HEART DISEASE CLASSIFICATION USING MACHINE LEARNING ALGORITHMS

PROJECT REPORT

Submitted

in partial fulfillment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

in the faculty of

COMPUTER SCIENCE & ENGINEERING

By

CHITTURI RAJA RAJESWARI [R.NO. 17021A0525]

KASIMALLA ISWARYA [R.NO. 17021A0538]

KANCHARLA PREETHI CHOWDARY [R.NO. 17021A0549]

ADAPA VENKATA SAI NITESH [R.NO. 17021A0551]

Under the Guidance of

Dr. L. SUMALATHA

Professor

UNIVERSITY COLLEGE OF ENGINEERING (AUTONOMOUS)

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY

KAKINADA – 533003, A.P., INDIA

2020-2021

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY COLLEGE OF ENGINEERING (AUTONOMOUS)
JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY
KAKINADA – 533003, A.P., INDIA

DECLARATION FROM THE STUDENTS

We hereby declare that the project work described in this thesis, entitled ***"Heart Disease Classification Using Machine Learning Algorithms"***, which is being submitted by us in partial fulfillment of the requirements for the award of degree of ***Bachelor of Technology (B.Tech.)*** in the faculty of Computer Science & Engineering to the University College of Engineering (Autonomous), Jawaharlal Nehru Technological University Kakinada is the result of investigations carried out by us under the guidance of Dr. L. SumaLatha, Associate Professor in the Dept. of Computer Science & Engineering.

The work is original and has not been submitted to any other University or Institute for the award of any degree or diploma.

Place: Kakinada **Signature:**

Date:

CHITTURI RAJA RAJESWARI [17021A0525]

KASIMALLA ISWARYA [17021A0538]

KANCHARLA PREETHI CHOWDARY [17021A0549]

ADAPA VENKATA SAI NITESH [17021A0551]

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING (AUTONOMOUS)

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY

KAKINADA – 533003, A.P., INDIA

CERTIFICATE FROM THE SUPERVISOR

This is to certify that the project work entitled ***"Heart Disease Classification Using Machine Learning Algorithms"*** that is being submitted by Chitturi Raja Rajeswari (17021A0525), Kasimalla Iswarya (17021A0538), Kancharla Preethi Chowdary (17021A0549), Adapa Venkata Sai Nitesh (17021A0551) in partial fulfillment of the requirements for the award of degree of ***Bachelor of Technology (B.Tech.)*** in the faculty of Computer Science and Engineering to the Jawaharlal Nehru Technological University Kakinada is a record of Bonafide project work carried out by them under my guidance in the Department of Computer Science and Engineering.

Signature of Supervisor

Dr. L. SUMALATHA

Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING (AUTONOMOUS)

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY

KAKINADA – 533003, A.P., INDIA

CERTIFICATE FROM THE HEAD OF DEPARTMENT

This is to certify that the project work entitled ***"Heart Disease Classification Using Machine Learning Algorithms"*** that is being submitted by Chitturi Raja Rajeswari (17021A0525), Kasimalla Iswarya (17021A0538), Kancharla Preethi Chowdary (17021A0549), Adapa Venkata Sai Nitesh (17021A0551) in partial fulfillment of the requirements for the award of degree of ***Bachelor of Technology (B.Tech.)*** in the faculty of Computer Science and Engineering to the Jawaharlal Nehru Technological University Kakinada is a record of Bonafide project work carried out by them under my guidance in the Department of Computer Science and Engineering.

Signature of Head of the Department

Dr. D. Haritha

Head & Prof of CSE

ACKNOWLEDGEMENTS

We express our deep gratitude and regards to Dr. L. SumaLatha, internal guide and Associate Professor, department of Computer Science and Engineering, University College of Engineering (Autonomous), Jawaharlal Nehru Technological University, Kakinada for her encouragement and valuable guidance in bringing shape to this dissertation.

We are grateful to Dr. D. Haritha, Head of the department of Computer Science and Engineering, University College of Engineering (Autonomous), Jawaharlal Nehru Technological University, Kakinada for her encouragement and motivation.

We are thankful to all the Professors and Faculty Members in the department of their teachings and academic support and thanks to Technical Staff and Non-Teaching staff in the department for their support.

CHITTURI RAJA RAJESWARI [17021A0525]

KASIMALLA ISWARYA [17021A0538]

KANCHARLA PREETHI CHOWDARY [17021A0549]

ADAPA VENKATA SAI NITESH [17021A0551]

ABSTRACT

The heart-related sickness is the death of everyone these days. Spotting and safeguarding the diseases at a primitive stage is very crucial. Cardiovascular disease refers to the trouble that occurs with the heart. It specifically implies the condition of the heart that contracts or obstructs blood vessels which result in pain in chest and heart attack. Heart diseases can be predicted using Machine Learning Algorithms. This paper emphasizes on the diagnosis of heart diseases at a primitive stage so that it will lead to a successful cure of the diseases.

In this project, the dataset is taken from the Machine Learning UCI repository. The dataset is first Pre-processed using One Hot Encoding and Normalization. Chi-Square Test and Rough Sets are used to predict the important features of the Dataset for Feature Selection. The Algorithms that are used to predict heart disease are Logistic Regression, K-Nearest Neighbours, Support Vector Machine, Random Forest, Decision Trees, AdaBoost. The metrics used to measure the performance of our models are F1-score, Accuracy score and Classification Report, AUC_ROC curve. Out of all Algorithms used, Logistic Regression performs best by giving an accuracy of 90.47%.

Table of Contents

LIST OF FIGURES

CHAPTER - 1

INTRODUCTION

1.1 Introduction to Heart Diseases:

Human heart is the principal part of the human body. Basically, it regulates blood flow throughout our body. Any irregularity to heart can cause distress in other parts of body. Any sort of disturbance to normal functioning of the heart can be classified as a heart disease.

In today's contemporary world, heart disease is one of the primary reasons for occurrence of most deaths. Heart disease may occur due to unhealthy lifestyle, smoking, alcohol and high intake of fat which may cause hypertension. According to the World Health Organization more than 10 million die due to heart diseases every single year around the world. A healthy lifestyle and earliest detection are only ways to prevent the heart related diseases.

The main challenge in today's healthcare is provision of best quality services and effective accurate diagnosis. Even if heart diseases are found as the prime source of death in the world in recent years, they are also the ones that can be controlled and managed effectively. The whole accuracy in management of a disease lies on the proper time of detection of that disease. The proposed work makes an attempt to detect these heart diseases at early stage to avoid disastrous consequences.

Records of large set of medical data created by medical experts are available for analysing and extracting valuable knowledge from it. Data mining techniques are the means of extracting valuable and hidden information from the large amount of data available. Mostly the medical database consists of discrete information.

Hence, decision making using discrete data becomes complex and tough task. Machine Learning (ML) which is subfield of data mining handles large scale well-formatted dataset efficiently.

In the medical field, machine learning can be used for diagnosis, detection and prediction of various diseases. The main goal of this paper is to provide a tool for doctors to detect heart disease at early stage.

This in turn will help to provide effective treatment to patients and avoid severe consequences. ML plays a very important role to detect the hidden discrete patterns and thereby analyse the given data.

1.2 Introduction to Machine Learning:

1.2.1 What is Machine Learning:

Artificial Intelligence can enable the computer to think. Computer is made much more intelligent by AI. Machine learning is the subfield of AI study. Various researchers think that without learning, intelligence cannot be developed. **Arthur Samuel**, a pioneer in the field of artificial intelligence and computer gaming, coined the term **"Machine Learning"**. He defined machine learning as – **"Field of study that gives computers the capability to learn without being explicitly programmed"**.

In a very layman manner, Machine Learning (ML) can be explained as automating and improving the learning process of computers based on their experiences without being actually programmed i.e.; without any human assistance. The process starts with feeding good quality data and then training our machines (computers) by building machine learning models using the data and different algorithms. The choice of algorithms depends on what type of data do we have and what kind of task we are trying to automate.

1.2.2 Types in Machine Learning:

Two of the most widely adopted machine learning methods are **supervised learning** which trains algorithms based on example input and output data that is labeled by humans, and **unsupervised learning** which provides the algorithm with no labeled data in order to allow it to find structure within its input data.

1.2.2.1 Supervised Learning:

The majority of practical machine learning uses supervised learning.

Supervised learning is where you have input variables (x) and an output variable (Y) and you use algorithm to learn the mapping function from the input to the output.

$Y = f(X)$

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. It is called supervised learning because the process of algorithm learning from the training

dataset can be thought of as a teacher supervising the learning process. We know the correct answers; the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance.

Supervised learning problems can be further grouped into regression and classification problems

a) ***Classification:*** It is a Supervised Learning task where output is having defined labels (discrete value). The goal here is to predict discrete values belonging to a particular class and evaluates on the basis of accuracy. It can be either binary or multi class classification. In **binary** classification, model predicts either 0 or 1; yes or no but in case of **multi class** classification, model predicts more than one class.

   Example: Gmail classifies mails in more than one class like social, promotions, updates, and forum.

b) ***Regression:*** It is a Supervised Learning task where output is having continuous value. Example in above Figure B, Output – Wind Speed is not having any discrete value but is continuous in the particular range. The goal here is to predict a value as much closer to actual output value as our model can and then evaluation is done by calculating error value. The smaller the error the greater the accuracy of our regression model.

User ID	Gender	Age	Salary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	1
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	1
15728773	Male	27	58000	1
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	1
15727311	Female	35	65000	0
15570769	Female	26	80000	1
15606274	Female	26	52000	0
15746139	Male	20	86000	1
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1

Figure A: CLASSIFICATION

Temperature	Pressure	Relative Humidity	Wind Direction	Wind Speed
10.69261758	986.882019	54.19337313	195.7150879	3.278597116
13.59184184	987.8729248	48.0648859	189.2951202	2.909167767
17.70494885	988.1119385	39.11965597	192.9273834	2.973036289
20.95430404	987.8500366	30.66273218	202.0752869	2.965289593
22.9278274	987.2833862	26.06723423	210.6589203	2.798230886
24.04233986	986.2907104	23.46918024	221.1188507	2.627005816
24.41475295	985.2338867	22.25082295	233.7911987	2.448749781
23.93361956	984.8914795	22.35178837	244.3504333	2.454271793
22.68800023	984.8461304	23.7538641	253.0864716	2.418341875
20.56425726	984.8380737	27.07867944	264.5071106	2.318677425
17.76400389	985.4262085	33.54900114	280.7827454	2.343950987
11.25680746	988.9386597	53.74139903	68.15406036	1.650191426
14.37810685	989.6819458	40.70884681	72.62069702	1.553469896
18.45114201	990.2960205	30.85038484	71.70604706	1.005017161
22.54895853	989.9562988	22.81738811	44.66042709	0.264133632
24.23155922	988.796875	19.74790765	318.3214111	0.329656571

Figure B: REGRESSION

Figure 1.1

Both the above figures have labeled data set –

- **Figure A:** It is a dataset of a shopping store which is useful in predicting whether a customer will purchase a particular product under consideration or not based on his/her gender, age and salary.
  Input: Gender, Age, Salary
  Output: Purchased i.e., 0 or 1; 1 means yes, the customer will purchase and 0 means that customer won't purchase it.

- **Figure B:** It is a Meteorological dataset which serves the purpose of predicting wind speed based on different parameters.
  Input: Dew Point, Temperature, Pressure, Relative Humidity, Wind Direction
  Output: Wind Speed.

Some popular examples of supervised machine learning algorithms are:

1) Linear regression for regression problems.

2) Random forest for classification and regression problems.

3) Support vector machines for classification problems.

1.2.2.2 Unsupervised Learning:

Unsupervised learning is where you only have input data (X) and no corresponding output variables.

The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data.

These are called unsupervised learning because unlike supervised learning above there are no correct answers. Algorithms are left to their own devises to discover and present the interesting structure in the data. Here the training data that we are feeding is unstructured and unlabeled.

Unsupervised learning problems can be further grouped into clustering and association problems.

a) ***Clustering:*** Broadly this technique is applied to group data based on different patterns, our machine model finds. For example, in the given dataset we will not be given output parameter value, so this technique will be used to group data based on the input parameters provided by our data.
b) ***Association:*** This technique is a rule-based machine learning technique which finds out some very useful relations between parameters of a large

data set. For e.g., shopping stores use algorithms based on this technique to find out relationship between sale of one product w.r.t to others sale based on customer behavior. Once trained well, such models can be used to increase their sales by planning different offers.

Some popular examples of unsupervised learning algorithms are:

1) k-means for clustering problems.

2) Apriori algorithm for association rule learning problems.

1.2.2.3 Reinforcement Learning:

Reinforcement learning describes a class of problems where an agent operates in an environment and must *learn* to operate using feedback. In this technique, model keeps on increasing its performance using a Reward Feedback to learn the behavior or pattern. These algorithms are specific to a particular problem e.g., Google Self Driving car, AlphaGo where a bot competes with human and even itself to getting better and better performer of Go Game. Each time we feed in data, they learn and add the data to its knowledge that is training data. So, more it learns the better it gets trained and hence experienced.

1) Agents observe input.

2) Agent performs an action by making some decisions.

3) After its performance, agent receives reward and accordingly reinforce and the model stores in state-action pair of information.

Some popular examples of reinforcement learning algorithms include Q-learning, temporal-difference learning, and deep reinforcement learning.

1.2.3 Success of machine learning in classification:

Let us take a binary classification problem in Machine learning in medical science like heart disease prediction. Heart disease has created a lot of serious concerned among researches; one of the major challenges in heart disease is correct detection and finding presence of it inside a human. Early techniques have not been so much efficient in finding it even medical professor are not so much efficient enough in predicating the heart disease. There are various medical instruments available in the market for predicting heart disease there are two major problems in them, the first one is that they are very much expensive and second one is that they are not efficiently able to calculate the chance of heart

disease in human. According to latest survey conducted by WHO, the medical professional able to correctly predicted only 67% of heart disease so there is a vast scope of research in area of predicating heart disease in human.

In medical science classification problems like heart disease prediction, diabetics prediction, Cancer prediction etc are major challenges; because a lot of parameters and technicality is involved for accurately predicating these diseases. Machine learning could be a better choice for achieving high accuracy for predicating these diseases because this vary tool utilizes feature vector and its various data types under various condition for predicating these diseases, algorithms such as Naïve Bayes, Decision Tree, KNN, Neural Network, are used to predicate risk of these diseases each algorithm has its specialty such as Naive Bayes used probability for predicating disease, whereas decision tree is used to provide classified report for the disease, whereas the Neural Network provides opportunities to minimize the error in predication of disease. All these techniques are using old patient record for getting predication about new patient. This predication system for disease helps doctors to predict these diseases in the early stage of disease resulting in saving millions of lives.

1.2.4 Implementing machine learning using python:

Machine learning projects differ from traditional software projects. The differences lie in the technology stack, the skills required for a machine learning based project, and the necessity of deep research. To implement these types of ml projects, you should use a programming language that is stable, flexible, and has tools available. Python offers all of this, which is why we see lots of Python machine learning projects today.

From development to deployment and maintenance, Python helps developers be productive and confident about the software they're building. Benefits that make python the best fit for machine learning and AI-based projects include simplicity and consistency, access to great libraries and frameworks for AI and machine learning (ML), flexibility, platform independence, and a wide community. These add to the overall popularity of the language.

Implementing AI and ML algorithms can be tricky and requires a lot of time. To reduce development time, programmers turn to a number of Python frameworks and libraries. Here are some of them:

1) Keras, Tensor Flow, and Scikit-learn for machine learning.

2) NumPy for high-performance scientific computing and data analysis.

3) SciPy for advanced computing.

4) Pandas for general-purpose data analysis

5) Seaborn for data visualization

CHAPTER – 2

LITERATURE REVIEW

2.1 Related Work:

In the literature, there exists a number of studies that perform disease classification using machine learning algorithms on a variety of datasets.

Below are the studies that employ classification algorithms.

Youness Khourdifi and Mohamed Bahaj "**Heart Disease Prediction and Classification Using Machine Learning Algorithms Optimized by Particle Swarm Optimization and Ant Colony Optimization"**. In this paper the Fast Correlation-Based Feature Selection (FCBF) method applied as a first step (pre-treatment). When all continuous attributes are discretized, the attribute selection attributes relevant to mining, from among all the original attributes, are selected. Feature selection, as a pre-processing step to machine learning, is effective in reducing dimensionality, eliminating irrelevant data, increasing learning accuracy and improving understanding of results. In the second step, PSO and ACO are applied to select the relevant characteristics of the data set. The best subset of characteristics selected by the characteristic selection methods improves the accuracy of the classification. Therefore, the third step applies classification methods to diagnose heart disease and measures the classification accuracy to evaluate the performance of characteristic selection methods. The weka data-mining tool is used to analyze data from a heart disease. The main contributions of this paper are: Extraction of classified accuracy useful for heart disease prediction, Remove redundant and irrelevant features with Fast Correlation-Based Feature selection (FCBF) method, Optimizations with Particle Swarm Optimization PSO then they considered the result of PSO the initial values of Ant Colony Optimization ACO approaches, Comparison of different data mining algorithms on the heart disease dataset, Identification of the best performance-based algorithm for heart disease prediction.

G. Parthiban and S.K. Srivatsa "**Applying Machine Learning Methods in Diagnosing Heart Disease for Diabetic Patients"**. In this paper the classification algorithms like Naive bayes and Support vector machine are used to predict whether the diabetic patient is suffering from heart disease with indicating levels. They used the Naïve Bayes method to perform the mining and classification process. They used 10 folds cross validation to minimize any bias in the process and improve the efficiency of the process. The naïve bayes model exhibited a precision of 71% in average, recall of 74% in average, and F-measure of 71.2% in average. The SVM classifier was trained for different values of the RBF kernel parameters, C and γ. They used 10-fold cross validation again for testing the

accuracy of the classifier. Based on the exhaustive trials conducted, for C = 5.0 and γ = 1.0 the classifier exhibited the best accuracy of 94.60%. They showed that it is possible to diagnose heart disease vulnerability in diabetic patients with reasonable accuracy.

JIAN PING LI, AMIN UL HAQ, SALAH UD DIN, JALALUDDIN KHAN, ASIF KHAN, AND ABDUS SABOOR "**Heart Disease Identification Method Using Machine Learning Classification in E-Healthcare"**. In this paper, developed an efficient and accurate system to diagnose heart disease and the system is based on machine learning techniques. The system is developed based on classification algorithms includes Support vector machine, Logistic regression, Artificial neural network, K-nearest neighbor, Naïve bayes, and Decision tree while standard features selection algorithms have been used such as Relief, Minimal redundancy maximal relevance, least absolute shrinkage selection operator and Local learning for removing irrelevant and redundant features. They also proposed a novel fast conditional mutual information (FCMIM) feature selection algorithm to solve the feature selection problem. The features selection algorithms are used for feature selection to increase the classification accuracy and reduce the execution time of the classification systems. Furthermore, the leave one subject out cross-validation method has been used for learning the best practices of model assessment and for hyperparameter tuning. The performance measuring metrics are used for assessment of the performances of the classifiers. The performances of the classifiers have been checked on the selected features as selected by features selection algorithms. The experimental results showed that the proposed feature selection algorithm (FCMIM) is feasible with a classifier support vector machine for designing a high-level intelligent system to identify heart disease. The suggested diagnosis system (FCMIM-SVM) achieved good accuracy as compared to other proposed methods.

J. Vijayashree and H. Parveen Sultana "**A Machine Learning Framework for Feature Selection in Heart Disease Classification Using Improved Particle Swarm Optimization with Support Vector Machine Classifier**". In this paper, they presented a novel feature selection method on the basis of the PSO algorithm with SVM. The proposed PSO with SVM algorithm-based feature selection method identifies 6 significant features it includes sex, fasting blood sugar level, resting electrocardiographic result, maximum heart rate, exercise-induced angina real and a number of major vessels (3, 7, 8, 9, 10, 13) to classify the heart disease. The resultant features are supplied to SVM for finding the accuracy. The SVM produces 79.35% of accuracy when classifying heart disease with the whole

features. However, the SVM classifier produces 84.36% of accuracy when classifying heart disease with the selected features. They evaluated the performance of the proposed PSO-SVM method on the basis of comparison between the various existing feature selection algorithms such as Info gain, Chi-squared, one attribute based, Consistency subset, Relief, CFS, filtered subset, filtered attribute, Gain ratio and PSO algorithm. The good performance of the SVM classifier is proved on the basis of Receiver Operating Characteristic (ROC) analysis. They used the Cleveland heart disease database for the demonstration of the proposed PSO-SVM based features selection method to predict the heart disease. The MATLAB environment is used to perform the experimental evaluation. The SVM classifier is also compared with the various familiar classifier methods such as Naïve Bayes, Random Forest and MLP to prove the significance of the selected features.

M. Akhil Jabbar, B. L. Deekshatulu and Priti Chandra "**HEART DISEASE CLASSIFICATION USING NEAREST NEIGHBOR CLASSIFIER WITH FEATURE SUBSET SELECTION**". In this paper they investigate by applying KNN with feature subset selection to predict heart disease of a patient for Andhra Pradesh population. They used symmetrical uncertainty as a goodness measure to rank the attributes and based on the ranking they prune least ranking attributes. These evaluated attributes are given to the KNN algorithm which helps in classification of heart disease. The main package used in their experiments is WEKA 6.4. specifications of medical data sets, they ran full training and 10 cross fold validation. They showed the learning accuracy of algorithms like j48, naïve bayes and their approach on different data sets. From the averaged accuracy over all data sets, it is observed that their approach improves the accuracy. From the individual accuracy values, it can also be observed that their approach maintains or even increases the accuracy for most of the data sets. Their proposed algorithm reached 11.09% improvement over j48 and 17.7% improvement over naive bayes. Their approach achieved improved accuracy compared with other traditional classification algorithms. So, their approach performs better than traditional classification techniques. When comparing their proposed method in the diagnosis of heart disease for Andhra Pradesh population with other data mining techniques, their approach achieved higher accuracy of 2.5% over j48 and 25% over naïve bayes classification algorithms. The experimental results suggest that their approach efficiently achieves high degree dimensionality reduction and enhances accuracy with predominant features. This indirectly helps patient's no. of diagnosis tests to be taken for prediction of heart disease.

Nazia Hameed, Antesar M. Shabut, Miltu K. Ghosh, M.A. Hossain "**Multi-class multi-level classification algorithm for skin lesions classification using machine learning techniques**". In this paper, they proposed an intelligent digital diagnosis scheme to improve the classification accuracy of multiple diseases. A Multi-Class Multilevel (MCML) classification algorithm inspired by the "divide and conquer" rule is explored to address the research challenges. This paper presented an investigation into the development of an intelligent multi-class multi-level (MCML) classification algorithm to classify multiple skin diseases. The developed intelligent diagnosis scheme is expected to help the users and skin specialists in early skin lesion assessment. The proposed algorithm is implemented using two approaches, the traditional machine learning and deep learning. Pre-processing, segmentation, feature extraction and classification steps are involved in the traditional machine learning approach. Noise is removed in the pre-processing step, and a hybrid technique is implemented to get the region of interest in the segmentation step. Color and texture features are extracted in the feature extraction step and the image is classified in the classification step. Transfer learning is used for the deep learning approach and learns directly from the images. The proposed algorithm is compared with the multi-class single-level classification algorithm, and high accuracy is achieved by MCML algorithm in both traditional machine learning and deep learning approaches. Improved techniques are proposed for noise removal in the traditional machine learning approach. The proposed algorithm is evaluated on 3672 classified images, collected from different sources and the diagnostic accuracy of 96.47% is achieved. To verify the performance of the proposed algorithm, its metrics are compared with the Multi-Class Single-Level classification algorithm which is the main algorithm used in most of the existing literature. The results also indicate that the MCML classification algorithm is capable of enhancing the classification performance of multiple skin lesions.

Min Chen, Yixue Hao, Kai Hwang, Fellow, Lu Wang, and Lin Wang "**Disease Prediction by Machine Learning over Big Data from Healthcare Communities**". In this paper, they streamline machine learning algorithms for effective prediction of chronic disease outbreak in disease-frequent communities. They tested the modified prediction models over real-life hospital data collected from central China in 2013-2015. To overcome the difficulty of incomplete data, they used a latent factor model to reconstruct the missing data. The latent factor model is presented to explain the observable variables in terms of the latent variables They experimented on a regional chronic disease of cerebral infarction. They proposed a new convolutional neural network based multimodal disease risk prediction

(CNN-MDRP) algorithm using structured and unstructured data from hospitals. For the processing of medical text data, they utilized CNN based unimodal disease risk prediction (CNN-UDRP) algorithm which can be divided into the following five steps: Representation of text data, Convolution layer of text CNN, Pool layer of text CNN, Full connection layer of text CNN, CNN classifier. For Structured data, they used traditional machine learning algorithms, i.e., NB, KNN and DT algorithms to predict the risk of cerebral infarction disease. The accuracy of the three machine learning algorithms is roughly around 50%. Among them, the accuracy of DT which is 63% is highest, followed by NB and KNN. The recall of NB is 0.80 which is the highest, followed by DT and KNN. Compared to several typical prediction algorithms, the prediction accuracy of their proposed algorithm reaches 94.8% with a convergence speed which is faster than that of the CNN-based unimodal disease risk prediction (CNN-UDRP) algorithm.

Radhanath Patra "**Prediction of Lung Cancer Using Machine Learning Classifier**". In this paper they analyzed various machine learning classifiers techniques to classify available lung cancer data in UCI machine learning repository into benign and malignant. The input data is prepossessed and converted into binary form followed by use of some well-known classifier technique in Weka tool to classify the data set into cancerous and non-cancerous. Weka explore has various tabs for data analysis such as prepossess, classify, cluster, association, select attribute and visualize. When data prepossessing is selected it enables uploading the input data in the weka tool. Weka tool clearly understands and represents the data for easy data analysis. Before running any classification algorithm, Weka tool asks for various options like splitting percentage, used training set, supplied test set, Cross validation option etc. Classification mostly occurs with splitting of 80% training and 20% testing. But In Weka tool our analysis process was carried out with 10-fold cross validation with selected classifier technique to obtain an output of interest. Algorithms like j48, KNN, Naive Bias and RBF are used in Weka tool and a comparative analysis was derived finally. They showed that with RBF classifier the accuracy is found to be 81.25% on lung cancer data. So, In the analysis it can be predicted that with suitable feature selection method and integrated approach with other supervised learning processes and modified functional approach in RBF, accuracy will be further improved.

Akib Mohi Ud Din Khanday, Syed Tanzeel Rabani, Qamar Rayees Khan, Nusrat Rouf and Masarat Mohi Ud Din "**Machine learning based approaches for detecting COVID-19 using clinical text data**". In this paper, they classified textual clinical reports into four classes by using classical and ensemble machine learning

algorithms. Feature engineering was performed using techniques like Term frequency/inverse document frequency (TF/IDF), Bag of words (BOW) and report length. They had clinical text reports of 212 patients that are labelled into four classes. The classification was done using machine learning algorithms by supplying them features that were extracted in the feature engineering step. In order to explore the generalization of their model from training data to unseen data and reduce the possibility of overfitting, they split the initial dataset into separate training and test subsets. The tenfold cross-validation strategy was conducted for all algorithms, and this process was repeated five times independently to avoid the sampling bias introduced by randomly partitioning the dataset in the cross-validation. The features were supplied to traditional and ensemble machine learning classifiers. The results showed that logistic regression and Multinomial Naive Bayes Algorithm shows better result than all other algorithms by having precision 94%, recall 96%, F1 score 95% and accuracy 96.2% other algorithms like random forest, gradient boosting also showed good results by having accuracy 94.3% respectively. They also showed that the efficiency of models can be improved by increasing the amount of data.

Julie L. Harvey and Sathish A.P. Kumar "**Machine Learning for Predicting Development of Asthma in Children**". In this paper, they developed predictive models to analyze a child asthma health dataset. Machine learning classifiers are used to develop these predictive models; including Linear Regression, Decision Tree, Random Forest, KNN, and Naive Bayes technique. Of all the classifiers implemented, random forest classifier resulted in highest prediction accuracy (90.9%). Following are the variables: Sex, Difficulty Breathing, Allergies, and Medication have the highest correlation with asthma. These features can be focused on by physicians to help in diagnosis of childhood asthma. The review of current research and the results of models that are presented in this paper can be used in a clinical setting by medical professionals to make predictions of asthma development in children and implement early intervention for the treatment of asthma development.

CHAPTER – 3

PROPOSED SYSTEM

3.1 Objective of Proposed System

Cardiovascular diseases are very common these days, they describe a range of conditions that could affect your heart. World health organization estimates that 17.9 million global deaths from (cardiovascular diseases) CVDs.

This Project can help predict the people who are likely to diagnose with heart disease with the help of their medical history. It recognizes who all are having symptoms of heart and can help in diagnosing disease with fewer medical tests and effective treatments so that they can be cured accordingly.

The objective of this project is to check whether the patient is likely to be diagnosed with any cardiovascular heart diseases based on their medical attributes such as gender, age, chest pain, fasting sugar level, etc. The dataset is selected from the UCI repository with the patient's medical history and attributes. By using this dataset, we predict whether the patient can have heart disease or not. This Dataset has 13 medical attributes of a patient and classifies him/her if the patient is likely to have heart disease.

We have used several Machine Learning Algorithms like Logistic Regression, KNN, SVM, Random Forest and AdaBoost. Out of all these algorithms, Logistic Regression algorithms performs well and given accuracy of 90.47%. And, finally, we classify patients that are at risk of getting heart disease or not, and also this method is totally cost-efficient.

Flow of Proposed System

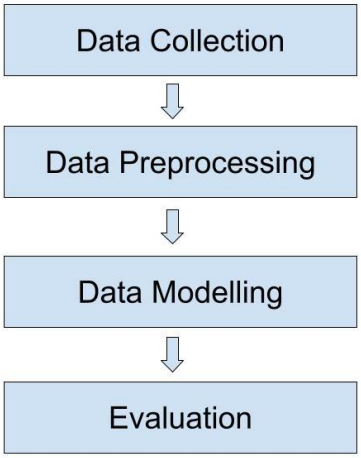

Figure – 3.1: Proposed Workflow

3.2 Data Collection

An Organized Dataset of individuals had been selected Keeping in mind their history of heart problems and in accordance with other medical conditions. Heart disease is the diverse conditions by which the heart is affected. According to World Health Organization (WHO), the greatest number of deaths in middle-aged people are due to cardiovascular diseases. We take a data source which is comprised of the medical history of 304 different patients of different age groups. This dataset gives us the much-needed information i.e., the medical attributes such as age, resting blood pressure, fasting sugar level, etc. of the patient that helps us in detecting the patient that is diagnosed with any heart disease or not. This dataset contains 13 medical attributes of 304 patients that help us detecting if the patient is at risk of getting a heart disease or not and it helps us classify patients that are at risk of having heart disease and those who are not at risk. This Heart Disease dataset is taken from the UCI repository. According to this dataset, the pattern which leads to the detection of a patient prone to getting heart disease is extracted. These records are split into two parts: Training and Testing. This dataset contains 303 rows and 14 columns, where each row corresponds to a single record.

3.3 Data Preprocessing

3.3.1 Analysis of Features

Features are the basic building blocks of datasets. The quality of the features in your dataset has a major impact on the quality of the insights you will gain when you use that dataset for machine learning. The dataset used for this project has 13 input features and one target feature. Detailed Explanation of Each Feature of the dataset is given below.

- age: The person's age in years
- sex: The person's sex (1 = male, 0 = female)
- cp: chest pain type
  — Value 0: asymptomatic
  — Value 1: atypical angina

— Value 2: non-anginal pain
— Value 3: typical angina

- trestbps: The person's resting blood pressure (mm Hg on admission to the hospital)
- chol: The person's cholesterol measurement in mg/dl
- fbs: The person's fasting blood sugar (> 120 mg/dl, 1 = true; 0 = false)
- restecg: resting electrocardiographic results
  — Value 0: showing probable or definite left ventricular hypertrophy by Estes' criteria
  — Value 1: normal
  — Value 2: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)
- thalach: The person's maximum heart rate achieved
- exang: Exercise-induced angina (1 = yes; 0 = no)
- oldpeak: ST depression induced by exercise relative to rest ('ST' relates to positions on the ECG plot. See more here)
- slope: the slope of the peak exercise ST segment — 0: downsloping; 1: flat; 2: upsloping
- ca: The number of major vessels (0–3)
- thal: A blood disorder called thalassemia Value 0: NULL (dropped from the dataset previously
  Value 1: fixed defect (no blood flow in some part of the heart)
  Value 2: normal blood flow
  Value 3: reversible defect (a blood flow is observed but it is not normal)
- target: heart disease (1 = no, 0= yes)

3.3.2 Encoding Categorical Features

One of the major problems with machine learning is that a lot of algorithms cannot work directly with categorical data. So, you must encode it to numbers before you can fit and evaluate a model. Encoding is a required pre-processing step when working with categorical data for machine learning algorithms.

Nominal and Ordinal Variables

- ***Nominal Variable (Categorical):*** Variable comprises a finite set of discrete values with no relationship between values.
- ***Ordinal Variable:*** Variable comprises a finite set of discrete values with a ranked ordering between values.

Encoding Categorical Data

There are three common approaches for converting ordinal and categorical variables to numerical values. They are:

- ***Ordinal Encoding****:* In ordinal encoding, each unique category value is assigned an integer value. For example, "*red*" is 1, "*green*" is 2, and "*blue*" is 3.
  This is called an ordinal encoding or an integer encoding and is easily reversible. Often, integer values starting at zero are used.
- ***One-Hot Encoding:*** For categorical variables where no ordinal relationship exists, the integer encoding may not be enough, at best, or misleading to the model at worst. Each bit represents a possible category. If the variable cannon belong to multiple categories at once, then only one bit in the group can be "on". This is called one-hot Encoding.
- ***Dummy Variable Encoding:*** The one-hot encoding creates one binary variable for each category. The problem is that this representation includes redundancy. For example, if we know that [1, 0, 0] represents "blue" and [0, 1, 0] represents "green" we don't need another binary variable to represent "red ", instead we could use 0 values for both "blue" and "green" alone, e.g. [0, 0]. This is called a dummy variable encoding, and always represents C categories with C-1 binary variables.

Implementing One Hot Encoding Using Pandas

We can implement one-hot encoding with pandas using the get_dummies () function. Pandas get_dummies () is the easiest way to implement a one-hot encoding method and it has very useful parameters. With get_dummies we can get a hot encoder data frame (dummy variables) in one row. We can implement this function in one line as given below.

```
dt['sex'].unique()
```

```
[1, 0]
Categories (2, int64): [1, 0]
```

Figure – 3.2

```
pd.get_dummies(dt['sex'])
```

	0	1
0	0	1
1	0	1
2	1	0
3	0	1
4	1	0
...	...	...
298	1	0
299	0	1
300	0	1
301	0	1
302	1	0

303 rows × 2 columns

Figure – 3.3

In the above pictures, sex is a feature with 2 values 0 for female and 1 for male. By using the get_dummies () function the featured sex is split into 2 features. In this way, one-hot encoding can be performed simply using pandas.

3.3.3 Normalization

Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to use a common scale, without distorting differences in the ranges of values or losing information. For machine learning, every dataset does not require normalization. It is required only when features have different ranges.

For example, assume your input dataset contains one column with values ranging from 0 to 1, and another column with values ranging from 10,000 to

100,000. The great difference in the *scale* of the numbers could cause problems when you attempt to combine the values as features during modelling. Normalization avoids these problems by creating new values that maintain the general distribution and ratios in the source data while keeping values within a scale applied across all numeric columns used in the model.

Normalization can be performed in different ways. In this project, Min-Max Normalization is used to normalize the data.

Min-Max Normalization

Min-max normalization is one of the most common ways to normalize data. For every feature, the minimum value of that feature gets transformed into a 0, the maximum value gets transformed into a 1, and every other value gets transformed into a decimal between 0 and 1.

For example, if the minimum value of a feature was 20, and the maximum value was 40, then 30 would be transformed to about 0.5 since it is halfway between 20 and 40. The formula is as follows:

$$\mathbf{v'} = \frac{\mathbf{v} - \mathbf{min(A)}}{\mathbf{max(A)} - \mathbf{min(A)}}(\mathbf{new\_max(A)} - \mathbf{new\_min(A)}) + \mathbf{new\_min(A)}$$

Figure – 3.4

Where A is the attribute data,

Min(A), Max(A) are the minimum and maximum absolute values of A respectively.
v' is the new value of each entry in data.
v is the old value of each entry in data.
new_max(A), new_min(A) is the max and min value of the range (i.e., boundary value of range required) respectively.

3.3.4 Feature Selection

The importance of feature selection can best be recognized when you are dealing with a dataset that contains a vast number of features. This type of dataset is often referred to as a high-dimensional dataset. Now, with this high dimensionality, comes a lot of problems such as - this high dimensionality will

significantly increase the training time of your machine learning model, it can make your model very complicated which in turn may lead to Overfitting.

Often in a high dimensional feature set, there remain several features that are redundant meaning these features are nothing but extensions of the other essential features. These redundant features do not effectively contribute to the model training as well. So, clearly, there is a need to extract the most important and the most relevant features for a dataset in order to get the most effective predictive modelling performance.

"The objective of variable selection is three-fold: improving the prediction performance of the predictors, providing faster and more cost-effective predictors, and providing a better understanding of the underlying process that generated the data."

In this project, the feature selection is performed using Chi-Square Test and Rough Sets.

Pearson's Chi-Squared Test

The Pearson's Chi-Squared test, or just the Chi-Squared test for short, is named for Karl Pearson, although there are variations on the test.

The Chi-Squared test is a statistical hypothesis test that assumes (the null hypothesis) that the observed frequencies for a categorical variable match the expected frequencies for the categorical variable. The test calculates a statistic that has a chi-squared distribution, named for the Greek capital letter Chi (X) pronounced "ki" as in kite.

The Formula for Chi Square Is

$$\chi_c^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

where:

c = degrees of freedom

O = observed value(s)

E = expected value(s)

Figure – 3.5

The Chi-Squared test does this for a contingency table, first calculating the expected frequencies for the groups, then determining whether the division of the groups, called the observed frequencies, matches the expected frequencies.

The result of the test is a test statistic that has a chi-squared distribution and can be interpreted to reject or fail to reject the assumption or null hypothesis that the observed and expected frequencies are the same.

We can interpret the test statistic in the context of the chi-squared distribution with the requisite number of degrees of freedom as follows:

- If Statistic >= Critical Value: significant result, reject the null hypothesis (H0), dependent.
- If Statistic < Critical Value: not significant result, fail to reject the null hypothesis (H0), independent.

The degrees of freedom for the chi-squared distribution is calculated based on the size of the contingency table as:

degrees of freedom: (rows - 1) * (cols - 1)

In terms of a p-value and a chosen significance level (alpha), the test can be interpreted as follows:

- If p-value <= alpha: significant result, reject null hypothesis (H0), dependent.
- If p-value > alpha: not significant result, fail to reject the null hypothesis (H0), independent.

For the test to be effective, at least five observations are required in each cell of the contingency table.

Rough Sets

Feature selection using rough set theory and Fuzzy-Rough has been done by many researchers. This area referring to the rough set is divided into two main categories as shown in Figure.

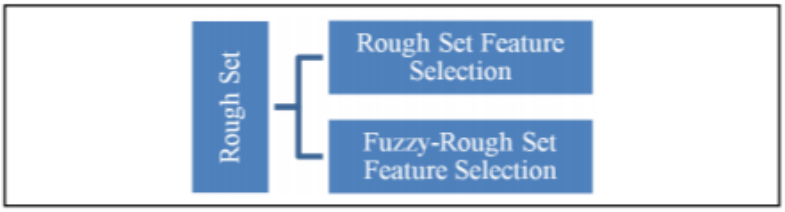

Figure – 3.6

Rough set-based feature selection methods Rough Set attribute selection is utilized when the value of attributes are discrete values whilst fuzzy rough approaches are employed where attributes have continuous values. Methods in both categories use the concept of dependency for finding reducts. One of the well-known methods is the so-called QuickReduct that is a greedy search algorithm using dependency.

The QuickReduct algorithm calculates a reduct without finding all subsets. It starts from an empty set and each time selects a feature that causes the greatest increase in dependency degree. The algorithm stops when adding more features does not increase the dependency degree. It does not guarantee to find minimal reduct as long as it employs a greedy algorithm which is a forward search and capable of being trapped in a local optimum.

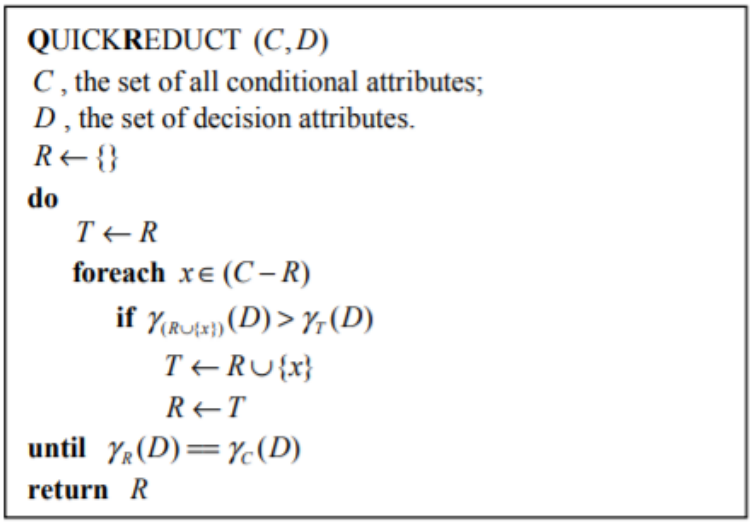

Figure – 3.7

3.4 Data Modelling

The process of training an ML model involves providing an ML algorithm (that is, the *learning algorithm*) with training data to learn from. The term *ML model* refers to the model artifact that is created by the training process.

The training data must contain the correct answer, which is known as a *target* or *target attribute*. The learning algorithm finds patterns in the training data that map the input data attributes to the target (the answer that you want to predict), and it outputs an ML model that captures these patterns. You can use the ML model to get predictions on new data for which you do not know the target.

The Algorithms used in this project are as follows: -

a) ***KNN:*** K-Nearest Neighbours (KNN) is one of the simplest algorithms used in Machine Learning for regression and classification problems. KNN algorithms use data and classify new data points based on similarity measures (e.g., distance function). Classification is done by a majority vote to its neighbours. The data is assigned to the class which has the nearest neighbours. As you increase the number of nearest neighbours, the value of k, accuracy might increase.

Algorithm :-

Input: Dtrain, K, xq (query point)

Output: yq (class to which out query point belongs to i.e. +ve or –ve)

```
k_pts=[ ]
for each xi in Dtrain:
    compute di (xi, xq)
sort di's in non-decreasing order
append k xi's in k_pts
cnt_pts=0, cnt_neg=0
for each xi in k_pts:
    if yi ==+ve:
        cnt_pts+=1
    else:
        cnt_neg+=1
if cnt_pst>cnt_neg:
    yq= +ve
else:
    yq= -ve
```

Figure – 3.8

b) ***Logistic Regression:*** Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of the target or dependent variable is dichotomous, which means there would be only two possible classes. In simple words, the dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no). Mathematically, a logistic regression model predicts P(Y=1) as a function of X. It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection, etc.

Training :

Step 1 : Set the following :
Acceptable threshold for Cost Function, ε.
Maximum Number of Epochs, N_{max}.
Number of Epochs, N.
Initialize Augmented Weight Matrix, $\theta = 1$.
Initialize the Cost Function, $J(\theta) = 0$.
Step 2 : Select a Mapping function for input features.
Step 3 : Update Augmented Weight Matrix θ using
$$\theta_j(n) = \theta_j(n-1) + \alpha \cdot \upsilon_j$$
Step 4 : Find the Cost Function or Average Cost $J(\theta)$ using
$$J(\theta) = J(\theta) + \left(-\frac{1}{m}\right)\left(\sum_{i=1}^{m} y^{(i)} \log\left(h_\theta(x^{(i)})\right) + (1 - y^{(i)}) \log\left(1 - h_\theta(x^{(i)})\right)\right)$$
Step 5 : If $|J(\theta)| \leq \varepsilon$ (or) $N = N_{max}$ Goto Step 6
Else Goto Step 3 to update the Augmented Weight Matrix θ
Step 6 : Optimum weights are obtained for θ.

Testing :

Find the output using optimum weights and test input
$y = sgn(\theta^T x)$

Figure – 3.9

c) ***SVM:*** Support vector machines (SVMs) are powerful yet flexible supervised machine learning algorithms that are used both for classification and regression. The objective of the SVM algorithm is to find a hyperplane in N-dimensional space (N- number of features) that distinctly classifies the data points. To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e.; the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

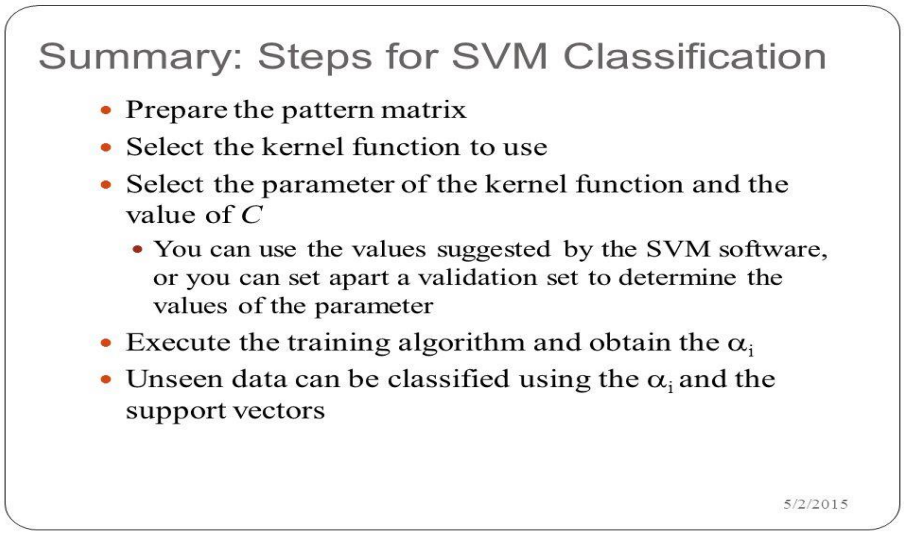

Figure – 3.10

d) ***Decision Trees:*** The decision tree Algorithm belongs to the family of supervised machine learning algorithms. It can be used for both a classification problem as well as for regression problem. The goal of this algorithm is to create a model that predicts the value of a target variable, for which the decision tree uses the tree representation to solve the problem in which the leaf node corresponds to a class label and attributes are represented on the internal node of the tree.

Algorithm: Generate_decision_tree. Generate a decision tree from the training tuples of data partition, *D*.

Input:

- Data partition, *D*, which is a set of training tuples and their associated class labels;
- *attribute_list*, the set of candidate attributes;
- *Attribute_selection_method*, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion consists of a *splitting_attribute* and, possibly, either a *split-point* or *splitting subset.*

Output: A decision tree.

Method:

```
(1)  create a node N;
(2)  if tuples in D are all of the same class, C, then
(3)      return N as a leaf node labeled with the class C;
(4)  if attribute_list is empty then
(5)      return N as a leaf node labeled with the majority class in D; // majority voting
(6)  apply Attribute_selection_method(D, attribute_list) to find the "best" splitting_criterion;
(7)  label node N with splitting_criterion;
(8)  if splitting_attribute is discrete-valued and
         multiway splits allowed then // not restricted to binary trees
(9)      attribute_list ← attribute_list − splitting_attribute; // remove splitting_attribute
(10) for each outcome j of splitting_criterion
     // partition the tuples and grow subtrees for each partition
(11)     let Dj be the set of data tuples in D satisfying outcome j; // a partition
(12)     if Dj is empty then
(13)         attach a leaf labeled with the majority class in D to node N;
(14)     else attach the node returned by Generate_decision_tree(Dj, attribute_list) to node N;
     endfor
(15) return N;
```

Figure – 3.11

e) ***Random Forest:*** Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.* As

the name suggests, ***"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."*** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and predicts the final output. **The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.**

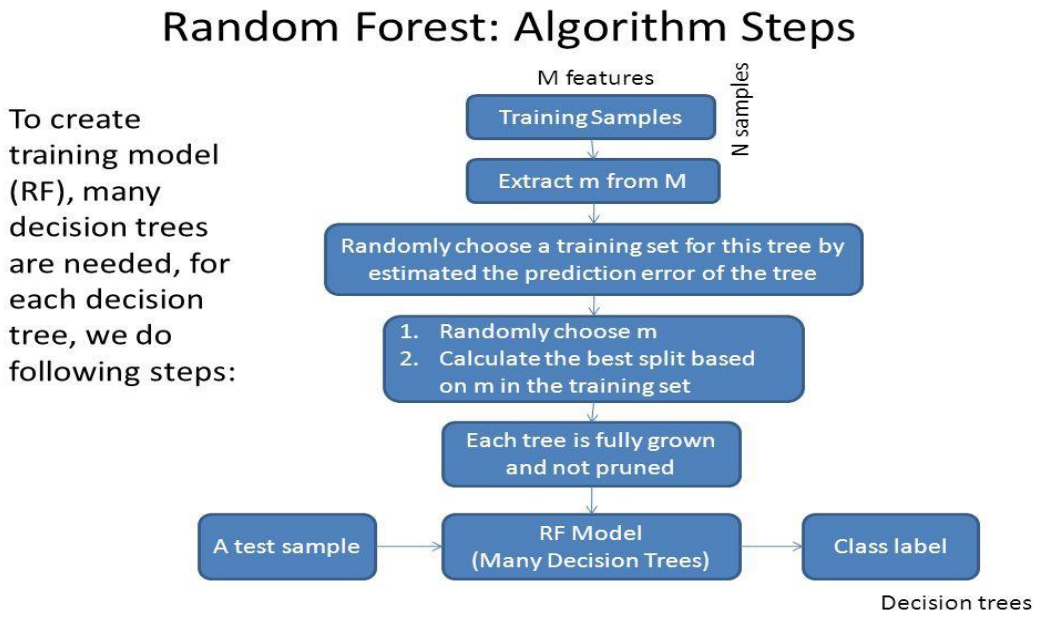

Figure – 3.12

f) ***AdaBoost:*** AdaBoost algorithm, short for Adaptive Boosting, is a Boosting technique that is used as an Ensemble Method in Machine Learning. It is called Adaptive Boosting as the weights are re-assigned to each instance, with higher weights to incorrectly classified instances. Boosting is used to reduce bias as well as the variance for supervised learning. It works on the principle where learners are grown sequentially. Except for the first, each subsequent learner is grown from previously grown learners. In simple words, weak learners are converted into strong ones.

Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

Figure – 3.13

Train Test Split

The train-test split is a technique for evaluating the performance of a machine learning algorithm. It can be used for classification or regression problems and can be used for any supervised learning algorithm.

The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the model; instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset.

- ***Train Dataset:*** Used to fit the machine learning model.
- ***Test Dataset:*** Used to evaluate the fit machine learning model.

The objective is to estimate the performance of the machine learning model on new data: data not used to train the model.

3.5 Evaluation

While data preparation and training a machine learning model is a key step in the machine learning pipeline, it's equally important to measure the performance of this trained model. How well the model generalizes on the unseen data is what defines adaptive vs non-adaptive machine learning models.

By using different metrics for performance evaluation, we should be in a position to improve the overall predictive power of our model before we roll it out for production on unseen data.

Without doing a proper evaluation of the ML model using different metrics, and depending only on accuracy, it can lead to a problem when the respective model is deployed on unseen data and can result in poor predictions.

This happens because, in cases like these, our models don't learn but instead memorize; hence, they cannot generalize well on unseen data.

Confusion Matrix

A confusion matrix is a matrix representation of the prediction results of any binary testing that is often used to describe the performance of the classification model (or "classifier") on a set of test data for which the true values are known.

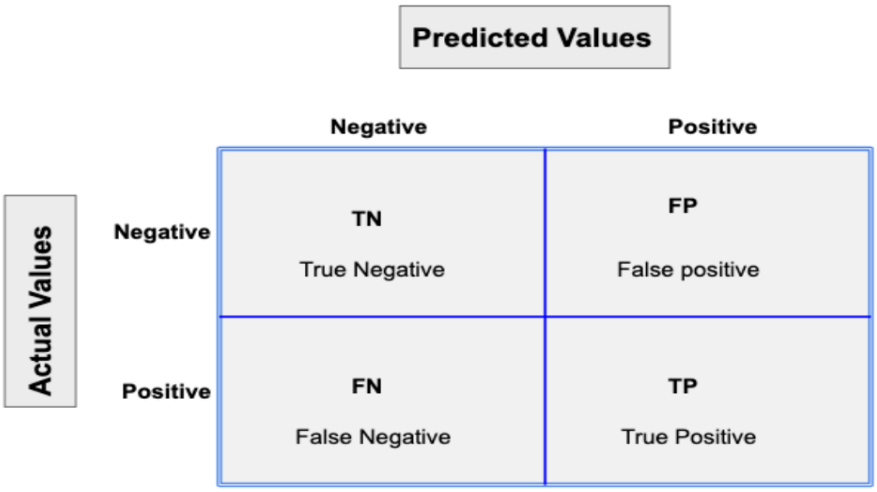

Figure – 3.14

Each prediction can be one of the four outcomes, based on how it matches up to the actual value:

- **True Positive (TP):** Predicted True and True in reality.
- **True Negative (TN):** Predicted False and False in reality.
- **False Positive (FP):** Predicted True and False in reality.
- **False Negative (FN):** Predicted False and True in reality.

Accuracy Score

Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right. Formally, accuracy has the following definition:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Figure – 3.15

For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Figure – 3.16

F1-Score

The F-score, also called the F1-score, is a measure of a model's accuracy on a dataset. It is used to evaluate binary classification systems, which classify examples into 'positive' or 'negative'.

The F-score is a way of combining the precision and recall of the model, and it is defined as the harmonic mean of the model's precision and recall.

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall}$$

Figure – 3.17

Classification Report

The classification report visualizer displays the precision, recall, F1, and support scores for the model.

```
from sklearn import metrics
import numpy as np
y_pred = np.around(model.predict(x_test))
print(metrics.classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.92      0.93      0.92     12500
           1       0.93      0.92      0.92     12500

   micro avg       0.92      0.92      0.92     25000
   macro avg       0.92      0.92      0.92     25000
weighted avg       0.92      0.92      0.92     25000
```

Figure – 3.18

Area Under Curve (A U C)

The **Receiver Operator Characteristic (ROC)** curve is an evaluation metric for binary classification problems. It is a probability curve that plots the **TPR** against **FPR** at various threshold values and essentially **separates the 'signal' from the 'noise'**. The **Area Under the Curve (AUC)** is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.

True positive rate: Also called or termed as sensitivity. True Positive Rate is considered as a portion of positive data points which are correctly considered as positive, with respect to all data points those are positives.

True Negative Rate: Also called or termed as specificity. False Negative Rate is considered as a portion of negative data points which are correctly considered as negative, with respected to all data points those are negatives.

False-positive Rate: False Negative Rate is considered as a portion of negative data points which are mistakenly considered as negative, with respected to all data points those are negatives.

Greater the value of AUC better the performance of the model.

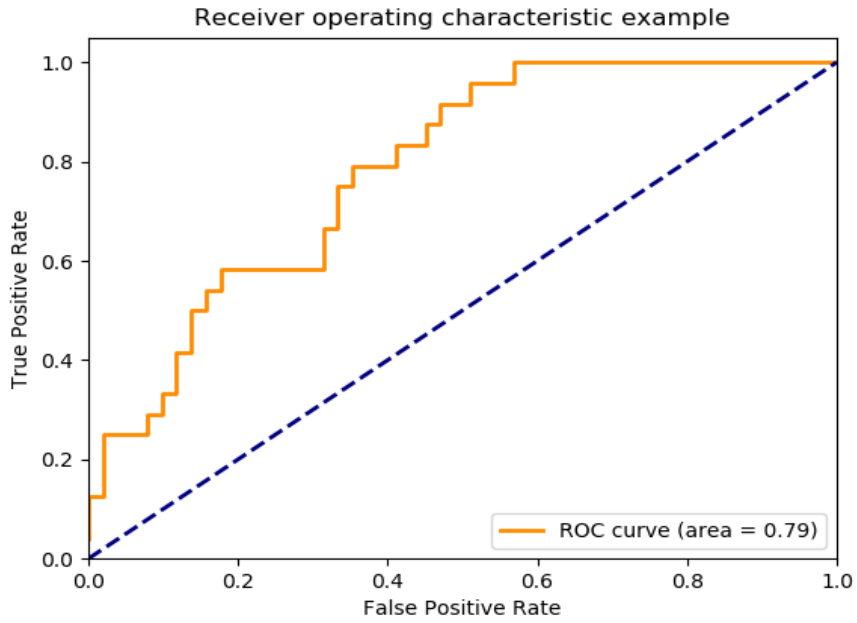
Receiver operating characteristic example
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.79)

Figure – 3.19

CHAPTER – 4

SYSTEM DESIGN

4.1 Use Case Diagram

Use case diagrams are usually referred to as behavior diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors). Each use case should provide some observable and valuable result to the actors or other stakeholders of the system. The use case diagram for this project is as follows

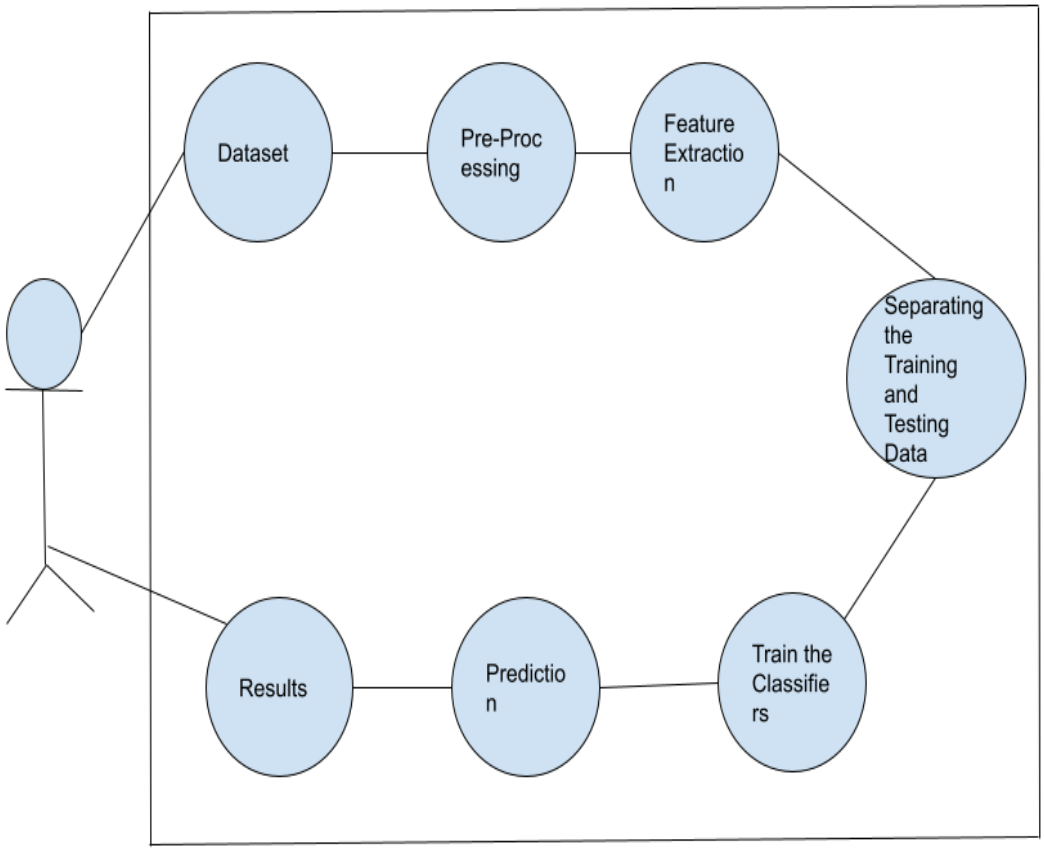

Figure – 4.1

4.2 Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language (UML), activity diagrams are intended to model both computational and organizational processes. The steps involved in our project are shown in the form of an activity diagram below.

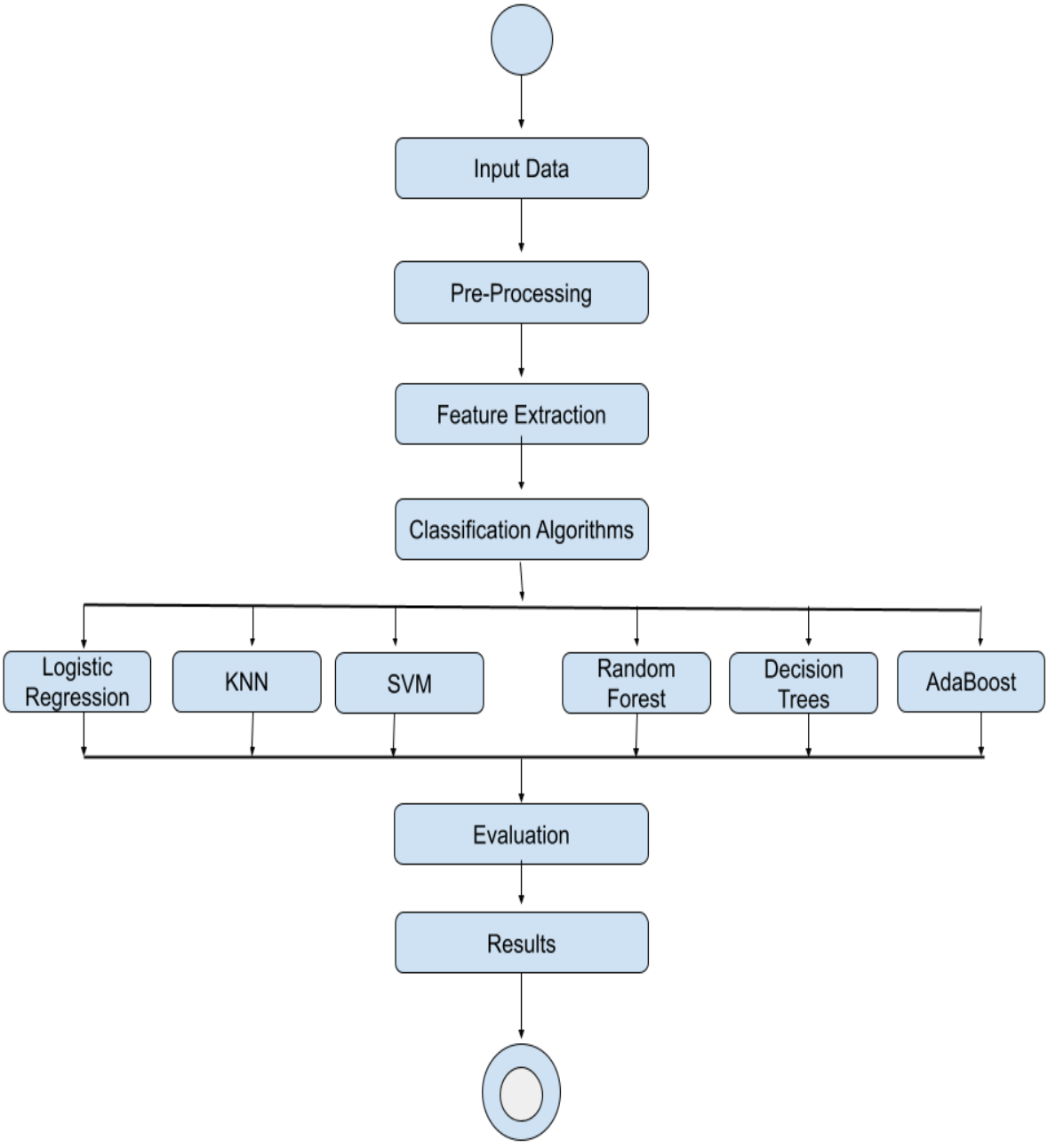

Figure – 4.2

4.3 Sequence Diagram

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time. They're also called event diagrams. A sequence diagram is a good way to visualize and validate various runtime scenarios. These can help to how a system will behave and to discover responsibilities a class may need to have in the process of modeling a new system. The Sequence diagram for this project is as follows,

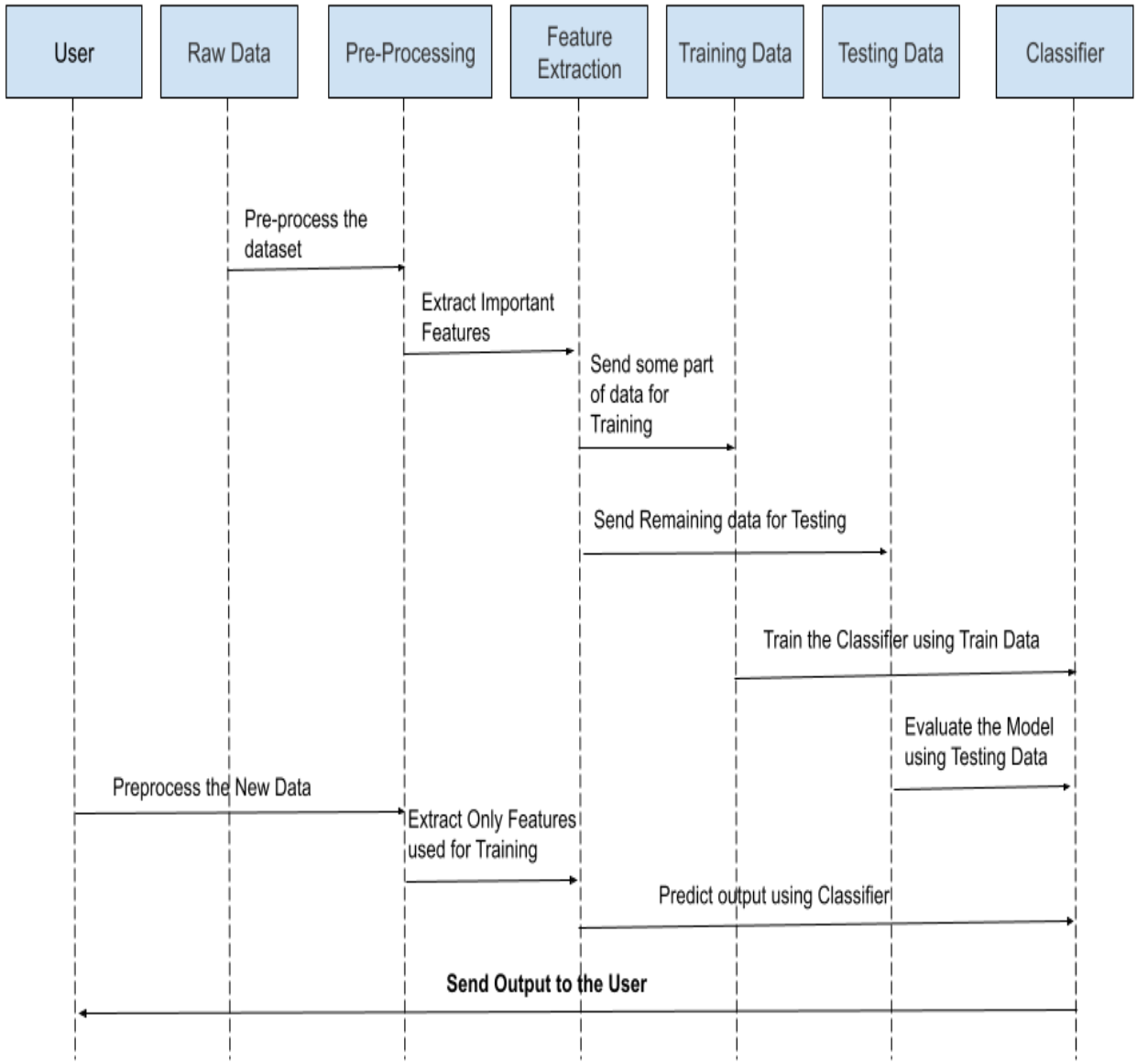

Figure – 4.3

4.4 Data Flow Diagram

A data flow diagram (DFD) illustrates how data is processed by a system in terms of inputs and outputs. As its name indicates its focus is on the flow of Information, where data comes from, where it goes and how it gets stored. The Dataflow diagram for this project is as follows,

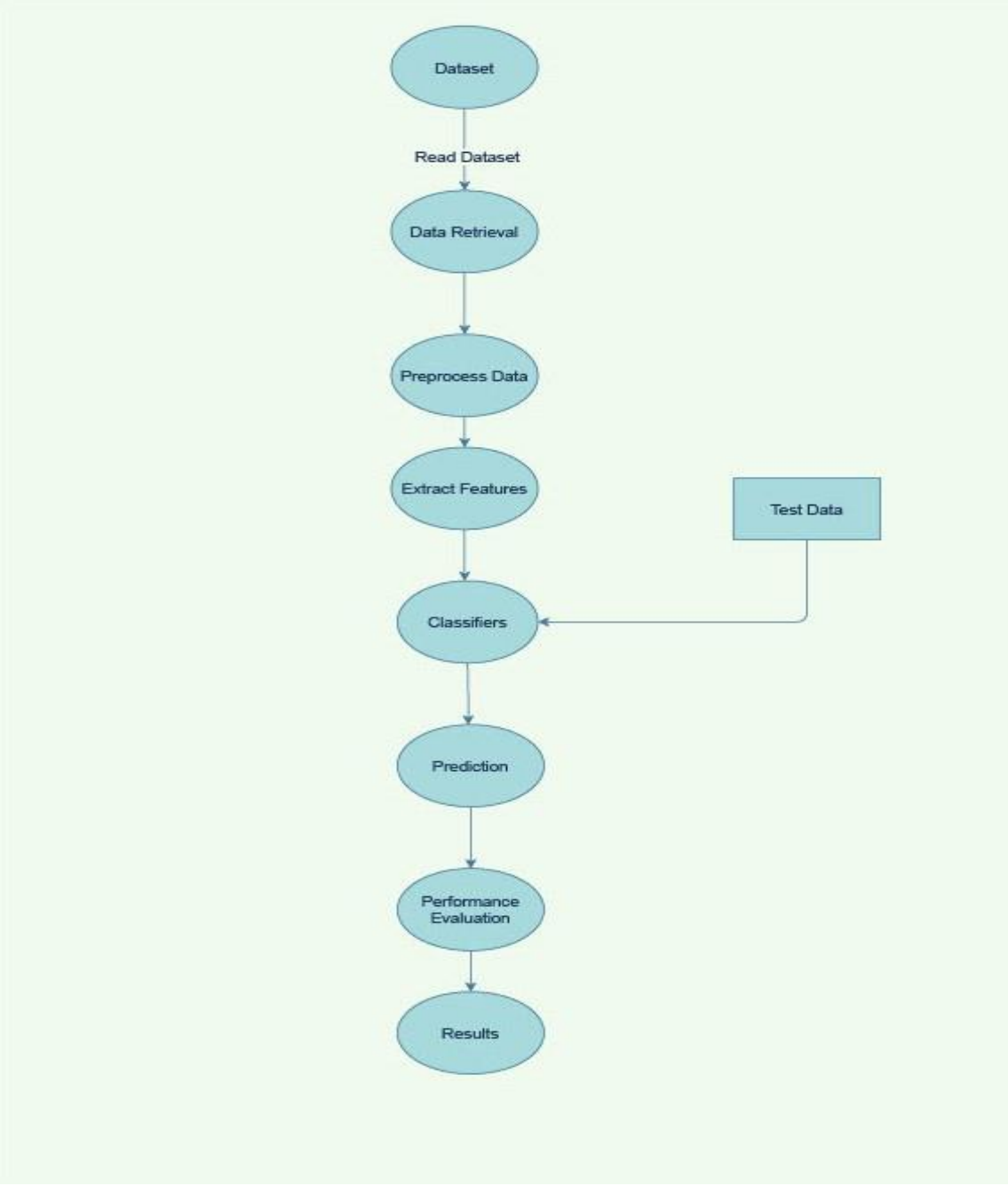

Figure – 4.4

CHAPTER – 5

IMPLEMENTATION AND RESULT ANALYSIS

5.1 Configuration of the system:

5.1.1 Hardware specifications:

As the project involves dealing with a small dataset which contains 303 rows and 14 columns, less hardware specifications are enough to run this project. We used an Intel Core I5 processor with 8GB RAM and 1TB storage working with more than 2.4 GHz processing power to implement this project.

***Processor*:** The processor should have a configuration more than Intel Core I3.

***RAM*:** The RAM should be 8GB or more.

***Storage*:** The system storage can be 500GB or more. Processing power: The processing power should be more than 2.40 GHz

5.1.2 Software specifications:

We implemented the project in Jupyter notebook using Python Programming Language. We have Imported different libraries provided by Python to implement this project.

Programming language: Python is used as the programming language for this project. Python is chosen because of its support to machine learning field by providing a wide range of libraries. Each algorithm can be implemented using a combination of libraries. Not only for the algorithms, it provides support to representation of the output in terms of graphs or tables. Python is easy to learn and it is fast as it is an interpreted language. The version of python is 3.6 or higher.

Google Colab Notebook: Google Colab is a free Jupyter notebook environment that runs entirely in the cloud. Most importantly, it does not require a setup and the notebooks that you create can be simultaneously edited by your team members - just the way you edit documents in Google Docs. Colab supports many popular machine learning libraries which can be easily loaded in your notebook.

Packages:

scikit-roughsets: This is an implementation of rough sets feature reduction algorithm, which can be used to filter the features which contribute more towards predicting target value. This is used for feature importance as a pre-processing step.

Libraries:

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.

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 time series. This library is built on the top of the NumPy library. Pandas is fast and it has high-performance & productivity for users.

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+.

Sklearn: Scikit-learn is a free machine learning library for Python. It features various algorithms like support vector machines, random forests, and k-neighbors, and it also supports Python numerical and scientific libraries like NumPy and SciPy and its core algorithms are written in Cython for even better performance.

Steps Of Implementation: -

1) Installing required packages
2) Importing required libraries
3) Loading dataset
4) Encoding the Categorical Variables
5) Normalizing the features of dataset
6) Feature Importance using chi-square Test and Roughsets
7) Training and Testing
8) Source Code
9) Result Analysis

5.2 Installing packages

pip is the standard package manager for Python. We can use pip to install additional packages that are not available in the Python standard library

To install a new package, use pip with an install command followed by the name of the package you want to install. pip looks for the package in PyPI, calculates its dependencies, and installs them to ensure requests will work.

In this project we have installed a package named ***scikit-roughsets.*** This package allows to construct rough sets by defining lower and upper approximations. Furthermore, recent methods for tackling common tasks in data mining, such as data preprocessing like discretization, feature selection, missing value completion, and instance selection, rule induction, and prediction classes or decision values of new datasets are available as well.

5.3 Importing Libraries

Import statement can be used to import the libraries or packages or modules required to implement the logic.

we can import libraries or, packages and modules. we can also import specific objects from a package or module. There are generally two types of import syntax.

When you use the first one, you import the resource directly.

import numpy

where numpy is a package.
When user uses the second syntax, then user import the resource from another package or module.

from math import pi

where math is a module, pi is a variable

5.4 Loading Dataset in Google Colab

In your Google Drive ("**My Drive**"), create a folder called **data** in the location of your choosing. This is where you will upload your data.
From a Colab notebook, type the following:

```
from google. colab import drive
drive. Mount('/content/drive')
```

the commands will bring you to a Google Authentication step. You should see a screen with **Google Drive File Stream wants to access your Google Account.** After you allow permission, copy the given verification code and paste it in the box in Colab.

Now you can access the csv file from the google drive using read_csv () method by passing the location of file in google drive.

5.5 Encoding Categorical Variables

In this project the categorical variables that are found in the dataset are **sex, cp, fbs, restecg, exang, slope, ca, thal**.

These categorical features can be encoded using one-hot encoding so that they contribute more information for the prediction of the target variable.

In this project we have used pandas get_dummies () method to perform encoding of all the categorical variables.
The dataset before performing encoding is as follows:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Figure – 5.1

The dataset after performing encoding is as follows:

	age	trestbps	chol	thalach	oldpeak	sex_1	cp_1	cp_2	cp_3	fbs_1	restecg_1	restecg_2	exang_1	slope_1	slope_2	ca_1	ca_2	ca_3	ca_4	thal_1	thal_2	thal_3
0	63	145	233	150	2.3	1	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0
1	37	130	250	187	3.5	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0
2	41	130	204	172	1.4	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0
3	56	120	236	178	0.8	1	1	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0
4	57	120	354	163	0.6	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	1	0

Figure – 5.2

5.6 Normalization

The process of Normalization is used when a dataset has features that have values of different ranges. Many Machine Learning Algorithms were internally implemented by calculating distance between the points. So, using those algorithms for the dataset that have values in different ranges will not give good accuracy.

So, normalization is used to convert all the values of the dataset into a same range so that the algorithms work better and give good accuracy.

In this project Min-Max Normalization is used to normalize the features of the dataset.

Before Normalization the dataset is same as Figure – 5.2

After Normalization the dataset is as follows:

	age	trestbps	chol	thalach	oldpeak	sex_1	cp_1	cp_2	cp_3	fbs_1	restecg_1	restecg_2	exang_1	slope_1	slope_2	ca_1	ca_2	ca_3	ca_4	thal_1	thal_2	thal_3
0	0.708333	0.481132	0.244292	0.603053	0.370968	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
1	0.166667	0.339623	0.283105	0.885496	0.564516	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
2	0.250000	0.339623	0.178082	0.770992	0.225806	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
3	0.562500	0.245283	0.251142	0.816794	0.129032	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
4	0.583333	0.245283	0.520548	0.702290	0.096774	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0

Figure – 5.3

5.7 Feature Importance

Feature Importance is the most necessary step to be performed before training the dataset. A dataset may contain many features where some features are very important and some features are not much important. Using Features that are not important will degrade the performance of the algorithm.

Using Feature Importance techniques, we can find the important features that contribute more for the prediction of target variable. There are many techniques available for finding the important features.

In this project we have used chi-square Test and Roughsets for finding the important features.

By using Chi-Square Test, we found that the features that are more important are thalach, oldpeak, sex_1, cp_1, cp_2, restecg_1, exang_1, slope_1, slope_2, ca_1, ca_2, ca_3, thal_2, thal_3.

By using Roughsets directly on the dataset without performing encoding and normalization, we found the features that are more important are age, trestbps, chol.

5.8 Training and Testing

Training is a step where an Algorithm is given a dataset as input and the algorithm gets trained based on the combinations of inputs and outputs of the provided dataset so that it can predict the output for a given new input.

As the dataset taken for this project contains 303 rows, we have divided the dataset into Train Dataset and Test Dataset. The Algorithms will be trained on the Train Dataset and the testing is performed on new data which is Test Dataset.

We can split our Dataset into 2 parts using train_test_split() method available in Sklearn library easily.

In this way we can evaluate our model based on Training Accuracy and Testing Accuracy and confirm whether the model is overfitting or underfitting.

Overfitting: when a model performs too well on the training data but the performance drops significantly on test data it is called as Overfitting
Underfitting: when a model performs poorly on both train data and test data then it is called as underfitting.

In this project we have trained on different algorithms such as Logistic Regression, K-Nearest Neighbor's, Support Vector Machines, Decision Trees and Random Forests. We have trained the algorithms 2 times using 2 different feature importance techniques which are Chi-Square Test and Roughsets.

5.9 Result Analysis

In this project we have used accuracy_score and f1-score to measure the performance of different algorithms.

The accuracies of different algorithms are as follows,

	Model	Accuracy Score	F1 Score
2	LR_model	89.473684	90.476190
1	KNN	89.473684	90.243902
0	SVM	86.842105	88.095238
4	RF_model	82.894737	84.337349
3	DT_model	75.000000	77.108434
5	AdaBoost_model	75.000000	77.108434

Figure – 5.4

Out of all the algorithms, Logistic Regression is performing best by giving the accuracy of 90.47%.

The accuracies obtained by using Roughsets as Feature Importance technique is,

	Model	Accuracy Score	F1 Score
0	SVM	53.947368	70.085470
2	LR_model	59.210526	65.934066
1	KNN	55.263158	60.465116
4	RF_model	52.631579	56.097561
5	AdaBoost_model	51.315789	51.948052
3	DT_model	48.684211	49.350649

Figure – 5.5

By using Roughsets Support Vector Machine Algorithm is performing best by giving accuracy of 70%.

But compared to Chi-Square Test the performance of Roughsets is very poor.

5.10 Source Code

1 Installing Required Packages

In [ ]:

```
pip install --upgrade https://github.com/paudan/scikit-roughsets/tarball/master
```

2 Importing Required Libraries

In [ ]:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,precision_recall_curve,classification_report,f1_score
from sklearn.preprocessing  import  MinMaxScaler
from scikit_roughsets.rs_reduction import RoughSetsSelector
from scipy import stats
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.svm import SVC
import matplotlib.pyplot as plt
```

In [ ]:

```
from google.colab import drive
drive.mount("/content/gdrive")
```

3 Loading Dataset

In [ ]:

```
dt = pd.read_csv('/content/gdrive/My Drive/42 Project/h
```

```
eart.csv')
```

In [ ]:

```
dt.describe()
```

In [ ]:

```
dt.head()
```

4 Data Preprocessing

In [ ]:

```
dt.shape
```

4.2 Checking for null Values

In [ ]:

```
dt.isnull().sum()
```

4.3 Types of Features

In [ ]:

```
dt.dtypes
```

4.4 Encoding Categorical Features

In [ ]:

```
dt.sex=dt.sex.astype('category')
dt.cp=dt.cp.astype('category')
dt.fbs=dt.fbs.astype('category')
dt.restecg=dt.restecg.astype('category')
dt.exang=dt.exang.astype('category')
dt.ca=dt.ca.astype('category')
dt.slope=dt.slope.astype('category')
dt.thal=dt.thal.astype('category')
```

In [ ]:

```
model_label_values=dt['target']
model_label=pd.DataFrame(model_label_values)
del dt['target']
```

In [ ]:

```
dt1=pd.get_dummies(dt,drop_first=True)
dt1.head()
```

4.5 Normalizing the dataset using MinMaxScaler

In [ ]:

```
dt1_scaled=MinMaxScaler().fit_transform(dt1)
dt1_scaled=pd.DataFrame(data=dt1_scaled, columns=dt1.co
lumns)
dt1_scaled.head()
```

4.6 Feaure Importance Using ChiSquare Test

In [ ]:

```
x = dt1_scaled.columns
l=[]
for i in x:
    statistic,p,dof,freq=stats.chi2_contingency(pd.cros
stab(dt1_scaled[i],model_label_values))
    prob=0.90
    alpha=1-prob
    critical=stats.chi2.ppf(prob,dof)
    if abs(statistic)>=critical:
        l.append(i)
        print(i,":dependant")
    else:
        print(i,":independant")
    if p<=alpha:
        print(i,":dependant")
    else:
        print(i,":independent")
l
```

In [ ]:

```
dt1_scaled_new = dt1_scaled[1]
```

In [ ]:

```
dt1_scaled_new.head()
```

5 Modelling

5.1 Different Algorithms used for training

In [ ]:

```
# define models to test:
base_models = [("SVM",       SVC(kernel='rbf',C=0.1,gamm
a=0.1,probability=True)),                    #Support Vec
tor Machines
               ("KNN", KNeighborsClassifier(n_neighbors
=18)),                                       #KNN
               ("LR_model", LogisticRegression(random_s
tate=42,n_jobs=-1)),                         #Logistic Re
gression model
               ("DT_model", DecisionTreeClassifier(rand
om_state=42)),                               #Decision tr
ee model
               ("RF_model", RandomForestClassifier(rand
om_state=42, n_jobs=-1)),                    #Random Fore
st
                ("AdaBoost_model",AdaBoostClassifier(De
cisionTreeClassifier(),                       #AdaBoost
model
                                                    n_e
stimators=100,
                                                    lea
rning_rate=0.01)),
               ]
```

5.2 Train Test Split

In [ ]:

```
model = base_models
```

```
X_train, X_test, y_train, y_test = train_test_split(dt1
_scaled,np.ravel(model_label),random_state=25,stratify=
model_label)
```

5.3 Training and Testing

In [ ]:

```
train_accuracies = {}
test_accuracies = {}
model = base_models
y_test_pred_values_for_models = []
X_train, X_test, y_train, y_test = train_test_split(dt1
_scaled_new,np.ravel(model_label),random_state=25,strat
ify=model_label)
for name,model in base_models:
    model.fit(X_train, y_train)

    y_train_pred = model.predict(X_train)
    fscore_train = f1_score(y_train,y_train_pred)*100
    acc_train = model.score(X_train,y_train)*100

    acc = model.score(X_test,y_test)*100
    y_pred = model.predict(X_test)
    fscore = f1_score(y_test,y_pred)*100
    y_test_pred_values_for_models.append(y_pred)

    train_accuracies[name] = [acc_train,fscore_train]
    test_accuracies[name] = [acc,fscore]
```

6 Evaluation

6.1 Accuracies on Train Data

In [ ]:

```
names = list(train_accuracies.keys())
models_res = pd.DataFrame(data=train_accuracies.values(
),columns=["Accuracy Score","F1 Score"])
models_res.insert(0,"Model",names,True)
models_res.sort_values('F1 Score',ascending=False)
```

6.2 Accuracies on Test Data

In [ ]:

```
names = list(test_accuracies.keys())
models_res = pd.DataFrame(data=test_accuracies.values()
,columns=["Accuracy Score","F1 Score"])
models_res.insert(0,"Model",names,True)
models_res.sort_values('F1 Score',ascending=False)
```

6.3 Classification Report for Logistic Regression Algorithm

In [ ]:

```
print(classification_report(y_test,y_test_pred_values_f
or_models[2], target_names=["Yes","No"]))
```

6.4 Precision Recall Curve and ROC Curve for Logistic Regression Algorithm

In [ ]:

```
from sklearn.metrics import roc_curve,auc#x-axis->False
positives,y-axis->true positives
from sklearn.metrics import average_precision_score#wei
ghted mean of precision with weight
def plotting(true_values,pred_values):
    fig,axe=plt.subplots(1,2,figsize=(12,6))
    precision,recall,threshold = precision_recall_curve
(true_values,pred_values[:,1])#returns three arrays of
precision,recalls and thresholds with respect to which
those are attained
    axe[0].plot(precision,recall,'r--')
    axe[0].set_xlabel('Precision')
    axe[0].set_ylabel('Recall')
    axe[0].set_title("Average Precision Score : {}".for
mat(average_precision_score(true_values,pred_values[:,1
])))#probabilities of 1's-pred[:,1] means
    fpr,tpr,threshold = roc_curve(true_values,pred_valu
es[:,1])#fpr->false positives,tpr->true positive
    axe[1].plot(fpr,tpr)
```

```
    axe[1].set_title("AUC Score is: {}".format(auc(fpr,
tpr)))#area under curve
    axe[1].set_xlabel('False Positive Rate')
    axe[1].set_ylabel('True Positive Rate')
```

In [ ]:

```
plotting(y_test,base_models[2][1].predict_proba(X_test)
)
```

7 Feature Importance using Rough sets

In [ ]:

```
dt1_scaled.head()
```

In [ ]:

```
df1=1000*dt1_scaled
df1.head()
```

7.1 segregating input features and output feature

In [ ]:

```
x_data = df1.iloc[:,:-1]
y_data = model_label
x_data,y_data
```

7.2 converting the values to integer type

In [ ]:

```
x_data = x_data.values.astype(int)
y_data = y_data.values.astype(int)
x_data,y_data
```

7.3 Using RoughSets

In [ ]:

```
selector = RoughSetsSelector()
X_selected = selector.fit(x_data, y_data).transform(x_d
ata)
```

In [ ]:

```
X_selected
```

In [ ]:

```
dt_rough = df1[['age','trestbps','chol']]
dt_rough.head()
```

7.4 Train Test Split

In [ ]:

```
model = base_models
X_train, X_test, y_train, y_test = train_test_split(dt_
rough,np.ravel(model_label),random_state=25,stratify=mo
del_label)
```

7.5 Training and Testing

In [ ]:

```
train_accuracies = {}
test_accuracies = {}
for name,model in base_models:
    model.fit(X_train, y_train)

    y_train_pred = model.predict(X_train)
    fscore_train = f1_score(y_train,y_train_pred)*100
    acc_train = model.score(X_train,y_train)*100

    acc = model.score(X_test,y_test)*100
    y_pred = model.predict(X_test)
    fscore = f1_score(y_test,y_pred)*100
    y_test_pred_values_for_models.append(y_pred)

    train_accuracies[name] = [acc_train,fscore_train]
    test_accuracies[name] = [acc,fscore]
```

8 Evaluation

8.1 Accuracies on Train Data

In [ ]:

```
names = list(train_accuracies.keys())
models_res = pd.DataFrame(data=train_accuracies.values(
),columns=["Accuracy Score","F1 Score"])
models_res.insert(0,"Model",names,True)
models_res.sort_values('F1 Score',ascending=False)
```

8.2 Accuracies on Test Data

In [ ]:

```
names = list(test_accuracies.keys())
models_res = pd.DataFrame(data=test_accuracies.values()
,columns=["Accuracy Score","F1 Score"])
models_res.insert(0,"Model",names,True)
models_res.sort_values('F1 Score',ascending=False)
```

CHAPTER – 6
EVALUATION

6.1 Evaluation of Models

While training a model is a key step, how the model generalizes on unseen data is an equally important aspect that should be considered in every machine learning pipeline. We need to know whether it actually works and, consequently, if we can trust its predictions.

The above issues can be handled by evaluating the performance of a machine learning model. Model evaluation aims to estimate the generalization accuracy of a model on future (unseen/out-of-sample) data.

In this project model evaluation is performed using f1-score and accuracy_score as well as AUC_ROC curve and precision_recall curve.

Evaluation of Model trained by using Chi-square Test is as follows,

6.1.1 Accuracy and F1-Score

Accuracies obtained on Train Data are as follows,

	Model	Accuracy Score	F1 Score
3	DT_model	100.000000	100.000000
4	RF_model	100.000000	100.000000
5	AdaBoost_model	100.000000	100.000000
2	LR_model	83.700441	85.258964
0	SVM	81.938326	83.794466
1	KNN	81.938326	83.400810

Figure – 6.1

These are accuracies obtained for different algorithms by predicting on the trained Data. Here Decision Trees, Random Forests and Adaboost Algorithms are giving an accuracy of 100%, which is not good if they give less accuracy on the test data. Our best model for this problem Logistic Regression is giving an accuracy of 85.25% on train data.

Accuracies obtained on Test Data are as follows,

	Model	Accuracy Score	F1 Score
2	LR_model	89.473684	90.476190
1	KNN	89.473684	90.243902
0	SVM	86.842105	88.095238
4	RF_model	82.894737	84.337349
3	DT_model	75.000000	77.108434
5	AdaBoost_model	75.000000	77.108434

Figure – 6.2

These are accuracies obtained for different algorithms by predicting on the test data. Logistic Regression is giving best accuracy of 90.47%, where as its training accuracy is 85% so, it is performing well on this dataset.

Decision Trees, Random Forest and AdaBoost Algorithms gave 100% accuracy on training data whereas coming to test data Decision trees accuracy is 77%, Random Forest accuracy is 84% and AdaBoost accuracy is 74% which specified that these models are overfitting on the data.

The values of Accuracies are used when the True Positives and True negatives are more important while F1-score is used when the False Negatives and False Positives are crucial.

6.1.2 Classification Report

	precision	recall	f1-score	support
Yes	0.91	0.86	0.88	35
No	0.88	0.93	0.90	41
accuracy			0.89	76
macro avg	0.90	0.89	0.89	76
weighted avg	0.90	0.89	0.89	76

Figure – 6.3

The above figure describes the classification report of the best model Logistic Regression.

Precision is a measure of finding out of all predicted true values how many are actually true or out of all predicted false values how many are actually false.

Recall is a measure of finding out of all actual true values how many are predicted true or out of all actual false values how many are predicted false.

F1-score is a weighted harmonic mean of both precision and recall so it can be used to compare 2 or more classifier models.

Support describes about number of actual occurrences of that value in the dataset. The value of support does not change between models.

So, for the Logistic Regression model the value of precision for true values is 0.91 which means out of all predicted persons with heart disease 91% of people are actually having heart disease and precision for false values is 0.88 which means out of all predicted persons without heart disease 88% of people are actually not having heart disease.

The value of recall for true values is 0.86 which means out of all persons suffering with heart disease 86% of people are predicted having heart disease and recall for false values is 0.93 which means out of all persons without heart disease 93% of people are predicted not having heart disease.

The values of support for true values are 35% and for false values is 41% across the dataset. So, the true values and false values are distributed uniformly across the dataset.

6.1.3 AUC-ROC Curve

The **Receiver Operator Characteristic (ROC)** curve is an evaluation metric for binary classification problems. It is a probability curve that plots the **True Positive Rate** against **False Positive Rate** at various threshold values. The **Area Under the Curve (AUC)** is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.

The AUC-ROC curve for Logistic Regression is as follows,

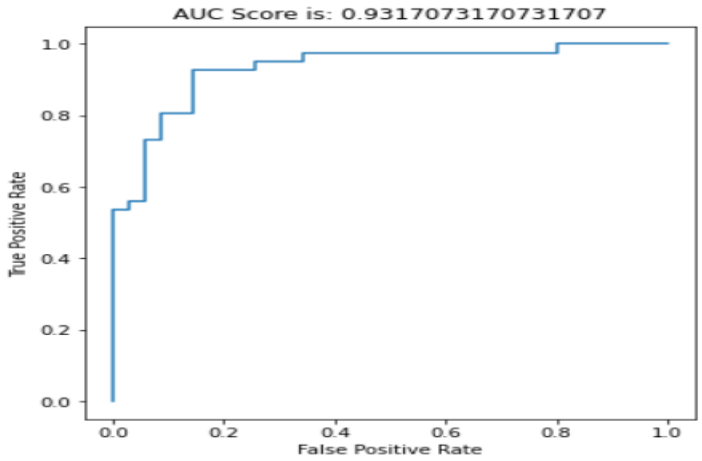

Figure – 6.4

When 0.5<AUC<1, there is a high chance that the classifier will be able to distinguish the positive class values from the negative class values. This is so because the classifier is able to detect more numbers of True positives and True negatives than False negatives and False positives.

The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.

The AUC for Logistic Regression is 0.93 so the model is performing best.

6.1.4 Precision-Recall Curve

The precision-recall curve shows the tradeoff between precision and recall for different threshold. A high area under the curve represents both high recall and high precision, where high precision relates to a low false positive rate, and high recall relates to a low false negative rate. High scores for both show that the classifier is returning accurate results (high precision), as well as returning a majority of all positive results (high recall).

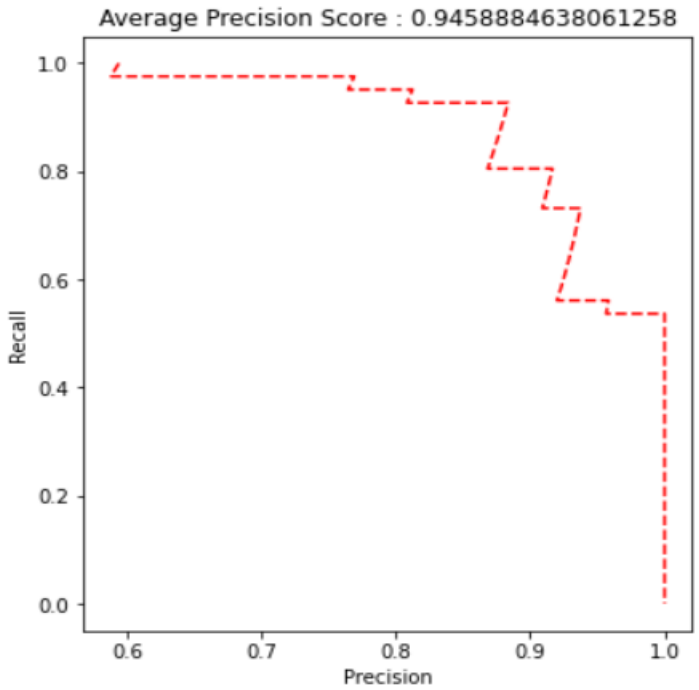

Figure – 6.5

The Average Precision Score for Logistic Regression is 0.94 which is high so this model performing better

CHAPTER - 7

CONCLUSION AND FUTURE SCOPE

CONCLUSION AND FUTURE SCOPE

In this project we have addresses the classification of heart diseases using machine learning. We have used Machine Learning UCI dataset for heart diseases as a dataset and performed different preprocessing techniques and found the important features using Chi-Square test and Roughsets Techniques then trained the data using different classic algorithms like Logistic Regression, KNN, SVM and several ensemble methods.

This finally resulted as a model which takes input values and predicts the output whether a person is suffering from heart disease or not in an efficient and effective manner.

7.1 Conclusion

From this experiment it is concluded that the Feature Importance technique Chi-square test is giving better features to predict the heart disease as compared to Roughsets. So, we considered the best features given by Chi-Square Test for increasing the accuracy and for decreasing false positives and negative rates.

Among all the Algorithms used for training, Logistic Regression is performing best by giving the accuracy of 90.47%, KNN is giving accuracy of 90.24%, SVM is giving an accuracy of 88%, Random Forest is giving an accuracy of 84.33%. The ensemble methods used in this project does not perform best compared to the classic algorithms.

7.2 Future Scope

The primary motive of this research is the prediction of heart diseases with high rate of accuracy. One of the major drawbacks of many works is that the main focus has been on the application of classification techniques for heart disease prediction, rather than studying various data cleaning, pruning and normalization techniques that prepare and make a dataset suitable for mining. It has been observed that a properly cleaned, pruned and normalized dataset provides much better accuracy than an unclean one with missing values. Selection of suitable techniques for data cleaning along with proper classification algorithms will lead to the development of prediction systems that give enhanced accuracy. In future an intelligent system may be developed that can lead to selection of proper treatment methods for a patient diagnosed with heart disease. A lot of work has

been done already in making models that can predict whether a patient is likely to develop heart disease or not. There are several treatments methods for a patient once diagnosed with a particular form of heart disease.

Machine learning can be of very good help in deciding the line of treatment to be followed by extracting knowledge from such suitable databases. The future scope of the paper is the prediction of heart diseases by using advanced techniques and algorithms in less time complexity and with high accuracy.

REFERENCES

[1] Youness Khourdifi and Mohamed Bahaj "**Heart Disease Prediction and Classification Using Machine Learning Algorithms Optimized by Particle Swarm Optimization and Ant Colony Optimization**", *Laboratory of Innovation for New Energy Technologies and Nanomaterials (LITEN),* October 22, 2018

[2] G. Parthiban and S.K. Srivatsa "**Applying Machine Learning Methods in Diagnosing Heart Disease for Diabetic Patients**", *International Journal of Applied Information Systems (IJAIS) – ISSN, August 2012*

[3] J. P. Li, A. U. Haq, S. U. Din, J. Khan, A. Khan and A. Saboor, **"Heart Disease Identification Method Using Machine Learning Classification in E-Healthcare,"** in IEEE Access 2020.

[4] J. Vijayashree and H. Parveen Sultana "**A Machine Learning Framework for Feature Selection in Heart Disease Classification Using Improved Particle Swarm Optimization with Support Vector Machine Classifier**", *School of Computer Science and Engineering, Vellore Institute of Technology,* July 9, 2018

[5] M. Akhil Jabbar, B. L. Deekshatulu and Priti Chandra "**HEART DISEASE CLASSIFICATION USING NEAREST NEIGHBOR CLASSIFIER WITH FEATURE SUBSET SELECTION**", Aurora's Engineering College, A.P, India, 2013

[6] Sinkon Naya, Mahendra Kumar Gourisari, Manjusha Pandey, Siddharth Swarup **Rautaray "Prediction of Heart Disease by Mining Frequent Items and Classification Technique"**, International Conference on Intelligent Computing and Control Systems (ICICCS 2019)

[7] M. Chen, Y. Hao, K. Hwang, L. Wang and L. Wang, **"Disease Prediction by Machine Learning Over Big Data from Healthcare Communities,"** in IEEE Access, vol. 5, 2017.

[8] S. S. Raoof, M. A. Jabbar and S. A. Fathima, **"Lung Cancer Prediction using Machine Learning: A Comprehensive Approach,"** 2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), 2020

[9] Akib Mohi Ud Din Khanday, Syed Tanzeel Rabani, Qamar Rayees Khan, Nusrat Rouf and Masarat Mohi Ud Din "**Machine learning based approaches for detecting COVID-19 using clinical text data**", Bharati Vidyapeeth's Institute of Computer Applications and Management 2020, September 2020.

[10] Julie L. Harvey and Sathish A.P. Kumar "**Machine Learning for Predicting Development of Asthma in Children**", 2019 IEEE Symposium Series on Computational Intelligence (SSCI), December 6-9 2019