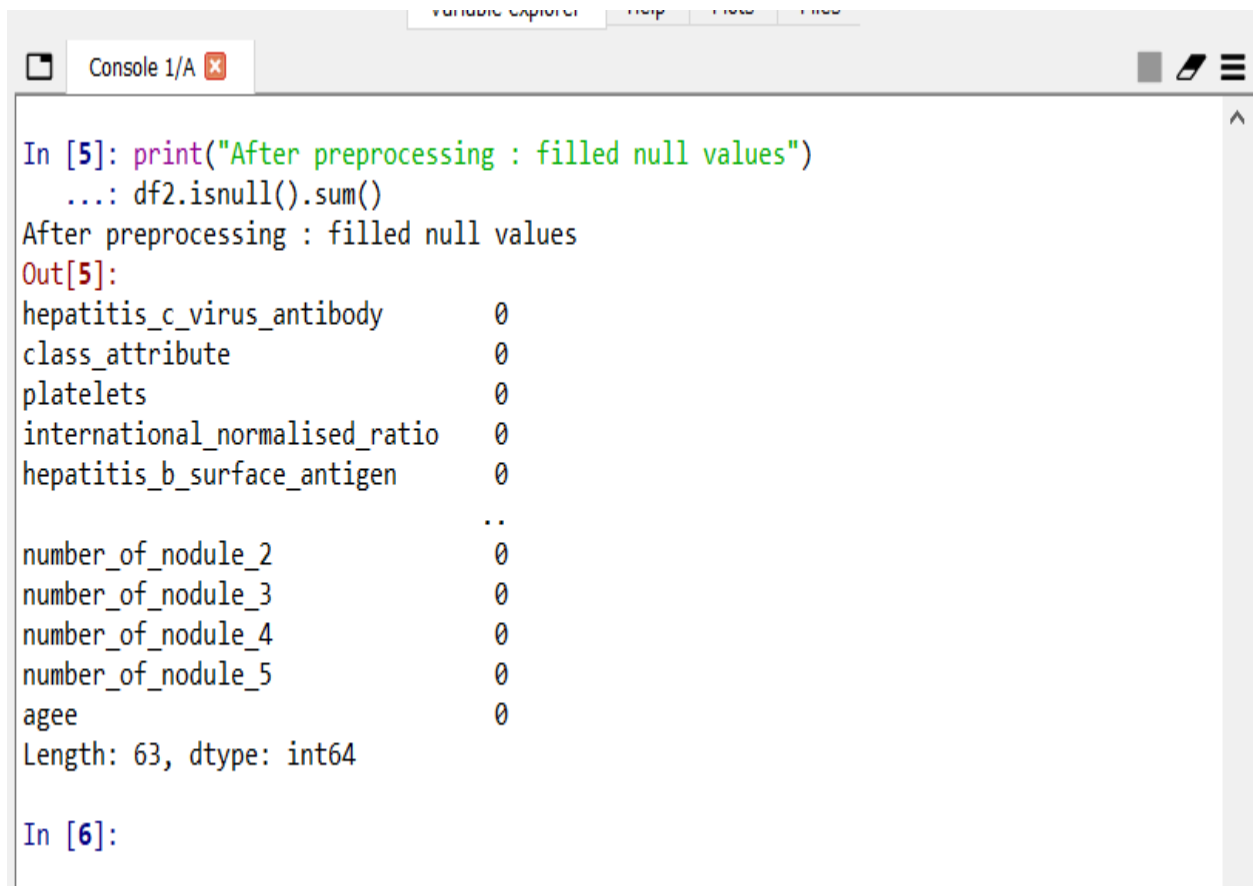
Fig 4.6 Null value count

Data Pre-Processing: Technique for each data type:

Nominal Datatype	Continuous Datatype	Ordinal Datatype	Integer
Using class output value finding out maximum used feature value	Using KLN algorithm missed values are filled.	Using class output value finding out maximum used feature value.	Using mean, missed values are filled.
-	-	Ordinal datatype was binarized which remove dummies variable.	-

Table 4.1 Data Preprocessing Technique for each datatype

Data Pre-Processing: After Pre-Processing



```
Console 1/A x
```

```
In [5]: print("After preprocessing : filled null values")
...: df2.isnull().sum()
After preprocessing : filled null values
Out[5]:
hepatitis_c_virus_antibody      0
class_attribute                 0
platelets                       0
international_normalised_ratio  0
hepatitis_b_surface_antigen     0
..
number_of_nodule_2              0
number_of_nodule_3              0
number_of_nodule_4              0
number_of_nodule_5              0
age                             0
Length: 63, dtype: int64

In [6]:
```

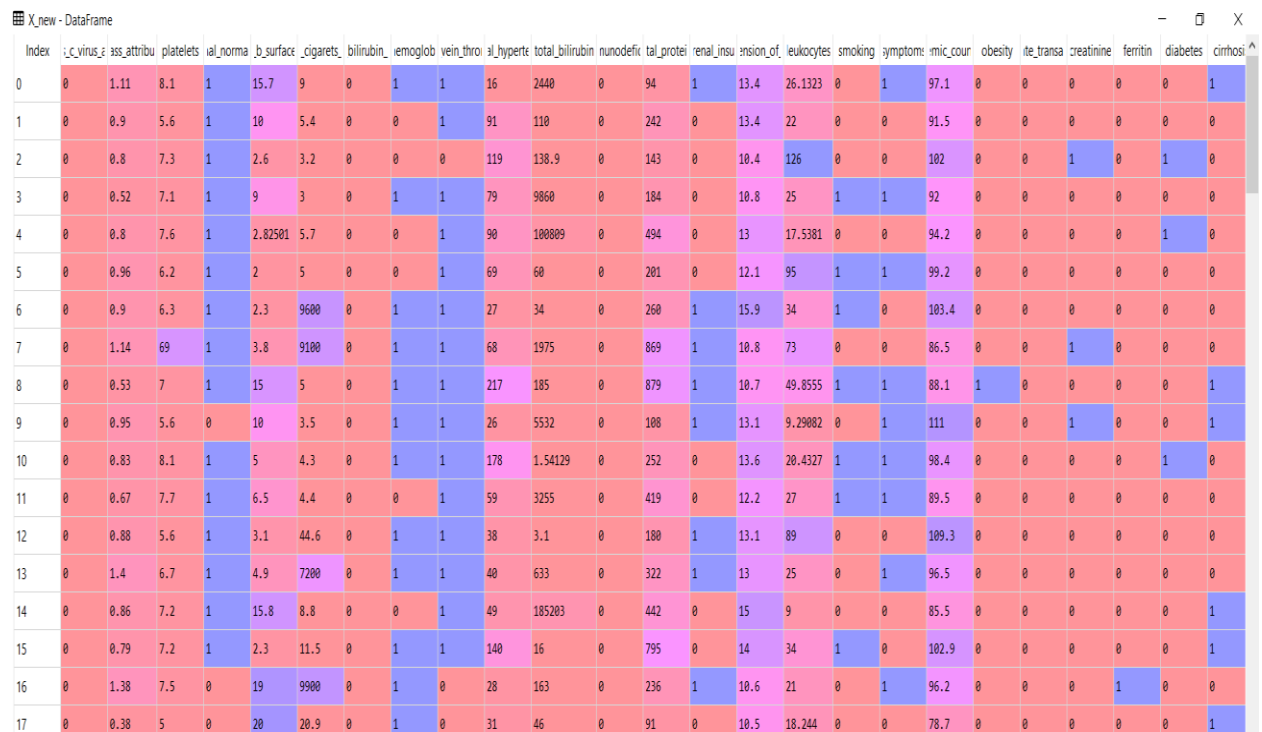
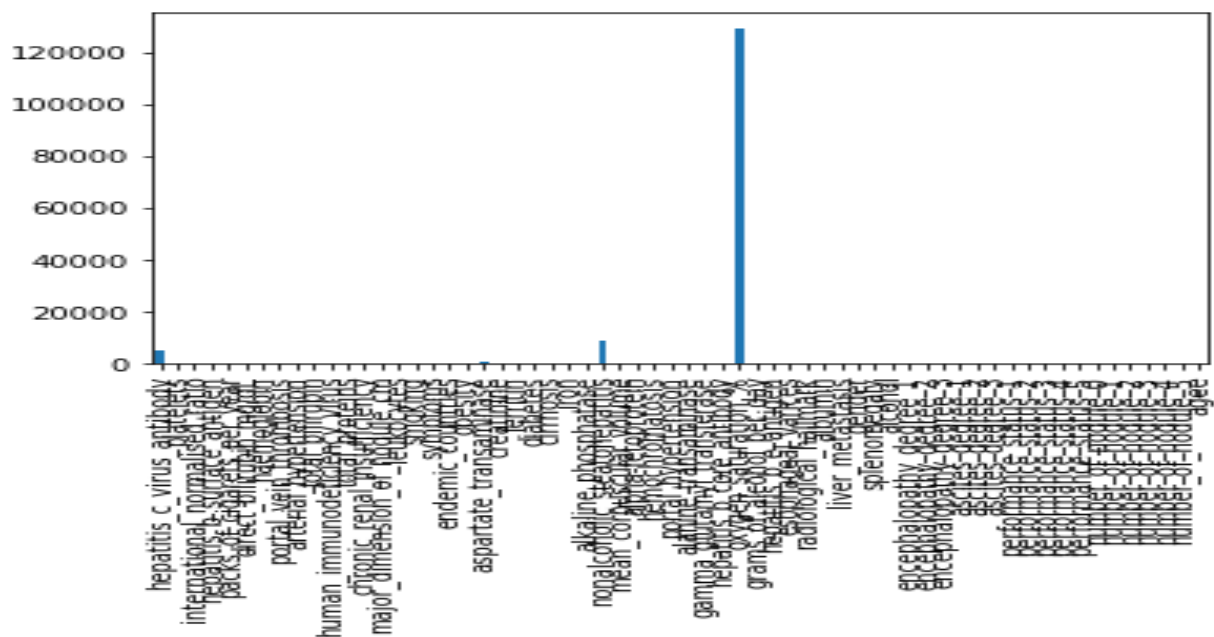


Fig 4.7 After Preprocessing

Feature Select:



Comparison Table:

	SelectKbest	Rfe
SVM	0.95	0.79
Naïve Bayes	0.69	0.80
KNN	0.66	0.56

Table 4.2 Comparison Table

Input UI:

← → ↺ 127.0.0.1:8000

Home Liver

Liver disease patient Survival

Please enter repective fields

Hepatitis c vir antibody*

Aspartate Transaminase*

Chronic renal insufficenc*

Nonalcoholic steatohepatitis*

Hemochromatosis*

Oxygen saturation*

upload

Fig 4.9 INPUT UI

Output UI:

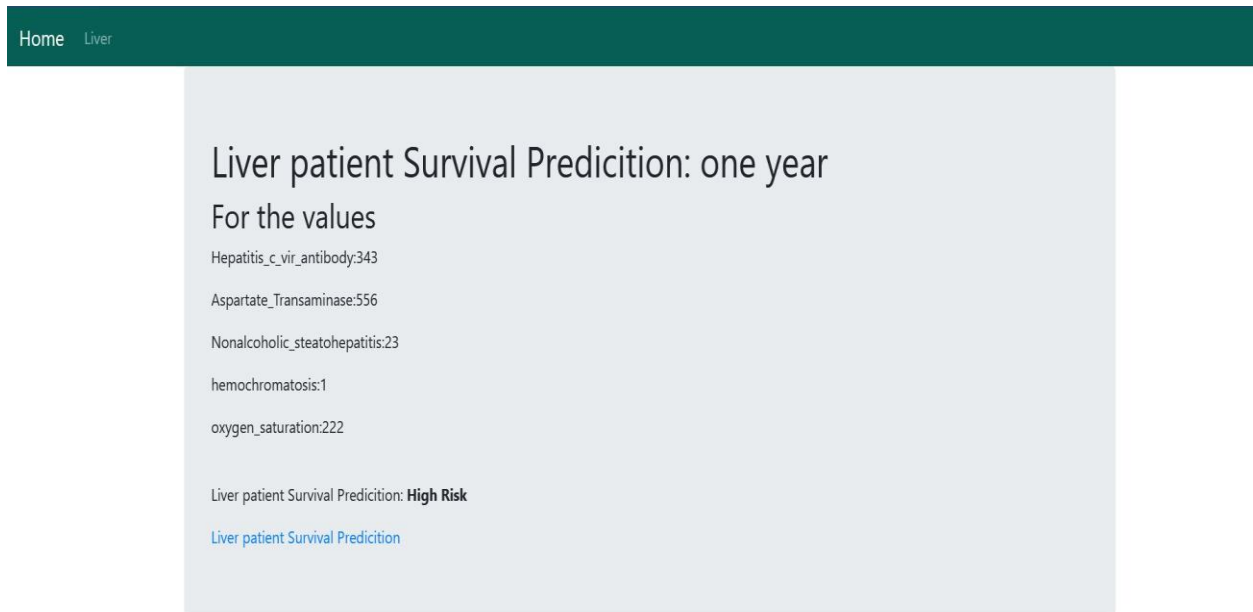


Fig 4.10 OUTPUT UI

4.3 TESTING

Since the error in the software can be injured at any stage. So, we have carry out the testing process at different levels during the development. The basic levels of testing are,

- Unit Testing
- Integration Testing
- Validation Testing
- Functional Testing
- Structural Testing

4.3.1 Unit Testing

Unit testing was used to test individual units in the system and ensure that they operate correctly. Alternate logic analysis and screen validations were tested in

this to ensure optimum efficiency in the system. The procedures and functions used and their association with data were tested.

4.3.2 Integration Testing

This testing process focuses on identifying the interfaces between components and their functionality. The bottom up approach was adopted during this testing. Low-level modules are integrated and combined as a cluster before testing. This allowed identifying any wrong linkages or parameters passing early in the development process as it just can be passed in the set of data and checked if the result returned is an accepted one.

4.3.3 Validation Testing

Software testing and validation is achieved through a series of block box tests that demonstrate conformity with requirements. A test procedure defines specific test cases that will be used to demonstrate conformity with requirements. Both, the plan and the procedure are designed to ensure that all functional requirements are achieved, documentation is correct and other requirements are met. After each validation test case has been conducted, one of the two possible conditions exists.

4.3.4 Functional Testing

Functional testing, also known as block box or closed box testing, is normally applied to HDL (High-Level Data Link) code that operates concurrently and concentrates on checking the interaction between modules, blocks or functional boundaries. The objective here is to ensure that ‘correct results’ are obtained when ‘good inputs’ are applied operates in a predictable manner. Functional testing can therefore be considered as concentrating on checking that the data paths operate correctly. The coverage measurements that fall into this category are toggle, triggering, and signal trace coverage.

4.3.5 Structural Testing

Structural testing, are known as white box or open box testing, is normally applied to sequential HDL (High-Level Data Link) code and concentrates on checking that all executable statements within each module have been exercised and the corresponding branches and paths through that module have been covered. If there is a section of HDL code that has never been exercised then there is a high possibility that it could contain an error that will remain undetected.

4.4 DRIVING TEST CASES

A test case is a set of conditions or variables under which a tester will determine if a requirement upon an application is partially or fully satisfied. The types of testing that are to be carried out on the system is as follows.

Test Case no	Description	Pre-condition	Pass/Fail	Expected Results
1	Check the homepage is available at http://127.0.0.1:8000/	Page should be available	Pass	Home page should appear
2	Check feature selected attribute is selected	Attributes should be present	Pass	All the attributes are present
3	Valid data should be entered	No data other than number should be entered	Pass	Alert should be displayed when wrong datatype is entered

4	Once upload button is clicked, data must to be send to prediction system	Prediction display must to be appear	Pass	Prediction display page should appear
5	Check whether correct output is shown	Output should be generated correctly	Pass	Output should be generated

Table 4.3 Driving Test Cases

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

The HCC affected person's risk factor was classified with Support Vector Machine.

This was achieved with feature selection method select-K parameter with chi square. The effective five features were selected from 50 features using feature selection method. The result achieved was 95% accuracy. The trained model with SVM for 5 feature input are able to predict the low risk or high risk. Advantage of using feature selection has eliminated the unwanted feature which may increase the blood test cost of the person.

In the proposed work, different classifiers were implemented on liver patient diseases dataset to predict liver diseases based on developed software. Dataset was processed and implemented using feature selection techniques. The results of the proposed work were compared using feature selection and without using feature selection techniques after the implementation of different classifiers in terms of execution time and accuracy.

The best result was achieved using Logistic Regression classifier with feature selection techniques and execution time of different classifiers was decreased after the implementation of feature selection technique. Finally, liver disease prediction Software (LDPS) is developed using concept of software engineering life cycle.

5.2 FUTURE WORK

The application developed is simple prototype to explain the basic functionalities of the upcoming application. In the upcoming release following features will be added

- **Prediction of liver disease can be added to hospital management system**
- **Image processing system can be added.**
- **Prescription can be suggested based on the risk.**

REFERENCES

1. Performance Assessment of Classification Algorithms on Early Detection of Liver Syndrome Rashid Naseem, Bilal Khan, M. A. Shah, Karzan Wakil, Atif Khan, Wael Alosaimi, M. I. Uddin, Badar Alouffi Published on 2020 in “National Library of medicine, USA”.
2. Diagnosis of Liver Disease using Machine Learning Models A. Sivasangari, Baddigam Jaya Krishna Reddy, Annamareddy Kiran, P. Ajitha Published 2020 on “IEEE xplore”
3. A.N.Arbainand, B.Y.P.Balakrishnan, “A comparison of data mining algorithms for liver disease prediction on imbalanced data, published in ” International Journal of Data Science and Analytics, vol. 1, on 2019.
4. Jagdeep Singha, Sachin Baggab, Ranjodh Kaur Software-based Prediction of Liver Disease with Feature Selection and Classification Techniques International Conference on Computational Intelligence and Data Science (ICCIDS 2019)
5. Binish Khan Piyush Kumar Shukla Manish Kumar Ahirwar Strategic Analysis in Prediction of Liver Disease Using Different Classification Algorithms INTERNATIONAL JOURNAL OF COMPUTER SCIENCES AND ENGINEERING · July 2019.
6. A Comparison of Data Mining Algorithms for Liver Disease Prediction on Imbalanced Data “Ain Najwa Arba, B. P. Balakrishnan Published 2019 published in International Journal of Data Science and Advanced Analytics (ISSN 2563-4429).
7. T. Choudhury, and A. Thakral. (2019), "Liver Disease Detection Due to Excessive Alcoholism Using Data Mining Techniques” published in IEEE International Conference on Advances in Computing and Communication Engineering (ICACCE), pp. 163-168.
8. I. Arshad, C. Dutta, T. Choudhury, and A. Thakral. (2019), "Liver Disease Detection Due to Excessive Alcoholism Using Data Mining Techniques." Published in IEEE xplore.

URL

1. Source: ieeexplore.ieee.org
2. Source: www.ResearchGate.com
3. Source: www.irjet.net

APPENDIX: Source Code

Pre-Processing 1:

```
import operator
import keras
from fancyimpute import KNN
from sklearn.preprocessing import LabelBinarizer
import math
from operator import itemgetter
import numpy as np
import pandas as pd
import seaborn as sns
#from sklearn import cross_validation
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler,
RobustScaler
from sklearn.metrics import roc_curve,
accuracy_score, roc_auc_score, classification_report,
r2_score, make_scorer, roc_curve, auc
from sklearn.model_selection import cross_validate,
train_test_split, cross_val_score, StratifiedKFold,
KFold, cross_val_predict
from sklearn.linear_model import LogisticRegression
```

```
columns = [
    # nominal
    'gender', #0-1
    'symptoms', #0-1
    'alcohol', #0-1
    'hepatitis b surface antigen', #0-1
    'hepatitis b e antigen', #0-1
    'hepatitis b core antibody', #0-1
    'hepatitis c virus antibody', #0-1
    'cirrhosis', #0-1
    'endemic countries', #0-1
    'smoking', #0-1
    'diabetes', #0-1
    'obesity', #0-1
    'hemochromatosis', #0-1
    'arterial hypertension', #0-1
    'chronic renal insufficiency', #0-1
    'human immunodeficiency virus', #0-1
    'nonalcoholic steatohepatitis', #0-1
    'esophageal varices', #0-1
    'splenomegaly', #0-1
    'portal hypertension', #0-1
    'portal vein thrombosis', #0-1
    'liver metastasis', #0-1
    'radiological hallmark', #0-1
    # integer
    'age', # age at diagnosis
    # continuous
    'grams of alcohol per day',
    'packs of cigs per year',
```

```
# ordinal
'performance status',
'encephalopathy degree',
'ascites degree',

# continuous
'international normalised ratio',
'alpha-fetoprotein',
'haemoglobin',
'mean corpuscular volume',
'leukocytes',
'platelets',
'albumin',
'total bilirubin',
'alanine transaminase',
'aspartate transaminase',
'gamma glutamyl transferase',
'alkaline phosphatase',
'total proteins',
'creatinine',
# integer
'number of nodules',
# continuous
'major dimension of nodule cm',
'direct bilirubin mg/dL',
'iron',
'oxygen saturation %',
'ferritin',
#nominal
'class attribute', #0-1
```

```
]
columns = list([x.replace(' ', '_').strip() for x in
columns])
df = pd.read_csv('hcc-data.csv', names=columns,
header=None, na_values=['?'])
data = df.copy()
print("Null values colunt for each column")
data.isnull().sum(axis=0)
def prepare_missing_values_for_nans(df=None,
columns=None):
    """
    Looking for the most frequent value for both
    decision classes outputs - 0,1.
    """
    to_update_nans_dict = {}
    if columns:
        for decision_class in [0, 1]:
            for column in columns:
                vals = df[df.class_attribute ==
decision_class][column].value_counts()
to_update_nans_dict['{decision_class}_{column}'].for
mat(
                    decision_class=decision_class,
                    column=column
                ) = vals.idxmax()
```

```

        return to_update_nans_dict
def replace_missing_values(df=None,
columns=None, to_update_nans_dict=None):
    """
    Replacing NaN with the most frequent values for
    both decision classes outputs - 0,1.
    """
    df_list = []
    if columns:
        for decision_class in [0, 1]:
            _df = df[df.class_attribute ==
decision_class].reset_index(drop=True)
            for column in columns:
                _df[column] = _df[column].fillna(
to_update_nans_dict['{}_{}'.format(decision_class,
column)])
            df_list.append(_df)
    return df_list
nominal_indexes = [
    1, 3, 4, 5,
    6, 8, 9, 10,
    11, 12, 13,
    14, 15, 16,
    17, 18, 19,
    20, 21, 22
]
nominal_columns_to_discretize =
list(itemgetter(*nominal_indexes)(columns))
cons = data.loc[:, :]
cons['null_values'] = cons.isnull().sum(axis=1)
data2 = data.drop(columns=['null_values'])
nominal_dict =
prepare_missing_values_for_nans(df=data2,
columns=nominal_columns_to_discretize)
missing_nominal_values_list =
replace_missing_values(
    df=data2,
    columns=nominal_columns_to_discretize,
    to_update_nans_dict=nominal_dict
)
data2 =
pd.concat(missing_nominal_values_list).reset_index(
drop=True)
continuous_indexes = [
    24,25,29,30,
    31,32,33,34,
    35,36,37,38,
    39,40,41,42,
    44,45,46,47,
    48]
continuous_columns_to_discretize = list(
    itemgetter(*continuous_indexes)(columns)
)

```

```

continuous_data =
data2[continuous_columns_to_discretize].as_matrix(
X_filled_knn =
KNN(k=3).fit_transform(continuous_data)
data2[continuous_columns_to_discretize] =
X_filled_knn
X_filled_knn.shape
integer_columns = ['age', 'number_of_nodules']
# prepare missing integer values
integer_dict = prepare_missing_values_for_nans(
    df=data2,
    columns=integer_columns
)
missing_integer_values_list =
replace_missing_values(
    df=data2,
    columns=integer_columns,
    to_update_nans_dict=integer_dict
)
data2 =
pd.concat(missing_integer_values_list).reset_index(
drop=True)
data2['ascites_degree'].value_counts()
ordinal_columns = ['encephalopathy_degree',
'ascites_degree', 'performance_status']
ordinal_dict = prepare_missing_values_for_nans(
    df=data2,
    columns=ordinal_columns
)
missing_ordinal_values_list =
replace_missing_values(
    df=data2,
    columns=ordinal_columns,
    to_update_nans_dict=ordinal_dict
)
data2 =
pd.concat(missing_ordinal_values_list).reset_index(
drop=True)
data2[data2.isnull().any(axis=1)]
ordinal_columns
binarized_data = []
for c in ordinal_columns:
    lb = LabelBinarizer()
    lb.fit(data2[c].values)
    binarized = lb.transform(data2[c].values)
    binarized_data.append(binarized)
binarized_ordinal_matrix_data =
pd.DataFrame(np.hstack(binarized_data))
list(set(data2.number_of_nodules.values))
lb = LabelBinarizer()
lb.fit(data2.number_of_nodules.values)

```

```

binarized_number_of_nodules =
pd.DataFrame(lb.transform(data2.number_of_nodule
s.values))
data2['age_'] = data2.age.apply(lambda x: x /
data2.age.max())
data2['age_'].head(10)
age_ = data2.age_.values.reshape(-1,1)
to_drop_columns = [
    'age',
    'encephalopathy_degree',
    'ascites_degree',
    'performance_status',
    'number_of_nodules'
]
columns_set = set(columns)
columns_ =
list(columns_set.difference(to_drop_columns))
len(columns)
#len(_columns)
data2.to_csv("Preprocessed_HCC.csv",index=False)
X = pd.DataFrame(data2[columns_].as_matrix())
y = pd.DataFrame(data2.class_attribute.values)
binary_columns=['encephalopathy_degree_1','enceph
alopathy_degree_2','encephalopathy_degree_3',
'ascites_degree_1','ascites_degree_2','ascites_degree_
3','performance_status_1',
'performance_status_2','performance_status_3','perfor
mance_status_4','performance_status_5']
nodules_columns=['number_of_nodule_0','number_o
f_nodule_1','number_of_nodule_2',
'number_of_nodule_3','number_of_nodule_4','numbe
r_of_nodule_5']
X.columns=columns_
binarized_ordinal_matrix_data.columns=binary_colu
mns
binarized_number_of_nodules.columns=nodules_cou
lmns
cc=[columns_,binary_columns,binarized_number_of
_nodules,age_]
ccc=['hepatitis_c_virus_antibody',
'class_attribute',
'platelets',
'international_normalised_ratio',
'hepatitis_b_surface_antigen',
'packs_of_cigaretts_per_year',
'direct_bilirubin_mg/dL',
'haemoglobin',
'portal_vein_thrombosis',
'arterial_hypertension',
'total_bilirubin',
'human_immunodeficiency_virus',
'total_proteins',
'chronic_renal_insufficiency',
'major_dimension_of_nodule_cm',
'leukocytes',
'smoking',

```

```

'symptoms',
'endemic_countries',
'obesity',
'aspartate_transaminase',
'creatinine',
'ferritin',
'diabetes',
'cirrhosis',
'iron',
'alkaline_phosphatase',
'nonalcoholic_steatohepatitis',
'mean_corpuscular_volume',
'alpha-fetoprotein',
'hemochromatosis',
'portal_hypertension',
'alanine_transaminase',
'gamma_glutamyl_transferase',
'hepatitis_b_core_antibody',
'oxygen_saturation_%',
'grams_of_alcohol_per_day',
'hepatitis_b_e_antigen',
'esophageal_varices',
'radiological_hallmark',
'albumin',
'liver_metastasis',
'gender',
'splenomegaly',
'alcohol','encephalopathy_degree_1','encephalopathy_
degree_2','encephalopathy_degree_3',
'ascites_degree_1','ascites_degree_2','ascites_degree_
3','performance_status_1',
'performance_status_2','performance_status_3','perfor
mance_status_4','performance_status_5',
'number_of_nodule_0','number_of_nodule_1','numbe
r_of_nodule_2',
'number_of_nodule_3','number_of_nodule_4','numbe
r_of_nodule_5','agee']
X_new = pd.DataFrame(np.hstack((X,
binarized_ordinal_matrix_data,binarized_number_of
_nodules, age_)))
#X_new = pd.DataFrame([X,
binarized_ordinal_matrix_data,binarized_number_of
_nodules, age_])
X_new.columns=ccc
df2=X_new
X_new.to_csv("finalpre.csv",index=False)
print("After preprocessing : filled null values")
df2.isnull().sum()
X_new=X_new.drop('class_attribute', 1)
#y=df2['class_attribute']
X_new.to_csv("finalpre.csv",index=False)
X_new.shape
std_scaler = StandardScaler() #StandardScaler() #
RobustScaler
X_new = std_scaler.fit_transform(X_new)

```



```

X_train, X_test, y_train, y_test = train_test_split(
    X_new,
    y,
    random_state=42,
    test_size=0.20
)
log_reg = LogisticRegression(
    solver='lbfgs',
    random_state=42,
    C=0.1,
    multi_class='ovr',
    penalty='l2',
)
log_reg.fit(X_train, y_train)
log_reg_predict = log_reg.predict(X_test)
log_reg.score(X_test, y_test)
preds = log_reg.predict(X_test)
print("\nLogistic Regression Accuracy:
{:.2f}%'.format(accuracy_score(y_test,
log_reg_predict) * 100))
print('Logistic Regression AUC:
{:.2f}%'.format(roc_auc_score(y_test,
log_reg_predict) * 100))
print('Logistic Regression Classification report:\n\n',
classification_report(y_test, log_reg_predict))
kfold = StratifiedKFold(
    n_splits=3,
    shuffle=True,
    random_state=42
)
predicted = cross_val_predict(
    log_reg,
    X_new,
    y,
    cv=kfold
)
scores = cross_val_score(
    log_reg,
    X_new,
    y,
    cv=kfold,
    scoring='f1'
)
print('Cross-validated scores: { }\n'.format(scores))
print(classification_report(y, predicted))
print("LogisticRegression: F1 after 5-fold cross-
validation: {:.2f}% (+/- {:.2f}%)".format(
    scores.mean() * 100,
    scores.std() * 2
))

```

Pre-Processing 2:

```

import pandas as pd
dataset=pd.read_csv("finalpree.csv",index_col=None
)
#dataset=dataset.drop(columns=['age_'])
"""nominal_indexes = [
    0,1,2, 3, 4, 5,
    6, 8, 9, 7,10,
    11, 12, 13,
    14, 15, 16,
    17, 18, 19,
    20, 21, 22
]
dataset.columns[1]
for i in nominal_indexes:
    dictt={ 1:'yes',0:'no'}
    col=dataset.columns[i]
    print(dataset.columns[i])
    dataset[col]=dataset[col].replace(dictt)"""
dataset1 = pd.get_dummies(dataset, drop_first=True)
import pandas as pd
#dataset1=pd.read_csv("prep.csv",index_col=None)
df2=dataset1
import warnings
warnings.filterwarnings('always')
from sklearn.tree import export_graphviz #plot tree
from sklearn.metrics import roc_curve, auc #for
model evaluation
from sklearn.metrics import classification_report #for
model evaluation
from sklearn.metrics import confusion_matrix #for
model evaluation
from sklearn.model_selection import train_test_split
#X_train, X_test, y_train, y_test =
train_test_split(df2.drop('classification_yes', 1),
df2['classification_yes'], test_size = .2,
random_state=10)
import time
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
import pickle
import matplotlib.pyplot as plt
from sklearn.svm import SVC
#df2 = pd.get_dummies(df2, drop_first=True)
def selectkbest(indep_X,dep_Y):
    test = SelectKBest(score_func=chi2, k=3)
    fit1= test.fit(indep_X,dep_Y)

```

```

# summarize scores
features = indep_X.columns.values.tolist()
np.set_printoptions(precision=2)
print(features)
print(fit1.scores_)
#plt.figure(figsize=(12,3))
#plt.bar(fit1.scores_,height=0.6)
feature_series =
pd.Series(data=fit1.scores_,index=features)
feature_series.plot.bar()
selectk_features = fit1.transform(indep_X)
return selectk_features
def rfeFeature(indep_X,dep_Y):
#model=SVR(kernel="linear")
model = LogisticRegression(solver='lbfgs')
rfe = RFE(model,3)
fit3 = rfe.fit(indep_X, dep_Y)
rfe_feature=fit3.transform(indep_X)
features = indep_X.columns.values.tolist()
#feature_series =
pd.Series(data=rfe_feature,index=features)
#feature_series.plot.bar()
return rfe_feature
def pca(features,dep_Y):
pca = PCA(n_components=3)
fit2 = pca.fit(features)
pca_feature=fit2.transform(features)
return pca_feature
def svm(features,indep_X,dep_Y):
X_train, X_test, y_train, y_test =
train_test_split(features, dep_Y, test_size = 0.25,
random_state = 0)
#X_train, X_test, y_train, y_test =
train_test_split(rfe_feature, dep_Y, test_size = 0.25,
random_state = 0)
#X_train, X_test, y_train, y_test =
train_test_split(pca_feature, dep_Y, test_size = 0.25,
random_state = 0)
#X_train, X_test, y_train, y_test =
train_test_split(feature_import, dep_Y, test_size =
0.25, random_state = 0)
#X_train, X_test, y_train, y_test =
train_test_split(indep_X,dep_Y, test_size = 0.25,
random_state = 0)
#Feature Scaling
#from sklearn.preprocessing import
StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
# Fitting K-NN to the Training set
from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state =
0)
classifier.fit(X_train, y_train)

```

```

# Predicting the Test set results
y_pred = classifier.predict(X_test)
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
#from sklearn.metrics import confusion_matrix
#cm = confusion_matrix(y_test, y_pred)
Accuracy=accuracy_score(y_test, y_pred )
report=classification_report(y_test, y_pred)
return
classifier,Accuracy,report,X_test,y_test,cm
def naives(features,indep_X,dep_Y):
X_train, X_test, y_train, y_test =
train_test_split(features, dep_Y, test_size = 0.25,
random_state = 0)
#X_train, X_test, y_train, y_test =
train_test_split(rfe_feature, dep_Y, test_size = 0.25,
random_state = 0)
#X_train, X_test, y_train, y_test =
train_test_split(pca_feature, dep_Y, test_size = 0.25,
random_state = 0)
#X_train, X_test, y_train, y_test =
train_test_split(feature_import, dep_Y, test_size =
0.25, random_state = 0)
#X_train, X_test, y_train, y_test =
train_test_split(indep_X,dep_Y, test_size = 0.25,
random_state = 0)
#Feature Scaling
#from sklearn.preprocessing import
StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
# Fitting K-NN to the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
# Predicting the Test set results
y_pred = classifier.predict(X_test)
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
#from sklearn.metrics import confusion_matrix
#cm = confusion_matrix(y_test, y_pred)
Accuracy=accuracy_score(y_test, y_pred)
report=classification_report(y_test, y_pred)
return
classifier,Accuracy,report,X_test,y_test,cm
def Decision(features,indep_X,dep_Y):
X_train, X_test, y_train, y_test =
train_test_split(features, dep_Y, test_size = 0.25,
random_state = 0)

```

```

#X_train, X_test, y_train, y_test =
train_test_split(rfe_feature, dep_Y, test_size = 0.25,
random_state = 0)
#X_train, X_test, y_train, y_test =
train_test_split(pca_feature, dep_Y, test_size = 0.25,
random_state = 0)
#X_train, X_test, y_train, y_test =
train_test_split(feature_import, dep_Y, test_size =
0.25, random_state = 0)
#X_train, X_test, y_train, y_test =
train_test_split(indep_X, dep_Y, test_size = 0.25,
random_state = 0)
#Feature Scaling
#from sklearn.preprocessing import
StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
# Fitting K-NN to the Training set
from sklearn.linear_model import
LogisticRegression
classifier = LogisticRegression(random_state =
0)
classifier.fit(X_train, y_train)
# Predicting the Test set results
y_pred = classifier.predict(X_test)
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
#from sklearn.metrics import confusion_matrix
#cm = confusion_matrix(y_test, y_pred)
Accuracy=accuracy_score(y_test, y_pred )
report=classification_report(y_test, y_pred)
return
classifier, Accuracy, report, X_test, y_test, cm
def knn(features, indep_X, dep_Y):
    X_train, X_test, y_train, y_test =
train_test_split(features, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(rfe_feature, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(pca_feature, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(feature_import, dep_Y, test_size =
0.25, random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(indep_X, dep_Y, test_size = 0.25,
random_state = 0)
    #Feature Scaling
    #from sklearn.preprocessing import
StandardScaler
    sc = StandardScaler()
    X_train = sc.fit_transform(X_train)
    X_test = sc.transform(X_test)
    # Fitting K-NN to the Training set
    from sklearn.ensemble import
RandomForestClassifier
    classifier =
RandomForestClassifier(n_estimators = 10, criterion
= 'entropy', random_state = 0)
    classifier.fit(X_train, y_train)
    # Predicting the Test set results
    y_pred = classifier.predict(X_test)
    # Making the Confusion Matrix
    from sklearn.metrics import confusion_matrix
    cm = confusion_matrix(y_test, y_pred)

```

```

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
# Fitting K-NN to the Training set
from sklearn.neighbors import
KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors =
5, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)
# Predicting the Test set results
y_pred = classifier.predict(X_test)
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
#from sklearn.metrics import confusion_matrix
#cm = confusion_matrix(y_test, y_pred)
Accuracy=accuracy_score(y_test, y_pred )
report=classification_report(y_test, y_pred)
return
classifier, Accuracy, report, X_test, y_test, cm
def random(features, indep_X, dep_Y):
    X_train, X_test, y_train, y_test =
train_test_split(features, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(rfe_feature, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(pca_feature, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(feature_import, dep_Y, test_size =
0.25, random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(indep_X, dep_Y, test_size = 0.25,
random_state = 0)
    #Feature Scaling
    #from sklearn.preprocessing import
StandardScaler
    sc = StandardScaler()
    X_train = sc.fit_transform(X_train)
    X_test = sc.transform(X_test)
    # Fitting K-NN to the Training set
    from sklearn.ensemble import
RandomForestClassifier
    classifier =
RandomForestClassifier(n_estimators = 10, criterion
= 'entropy', random_state = 0)
    classifier.fit(X_train, y_train)
    # Predicting the Test set results
    y_pred = classifier.predict(X_test)
    # Making the Confusion Matrix
    from sklearn.metrics import confusion_matrix
    cm = confusion_matrix(y_test, y_pred)

```

```

from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
#from sklearn.metrics import confusion_matrix
#cm = confusion_matrix(y_test, y_pred)
Accuracy=accuracy_score(y_test, y_pred )
report=classification_report(y_test, y_pred)
return
classifier,Accuracy,report,X_test,y_test,cm
def logistics(features,indep_X,dep_Y):
    X_train, X_test, y_train, y_test =
train_test_split(features, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(rfe_feature, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(pca_feature, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(feature_import, dep_Y, test_size =
0.25, random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(indep_X,dep_Y, test_size = 0.25,
random_state = 0)
    #Feature Scaling
    #from sklearn.preprocessing import
StandardScaler
    sc = StandardScaler()
    X_train = sc.fit_transform(X_train)
    X_test = sc.transform(X_test)
    # Fitting K-NN to the Training set
    from sklearn.linear_model import
LogisticRegression
    classifier = LogisticRegression(solver='lbfgs',
                                   random_state=42,
                                   C=0.1,
                                   multi_class='ovr',
                                   penalty='l2')
    classifier.fit(X_train, y_train)
    # Predicting the Test set results
    y_pred = classifier.predict(X_test)
    # Making the Confusion Matrix
    from sklearn.metrics import confusion_matrix
    cm = confusion_matrix(y_test, y_pred)
    from sklearn.metrics import accuracy_score
    from sklearn.metrics import classification_report
    #from sklearn.metrics import confusion_matrix
    #cm = confusion_matrix(y_test, y_pred)
    Accuracy=accuracy_score(y_test, y_pred )
    report=classification_report(y_test, y_pred)
    return
classifier,Accuracy,report,X_test,y_test,cm
#import warnings
#warnings.simplefilter(action='ignore',
category=FutureWarning)

```

```

#warnings.simplefilter(action='ignore',
category=Conve)
indep_X=df2.drop('class_attribute', 1)
dep_Y=df2['class_attribute']
selectk_feature=selectkbest(indep_X,dep_Y)
rfe_feature=rfeFeature(indep_X,dep_Y)
selectk_pca=pca(selectk_feature,dep_Y)
rfe_pca=pca(rfe_feature,dep_Y)
""""SVM""""
classifier,Accuracy,report,X_test,y_test,cm=svm(sele
ctk_feature,indep_X,dep_Y)
classifier,Accuracy,report,X_test,y_test,cm=svm(rfe_
feature,indep_X,dep_Y)
classifier,Accuracy,report,X_test,y_test,cm=svm(sele
ctk_pca,indep_X,dep_Y)
classifier,Accuracy,report,X_test,y_test,cm=svm(rfe_
pca,indep_X,dep_Y)
""""Navies bay""""
classifier,Accuracy,report,X_test,y_test,cm=naives(se
lectk_feature,indep_X,dep_Y)
classifier,Accuracy,report,X_test,y_test,cm=naives(rf
e_feature,indep_X,dep_Y)
classifier,Accuracy,report,X_test,y_test,cm=naives(se
lectk_pca,indep_X,dep_Y)
classifier,Accuracy,report,X_test,y_test,cm=naives(rf
e_pca,indep_X,dep_Y)
""""Random Forest""""
classifier,Accuracy,report,X_test,y_test,cm=random(
selectk_feature,indep_X,dep_Y)
classifier,Accuracy,report,X_test,y_test,cm=random(r
fe_feature,indep_X,dep_Y)
classifier,Accuracy,report,X_test,y_test,cm=random(
selectk_pca,indep_X,dep_Y)
classifier,Accuracy,report,X_test,y_test,cm=random(r
fe_pca,indep_X,dep_Y)
""""Decision Tree""""
classifier,Accuracy,report,X_test,y_test,cm=Decision
(selectk_feature,indep_X,dep_Y)
classifier,Accuracy,report,X_test,y_test,cm=Decision
(rfe_feature,indep_X,dep_Y)
classifier,Accuracy,report,X_test,y_test,cm=Decision
(selectk_pca,indep_X,dep_Y)
classifier,Accuracy,report,X_test,y_test,cm=Decision
(rfe_pca,indep_X,dep_Y)
""""knn""""
classifier,Accuracy,report,X_test,y_test,cm=knn(sele
ctk_feature,indep_X,dep_Y)
classifier,Accuracy,report,X_test,y_test,cm=knn(rfe_
feature,indep_X,dep_Y)
classifier,Accuracy,report,X_test,y_test,cm=knn(sele
ctk_pca,indep_X,dep_Y)
classifier,Accuracy,report,X_test,y_test,cm=knn(rfe_
pca,indep_X,dep_Y)

```

```
"""Logistic"""
```

```
classifier,Accuracy,report,X_test,y_test,cm=logistics(
selectk_feature,indep_X,dep_Y)
classifier,Accuracy,report,X_test,y_test,cm=logistics(
rfe_feature,indep_X,dep_Y)
classifier,Accuracy,report,X_test,y_test,cm=logistics(
selectk_pca,indep_X,dep_Y)
classifier,Accuracy,report,X_test,y_test,cm=logistics(
rfe_pca,indep_X,dep_Y)
```

Code:

```
import pandas as pd
dataset=pd.read_csv("Preprocessed_HCC.csv",index_
col=None)
import pandas as pd
#dataset1=pd.read_csv("prep.csv",index_col=None)
df2=dataset
from sklearn.tree import export_graphviz #plot tree
from sklearn.metrics import roc_curve, auc #for
model evaluation
from sklearn.metrics import classification_report #for
model evaluation
from sklearn.metrics import confusion_matrix #for
model evaluation
from sklearn.model_selection import train_test_split
#X_train, X_test, y_train, y_test =
train_test_split(df2.drop('classification_yes', 1),
df2['classification_yes'], test_size = .2,
random_state=10)
from sklearn.metrics import roc_curve,
accuracy_score, roc_auc_score, classification_report,
r2_score, make_scorer, roc_curve, auc
from sklearn.model_selection import cross_validate,
train_test_split, cross_val_score, StratifiedKFold,
KFold, cross_val_predict
from sklearn.linear_model import LogisticRegression
import time
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
import pickle
import matplotlib.pyplot as plt
from sklearn.svm import SVC
#df2 = pd.get_dummies(df2, drop_first=True)
def selectkbest(indep_X,dep_Y):
    test = SelectKBest(score_func=chi2, k=10)
    fit1= test.fit(indep_X,dep_Y)
```

```
# summarize scores
features = indep_X.columns.values.tolist()
np.set_printoptions(precision=2)
print(features)
print(fit1.scores_)
#plt.figure(figsize=(12,3))
#plt.bar(fit1.scores_,height=0.6)
feature_series =
pd.Series(data=fit1.scores_,index=features)
feature_series.plot.bar()
selectk_features = fit1.transform(indep_X)
return selectk_features
def rfeFeature(indep_X,dep_Y):
    #model=SVR(kernel="linear")
    model = LogisticRegression(solver='lbfgs')
    rfe = RFE(model,10)
    fit3 = rfe.fit(indep_X, dep_Y)
    rfe_feature=fit3.transform(indep_X)
    features = indep_X.columns.values.tolist()
    #feature_series =
pd.Series(data=rfe_feature,index=features)
    #feature_series.plot.bar()
    return rfe_feature
def pca(features,dep_Y):
    pca = PCA(n_components=3)
    fit2 = pca.fit(features)
    pca_feature=fit2.transform(features)
    return pca_feature
def svm(features,indep_X,dep_Y):
    X_train, X_test, y_train, y_test =
train_test_split(features, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(rfe_feature, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(pca_feature, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(feature_import, dep_Y, test_size =
0.25, random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(indep_X,dep_Y, test_size = 0.25,
random_state = 0)
    #Feature Scaling
    #from sklearn.preprocessing import
StandardScaler
    sc = StandardScaler()
    X_train = sc.fit_transform(X_train)
    X_test = sc.transform(X_test)
    # Fitting K-NN to the Training set
    from sklearn.svm import SVC
    classifier = SVC(kernel = 'rbf', random_state =
0)
    classifier.fit(X_train, y_train)
```

```

# Predicting the Test set results
y_pred = classifier.predict(X_test)
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
#from sklearn.metrics import confusion_matrix
#cm = confusion_matrix(y_test, y_pred)
Accuracy=accuracy_score(y_test, y_pred )
report=classification_report(y_test, y_pred)
return
classifier,Accuracy,report,X_test,y_test,cm
def naives(features,indep_X,dep_Y):
    X_train, X_test, y_train, y_test =
train_test_split(features, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(rfe_feature, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(pca_feature, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(feature_import, dep_Y, test_size =
0.25, random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(indep_X,dep_Y, test_size = 0.25,
random_state = 0)
    #Feature Scaling
    #from sklearn.preprocessing import
StandardScaler
    sc = StandardScaler()
    X_train = sc.fit_transform(X_train)
    X_test = sc.transform(X_test)
    # Fitting K-NN to the Training set
    from sklearn.naive_bayes import GaussianNB
    classifier = GaussianNB()
    classifier.fit(X_train, y_train)
    # Predicting the Test set results
    y_pred = classifier.predict(X_test)
    # Making the Confusion Matrix
    from sklearn.metrics import confusion_matrix
    cm = confusion_matrix(y_test, y_pred)
    from sklearn.metrics import accuracy_score
    from sklearn.metrics import classification_report
    #from sklearn.metrics import confusion_matrix
    #cm = confusion_matrix(y_test, y_pred)
    Accuracy=accuracy_score(y_test, y_pred )
    report=classification_report(y_test, y_pred)
    return
    classifier,Accuracy,report,X_test,y_test,cm
def Decision(features,indep_X,dep_Y):
    X_train, X_test, y_train, y_test =
train_test_split(features, dep_Y, test_size = 0.25,
random_state = 0)

```

```

    #X_train, X_test, y_train, y_test =
train_test_split(rfe_feature, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(pca_feature, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(feature_import, dep_Y, test_size =
0.25, random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(indep_X,dep_Y, test_size = 0.25,
random_state = 0)
    #Feature Scaling
    #from sklearn.preprocessing import
StandardScaler
    sc = StandardScaler()
    X_train = sc.fit_transform(X_train)
    X_test = sc.transform(X_test)
    # Fitting K-NN to the Training set
    from sklearn.linear_model import
LogisticRegression
    classifier = LogisticRegression(random_state =
0)
    classifier.fit(X_train, y_train)
    # Predicting the Test set results
    y_pred = classifier.predict(X_test)
    # Making the Confusion Matrix
    from sklearn.metrics import confusion_matrix
    cm = confusion_matrix(y_test, y_pred)
    from sklearn.metrics import accuracy_score
    from sklearn.metrics import classification_report
    #from sklearn.metrics import confusion_matrix
    #cm = confusion_matrix(y_test, y_pred)
    Accuracy=accuracy_score(y_test, y_pred )
    report=classification_report(y_test, y_pred)
    return
    classifier,Accuracy,report,X_test,y_test,cm
def knn(features,indep_X,dep_Y):
    X_train, X_test, y_train, y_test =
train_test_split(features, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(rfe_feature, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(pca_feature, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(feature_import, dep_Y, test_size =
0.25, random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(indep_X,dep_Y, test_size = 0.25,
random_state = 0)
    #Feature Scaling
    #from sklearn.preprocessing import
StandardScaler

```

```

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
# Fitting K-NN to the Training set
from sklearn.neighbors import
KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors =
5, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)
# Predicting the Test set results
y_pred = classifier.predict(X_test)
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
#from sklearn.metrics import confusion_matrix
#cm = confusion_matrix(y_test, y_pred)
Accuracy=accuracy_score(y_test, y_pred )
report=classification_report(y_test, y_pred)
return
classifier,Accuracy,report,X_test,y_test,cm
def random(features,indep_X,dep_Y):
    X_train, X_test, y_train, y_test =
train_test_split(features, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(rfe_feature, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(pca_feature, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(feature_import, dep_Y, test_size =
0.25, random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(indep_X,dep_Y, test_size = 0.25,
random_state = 0)
    #Feature Scaling
    #from sklearn.preprocessing import
StandardScaler
    sc = StandardScaler()
    X_train = sc.fit_transform(X_train)
    X_test = sc.transform(X_test)
    # Fitting K-NN to the Training set
    from sklearn.ensemble import
RandomForestClassifier
    classifier =
RandomForestClassifier(n_estimators = 10, criterion
= 'entropy', random_state = 0)
    classifier.fit(X_train, y_train)
    # Predicting the Test set results
    y_pred = classifier.predict(X_test)
    # Making the Confusion Matrix
    from sklearn.metrics import confusion_matrix
    cm = confusion_matrix(y_test, y_pred)

```

```

from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
#from sklearn.metrics import confusion_matrix
#cm = confusion_matrix(y_test, y_pred)
Accuracy=accuracy_score(y_test, y_pred )
report=classification_report(y_test, y_pred)
return
classifier,Accuracy,report,X_test,y_test,cm
def logistics(features,indep_X,dep_Y):
    X_train, X_test, y_train, y_test =
train_test_split(features, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(rfe_feature, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(pca_feature, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(feature_import, dep_Y, test_size =
0.25, random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(indep_X,dep_Y, test_size = 0.25,
random_state = 0)
    #Feature Scaling
    #from sklearn.preprocessing import
StandardScaler
    sc = StandardScaler()
    X_train = sc.fit_transform(X_train)
    X_test = sc.transform(X_test)
    # Fitting K-NN to the Training set
    from sklearn.linear_model import
LogisticRegression
    classifier = LogisticRegression(solver='lbfgs',
random_state=42,
C=0.1,
multi_class='ovr',
penalty='l2')
    classifier.fit(X_train, y_train)
    # Predicting the Test set results
    y_pred = classifier.predict(X_test)
    # Making the Confusion Matrix
    from sklearn.metrics import confusion_matrix
    cm = confusion_matrix(y_test, y_pred)
    from sklearn.metrics import accuracy_score
    from sklearn.metrics import classification_report
    #from sklearn.metrics import confusion_matrix
    #cm = confusion_matrix(y_test, y_pred)
    Accuracy=accuracy_score(y_test, y_pred )
    report=classification_report(y_test, y_pred)
    return
classifier,Accuracy,report,X_test,y_test,cm
def kfolddd(classifier,indep_X,n_split):
    #sc = StandardScaler()
    #feature = sc.fit_transform(feature)

```

```

#X_test = sc.transform(X_test)
kfold = StratifiedKfold(
    n_splits=n_split,
    shuffle=True,
    random_state=42
)
predicted = cross_val_predict(
    classifier,
    indep_X,
    #feature,
    # rfe_feature,
    dep_Y,
    cv=kfold
)
scores = cross_val_score(
    classifier,
    indep_X,
    #feature,
    #rfe_feature,
    dep_Y,
    cv=kfold,
    scoring='f1'
)
return scores
import warnings
#import warnings
warnings.filterwarnings("ignore")
#warnings.simplefilter(action='ignore',
category=FutureWarning, UndefinedMetricWarning)
#warnings.simplefilter(action='ignore',
category=Conve)
indep_X=df2.drop('class_attribute', 1)
#std_scaler = StandardScaler() #StandardScaler() #
RobustScaler
#indep_X = std_scaler.fit_transform()
dep_Y=df2['class_attribute']
selectk_feature=selectkbest(indep_X,dep_Y)
rfe_feature=rfeFeature(indep_X,dep_Y)
selectk_pca=pca(selectk_feature,dep_Y)
rfe_pca=pca(rfe_feature,dep_Y)
"""SVM"""
classifier,Accuracy,report,X_test,y_test,cm=svm(selectk_feature,indep_X,dep_Y)
print("Svm_sk:",Accuracy)
scores=kfolddd(classifier,selectk_feature,5)
print("Cross-validated scores: { }\n".format(scores))
print("5-fold cross-validation: {:.2f}% (+/-
{:.2f}%)" .format(
    scores.mean() * 100,
    scores.std() * 2
))
classifier,Accuracy,report,X_test,y_test,cm=svm(rfe_feature,indep_X,dep_Y)
print("svm_rfe:",Accuracy)
scores=kfolddd(classifier,rfe_feature,5)
print("Cross-validated scores: { }\n".format(scores))

```

```

print("5-fold cross-validation: {:.2f}% (+/-
{:.2f}%)" .format(
    scores.mean() * 100,
    scores.std() * 2
))
classifier,Accuracy,report,X_test,y_test,cm=svm(selectk_pca,indep_X,dep_Y)
print("svm_sk_pca:",Accuracy)
scores=kfolddd(classifier,selectk_pca,5)
print("Cross-validated scores: { }\n".format(scores))
print("5-fold cross-validation: {:.2f}% (+/-
{:.2f}%)" .format(
    scores.mean() * 100,
    scores.std() * 2
))
classifier,Accuracy,report,X_test,y_test,cm=svm(rfe_pca,indep_X,dep_Y)
print("svm_rfe_pca:",Accuracy)
scores=kfolddd(classifier,rfe_pca,5)
print("Cross-validated scores: { }\n".format(scores))
print("5-fold cross-validation: {:.2f}% (+/-
{:.2f}%)" .format(
    scores.mean() * 100,
    scores.std() * 2
))
"""Random Forest"""
classifier,Accuracy,report,X_test,y_test,cm=random(selectk_feature,indep_X,dep_Y)
print("Random_sk:",Accuracy)
scores=kfolddd(classifier,selectk_feature,5)
print("Cross-validated scores: { }\n".format(scores))
print("5-fold cross-validation: {:.2f}% (+/-
{:.2f}%)" .format(
    scores.mean() * 100,
    scores.std() * 2
))
classifier,Accuracy,report,X_test,y_test,cm=random(rfe_feature,indep_X,dep_Y)
print("Random_rfe:",Accuracy)
scores=kfolddd(classifier,rfe_feature,5)
print("Cross-validated scores: { }\n".format(scores))
print("5-fold cross-validation: {:.2f}% (+/-
{:.2f}%)" .format(
    scores.mean() * 100,
    scores.std() * 2
))
classifier,Accuracy,report,X_test,y_test,cm=random(selectk_pca,indep_X,dep_Y)
print("Random_sk_pca:",Accuracy)
scores=kfolddd(classifier,selectk_pca,5)
print("Cross-validated scores: { }\n".format(scores))
print("5-fold cross-validation: {:.2f}% (+/-
{:.2f}%)" .format(
    scores.mean() * 100,
    scores.std() * 2
))

```



```

classifier,Accuracy,report,X_test,y_test,cm=random(r
fe_pca,indep_X,dep_Y)
print("Random_rfe_pca:",Accuracy)
scores=kfolddd(classifier,rfe_pca,5)
print('Cross-validated scores: {}'.format(scores))
print("5-fold cross-validation: {:.2f}% (+/-
{:.2f}%)").format(
    scores.mean() * 100,
    scores.std() * 2
))
"""Decision Tree"""
classifier,Accuracy,report,X_test,y_test,cm=Decision
(selectk_feature,indep_X,dep_Y)
print("Decision_sk:",Accuracy)
scores=kfolddd(classifier,selectk_feature,5)
print('Cross-validated scores: {}'.format(scores))
print("5-fold cross-validation: {:.2f}% (+/-
{:.2f}%)").format(
    scores.mean() * 100,
    scores.std() * 2
))
classifier,Accuracy,report,X_test,y_test,cm=Decision
(rfe_feature,indep_X,dep_Y)
print("Decision_rfe:",Accuracy)
scores=kfolddd(classifier,rfe_feature,5)
print('Cross-validated scores: {}'.format(scores))
print("5-fold cross-validation: {:.2f}% (+/-
{:.2f}%)").format(
    scores.mean() * 100,
    scores.std() * 2
))
classifier,Accuracy,report,X_test,y_test,cm=Decision
(selectk_pca,indep_X,dep_Y)
print("Decision_sk_pca:",Accuracy)
scores=kfolddd(classifier,selectk_pca,5)
print('Cross-validated scores: {}'.format(scores))
print("5-fold cross-validation: {:.2f}% (+/-
{:.2f}%)").format(
    scores.mean() * 100,
    scores.std() * 2
))
classifier,Accuracy,report,X_test,y_test,cm=Decision
(rfe_pca,indep_X,dep_Y)
print("Decision_rfe_pca:",Accuracy)
scores=kfolddd(classifier,rfe_pca,5)
print('Cross-validated scores: {}'.format(scores))
print("5-fold cross-validation: {:.2f}% (+/-
{:.2f}%)").format(
    scores.mean() * 100,
    scores.std() * 2
))
"""Navies bay"""
classifier,Accuracy,report,X_test,y_test,cm=naives(se
lectk_feature,indep_X,dep_Y)
print("Naives_sk:",Accuracy)
scores=kfolddd(classifier,selectk_feature,5)

```

```

print('Cross-validated scores: {}'.format(scores))
print("5-fold cross-validation: {:.2f}% (+/-
{:.2f}%)").format(
    scores.mean() * 100,
    scores.std() * 2
))
classifier,Accuracy,report,X_test,y_test,cm=naives(rf
e_feature,indep_X,dep_Y)
print("Naives_rfe:",Accuracy)
scores=kfolddd(classifier,rfe_feature,5)
print('Cross-validated scores: {}'.format(scores))
print("5-fold cross-validation: {:.2f}% (+/-
{:.2f}%)").format(
    scores.mean() * 100,
    scores.std() * 2
))
classifier,Accuracy,report,X_test,y_test,cm=naives(se
lectk_pca,indep_X,dep_Y)
print("Naives_sk_pca:",Accuracy)
scores=kfolddd(classifier,selectk_pca,5)
print('Cross-validated scores: {}'.format(scores))
print("5-fold cross-validation: {:.2f}% (+/-
{:.2f}%)").format(
    scores.mean() * 100,
    scores.std() * 2
))
classifier,Accuracy,report,X_test,y_test,cm=naives(rf
e_pca,indep_X,dep_Y)
print("Naives_rfe_pca:",Accuracy)
scores=kfolddd(classifier,rfe_pca,5)
print('Cross-validated scores: {}'.format(scores))
print("5-fold cross-validation: {:.2f}% (+/-
{:.2f}%)").format(
    scores.mean() * 100,
    scores.std() * 2
))
"""knn"""
classifier,Accuracy,report,X_test,y_test,cm=knn(sele
ctk_feature,indep_X,dep_Y)
print("knn_sk:",Accuracy)
scores=kfolddd(classifier,selectk_feature,5)
print('Cross-validated scores: {}'.format(scores))
print("5-fold cross-validation: {:.2f}% (+/-
{:.2f}%)").format(
    scores.mean() * 100,
    scores.std() * 2
))
classifier,Accuracy,report,X_test,y_test,cm=knn(rfe_
feature,indep_X,dep_Y)
print("knn_rfe:",Accuracy)
scores=kfolddd(classifier,rfe_feature,5)
print('Cross-validated scores: {}'.format(scores))
print("5-fold cross-validation: {:.2f}% (+/-
{:.2f}%)").format(
    scores.mean() * 100,
    scores.std() * 2
)

```

```

))
classifier,Accuracy,report,X_test,y_test,cm=knn(selectk_pca,indep_X,dep_Y)
print("knn_sk_pca:",Accuracy)
scores=kfolddd(classifier,selectk_pca,5)
print('Cross-validated scores: {}'.format(scores))
print("5-fold cross-validation: {:.2f}% (+/- {:.2f}%)".format(
    scores.mean() * 100,
    scores.std() * 2
))
classifier,Accuracy,report,X_test,y_test,cm=knn(rfe_pca,indep_X,dep_Y)
print("knn_rfe_pca:",Accuracy)
scores=kfolddd(classifier,rfe_pca,5)
print('Cross-validated scores: {}'.format(scores))
print("5-fold cross-validation: {:.2f}% (+/- {:.2f}%)".format(
    scores.mean() * 100,
    scores.std() * 2
))
"""Logistic"""
classifier,Accuracy,report,X_test,y_test,cm=logistics(selectk_feature,indep_X,dep_Y)
print("Logistics_sk:",Accuracy)
scores=kfolddd(classifier,selectk_feature,5)
print('Cross-validated scores: {}'.format(scores))
print("5-fold cross-validation: {:.2f}% (+/- {:.2f}%)".format(
    scores.mean() * 100,
    scores.std() * 2
))
classifier,Accuracy,report,X_test,y_test,cm=logistics(rfe_feature,indep_X,dep_Y)
print("Logistics_rfe:",Accuracy)
scores=kfolddd(classifier,rfe_feature,5)
print('Cross-validated scores: {}'.format(scores))
print("5-fold cross-validation: {:.2f}% (+/- {:.2f}%)".format(
    scores.mean() * 100,
    scores.std() * 2
))
classifier,Accuracy,report,X_test,y_test,cm=logistics(selectk_pca,indep_X,dep_Y)
print("Logistics_sk_pca:",Accuracy)
scores=kfolddd(classifier,selectk_pca,5)
print('Cross-validated scores: {}'.format(scores))
print("5-fold cross-validation: {:.2f}% (+/- {:.2f}%)".format(
    scores.mean() * 100,
    scores.std() * 2
))
classifier,Accuracy,report,X_test,y_test,cm=logistics(rfe_pca,indep_X,dep_Y)
print("Logistics_rfe_pca:",Accuracy)
scores=kfolddd(classifier,rfe_pca,5)

```

```

print('Cross-validated scores: {}'.format(scores))
print("5-fold cross-validation: {:.2f}% (+/- {:.2f}%)".format(
    scores.mean() * 100,
    scores.std() * 2
))

```

Model:

```

import pandas as pd
dataset=pd.read_csv("finalpree.csv",index_col=None)
#dataset=dataset.drop(columns=['age_'])
"""nominal_indexes = [
    0,1,2, 3, 4, 5,
    6, 8, 9, 7,10,
    11, 12, 13,
    14, 15, 16,
    17, 18, 19,
    20, 21, 22
]
dataset.columns[1]

for i in nominal_indexes:
    dictt={1:'yes',0:'no'}
    col=dataset.columns[i]
    print(dataset.columns[i])
    dataset[col]=dataset[col].replace(dictt)"""
dataset1 = pd.get_dummies(dataset, drop_first=True)
import pandas as pd
#dataset1=pd.read_csv("prep.csv",index_col=None)
df2=dataset1
from sklearn.tree import export_graphviz #plot tree
from sklearn.metrics import roc_curve, auc #for model evaluation
from sklearn.metrics import classification_report #for model evaluation
from sklearn.metrics import confusion_matrix #for model evaluation
from sklearn.model_selection import train_test_split
#X_train, X_test, y_train, y_test =
train_test_split(df2.drop('classification_yes', 1),
df2['classification_yes'], test_size = .2,
random_state=10)
import time
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
import pickle

```

```

import matplotlib.pyplot as plt
from sklearn.svm import SVC
#df2 = pd.get_dummies(df2, drop_first=True)
def selectkbest(indep_X,dep_Y):
    test = SelectKBest(score_func=chi2, k=6)
    fit1= test.fit(indep_X,dep_Y)
    # summarize scores
    features = indep_X.columns.values.tolist()
    np.set_printoptions(precision=2)
    print(features)
    print(fit1.scores_)
    #plt.figure(figsize=(12,3))
    #plt.bar(fit1.scores_,height=0.6)
    feature_series =
pd.Series(data=fit1.scores_,index=features)
    feature_series.plot.bar()
    selectk_features = fit1.transform(indep_X)
    return selectk_features
def rfeFeature(indep_X,dep_Y):
    #model=SVR(kernel="linear")
    model = LogisticRegression(solver='lbfgs')
    rfe = RFE(model,3)
    fit3 = rfe.fit(indep_X, dep_Y)
    rfe_feature=fit3.transform(indep_X)
    features = indep_X.columns.values.tolist()
    #feature_series =
pd.Series(data=rfe_feature,index=features)
    #feature_series.plot.bar()
    return rfe_feature
def pca(features,dep_Y):
    pca = PCA(n_components=3)
    fit2 = pca.fit(features)
    pca_feature=fit2.transform(features)
    return pca_feature
def svm(features,indep_X,dep_Y):
    X_train, X_test, y_train, y_test =
train_test_split(features, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(rfe_feature, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(pca_feature, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(feature_import, dep_Y, test_size =
0.25, random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(indep_X,dep_Y, test_size = 0.25,
random_state = 0)
    #Feature Scaling
    #from sklearn.preprocessing import
StandardScaler
    sc = StandardScaler()
    X_train = sc.fit_transform(X_train)
    X_test = sc.transform(X_test)

```

```

# Fitting K-NN to the Training set
from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state =
0)
classifier.fit(X_train, y_train)
# Predicting the Test set results
y_pred = classifier.predict(X_test)
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
#from sklearn.metrics import confusion_matrix
#cm = confusion_matrix(y_test, y_pred)
Accuracy=accuracy_score(y_test, y_pred )
report=classification_report(y_test, y_pred)
return
classifier,Accuracy,report,X_test,y_test,cm
def naives(features,indep_X,dep_Y):
    X_train, X_test, y_train, y_test =
train_test_split(features, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(rfe_feature, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(pca_feature, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(feature_import, dep_Y, test_size =
0.25, random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(indep_X,dep_Y, test_size = 0.25,
random_state = 0)
    #Feature Scaling
    #from sklearn.preprocessing import
StandardScaler
    sc = StandardScaler()
    X_train = sc.fit_transform(X_train)
    X_test = sc.transform(X_test)
    # Fitting K-NN to the Training set
    from sklearn.naive_bayes import GaussianNB
    classifier = GaussianNB()
    classifier.fit(X_train, y_train)
    # Predicting the Test set results
    y_pred = classifier.predict(X_test)
    # Making the Confusion Matrix
    from sklearn.metrics import confusion_matrix
    cm = confusion_matrix(y_test, y_pred)
    from sklearn.metrics import accuracy_score
    from sklearn.metrics import classification_report
    #from sklearn.metrics import confusion_matrix
    #cm = confusion_matrix(y_test, y_pred)
    Accuracy=accuracy_score(y_test, y_pred )

```

```

        report=classification_report(y_test, y_pred)
        return
classifier,Accuracy,report,X_test,y_test,cm
def knn(features,indep_X,dep_Y):
    X_train, X_test, y_train, y_test =
train_test_split(features, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(rfe_feature, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(pca_feature, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(feature_import, dep_Y, test_size =
0.25, random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(indep_X,dep_Y, test_size = 0.25,
random_state = 0)
    #Feature Scaling
    #from sklearn.preprocessing import
StandardScaler
    sc = StandardScaler()
    X_train = sc.fit_transform(X_train)
    X_test = sc.transform(X_test)
    # Fitting K-NN to the Training set
    from sklearn.neighbors import
KNeighborsClassifier
    classifier = KNeighborsClassifier(n_neighbors =
5, metric = 'minkowski', p = 2)
    classifier.fit(X_train, y_train)
    # Predicting the Test set results
    y_pred = classifier.predict(X_test)
    # Making the Confusion Matrix
    from sklearn.metrics import confusion_matrix
    cm = confusion_matrix(y_test, y_pred)
    from sklearn.metrics import accuracy_score
    from sklearn.metrics import classification_report
    #from sklearn.metrics import confusion_matrix
    #cm = confusion_matrix(y_test, y_pred)
    Accuracy=accuracy_score(y_test, y_pred )
    report=classification_report(y_test, y_pred)
    return
classifier,Accuracy,report,X_test,y_test,cm
import warnings
warnings.simplefilter(action='ignore',
category=FutureWarning)
#warnings.simplefilter(action='ignore',
category=Conve)
indep_X=df2.drop('class_attribute', 1)
dep_Y=df2['class_attribute']
selectk_feature=selectkbest(indep_X,dep_Y)
rfe_feature=rfeFeature(indep_X,dep_Y)
selectk_pca=pca(selectk_feature,dep_Y)
rfe_pca=pca(rfe_feature,dep_Y)

```

```

""""SVM""""
classifier,Accuracy,report,X_test,y_test,cm=svm(sele
ctk_feature,indep_X,dep_Y)

classifier,Accuracy,report,X_test,y_test,cm=svm(rfe_
feature,indep_X,dep_Y)

classifier,Accuracy,report,X_test,y_test,cm=svm(sele
ctk_pca,indep_X,dep_Y)

classifier,Accuracy,report,X_test,y_test,cm=svm(rfe_
pca,indep_X,dep_Y)
""""Navies bay""""
classifier,Accuracy,report,X_test,y_test,cm=naives(se
lectk_feature,indep_X,dep_Y)
classifier,Accuracy,report,X_test,y_test,cm=naives(rf
e_feature,indep_X,dep_Y)
classifier,Accuracy,report,X_test,y_test,cm=naives(se
lectk_pca,indep_X,dep_Y)
classifier,Accuracy,report,X_test,y_test,cm=naives(rf
e_pca,indep_X,dep_Y)
""""knn""""
classifier,Accuracy,report,X_test,y_test,cm=knn(sele
ctk_feature,indep_X,dep_Y)
classifier,Accuracy,report,X_test,y_test,cm=knn(rfe_
feature,indep_X,dep_Y)
classifier,Accuracy,report,X_test,y_test,cm=knn(sele
ctk_pca,indep_X,dep_Y)
classifier,Accuracy,report,X_test,y_test,cm=knn(rfe_
pca,indep_X,dep_Y)

```

Final Model:

```

import pandas as pd
import pandas as pd
#dataset1=pd.read_csv("prep.csv",index_col=None)
from sklearn.tree import export_graphviz #plot tree
from sklearn.metrics import roc_curve, auc #for
model evaluation
from sklearn.metrics import classification_report #for
model evaluation
from sklearn.metrics import confusion_matrix #for
model evaluation
from sklearn.model_selection import train_test_split
#X_train, X_test, y_train, y_test =
train_test_split(df2.drop('classification_yes', 1),
df2['classification_yes'], test_size = .2,
random_state=10)
import time
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.model_selection import train_test_split

```

```

from sklearn.decomposition import PCA
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
import pickle
import matplotlib.pyplot as plt
from sklearn.svm import SVC
#df2 = pd.get_dummies(df2, drop_first=True)
def selectkbest(indep_X,dep_Y):
    test = SelectKBest(score_func=chi2, k=6)
    fit1= test.fit(indep_X,dep_Y)
    # summarize scores
    features = indep_X.columns.values.tolist()
    np.set_printoptions(precision=2)
    #print(features)
    #print(fit1.scores_)
    #plt.figure(figsize=(12,3))
    #plt.bar(fit1.scores_,height=0.6)
    feature_series =
pd.Series(data=fit1.scores_,index=features)
    feature_series.plot.bar()
    selectk_features = fit1.transform(indep_X)
    return selectk_features
def rfeFeature(indep_X,dep_Y):
    #model=SVR(kernel="linear")
    model = LogisticRegression(solver='lbfgs')
    rfe = RFE(model,6)
    fit3 = rfe.fit(indep_X, dep_Y)
    rfe_feature=fit3.transform(indep_X)
    features = indep_X.columns.values.tolist()
    #feature_series =
pd.Series(data=rfe_feature,index=features)
    #feature_series.plot.bar()
    return rfe_feature
def pca(features,dep_Y):
    pca = PCA(n_components=5)
    fit2 = pca.fit(features)
    pca_feature=fit2.transform(features)
    return pca_feature
def svm(features,indep_X,dep_Y):
    X_train, X_test, y_train, y_test =
train_test_split(features, dep_Y, test_size = 0.25,
random_state = 56,shuffle=True)
    #X_train, X_test, y_train, y_test =
train_test_split(rfe_feature, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(pca_feature, dep_Y, test_size = 0.25,
random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(feature_import, dep_Y, test_size =
0.25, random_state = 0)
    #X_train, X_test, y_train, y_test =
train_test_split(indep_X,dep_Y, test_size = 0.25,
random_state = 0)
    #Feature Scaling

```

```

#from sklearn.preprocessing import
StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
# Fitting K-NN to the Training set
from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state =
0,probability=True)
classifier.fit(X_train, y_train)
# Predicting the Test set results
y_pred = classifier.predict(X_test)
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
#from sklearn.metrics import confusion_matrix
#cm = confusion_matrix(y_test, y_pred)
Accuracy=accuracy_score(y_test, y_pred )
report=classification_report(y_test, y_pred)
return
classifier,Accuracy,report,X_test,y_test,cm
dataset=pd.read_csv("finalpre.csv",index_col=None
)
df2=dataset
import warnings
#import warnings
warnings.filterwarnings("ignore")
indep_X=df2.drop('class_attribute', 1)
dep_Y=df2['class_attribute']
selectk_feature=selectkbest(indep_X,dep_Y)
rfe_feature=rfeFeature(indep_X,dep_Y)
selectk_pca=pca(selectk_feature,dep_Y)
rfe_pca=pca(rfe_feature,dep_Y)
classifier,Accuracy,report,X_test,y_test,cm=svm(sele
ctk_feature,indep_X,dep_Y)
print("Svm_sk:",Accuracy)
#dir(classifier)

```