

CARDIOVASCULAR DISEASE PREDICTION USING SUPERVISED MACHINE LEARNING TECHNIQUES

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

By

NIVETHA.V (42739018)



**DEPARTMENT OF BIOINFORMATICS
SCHOOL OF BIO AND CHEMICAL ENGINEERING**

SATHYABAMA

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

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

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

APRIL – 2024

“CARDIOVASCULAR DISEASE PREDI



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC | 12B Status by UGC | Approved by AICTE
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119
www.sathyabama.ac.in

DEPARTMENT OF BIOINFORMATICS BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **NIVETHA V (42739018)**, who carried out the final year project entitled "**CARDIOVASCULAR PREDICTION USING SUPERVISED MACHINE LEARNING TECHNIQUES**" under our supervision from January 2024 to April 2024.

Dr. DANIEL ALEX ANAND

Internal Guide

Dr. DANIEL ALEX ANAND

External Guide

Dr. H. JEMMY CHRISTY

Head of the Department

Submitted for Viva Voce Examination held on 20/4/24

Internal Examiner

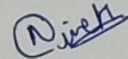
External Examiner

DECLARATION

I, **NIVETHA.V**, hereby declare that the Project Report entitled **CARDIOVASCULAR PREDICTION USING SUPERVISED MACHINE LEARNING TECHNIQUES** done by me under the guidance of **Dr. DANIEL ALEX ANAND** at Sathyabama Institute of Science and Technology, Jeppiaar Nagar, Rajiv Gandhi Road, Chennai-600119, Tamil Nadu, India, is submitted in partial fulfillment of the requirements for the award of **Master of Science in Bioinformatics and Data Science**.

DATE: 20/4/2024

PLACE: Chennai



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

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

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

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

I am deeply indebted to my Project Guide, DR. DANIEL ALEX ANAND, for his invaluable guidance, suggestions, and constant encouragement throughout the project. His expertise and insights have been instrumental in shaping my work and ensuring its successful completion.

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

ABSTRACT

Cardiovascular diseases (CVDs) are a leading cause of mortality and morbidity worldwide, necessitating the development of accurate predictive models for early detection and intervention. This research focuses on the application of supervised machine learning techniques to predict cardiovascular events and outcomes based on patient data. The performance of each model is evaluated using appropriate evaluation metrics, such as accuracy, precision, recall, and area under the receiver operating characteristic. Additionally, feature importance analysis is conducted to identify the key variables contributing to the prediction of cardiovascular events. This research highlights the effectiveness of supervised machine learning techniques in predicting cardiovascular events using patient data. The developed models have the potential to aid healthcare professionals in identifying at-risk individuals and implementing personalized preventive strategies. Further research can explore the integration of additional data sources, such as genetic information and wearable device data, to enhance the predictive capabilities and enable proactive cardiovascular care.

Keywords: cardiovascular diseases, supervised machine learning, predictive modelling, risk assessment.

TABLE OF CONTENTS

SL. NO	TITLE	PAGE NO
	ABSTRACT	I
	LIST OF CONTENTS	II
	LIST OF FIGURES	IV
	LIST OF ABBREVIATION	V
1	INTRODUCTION	1
1.1	Data Science	1
1.2	Artificial Intelligence	2
1.3	Machine Learning	3
2	REVIEW OF LITERATURE	5
3	AIM AND SCOPE	7
3.1	Aim	7
3.2	Scope	7
4	MATERIALS AND METHODS	8
4.1	Data Wrangling	8
4.1.1	Data Collection	8
4.1.2	Data Preprocessing	8
4.1.3	Software Requirements	9
4.2	Construction of Predicted model	13
4.2.1	Data analysis and Visualization	18
4.2.2	Comparing Algorithm with prediction in the form best accuracy result	18
4.2.3	Algorithms and Techniques	19
4.2.4	Deployment	27
5	RESULTS AND DISCUSSION	41
5.1	Comparing Algorithm with prediction in the form best accuracy result	41

CHAPTER NO	TITLE	PAGE NO
5.2	Prediction result by accuracy	41
5.3	Accuracy Calculation	42
6	SUMMARY AND CONCLUSION	49
7	REFERENCES	51

LIST OF FIGURES

S.NO	PARTICULARS	PAGE NO
1	SYSTEM ARCITECTURE	14
2	WORKFLOW DIAGRAM	15
3	USECASE DIAGRAM	15
4	CLASS DIAGRAM	16
5	ACTIVITY DIAGRAM	16
6	SEQUENCE DIAGRAM	17
7	ER-DIAGRAM	17
8	MODULE DIAGRAM	18

LIST OF ABBREVIATIONS

IPYNB	INTERACTIVE PYTHON NOTEBOOK
AI	ARTIFICIAL INTELLIGENCE
ML	MACHINE LEARNING
RAM	RANDOM ACCESS MEMORY
RAII	RESOURCE ACQUISITION IS INITIALIZATION
NOI	NO OPERATION
NLP	NATURAL LANGUAGE PROCESSING

CHAPTER 1

INTRODUCTION

Cardiovascular Prediction using Machine Learning is the system that is used to predict the diseases from the symptoms which are given by the patients or any user. The system processes the symptoms provided by the user as input and gives the output as the probability of the disease.(Chitral et al.,2013)

1.1 Data Science

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured data, and apply knowledge and actionable insights from data across a broad range of application domains. The term "data science" has been traced back to 1974, when Peter Naur proposed it as an alternative name for computer science. In 1996, the International Federation of Classification Societies became the first conference to specifically feature data science as a topic. However, the definition was still in flux. The term "data science" was first coined in 2008 by D.J. Patil, and Jeff Hammerbacher, the pioneer leads of data and analytics efforts at LinkedIn and Facebook. In less than a decade, it has become one of the hottest and most trending professions in the market.(Chaudary et al,2020)

Data science is the field of study that combines domain expertise, programming skills, and knowledge of mathematics and statistics to extract meaningful insights from data. Data science can be defined as a blend of mathematics, business acumen, tools, algorithms and machine learning techniques, all of which help us in finding out the hidden insights or patterns from raw data which can be of major use in the formation of big business decisions.(Suganya et al,2020)

1.2 ARTIFICIAL INTELLIGENCE

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

term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving. Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to the natural intelligence displayed by humans or animals. Leading AI textbooks define the field as the study of "intelligent agents" any system that perceives its environment and takes actions that maximize its chance of achieving its goals. Some popular accounts use the term "artificial intelligence" to describe machines that mimic "cognitive" functions that humans associate with the human mind, such as "learning" and "problem solving", however this definition is rejected by major AI researchers.(Golande et al,2019)

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

In general, AI systems work by ingesting large amounts of labeled training data, analyzing the data for correlations and patterns, and using these patterns to make predictions about future states. In this way, a chatbot that is fed examples of text chats can learn to produce life like exchanges with people, or an image recognition tool can learn to identify and describe objects in images by reviewing millions of examples. AI programming focuses on three cognitive skills: learning, reasoning and self-correction.(Hazara et al.,2017)

Learning processes. This aspect of AI programming focuses on acquiring data and creating rules for how to turn the data into actionable information. The rules, which are called algorithms, provide computing devices with step-by-step instructions for how to complete a specific task.(Seenivasagam et al.,2013)

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

Self-correction processes. This aspect of AI programming is designed to continually fine-tune algorithms and ensure they provide the most accurate results possible.(Limbitote et al,2020)

1.3 MACHINE LEARNING

Machine learning is to predict the future from past data. Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data and the basics of Machine Learning, implementation of a simple machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms. It feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. Machine learning can be roughly separated in to three categories. There are supervised learning, unsupervised learning and reinforcement learning. Supervised learning program is both given the input data and the corresponding labelling to learn data has to be labelled 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.(Abinaya et al., 2018)

Data scientists use many different kinds of machine learning algorithms to discover patterns in python that lead to actionable insights. At a high level, these different algorithms can be classified into two groups based on the way they “learn” about data to make predictions: supervised and unsupervised learning. Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function from input variables(X) to discrete output variables(y). In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc.(Mahajan et al., 2020)



Process Of Machine Learning

[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, multi-class classification, Decision Trees and support vector machines etc. Supervised learning requires that the data used to train the algorithm is already labelled with correct answers. Supervised learning problems can be further grouped into Classification problems. This problem has as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for categorical for classification. A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. A classification problem is when the output variable is a category, such as “red” or “blue”.(Mandal et al,2017)

CHAPTER 2

REVIEW OF LITERATURE

Cardiovascular diseases are one of the diseases that account for the loss of millions of lives each year. Lack of early prediction is the primary reason for the loss of lives, and this encourages researchers to develop intelligent systems for better prediction. In this paper, a novel ensemble methodology is introduced which uses the voting of Logistic Regression(LR), Random Forest(RF), Artificial Neural Network activated with ReLU function(NNR), K-Nearest Neighbors (KNN) and Gaussian Naive Bayes(GNB) to predict the possibility of heart disease. The model is developed using Python-based Jupyter Notebook and Flask and is trained using the standard dataset from Kaggle. The model is tested and evaluated based on accuracy, precision, specificity, sensitivity, error. Testing witnessed an accuracy of 89% and a precision of 91.6%, along with a sensitivity of 86% and specificity of 91%. The results upon comparison with the individual models witness the better accuracy of using ensemble modeling and hence a better prediction leading to life-saving.(Chowdary et al., 2020)

The Healthcare industry generally clinical diagnosis is done mostly by doctor's expertise and experience. Computer Aided Decision Support System plays a major role in medical field. With the growing research on heart disease predicting system, it has become important to categories the research outcomes and provides readers with an overview of the existing heart disease prediction techniques in each category. Neural Networks are one of many data mining analytical tools that can be utilized to make predictions for medical data. From the study it is observed that Hybrid Intelligent Algorithm improves the accuracy of the heart disease prediction system. The commonly used techniques for Heart Disease Prediction and their complexities are summarized in this paper. (Chitral et al., 2013)

In today's era deaths due to heart disease has become a major issue approximately one person dies per minute due to heart disease. This is considering both male and female category and this ratio may vary according to the region also this ratio is considered for the people of age group 25-69. This does not indicate that the people

with other age group will not be affected by heart diseases. This problem may start in early age group also and predict the cause and disease is a major challenge nowadays. Here in this paper, we have discussed various algorithms and tools used for prediction of heart diseases. (Golande et al., 2019)

A popular saying goes that we are living in an “information age”. Terabytes of data are produced every day. Data mining is the process which turns a collection of data into knowledge. The health care industry generates a huge amount of data daily. However, most of it is not effectively used. Efficient tools to extract knowledge from these databases for clinical detection of diseases or other purposes are not much prevalent. The aim of this paper is to summarize some of the current research on predicting heart diseases using data mining techniques, analyse the various combinations of mining algorithms used and conclude which technique(s) are effective and efficient. Also, some future directions on prediction systems have been addressed. (Hazra et al., 2017)

Heart is one of the most important part of the body. It helps to purify and circulate blood to all parts of the body. Most number of deaths in the world are due to Heart Diseases. Some symptoms like chest pain, faster heartbeat, discomfort in breathing are recorded. This data is analysed on regular basis. In this review, an overview of the heart disease and its current procedures is firstly introduced. Furthermore, an in-depth analysis of the most relevant machine learning techniques available on the literature for heart disease prediction is briefly elaborated. The discussed machine learning algorithms are Decision Tree, SVM, ANN, Naive Bayes, Random Forest, KNN. The algorithms are compared on the basis of features. We are working on the algorithm with best accuracy. This will help the doctors to assist the heart problem easily. (Limbitote et al., 2020)

CHAPTER 3

AIM AND SCOPE

3.1 Aim

Heart attack is one of the major factors in our healthcare domain. There are lot of patients who are actively present in the world. It is difficult to find the heart attack. So this project can easily find out the attack.

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

- 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

3.2 Scope

Here the scope of the project is that integration of patients attack with computer-based prediction could reduce errors and improve prediction outcome. This suggestion is promising as data modeling and analysis tools, e.g., data mining, have the potential to generate a knowledge-rich environment which can help to significantly improve the quality of heart attack prediction.

CHAPTER 4

MATERIALS AND METHODS

4.1 Data Wrangling

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

4.1.1 Data collection

The data set collected for predicting given data is split into Training set and Test set. Generally, 7:3 ratios are applied to split the Training set and Test set. The Data Model which was created using Random Forest, logistic, Decision tree algorithms and Support vector classifier (SVC) are applied on the Training set and based on the test result accuracy, Test set prediction is done.

4.1.2 Data Preprocessing

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.

Building the classification model: The prediction of Heart attack, a high accuracy prediction model is effective because of the following reasons: It provides better results in classification problem.

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

4.1.3 Software and Hardware Requirements

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,
- Problem define
- Preparing data
- Evaluating algorithms
- Improving results
- 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 GB

RAM : minimum 2 GB

➤ ***SOFTWARE DESCRIPTION***

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

➤ **ANACONDA NAVIGATOR**

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

For data science work, Anaconda is a great option. Anaconda is created by Continuum Analytics, and it is a Python distribution that comes preinstalled with lots of useful python libraries for data science.

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

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

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

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

➤ **JUPYTER NOTEBOOK**

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

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

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

File Extension:

An **IPYNB** file is a notebook document created by Jupyter Notebook, an interactive computational environment that helps scientists manipulate and analyze data using Python.

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.

➤ **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 overarching perspective to step through a small project.
- It will give you confidence, maybe to go on to your own small projects.

➤ **Python:**

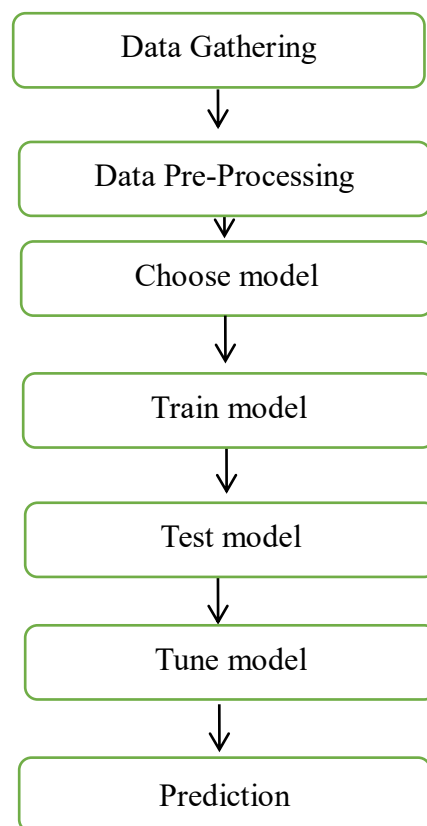
Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting.

Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible. Python 2 was discontinued with version 2.7.18 in 2020.

4.2 Construction of a Predictive Model

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



Process of dataflow diagram

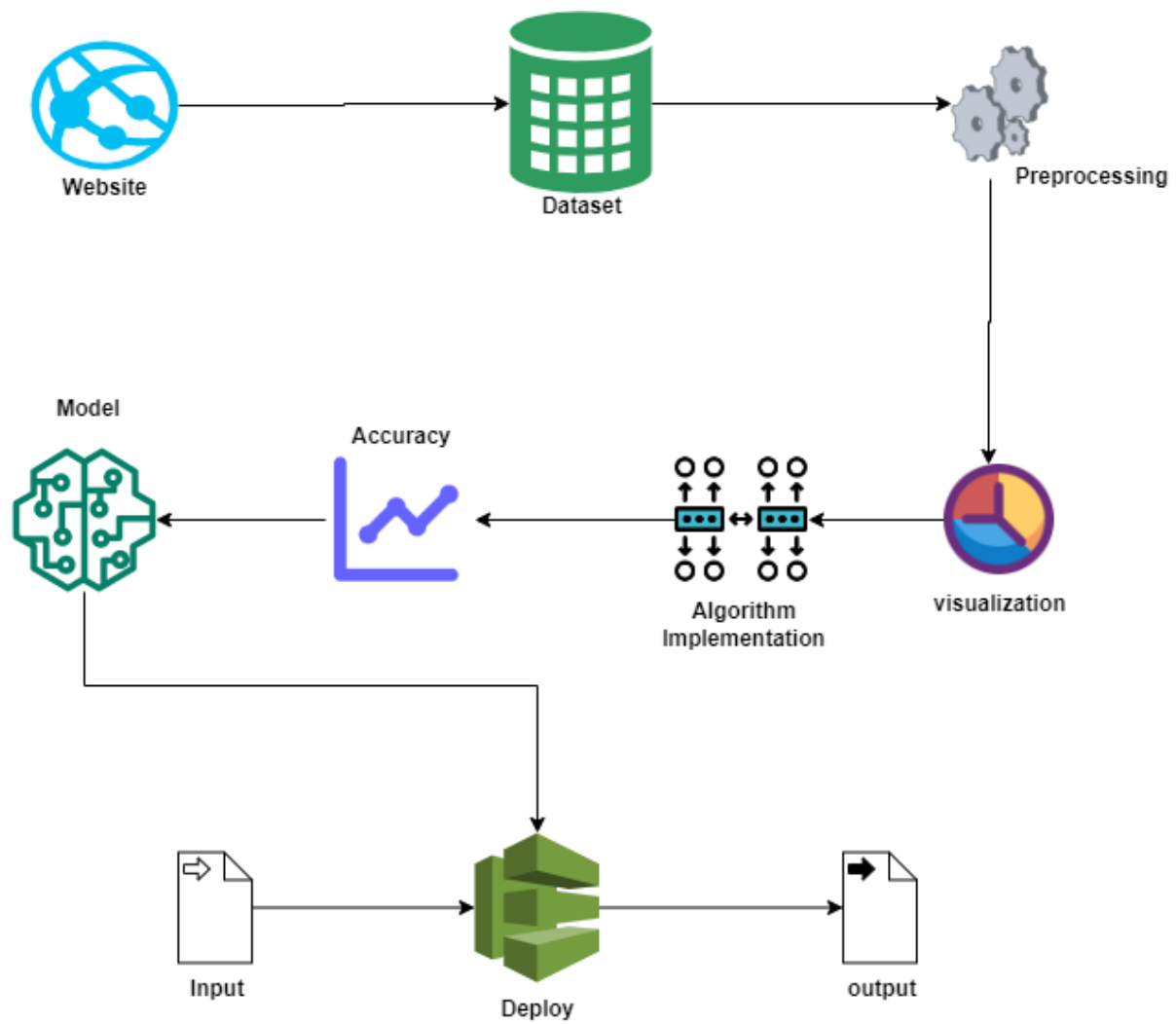
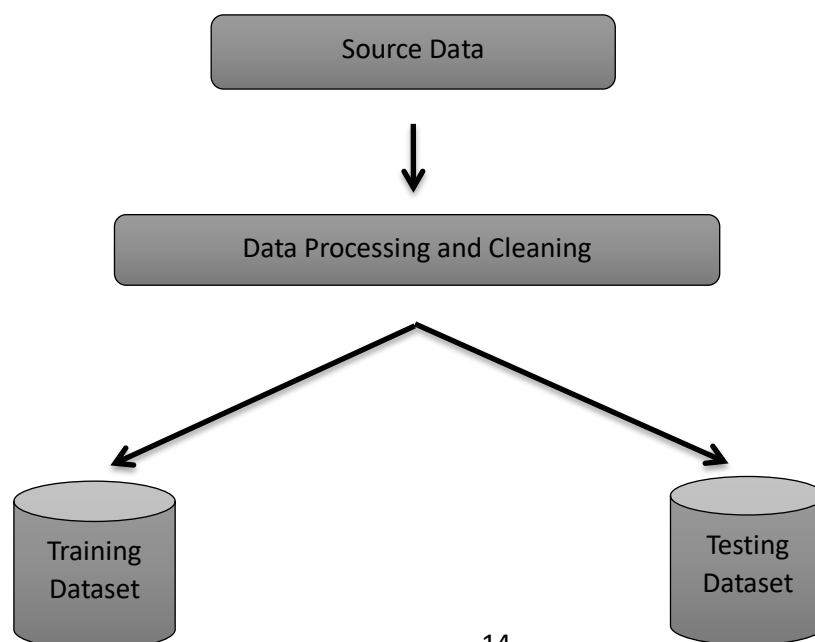


Figure 1: System Architecture



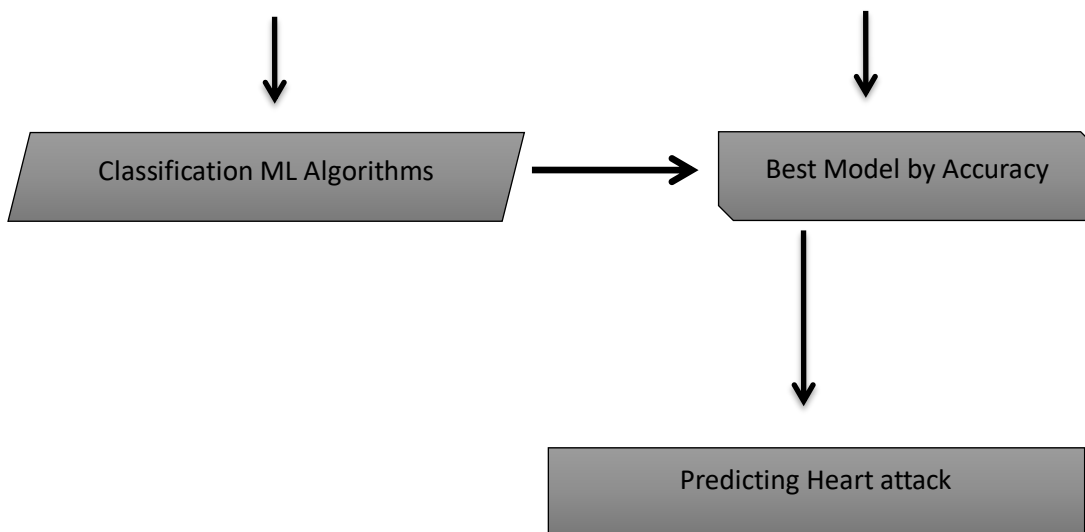


Figure 2: Work flow diagram

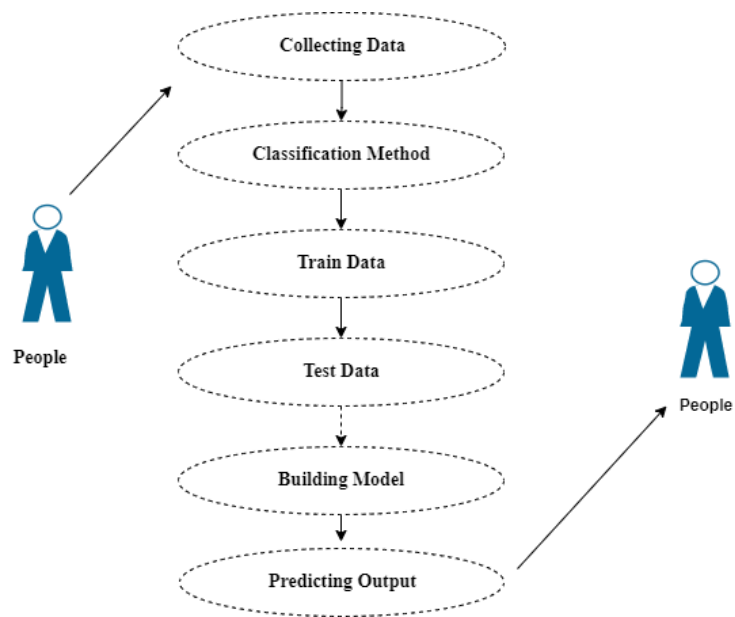


Figure 3: Use Case Diagram

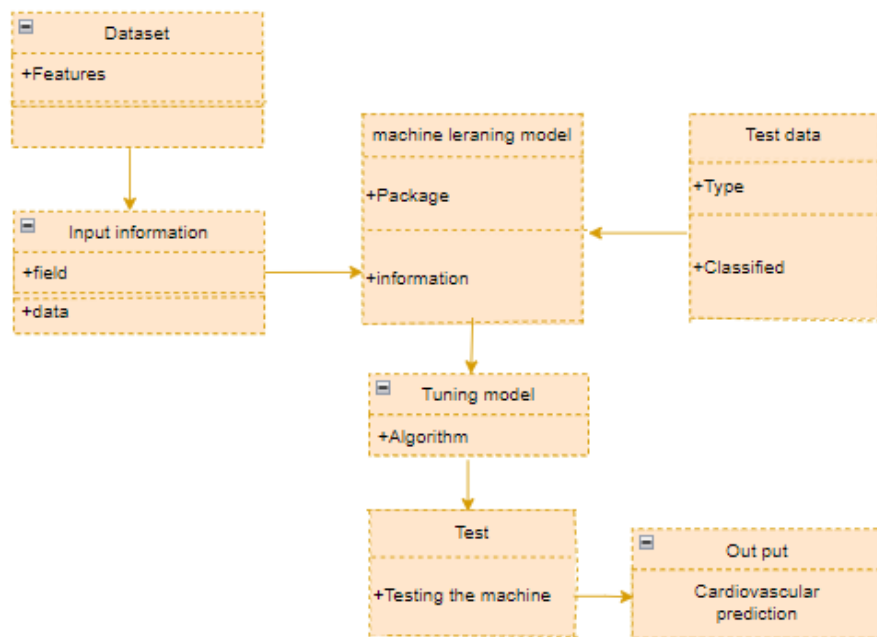


Figure 4: Class Diagram:

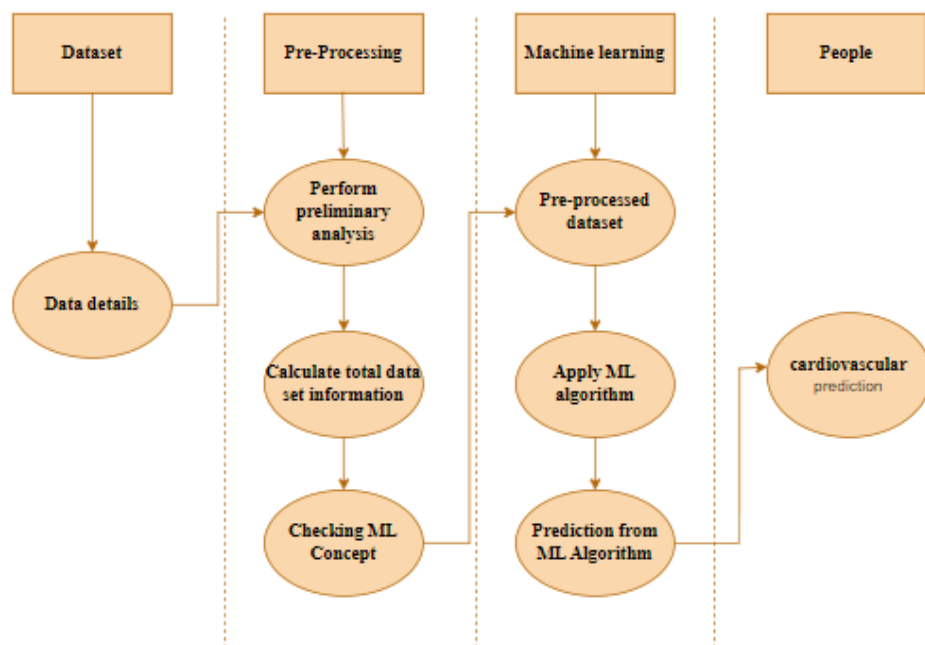


Figure 5: Activity Diagram:

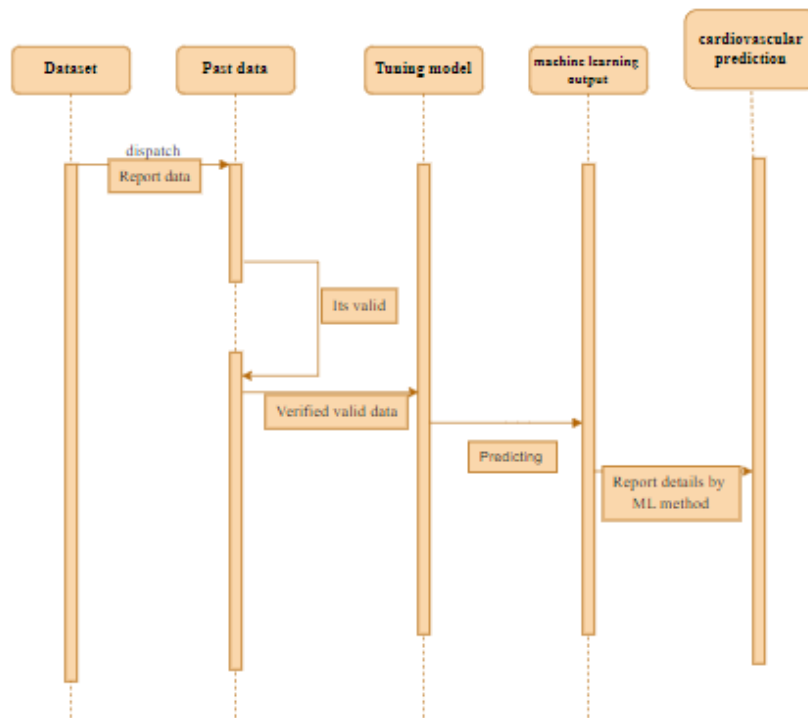


Figure 6: Sequence Diagram:

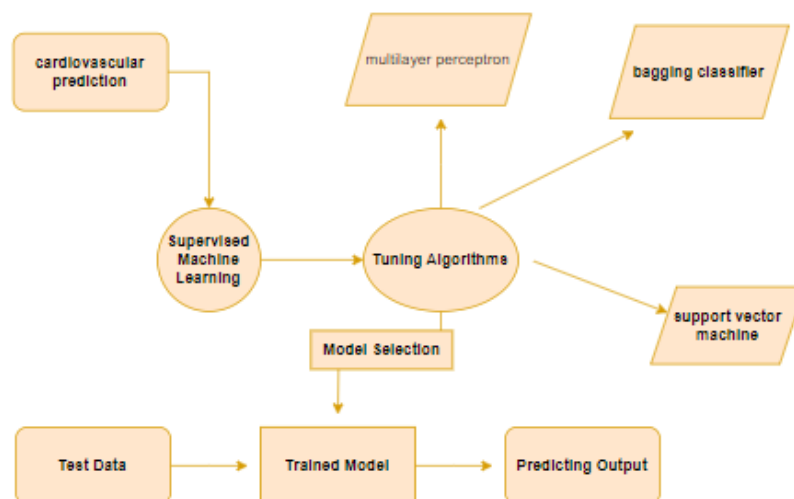


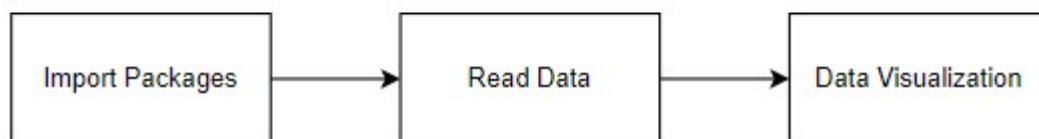
Figure 7: Entity Relationship Diagram (ERD)

4.2.1 Data analysis and 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.

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input: data

output: visualized data

4.2.2 Comparing Algorithm with prediction in the form of best accuracy result

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

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

In the example below 3 different algorithms are compared:

- Bagging Classifier
- Support vector machine
- MLP Classifier

4.2.3 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 bi-class (like identifying whether the person is male or female or that the mail is spam or non-

spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc. In Supervised Learning, algorithms learn from labelled 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 unlabelled 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.

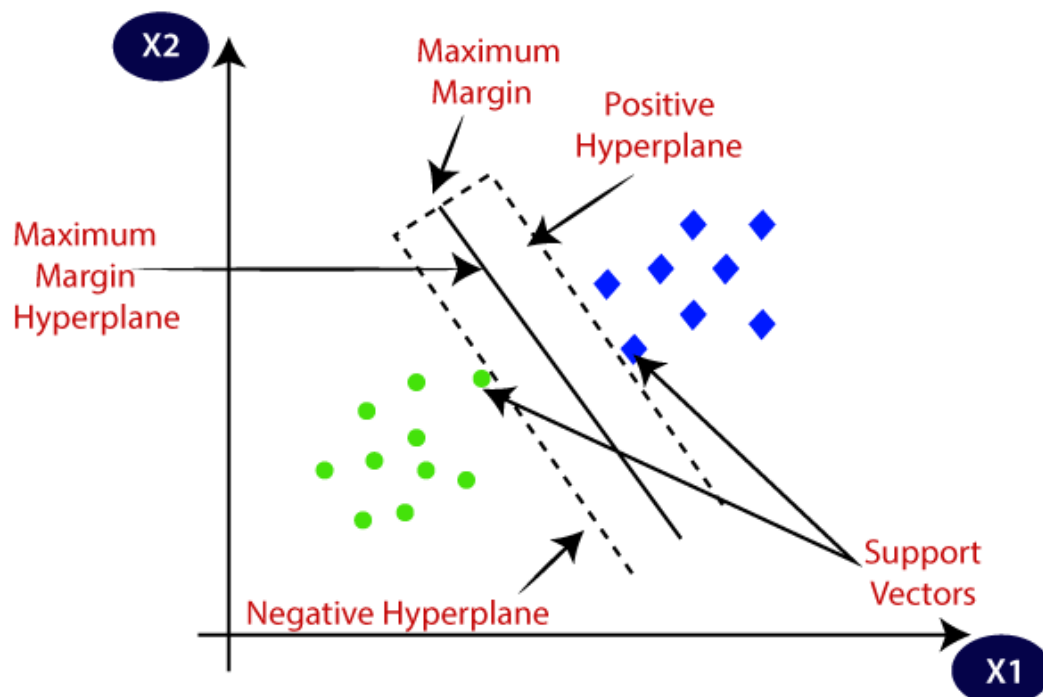
SVM:

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new

data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



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

THE ACCURACY SCORE OF SUPPORT VECTOR MACHINE IS : 68.72037914691943

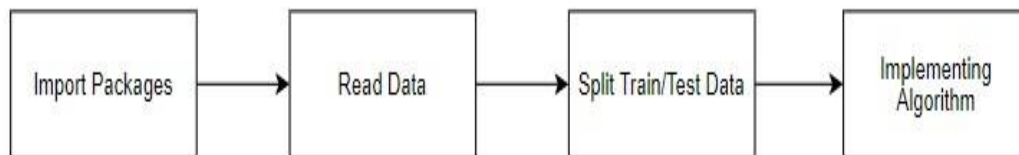
```
from sklearn.metrics import hamming_loss
hl = hamming_loss(y_test,predicted)
print("THE HAMMING LOSS OF SUPPORT VECTOR MACHINE IS :",hl*100)
```

THE HAMMING LOSS OF SUPPORT VECTOR MACHINE IS : 31.27962085308057

```
from sklearn.metrics import precision_score
P = precision_score(y_test,predicted)
print("THE PRECISION SCORE OF SUPPORT VECTOR MACHINE IS :",P*100)
```

THE PRECISION SCORE OF SUPPORT VECTOR MACHINE IS : 67.56756756756756

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input: data

output: getting accuracy

Bagging Classifier

Bagging, or bootstrap aggregation, is the ensemble getting-to-know method generally used to lessen variance within a loud dataset. In Bagging, a random pattern of statistics in this study set is selected with replacement, meaning that the character statistics factors may be chosen more soon as possible. After numerous facts samples are generated, those susceptible fashions are trained independently. For example, the common or Majority of these predictions yield a correct estimate depending on the sort of task- regression or type. As a note, the random woodland set of rules is considered an extension of the bagging approach, using both bagging and function randomness to create an uncorrelated wooded area of selection trees.

The Bagging is an assembling approach that tries to resolve overfitting for class or the regression problems. Bagging pursuits to improve the accuracy and overall performance of gadget mastering algorithms. It does this by taking random subsets of an original dataset, with substitute, and fits either a classifier (for classification) or regressor (for regression) to each subset. Bagging is also known as Bootstrap aggregating. It is an ensemble learning approach that enhances the overall performance and accuracy of the gadget for learning algorithms. It is miles used to address bias-variance alternate-off increases and decreases the variance of a prediction version. The Bagging avoids overfitting of data and is used for each regression and classification of the class, in particular for the decision tree algorithms.

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

THE ACCURACY SCORE OF BAGGING CLASSIFIER IS : 100.0

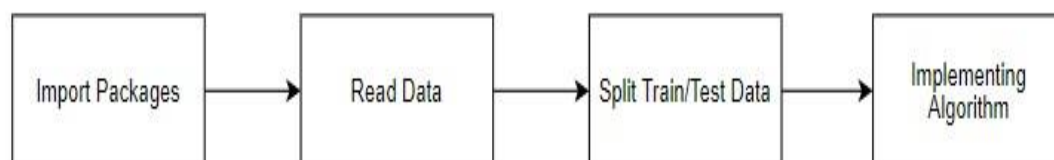
```
from sklearn.metrics import hamming_loss
hl = hamming_loss(y_test,predicted)
print("THE HAMMING LOSS OF BAGGING CLASSIFIER IS :",hl*100)
```

THE HAMMING LOSS OF BAGGING CLASSIFIER IS : 0.0

```
from sklearn.metrics import precision_score
P = precision_score(y_test,predicted)
print("THE PRECISION SCORE OF BAGGING CLASSIFIER IS :",P*100)
```

THE PRECISION SCORE OF BAGGING CLASSIFIER IS : 100.0

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input: data

output: getting accuracy

MLP Classifier

MLP Classifier stands for Multi-layer Perceptron classifier which in the name itself connects to a Neural Network. Unlike other classification algorithms such as Support Vectors or Naive Bayes Classifier, MLP Classifier relies on an underlying Neural Network to perform the task of classification. Since there are multiple layers of neurons, MLP is a deep learning technique. MLP is widely used for solving problems that require supervised learning as well as research into computational neuroscience and parallel distributed processing. MLPs are suitable for classification prediction problems where inputs are assigned a class or label. They are also suitable for regression prediction problems where a real-valued quantity is predicted given a set of inputs.

Disadvantages of MLP include too many parameters because it is fully connected.

The MLP (Multilayer Perceptron) is a type of artificial neural network used in machine learning. It is a feedforward neural network that consists of multiple layers, including an input layer, one or more hidden layers, and an output layer. MLPs are a type of supervised learning algorithm used for both regression and classification tasks.

Here's a detailed explanation of the Multilayer Perceptron algorithm:

1. Neural Network Basics:

- An MLP is composed of nodes (neurons) organized into layers. Each connection between nodes has an associated weight, and each node has an associated bias. The layers are:
 - **Input Layer:** Receives the input features.
 - **Hidden Layers:** Intermediate layers between the input and output layers.
 - **Output Layer:** Produces the final output.

2. Feedforward Process:

- During the feedforward process, input data is passed through the network layer by layer. The input values are multiplied by the weights, and the sum is passed through an activation function to produce the output of each node.

3. **Activation Functions:**

- Activation functions introduce non-linearity into the model, enabling it to learn complex patterns. Common activation functions include:
 - **Sigmoid:** Scales the output between 0 and 1.
 - **Hyperbolic Tangent (tanh):** Scales the output between -1 and 1.
 - **Rectified Linear Unit (ReLU):** Returns 0 for negative inputs and the input for positive inputs.

4. **Training - Backpropagation:**

- MLPs are trained using the backpropagation algorithm. The key steps are:
 - **Forward Pass:** Compute the predicted output based on the input.
 - **Compute Loss:** Compare the predicted output with the actual target and compute the loss.
 - **Backward Pass (Backpropagation):** Propagate the error backward through the network, adjusting weights and biases to minimize the loss.

5. **Loss Function:**

- The loss function measures the difference between the predicted output and the actual target. Common loss functions include:
 - **Mean Squared Error (MSE):** Used for regression tasks.
 - **Cross-Entropy Loss:** Used for classification tasks.

6. **Optimization Algorithm:**

- During training, an optimization algorithm is used to minimize the loss. Popular optimization algorithms include:
 - **Gradient Descent:** Updates weights in the opposite direction of the gradient.
 - **Stochastic Gradient Descent (SGD):** Optimizes with a random subset of data in each iteration.
 - **Adam, RMSprop, and others:** Variants of gradient descent with adaptive learning rates.

7. Regularization Techniques:

- To prevent overfitting, regularization techniques may be applied, such as:
 - **L1 and L2 Regularization:** Add penalties to the weights to prevent large values.
 - **Dropout:** Randomly dropout nodes during training to improve generalization.

8. Hyperparameter Tuning:

- MLPs have several hyperparameters, including the number of hidden layers, the number of nodes in each layer, the learning rate, and the choice of activation functions. Hyperparameter tuning is crucial for achieving optimal performance.

9. Batch Training vs. Online Training:

- Training can be done using batches of data or one data point at a time. Batch training provides stability, while online training adapts to new data quickly.

10. Applications:

- MLPs are used for a variety of tasks, including image recognition, natural language processing, speech recognition, and many other machine learning applications.

11. Deep Learning:

- MLPs are a foundational concept in deep learning. Deep neural networks refer to networks with many hidden layers, and training deep networks has become more feasible with advancements like better initialization techniques and the development of deep learning frameworks.

12. Frameworks:

- Implementing MLPs can be done using various machine learning frameworks such as TensorFlow, PyTorch, and Keras, which provide high-level abstractions for building and training neural networks.

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

THE ACCURACY SCORE OF MLP CLASSIFIER IS : 85.78199052132702

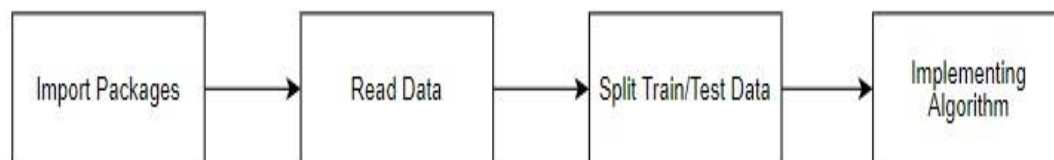
```
from sklearn.metrics import hamming_loss
hl = hamming_loss(y_test,predicted)
print("THE HAMMING LOSS OF MLP CLASSIFIER IS :",hl*100)
```

THE HAMMING LOSS OF MLP CLASSIFIER IS : 14.218009478672986

```
from sklearn.metrics import precision_score
P = precision_score(y_test,predicted)
print("THE PRECISION SCORE OF MLP CLASSIFIER IS :",P*100)
```

THE PRECISION SCORE OF MLP CLASSIFIER IS : 85.71428571428571

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input: data

output: getting accuracy

4.3.1 Deployment

Django (Web FrameWork) :

Django is an extremely popular and fully featured server-side web framework, written in Python. Django is one of the most popular web server frameworks. It has some advanced functionality. Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel.

Django follows the "Batteries included" philosophy and provides almost everything developers might want to do "out of the box". Because everything you need is part of the one "product", it all works seamlessly together, follows consistent design principles, and has extensive and up-to-date documentation.

Versatile

Django can be (and has been) used to build almost any type of website — from content management systems and wikis, through to social networks and news sites. It can work with any client-side framework, and can deliver content in almost any format (including HTML, RSS feeds, JSON, XML, etc). The site you are currently reading is built with Django.

Internally, while it provides choices for almost any functionality you might want (e.g. several popular databases, templating engines, etc.), it can also be extended to use other components if needed.

Secure

Django helps developers avoid many common security mistakes by providing a framework that has been engineered to "do the right things" to protect the website automatically. For example, Django provides a secure way to manage user accounts and passwords, avoiding common mistakes like putting session information in cookies where it is vulnerable (instead cookies just contain a key, and the actual data is stored in the database) or directly storing passwords rather than a password hash.

A password hash is a fixed-length value created by sending the password through a cryptographic hash function. Django can check if an entered password is correct by running it through the hash function and comparing the output to the stored hash value. However due to the "one-way" nature of the function, even if a stored hash value is compromised it is hard for an attacker to work out the original password.

Django enables protection against many vulnerabilities by default, including SQL injection, cross-site scripting, cross-site request forgery and clickjacking (see Website security for more details of such attacks).

Scalable

Django uses a component-based “shared-nothing” architecture (each part of the architecture is independent of the others, and can hence be replaced or changed if needed). Having a clear separation between the different parts means that it can scale for increased traffic by adding hardware at any level: caching servers, database servers, or application servers. Some of the busiest sites have successfully scaled Django to meet their demands (e.g. Instagram and Disqus, to name just two).

Maintainable

Django code is written using design principles and patterns that encourage the creation of maintainable and reusable code. In particular, it makes use of the Don't Repeat Yourself (DRY) principle so there is no unnecessary duplication, reducing the amount of code. Django also promotes the grouping of related functionality into reusable "applications" and, at a lower level, groups related code into modules (along the lines of the Model View Controller (MVC) pattern).

Portable

Django is written in Python, which runs on many platforms. That means that you are not tied to any particular server platform, and can run your applications on many flavours of Linux, Windows, and Mac OS X. Furthermore, Django is well-supported by many web hosting providers, who often provide specific infrastructure and documentation for hosting Django sites.

➤ ***Start Using an API***

- Most APIs require an API key. ...

- The easiest way to start using an API is by finding an HTTP client online, like REST-Client, Postman, or Paw.

1. The next best way to pull data from an API is by building a URL from existing API documentation.
2. The flask object implements a WSGI application and acts as the central object. It is passed the name of the module or package of the application. Once it is created it will act as a central registry for the view functions, the URL rules, template configuration and much more.
3. The name of the package is used to resolve resources from inside the package or the folder the module is contained in depending on if the package parameter resolves to an actual python package (a folder with an `__init__.py` file inside) or a standard module (just a `.py` file).
4. For more information about resource loading, see `open_resource()`.
5. Usually you create a Flask instance in your main module or in the `__init__.py` file of your package.

➤ **Parameters**

- **rule** (*str*) – The URL rule string.
- **endpoint** (*Optional[str]*) – The endpoint name to associate with the rule and view function. Used when routing and building URLs. Defaults to `view_func.__name__`.
- **view_func** (*Optional[Callable]*) – The view function to associate with the endpoint name.
 - **provide_automatic_options** (*Optional[bool]*) – Add the `OPTIONS` method and respond to `OPTIONS` requests automatically.
- **options** (*Any*) – Extra options passed to the Rule object.

Return type -- None

`After_Request(f)`

Register a function to run after each request to this object.

The function is called with the response object, and must return a response object. This allows the functions to modify or replace the response before it is sent.

If a function raises an exception, any remaining after request functions will not be called. Therefore, this should not be used for actions that must execute, such as to close resources. Use `teardown_request()` for that.

Parameters:

(*Callable*[[*Response*], *Response*])

Return type

Callable[[*Response*], *Response*]

`after_request_funcs`: `t.Dict[AppOrBlueprintKey,`

`t.List[AfterRequestCallable]]`

A data structure of functions to call at the end of each request, in the format `{scope: [functions]}`. The scope key is the name of a blueprint the functions are active for, or `None` for all requests.

To register a function, use the `after_request()` decorator.

This data structure is internal. It should not be modified directly and its format may change at any time.

`app_context()`

Create an `AppContext`. Use as a with block to push the context, which will make `current_app` point at this application.

An application context is automatically pushed by `RequestContext.push()` when handling a request, and when running a CLI command. Use this to manually create a context outside of these situations.

With `app.app_context()`:

`Init_db()`

HTML

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

Basic Construction of an HTML Page

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



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

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

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

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

Further Tags

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

`<title>`

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

`<meta>`

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

HeadTag

`<head>`

`<title>My First Webpage</title>`

`<meta charset="UTF-8">`

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

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

`</header>`

Adding Content

Next, we will make `<body>` tag.

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

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

Add HTML Headings To Web Page

In HTML, headings are written in the following elements:

- <h1>
- <h2>
- <h3>
- <h4>
- <h5>
- <h6>

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

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

Creating Your Heading

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

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

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

Add Text In HTML

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

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

Add Links In HTML

The internet is made up of lots of [links](#).

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

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

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

Image Tag

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

```

```

CSS

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

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

CSS Syntax

```
Selector {
```

```
Property 1 : value;
```

```
Property 2 : value;
```

```
Property 3 : value;
```

```
}
```

For example:

```
h1
```

```
{
```

```
Color: red;
```

```
Text-align: center;
```

```
}
```

```
#unique
```

```
{
```

```
color: green;
```

```
}
```

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

CSS Comment

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

CSS How-To

- There are 3 ways to write CSS in our HTML file.
 - Inline CSS

- Internal CSS
- External CSS
- Priority order
 - Inline > Internal > External

Inline CSS

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

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

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

Internal CSS

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

`<style>`

`H1{`

`Color:red;`

`}`

`</style>`

<h3> Have a great day </h3>

External CSS

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

<head>

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

</head>

h1{

color:red; //.css file

}

CSS Selectors

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

CSS Colors

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

CSS Background

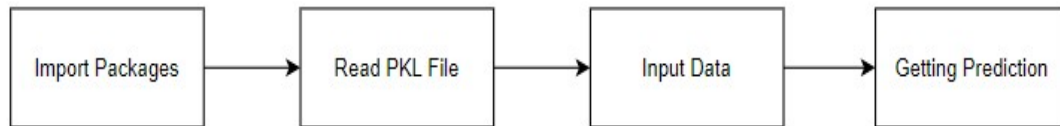
1. There are different ways by which CSS can have an effect on HTML elements
2. Few of them are as follows:
 - Color – used to set the color of the background
 - Repeat – used to determine if the image has to repeat or not and if it is repeating then how it should do that
 - Image – used to set an image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

CSS BoxModel

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

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

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data values output : predicting output

CHAPTER 5

RESULT AND DISCUSSION

5.1 Comparing Algorithm with prediction in the form of best accuracy result

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

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

In the example below 3 different algorithms are compared:

- Bagging Classifier
- Support vector machine
- MLP Classifier

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

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

5.2 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.

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 in 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.

$$\text{True Positive Rate(TPR)} = \text{TP} / (\text{TP} + \text{FN})$$

$$\text{False Positive rate(FPR)} = \text{FP} / (\text{FP} + \text{TN})$$

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

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target	
2	52	1	0	125	212	0	1	168	0	1	2	2	3	0	
3	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0	
4	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0	
5	61	1	0	148	203	0	1	161	0	0	2	1	3	0	
6	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0	
7	58	0	0	100	248	0	0	122	0	1	1	0	2	1	
8	58	1	0	114	318	0	2	140	0	4.4	0	3	1	0	
9	55	1	0	160	289	0	0	145	1	0.8	1	1	3	0	
10	46	1	0	120	249	0	0	144	0	0.8	2	0	3	0	
11	54	1	0	122	286	0	0	116	1	3.2	1	2	2	0	
12	71	0	0	112	149	0	1	125	0	1.6	1	0	2	1	
13	43	0	0	132	341	1	0	136	1	3	1	0	3	0	
14	34	0	1	118	210	0	1	192	0	0.7	2	0	2	1	
15	51	1	0	140	298	0	1	122	1	4.2	1	3	3	0	
16	52	1	0	128	204	1	1	156	1	1	1	0	0	0	
17	34	0	1	118	210	0	1	192	0	0.7	2	0	2	1	
18	51	0	2	140	308	0	0	142	0	1.5	2	1	2	1	
19	54	1	0	124	266	0	0	109	1	2.2	1	1	3	0	
20	50	0	1	120	244	0	1	162	0	1.1	2	0	2	1	
21	58	1	2	140	211	1	0	165	0	0	2	0	2	1	
22	60	1	2	140	185	0	0	155	0	3	1	0	2	0	
23	67	0	0	106	223	0	1	142	0	0.3	2	2	2	1	
24	45	1	0	104	208	0	0	148	1	3	1	0	2	1	
25	63	0	2	135	252	0	0	172	0	0	2	0	2	1	

DATA COLLECTION

	age	sex	cp	trestbps	chol	fbs	restecg	thalach
age	1.000000	-0.103240	-0.071966	0.271121	0.219823	0.121243	-0.132696	-0.390227
sex	-0.103240	1.000000	-0.041119	-0.078974	-0.198258	0.027200	-0.055117	-0.049365
cp	-0.071966	-0.041119	1.000000	0.038177	-0.081641	0.079294	0.043581	0.306839
restbps	0.271121	-0.078974	0.038177	1.000000	0.127977	0.181767	-0.123794	-0.039264
chol	0.219823	-0.198258	-0.081641	0.127977	1.000000	0.026917	-0.147410	-0.021772
fbs	0.121243	0.027200	0.079294	0.181767	0.026917	1.000000	-0.104051	-0.008866
restecg	-0.132696	-0.055117	0.043581	-0.123794	-0.147410	-0.104051	1.000000	0.048411
thalach	-0.390227	-0.049365	0.306839	-0.039264	-0.021772	-0.008866	0.048411	1.000000
exang	0.088163	0.139157	-0.401513	0.061197	0.067382	0.049261	-0.065606	-0.380281
oldpeak	0.208137	0.084687	-0.174733	0.187434	0.064880	0.010859	-0.050114	-0.349796

DATA PREPROCESSING AND DATA CLEANING

5.3 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.

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

THE ACCURACY SCORE OF MLP CLASSIFIER IS : 85.78199052132702
```

```
from sklearn.metrics import hamming_loss
hl = hamming_loss(y_test,predicted)
print("THE HAMMING LOSS OF MLP CLASSIFIER IS :",hl*100)

THE HAMMING LOSS OF MLP CLASSIFIER IS : 14.218009478672986
```

```
from sklearn.metrics import precision_score
P = precision_score(y_test,predicted)
print("THE PRECISION SCORE OF MLP CLASSIFIER IS :",P*100)

THE PRECISION SCORE OF MLP CLASSIFIER IS : 85.71428571428571
```

Precision: The proportion of positive predictions that are actually 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 labelled as survived, how many actually survived? High precision relates to the low false positive rate. We have got 0.788 precision which is pretty good.

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

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

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

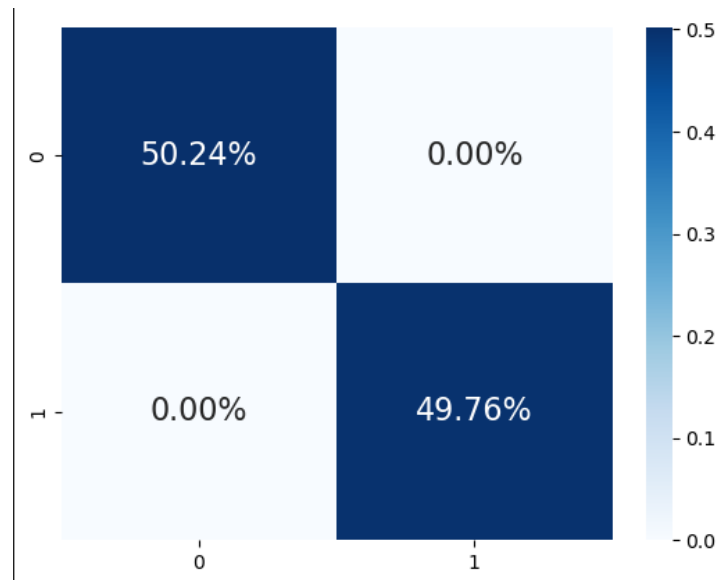
F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy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

General Formula:

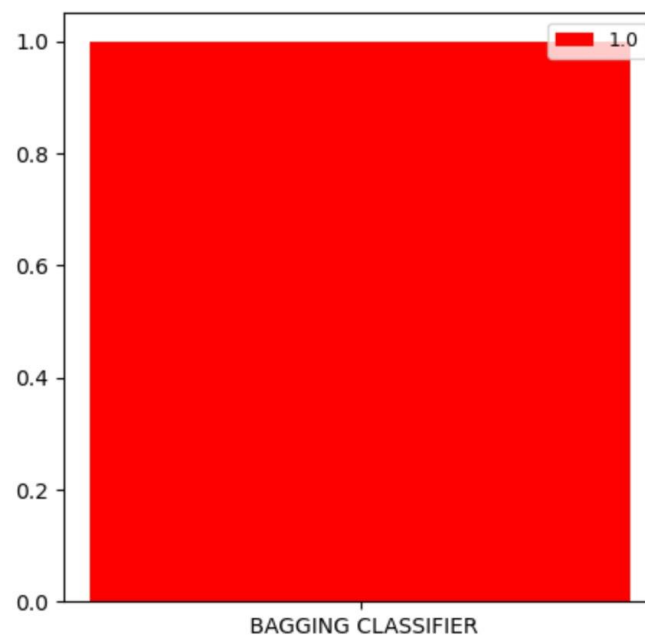
F- Measure = $2TP / (2TP + FP + FN)$

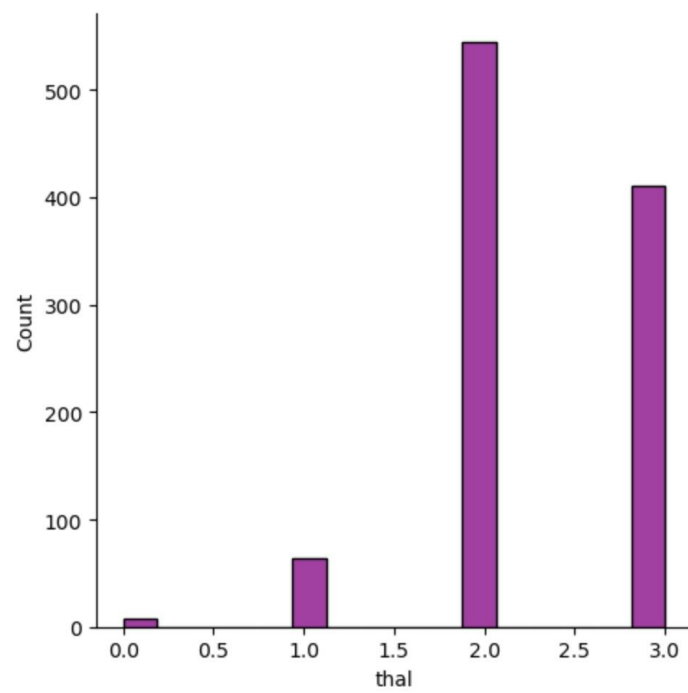
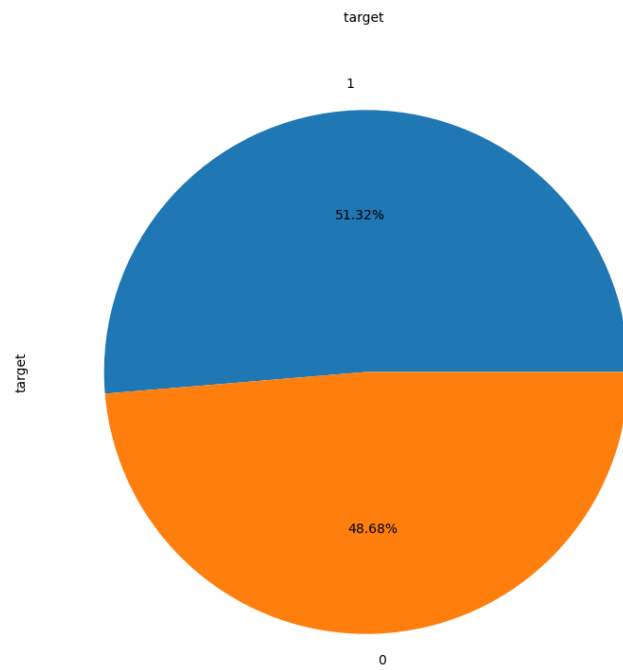
F1-Score Formula:

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

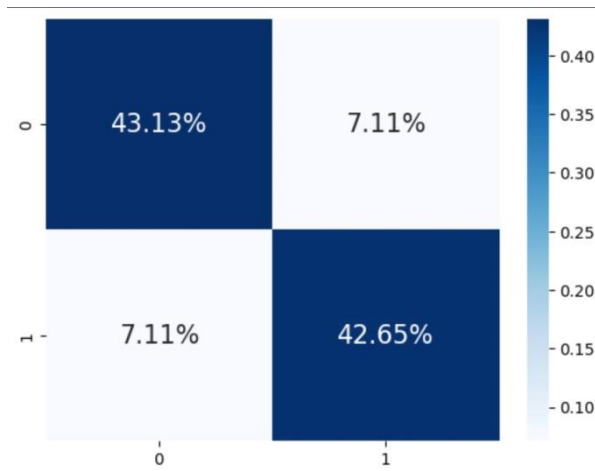


Bagging Classifier Algorithm

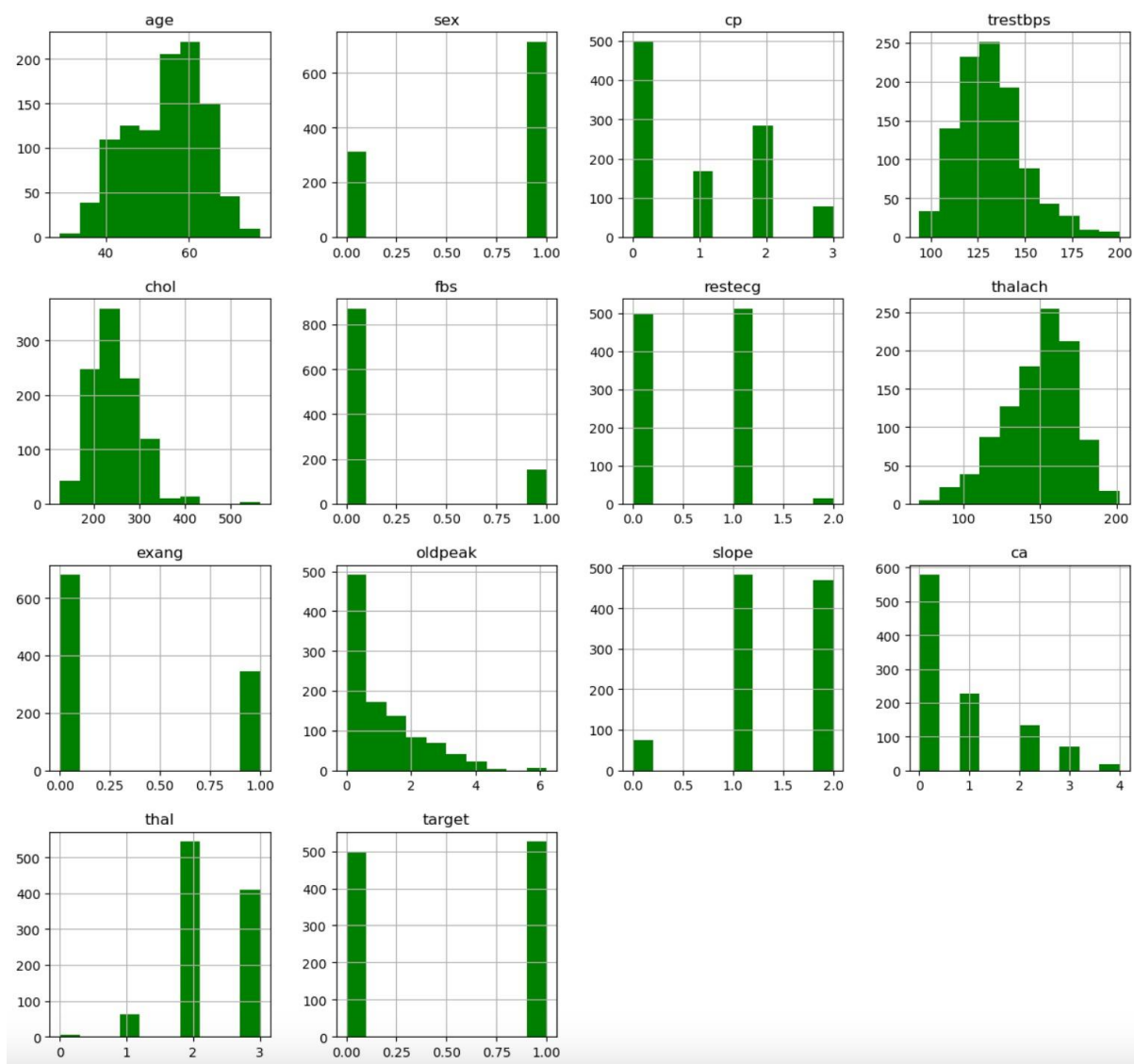




DATA VISUALIZATION AND DATA ANALYSIS



MULTI LAYER PERCEPTRON



DATA VISUALIZATION AND DATA ANALYSIS

OUTPUT SCREENSHOT



CARDIOVASCULAR PREDICTION USING SUPERVISED MACHINE LEARNING TECHNIQUES

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

AGE

AGE

SEX

SEX

CHEST_PAIN

CHEST_PAIN

TREST_BPS

TREST_BPS

CHOLESTEROL

CHOLESTEROL

FBS

FBS

REST_ECG

REST_ECG

HEART_RATE

HEART_RATE

EXANG

EXANG

OLDPEAK

OLDPEAK

SLOPE

SLOPE

CHAPTER 6

SUMMARY AND CONCLUSION

Summary

A dataset is selected from the UCI repository with patient's medical history and attributes. By using this dataset, we predict whether the patient can have a heart disease or not. To predict this, we use 14 medical attributes of a patient and classify him if the patient is likely to have a heart disease

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

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

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

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

(y) for that data. Techniques of Supervised Machine Learning algorithms include **logistic regression, multi-class classification, Decision Trees** and **support vector machines etc.** Supervised learning requires that the data used to train the algorithm is already labelled with correct answers. Supervised learning problems can be further grouped into **Classification** problems. This problem has as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for categorical for classification. A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes.

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 will be find out. This application can help to find the Prediction of Heart Attack.

Future Work

1. Heart attack prediction to connect with cloud.
2. To optimize the work to implement in Artificial Intelligence environment.

REFERENCES

- Anooj,P.K.,2012.Clinical decision support system: risk level prediction of heart disease using weighted fuzzyrules.J.KingSaudUniv.-ComputerInf.Sci.24(1),27–40.
<https://doi.org/10.1016/j.jksuci.2011.09.002>
- Bhatla,N.,Jyoti,K.,2012.Ananalysis of heart disease prediction using different data mining techniques. Int.J.Eng.1(8),1–4.
- Chaurasia,V.,Pal,S.,2013.Early prediction of heart diseases using data mining techniques. Carib. J. SciTech.1,208–217.
- Dey,A.,Singh,J.,Singh,N.,2016.Analysis of supervised machine learning algorithms for heart disease prediction with reduced number of attributes using principal componentanalysis.Analysis140(2),27–31.
- Khemphila,A.,Boonjing,V.,2011.Heart disease classification using neural network and feature selection.In:21st International Conference on Systems Engineering(ICSEng).IEEE,LasVegas,pp.406– 409.
<https://doi.org/10.1109/ICSEng.2011.80>
- Liu,X.,Wang,X.,Su,Q.,Zhang,M.,Zhu,Y.,Wang,Q.,Wang,Q.,2017.A hybrid classification system for heart disease diagnosis based on the RFRS method. Comput. Math. Methods Med. <https://doi.org/10.1155/2017/8272091>
- Marimuthu M, Abinaya A, 2018, A Review on Heart Disease Prediction using Machine Learning and Data Analytics Approach, International Journal of Computer Applications, Volume 181 – No. 18, pp20-25 .
- Nahar,J. ,Imam, T., Tickle, K.S., Chen, Y.P.P., 2013. Computational intelligence for heart disease diagnosis : a medical knowledge driven approach. Expert Syst. Appl. 40(1),96–104. <https://doi.org/10.1016/j.eswa.2012.07.032>
- Nahato,K.B.,Harichandran,K.N.,Arputharaj,K.,2015.Knowledge mining from clinical dataset susin grouch sets and back propagation neural network. Compute. Math.MethodsMed.2015,1– 13. <http://dx.doi.org/10.1155/2015/460189>

Paul,A.K.,Shill,P.C.,Rabin,M.R.I.,Akhand,M.A.H.,2016.Genetic algorithm based fuzzy decision support system for the diagnosis of heart disease.(ICIEV).In:5th International Conference on Informatics, Electronics and Vision.IEEE,pp.145–150. <https://dx.doi.org/10.1109/ICIEV.2016.7759984>.

Sadeghian,A Ismaeel,S.,Miri,A.,,Chourishi,D.,2015.An Extreme Learning Machine(ELM) Predictor for Electric Arc Furnaces' vi Characteristics. IEEE2nd International Conference on Cyber Security and Cloud Computing (Cloud) , New York ,pp.329–334. Kavitha, R., Kannan, E., 2016. An efficient framework for heart disease classification using feature extraction and features election technique in data mining. International Conference on Emerging Trends in Engineering, Technology and Science (ICETETS),pp.1–5. <https://dx.doi.org/10.1109/CSCloud.2015.94>.

Sen,A.K.,Patel,S.B.,Shukla,D.D.,2013.A data mining technique for prediction of coronary heart disease using neuro-fuzzy integrated approach two level.Int.J.Eng. Comput.Sci.(IJECS)2(8),2663–2671.

APPENDIX

Module – 1

Importing Warnings

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

```
import warnings  
warnings.filterwarnings('ignore')  
df = pd.read_csv('CARDIO.csv')  
df.head()  
df.tail()  
df.shape  
df.size  
df.columns  
df.isnull()  
df = df.dropna()  
df['thal'].unique()  
df.describe()  
df.corr()  
df.info()  
pd.crosstab(df["fbs"], df["target"])  
df.groupby(["trestbps", "age"]).groups  
df["target"].value_counts()  
pd.Categorical(df["slope"]).describe()  
df.duplicated()  
sum(df.duplicated())  
df=df.drop_duplicates()  
sum(df.duplicated())
```

Module – 2

Data Visualisation

```
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
df = pd.read_csv('CARDIO.csv')  
df.head()  
df.columns  
plt.figure(figsize=(12,7))
```

```

sns.countplot(x='slope',data=df)
plt.figure(figsize=(15,5))
plt.subplot(1,2,1)
plt.hist(df['trestbps'],color='red')
plt.subplot(1,2,2)
plt.hist(df['fbs'],color='blue')
df.hist(figsize=(15,55),layout=(15,4), color='green')
plt.show()
df['restecg'].hist(figsize=(10,5),color='yellow')
sns.ecdfplot(df['thal'], color='brown') # scatter, plot, triplot, stackplot
sns.kdeplot(df['age'], color='purple')
df['ca'].plot(kind='density')
sns.displot(df['thal'], color='purple')
# barplot, boxenplot, boxplot, countplot, displot, distplot, ecdfplot, histplot, kdeplot, pointplot, violinplot,
striplot
sns.ecdfplot(df['exang'], color='coral') # residplot, scatterplot
fig, ax = plt.subplots(figsize=(20,15))
sns.heatmap(df.corr(),annot = True, fmt='0.2%',cmap = 'autumn',ax=ax)
def plot(df, variable):
    dataframe_pie = df[variable].value_counts()
    ax = dataframe_pie.plot.pie(figsize=(9,9), autopct='%1.2f%%', fontsize = 10)
    ax.set_title(variable + ' \n', fontsize = 10)
    return np.round(dataframe_pie/df.shape[0]*100,2)

plot(df, 'target')

```

Module – 3

SVM:

```

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

import warnings
warnings.filterwarnings('ignore')
df = pd.read_csv('CARDIO.csv')
df.head()

df.columns
df=df.dropna()
df.tail()

x1 = df.drop(labels='target', axis=1)
y1 = df.loc[:, 'target']

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

ros =RandomOverSampler(random_state=42)
x,y=ros.fit_resample(x1,y1)

```

```

print("OUR DATASET COUNT      : ", Counter(y1))
print("OVER SAMPLING DATA COUNT : ", Counter(y))

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=42, stratify=y)
print("NUMBER OF TRAIN DATASET   : ", len(x_train))
print("NUMBER OF TEST DATASET    : ", len(x_test))
print("TOTAL NUMBER OF DATASET   : ", len(x_train)+len(x_test))

print("NUMBER OF TRAIN DATASET   : ", len(y_train))
print("NUMBER OF TEST DATASET    : ", len(y_test))
print("TOTAL NUMBER OF DATASET   : ", len(y_train)+len(y_test))

from sklearn.svm import SVC

SVM = SVC()
SVM.fit(x_train,y_train)

predicted = SVM.predict(x_test)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,predicted)
print('THE CONFUSION MATRIX SCORE OF SUPPORT VECTOR MACHINE:\n\n',cm)

from sklearn.model_selection import cross_val_score
accuracy = cross_val_score(SVM, x, y, scoring='accuracy')
print('THE CROSS VALIDATION TEST RESULT OF ACCURACY :\n\n', accuracy*100)

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

from sklearn.metrics import hamming_loss
hl = hamming_loss(y_test,predicted)
print("THE HAMMING LOSS OF SUPPORT VECTOR MACHINE IS :",hl*100)

from sklearn.metrics import precision_score
P = precision_score(y_test,predicted)
print("THE PRECISION SCORE OF SUPPORT VECTOR MACHINE IS :",P*100)

from sklearn.metrics import recall_score
R = recall_score(y_test,predicted)
print("THE RECALL SCORE OF SUPPORT VECTOR MACHINE IS :",R*100)

from sklearn.metrics import f1_score
f1 = f1_score(y_test,predicted)
print("THE PRECISION SCORE OF SUPPORT VECTOR MACHINE IS :",f1*100)

def plot_confusion_matrix(cm, title='THE CONFUSION MATRIX SCORE OF SUPPORT VECTOR
MACHINE\n\n', cmap=plt.cm.Blues):
    target_names=[""]
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(target_names))
    plt.xticks(tick_marks, target_names, rotation=45)
    plt.yticks(tick_marks, target_names)
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

cm=confusion_matrix(y_test, predicted)
print('THE CONFUSION MATRIX SCORE OF SUPPORT VECTOR MACHINE:\n\n')
print(cm)

```

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

```
def graph():
    import matplotlib.pyplot as plt
    data=[a]
    alg=" SUPPORT VECTOR MACHINE "
    plt.figure(figsize=(5,5))
    b=plt.bar(alg,data,color=("GREEN"))
    plt.title("THE ACCURACY SCORE OF SUPPORT VECTOR MACHINE IS\n\n\n")
    plt.legend(b,data,fontsize=9)
graph()
```

Module – 4

Bagging

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

```
import warnings
warnings.filterwarnings('ignore')
df = pd.read_csv('CARDIO.csv')
df.head()
```

```
df.columns
```

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

```
df.tail()
```

```
x1 = df.drop(labels='target', axis=1)
```

```
y1 = df.loc[:, 'target']
```

```
# import imblearn
```

```
from imblearn.over_sampling import RandomOverSampler
```

```
from collections import Counter
```

```
# from sklearn.utils.metaestimators import if_delegate_has_method
```

```
ros =RandomOverSampler(random_state=42)
```

```
x,y=ros.fit_resample(x1,y1)
```

```
print("OUR DATASET COUNT      :", Counter(y1))
```

```
print("OVER SAMPLING DATA COUNT :", Counter(y))
```

```
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=42, stratify=y)
```

```
print("NUMBER OF TRAIN DATASET  :", len(x_train))
```

```
print("NUMBER OF TEST DATASET   :", len(x_test))
```

```
print("TOTAL NUMBER OF DATASET  :", len(x_train)+len(x_test))
```

```
print("NUMBER OF TRAIN DATASET  :", len(y_train))
```

```
print("NUMBER OF TEST DATASET   :", len(y_test))
```

```
print("TOTAL NUMBER OF DATASET  :", len(y_train)+len(y_test))
```

```
from sklearn.ensemble import BaggingClassifier
```

```
BAG = BaggingClassifier()
```

```
BAG.fit(x_train,y_train)
```

```

predicted = BAG.predict(x_test)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,predicted)
print('THE CONFUSION MATRIX SCORE OF BAGGING CLASSIFIER:\n\n',cm)

from sklearn.model_selection import cross_val_score
accuracy = cross_val_score(BAG, x, y, scoring='accuracy')
print('THE CROSS VALIDATION TEST RESULT OF ACCURACY :'\n\n\n', accuracy*100)

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

from sklearn.metrics import hamming_loss
hl = hamming_loss(y_test,predicted)
print("THE HAMMING LOSS OF BAGGING CLASSIFIER IS :",hl*100)

from sklearn.metrics import precision_score
P = precision_score(y_test,predicted)
print("THE PRECISION SCORE OF BAGGING CLASSIFIER IS :",P*100)

from sklearn.metrics import recall_score
R = recall_score(y_test,predicted)
print("THE RECALL SCORE OF BAGGING CLASSIFIER IS :",R*100)

from sklearn.metrics import f1_score
f1 = f1_score(y_test,predicted)
print("THE PRECISION SCORE OF BAGGING CLASSIFIER IS :",f1*100)

def plot_confusion_matrix(cm, title='THE CONFUSION MATRIX SCORE OF BAGGING
CLASSIFIER\n\n', cmap=plt.cm.Blues):
    target_names=[]
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(target_names))
    plt.xticks(tick_marks, target_names, rotation=45)
    plt.yticks(tick_marks, target_names)
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

cm=confusion_matrix(y_test, predicted)
print('THE CONFUSION MATRIX SCORE OF BAGGING CLASSIFIER:\n\n')
print(cm)

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

def graph():
    import matplotlib.pyplot as plt
    data=[a]
    alg=" BAGGING CLASSIFIER"
    plt.figure(figsize=(5,5))
    b=plt.bar(alg,data,color=("RED"))
    plt.title("THE ACCURACY SCORE OF BAGGING CLASSIFIER IS'\n\n\n")
    plt.legend(b,data,fontsize=9)
graph()

import joblib

```

```
joblib.dump(BAG, 'CARDIO.pkl')
```

Module – 5

MLP Classifier

```
import pandas as pd
```

```
import numpy as np
```

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

```
import seaborn as sns
```

```
import warnings
```

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

```
df = pd.read_csv('CARDIO.csv')
```

```
df.head()
```

```
df.columns
```

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

```
df.tail()
```

```
x1 = df.drop(labels='target', axis=1)
```

```
y1 = df.loc[:, 'target']
```

```
import imblearn
```

```
from imblearn.over_sampling import RandomOverSampler
```

```
from collections import Counter
```

```
ros =RandomOverSampler(random_state=42)
```

```
x,y=ros.fit_resample(x1,y1)
```

```
print("OUR DATASET COUNT      :", Counter(y1))
```

```
print("OVER SAMPLING DATA COUNT :", Counter(y))
```

```
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20,  
random_state=42, stratify=y)
```

```
print("NUMBER OF TRAIN DATASET  :", len(x_train))
```

```
print("NUMBER OF TEST DATASET   :", len(x_test))
```

```
print("TOTAL NUMBER OF DATASET  :", len(x_train)+len(x_test))
```

```
print("NUMBER OF TRAIN DATASET  :", len(y_train))
```

```
print("NUMBER OF TEST DATASET   :", len(y_test))
```

```
print("TOTAL NUMBER OF DATASET  :", len(y_train)+len(y_test))
```

```
from sklearn.neural_network import MLPClassifier
```

```
MLP = MLPClassifier()
```

```
MLP.fit(x_train,y_train)
```

```
predicted = MLP.predict(x_test)
```

```
from sklearn.metrics import confusion_matrix
```

```
cm = confusion_matrix(y_test,predicted)
```

```
print('THE CONFUSION MATRIX SCORE OF MLP CLASSIFIER:\n\n\n',cm)
```

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

```
accuracy = cross_val_score(MLP, x, y, scoring='accuracy')
```

```

print('THE CROSS VALIDATION TEST RESULT OF ACCURACY :\n\n\n',
accuracy*100)

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

from sklearn.metrics import hamming_loss
hl = hamming_loss(y_test,predicted)
print("THE HAMMING LOSS OF MLP CLASSIFIER IS :",hl*100)

from sklearn.metrics import precision_score
P = precision_score(y_test,predicted)
print("THE PRECISION SCORE OF MLP CLASSIFIER IS :",P*100)

from sklearn.metrics import recall_score
R = recall_score(y_test,predicted)
print("THE RECALL SCORE OF MLP CLASSIFIER IS :",R*100)

from sklearn.metrics import f1_score
f1 = f1_score(y_test,predicted)
print("THE PRECISION SCORE OF MLP CLASSIFIER IS :",f1*100)

def plot_confusion_matrix(cm, title='THE CONFUSION MATRIX SCORE OF MLP
CLASSIFIER\n\n', cmap=plt.cm.Blues):
    target_names=[""]
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(target_names))
    plt.xticks(tick_marks, target_names, rotation=45)
    plt.yticks(tick_marks, target_names)
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

cm=confusion_matrix(y_test, predicted)
print('THE CONFUSION MATRIX SCORE OF MLP CLASSIFIER:\n\n')
print(cm)

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

def graph():
    import matplotlib.pyplot as plt
    data=[a]
    alg="MLP CLASSIFIER"
    plt.figure(figsize=(5,5))
    b=plt.bar(alg,data,color=("ORANGE"))
    plt.title("THE ACCURACY SCORE OF MLP CLASSIFIER IS\n\n\n")
    plt.legend(b,data,fontsize=9)
    graph()

```

