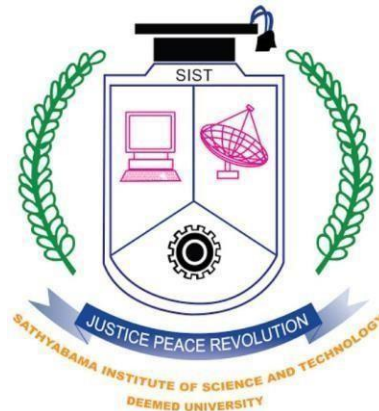


ESTIMATION OF CHILD MORTALITY

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

by

KOTHA MANIKANTA (Reg. No. 37110372)
KOVVURI RAMAKRISHNA (Reg. No. 37110380)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

March - 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **KOTHA MANIKANTA (Reg.no. 37110372)** and **KOVVURI RAMAKRISHNA REDDY (Reg.no. 37110380)** who carried out the project entitled "**ESTIMATION OF CHILD MORTALITY**" under my supervision from **August 2019** to **March 2020**.

Internal Guide
Dr. P. ASHA, M.E., Ph.D.,

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

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **KOTHA MANIKANTA** and **KOVVURI RAMAKRISHNA REDDY** hereby declare that the Project Report entitled “**ESTIMATION OF CHILD MORTALITY**” is done by me under the guidance of **Dr. P. ASHA, M.E.,Ph.D.**, Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

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

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

I would like to express my sincere and deep sense of gratitude to my Project Guide **Mrs. P ASHA, M.E., Assistant Professor**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

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

ABSTRACT

Children Mortality refers to mortality of children under the age of five. The child mortality rate, also under-five mortality rate, refers to the probability of dying between birth and exactly five years of age. The mortality of children also happens in fetus. The aim is to investigate machine learning based techniques for classification of mortality fetal health classification results in best accuracy. The analysis of dataset by supervised machine learning technique(SMLT) to capture several information's like, variable identification, uni-variate analysis, bi-variate and multi-variate analysis, missing value treatments and analyze the data validation, data cleaning/preparing and data visualization will be done on the entire given dataset. Our analysis provides a comprehensive guide to sensitivity analysis of model parameters with regard to performance in the classification of fetal health. To propose a machine learning-based and additionally, to compare and discuss the performance of various machine learning algorithms for the given dataset.

TABLE OF CONTENTS

ABSTRACT

v

LIST OF FIGURES

vii

CHAPTER No.	TITLE	PAGE No.
1.	INTRODUCTION	1
1.1	INTRODUCTION	1
1.2	DOMAIN OVERVIEW	2
2.	AIM AND SCOPE	4
2.1	AIM OF PROJECT	4
2.1.1	<i>OBJECTIVES</i>	4
2.2	SCOPES	4
2.2.1	<i>ADVANTAGES</i>	4
3.	SYSTEM DESIGN & METHODOLOGY	5
3.1	EXISTING SYSTEM	5
3.2	PROPOSED SYSTEM	5
3.3	DATA SPLITTING	5
3.3.1	SYSTEM STUDY	6
3.3.2	PROJECT REQUIREMENTS	7
3.4	SOFTWARE DISCRIPTION	8
3.5	MODULES	12
4.	SOFTWARE DEVELOPMENT METHODOLOGY	16

4.1	WORKFLOW DIAGRAM	16
4.2	ARCHITECTURAL DESIGN	17
4.3	ALGORITHM AND TECHNIQUES	19
4.4	USE CASE DIAGRAM	22
5.	RESULTS AND DISCUSSION	24
5.1	Classification of logistic regression	26
5.2	Classification of decision tree	27
5.3	Classification of random forest	27
5.4	Classification of support vector machine	28
5.5	Accuracy Comparision	28
6.	CONCLUSION AND FUTURE WORK	29
	REFERENCES	30

LIST OF FIGURES

FIGURE No.	FIGURE NAME	PAGE No.
4.1	WORKFLOW DIAGRAM	16
4.2	ARCHITECTURAL DESIGN	17
4.3	USECASE DIAGRAM	22

CHAPTER-1

INTRODUCTION

1.1 INTRODUCTION

Machine learning is to predict the future from past data. Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data and the basics of Machine Learning, implementation of a simple machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms. It feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. Machine learning can be roughly separated in to three categories. There are supervised learning, unsupervised learning and reinforcement learning. Supervised learning program is both given the input data and the corresponding labeling to learn data has to be labeled by a human being beforehand. Unsupervised learning is no labels. It provided to the learning algorithm. This algorithm has to figure out the clustering of the input data. Finally, Reinforcement learning dynamically interacts with its environment and it receives positive or negative feedback to improve its performance.

Data scientists use many different kinds of machine learning algorithms to discover patterns in python that lead to actionable insights. At a high level, these different algorithms can be classified into two groups based on the way they “learn” about data to make predictions: supervised and unsupervised learning. Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function from input variables(X) to discrete output variables(y). In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document.

Under 5 mortality rates: number of child deaths within 5th birthday/ total number of birth

The leading causes of death of children under five include: classification etc.

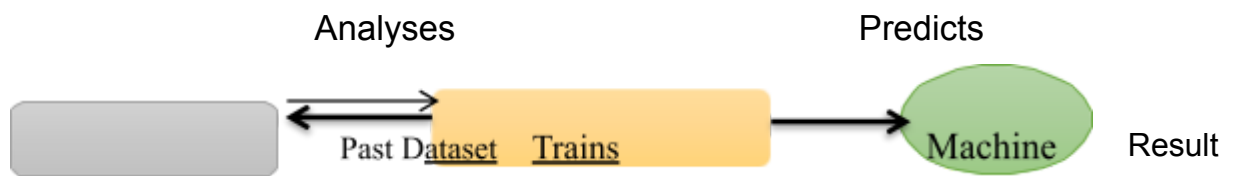


Fig: Process of Machine learning

1.2 DOMAIN OVERVIEW

Supervised Machine Learning is the majority of practical machine learning uses supervised learning. Supervised learning is where we have input variables (X) and an output variable (y) and use an algorithm to learn the mapping function from the input to the output is $y = f(X)$. The goal is to approximate the mapping function so well that when you have new input data (X) that you can predict the output variables (y) for that data. Techniques of Supervised Machine Learning algorithms include logistic regression, multiclass classification, Decision Trees and support vector machines etc. Supervised learning requires that the data used to train the algorithm is already labeled with correct answers. Supervised learning problems can be further grouped into Classification problems. This problem has as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for regression and categorical for classification. A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. A classification problem is when the output variable is a category, such as Yes or No.

Child mortality refers to number of child deaths under the age of 5 per 1000 live births. However, the child mortality could be simplified into more specific terms such as prenatal, perinatal, Neonatal, infancy and under 5. Prenatal: child death before the birth, Perinatal: child death, Neonatal: child death before 28 days of birth, Infancy: child death before 1st birthday, and child mortality under 5 refer to any deaths from birth to the 5th birthday.

Perinatal mortality rate: Number of child deaths within first week of birth/ total number of birth

Neonatal mortality rate: number of child deaths within first 28 days of life/ total number of birth

Infancy mortality rate: number of child deaths within first 12 months of life/ total number of birth

- Preterm birth complications (18%)
- Pneumonia (16%)
- Interpartum-related events (12%)
- Neonatal **sepsis** (7%)
- Diarrhea (8%)
- Malaria (5%)
- Malnutrition and Under nutrition

CHAPTER -2

AIM AND SCOPE

2.1 AIM

The scope of this project is to investigate a dataset of child mortality which is fetal classification using machine learning technique. To identify the what are the major complications which is leading to these classification

2.1.1 OBJECTIVES

The goal is to develop a machine learning model for predicting the child mortality by building a machine learning classification models by predicting results in the form of best accuracy by comparing supervised algorithm

Problem Description/ Problem Statements

Child mortality is not only caused by infection and disorder: it is also caused by premature birth; birth defect; new born infection; birth complication; and diseases like malaria, sepsis, and diarrhea. In less developed countries, malnutrition is the main cause of child mortality

2.2 SCOPE

The scope of this project is to investigate a dataset of child mortality which is fetal classification using machine learning technique. To identify the what are the major complications which is leading to these classification

2.2.1 ADVANTAGES

These reports are to the investigation of applicability of machine learning techniques for classification of the mortality.

Finally, it highlights some observations on future research issues, challenges, and needs.

Chapter – 3

SYSTEM DESIGN AND METHODOLOGY

3.1 EXISTING SYSTEM

World Health Organization (WHO) has shown numerous report regarding child mortality and these are supported by many kinds of literature.

They had utilized data mining technique using decision tree called J48 algorithm in classifying child mortality rate, life expectancy at birth, annual population growth, and the gross domestic product.

Drawbacks:

They had only focused on one algorithm and no comparison is done check for better algorithm.

Model can be further enhance by applying EDA meth

3.2 PROPOSED SYSTEM

Data will be loaded using pandas check for cleanliness, and then trim and clean given dataset for analysis. The data which was collected might contain missing values that may lead to inconsistency. To gain better results data need to be preprocessed so as to improve the efficiency of the algorithm. The outliers have to be removed and also variable conversion need to be done.

3.3 Data splitting and building classification model

The data set collected for predicting given data is split into Training set and Test set. Generally, 7:3 ratios are applied to split the Training set and Test set. After the splitting is done we apply supervised Machine Learning algorithm like logistic regression, decision tree, Random forest and support vector machines and compare the algorithms by their accuracy

3.3.1 SYSTEM STUDY

Overview of the system

It have to find Accuracy of the training dataset, Accuracy of the testing dataset, Specification, False Positive rate, precision and recall by comparing algorithm using python code. The following Involvement steps are,

Define a problem

Preparing data

Evaluating algorithms

Improving results

Predicting results

Project Goals

Exploration data analysis of variable identification

- Loading the given dataset
- Import required libraries packages
- Analyze the general properties
- Find duplicate and missing values
- Checking unique and count values

Uni-variate data analysis

- Rename, add data and drop the data
- To specify data type

Exploration data analysis of bi-variate and multi-variate

- Plot diagram of pair plot, heat map, bar chart and Histogram

Method of Outlier detection with feature engineering

- Pre-processing the given dataset
- Splitting the test and training dataset
- Comparing the Decision tree and Logistic regression model and random

forest etc.

Comparing algorithm to predict the result

- Based on the best accuracy

3.3.2 Project Requirements General:

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

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

requirements:

The software requirements specification is a technical specification of requirements for the software product. It is the first step in the requirements analysis process. It lists requirements of a particular software system. The following details to follow the special libraries like sk-learn, pandas, numpy, matplotlib and seaborn.

Non-Functional Requirements:

Process of functional steps,

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

Environmental Requirements:

1. Software Requirements: Operating System :
Windows Tool : Anaconda with Jupyter Notebook
2. Hardware requirements:
Processor : Pentium IV/III

Hard disk : minimum 80 RAM

RAM : minimum 2 GB

3.4 SOFTWARE DESCRIPTION

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

Anaconda Navigator

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, macOS and Linux.

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook
- QtConsole
- Spyder
- Glueviz
- Orange
- Rstudio
- Visual Studio Code

Conda:

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

The Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

Notebook document:

Notebook documents (or “notebooks”, all lower case) are documents produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc...). Notebook documents are both human-readable documents containing the analysis

description and the results (figures, tables, etc.) as well as executable documents which can be run to perform data analysis.

Jupyter Notebook App:

The Jupyter Notebook App is a server-client application that allows editing and running notebook documents via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet. In addition to displaying/editing/running notebook documents, the Jupyter Notebook App has a “Dashboard” (Notebook Dashboard), a “control panel” showing local files and allowing to open notebook documents or shutting down their kernels.

kernel:

A notebook kernel is a “computational engine” that executes the code contained in a Notebook document. The ipython kernel, referenced in this guide, executes python code. Kernels for many other languages exist (official kernels). When you open a Notebook document, the associated kernel is automatically launched. When the notebook is executed (either cell-by-cell or with menu Cell -> Run All), the kernel performs the computation and produces the results. Depending on the type of computations, the kernel may consume significant CPU and RAM. Note that the RAM is not released until the kernel is shut-down.

Notebook Dashboard:

The Notebook Dashboard is the component which is shown first when you launch Jupyter Notebook App. The Notebook Dashboard is mainly used to open notebook documents, and to manage the running kernels (visualize and shutdown). The Notebook Dashboard has other features similar to a file manager, namely navigating folders and renaming/deleting files.

Working Process:

Download and install anaconda and get the most useful package for machine learning in Python.

Load a dataset and understand its structure using statistical summaries and data visualization.

machine learning models, pick the best and build confidence that the accuracy is reliable.

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

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

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

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

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

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

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

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

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

3.5 Modules:

Variable Identification Process.

Exploration data analysis of visualization

Comparing of Logistic Regression and Decision Tree Algorithm

Comparing of Random Forest and Support Vector Machines Algorithm.

Module description:

Variable Identification Process

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

The evaluation becomes more biased as skill on the validation dataset is

incorporated into the model configuration. The validation set is used to evaluate a given model, but this is for frequent evaluation. It as machine learning engineers

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

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

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

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

- import libraries for access and functional purpose and read the given dataset

- General Properties of Analyzing the given dataset

- Display the given dataset in the form of data frame

- show columns

- shape of the data frame

- To describe the data frame

- Checking data type and information about dataset

- Checking for duplicate data

- Checking Missing values of data frame

- Checking unique values of data frame

- Checking count values of data frame

Rename and drop the given data frame

To specify the type of values

To create extra columns

Exploration data analysis of visualization

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

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

How to chart time series data with line plots and categorical quantities with bar charts.

How to summarize data distributions with histograms and box plots.

Comparison of Logistic Regression and Decision Tree Algorithm

In this module the data is split in train and test data using `train_test_split()` function in sklearn package. Then the algorithm is applied on the training data to

build the classification model. The training is done by fit function and the testing is done by using predict function provided by the algorithm and the accuracy is compared

Comparison of Random Forest and Support Vector Machine Algorithm

Random forest is a collection of decision tree it is considered as CART which means Classification and Regression technique. The Randomclassifier package is used to classify the output the fit function is to train the model and predict function is used to test the model. In this module the accuracy comparison is done between support vector machine and Random forest. Support Vector Classifier is used to classify the model

Parameters in the Confusion Matrix

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

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

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

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

CHAPTER - 4

SOFTWARE DEVELOPMENT AND METHODOLOGY

4.1 Work flow diagram

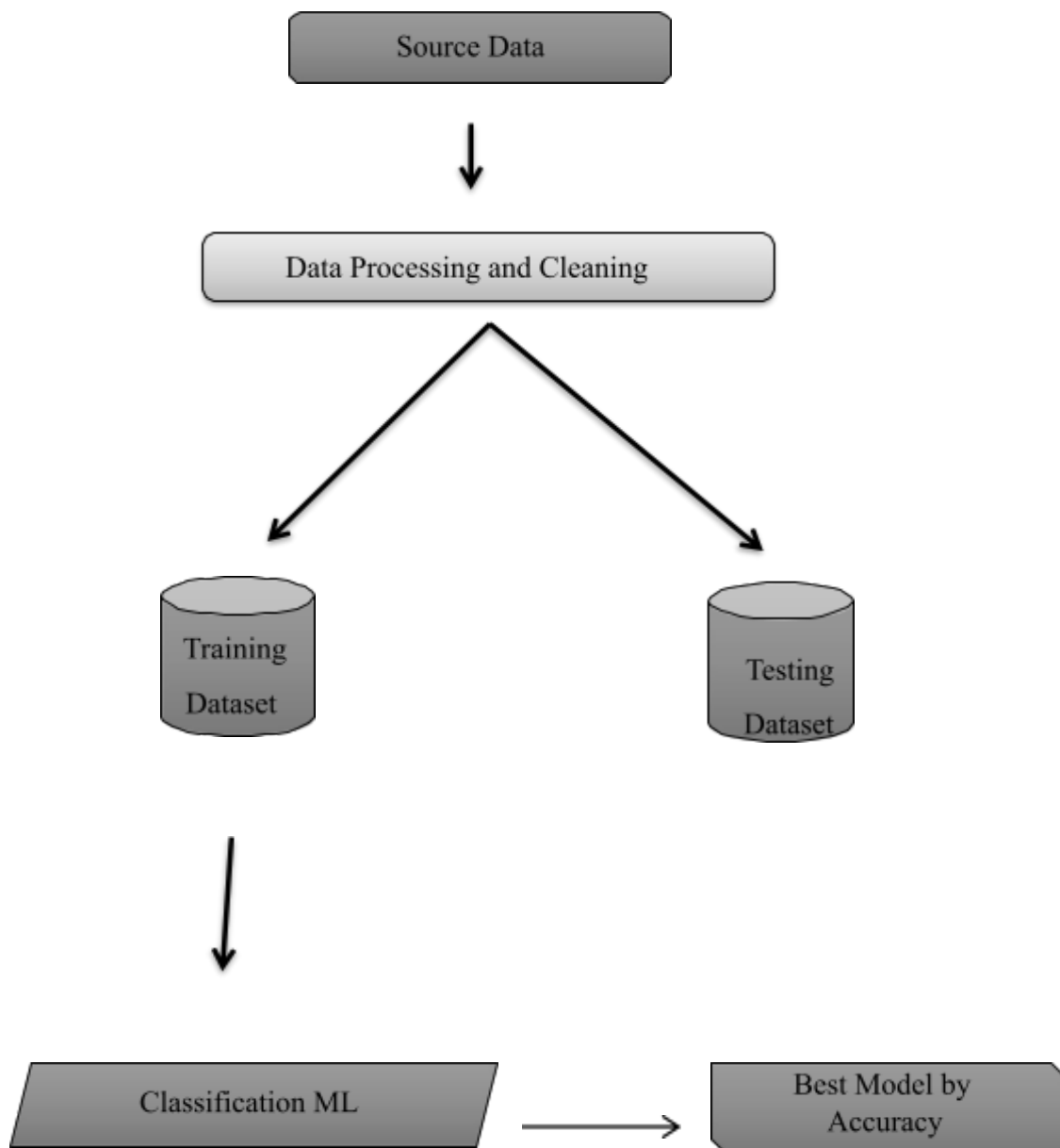
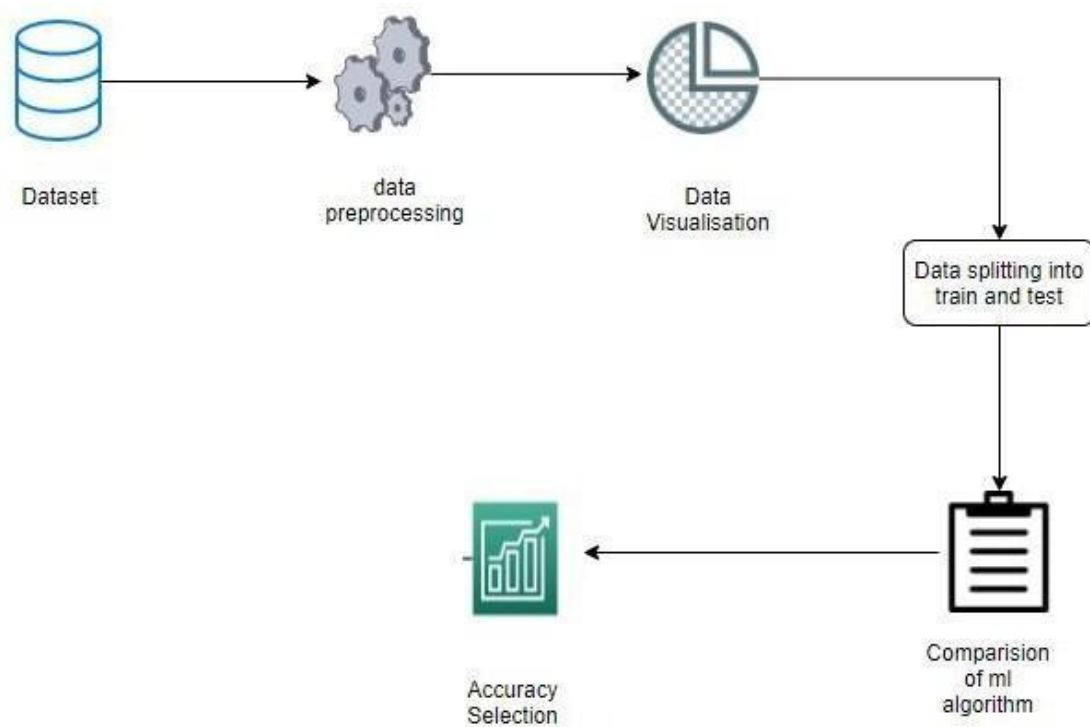


Fig: Workflow Diagram

4.2 Architectural Design



Comparing Algorithm with prediction in the form of best accuracy result

It is important to compare the performance of multiple different machine learning algorithms consistently and it will discover to create a test harness to compare multiple different machine learning algorithms in Python with scikit-learn. It can use this test harness as a template on your own machine learning problems and add more and different algorithms to compare. Each model will have different performance characteristics. Using resampling methods like cross validation, you can get an estimate for how accurate each model may be on unseen data. It needs to be able to use these estimates to choose one or two best models from the suite of models that you have created. When have a

new dataset, it is a good idea to visualize the data using different techniques in order to look at the data from

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

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

Prediction result by accuracy:

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

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

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

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

Accuracy calculation:

Accuracy = $(TP + TN) / (TP + TN + FP + FN)$

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

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

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

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

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

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

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

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

General Formula:

$$\text{F-Measure} = 2\text{TP} / (2\text{TP} + \text{FP} + \text{FN}) \quad \textbf{F1-Score}$$

Formula:

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

4.3 ALGORITHM AND TECHNIQUES Algorithm Explanation

In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be biclass (like identifying whether the person is male or female or that the mail is

spam or non-spam) or it may be multi-class too. Some examples of classification

problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc. In Supervised Learning, algorithms learn from labeled data. After understanding the data, the algorithm determines which label should be given to new data based on pattern and associating the patterns to the unlabeled new data.

Used Python Packages:

sklearn:

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

NumPy:

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

Pandas:

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

Matplotlib:

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

Logistic Regression

It is a statistical method for analysing a data set in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). The goal of logistic regression is to find the best fitting model to

describe the relationship between the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or

explanatory) variables. Logistic regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.).

In other words, the logistic regression model predicts $P(Y=1)$ as a function of X . Logistic regression Assumptions:

Binary logistic regression requires the dependent variable to be binary.

For a binary regression, the factor level 1 of the dependent variable should represent the desired outcome.

Only the meaningful variables should be included.

The independent variables should be independent of each other. That is, the model should have little.

The independent variables are linearly related to the log odds.

Logistic regression requires quite large sample sizes.

Support Vector machine

The objective of the support vector machine algorithm is to find a hyperplane in an N -dimensional space (N — the 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. Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the

dimension of the hyperplane depends upon the number of features. If the number of input features

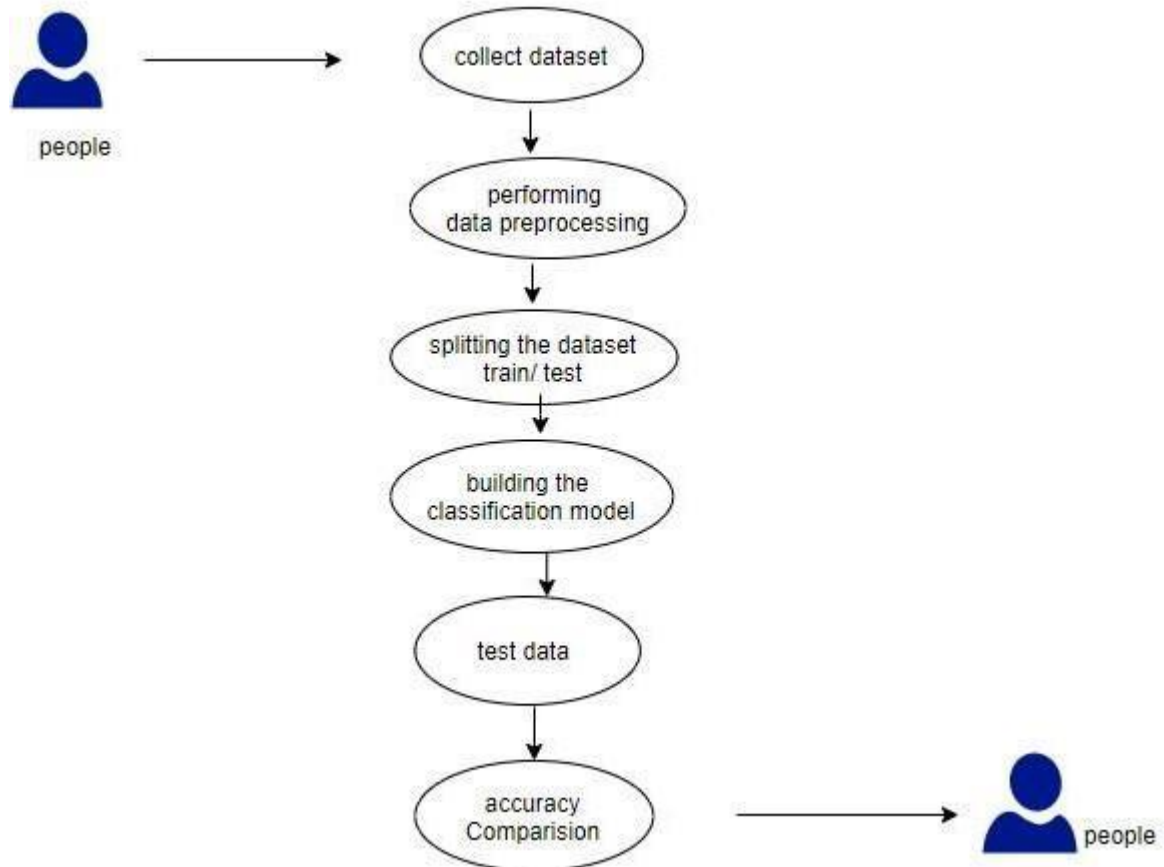
is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3. Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM.

Random Forest

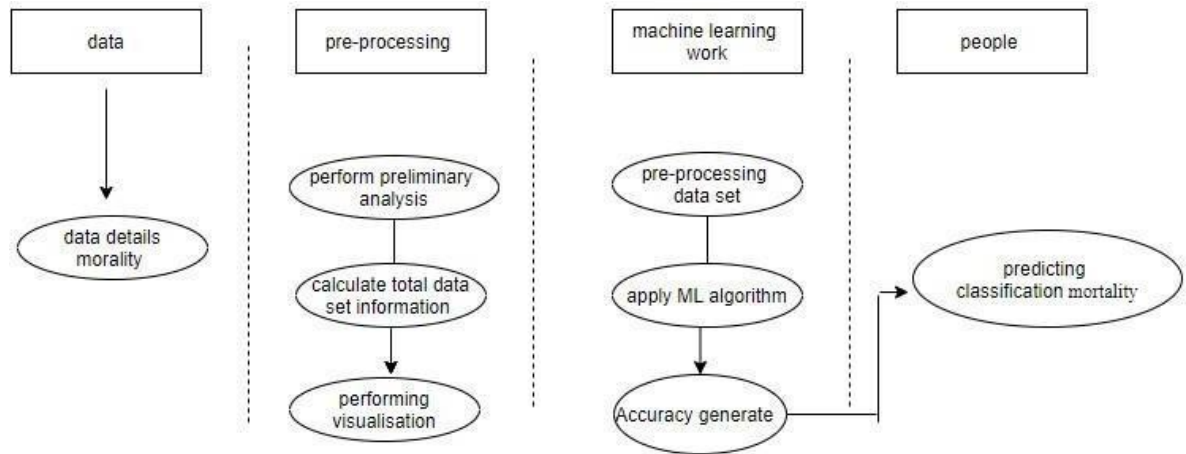
Random forest consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction. A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models. The reason for this wonderful effect is that the trees protect each other from their individual errors (as long as they don't constantly all err in the same direction). While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction. So the prerequisites for random forest to perform well are: There needs to be some actual signal in our features so that models built using those features do better than random guessing.

The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.

4.4 Use Case Diagram



Activity Diagram:



CHAPTER - 5

RESULT AND DISCUSSION

```
In [4]: data.head()
```

```
Out[4]:
```

	baseline value	accelerations	fetal_movement	uterine_contractions	light_decelerations	severe_decelerations	prolongued_decelerations	abnormal_short_term_variat
0	120	0.000	0.0	0.000	0.000	0.0	0.0	
1	132	0.006	0.0	0.006	0.003	0.0	0.0	
2	133	0.003	0.0	0.008	0.003	0.0	0.0	
3	134	0.003	0.0	0.008	0.003	0.0	0.0	
4	132	0.007	0.0	0.008	0.000	0.0	0.0	

5 rows × 22 columns

```
In [5]: #shape
data.shape
```

```
Out[5]: (2126, 22)
```

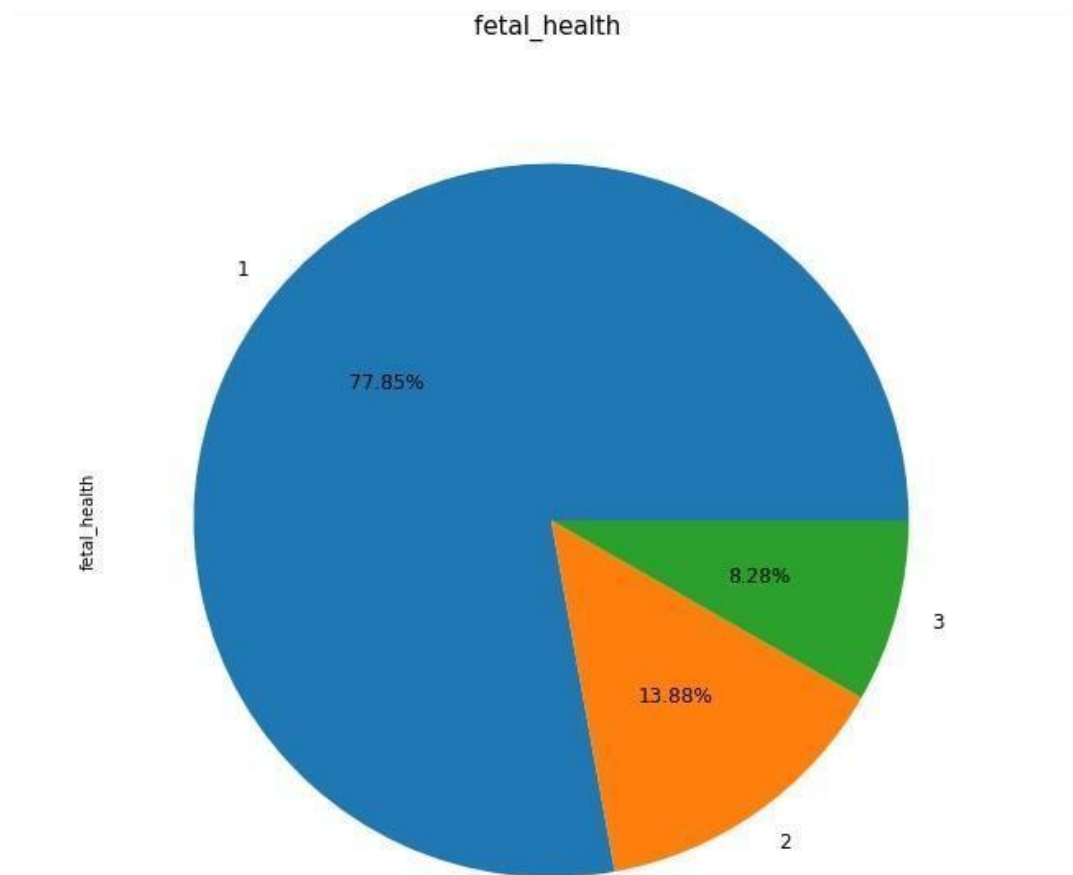
Fig: Data show and checking the size of the data

```
In [11]: #Checking datatype and information about dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2126 entries, 0 to 2125
Data columns (total 22 columns):
baseline value                2126 non-null int64
accelerations                 2126 non-null float64
fetal_movement                2126 non-null float64
uterine_contractions          2126 non-null float64
light_decelerations           2126 non-null float64
severe_decelerations          2126 non-null float64
prolongued_decelerations      2126 non-null float64
abnormal_short_term_variability 2126 non-null int64
mean_value_of_short_term_variability 2126 non-null float64
percentage_of_time_with_abnormal_long_term_variability 2126 non-null int64
mean_value_of_long_term_variability 2126 non-null float64
histogram_width               2126 non-null int64
histogram_min                 2126 non-null int64
histogram_max                 2126 non-null int64
histogram_number_of_peaks     2126 non-null int64
histogram_number_of_zeroes    2126 non-null int64
histogram_mode                2126 non-null int64
histogram_mean                2126 non-null int64
histogram_median              2126 non-null int64
histogram_variance            2126 non-null int64
histogram_tendency            2126 non-null int64
fetal_health                  2126 non-null int64
dtypes: float64(8), int64(14)
memory usage: 382.0 KB
```

Checking duplicate values of dataframe

Dataset Information



5.1 CLASSIFICATION REPORT OF LOGISTIC REGRESSION

```
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
print("")
```

Classification report of Logistic Regression Results:

	precision	recall	f1-score	support
0	0.94	0.97	0.95	497
1	0.73	0.67	0.70	88
2	0.89	0.77	0.83	53
accuracy			0.91	638
macro avg	0.85	0.80	0.83	638
weighted avg	0.91	0.91	0.91	638

Accuracy result of Logistic Regression is: 90.9090909090909

Confusion Matrix result of Logistic Regression is:

```
[[480 14  3]
 [ 27 59  2]
 [  4  8 41]]
```

Sensitivity : 0.97165991902834

Specificity : 0.686046511627907

5.2 CLASSIFICATION REPORT OF DECISION TREE

```
print('Specificity : ', specificity1)
```

Classification report of Decision Tree Classifier Results:

	precision	recall	f1-score	support
0	0.94	0.95	0.94	497
1	0.73	0.65	0.69	88
2	0.88	0.96	0.92	53
accuracy			0.91	638
macro avg	0.85	0.85	0.85	638
weighted avg	0.91	0.91	0.91	638

Accuracy result of Decision Tree Classifier is 90.9090909090909

Confusion Matrix result of Decision Tree Classifier is:

```
[[472 21  4]
 [ 28 57  3]
 [  2  0 51]]
```

5.3 CLASSIFICATION REPORT OF RANDOM FOREST

```
print("")
specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
print("")
```

Classification report of Random Forest Results:

	precision	recall	f1-score	support
0	0.94	0.99	0.96	497
1	0.88	0.65	0.75	88
2	0.94	0.89	0.91	53
accuracy			0.93	638
macro avg	0.92	0.84	0.87	638
weighted avg	0.93	0.93	0.93	638

Accuracy result of Random Forest is: 93.2601880877743

Confusion Matrix result of Random Forest is:

```
[[491  4  2]
 [ 30 57  1]
 [  2  4 47]]
```

Sensitivity : 0.9919191919191919

Specificity : 0.6551724137931034

5.4 CLASSIFICATION REPORT OF SUPPORT VECTOR MACHINE

```
sensitivity1 = cm2[0,0]/(cm2[0,0]+cm2[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm2[1,1]/(cm2[1,0]+cm2[1,1])
print('Specificity : ', specificity1)
print("")
```

Classification report of Support Vector Machines Results:

	precision	recall	f1-score	support
0	0.79	1.00	0.88	497
1	1.00	0.02	0.04	88
2	1.00	0.09	0.17	53
accuracy			0.79	638
macro avg	0.93	0.37	0.37	638
weighted avg	0.83	0.79	0.71	638

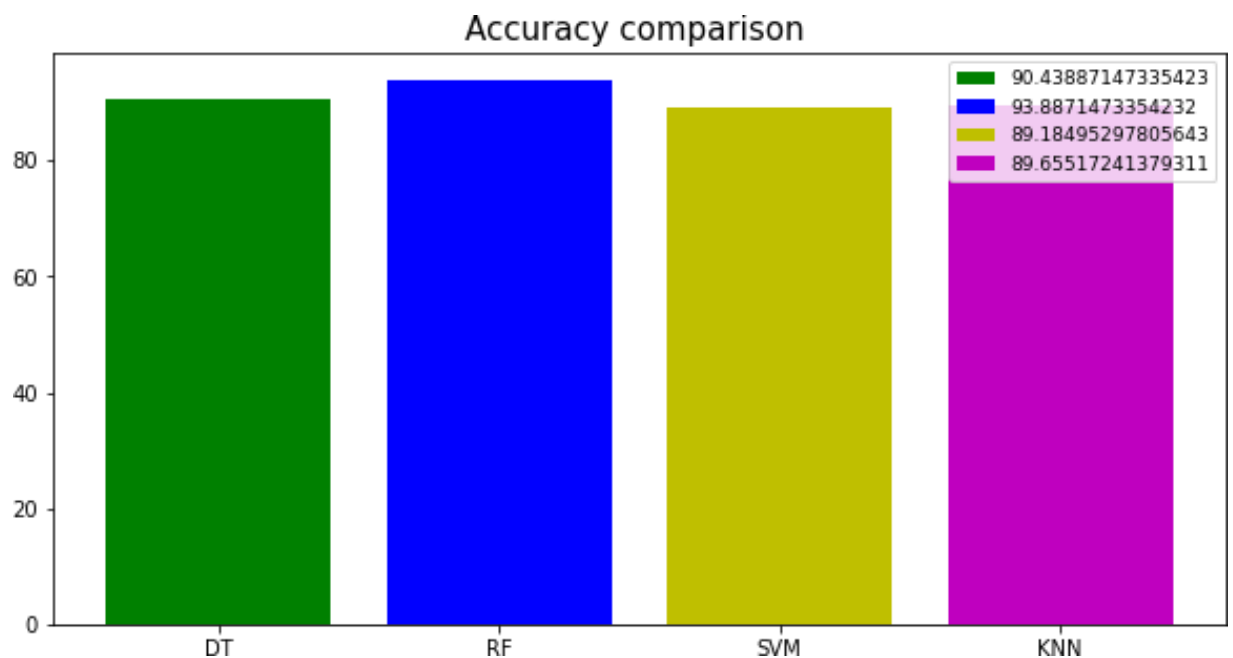
Accuracy result of Support Vector Machines is: 78.99686520376176

Confusion Matrix result of Support Vector Machines is:

```
[[497  0  0]
 [ 86  2  0]
 [ 48  0  5]]
```

Sensitivity : 1.0

5.5 ACCRACY COMPARISION



CHAPTER – 6

CONCLUSION AND FUTURE WORK

CONCLUSION

The analytical process started from data cleaning and processing, missing value, exploratory analysis and finally model building and evaluation. The best accuracy on public test set is higher accuracy score is will be find out. This application can help in finding the mortality case status and depends on that they can take action.

FUTURE WORK

To automate this process by show the prediction result in web application or desktop application.

To optimize the work to implement in Artificial Intelligence environment

REFERENCES

- [1] Ambili K, Afsar P, "A Prediction Model for Child Development Analysis using Naive Bayes and Decision Tree Fusion Technique – NB Tree" International Research Journal of Engineering and Technology (IRJET) July 2016 .
- [2] Taiwan ROC "A study of applying data mining to early intervention for developmentally-delayed children".
- [3] Fadzli Syed Abdullah¹, Nor Saidah Abd Manan¹, Aryati Ahmad¹, Sharifah Wajihah Wafa¹, Mohd Razif Shahril¹, Nurzaime Zulaily¹, Rahmah Mohd Amin¹, and Amran Ahmed² "Data Mining Techniques for Classification of Childhood Obesity among Year 6 School Children" University Sultan Zainal Abidin, 21300 Kuala Terengganu.
- [4] Anand Bahety "Extension and Evaluation of ID3 – Decision Tree Algorithm".
- [5] Deepti Sisodia and Dilip Singh Sisodia, "Prediction of Diabetes using Classification Algorithms."
- [6] Likai Wang , Changchun Shi , "The Research and Development of Growth Curve for Children's Height and Weight on Android Platform." 10th International Congress on Image and Signal Processing, Biomedical Engineering and Informatics (CISP-BMEI 2017).
- [7] Ashvini Kale, Nisha Abhijeet Auti "Automated Menu Planning Algorithm for Children: Food Recommendation by Dietary Management System using ID3 for Indian Food Database."
- [8] Masud Karim and Rashedur M. Rahman , "Decision Tree and Naïve Bayes Algorithm for Classification and Generation of Actionable Knowledge for Direct Marketing" Journal of Software Engineering and Applications, 2013, 6, 196-206.
- [19] Shaoyan Zhang, Christos Tjortjis, Xiaojun Zeng, Hong Qiao , Iain Buchan and John Keane "Comparing data mining methods with logistic regression in childhood obesity prediction" Published online: 24 February 2009 # Springer Science + Business Media, LLC 2009.

CODE

```
import pandas as p import numpy as n import pandas as p
import numpy as n = p.read_csv("child.csv") import warnings
warnings.filterwarnings('ignore') data.head() data.shape
df=data.dropna() df.head() df.shape df.columns df.describe()
df.info() df.duplicated() sum(df.duplicated()) df.isnull().sum()
df.accelerations.unique() df.fetal_movement.unique()
df.uterine_contractions.unique()
df.light_decelerations.unique()
df.severe_decelerations.unique()
df.abnormal_short_term_variability.unique() print("Minimum
value of fetal_health:", df.fetal_health.min()) print("Maximum
value of fetal_health:", df.fetal_health.max())
print("fetal_health ranges :",
sorted(df['fetal_health'].unique()))
p.Categorical(df['accelerations']).describe()
p.Categorical(df['fetal_movement']).describe(
) df['uterine_contractions'].value_counts()
df.corr() df.head() df.columns()

from sklearn.preprocessing import LabelEncoder var_mod =
['baseline value', 'accelerations', 'fetal_movement',
'uterine_contractions', 'light_decelerations', 'severe_decelerations',
    'prolongued_decelerations', 'abnormal_short_term_variability',
    'mean_value_of_short_term_variability',
    'percentage_of_time_with_abnormal_long_term_variability',
    'mean_value_of_long_term_variability', 'histogram_width',
    'histogram_min', 'histogram_max', 'histogram_number_of_peaks',
    'histogram_number_of_zeroes', 'histogram_mode', 'histogram_mean',
    'histogram_median', 'histogram_variance', 'histogram_tendency',
    'fetal_health']
le = LabelEncoder()
```

```
for i in var_mod:
```

```
df[i] = le.fit_transform(df[i]).astype(str)
df.head()
```

```
import pandas as p import
matplotlib.pyplot as plt import
seaborn as s import numpy as
n data =
p.read_csv("child.csv") import
warnings
warnings.filterwarnings('ignore'
) df=data.dropna() df.columns()
p.crosstab(df.accelerations,df.fetal_movement)
```

```
df['fetal_movement'].hist(figsize=(7,6), color='green',
alpha=0.7) plt.xlabel('fetal_movement') plt.ylabel('No of
fetal_movement') plt.title('fetal_movement of child mortality')
def PropByVar(df, variable):
    dataframe_pie = df[variable].value_counts() ax =
dataframe_pie.plot.pie(figsize=(10,10), autopct='%1.2f%%', fontsize = 12)
ax.set_title(variable + ' \n', fontsize = 15) return
n.round(dataframe_pie/df.shape[0]*100,2) PropByVar(df, 'fetal_movement') def
PropByVar(df, variable):
    dataframe_pie = df[variable].value_counts() ax =
dataframe_pie.plot.pie(figsize=(10,10), autopct='%1.2f%%', fontsize =
12) ax.set_title(variable + ' \n', fontsize = 15) return
n.round(dataframe_pie/df.shape[0]*100,2)
```

```
PropByVar(df,
'uterine_contractions')
PropByVar(df, 'fetal_health') fig, ax
= plt.subplots(figsize=(15,10))
s.heatmap(df.corr(), ax=ax, annot=True) a = df["accelerations"] fm =
```

```
df["fetal_movement"] uc = df["uterine_contractions"] ld =  
df["light_decelerations"] sd = df["severe_decelerations"] pd =
```

```

df["prolongued_decelerations"] ast =
df["abnormal_short_term_variability"] mv =
df["mean_value_of_short_term_variability"] from
sklearn.preprocessing import LabelEncoder var_mod = ['baseline
value', 'accelerations', 'fetal_movement',      'uterine_contractions',
'light_decelerations', 'severe_decelerations',
    'prolongued_decelerations', 'abnormal_short_term_variability',
    'mean_value_of_short_term_variability',
    'percentage_of_time_with_abnormal_long_term_variability',
    'mean_value_of_long_term_variability', 'histogram_width',
    'histogram_min', 'histogram_max', 'histogram_number_of_peaks',
    'histogram_number_of_zeroes', 'histogram_mode', 'histogram_mean',
    'histogram_median', 'histogram_variance', 'histogram_tendency',
    'fetal_health']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i]).astype(str)
df.head()
X = df.drop(labels='fetal_health', axis=1)
#Response variable
y = df.loc[:, 'fetal_health']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1,
stratify=y) print("Number of training dataset: ", len(X_train)) print("Number of test
dataset: ", len(X_test)) print("Total number of dataset: ", len(X_train)+len(X_test)) def
qul_No_qul_bar_plot(df, bygroup):
    dataframe_by_Group = p.crosstab(df[bygroup], columns=df["fetal_health"], normalize
= 'index')    dataframe_by_Group = n.round((dataframe_by_Group * 100), decimals=2)
ax = dataframe_by_Group.plot.bar(figsize=(15,7));        vals = ax.get_yticks()
ax.set_yticklabels(['{:3.0f}%'.format(x)        for        x        in        vals]);
ax.set_xticklabels(dataframe_by_Group.index, rotation    =    0,    fontsize    =    15);
ax.set_title('mortality    having    or    not    by    given    attributes    (%)    (by    '    +
dataframe_by_Group.index.name    +    '\n',        fontsize        =

```

```
15) ax.set_xlabel(dataframe_by_Group.index.name, fontsize = 12) ax.set_ylabel('%'),
```

```
fontsize = 12)      ax.legend(loc = 'upper left',bbox_to_anchor=(1.0,1.0), fontsize= 12)
rects = ax.patches
```

```
# Add Data Labels
```

```
for rect in rects:
    height = rect.get_height()
    ax.text(rect.get_x() + rect.get_width()/2,
    height + 2,          str(height)+'%',
    ha='center',        va='bottom',
    fontsize = 12)

    return dataframe_by_Group
qul_No_qul_bar_plot(df, 'fetal_movement')
```

```
import pandas as p import matplotlib.pyplot as plt import
seaborn as s import numpy as n data =
p.read_csv("child.csv") import warnings
warnings.filterwarnings('ignore') df=data.dropna()
df.columns() from sklearn.preprocessing import
LabelEncoder var_mod = ['baseline value', 'accelerations',
'fetal_movement',
    'uterine_contractions', 'light_decelerations', 'severe_decelerations',
    'prolongued_decelerations', 'abnormal_short_term_variability',
    'mean_value_of_short_term_variability',
    'percentage_of_time_with_abnormal_long_term_variability',
    'mean_value_of_long_term_variability', 'histogram_width',
    'histogram_min', 'histogram_max', 'histogram_number_of_peaks',
    'histogram_number_of_zeroes', 'histogram_mode', 'histogram_mean',
    'histogram_median', 'histogram_variance', 'histogram_tendency',
    'fetal_health']
le = LabelEncoder()
```

```
for i in var_mod:
```

```

df[i] = le.fit_transform(df[i]).astype(str) from sklearn.metrics import confusion_matrix,
classification_report, matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score
X = df.drop(labels='fetal_health', axis=1)
#Response variable
y = df.loc[:, 'fetal_health'] del df from
sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.3,
random_state=1, stratify=y)
Del x,y
X_train.is_copy = False X_test.is_copy = False from
sklearn.linear_model import LogisticRegression
logR= LogisticRegression()

logR.fit(X_train,y_train)

predictR = logR.predict(X_test)

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

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

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

```

```
cm1[1,1]/(cm1[1,0]+cm1[1,1])  
print('Specificity : ', specificity1) print("")  
TN = cm1[0][0]
```

```

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

dtree.fit(X_train, y_train)

predictDT = dtree.predict(X_test)

print("") print('Classification report of Decision Tree Classifier
Results:') print("")

```

```
print(classification_report(y_test,predictDT))  
x = (accuracy_score(y_test,predictDT)*100)
```

```

print('Accuracy result of Decision Tree Classifier is', x)
print("")
cm2=confusion_matrix(y_test,predictDT)
print('Confusion Matrix result of Decision Tree Classifier is:\n',
confusion_matrix(y_test,predictDT)) print("")

sensitivity1 =
cm2[0,0]/(cm2[0,0]+cm2[0,1])
print('Sensitivity : ', sensitivity1 ) print("")
specificity1 =
cm2[1,1]/(cm2[1,0]+cm2[1,1])
print('Specificity : ', specificity1)
TN = cm2[0][0]
FN = cm2[1][0]
TP = cm2[1][1] FP =
cm2[0][1] print("True
Positive :",TP) print("True
Negative :",TN)
print("False Positive :",FP)
print("False Negative
:",FN) print("")
TPR = TP/(TP+FN)
TNR = TN/(TN+FP)
FPR = FP/(FP+TN)
FNR = FN/(TP+FN)
print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)
print("")
PPV = TP/(TP+FP)
NPV =
TN/(TN+FN)

```

```
print("Positive Predictive Value :",PPV)
print("Negative predictive value :",NPV) TN =
cm2[0][0]
```

```

FN = cm2[1][0]
TP = cm2[1][1] FP =
cm2[0][1] print("True
Positive :",TP) print("True
Negative :",TN)
print("False Positive :",FP)
print("False Negative
:",FN) print("")
TPR = TP/(TP+FN)
TNR = TN/(TN+FP)
FPR = FP/(FP+TN)
FNR = FN/(TP+FN)
print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)
print("")
PPV = TP/(TP+FP)
NPV =
TN/(TN+FN)
print("Positive Predictive Value :",PPV) print("Negative predictive value :",NPV)
from sklearn.preprocessing import LabelEncoder
var_mod = ['baseline value', 'accelerations',
'fetal_movement',
'uterine_contractions', 'light_decelerations', 'severe_decelerations',
'prolongued_decelerations', 'abnormal_short_term_variability',
'mean_value_of_short_term_variability',
'percentage_of_time_with_abnormal_long_term_variability',
'mean_value_of_long_term_variability', 'histogram_width',
'histogram_min', 'histogram_max', 'histogram_number_of_peaks',
'histogram_number_of_zeroes', 'histogram_mode', 'histogram_mean',
'histogram_median', 'histogram_variance', 'histogram_tendency',
'fetal_health']
le = LabelEncoder()

```

```
for i in var_mod:  
    df[i] = le.fit_transform(df[i]).astype(str)
```

```

from sklearn.metrics import confusion_matrix, classification_report, matthews_corrcoef,
cohen_kappa_score, accuracy_score, average_precision_score, roc_auc_score
X = df.drop(labels='fetal_health', axis=1)
#Response variable
y = df.loc[:, 'fetal_health']
del df
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1,
stratify=y)
Del x,y
X_train.is_copy = False X_test.is_copy = False
from sklearn.ensemble import RandomForestClassifier rfc
= RandomForestClassifier()

rfc.fit(X_train,y_train)

predictR = rfc.predict(X_test)

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

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

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

```

FN = cm1[1][0]

```

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

s.fit(X_train,y_train)

predicts = s.predict(X_test)

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

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

```

```
print('Accuracy result of Support Vector Machines is:', x)
print("")
```

```

cm2=confusion_matrix(y_test,predicts) print('Confusion Matrix
result of Support Vector Machines is:\n',cm2) print("") sensitivity1 =
cm2[0,0]/(cm2[0,0]+cm2[0,1]) print('Sensitivity : ', sensitivity1 )
print("") specificity1 = cm2[1,1]/(cm2[1,0]+cm2[1,1]) print('Specificity
: ', specificity1) print("")
TN = cm2[0][0]
FN = cm2[1][0]
TP = cm2[1][1] FP =
cm2[0][1] print("True
Positive :",TP) print("True
Negative :",TN)
print("False Positive :",FP)
print("False Negative
:",FN) print("")
TPR = TP/(TP+FN)
TNR = TN/(TN+FP)
FPR = FP/(FP+TN)
FNR = FN/(TP+FN)
print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)
print("")
PPV = TP/(TP+FP)
NPV =
TN/(TN+FN)
print("Positive Predictive Value :",PPV)
print("Negative predictive value :",NPV)

```

```

import pandas as p import matplotlib.pyplot as plt import
seaborn as s import numpy as n data =
p.read_csv("child.csv") import warnings

```

```
warnings.filterwarnings('ignore') df=data.dropna()  
df.columns() from sklearn.preprocessing import
```

```

LabelEncoder var_mod = ['baseline value', 'accelerations',
'fetal_movement',
    'uterine_contractions', 'light_decelerations', 'severe_decelerations',
    'prolongued_decelerations', 'abnormal_short_term_variability',
    'mean_value_of_short_term_variability',
    'percentage_of_time_with_abnormal_long_term_variability',
    'mean_value_of_long_term_variability', 'histogram_width',
    'histogram_min', 'histogram_max', 'histogram_number_of_peaks',
    'histogram_number_of_zeroes', 'histogram_mode', 'histogram_mean',
    'histogram_median', 'histogram_variance', 'histogram_tendency',
    'fetal_health']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i]).astype(str)
from sklearn.metrics import confusion_matrix,
classification_report, matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score
X = df.drop(labels='fetal_health', axis=1)
#Response variable
y = df.loc[:, 'fetal_health']
del df
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1,
stratify=y)
Del x ,y
X_train.is_copy = False X_test.is_copy = False
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier()

dtree.fit(X_train, y_train)

predictDT = dtree.predict(X_test) dt=
(accuracy_score(y_test,predictDT)*100)
from

```

```
sklearn.ensemble import RandomForestClassifier rfc
= RandomForestClassifier()
```

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

```
predictR = rfc.predict(X_test) rf =
(accuracy_score(y_test,predictR)*100)
from sklearn.svm import SVC s = SVC()
```

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

```
predicts = s.predict(X_test) s =
(accuracy_score(y_test,predicts)*100) from
sklearn.neighbors import KNeighborsClassifier
```

```
knn = KNeighborsClassifier()
```

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

```
predictknn = knn.predict(X_test) knn =
(accuracy_score(y_test,predictknn)*100) def
graph():
```

```
    import matplotlib.pyplot as plt
```

```
data=[dt,rf,s,knn]
```

```
alg="DT","RF","SVM","KNN"
```

```
plt.figure(figsize=(10,5))
```

```
b=plt.bar(alg,data,color=("g","b","y","m"))
```

```
plt.title("Accuracy comparison ",fontsize=15)
```

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

```
plt.savefig('accuracy.png') graph() import tkinter
```

```
from      matplotlib.backends.backend_tkagg      import      (FigureCanvasTkAgg,
```

```
NavigationToolbar2Tk) from matplotlib.backend_bases import
```

```
key_press_handler from matplotlib.figure import Figure import numpy as
```

```
np root = tkinter.Tk() root.wm_title("Accuracy plot") fig =  
Figure(figsize=(10,10),dpi=1) canvas = FigureCanvasTkAgg(fig,  
master=root) canvas.draw()
```

```
canvas.get_tk_widget().pack(side=tkinter.TOP, fill=tkinter.BOTH,  
expand=1) icon=tkinter.PhotoImage(file='accuracy.png')  
label=tkinter.Label(root,image=icon) label.pack() root.mainloop()
```