

PREDICTING LIVER FAILURE USING SUPERVISED MACHINE LEARNING APPROACH

Submitted in partial fulfillment of the requirements for
the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

Adavala Sai Poojitha (38110014)
P. A. Reena Jennifer (38110459)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600 119**

MARCH-2022



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of Adavala Sai Poojitha (REG NO: 38110014), P. A. Reena Jennifer (REG NO: 38110459) who have done Project work as a team who carried out the project entitled **“PREDICTING LIVER FAILURE USING SUPERVISED MACHINE LEARNING APPROACH”** under my supervision from November 2021 to April 2022.

Internal Guide
Dr. T. Prem Jacob

External Guide

Head of the Department
Dr. L. Lakshmanan M.E., Ph.D.,
Dr.S.Vigneshwari M.E., Ph.D.,

Submitted for Viva voce Examination held on_____

Internal Examiner

External Examiner

DECLARATION

We Adavala Sai Poojitha (REG NO: 38110014) and P. A. Reena Jennifer (REG NO: 38110459) hereby declare that the Project Report entitled “**PREDICTING LIVER FAILURE USING SUPERVISED MACHINE LEARNING APPROACH**” done by me under the guidance of **Dr. T. Prem Jacob**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D, Dean**, School of Computing **Dr. L. Lakshmanan M.E., Ph.D. ,** and **Dr.S.Vigneshwari M.E., Ph.D. Heads** of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. T. Prem Jacob**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

The function of liver is to filter blood that circulates through the body, converting nutrients and drugs absorbed from the digestive tract into ready-to-use chemicals. The liver performs many other important functions, such as removing toxins and other chemical waste products from the blood and readying them for excretion. Liver failure that begins in the cells of your liver. Nowadays machine learning is applied to healthcare system where there is a chance of predicting the disease early. The main necessity of Artificial intelligence is data. The past dataset is collected and that dataset is used to build a machine learning model. The necessary pre-processing techniques are applied like univariate analysis and bivariate analysis are implemented. The data is visualized for better understanding of the features and based on that a classification model is built by using machine learning algorithm and comparison of algorithms are done based on their performance metrics like accuracy, F1 score recall etc.

Keywords— liver disease, demographic variables, prognostic/biochemical variables, statistical learning for variable selection and classification, support vector machine.

TABLE OF CONTENTS

CHAPTER NO	TITLE NAME	PAGE NO
	ABSTRACT	V
	LIST OF FIGURES	VIII
1	INTRODUCTION	
1.1	GENERAL	1
1.2	DOMAIN OVERVIEW	3
1.3	PROBLEM STATEMENT	9
2	LITERATURE SURVEY	
2.1	LITERATURE REVIEW	11
2.2	SURVEY WALKTHROUGH	15
2.2.1	NUMPY	15
2.2.2	PANDAS	18
2.2.3	MATPLOTLIB	20
2.2.4	SKLEARN	23
2.3	PROJECT GOALS	25
3	AIM AND SCOPE OF PRESENT INVESTIGATION	
3.1	AIM	27
3.2	SCOPE	27
3.3	SYSTEM STUDY	27
3.4	EXISTING SYSTEM	29
3.5	PROPOSED SYSTEM	30
3.5.1	SYSTEM ARCHITECTUR	31

3.5.2	WORKING PROCESS	33
4	EXPERIMENTAL OR MATERIALS AND METHODS AND ALGORITHMS USED	
4.1	DATA GATHERING	35
4.1.1	PREPARING DATASET	35
4.1.2	DATA PREPARATION	37
4.2	DATA PRE-PROCESSING	38
4.2.1	DATA WRANGLING	39
4.3	DATA VISUALIZATION	41
4.4	DATA ANALYSIS	45
4.4.1	DATA COLLECTION	46
4.4.2	CONSTRUCTION OF PREDICTIVE MODEL	46
4.4.3	PROJECT REQUIREMENTS	47
4.4.4	ENVIRONMENTAL REQUIREMENTS	48
4.5	WORKFLOW PROCESS	48
4.5.1	USECASE DIAGRAM	49
4.5.2	CLASS DIAGRAM	50
4.5.3	ACTIVITY DIAGRAM	51
4.5.4	SEQUENCE DIAGRAM	52
4.5.5	ER – DIAGRAM	53
4.5.6	PREDICTION RESULT BY ACCURACY	55
4.5.7	TRAINING AND TESTING MODEL	58
4.6	ALGORITHM AND TECHNIQUES	59
4.6.1	LOGISTIC REGRESSION	60
4.6.2	DECISION TREE	63
4.6.3	RANDOM FOREST	66
4.6.4	SUPPORT VECTOR MACHINE	71
4.6.5	DEPLOYMENT	75
5	RESULTS AND DISCUSSION	79
6	CONCLUSION AND FUTURE WORK	82
7	APPENDICES AND SCREENSHOTS	85

LIST OF FIGURES

Figure no.	Name of the Figure	Page no.
1.1	Stages of Liver Damage	2
1.2	Machine Learning	8
1.3	Types of Machine Learning	9
3.1	System Overview	31
3.2	System Architecture Diagram	32
4.1	Dataset Used	37
4.2	Unbiased evaluation of data	38
4.3	Missing Data	39
4.4	Total male and female patients	42
4.5	Liver Failure by Gender	43
4.6	Visualization of Total proteins and Albumin	44
4.7	Correlation between Data	44
4.8	Correlation Heatmap	45
4.9	Process of Dataflow Diagram	47
4.10	Workflow Diagram	49
4.11	Use case Diagram	50
4.12	Class Diagram	51
4.13	Activity Diagram	52
4.14	Sequence Diagram	53
4.15	E – R Diagram	54
4.16	Confusion Matrix of LR	61
4.17	Classification Report of LR	62

4.18	Confusion Matrix of DT	65
4.19	Classification Report of DT	66
4.20	Confusion Matrix of RF	69
4.21	Classification Report of RF	70
4.22	Confusion Matrix of SVM	73
4.23	Classification Report of SVM	74

CHAPTER 1

INTRODUCTION

1.1 GENERAL

The liver is a large, pyramid-shaped organ that lies behind your ribs on the right side of your body. It's under the right lung. It's divided into right and left lobes. The liver helps break down and store nutrients. These include sugars, starch, fats, and proteins. It also makes proteins, such as albumin. This helps the body balance fluids. The liver makes clotting factors, which help blood thicken or clot when a person is bleeding. Bile made in the liver is important for digesting food and for other bodily functions.

One of the liver's most important jobs is to filter out and destroy toxins in the blood. When the liver isn't working well, chemicals can build up inside the body and cause damage. Liver cancer is cancer that starts in your liver. It's also called primary liver cancer. Primary liver cancer is not the same as cancer that started somewhere else in the body and then spread (metastasized) to the liver.

Cancer that starts in another organ, such as the colon, breast, or lung, and then spreads to the liver is called secondary liver cancer. Secondary liver cancer is far more common in the U.S. than primary liver cancer. Cancer that has spread to the liver from somewhere else is treated like the original cancer. For instance, lung cancer that has spread to the liver is treated like lung cancer.

Fatty liver disease (FLD) has become a rampant condition. It is associated with a high rate of morbidity and mortality in a population. The condition is commonly referred as FLD. Early prediction of FLD would allow patients to take necessary preventive, diagnosis, and treatment. Chronic liver diseases and cirrhosis are the 11th leading cause of death in the world, accounting for 1.1 million deaths annually. The global prevalence of cirrhosis has been substantially rising from 71 million in 1990 to over 122 million in 2017. Common causes of cirrhosis are chronic hepatitis B virus (HBV) and hepatitis C virus (HCV) infections, alcohol-related liver disease and nonalcoholic steatohepatitis (NASH). Over the past decade, there has been a

temporal shift in the prevalence of causes of cirrhosis, i.e., the prevalence of NASH has been dramatically increasing, whereas the prevalence of other causes has been slowly decreasing. The estimated worldwide prevalence of nonalcoholic fatty liver disease (NAFLD) is 25% and is projected to be to 33.5% by 2030, emphasizing the importance of both cirrhosis and NAFLD.

The computer-based approach is required for the non-invasive detection of chronic liver diseases that are asymptomatic, progressive, and potentially fatal in nature. 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. By using machine learning techniques medical problems can be easily solved and the cost of diagnosis will be reduced.

The gold standard for the diagnosis of liver fibrosis and nonalcoholic fatty liver disease (NAFLD) is liver biopsy. Various noninvasive modalities, e.g., ultrasonography, elastography and clinical predictive scores, have been used as alternatives to liver biopsy, with limited performance. Recently, artificial intelligence (AI) models have been developed and integrated into noninvasive diagnostic tools to improve their performance.



FIG: 1.1 STAGES OF LIVER DAMAGE

1.2 Domain Overview

DATA SCIENCE:

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

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

However, the definition was still in flux. The term "data science" was first coined in 2008 by **D.J. Patil, and Jeff Hammerbacher**, the pioneer leads of data and analytics efforts at LinkedIn and Facebook. In less than a decade, it has become one of the hottest and most trending professions in the market. Data science is the field of study that combines domain expertise, programming skills, and knowledge of mathematics and statistics to extract meaningful insights from data.

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

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

ARTIFICIAL INTELLIGENCE:

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The

term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving.

Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to the natural intelligence displayed by humans or animals. Leading AI textbooks define the field as the study of "intelligent_agents" any system that perceives its environment and takes actions that maximize its chance of achieving its goals. Some popular accounts use the term "artificial intelligence" to describe machines that mimic "cognitive" functions that humans associate with the human_mind, such as "learning" and "problem solving", however this definition is rejected by major AI researchers.

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, speech recognition and machine vision.

AI applications include advanced web search engines, recommendation systems (used by YouTube, Amazon and Netflix), Understanding human speech (such as Siri or Alexa), self-driving cars (e.g., Tesla), and competing at the highest level in strategic game systems (such as chess and go), As machines become increasingly capable, tasks considered to require "intelligence" are often removed from the definition of AI, a phenomenon known as the AI effect. For instance, optical character recognition is frequently excluded from things considered to be AI, having become a routine technology.

Artificial intelligence was founded as an academic discipline in 1956, and in the years since has experienced several waves of optimism, followed by disappointment and the loss of funding (known as an "AI winter"), followed by new approaches, success and renewed funding. AI research has tried and discarded many different approaches during its lifetime, including simulating the brain, modeling human problem solving, formal logic, large databases of knowledge and imitating animal behavior. In the first decades of the 21st century, highly mathematical statistical machine learning has dominated the field, and this technique has proved highly successful, helping to solve many challenging problems throughout industry and academia.

The various sub-fields of AI research are centered around particular goals and the use of particular tools. The traditional goals of AI research

include reasoning, knowledge representation, planning, learning, natural language processing, perception and the ability to move and manipulate objects. General intelligence (the ability to solve an arbitrary problem) is among the field's long-term goals. To solve these problems, AI researchers use versions of search and mathematical optimization, formal logic, artificial neural networks, and methods based on statistics, probability and economics. AI also draws upon computer science, psychology, linguistics, philosophy, and many other fields.

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

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

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

AI programming focuses on three cognitive skills: learning, reasoning and self-correction.

Learning processes. This aspect of AI programming focuses on acquiring data and creating rules for how to turn the data into actionable information. The rules, which

are called algorithms, provide computing devices with step-by-step instructions for how to complete a specific task.

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

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

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

Artificial neural networks and deep learning artificial intelligence technologies are quickly evolving, primarily because AI processes large amounts of data much faster and makes predictions more accurately than humanly possible.

Natural Language Processing (NLP):

Natural language processing (NLP) allows machines to read and understand human language. A sufficiently powerful natural language processing system would enable natural-language user interfaces and the acquisition of knowledge directly from human-written sources, such as newswire texts. Some straightforward applications of natural language processing include information retrieval, text mining, question answering and machine translation.

Many current approaches use word co-occurrence frequencies to construct syntactic representations of text. "Keyword spotting" strategies for search are popular and scalable but dumb; a search query for "dog" might only match documents with the literal word "dog" and miss a document with the word "poodle".

"Lexical affinity" strategies use the occurrence of words such as "accident" to assess the sentiment of a document. Modern statistical NLP approaches can combine all these strategies as well as others, and often achieve acceptable accuracy at the page or paragraph level. Beyond semantic NLP, the ultimate goal of "narrative" NLP is to embody a full understanding of commonsense reasoning.

By 2019, transformer-based deep learning architectures could generate coherent text. AI-assisted systems have promising potential for the diagnosis of liver fibrosis and NAFLD. Validations of their performances are warranted before implementing these AI-assisted systems in clinical practice.

MACHINE LEARNING

Machine learning is to predict the future from past data. Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data and the basics of Machine Learning, implementation of a simple machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms. It feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. Machine learning can be roughly separated in to three categories. There are supervised learning, unsupervised learning and reinforcement learning.

Supervised learning program is both given the input data and the corresponding labeling to learn data has to be labeled by a human being beforehand. Unsupervised learning is no labels. It provided to the learning algorithm. This algorithm has to figure out the clustering of the input data. Finally, Reinforcement learning dynamically interacts with its environment and it receives positive or negative feedback to improve its performance.

Data scientists use many different kinds of machine learning algorithms to discover patterns in python that lead to actionable insights. At a high level, these different algorithms can be classified into two groups based on the way they "learn" about

data to make predictions: supervised and unsupervised learning. Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function from input variables(X) to discrete output variables(y). In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc.



Fig: 1.2 MACHINE LEARNING

Supervised Machine Learning is the majority of practical machine learning uses supervised learning. Supervised learning is where have input variables (X) and an output variable (y) and use an algorithm to learn the mapping function from the input to the output is $y = f(X)$. The goal is to approximate the mapping function so well that when you have new input data (X) that you can predict the output variables (y) for that data.

Techniques of Supervised Machine Learning algorithms include **logistic regression**, **multi-class classification**, **Decision Trees** and **support vector machines** etc.

Supervised learning requires that the data used to train the algorithm is already labeled with correct answers. Supervised learning problems can be further grouped into **Classification** problems. This problem has as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables.

The difference between the two tasks is the fact that the dependent attribute is numerical for categorical for classification. A classification model attempts to draw some conclusion from observed values.

Given one or more inputs a classification model will try to predict the value of one or more outcomes. A classification problem is when the output variable is a category, such as “red” or “blue”.

Training set and testing set

Machine learning is about learning some properties of a data set and then testing those properties against another data set. A common practice in machine learning is to evaluate an algorithm by splitting a data set into two.

We call one of those sets the **training set**, on which we learn some properties; we call the other set the **testing set**, on which we test the learned properties.

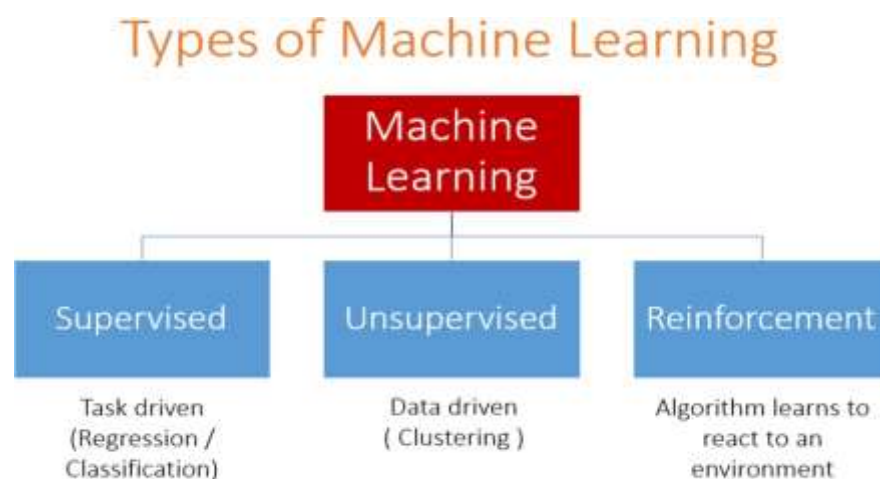


FIG: 1.3 TYPES OF MACHINE LEARNING

1.2 PROBLEM STATEMENT

Liver disease is any trouble of liver function that causes sickness. The liver is responsible for many dangerous functions within the body and should it become diseased or damaged, the loss of those functions can cause significant injury to the body. Liver disease is also referred to as hepatic disease.

Liver disease is a large term that covers all the potential problems that cause the liver to fail to perform its designated functions. Usually, more than 75% or three quarters of liver tissue needs to be affected before a decrease in function occurs. we can make predict the problem using predictive model by using machine learning technique and compare machine learning algorithms.

CHAPTER 2

LITERATURE SURVEY

2.1 LITERATURE REVIEW

2.1.1

Title : The Diagnosis of Chronic Liver Disease using Machine Learning Techniques

Author: Golmei Shaheamlung, Harshpreet Kaur

Year : 26-March-2021

In the 21st-century, the issue of liver disease has been increasing all over the world. As per the latest survey report, liver disease death toll has been rising approximately 2 million per year worldwide. The overall percentage of death by liver disease is 3.5% worldwide. Chronic Liver disease is also considered to be one of the deadly diseases, so early detection and treatment can recover the disease easily. The hidden knowledge of liver disease is recognized and extracted using a historical liver disease database. The complex queries are responded to diagnose liver disease. The proposed model improved by applying a combination of three classifiers, Logistic regression, Random Forest, and KNN algorithm. The python is employed for the implementation of the suggested model and the result proved regarding accuracy that is achieved 77.58 percent. For the future, the execution of the clustering algorithm is performed with the hybrid classifier technique for the division of data.

2.1.2

Title : Prediction and Analysis of Liver Disorder Diseases by using Data Mining Technique: Survey

Author: Shambel Kefelegn, Pooja Kamat

Year : 2018

Liver disorder diseases one of the major diseases in the world, Liver is one of the huge solid organs in the human body; and is also considered a gland because, among its many functions, it makes and secretes bile. The liver theatre's vital role in many physical functions from protein manufacture and blood clotting to fat, sugar and iron metabolism. Liver disorder diseases are any trouble of liver purpose that reason for sickness. The study of paper to predicting and analysing liver disorder diseases to produce better performance accuracy by comparing various data mining classification algorithm and the performance of the accuracy is measured by confusion matrices. Decision Tree considered for performance evaluation in liver disorder diseases prediction Future work we can use the Hybrid approach to get better performance accuracy for liver disorder diseases prediction with their suitable data sets.

2.1.3

Title: Liver disease Prediction by using different Decision Tree techniques

Author: Nazmun Nahar and Ferdous Ara2

Year : 2018

Early prediction of liver disease is very important to save human life and take proper steps to control the disease. Decision Tree algorithms have been successfully

applied in various fields especially in medical science. This research work explores the early prediction of liver disease using various decision tree techniques. The liver disease dataset which is select for this study is consisting of attributes like total bilirubin, direct bilirubin, age, gender, total proteins, albumin and globulin ratio. The main purpose of this work is to calculate the performance of various decision tree techniques and compare their performance. The study employed some decision tree algorithm such as J48, LMT, Random Forest, Random tree, REPTree, Decision Stump and Hoeffding Tree to predict the liver disease at an earlier stage. These algorithm gives various result based on Accuracy, Mean Absolute Error, Precision, Recall, Kappa statistics and Runtime. These techniques were evaluated and their performance was compared. From the analysis. The results of this study will encourage us to continue developing other advanced decision trees such as CART.

2.1.4

Title: A Comparative Study on Liver Disease Prediction Using Supervised Machine Learning Algorithms

Author: A.K.M Sazzadur Rahman, F. M. Javed Mehedi Shamrat, Zarrin Tasnim,
Joy Roy, Syed Akhter Hossain

Year : 11, NOVEMBER 2019

Chronic Liver Disease is the leading cause of global death that impacts the massive quantity of humans around the world. This disease is caused by an assortment of elements that harm the liver. For example, obesity, an undiagnosed hepatitis infection, alcohol misuse. Which is responsible for abnormal nerve function, coughing up or vomiting blood, kidney failure, liver failure, jaundice, liver encephalopathy and there are many more. This disease diagnosis is very costly and complicated. Therefore, the goal of this work is to evaluate the performance of different Machine Learning algorithms in order to reduce the high cost of chronic liver disease diagnosis by prediction. In this work, we used six algorithms Logistic

Regression. We just explored some popular supervised machine learning algorithms; more algorithms can be picked to assemble an increasingly precise model of liver disease prediction and performance can be progressively improved. Additionally, this work likewise ready to assume a significant role in health care research and just as restorative focuses to anticipate liver infection.

2.1.5

Title: Performance Evolution of Different Machine Learning Algorithms for Prediction of Liver Disease

Author: Muktevi Sri Venkatesh

Year : December 2019

Liver malady is an overall medical issue that is related with different inconveniences and high mortality. It is of basic significance that illness be recognized before such huge numbers of these lives can be spared. The phases of liver ailment are a significant viewpoint for focused treatment. It is a terribly troublesome undertaking for therapeutic analysts to foresee the disease inside the beginning times on account of sensitive manifestations. Generally, the side effects become evident once it's past the point of no return. To beat this issue, we have liver infection forecast. Liver sickness might be distinguished with incalculable order systems, and these have been classified the utilization forecast of a number highlight and classifier blends. As end, the use of information digging systems for prescient examination is significant in the wellbeing field since it enables us to confront ailments prior and accordingly spare individuals' lives through the expectation of fix. In this work, we utilized a few learning calculations K-Nearest Neighbor, Support Vector Machines, Logistic Regression, Navi Bayes, Random Forest to foresee patients with constant liver disappointment infection, and patients who are not experiencing this illness. Re-enactment results demonstrated that Logistic Regression classifier demonstrated its exhibition in foreseeing with best outcomes regarding precision and least execution time.

2.2 SURVEY WALKTHROUGH

2.2.1 NUMPY

NumPy library is an important foundational tool for studying Machine Learning. Many of its functions are very useful for performing any mathematical or scientific calculation. As it is known that mathematics is the foundation of machine learning, most of the mathematical tasks can be performed using NumPy. NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.

NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numarray into Numeric package. There are many contributors to this open-source project.

Using NumPy, a developer can perform the following operations –

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

NumPy – A Replacement for MatLab

NumPy is often used along with packages like **SciPy** (Scientific Python) and **Matplotlib** (plotting library). This combination is widely used as a

replacement for MatLab, a popular platform for technical computing. However, Python alternative to MatLab is now seen as a more modern and complete programming language.

It is open-source, which is an added advantage of NumPy.

The most important object defined in NumPy is an N-dimensional array type called **ndarray**. It describes the collection of items of the same type. Items in the collection can be accessed using a zero-based index.

Every item in a ndarray takes the same size as the block in the memory. Each element in ndarray is an object of the data-type object (called **dtype**). Any item extracted from ndarray object (by slicing) is represented by a Python object of one of array scalar types.

An instance of ndarray class can be constructed by different array creation routines described later in the tutorial. The basic ndarray is created using an array function in NumPy.

NumPy library lets us store and work on a large amount of dense data effectively and efficiently. In NumPy data is stored in the form of arrays. These arrays are the crux of this library. Arrays in NumPy are similar to python built-in type list. The closest approximate to a numpy array is a C array. This can be attributed to the fact that NumPy is written in c language. But NumPy arrays provide more efficient data storage and operations compared to both.

NumPy. Array

It creates a ndarray from any object exposing an array interface, or from any method that returns an array.

numpy. array (object, dtype = None, copy = True, order = None, subok = False, ndmin = 0)

The **ndarray** object consists of a contiguous one-dimensional segment of computer memory, combined with an indexing scheme that maps each item to a location in the memory block. The memory block holds the elements in row-major order (C style) or a column-major order (FORTRAN or MatLab style). The **ndarray** object consists of a contiguous one-dimensional segment of computer memory, combined with an indexing scheme that maps each item to a location in the memory block. The memory block holds the elements in row-major order (C style) or a column-major order (FORTRAN or MatLab style).

NumPy's arrays are more compact than Python lists -- a list of lists as you describe, in Python, would take at least 20 MB or so, while a NumPy 3D array with single-precision floats in the cells would fit in 4 MB. Access in reading and writing items is also faster with NumPy.

Maybe you don't care that much for just a million cells, but you definitely would for a billion cells -- neither approach would fit in a 32-bit architecture, but with 64-bit builds NumPy would get away with 4 GB or so, Python alone would need at least about 12 GB (lots of pointers which double in size) -- a much costlier piece of hardware.

The difference is mostly due to "indirectness" -- a Python list is an array of pointers to Python objects, at least 4 bytes per pointer plus 16 bytes for even the smallest Python object (4 for type pointer, 4 for reference count, 4 for value -- and the memory allocators rounds up to 16). A NumPy array is an array of uniform values -- single-precision numbers takes 4 bytes each, double-precision ones, 8 bytes. Less flexible, but you pay substantially for the flexibility of standard Python lists.

NumPy is not just more efficient; it is also more convenient. You get a lot of vector and matrix operations for free, which sometimes allow one to avoid unnecessary work. And they are also efficiently implemented.

ADVANTAGES:

- The core of Numpy is its arrays. One of the main advantages of using Numpy arrays is that they take less memory space and provide better runtime speed when compared with similar data structures in python (lists and tuples).
- Numpy support some specific scientific functions such as linear algebra. They help us in solving linear equations.
- Numpy support vectorized operations, like elementwise addition and multiplication, computing Kronecker product, etc. Python lists fail to support these features.
- It is a very good substitute for MATLAB, OCTAVE, etc as it provides similar functionalities and supports with faster development and less mental overhead (as python is easy to write and comprehend)
- NumPy is very good for data analysis.

DISADVANTAGES:

- **Using “nan” in Numpy:** “Nan” stands for “not a number”. It was designed to address the problem of missing values. NumPy itself supports “nan” but lack of cross-platform support within Python makes it difficult for the user. That’s why we may face problems when comparing values within the Python interpreter.
- **Require a contiguous allocation of memory:** Insertion and deletion operations become costly as data is stored in contiguous memory locations as shifting it requires shifting.

2.2.2 PANDAS

Pandas is an open – source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and times series. This library is built on

top of the NumPy library. Pandas is fast and it has high performance and productivity for users.

Pandas are generally used for data science this is because pandas are used in conjunction with other libraries that are used for data science. It is built on the top of the NumPy library which means that a lot of structures of NumPy are used or replicated in Pandas. The data produced by Pandas are often used as input for plotting functions of Matplotlib, statistical analysis in SciPy, machine learning algorithms in SciKit-learn.

Pandas program can be run from any text editor but it is recommended to use Jupyter notebook for this as Jupyter given the ability to execute code in a particular cell rather than executing the entire file. Jupyter also provides an easy way to visualize pandas' data frames and plots.

DATAFRAMES:

Pandas is mainly used for data analysis and associated manipulation of tabular data in Dataframes. Pandas allows importing data from various file formats such as comma-separated values, JSON, Parquet, SQL database tables or queries, and Microsoft Excel. Pandas allows various data manipulation operations such as merging, reshaping, selecting, as well as data cleaning, and data wrangling features. The development of pandas introduced into Python many comparable features of working with Dataframes that were established in the R programming language. The pandas library is built upon another library NumPy, which is oriented to efficiently working with arrays instead of the features of working on Dataframes.

ADVANTAGES:

- Fast and efficient for manipulating and analyzing data.
- Data from different file objects can be loaded.
- Easy handling of missing data in floating point as well as non – floating point data.

- Size mutability: columns can be inserted and deleted from DataFrame and higher dimensional objects.
- Data set merging and joining.
- Flexible reshaping and pivoting of data sets.
- Provides time-series functionality.
- Powerful group by functionality for performing split-apply-combine operations on data sets.

DISADVANTAGES:

- A Complex syntax which is not always in line with python when using advanced levels of Pandas.
- Learning curve.
- Poor Documentation
- Poor 3D matrix Compatibility.

2.2.3 MATPLOTLIB

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of Matplotlib. Matplotlib is a NumFOCUS fiscally sponsored project.

Matplotlib 2.0.x supports Python versions 2.7 through 3.10. Python 3 support started with Matplotlib 1.2. Matplotlib 1.4 is the last version to support Python 2.6. Matplotlib has pledged not to support Python 2 past 2020 by signing the Python 3 Statement.

Pyplot is a Matplotlib module which provides a MATLAB-like interface. Matplotlib is designed to be as usable as MATLAB, with the ability to use Python, and the advantage of being free and open-source.

As a python developer, one should be making use of the Pyplot module within the matplotlib library. Pyplot is an Application Programming Interface that consists of functions/ methods which help process data for visualization. The plot is quick when it comes to generating visual graphs. In addition, its similarities with MATLAB make working easy for individuals who have prior experience working with MATLAB and vice versa.

MATPLOTLIB TOOLKITS:

Several toolkits are available which extend Matplotlib functionality. Some are separate downloads; others ship with the Matplotlib source code but have external dependencies.

- Basemap: map plotting with various map projections, coastlines, and political boundaries
- Cartopy: a mapping library featuring object-oriented map projection definitions, and arbitrary point, line, polygon and image transformation capabilities. (Matplotlib v1.2 and above)
- Excel tools: utilities for exchanging data with Microsoft Excel
- GTK tools: interface to the GTK library
- Qt interface
- Mplot3d: 3-D plots
- Natgrid: interface to the natgrid library for gridding irregularly spaced data.
- tikzplotlib: export to Pgfplots for smooth integration into LaTeX documents (formerly known as *matplotlib2tikz*)
- Seaborn: provides an API on top of Matplotlib that offers sane choices for plot style and color defaults, defines simple high-level functions for common statistical plot types, and integrates with the functionality provided by Pandas.

ADVANTAGES:

- Simple and easy to grasp for beginners.
- Easier to use for people who have had prior experience with MATLAB or other graph plotting tools.
- It provides high-quality images and plots in various formats such as png, pdf, pgf, etc.
- Provides control to various elements of a figure such as DPI, figure colour, figure size.
- It uses Python: Python is a very interesting language for scientific purposes (it's interpreted, high-level, easy to learn, easily extensible, and has a powerful standard library) and is now used by major institutions such as NASA, JPL, Google, DreamWorks, Disney, and many more.
- It's open source, so no license to pay: This makes it very appealing for professors and students, who often have a low budget.
- It's a real programming language: The MATLAB language (while being Turing-complete) lacks many of the features of a general-purpose language like Python.
- It's much more complete: Python has a lot of external modules that will help us perform all the functions we need to. So it's the perfect tool to acquire data, elaborate the data, and then plot the data.
- It's very customizable and extensible: Matplotlib can fit every use case because it has a lot of graph types, features, and configuration options.
- It's integrated with LaTeX markup: This is really useful when writing scientific papers.
- It's cross-platform and portable: Matplotlib can run on Linux, Windows, Mac OS X, and Sun Solaris (and Python can run on almost every architecture available).

Apart from these, python matplotlib has some disadvantages. Some of them are listed below:

DISADVANTAGES:

- They are heavily reliant on other packages, such as NumPy.
- It only works for python, so it is hard or impossible to be used in languages other than python. (But it can be used from Julia via PyPlot package).

2.2.4 SKLEARN (SciKit-Learn)

Scikit-learn is probably the most useful library for machine learning in Python. The sklearn library contains a lot of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction.

Please note that sklearn is used to build machine learning models. It should not be used for reading the data, manipulating and summarizing it. There are better libraries for that (e.g., NumPy, Pandas etc.)

Scikit-learn is largely written in Python, and uses NumPy extensively for high-performance linear algebra and array operations. Furthermore, some core algorithms are written in Cython to improve performance. Support vector machines are implemented by a Cython wrapper around LIBSVM; logistic regression and linear support vector machines by a similar wrapper around LIBLINEAR. In such cases, extending these methods with Python may not be possible.

Scikit-learn integrates well with many other Python libraries, such as Matplotlib and plotly for plotting, NumPy for array vectorization, Pandas data frames, SciPy, and many more.

COMPONENTS OF SciKit – LEARN:

Scikit-learn comes loaded with a lot of features. Here are a few of them:

- **Supervised learning algorithms:** Think of any supervised machine learning algorithm you might have heard about and there is a very high chance that it is part of scikit-learn. Starting from Generalized linear models (e.g Linear Regression), Support Vector Machines (SVM), Decision Trees to Bayesian methods – all of them are part of scikit-learn toolbox. The spread of machine learning algorithms is one of the big reasons for the high usage of scikit-learn. I started using scikit to solve supervised learning problems and would recommend that to people new to scikit / machine learning as well.
- **Cross-validation:** There are various methods to check the accuracy of supervised models on unseen data using sklearn.
- **Unsupervised learning algorithms:** Again, there is a large spread of machine learning algorithms in the offering – starting from clustering, factor analysis, principal component analysis to unsupervised neural networks.
- **Various toy datasets:** This came in handy while learning scikit-learn. I had learned SAS using various academic datasets (e.g., IRIS dataset, Boston House prices dataset). Having them handy while learning a new library helped a lot.
- **Feature extraction:** Scikit-learn for extracting features from images and text (e.g., Bag of words)
- **Feature selection** – It is used to identify useful attributes to create supervised models.
- **Open Source** – It is open-source library and also commercially usable under BSD license.
- **Clustering** – This model is used for grouping unlabeled data.
- **Dimensionality Reduction** – It is used for reducing the number of attributes in data which can be further used for summarisation, visualisation and feature selection.
- **Ensemble methods** – As name suggest, it is used for combining the predictions of multiple supervised models.

Pros and cons of scikit-learn

ADVANTAGES:

- The library is distributed under the BSD license, making it free with minimum legal and licensing restrictions.
- It is easy to use.
- The scikit-learn library is very versatile and handy and serves real-world purposes like the prediction of consumer behavior, the creation of neuroimages, etc.
- Scikit-learn is backed and updated by numerous authors, contributors, and a vast international online community.
- The scikit-learn website provides elaborate API documentation for users who want to integrate the algorithms with their platforms.

DISADVANTAGES:

- It is not the best choice for in-depth learning.
- Scikit-learn is a fantastic tool for exploring, transforming and classifying data. But it is optimized for learning algorithms, such as Support Vector Machines (SVMs), logistic regression, and Linear Discriminant Analysis (LDA).
- It is not optimized for graph algorithms, and it is not very good at string processing. For example, scikit-learn does not provide a built-in way to produce a simple word cloud.
- Scikit-learn doesn't have a strong linear algebra library, hence scipy and NumPy are used. It doesn't contain a plotting library, but it allows to use different plotting libraries.

2.3 PROJECT GOALS

- Exploration data analysis of variable identification
 - Loading the given Liver Failure dataset

- Import required libraries packages
- Analyze the general properties
- Find duplicate and missing values
- Checking unique and count values
- Uni-variate data analysis
 - Rename, add data and drop the data
 - To specify data type
- Exploration data analysis of bi-variate and multi-variate
 - Plot diagram of pairplot, heatmap, bar chart and Histogram
- Method of Outlier detection with feature engineering
 - Pre-processing the given dataset
 - Splitting the test and training dataset
 - Comparing the Decision tree and Logistic regression model and random forest etc
- Comparing algorithm to predict the result
 - Based on the best accuracy

CHAPTER 3

AIM AND SCOPE

3.1 AIM

The goal is to develop a machine learning model for Predicting Liver Failure using, to potentially replace the updatable supervised machine learning classification models by predicting results in the form of best accuracy by comparing supervised algorithm.

3.2 SCOPE

The scope of this project is to investigate a liver dataset to find the diabetes using machine learning technique.

To find the liver failure is based on the people's health actual conditions it's not easy one to predict, but using the AI machine learning technique compare with different algorithms we can build a predictive model. Using metrics (accuracy, precision, recall, etc..) we can validate the model accuracy.

3.3 SYSTEM STUDY

Liver is largest internal organ of the body. It plays a significant role in transfer of blood throughout our body. Numerous liver diseases are accompanied by jaundice caused by augmented levels of bilirubin in the body.

Bilirubin is the result of degradation of hemoglobin of dead red blood cells which are normally removed by the liver and excreted via bile.

In hepatitis, inflammation of the liver, is caused by different viruses, but also some toxic substances, autoimmune diseases and inherited conditions;

The levels of most chemicals in our blood are regulated by the liver. It helps in metabolism of the alcohol, drugs and destroys toxic substances. Liver can be infected by parasites, viruses which cause inflammation and diminish its function.

It has the potential to maintain the customary function, even when a part of it is damaged. However, it is important to diagnose liver disease early which can increase the patient's survival rate.

Expert physicians are required for various examination tests to diagnose the liver disease, but it cannot assure the correct diagnosis.

- Define a problem
- Preparing data
- Evaluating algorithms
- Improving results
- Predicting results

The best way to really come to terms with a new platform or tool is to work through a machine learning project end-to-end and cover the key steps.

Namely, from loading data, summarizing data, evaluating algorithms and making some predictions.

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

- Installing the Python anaconda platform.
- Loading the dataset.
- Summarizing the dataset.
- Visualizing the dataset.
- Evaluating some algorithms.

- Making some predictions.

3.4 EXISTING SYSTEM

They proposed a B-mode ultrasound (BUS) imaging tool for diagnosis of liver failures.

They tried to improve the BUS-based computer aided diagnosis for liver failures by transferring knowledge from the multi-view CEUS images, including the arterial phase, portal venous phase, and delayed phase, respectively.

A support vector machine plus (SVM+), a specifically designed transfer learning (TL) classifier for paired data with shared labels was adopted for this supervised TL.

A nonparallel hyper plane based SVM+ (NHSVM+) was also first proposed to improve the TL performance by transferring the per-class knowledge from source domain to the corresponding target domain.

The patients with malignant liver failure and some of the patients with benign tumors were confirmed by biopsy or surgical specimens, and the remainder were annotated as benign cases with the follow-up for more than one year without significant changes.

During the training stage, features are first extracted from the BUS image and its corresponding three CEUS images from the AP, PVP, and DP, respectively.

All the hand-crafted features are fed to the proposed MKL-NHSVM+ to train the diagnosis model for liver failures, where BUS image is the diagnosis modality and three CEUS images.

Disadvantages:

1. It takes more time to make the transfer learning if we want to change some features and train the model.
2. Support vector machine algorithm only used and no comparison of algorithm is done for evaluating best model.

3.5 PROPOSED SYSTEM

Liver failure is a condition or disease that happens when normal cells in the liver become abnormal in appearance and behavior.

The failure cells can then become destructive to adjacent normal tissues, and can spread both to other areas of the liver and to organs outside the liver.

The proposed method is a machine learning model based on the past data of liver failure like the features and target column is identified first using our domain knowledge related to health care.

Then dataset is viewed for better understanding of features and then the dataset is split into two parts normally in 7:3 ratio where the data is used for training and testing.

The algorithm is applied on the trained data to get better understanding of the features and a classification model is built based on their learning and different algorithms are compared and performance is measured and compared using their performance metrics.

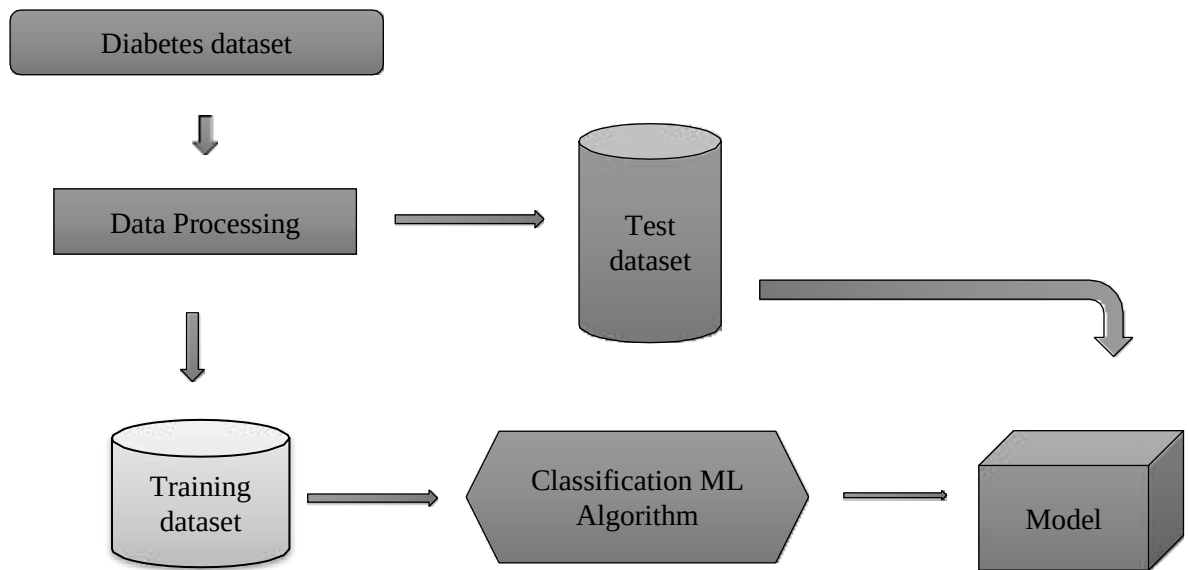


FIG: 3.1 SYSTEM OVERVIEW DIAGRAM

Advantages:

1. Automated process and the deployment can also be done in desktop application.
2. Algorithms are compared and best model is evaluated for better prediction.

3.5.1 SYSTEM ARCHITECTURE

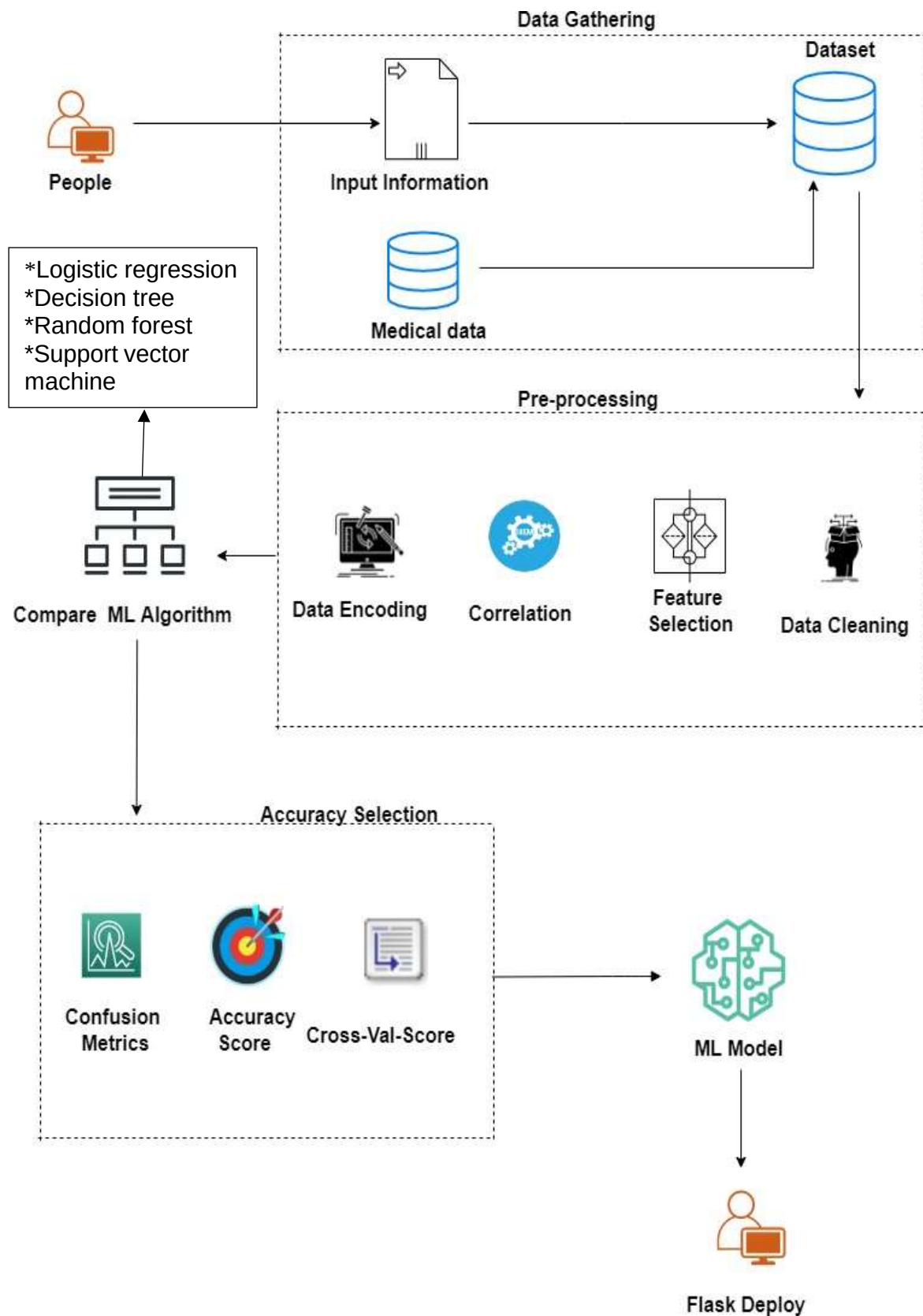


FIG: 3.2 SYSTEM ARCHITECTURE

3.5.2 WORKING PROCESS

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

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

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

- It will force you to install and start the Python interpreter (at the very least).
- It will give you a bird's eye view of how to step through a small project.
- It will give you confidence, maybe to go on to your own small projects.

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

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

The best way to really come to terms with a new platform or tool is to work

through a machine learning project end-to-end and cover the key steps. Namely, from loading data, summarizing data, evaluating algorithms and making some predictions.

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

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

Making some predictions.

CHAPTER 4

METHODOLOGY

4.1 DATA GATHERING

Information Gathering is the initial step of the AI life cycle. The objective of this progression is to distinguish and acquire all information related issues. In this Progression, we need to distinguish the distinctive information sources, as information can be gathered from different sources such as files, database, internet, or mobile gadgets. It is perhaps the main strides of the existence cycle. The amount and nature of the gathered information will decide the proficiency of the yield. The more will be the information, the more exact will be the expectation.

4.1.1 PREPARING DATASET

This data set contains 416 liver patient records and 167 non liver patient records collected from North East of Andhra Pradesh, India. The "Dataset" column is a class label used to divide groups into liver patient (liver disease) or not (no disease). This data set contains 441 male patient records and 142 female patient records.

Any patient whose age exceeded 89 is listed as being of age "90".

Columns:

- Age of the patient
- Gender of the patient
- Total Bilirubin:
 - Bilirubin is an orange-yellow pigment that occurs normally when part of your red blood cells breaks down. A bilirubin test

measures the amount of bilirubin in your blood. It's used to help find the cause of health conditions like jaundice, anemia, and liver disease.

- Direct Bilirubin:
 - Bilirubin attached by the liver to glucuronic acid, a glucose-derived acid, is called direct or conjugated, bilirubin. Bilirubin not attached to glucuronic acid is called indirect.
- Alkaline Phosphatase:
 - Alkaline phosphatase (ALP) is an enzyme in a person's blood that helps break down proteins. Using an ALP test, it is possible to measure how much of this enzyme is circulating in a person's blood.
- Alanine Aminotransferase:
 - Alanine aminotransferase (ALT) is an enzyme found primarily in the liver and kidney. ALT is increased with liver damage and is used to screen for and/or monitor liver disease.
- Aspartate Aminotransferase:
 - AST (aspartate aminotransferase) is an enzyme that is found mostly in the liver, but also in muscles. When your liver is damaged, it releases AST into your bloodstream. An AST blood test measures the amount of AST in your blood. The test can help your health care provider diagnose liver damage or disease.
- Total Proteins:
 - Albumin and globulin are two types of protein in your body. The total protein test measures the total amount of albumin and globulin in your body.
- Albumin
- Albumin and Globulin Ratio
- Dataset: field used to split the data into two sets (patient with liver disease, or no disease)

Data set used in the project:

	A	B	C	D	E	F	G	H	I	J	K
	Age	Gender	Total_Bilir	Direct_Bili	Alkaline_P	Alamine_A	Aspartate	Total_Prot	Albumin	Albumin_a Dataset	
2	65	Female	0.7	0.1	187	16	18	6.8	3.3	0.9	1
3	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	1
4	62	Male	7.3	4.1	490	60	68	7	3.3	0.89	1
5	58	Male	1	0.4	182	14	20	6.8	3.4	1	1
6	72	Male	3.9	2	195	27	59	7.3	2.4	0.4	1
7	46	Male	1.8	0.7	208	19	14	7.6	4.4	1.3	1
8	26	Female	0.9	0.2	154	16	12	7	3.5	1	1
9	29	Female	0.9	0.3	202	14	11	6.7	3.6	1.1	1
10	17	Male	0.9	0.3	202	22	19	7.4	4.1	1.2	2
11	55	Male	0.7	0.2	290	53	58	6.8	3.4	1	1
12	57	Male	0.6	0.1	210	51	59	5.9	2.7	0.8	1
13	72	Male	2.7	1.3	260	31	56	7.4	3	0.6	1
14	64	Male	0.9	0.3	310	61	58	7	3.4	0.9	2
15	74	Female	1.1	0.4	214	22	30	8.1	4.1	1	1

FIG: 4.1 DATASET USED

4.1.2 DATA PREPARATION

Subsequent to gathering the information, we need to set it up for additional means. Information readiness is a stage where we put our information into an appropriate place and set it up to use in our AI preparing. In this progression, first, we set up all information, and afterward randomize the requesting of information. This progression can be additionally partitioned into two cycles.

- **DATA EXPLORATION:**

It is utilized to comprehend the idea of information that we need to work with. We need to comprehend the attributes, organization, and nature of data. A better comprehension of information prompts a powerful result. In this, we discover Correlations, general patterns, and anomalies.

1. Missing Values
2. Duplicate data

3. Invalid data
4. Noise

So, we use various filtering techniques to clean the data. It is mandatory to detect and remove the above issues because it can negatively affect the quality of the outcome.

- **DATA PRE-PROCESSING**

Presently the stage is pre-handling of information for its investigation.

4.2 DATA PRE-PROCESSING

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

However, in real-world scenarios, to work with samples of data that may not be a true representative of the population of given dataset.

To finding the missing value, duplicate value and description of data type whether it is float variable or integer. The data provides an unbiased evaluation of a model fit on the training dataset while tuning model hyper parameters.

	Age	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphatase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Proteins	Albumin	AI
count	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	
mean	44.746141	3.298799	1.486106	290.576329	80.713551	109.910806	6.483190	3.141852	
std	16.189833	6.209522	2.808498	242.937989	182.820356	288.918529	1.085451	0.795519	
min	4.000000	0.400000	0.100000	63.000000	10.000000	10.000000	2.700000	0.900000	
25%	33.000000	0.800000	0.200000	175.500000	23.000000	25.000000	5.800000	2.600000	
50%	45.000000	1.000000	0.300000	208.000000	35.000000	42.000000	6.600000	3.100000	
75%	58.000000	2.600000	1.300000	298.000000	60.500000	67.000000	7.200000	3.800000	
max	90.000000	75.000000	19.700000	2110.000000	2000.000000	4529.000000	9.600000	5.500000	

FIG: 4.2 UNBIASED EVALUATION

The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set is used to evaluate a given model, but this is for frequent evaluation. It as machine learning engineers uses this data to fine-tune the model hyper parameters.

Data collection, data analysis, and the process of addressing data content, quality, and structure can add up to a time-consuming to-do list. During the process of data identification, it helps to understand your data and its properties; this knowledge will help you choose which algorithm to use to build your model.

4.2.1 DATA WRANGLING

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

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

	Age	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Prot
Age	1.000000	0.011000	0.006784	0.078878	-0.087799	-0.020499	-0.19
Total_Bilirubin	0.011000	1.000000	0.874481	0.205739	0.213375	0.237323	-0.00
Direct_Bilirubin	0.006784	0.874481	1.000000	0.234008	0.233180	0.257022	0.00
Alkaline_Phosphotase	0.078878	0.205739	0.234008	1.000000	0.124777	0.166580	-0.02
Alamine_Aminotransferase	-0.087799	0.213375	0.233180	0.124777	1.000000	0.791862	-0.04
Aspartate_Aminotransferase	-0.020499	0.237323	0.257022	0.166580	0.791862	1.000000	-0.02
Total_Protiens	-0.186248	-0.007906	0.000033	-0.027062	-0.042432	-0.025751	1.00
Albumin	-0.264211	-0.222087	-0.228409	-0.183419	-0.028658	-0.084915	0.78
Albumin_and_Globulin_Ratio	-0.216408	-0.206267	-0.200125	-0.234186	-0.002375	-0.070040	0.23
Result	-0.133164	-0.220218	-0.246273	-0.183363	-0.183117	-0.151834	0.03

FIG: 4.3 MISSING DATA

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

- User forgot to fill in a field.
- Data was lost while transferring manually from a legacy database.
- There was a programming error.
- Users chose not to fill out a field tied to their beliefs about how the results would be used or interpreted.

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin	Albumi
0	65	0	0.7	0.1	187	16	18	6.8	3.3	
1	62	1	10.9	5.5	699	64	100	7.5	3.2	
2	62	1	7.3	4.1	490	60	68	7.0	3.3	
3	58	1	1.0	0.4	182	14	20	6.8	3.4	
4	72	1	3.9	2.0	195	27	59	7.3	2.4	

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

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

- Checking unique values of data frame
- Checking count values of data frame
- Rename and drop the given data frame
- To specify the type of values
- To create extra columns

4.3 DATA VISUALIZATION

Importing the library packages with loading given dataset. To analyzing the variable identification by data shape, data type and evaluating the missing values, duplicate values.

A validation dataset is a sample of data held back from training your model that is used to give an estimate of model skill while tuning model's and procedures that you can use to make the best use of validation and test datasets when evaluating your models.

Data cleaning / preparing by rename the given dataset and drop the column etc. to analyze the uni-variate, bi-variate and multi-variate process. The steps and techniques for data cleaning will vary from dataset to dataset. The primary goal of data cleaning is to detect and remove errors and anomalies to increase the value of data in analytics and decision making.

Data visualization is an important skill in applied statistics and machine learning. Statistics does indeed focus on quantitative descriptions and estimations of data. Data visualization provides an important suite of tools for gaining a qualitative understanding.

This can be helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupt data, outliers, and much more. With a little domain knowledge, data visualizations can be used to express and demonstrate key relationships in plots and charts that are more visceral and stakeholders than measures of association or significance.

Data visualization and exploratory data analysis are whole fields themselves and it will recommend a deeper dive into some the books mentioned at the end.



FIG: 4.4 TOTAL MALE AND FEMALE PATIENTS

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

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

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.

To achieving better results from the applied model in Machine Learning method of the data has to be in a proper manner. Some specified Machine Learning model needs information in a specified format, for example, Random Forest algorithm does not support null values. Therefore, to execute random forest algorithm null values

have to be managed from the original raw data set.

And another aspect is that data set should be formatted in such a way that more than one Machine Learning and Deep Learning algorithms are executed in given dataset.

We can clearly see in the output as well as in the graph that, it is an **imbalanced dataset**, any patients diagnosed with liver disease are **higher** compared to the ones who are not diagnosed.

It can be clearly seen in the graph that, number of patient suffering from liver disease are **higher in males than in females**.

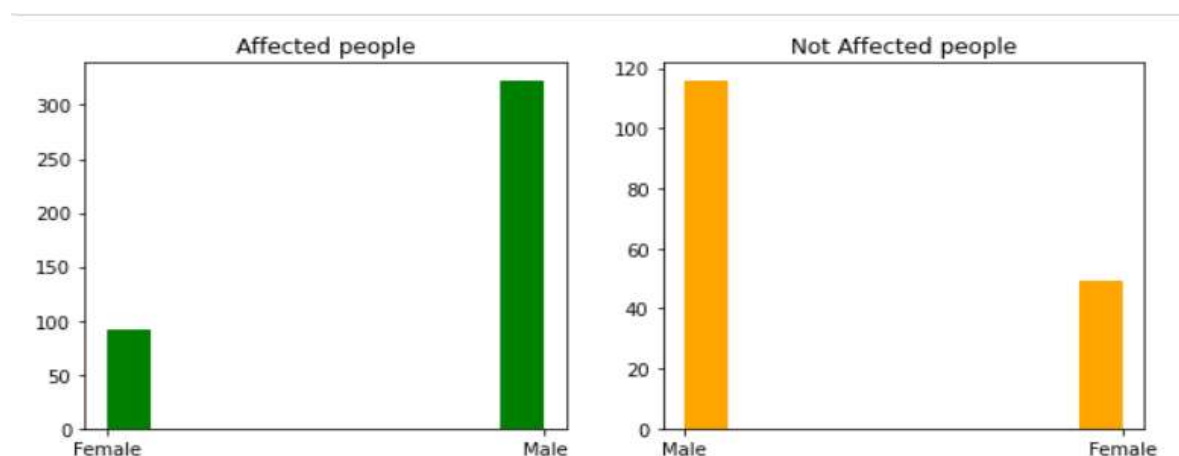


FIG: 4.5 LIVER FAILURE BY GENDER

Now with the help of the below plot we can find out that, **Total_protiens** and **Albumin features** are in positive **regressive nature**, with some **outliers**.

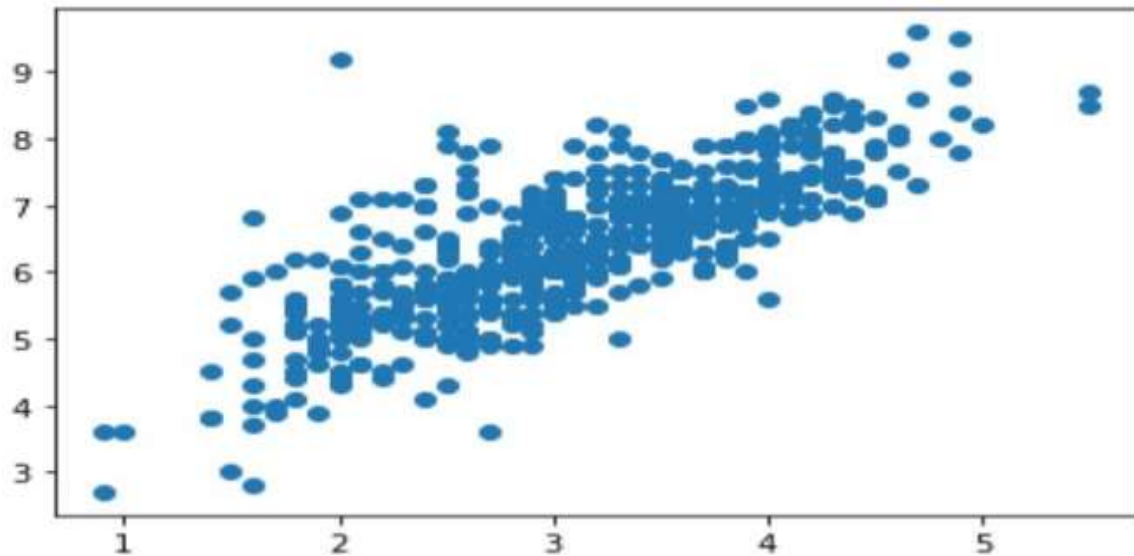


FIG: 4.6 VISUALIZATION OF TOTAL PROTEINS AND ALBUMIN

We can clearly see that the below scatter matrix plot can be used to clearly visualize between the same category. Different matrix can be obtained for all the categories which makes it even easier

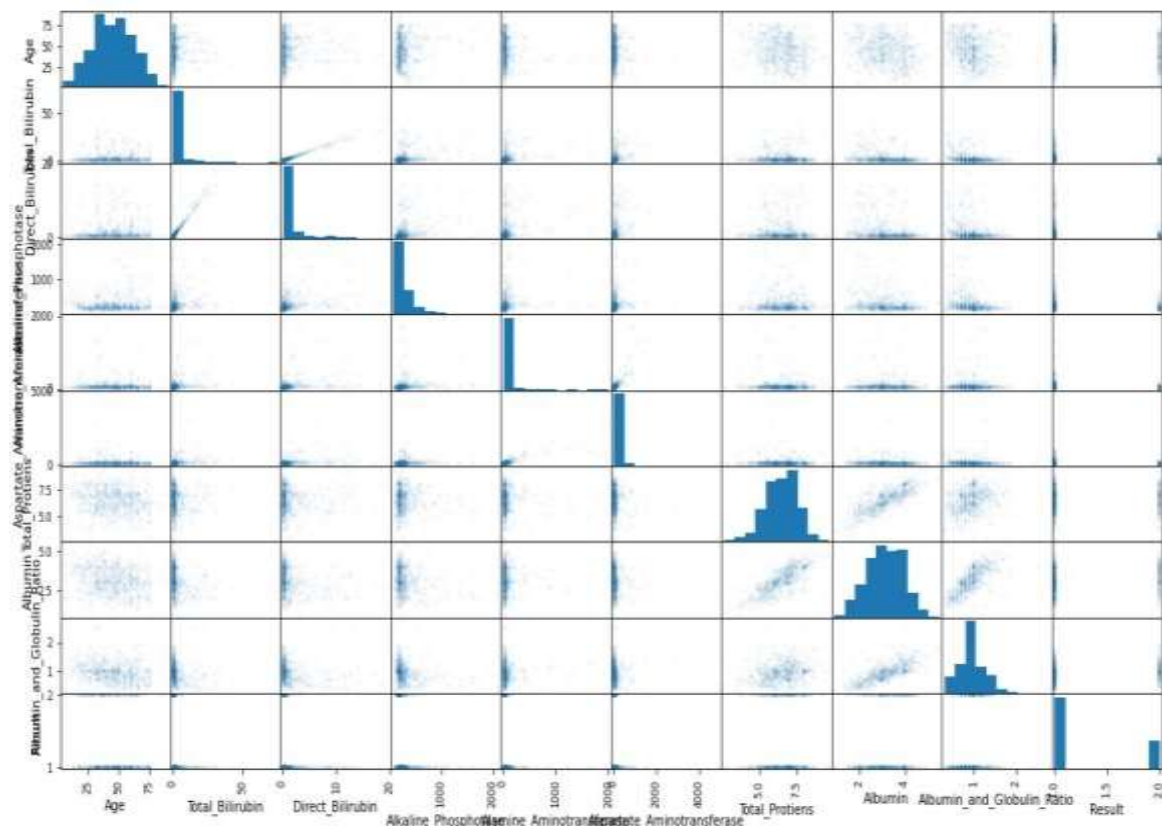


FIG: 4.7 CORRELATION BETWEEN DATA

The Heatmap is shown in FIG: 4.8 appear to have some correlated parameters. Some of these columns have a low correlation. Therefore, we omitted some of the features for better prediction of liver disease.

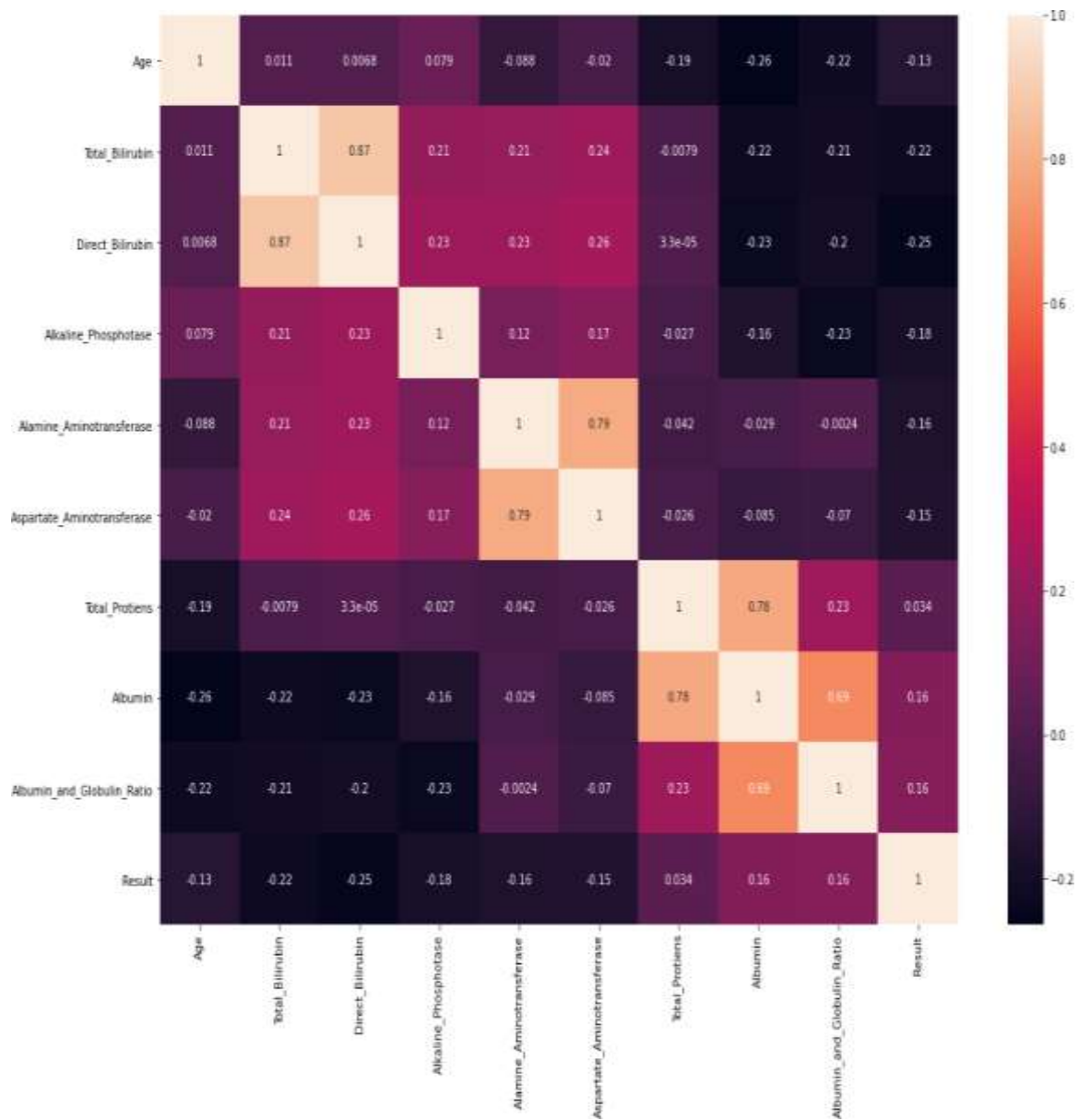


FIG: 4.8 CORRELATION HEATMAP

4.4 DATA ANALYSIS

Presently the cleaned and arranged information is given to the examination step. This progression includes: Selection of analytical techniques:

1. Building models
2. Review the Result

The point of this progression is to assemble an AI model to break down the information utilizing different insightful methods and survey the result.

It begins with the assurance of the sort of the issues, where we select the AI strategies, for example, classification i.e., calculated relapse, choice tree, arbitrary timberland and so on, at that point fabricate the model utilizing arranged information and assess the model. Consequently, in this progression, we take the information and use AI Calculations to construct the model

4.4.1 DATA COLLECTION

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

BUILDING THE CLASSIFICATION MODEL

The predicting the Diabetes disease problem, ML algorithms prediction model is effective because of the following reasons: It provides better results in classification problem.

- It is strong in preprocessing outliers, irrelevant variables, and a mix of continuous, categorical and discrete variables.
- It produces out of bag estimate error which has proven to be unbiased in many tests and it is relatively easy to tune with.

4.4.2 CONSTRUCTION OF A PREDICTIVE MODEL

Machine learning needs data gathering have lot of past data. Data gathering

have sufficient historical data and raw data. Before data pre-processing, raw data can't be used directly. It's used to preprocess then, what kind of algorithm with model. Training and testing this model working and predicting correctly with minimum errors. Tuned model involved by tuned time to time with improving the accuracy.

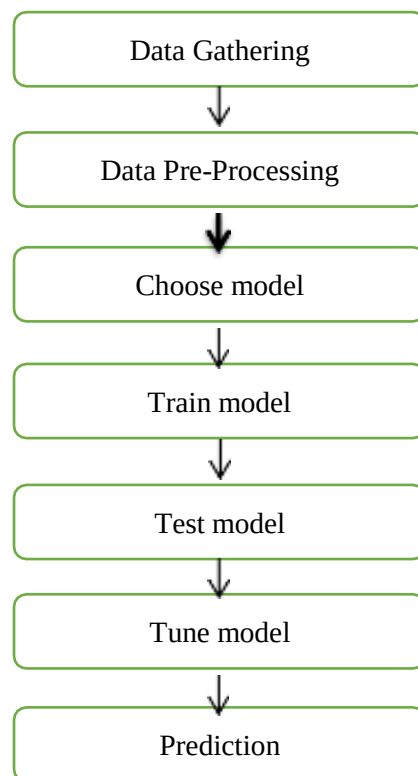


FIG: 4.9 PROCESS OF DATAFLOW DIAGRAM

LIST OF MODULES:

- Data Pre-processing
- Data Analysis of Visualization
- Comparing Algorithm with prediction in the form of best accuracy result
- Deployment Using Flask

4.4.3 PROJECT REQUIREMENTS

General:

Requirements are the basic constraints that are required to develop a system. Requirements are collected while designing the system. The following are the requirements that are to be discussed.

1. Functional requirements
2. Non-Functional requirements
3. Environment requirements
 - A. Hardware requirements
 - B. software requirements

2 Non-Functional Requirements:

Process of functional steps,

1. Problem defines
2. Preparing data
3. Evaluating algorithms
4. Improving results
5. Prediction the result

4.4.4 Environmental Requirements:

1. Software Requirements:

Operating System : Windows
Tool : Anaconda with Jupyter Notebook

2. Hardware requirements:

Processor : Pentium IV/III
Hard disk : minimum 80 GB
RAM : minimum 2 GB

4.5 WORKFLOW PROCESS

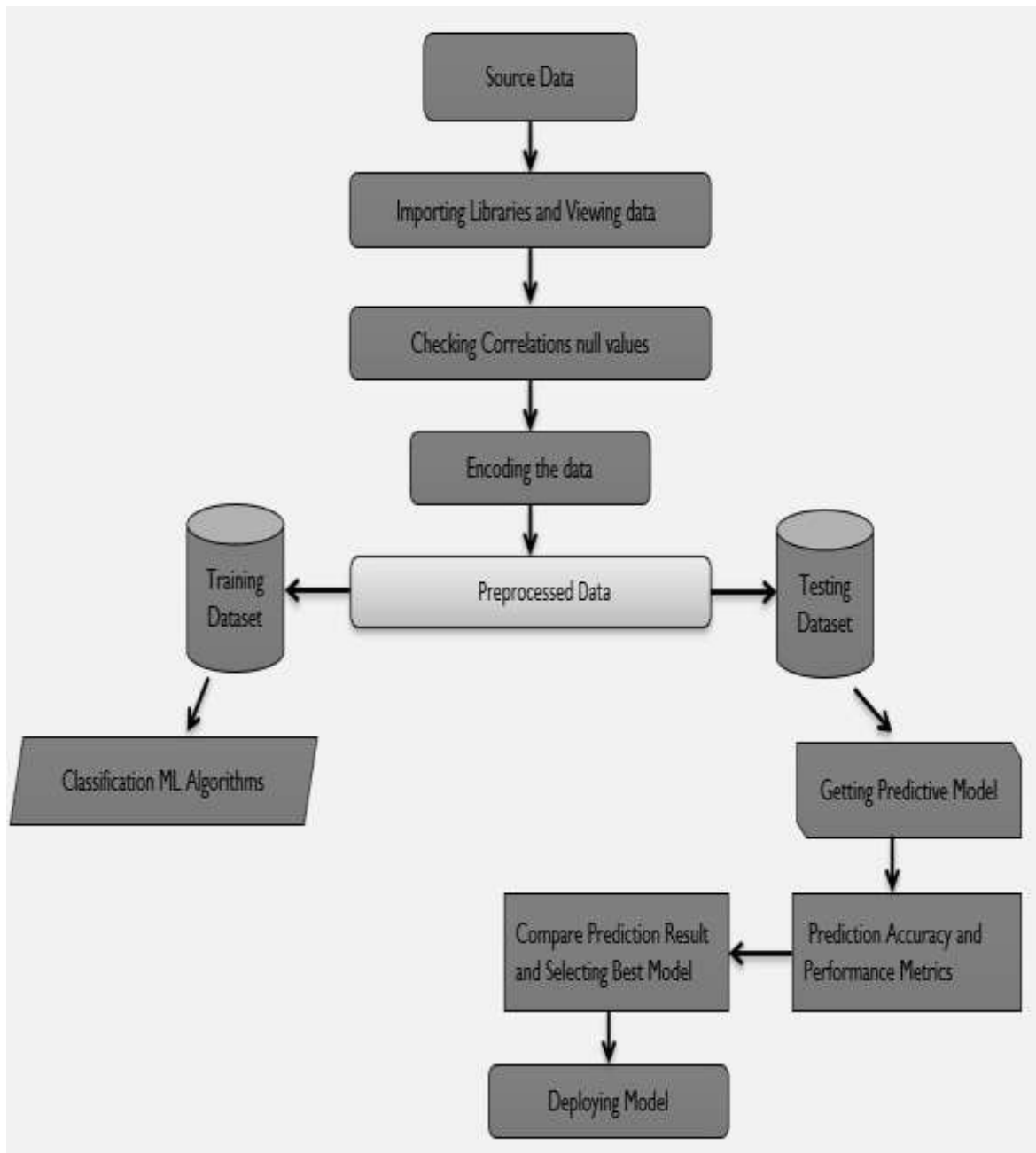


FIG: 4.10 WORKFLOW DIAGRAM

4.5.1 USECASE DIAGRAM

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

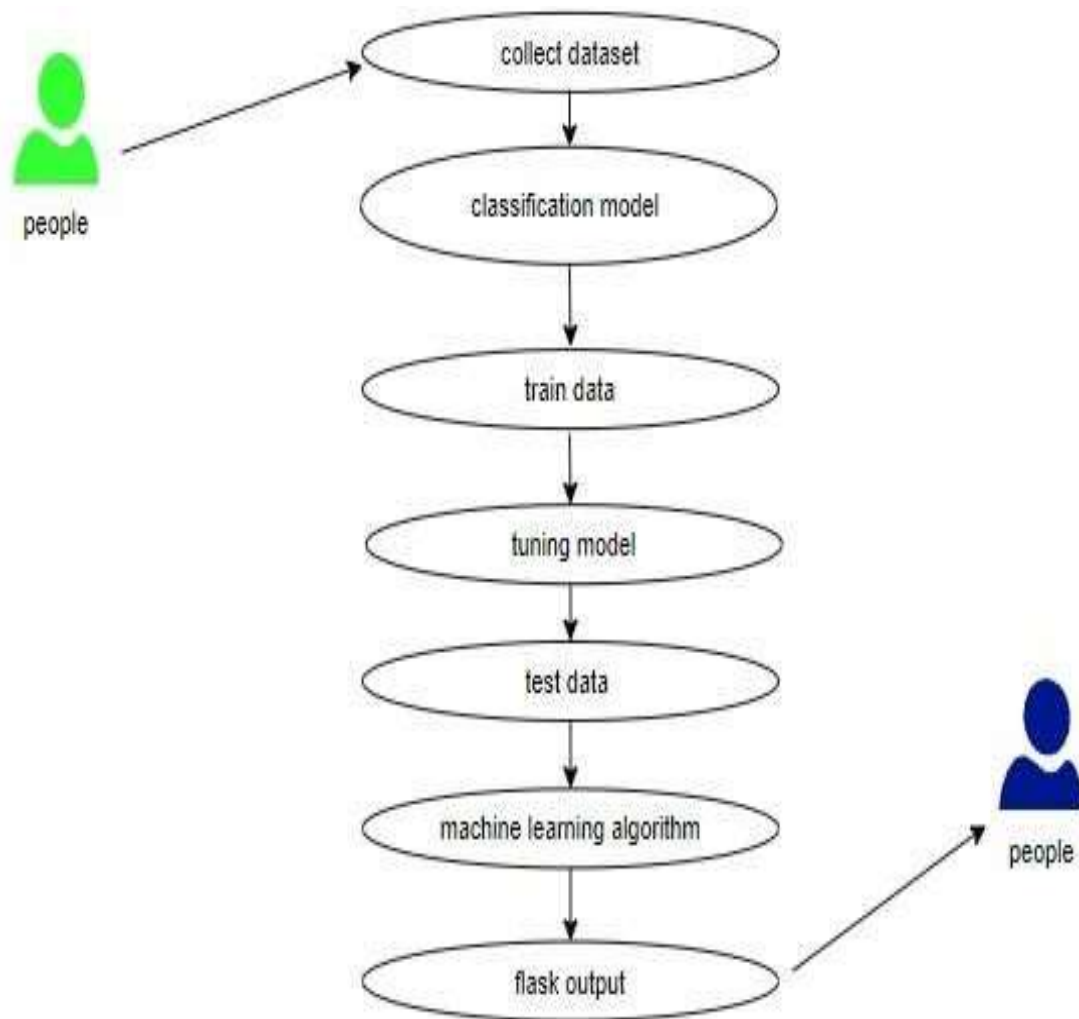


FIG: 4.11 USECASE DIAGRAM

4.5.2 CLASS DIAGRAM

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So, a collection of class diagrams represents the whole system. The name of the class diagram should be meaningful to describe the aspect of the system.

Each element and their relationships should be identified in advance Responsibility (attributes and methods) of each class should be clearly identified for each class minimum number of properties should be specified and because,

unnecessary properties will make the diagram complicated. Use notes whenever required to describe some aspect of the diagram and at the end of the drawing it should be understandable to the developer/coder.

Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

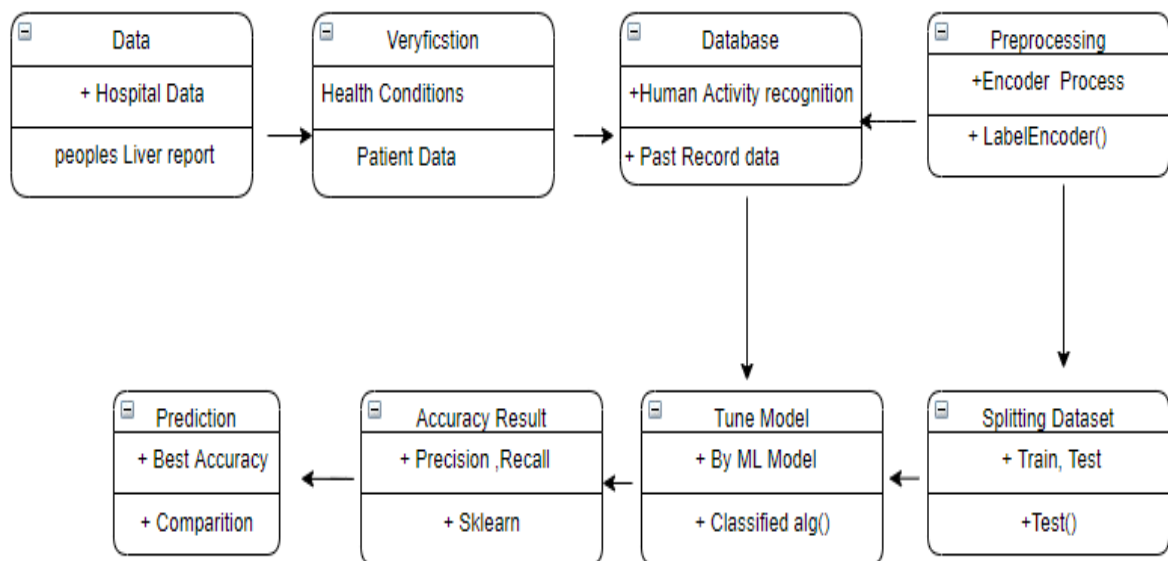


FIG: 4.12 CLASS DIAGRAM

4.5.3 ACTIVITY DIAGRAM

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques.

The only missing thing in activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is some time considered

as the flow chart. Although the diagrams look like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single.

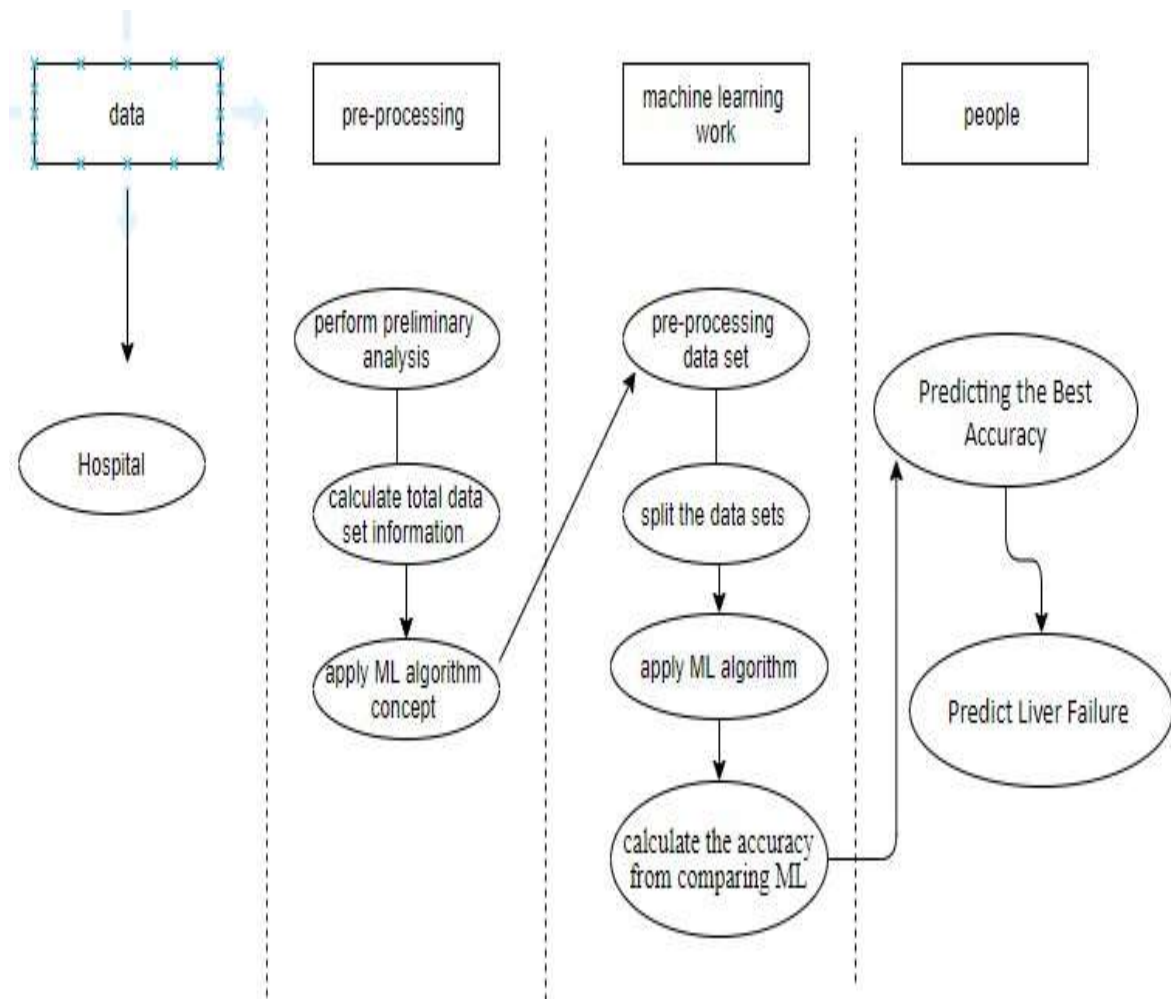


FIG: 4.13 ACTIVITY DIAGRAM

4.5.4 SEQUENCE DIAGRAM

Sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modeling, which focuses on identifying the behavior within your system.

Other dynamic modeling techniques include activity diagramming, communication diagramming, timing diagramming, and interaction overview diagramming. Sequence diagrams, along with class diagrams and physical data models are in my opinion the most important design-level models for modern business application development.

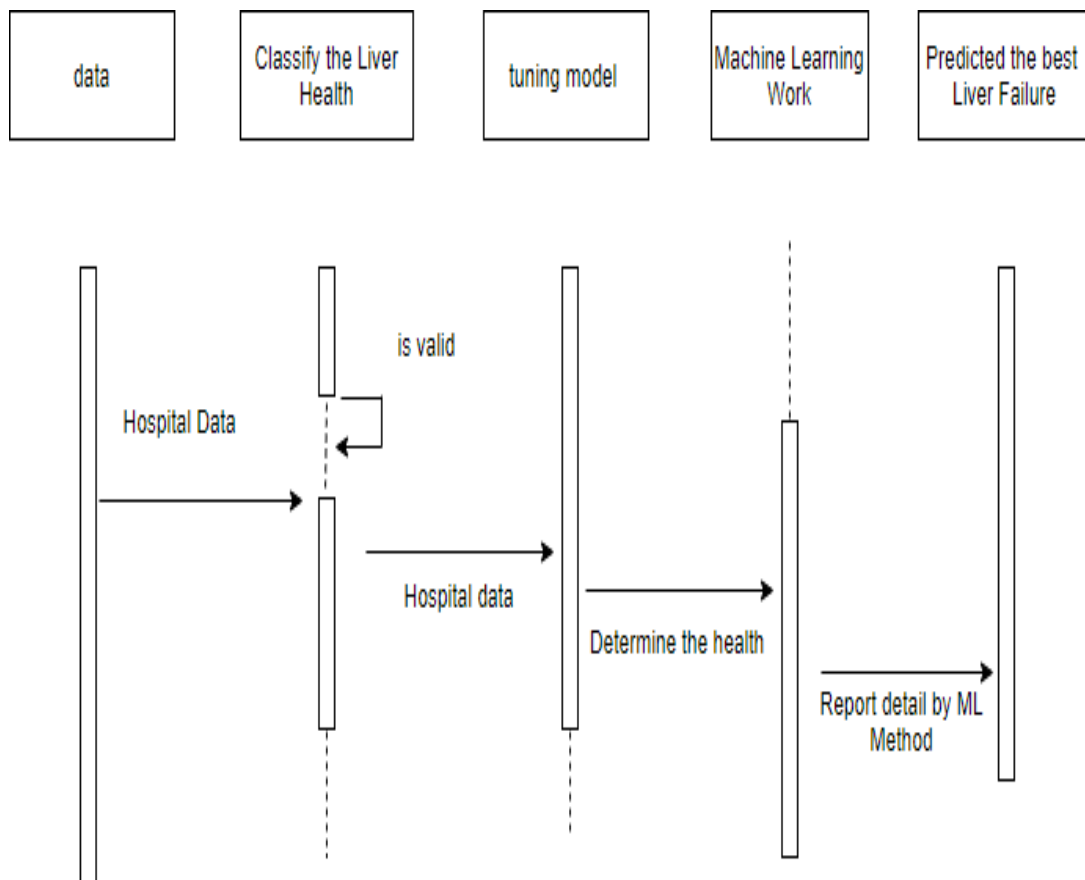


FIG: 4.14 SEQUENCE DIAGRAM

4.5.5 E – R DIAGRAM

An entity relationship diagram (ERD), also known as an entity relationship model, is

a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system.

An ERD is a data modeling technique that can help define business processes and be used as the foundation for a relational database. Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization.

After a relational database is rolled out, an ERD can still serve as a referral point, should any debugging or business process re-engineering be needed later.

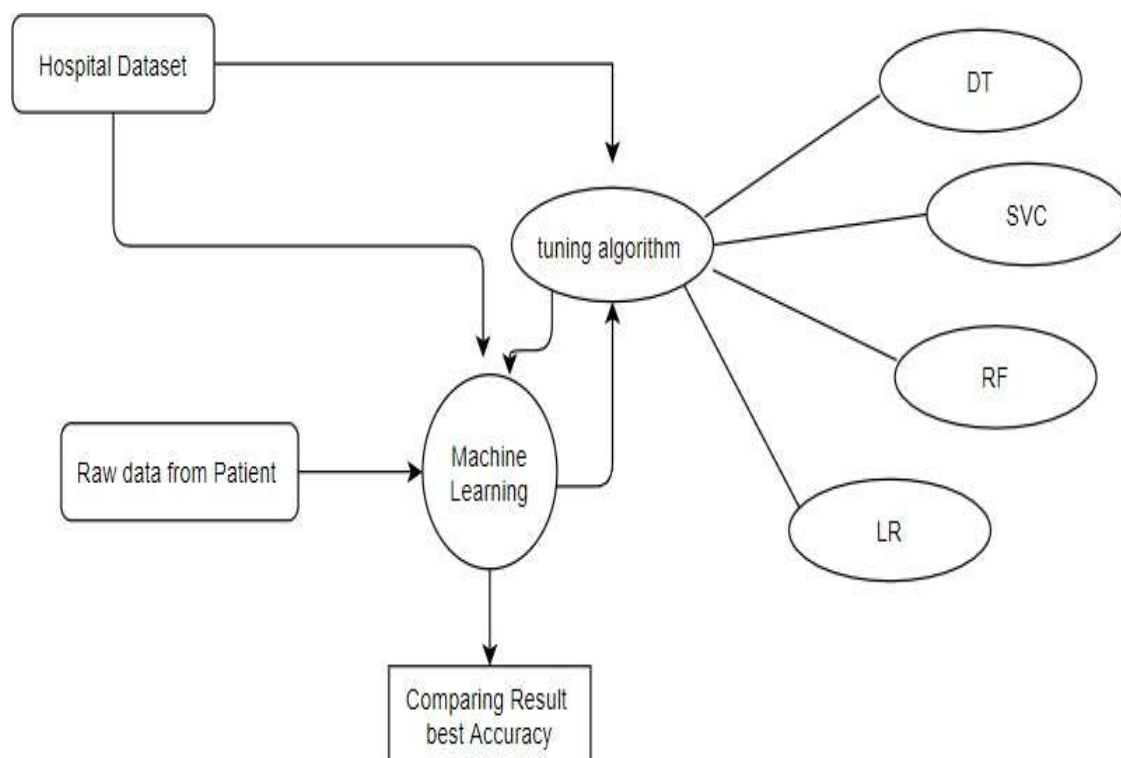


FIG: 4.15 E – R DIAGRAM

4.5.6 PREDICTION RESULT BY ACCURACY:

Logistic regression algorithm also uses a linear equation with independent predictors to predict a value. The predicted value can be anywhere between negative infinity to positive infinity. It needs the output of the algorithm to be classified variable data. Higher accuracy predicting result is logistic regression model by comparing the best accuracy.

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

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

False Positives (FP):

A person who will pay predicted as defaulter. When actual class is no and predicted class is yes. E.g., if actual class says this passenger did not survive but predicted class tells you that this passenger will survive.

False Negatives (FN):

A person who defaults predicted as payer. When actual class is yes but predicted class is no. E.g., if actual class value indicates that this passenger survived and predicted class tells you that passenger will die.

True Positives (TP):

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

True Negatives (TN):

A person who defaults predicted as payer. These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no. E.g. if actual class says this passenger did not survive and predicted class tells you the same thing.

ACCURACY:

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

ACCURACY CALCULATION:

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

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

PRECISION:

The proportion of positive predictions that are actually correct. (When the model predicts default: how often is correct?)

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

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

RECALL:

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

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

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

F1 Score:

It is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy,

especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.

General Formula:

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

F1-Score Formula:

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

COMPARING ALGORITHM WITH PREDICTION IN THE FORM OF BEST ACCURACY RESULT

It is important to compare the performance of multiple different machine learning algorithms consistently and it will discover to create a test harness to compare multiple different machine learning algorithms in Python with scikit-learn. It can use this test harness as a template on your own machine learning problems and add more and different algorithms to compare.

Each model will have different performance characteristics. Using resampling methods like cross validation, you can get an estimate for how accurate each model may be on unseen data. It needs to be able to use these estimates to choose one or two best models from the suite of models that you have created. When have a new dataset, it is a good idea to visualize the data using different techniques in order to look at the data from different perspectives.

The same idea applies to model selection. You should use a number of different ways of looking at the estimated accuracy of your machine learning algorithms in order to choose the one or two to finalize. A way to do this is to use different visualization methods to show the average accuracy, variance and other properties of the distribution of model accuracies.

In the next section you will discover exactly how you can do that in Python with scikit-learn. The key to a fair comparison of machine learning algorithms is

ensuring that each algorithm is evaluated in the same way on the same data and it can achieve this by forcing each algorithm to be evaluated on a consistent test harness.

In the example below 4 different algorithms are compared:

- Logistic Regression
- Decision Tree
- Random Forest
- Support Vector Machine

The K-fold cross validation procedure is used to evaluate each algorithm, importantly configured with the same random seed to ensure that the same splits to the training data are performed and that each algorithm is evaluated in precisely the same way.

Before that comparing algorithm, building a Machine Learning Model using install Scikit-Learn libraries. In this library package have to done preprocessing, linear model with logistic regression method, cross validating by K Fold method, ensemble with random forest method and tree with decision tree classifier. Additionally, splitting the train set and test set. To predicting the result by comparing accuracy.

4.5.7 TRAINING AND TESTING MODEL

TRAINING MODEL:

Now the next step in the testing is to train the model, in this step we training our model to improve its performance for better outcome of the problem. We use datasets to train the model using various machine learning algorithms. Training a model is required so that it can understand the various patterns, rules, and, features.

TESTING MODEL:

Presently the subsequent stage is to prepare the model, in this progression we train

our model to improve its exhibition for better result of the issue. We use datasets to prepare the model utilizing different AI calculations. Preparing a model is required so it can comprehend the different examples, rules, and, highlights.

4.6 ALGORITHM AND TECHNIQUES

ALGORITHM EXPLANATION

In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation.

This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc.

In Supervised Learning, algorithms learn from labeled data. After understanding the data, the algorithm determines which label should be given to new data based on pattern and associating the patterns to the unlabeled new data.

Used Python Packages:

sklearn:

- In python, sklearn is a machine learning package which include a lot of ML algorithms.
- Here, we are using some of its modules like `train_test_split`, `DecisionTreeClassifier` or `Logistic Regression` and `accuracy_score`.

NumPy:

- It is a numeric python module which provides fast math functions for calculations.
- It is used to read data in NumPy arrays and for manipulation purpose.

Pandas:

- Used to read and write different files.
- Data manipulation can be done easily with data frames.

Matplotlib:

- Data visualization is a useful way to help with identify the patterns from given dataset.
- Data manipulation can be done easily with data frames.

4.6.1 LOGISTIC REGRESSION

Logistic regression is one of the simpler classification models. Because of its parametric nature it can to some extent be interpreted by looking at the parameters making it useful when experimenters want to look at relationships between variables.

A parametric model can be described entirely by a vector of parameters = $(0, 1, \dots, p)$. An example of a parametric model would be a straight-line $y = kx + m$ where the parameters are k and m . With known parameters the entire model can be recreated.

Logistic regression is a parametric model where the parameters are coefficients to the predictor variables written as $0 + 1 + X_1 + \dots + P X_p$ Where 0 is called the intercept.

For convenience we instead write the above sum of the parameterized predictor variables in vector form as X . The name logistic regression is a bit unfortunate since a regression model is usually used to find a continuous response variable, whereas in classification the response variable is discrete.

The term can be motivated by the fact that we in logistic regression found the probability of the response variable belonging to a certain class, and this probability is continuous.

Logistic regression is quite possibly the most well - known Machine Learning calculations, which goes under the supervised learning method. It is utilized for anticipating the absolute ward variable utilizing a given arrangement of autonomous factors.

Strategic relapse predicts the yield of a downright reliant variable. Hence the result should be a straight out or discrete worth. It very well may be either Yes or No, 0 or 1, valid or False, and so on however as opposed to giving the specific worth as 0 and 1, it gives the probabilistic qualities which lie somewhere in the range of 0 and 1.

```
True Positive : 8
True Negative : 41
False Positive : 11
False Negative : 110
```

```
True Positive Rate : 0.06779661016949153
True Negative Rate : 0.7884615384615384
False Positive Rate : 0.21153846153846154
False Negative Rate : 0.9322033898305084
```

```
Positive Predictive Value : 0.42105263157894735
Negative predictive value : 0.271523178807947
Confusion matrix-LR:
```

```
[[110  11]
 [ 41   8]]
```

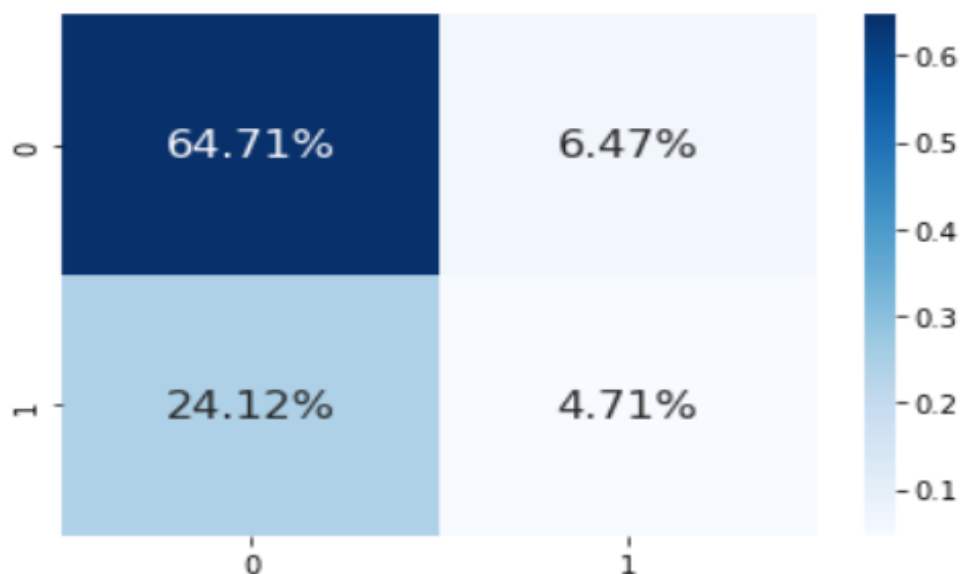


FIG: 4.16 CONFUSION MATRIX OF LR

```
#Predicting the test set result [11]
Predicted_y=classifier.predict(x_test)
```

In the above code, we have created a predicted_y vector to predict the test result. By executing the above code, a new vector (prediction_y) will be created under the variable explorer option. It can be seen as: [111 ... 111]

The FIG: 4.16, shows the confusion matrix of Logistic Regression algorithm. The True Positive, True Negative, False Positive, False Negative values for the given dataset have been predicted.

	precision	recall	f1-score	support
1	0.73	0.91	0.81	121
2	0.42	0.16	0.24	49
accuracy			0.69	170
macro avg	0.57	0.54	0.52	170
weighted avg	0.64	0.69	0.64	170

Accuracy result of LogisticRegression is: 69.41176470588235

Confusion Matrix result of Logistic Regression is:
[[110 11]
[41 8]]

Sensitivity : 0.9090909090909091

Specificity : 0.16326530612244897

Cross validation test results of accuracy:
[0.6754386 0.74336283 0.7079646 0.69026549 0.73451327]

Accuracy result of Logistic Regression is: 71.03089582362986

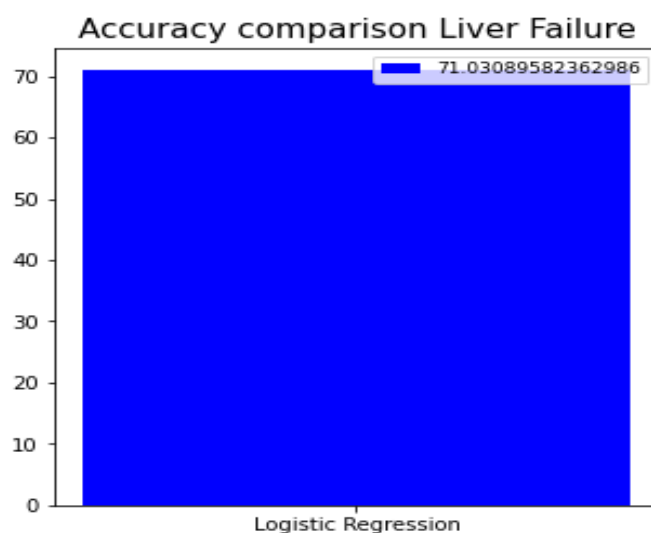


FIG: 4.17 CLASSIFICATION REPORT OF LR

4.6.2 DECISION TREE

Decision Tree calculation has a place with the supervised learning algorithms. In contrast to other supervised learning algorithms, a decision tree algorithm can be utilized for taking care of regression and classification issues as well.

The general thought process of utilizing Decision Tree is to make a training model that can use to predict class or estimation of objective factors by taking in choice standards derived from earlier data (training data).

Decision Tree is a supervised learning technique that can be utilized for both order and regression issues, however generally it is liked for taking care of Classification issues. It is a tree-organized classifier, where internal hubs address the highlights of a dataset, branches address the choice rules and each leaf hub addresses the result.

In a Decision tree, there are two hubs, which are the Decision Node and Leaf Node. Decision hubs are utilized to used on any choice and have different branches, while leaf hubs are the yield of those choices and don't contain any further branches.

λ A choice tree just poses an inquiry, and dependent on the appropriate response (Yes/No), it further split the tree into subtrees.

In general, Decision tree analysis is a predictive modelling tool that can be applied across many areas. Decision trees can be constructed by an algorithmic approach that can split the dataset in different ways based on different conditions. Decisions trees are the most powerful algorithms that falls under the category of supervised algorithms.

They can be used for both classification and regression tasks. The two main entities of a tree are decision nodes, where the data is split and leaves, where we got outcome.

We have the following two types of decision trees.

- **Classification decision trees** – In this kind of decision trees, the decision variable is categorical. The above decision tree is an example of classification decision tree.
 - **Regression decision trees** – In this kind of decision trees, the decision variable is continuous.
1. **Root Node:** It represents the entire population or sample and this further gets divided into two or more homogeneous sets.
 2. **Splitting:** It is a process of dividing a node into two or more sub-nodes.
 3. **Decision Node:** When a sub-node splits into further sub-nodes, then it is called the decision node.
 4. **Leaf / Terminal Node:** Nodes do not split is called Leaf or Terminal node.
 5. **Pruning:** When we remove sub-nodes of a decision node, this process is called pruning. You can say the opposite process of splitting.
 6. **Branch / Sub-Tree:** A subsection of the entire tree is called branch or sub-tree.
 7. **Parent and Child Node:** A node, which is divided into sub-nodes is called a parent node of sub-nodes whereas sub-nodes are the child of a parent node.

Decision trees classify the examples by sorting them down the tree from the root to some leaf/terminal node, with the leaf/terminal node providing the classification of the example. Each node in the tree acts as a test case for some attribute, and each edge descending from the node corresponds to the possible answers to the test case. This process is recursive in nature and is repeated for every subtree

rooted at the new node.

```
True Positive : 22
True Negative : 27
False Positive : 39
False Negative : 82

True Positive Rate : 0.21153846153846154
True Negative Rate : 0.4090909090909091
False Positive Rate : 0.5909090909090909
False Negative Rate : 0.7884615384615384

Positive Predictive Value : 0.36065573770491804
Negative predictive value : 0.24770642201834864
Confusion matrix-DT:
[[82 39]
 [27 22]]
```

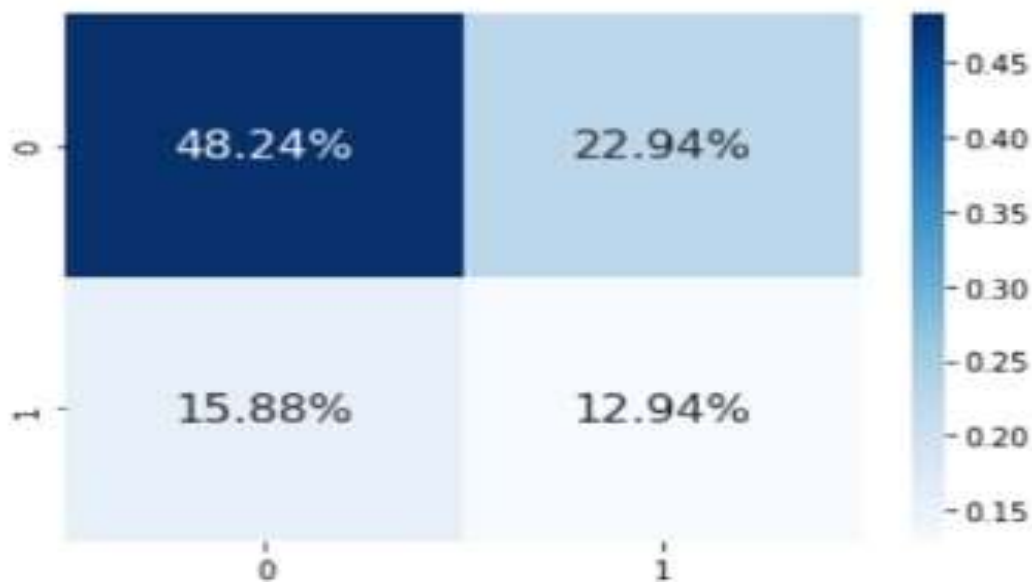


FIG: 4.18 CONFUSION MATRIX OF DT

The FIG: 4.17, shows the confusion matrix of Decision Tree algorithm. The True Positive, True Negative, False Positive, False Negative values for the given dataset have been predicted. The False positive values percentage seem higher which can be a negative influence for correct prediction of liver disease.

	precision	recall	f1-score	support
1	0.75	0.68	0.71	121
2	0.36	0.45	0.40	49
accuracy			0.61	170
macro avg	0.56	0.56	0.56	170
weighted avg	0.64	0.61	0.62	170

Accuracy result of DecisionTree is: 61.1764705882353

Confusion Matrix result of DecissionTree Classifier is:

```
[[82 39]
 [27 22]]
```

Sensitivity : 0.6776859504132231

Specificity : 0.4489795918367347

Cross validation test results of accuracy:

```
[0.65789474 0.71681416 0.59292035 0.56637168 0.7699115 ]
```

Accuracy result of DecisionTree Classifier is: 66.07824871914299

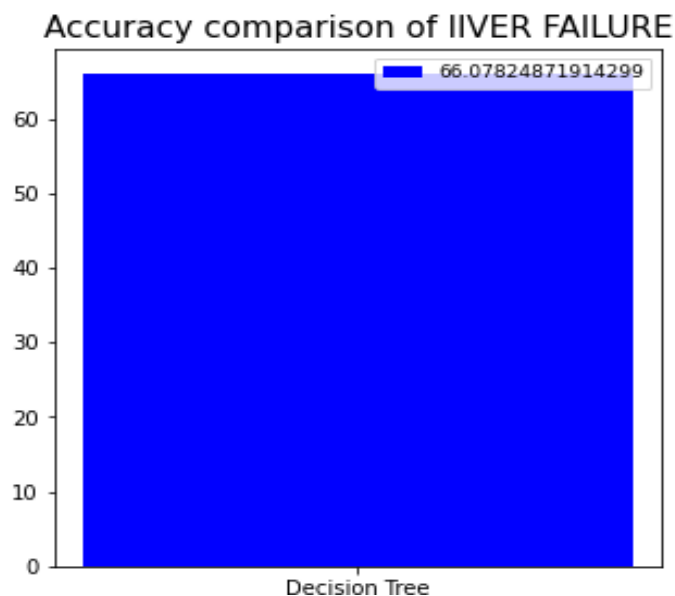


FIG: 4.19 CLASSIFICATION REPORT OF DT

4.6.3 RANDOM FOREST

Random forests or random decision forests are an ensemble learning technique for classification, regression and different assignments that works by developing a huge

number of decision trees at training time and yielding the class that is the method of the classes (classification) or mean forecast (regression) of the individual trees.

Random decision forests right for decision trees propensity for overfitting to their training set. In the forest of trees has been the immediate connection between the combine trees and outcome it can get.

To get increasingly effective and precise predictions, random forest inserts an additional layer of irregularity to stowing.

Random Forest is a mainstream AI calculation that has a place with the regulated learning strategy. It tends to be utilized for both classification and Regression issues in ML.

It depends on the idea of ensemble learning, which is a cycle of combining different classifiers to tackle a mind-boggling issue and to improve the exhibition of the model.

λ As the name suggests, “Random Forest is a classifier that contains various choice trees on different subsets of the given dataset and takes the normal to improve the prescient exactness of that dataset”.

Instead of depending on one choice tree, the irregular woods takes the exception from each tree and dependent on the dominant part votes of forecasts, and it predicts the last yield.

λ the more prominent number of trees in the woods prompts higher precision and forestalls the issue of overfitting.

Predicting the test set result [11]

```
Y_pred_r=clf_r.predict(X_test)ge : 27
```

In the above code, we have created a `y_pred_r` vector to predict the test set result. By executing the above code, a new vector (`y_pred_r`) will be created under the variable explorer option. It can be seen as: `[111 ... 111]`

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output.

Therefore, below are two assumptions for a better Random Forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

True Positive : 12
True Negative : 37
False Positive : 20
False Negative : 101

True Positive Rate : 0.10619469026548672
True Negative Rate : 0.6491228070175439
False Positive Rate : 0.3508771929824561
False Negative Rate : 0.8938053097345132

Positive Predictive Value : 0.375
Negative predictive value : 0.26811594202898553
Confusion matrix-RF:
[[101 20]
 [37 12]]

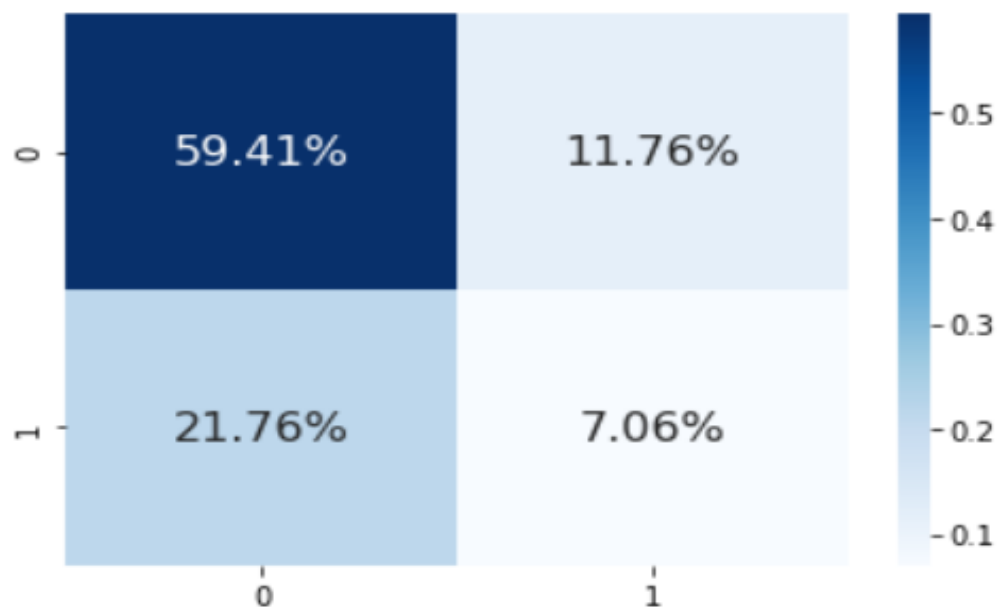


FIG: 4.20 CONFUSION MATRIX OF RF

The FIG: 4.18, shows the confusion matrix of Random Forest algorithm. The True Positive, True Negative, False Positive, False Negative values for the given dataset have been predicted. The False Negative values percentage seem higher which can

be of great help for prediction but when taken other factors into consideration the number of True Positive values predicted are low than True Negative.

	precision	recall	f1-score	support
1	0.73	0.83	0.78	121
2	0.38	0.24	0.30	49
accuracy			0.66	170
macro avg	0.55	0.54	0.54	170
weighted avg	0.63	0.66	0.64	170

Accuracy result of Random Forest is: 66.47058823529412

Confusion Matrix result of Random Forest is:

```
[[101  20]
 [ 37  12]]
```

Sensitivity : 0.8347107438016529

Specificity : 0.24489795918367346

Cross validation test results of accuracy:

```
[0.69298246 0.75221239 0.69911504 0.61061947 0.75221239]
```

Accuracy result of Random Forest is: 70.14283496351497

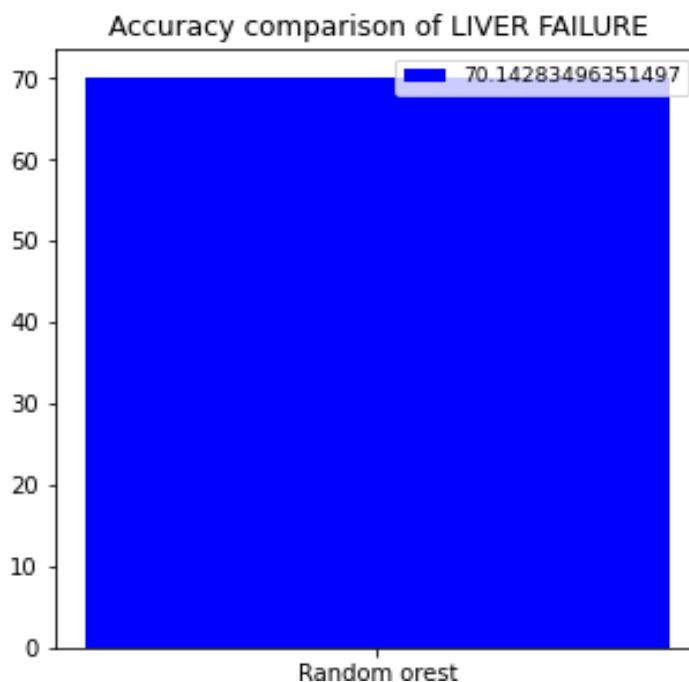


FIG: 4.21 CLASSIFICATION REPORT OF RANDOM FOREST

4.6.4 SUPPORT VECTOR MACHINE

SVM is a supervised learning calculation. It can utilize for both grouping or relapse issues however generally it is utilized in characterization issues and can comprehend both liner and non – linear issues.

SVM grouping strategy which is an endeavor to pass a linearly separable hyperplane to order the dataset into two classes [11 – 12]. At long last, the model can without a doubt gauge the objective groups for new cases.

SVM aims to find an optimal hyperplane that separates the data into different classes. The scikit-learn package in python is used for implementing SVM. The pre-processed data is split into test data and training set which is of 25% and 75% of the total dataset respectively.

A support vector machine constructs a hyper plane or set of hyper planes in a high- or infinite-dimensional space. A good separation is achieved by the hyper plane that has the largest distance to the nearest training data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

Classification algorithms are widely used in various medical applications. Classification aims to build an effective model for predicting class labels of unknown data. The model is built on the training data, which consists of data points chosen from input data space and their class labels.

A Support Vector Machine (SVM) separates the data into two categories of performing classification and constructing an N-dimensional hyper plane. These models are closely related to classical multilayer perceptron neural networks.

A support vector machine constructs a hyper plane or set of hyper planes in a high- or infinite-dimensional space. A good separation is achieved by the hyper plane that has the largest distance to the nearest training data point of any class (so-called

functional margin), since in general the larger the margin the lower the generalization error of the classifier.

There is an alternative training method for polynomial, radial basis function and multi-layer perceptron classifiers in which the weights of the network are found by solving a quadratic programming problem with linear constraints, rather than by solving a non-convex, the unconstrained minimization problem as in standard neural network training.

There are many possible kernel functions and the most common kernel are: Linear, polynomial, sigmoid and radial basis function (RBF).

In this paper we use linear kernel function shows in equation .1:

$$K(x_i, x_j) = x_i^T x_j \quad (1)$$

Depending on the kernel type we choose the kernel parameters have to be set. Which kernel type performs best, depends on the application and can be determined by using cross-validation.

In the SVM literature, a predictor variable which is called an attribute and a transformed attribute that is used to define the hyper plane is called a feature. Here, choosing the most suitable representation can be taken as feature selection.

A set of features that describes one case is called a vector. The goal of this modelling is to find the optimal hyperplane which separates clusters of vectors in such a way those cases with one category of the target Variable are on one side of the plane and cases with the other category are on the other side of the plane.

1. Solve the data points are not linearly separable
2. Effective in a higher dimension.
3. Suitable for small data set: effective when the number of features is more than training examples.
4. Overfitting problem: The hyperplane is affected by only the support vectors, so SVMs are not robust to the outlier.

```

True Positive : 0
True Negative : 49
False Positive : 0
False Negative : 121

True Positive Rate : 0.0
True Negative Rate : 1.0
False Positive Rate : 0.0
False Negative Rate : 1.0

Positive Predictive Value : nan
Negative predictive value : 0.28823529411764703
Confusion matrix-SVM:
[[121   0]
 [ 49   0]]

```

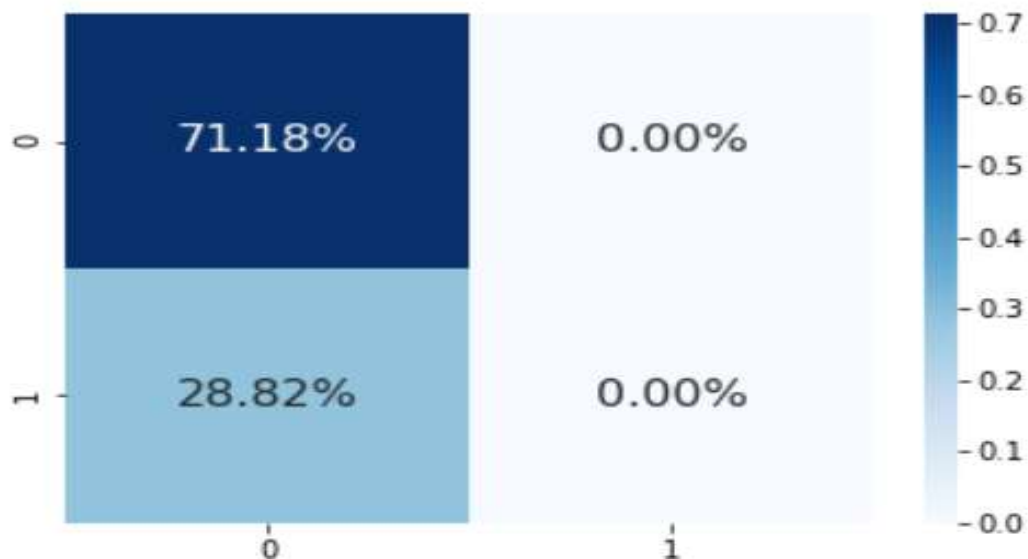


FIG: 4.22 CONFUSION MATRIX OF SVM

The FIG: 4.19, shows the confusion matrix of Support Vector Machine algorithm. The True Positive, True Negative, False Positive, False Negative values for the given dataset have been predicted.

The False Negative values percentage seem higher which can be of great help for prediction but when taken other factors into consideration the number of True Positive values predicted are 0 with True Negative value being 49.

As two True positive and False positives are 0% which is difficult to be taken into consideration.

	precision	recall	f1-score	support
1	0.71	1.00	0.83	121
2	0.00	0.00	0.00	49
accuracy			0.71	170
macro avg	0.36	0.50	0.42	170
weighted avg	0.51	0.71	0.59	170

Accuracy result of Support Vector Machines is: 71.17647058823529

Confusion Matrix result of Support Vector Machines is:

```
[[121  0]
 [ 49  0]]
```

Sensitivity : 1.0

Specificity : 0.0

Cross validation test results of accuracy:

```
[0.71052632 0.71681416 0.71681416 0.71681416 0.7079646 ]
```

Accuracy result of Support Vector Machine is: 71.37866790870982

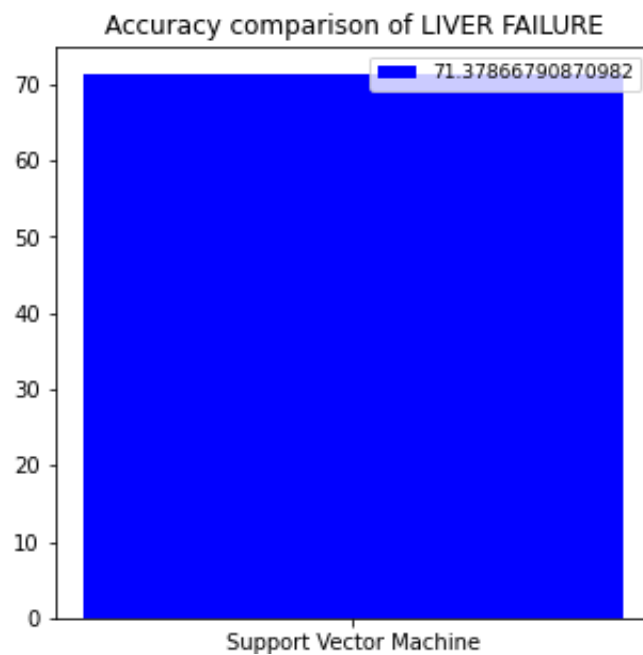


FIG: 4.23 CLASSIFICATION REPORT OF SVM

4.6.5 DEPLOYMENT

FLASK (WEB FRAMEWORK):

Flask is a micro web framework written in Python. It is classified as a micro-framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

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

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

FEATURES:

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

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

- Built-In Development server and Fast debugger
- integrated support for unit testing
- RESTful request dispatching

- Uses Jinja2 Templating
- support for secure cookies (Client-Side Sessions)
- Unicode based
- Extensive Documentation
- Google App Engine Compatibility
- Extensions available to enhance features desired

Plus, Flask gives you so much more CONTROL on the development stage of your project. It follows the principles of minimalism and let you decide how you will build your application.

- Flask has a lightweight and modular design, so it easy to transform it to the web framework you need with a few extensions without weighing it down
- ORM-agnostic: you can plug in your favorite ORM e.g., SQLAlchemy.
- Basic foundation API is nicely shaped and coherent.
- Flask documentation is comprehensive, full of examples and well structured. You can even try out some sample application to really get a feel of Flask.
- It is super easy to deploy Flask in production (Flask is 100% WSGI 1.0 compliant")
- HTTP request handling functionality
- High Flexibility

The configuration is even more flexible than that of Django, giving you plenty of solution for every production need.

To sum up, Flask is one of the most polished and feature-rich micro frameworks available. Still young, Flask has a thriving community, first-class extensions, and an elegant API. Flask comes with all the benefits of fast templates, strong WSGI features, thorough unit testability at the web application and library level, extensive documentation. So next time you are starting a new project where you need some good features and a vast number of extensions, definitely check out Flask

Flask is an API of Python that allows us to build up web-applications. It was

developed by Armin Ronacher. Flask's framework is more explicit than Django framework and is also easier to learn because it has less base code to implement a simple web-Application

Flask is a micro web framework written in Python. It is classified as a micro-framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

Overview of Python Flask Framework Web apps are developed to generate content based on retrieved data that changes based on a user's interaction with the site. The server is responsible for querying, retrieving, and updating data. This makes web applications to be slower and more complicated to deploy than static websites for simple applications.

Flask is an excellent web development framework for REST API creation. It is built on top of Python which makes it powerful to use all the python features. Flask is used for the backend, but it makes use of a templating language called Jinja2 which is used to create HTML, XML or other markup formats that are returned to the user via an HTTP request.

Django is considered to be more popular because it provides many out of box features and reduces time to build complex applications. Flask is a good start if you are getting into web development. Flask is a simple, un-opinionated framework; it doesn't decide what your application should look like developers do.

Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website.

Advantages of Flask:

- Higher compatibility with latest technologies.
- Technical experimentation.

- Easier to use for simple cases.
- Codebase size is relatively smaller.
- High scalability for simple applications.
- Easy to build a quick prototype.
- Routing URL is easy.
- Easy to develop and maintain applications.

Framework Flask is a web framework from Python language. Flask provides a library and a collection of codes that can be used to build websites, without the need to do everything from scratch. But Framework flask still doesn't use the Model View Controller (MVC) method.

Flask-RESTful is an extension for Flask that provides additional support for building REST APIs.

You will never be disappointed with the time it takes to develop an API. Flask-Restful is a lightweight abstraction that works with the existing ORM/libraries. Flask-RESTful encourages best practices with minimal setup.

Flask Restful is an extension for Flask that adds support for building REST APIs in Python using Flask as the back-end. It encourages best practices and is very easy to set up. Flask restful is very easy to pick up if you're already familiar with flask.

Flask is a web framework for Python, meaning that it provides functionality for building web applications, including managing HTTP requests and rendering templates and also, we can add to this application to create our API.

CHAPTER 5

RESULT AND DISCUSSION

Our main goal going into this project was to predict liver disease using various machine learning techniques and select one best model among them.

We predicted using Support Vector Machine (SVM), Logistic Regression, Decision Tree and Random Forest. All of them predicted with better results. With Each algorithm, we have observed Accuracy, Precision, Sensitivity and Specificity which can be defined as follows:

ACCURACY:

The accuracy of a classifier is the percentage of the test set tuples that are correctly classified by the classifier.

$$\text{Accuracy} = \frac{\text{no. of TP} + \text{no. of TN}}{\text{no. of TP} + \text{FP} + \text{FN} + \text{TN}}$$

SENSITIVITY:

Sensitivity is also referred as True positive rate i.e. the proportion of positive tuples that are correctly identified.

$$\text{Sensitivity} = \frac{\text{no. of TP}}{\text{no. of TP} + \text{no. of FN}}$$

PRECISION:

precision is defined as the proportion of the true positives against all the positive results (both true positives and false positives)

$$Precision = \frac{no. of TP}{no. of TP + FP}$$

SPECIFICITY:

Specificity is the True negative rate that is the proportion of negative tuples that are correctly identified the results of each of the classification algorithm is summarized in the table shown below.

$$Specificity = \frac{no. of TN}{no. of TN + FP}$$

The Results of each of the classification algorithm is summarized in the table shown below.

Classification Algorithms	Accuracy	Precision	Sensitivity	Specificity
Logistic Regression	71.03	0.73	0.90	0.16
Decision Tree	66.07	0.75	0.67	0.44
Random Forest	70.1	0.73	0.83	0.24
Support Vector Machine	71.3	0.71	1.0	0.0

TABLE 1.1 RESULTS OF CLASSIFICATION ALGORITHM

As clearly summarized in the table, Logistic Regression and Support vector machine gave the best results in case of just accuracy when was applied with the testing dataset

When the other factors were taken into consideration neither Logistic Regression nor Support Vector Machine is chosen.

Random Forest Algorithm shows best result when all factors are taken into consideration

CHAPTER 6

CONCLUSION AND FUTURE WORK

CONCLUSION:

The logical cycle starts with the cleaning and handling of data, the absence of significant worth, the examination of exploration, and afterward the development and assessment.

The genuine benefit of the overall test is the capacity to get high scores. This program can assist with diagnosing liver disappointment relying upon the patient's wellbeing status. Father About the existence of a patient's liver that matches constant man-made reasoning.

Adjust this technique to show the anticipated outcomes on the necessary site or work area application. Working on the execution of man-made brainpower.

FUTURE WORK:

- Patient Liver Health data connect with real time AI model.
- To automate this process by show the prediction result in web application or desktop application.
- To optimize the work to implement in Artificial Intelligence environment.

REFERENCES:

- [1] K. Sumeet, J.J. Larson, B. Yawn, T.M. Therneau, W.R. Kim, underestimation of liver-related mortality in the United States. *Gastroenterology*; (2013) 145:375-382, e371-372.
- [2] A.A. Mokdad, A.D. Lopez, S. Shahrzad, R. Lozano, A.H. Mokdad, J. Stanaway, et al, Liver cirrhosis mortality in 187 countries between 1980 and 2010: a systematic analysis. *BMC Med* 2014; 12:145.
- [3] Byass, Peter, The global burden of liver disease: a challenge for methods and for public health. *BMC medicine* 12.1 (2014); 159.
- [4] L. A. Auxilia, Accuracy Prediction Using Machine Learning Techniques for Indian Patient Liver Disease. 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI). IEEE (2018).
- [5] Hashem, M. Esraa, S. Mai, A study of support vector machine algorithm for liver disease diagnosis. *American Journal of Intelligent Systems* 4.1 (2014); 9-14.
- [6] P. Sajda, "Machine learning for detection and diagnosis of disease." *Annu. Rev. Biomed. Eng.* 8 (2006); 537-565.
- [7] UCI Machine Learning Repository. ILPD (Indian Liver Patient Dataset) Data Set.
[https://archive.ics.uci.edu/ml/datasets/ILPD+\(Indian+Liver+Patient+Dataset\)](https://archive.ics.uci.edu/ml/datasets/ILPD+(Indian+Liver+Patient+Dataset))
- [8] Logistic Regression, Retrieve from:
HTTPS://WWW.SAEDSAYAD.COM/LOGISTIC_REGRESSION.HTM, LAST Accessed: 5 October, 2019
- [9] L. Breiman, Random Forests. *Machine Learning*, 45(1), (2001); 5–32.
<https://doi.org/10.1023/A:1010933404324>
- [10] Decision Trees, Retrieve from: <https://dataaspirant.com/2017/01/30/how-decision-tree-algorithm-works/>, Last Accessed: 5 October, 2019
- [11] Support vector machine, Retrieve from:
<http://www.statsoft.com/textbook/support-vector-machines>, Last Accessed: 5 October, 2019

- [12] V. Vapnik, I. Guyon, T. H.-M, Learn, and undefined 1995. Support vector machines. statweb.stanford.edu (1995).
- [13] Zhang M, Zhou Z, "ML-KNN: A lazy learning approach to multi-label learning." Pattern recognition 40.7:(2007); 2038-2048.
- [14] G. Guo, H. Wang, D. Bell, Y. Bi, K. Greer, KNN Model-Based Approach in Classification (pp. 986–996). Springer, Berlin, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39964-3_62
- [15] Naive Bayes, , Retrieve from: <https://www.geeksforgeeks.org/naive-bayes-classifiers/>, Last Accessed: 5 October, 2019
- [16] S. M. Mahmud, et al. "Machine Learning Based Unified Framework for Diabetes Prediction." Proceedings of the 2018 International Conference on Big Data Engineering and Technology. ACM (2018).
- [17] S. Safavian, D. Landgrebe, A survey of decision tree classifier methodology. IEEE transactions on systems, man, and cybernetics, 21(3), (1991); 660-674.
- [18] A. Chervonenkis, Early history of support vector machines. In Empirical Inference (pp. 13-20). Springer, Berlin, Heidelberg (2013).
- [19] K.M. Leung, Naive bayesian classifier. Polytechnic University Department of Computer Science/Finance and Risk Engineering (2007).
- [20] A. K. M. Sazzadur Rahman, F. M. Javed Mehedi Shamrat, Zarrin Tasnim, Joy Roy, Syed Akhter Hossain, A Comparative study on liver disease prediction using supervised machine learning algorithms. INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH VOLUME 8, ISSUE 11, NOVEMBER 2019.

CHAPTER 7

APPENDICES AND SCREENSHOTS

APPENDICES

SAMPLE CODE:

Module 1: Data validation and pre-processing technique

```
#import library packages
import pandas as pd
import numpy as np
# warning
import warnings
warnings.filterwarnings('ignore')
#load given dataset
data = pd.read_csv("liver_patient.csv")
data.head()
#shape
data.shape
#size
data.size
#Checking datatype and information about dataset
data.info()
#To describe the dataframe
data.describe()
#duplicate data
data.duplicated()
# sum of duplicated
data.duplicated().sum()
data.shape
# null values
```

```

data.isnull()
#sum of null values
data.isnull().sum()
# shape before drop null values
data.shape
#drop null values
df=data.dropna()
#drop after drop null values
df.shape
df.isnull().sum()
#changing feature name
df = df.rename({'Dataset': 'Result'}, axis=1)
df.head(10)
df.columns
df.Result.unique()
df.Age.unique()
df['Total_Bilirubin'].unique()
df['Direct_Bilirubin'].unique()
df['Alkaline_Phosphotase'].unique()
df['Alamine_Aminotransferase'].unique()
df['Alamine_Aminotransferase'].unique()
df['Total_Protiens'].unique()
df['Albumin'].unique()
df['Albumin'].unique()
df['Albumin_and_Globulin_Ratio'].unique()
df['Result'].unique()
df.corr()
df.columns
pd.Categorical(df['Age']).describe()
pd.Categorical(df['Gender']).describe()
pd.Categorical(df['Result']).describe()
df['Alkaline_Phosphotase'].min()
df['Alkaline_Phosphotase'].max()
df['Alkaline_Phosphotase'].mean()

```

```

df["Age"].sort_values().unique()
df.columns
from sklearn.preprocessing import LabelEncoder
var_mod = {'Gender'}
le=LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i]).astype(int)
df.head()

```

Module 2: Exploration data analysis of visualization and training a model by given attributes

```

#import library packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
#warnings
import warnings
warnings.filterwarnings('ignore')
#load dataset
data = pd.read_csv("liver_patient.csv")
data.head()
data.shape
data.size
data.duplicated()
data.duplicated().sum()
df=data.dropna()
df.shape
df.isnull().sum()
df.columns
#changing feature name

```

```

df = df.rename({'Dataset': 'Result'}, axis=1)
plt.figure(figsize=(12,7))
sns.countplot(x='Result',data=df)
df1 = df[df["Result"]==1]
df2 = df[df["Result"]==2]
df2

plt.figure(figsize=(8,6))

plt.xlabel("")
plt.hist(df["Gender"],color="red")
plt.title("All category")
plt.show()
df.Result.unique()
plt.figure(figsize=(10,4))
plt.subplot(1, 2, 1)
plt.hist(df1["Gender"],color="green")
plt.title("Affected people")

plt.subplot(1, 2, 2)
plt.hist(df2["Gender"],color="orange")
plt.title("Not Affected people")

plt.show()
plt.scatter(df["Albumin"],df["Total_Protiens"])
df.head()
plt.figure(figsize=(12,6))
plt.plot(df["Albumin_and_Globulin_Ratio"])
plt.show()
# Heatmap plot diagram
fig, ax = plt.subplots(figsize=(18,13))
sns.heatmap(df.corr(), ax=ax, annot=True)
boxplot = df.boxplot(column=["Total_Protiens"], by="Result",figsize=(8,6),
fontsize=15)

```

```

boxplot = df.boxplot(column=['Albumin_and_Globulin_Ratio'],
by="Result",figsize=(8,6), fontsize=15)
plt.show()
plt.figure(figsize=(10,5))
plt.subplot(1,2,1)
plt.scatter(df["Albumin_and_Globulin_Ratio"],df["Result"],color="red")
plt.subplot(1,2,2)
plt.scatter(df["Albumin"],df["Result"],color="red")
plt.show()

a=pd.plotting.scatter_matrix(df, alpha=0.05, figsize=(18,10), diagonal='hist')
axis = 'off'

plt.show()
from sklearn.preprocessing import LabelEncoder
var_mod = ['Result']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i]).astype(int)
df.columns

```

Module 3: Performance measurements of Logistic regression algorithms

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# warning
import warnings
warnings.filterwarnings("ignore")
# load given data
data = pd.read_csv("liver_patient.csv")

```

```

data.head()
data.duplicated().sum()
data.shape
data = data.drop_duplicates()
data.shape
data.isnull().sum()
df=data.dropna()
df.shape
df.isnull().sum()
#changing feature name
df = df.rename({'Dataset': 'Result'}, axis=1)
df.columns
from sklearn.preprocessing import LabelEncoder
var_mod = ['Gender']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i]).astype(int)
df.Result.value_counts()
df.corr()
X = df.drop(labels='Result', axis=1)
#Response variable
y = df.loc[:, 'Result']

#According to the cross-validated MCC scores, the random forest is the best-
performing model, so now let's evaluate its performance on the test set.
from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score
#We'll use a test size of 30%. We also stratify the split on the response variable,
which is very important to do because there are so few fraudulent transactions.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=1, stratify=y)
from sklearn.linear_model import LogisticRegression

```

```
from sklearn.model_selection import cross_val_score
```

```
logR= LogisticRegression()
```

```
logR.fit(X_train,y_train)
```

```
predictLR = logR.predict(X_test)
```

```
print("")
```

```
print('Classification report of Logistic Regression Results:')
```

```
print("")
```

```
print(classification_report(y_test,predictLR))
```

```
print("")
```

```
x = (accuracy_score(y_test,predictLR)*100)
```

```
print('Accuracy result of LogisticRegression is:', x)
```

```
print("")
```

```
cm2=confusion_matrix(y_test,predictLR)
```

```
print('Confusion Matrix result of Logistic Regression is:\n',cm2)
```

```
print("")
```

```
sensitivity2 = cm2[0,0]/(cm2[0,0]+cm2[0,1])
```

```
print('Sensitivity : ', sensitivity2 )
```

```
print("")
```

```
specificity2 = cm2[1,1]/(cm2[1,0]+cm2[1,1])
```

```
print('Specificity : ', specificity2)
```

```
print("")
```

```
accuracy = cross_val_score(logR,X, y, scoring='accuracy')
```

```
print('Cross validation test results of accuracy:')
```

```
print(accuracy)
```

```
#get the mean of each fold
```

```

print("")
print("Accuracy result of Logistic Regression is:",accuracy.mean() * 100)
LR=accuracy.mean() * 100

```

```

def graph():

```

```

    import matplotlib.pyplot as plt
    data=[LR]
    alg="Logistic Regression"
    plt.figure(figsize=(5,5))
    b=plt.bar(alg,data,color=("b"))
    plt.title("Accuracy comparison Liver Failure",fontsize=15)
    plt.legend(b,data,fontsize=9)

```

```

graph()

```

```

TN = cm2[1][0]
FN = cm2[0][0]
TP = cm2[1][1]
FP = cm2[0][1]
print("True Positive :",TP)
print("True Negative :",TN)
print("False Positive :",FP)
print("False Negative :",FN)
print("")
TPR = TP/(TP+FN)
TNR = TN/(TN+FP)
FPR = FP/(FP+TN)
FNR = FN/(TP+FN)
print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)
print("")
PPV = TP/(TP+FP)
NPV = TN/(TN+FN)

```

```

print("Positive Predictive Value :",PPV)
print("Negative predictive value :",NPV)

cm2=confusion_matrix(y_test, predictLR)
print('Confusion matrix-LR:')
print(cm2)

sns.heatmap(cm2/np.sum(cm2), annot=True, cmap = 'Blues', annot_kws={"size":
16},fmt='.2%')
plt.show()

```

Module 4 : Performance measurements of Decision Tree:

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# warning
import warnings
warnings.filterwarnings("ignore")
# load given data
data = pd.read_csv("liver_patient.csv")
data.head()
data.duplicated().sum()
data.shape
data = data.drop_duplicates()
data.shape
data.isnull().sum()
df=data.dropna()
df.shape
df.isnull().sum()

```

```

#changing feature name
df = df.rename({'Dataset': 'Result'}, axis=1)
df.columns

from sklearn.preprocessing import LabelEncoder
var_mod = ['Gender']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i]).astype(int)
df.Result.value_counts()
df.corr()
X = df.drop(labels='Result', axis=1)
#Response variable
y = df.loc[:, 'Result']

# now let's evaluate its performance on the test set.
from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score
#We'll use a test size of 30%. We also stratify the split on the response variable,
which is very important to do because there are so few fraudulent transactions.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=1, stratify=y)

DecisionTree:
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score

DT=DecisionTreeClassifier()

DT.fit(X_train,y_train)

predictDT = DT.predict(X_test)

```

```

print("")
print('Classification report DecisionTree classifier Results:')
print("")

print(classification_report(y_test,predictDT))
print("")

x = (accuracy_score(y_test,predictDT)*100)
print('Accuracy result of DecisionTree is:', x)
print("")

cm2=confusion_matrix(y_test,predictDT)
print('Confusion Matrix result of DecissionTree Classifier is:\n',cm2)
print("")
sensitivity2 = cm2[0,0]/(cm2[0,0]+cm2[0,1])
print('Sensitivity : ', sensitivity2 )
print("")
specificity2 = cm2[1,1]/(cm2[1,0]+cm2[1,1])
print('Specificity : ', specificity2)
print("")

accuracy = cross_val_score(DT,X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of DecisionTree Classifier is:",accuracy.mean() * 100)
dt=accuracy.mean() * 100

```

```

def graph():
    import matplotlib.pyplot as plt
    data=[dt]
    alg="Decision Tree"
    plt.figure(figsize=(5,5))
    b=plt.bar(alg,data,color=("b"))
    plt.title("Accuracy comparison of IIVER FAILURE",fontsize=15)
    plt.legend(b,data,fontsize=9)

```

```

graph()
TN = cm2[1][0]
FN = cm2[0][0]
TP = cm2[1][1]
FP = cm2[0][1]
print("True Positive :",TP)
print("True Negative :",TN)
print("False Positive :",FP)
print("False Negative :",FN)
print("")
TPR = TP/(TP+FN)
TNR = TN/(TN+FP)
FPR = FP/(FP+TN)
FNR = FN/(TP+FN)
print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)
print("")
PPV = TP/(TP+FP)
NPV = TN/(TN+FN)
print("Positive Predictive Value :",PPV)
print("Negative predictive value :",NPV)

```

```

cm2=confusion_matrix(y_test, predictDT)
print('Confusion matrix-DT:')
print(cm2)

```

```

sns.heatmap(cm2/np.sum(cm2), annot=True, cmap = 'Blues', annot_kws={"size":
16},fmt='.2%')
plt.show()

```

Module 5: Performance measurements of Random Forest algorithms

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# warning
import warnings
warnings.filterwarnings("ignore")
# load given data
data = pd.read_csv("liver_patient.csv")
data.head()
data["Dataset"].unique()
data.duplicated().sum()
data.shape
data = data.drop_duplicates()
data.shape
data.isnull().sum()
df=data.dropna()
df.shape
df.isnull().sum()
#changing feature name
df = df.rename({'Dataset': 'Result'}, axis=1)

```

```

df.columns
from sklearn.preprocessing import LabelEncoder
var_mod = ['Gender']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i]).astype(int)
df.Result.value_counts()
df.corr()
X = df.drop(labels='Result', axis=1)
#Response variable
y = df.loc[:, 'Result']

# now let's evaluate its performance on the test set.
from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score
#We'll use a test size of 30%. We also stratify the split on the response variable,
which is very important to do because there are so few fraudulent transactions.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=1, stratify=y)

RANDOM FOREST:
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import cross_val_score

rfc = RandomForestClassifier()

rfc.fit(X_train,y_train)

predictR = rfc.predict(X_test)

print("")

```

```

print('Classification report of Random Forest Results:')
print("")

print(classification_report(y_test,predictR))
x = (accuracy_score(y_test,predictR)*100)

print('Accuracy result of Random Forest is:', x)
print("")
cm1=confusion_matrix(y_test,predictR)
print('Confusion Matrix result of Random Forest is:\n',cm1)
print("")
sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
print("")

accuracy = cross_val_score(rfc, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Random Forest is:",accuracy.mean() * 100)
RFC=accuracy.mean() * 100

def graph():
    import matplotlib.pyplot as plt
    data=[RFC]
    alg="Random orest"
    plt.figure(figsize=(5,5))
    b=plt.bar(alg,data,color=("b"))
    plt.title("Accuracy comparison of LIVER FAILURE")

```

```

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

graph()
TN = cm1[1][0]
FN = cm1[0][0]
TP = cm1[1][1]
FP = cm1[0][1]
print("True Positive :",TP)
print("True Negative :",TN)
print("False Positive :",FP)
print("False Negative :",FN)
print("")
TPR = TP/(TP+FN)
TNR = TN/(TN+FP)
FPR = FP/(FP+TN)
FNR = FN/(TP+FN)
print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)
print("")
PPV = TP/(TP+FP)
NPV = TN/(TN+FN)
print("Positive Predictive Value :",PPV)
print("Negative predictive value :",NPV)

def plot_confusion_matrix(cm2, title='Confusion matrix-RF', cmap=plt.cm.Blues):
    target_names=[""]
    plt.imshow(cm2, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(target_names))
    plt.xticks(tick_marks, target_names, rotation=45)
    plt.yticks(tick_marks, target_names)

```

```

plt.tight_layout()
plt.ylabel('True label')
plt.xlabel('Predicted label')

cm2=confusion_matrix(y_test, predictR)
print('Confusion matrix-RF:')
print(cm2)

sns.heatmap(cm2/np.sum(cm2), annot=True, cmap = 'Blues', annot_kws={"size":
16},fmt='.2%')
plt.show()

```

Module 6: Performance measurements of Support Vector Machines:

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# warning
import warnings
warnings.filterwarnings("ignore")
# load given data
data = pd.read_csv("liver_patient.csv")
data.head()
data.duplicated().sum()
data.shape
data = data.drop_duplicates()
data.shape
data.isnull().sum()
df=data.dropna()
df.shape
df.isnull().sum()

```

```

#changing feature name
df = df.rename({'Dataset': 'Result'}, axis=1)
df.columns
from sklearn.preprocessing import LabelEncoder
var_mod = ['Gender']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i]).astype(int)
df.Result.value_counts()
df.corr()
X = df.drop(labels='Result', axis=1)
#Response variable
y = df.loc[:, 'Result']
# now let's evaluate its performance on the test set.
from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score
#We'll use a test size of 30%. We also stratify the split on the response variable,
which is very important to do because there are so few fraudulent transactions.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=1, stratify=y)

```

Support Vector Machines:

```

from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import cross_val_score

```

```

s = SVC()

```

```

s.fit(X_train, y_train)

```

```

predicts = s.predict(X_test)

```

```

print("")
print('Classification report of Support Vector Machines Results:')
print("")

print(classification_report(y_test,predicts))
x = (accuracy_score(y_test,predicts)*100)

print('Accuracy result of Support Vector Machines is:', x)
print("")
cm2=confusion_matrix(y_test,predicts)
print('Confusion Matrix result of Support Vector Machines is:\n',cm2)
print("")
sensitivity1 = cm2[0,0]/(cm2[0,0]+cm2[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm2[1,1]/(cm2[1,0]+cm2[1,1])
print('Specificity : ', specificity1)
print("")

accuracy = cross_val_score(s, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Support Vector Machine is:",accuracy.mean() * 100)
S=accuracy.mean() * 100

def graph():
    import matplotlib.pyplot as plt
    data=[S]
    alg="Support Vector Machine"
    plt.figure(figsize=(5,5))
    b=plt.bar(alg,data,color=("b"))

```

```

plt.title("Accuracy comparison of LIVER FAILURE")
plt.legend(b,data,fontsize=9)

graph()
TN = cm2[1][0]
FN = cm2[0][0]
TP = cm2[1][1]
FP = cm2[0][1]
print("True Positive :",TP)
print("True Negative :",TN)
print("False Positive :",FP)
print("False Negative :",FN)
print("")
TPR = TP/(TP+FN)
TNR = TN/(TN+FP)
FPR = FP/(FP+TN)
FNR = FN/(TP+FN)
print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)
print("")
PPV = TP/(TP+FP)
NPV = TN/(TN+FN)
print("Positive Predictive Value :",PPV)
print("Negative predictive value :",NPV)

cm2=confusion_matrix(y_test, predicts)
print('Confusion matrix-SVM:')
print(cm2)

sns.heatmap(cm2/np.sum(cm2), annot=True, cmap = 'Blues', annot_kws={"size":
16},fmt='.2%')

```

plt.show()

HTML & CSS

```
<!DOCTYPE html>
<html >
<!--From https://codepen.io/frytyler/pen/EGdtg-->
<head>
  <meta charset="UTF-8">
  <title>TITLE</title>
<link rel="stylesheet" href="{{ url_for('static', filename='css/bootstrap.min.css') }}">
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
rel='stylesheet' type='text/css'>
<style>
.back{
  background-image: url('{{ url_for('static', filename='image/img.jpg') }}');
  background-repeat:no-repeat;
  background-size:cover;
}
.white{
color:white;
}
.space{
margin:15px 30px ;
padding:9px 10px;
background: palegreen;
```

```

width:400px
}
.gap{
padding:10px 20px;
}
.nspace{
margin:20px 30px ;
padding:9px 10px;
background: palegreen;
width:400px
}
</style>

```

```
</head>
```

```
<body class="back">
```

```
<div>
```

```
<div class="jumbotron">
```

```
<h1 style="text-align:center">LIVER FAILURE </h1>
```

```
</div>
```

```
<!-- Main Input For Receiving Query to our ML -->
```

```
<form class="form-group" action="{url_for('predict')}}"method="post">
```

```
<div class=" ">
```

```
<div class="row">
```

```
<div class="gap col-md-5 col-4 ">
```

```
<label class="white" for=""> Age </label>
```

```
<input type="number" class="space form-control"
step="0.01" name=" Age " placeholder=" Age " required="required" /><br>
```

```
<label class="white" for=""> Gender</label>
```

```
<select name='Gender' class="nspace form-control">
```

```
        <option value = 1> Male </option>
        <option value = 0> Female </option>
    </select>
```

```
        <label class="white" for=""> Total_Bilirubin</label>
        <input type="number" class="space form-control"
step="0.01" name=" Total_Bilirubin" placeholder=" Total_Bilirubin"
required="required" /><br>
```

```
        <label class="white" for=""> Direct_Bilirubin</label>
        <input type="number" class="space form-control"
step="0.01" name=" Direct_Bilirubin" placeholder=" Direct_Bilirubin"
required="required" /><br>
```

```
        <label class="white" for="">
Alkaline_Phosphotase</label>
        <input type="number" class="space form-control"
step="0.01" name=" Alkaline_Phosphotase" placeholder=" Alkaline_Phosphotase"
required="required" /><br>
```

```
</div>
```

```
<div class="gap col-md-5">
```

```
        <label class="white" for="">
Alamine_Aminotransferase</label>
        <input type="number" class="space form-control"
step="0.01" name=" Alamine_Aminotransferase" placeholder="
Alamine_Aminotransferase" required="required" /><br>
```

```

        <label class="white" for="">
Aspartate_Aminotransferase</label>
        <input type="number" class="space form-control"
step="0.01" name=" Aspartate_Aminotransferase" placeholder="
Aspartate_Aminotransferase" required="required" /><br>

        <label class="white" for=""> Total_Protiens</label>
        <input type="number" class="space form-control"
step="0.01" name=" Total_Protiens" placeholder=" Total_Protiens"
required="required" /><br>

        <label class="white" for=""> Albumin</label>
        <input type="number" class="space form-control"
step="0.01" name=" Albumin" placeholder=" Albumin" required="required" /><br>

        <label class="white" for="">
Albumin_and_Globulin_Ratio</label>
        <input type="number" class="space form-control"
step="0.01" name=" Albumin_and_Globulin_Ratio" placeholder="
Albumin_and_Globulin_Ratio" required="required" /><br>

```

```

</div>

```

```

</div>

```

```

</div>

```

```

<div style="padding:2% 35%">
    <button type="submit" class="btn btn-success btn-block"
style="width:350px;padding:20px">Predict</button>
</div>

```

```

    </form>
    <br>
    <br>
    <div style="background:skyblue;padding:2% 40%">
        {{ prediction_text }}
    </div>
</div>
</body>
</html>

```

DEPLOY- FLASK

```

import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle
import joblib

```

```

app = Flask(__name__)
model = joblib.load('rfc2.pkl')

```

```

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

```

```

@app.route('/predict',methods=['POST'])
def predict():
    """
    For rendering results on HTML GUI
    """
    int_features = [(x) for x in request.form.values()]
    final_features = [np.array(int_features)]
    print(final_features)

```

```

prediction = model.predict(final_features)

output = prediction[0]
print(output)
print(output)
if output == 2:
    output = 'Not Affected'
elif output == 1:
    output = 'Affected'

return render_template('index.html', prediction_text='LIVER FAILURE
{}'.format(output))

if __name__ == "__main__":
    app.run(host="localhost", port=7000)

```

OUTPUT SCREENSHOTS

LIVER FAILURE

Age

Age

Gender

Male

Alamine_Aminotransferase

Alamine_Aminotransferase

Aspartate_Aminotransferase

Aspartate_Aminotransferase

Total_Bilirubin

Total_Bilirubin

Total_Protiens

Total_Protiens

Direct_Bilirubin

Direct_Bilirubin

Albumin

Albumin

Alkaline_Phosphotase

Alkaline_Phosphotase

Albumin_and_Globulin_Ratio

Albumin_and_Globulin_Ratio

Predict

MAIN PAGE OF THE WEBPAGE

LIVER FAILURE

Age: 17

Gender: Male

Total Bilirubin: 0.9

Direct Bilirubin: 0.3

Alkaline Phosphatase: 202

Alanine Aminotransferase: 22

Aspartate Aminotransferase: 19

Total Proteins: 7.4

Albumin: 4.1

Albumin and Globulin Ratio: 1.2

Predict

LIVER FAILURE Not Affected

OUTPUT SHOWS THE PATIENT IS NOT AFFECTED BY LIVER DISEASE

LIVER FAILURE

Age: 45

Gender: Female

Alamine_Aminotransferase: 22

Aspartate_Aminotransferase: 30

Total_Bilirubin: 1.1

Total_Proteins: 8.1

Direct_Bilirubin: 0.4

Albumin: 4.1

Alkaline_Phosphatase: 214

Albumin_and_Globulin_Ratio: 1

Predict

LIVER FAILURE Affected

OUTPUT SHOWS THE PATIENT IS AFFECTED BY LIVER DISEASE

PLAIGARISM REPORT:

RE-2022-6614-plag-report

ORIGINALITY REPORT

9%

SIMILARITY INDEX

5%

INTERNET SOURCES

6%

PUBLICATIONS

4%

STUDENT PAPERS

PRIMARY SOURCES

1

www.ijitee.org

Internet Source

1%

2

Huili Zhang, Lehang Guo, Dan Wang, Jun Wang, Lili Bao, Shihui Ying, Huixiong Xu, Jun Shi. "Multi-Source Transfer Learning via Multi-Kernel Support Vector Machine Plus for B-Mode Ultrasound-Based Computer-Aided Diagnosis of Liver Cancers", IEEE Journal of Biomedical and Health Informatics, 2021

Publication

1%

3

Submitted to SRM University

Student Paper

1%

4

ijariie.com

Internet Source

1%

5

Submitted to University of Stirling

Student Paper

1%

6

acadpubl.eu

Internet Source

1%

7

"Advances in Data Science and Management", Springer Science and Business Media LLC,

1%

2022
Publication

8	ijstr.org Internet Source	1 %
9	www.mdpi.com Internet Source	<1 %
10	www.semanticscholar.org Internet Source	<1 %
11	Submitted to Liverpool John Moores University Student Paper	<1 %
12	arxiv.org Internet Source	<1 %

Exclude quotes ☐ On

Exclude matches ☐ Off

Exclude bibliography ☐ On