

INTERDISCIPLINARY PROJECT REPORT
at
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
(Deemed to be University)

Submitted in partial fulfilment of the requirements for the award,
Bachelor of Engineering Degree in Computer Science and Engineering.

By

NAME: SAPPASATYAHARSHA
(Reg.No: 40111142)

GUIDE – Dr.D.Sudha,M.E.,Ph.D.,



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 | 12 B Status

by UGC | Approved by AICTE

JEPPIAR NAGAR, RAJIV GANDHISALAI,

CHENNAI – 600119

APRIL-2023



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

Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Professional Training Report is the Bonafide work of **SAPPASATYAHARSHA** underwent the professional training in “**MACHINE LEARNING**” under our supervision from January 2023 to April 2022.

GUIDE – Dr.D.SUDHA,M.E., Ph.D.,

Head Of the Department Dr. LAKSHMAN L, M.E, Ph.D.

Submitted for Viva voice Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **SAPPASATYAHARSHA (Reg. No- 40111142)** here by declare that the Project Report entitled **ECHOCARDIOGRAM** done by me under the guidance of is **Dr.D.SUDHA,M.E., Ph.D.**, submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

Guidance and constant encouragement which paved way to successful completion of project.

We convey thanks to the management and all the teaching staffs and non-teaching staffs of the Computer Science and Engineering department who were helpful in many ways for the completion of the project. The satisfaction and elation that accompany the successful completion of any task would be incomplete without the mention of the people who have made it possible. It is our great privilege to express our gratitude and respect to all those who have guided during Professional Training.

I convey my thanks to **Dr LAKSHMAN L, M.E.,Ph.D.**, Head of the Department, Department of Computer Science and Engineering for providing necessary support and details at the right time during the progressive reviews.

Also, we thank the [Almighty](#) for supporting in the completion of the Professional Training Project.

TRAINING CERTIFICATE

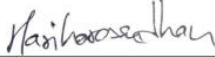


Certificate of Training

is hereby granted to

SappaSatyaharesha (40111142)

from Sathyabama Institute of Science and Technology for successfully completing the 45 hours professional training program on Machine Learning conducted by Cognibot between 10th Feb, 2023 and 16th Apr, 2023


HARIHARASUDHAN
INSTRUCTOR, COGNIBOT

16th April, 2023

DATE

Abstract

ECHOCARDIOGRAM

The to use Machine learning(ML) to more accurately predict survival after echocardiography. Predicting patient outcomes (e.g., survival) following echocardiography is primarily based on Ejection fraction (EF) and comorbidities. However, there may be significant predictive information within additional echocardiography-derived measurements combined with clinical electronic health record data.

Machine learning models achieved significantly higher prediction accuracy (all AUC >0.82) over common clinical risk scores (AUC = 0.61 to 0.79), with the nonlinear random forest models outperforming logistic regression ($p < 0.01$). The random forest model including all echocardiographic measurements yielded the highest prediction accuracy ($p < 0.01$ across all models and survival durations). Only 10 variables were needed to achieve 96% of the maximum prediction accuracy, with 6 of these variables being derived from echocardiography. Regurgitation velocity was more predictive of survival than LVEF. In a subset of studies with complete data for the top 10 variables, multivariate imputation by chained equations yielded slightly reduced predictive accuracies (difference in AUC of 0.003) compared with the original data. Machine learning can fully utilize large combinations of disparate input variables to predict survival after echocardiography with superior accuracy.

Predicting patient outcomes (e.g., survival) following echocardiography is primarily based on Ejection fraction (EF) and comorbidities. However, there may be significant predictive information within additional echocardiography-derived measurements combined with clinical electronic health record data. Machine learning can fully utilize large combinations of disparate input variables to predict survival after echocardiography with superior accuracy.

Table of Contents

CHAPTER	TITLE	PAGENO
1.	INTRODUCTION	8
1.1	BACKGROUND	8
1.2	MACHINE LEARNING	9
1.3	ECHOCARDIOGRAM	10
1.4	REPORT OUTLINE	11
2.	AIM & SCOPE	11
2.1	OBJECTIVE	11
2.2	PROJECT DESCRIPTION	12
3.	METHODS& MATERIAL	12
3.1	TEST/TRAIN SPLIT ALGORITHM	12
3.2	DECISION TREE	13
3.3	HOW ALGORITHM WORKS	14
3.4	ATTRIBUTE MEASURES	16
3.5	ACCURACY& PRECISION	17
4.	RESULT AND DISCUSSION	19
4.1	RESULTS	19
4.2	DISCUSSION	20
5.	IMPLEMENTATION	20
5.1	DECISION TREE CLASSIFIER	20
5.2	SYSTEM ARCHITECTURE	22
5.3	REFERENCES	23
6	CONCLUSION AND FUTURE WORK	24
	SOURCE CODE SCREENSHOTS	25

1. INTRODUCTION

1.1 Background

In this data science python project, we will perform one of the most essential applications of machine learning – decision tree. In this project, we will implement decision tree in python. Whenever you need to predict the best output for user, decision tree is the ideal methodology.

Decision Tree is the most important support tool with a tree-like structure that models probable outcomes, cost of resources, utilities, and possible consequences of supervised learning.

Decision trees are one of the best forms of learning algorithms based on various learning methods. They boost predictive models with accuracy, ease in interpretation, and stability. The tools are also effective in fitting non-linear relationships since they can solve data-fitting challenges, such as regression and classifications. Major applications like assessing prospective growth opportunities, using demographic data to predict prospective clients, serving as a support tool in several fields.

- Machine Learning (ML) is a branch of Artificial Intelligence (AI) & Computer Science which focuses on the use of data and algorithms to imitate the way that humans learn and gradually improving accuracy

1.2 Machine Learning

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people.

Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs.

Any technology user today has benefitted from machine learning. Facial recognition technology allows social media platforms to help users tag and share photos of friends. Optical character recognition (OCR) technology converts images of text into movable type. Recommendation engines, powered by machine learning, suggest what movies or television shows to watch next based on user preferences. Self-driving cars that rely on machine learning to navigate may soon be available to consumers.

Machine learning is a continuously developing field. Because of this, there are some considerations to keep in mind as you work with machine learning methodologies, or analyse the impact of machine learning process in machine learning, tasks are generally classified into broad categories. These categories are based on how learning is received or how feedback on the learning is given to the system developed. Two of the most widely adopted machine learning methods are **supervised learning** and **unsupervised learning**.

1.3 Echocardiogram

Echocardiography is a test that uses sound waves to produce live images of your heart. The image is called an echocardiogram. This test allows your doctor to monitor how your heart and its valves are functioning.

The images can help them get information about:

- the size of the heart, for instance, if there is any change in the chamber size, dilation, or thickening
- blood clots in the heart chambers
- fluid in the sac around the heart
- problems with the aorta, which is the main artery connected to the heart
- problems with the pumping function or relaxing function of the heart
- problems with the function of heart valves
- pressure in the heart

An echocardiogram is key in determining the health of the heart muscle, especially after a heart attack. It can also reveal heart defects, or irregularities, in unborn babies.

Getting an echocardiogram is painless. There are only risks in very rare cases with certain types of echocardiograms or if contrast is used for the echocardiogram.

Your doctor may order an echocardiogram for several reasons. For example, they may have discovered something unusual in other tests or while listening to your heartbeat through a stethoscope.

If you have an irregular heartbeat, your doctor may want to inspect the heart valves or chambers or check your heart's ability to pump. They may also order one if you are showing signs of heart problems, such as chest pain or shortness of breath or if you have an abnormal EKG (electrocardiogram).

1.4 Report Outline

1. Introduces the project's research area and the project's goals.
2. Presents some research papers that are related to this project.
3. Gives a thorough background about decision tree and the test/train methods. In this section, a set of chosen methods are chosen and presented.
4. Gives an evaluation of the chosen method.
5. The system implementation is described.
6. The results are discussed, and some future work is presented.

2. Aim & Scope

2.1 OBJECTIVE

The objective of the project is to find how to use machine learning techniques for analysis and making predictions using echocardiogram dataset. To find the classification goal is to predict still-alive using features from 3 to 10. The survival and still-alive variables, when taken together, indicate whether a patient survived for at least one year following the heart attack

This data set is used for exploratory data analysis, data visualization, dealing with missing values and classification modelling techniques

To predict still alive features 3 to 10 and classification of given data set

An ECG should be attached ensuring good tracings to facilitate the acquisition of complete digital loops. Loops should be examined and adjusted accordingly in order to ensure a clear representation of the image acquired.

The test is useful for diagnosing and monitoring heart problems and creating Treatment plans.

2.2 PROJECT DESCRIPTION

Data for classifying if patients will survive for at least one year after heart attack

Data Characteristics:	Set	Multivariate	Number of Instances:	132	Area:	Life
Attribute Characteristics:		Categorical, Integer, Real	Number of Attributes:	12	Date Donated	1989-02-28
Associated Tasks:		Classification	Missing Values?	Yes	Number of Web Hits:	229554

3. METHODS& MATERIAL

3.1.1TEST/TRAIN SPLIT ALGORITHM

The train-test split is a technique for evaluating the performance of a machine learning algorithm. It can be used for classification or regression problems and can be used for any supervised learning algorithm. The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the model; instead, the input element of the dataset is provided to model, then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset.

1. Train Dataset: Used to fit the machine learning model.
2. Test Dataset: Used to evaluate the fit machine learning model.

The objective is to estimate the performance of the machine learning model on new data, data not used to train the model. There is no optimal split percentage, common split percentages include:

1. Train: 80%, Test: 20%
2. Train: 67%, Test: 33%
3. Train: 50%, Test: 50%

3.1.2 DECISION TREE

A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements

Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal, but are also a popular tool machine learning

A decision tree is a flowchart-like structure in which each internal node represents a test on a feature (e. Whether a coin flip comes up heads or tails), each leaf node represents a class label (decision taken after computing all features) and branches represent conjunctions of features that lead to those class labels. The paths from root to leaf represent classification rules.

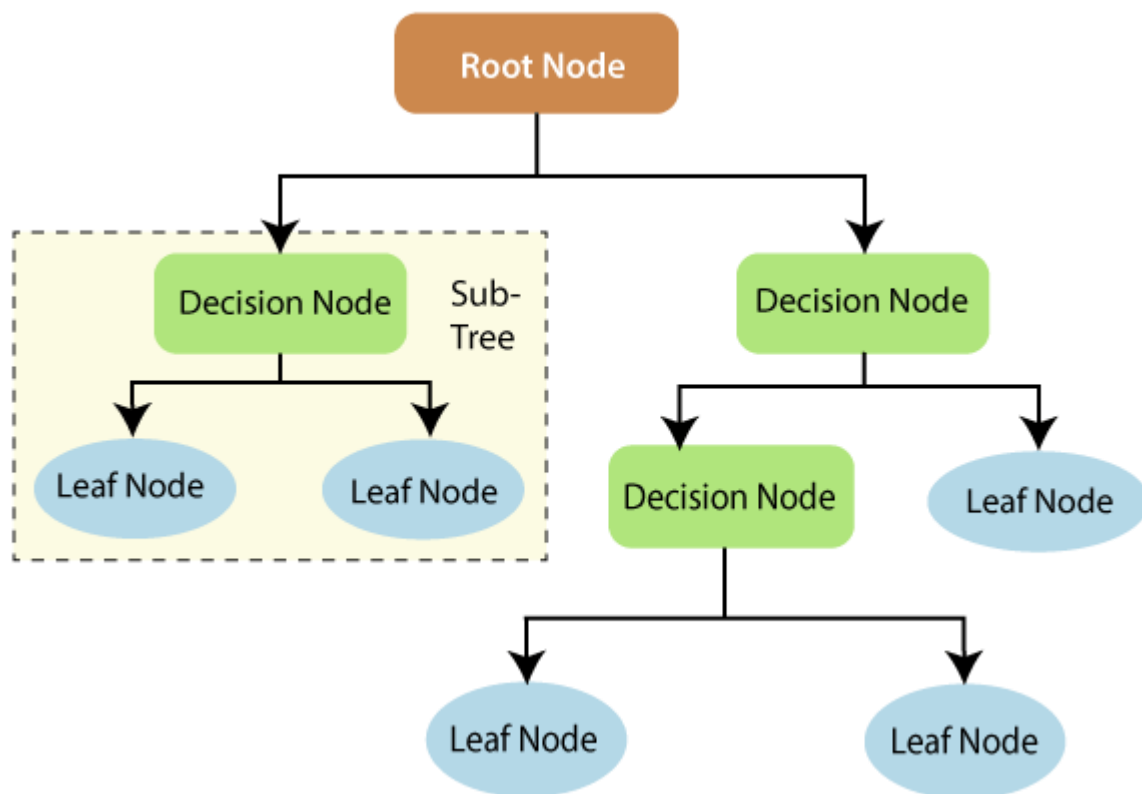
Below diagram illustrate the basic flow of decision tree for decision making with labels (Rain (Yes), No Decision tree is one of the predictive modelling approaches used in statistics, data mining and machine learning. Decision trees are constructed via an algorithmic approach that identifies ways to split a data set based on different conditions. It is one of the most widely used and practical methods for supervised learning. Decision Trees are a non-parametric supervised learning method used for both classification and regression tasks (No)).

3.1.3 HOW Algorithm works

Decision trees are the building blocks of a random forest algorithm. A decision tree is a decision support technique that forms a tree-like structure. An overview of decision trees will help us understand how random forest algorithms work.

A decision tree consists of three components: decision nodes, leaf nodes, and a root node. A decision tree algorithm divides a training dataset into branches, which further segregate into other branches. This sequence continues until a leaf node is attained. The leaf node cannot be segregated further.

The nodes in the decision tree represent attributes that are used for predicting the outcome. Decision nodes provide a link to the leaves. The following diagram shows the three types of nodes in a decision tree.



The information theory can provide more information on how decision trees work. Entropy and information gain are the building blocks of decision trees. An overview

of these fundamental concepts will improve our understanding of how decision trees are built.

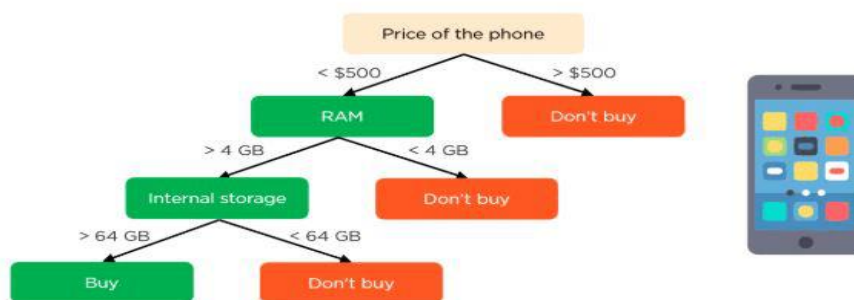
Entropy is a metric for calculating uncertainty. Information gain is a measure of how uncertainty in the target variable is reduced, given a set of independent variables.

The information gain concept involves using independent variables (features) to gain information about a target variable (class). The entropy of the target variable (Y) and the [conditional entropy](#) of Y (given X) are used to estimate the information gain. In this case, the conditional entropy is subtracted from the entropy of Y.

Information gain is used in the training of decision trees. It helps in reducing uncertainty in these trees. A high information gain means that a high degree of uncertainty (information entropy) has been removed. Entropy and information gain are important in splitting branches, which is an important activity in the construction of decision trees.

Let us take a simple example of how a decision tree works. Suppose we want to predict if a customer will purchase a mobile phone or not. The features of the phone form the basis of his decision. This analysis can be presented in a decision tree diagram.

The root node and decision nodes of the decision represent the features of the phone mentioned above. The leaf node represents the final output, either *buying* or *not buying*. The main features that determine the choice include the price, internal storage, and Random Access Memory (RAM). The decision tree will appear as follows



3.1.4 ATTRIBUTE MEASURES

1. survival -- the number of months patient survived (has survived, if patient is still alive). Because all the patients had their heart attacks at different times, it is possible that some patients have survived less than one year but they are still alive. Check the second variable to confirm this. Such patients cannot be used for the prediction task mentioned above.
2. still-alive -- a binary variable. 0=dead at end of survival period, 1 means still alive
3. age-at-heart-attack -- age in years when heart attack occurred
4. pericardial-effusion -- binary. Pericardial effusion is fluid around the heart. 0=no fluid, 1=fluid
5. fractional-shortening -- a measure of contractility around the heart lower numbers are increasingly abnormal
6. epss -- E-point septal separation, another measure of contractility. Larger numbers are increasingly abnormal.
7. lvdd -- left ventricular end-diastolic dimension. This is a measure of the size of the heart at end-diastole. Large hearts tend to be sick hearts.
8. wall-motion-score -- a measure of how the segments of the left ventricle is moving
9. wall-motion-index -- equals wall-motion-score divided by number of segments seen. Usually, 12-13 segments are seen in an echocardiogram. Use this variable INSTEAD of the wall motion score.
10. mult -- a derivate var which can be ignored
11. name -- the name of the patient (I have replaced them with "name")
12. group -- meaningless, ignore it
13. alive-at-1 -- Boolean-valued. Derived from the first two attributes. 0 means patient was either dead after 1 year or had been followed for less than 1 year. 1 means patient was alive at 1 year

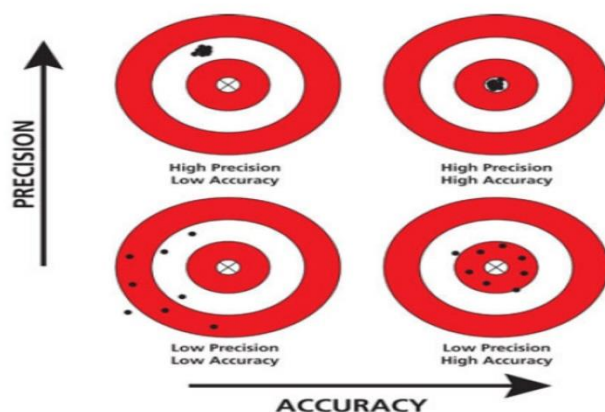
3.1.5 ACCURACY & PRECISION

What does accuracy mean?

If a measurement is accurate, it means that it agrees closely with the accepted standard for that measurement. For example, if we estimate a project's size to x and the actual size of the finished project is equal to or very close to x , then it is accurate, but it might not be precise. The closer a system's measurements to the accepted value, the more accurate the system is considered to be. Humans make errors all the time, but if you are using a project management software to help you scope, you will begin to have a more accurate project measurement and refined process.

What does precision mean?

A measurement that is precise means that it agrees with other measures of the same thing. In the sense of project scoping, let us take an estimation of workload as an example. If we estimate the size of several projects and they, in the end, are all close to or equal to what we predicted, then we can start to get a sense of the precision of our estimates. But first and foremost, each project needs to be as accurate as possible.

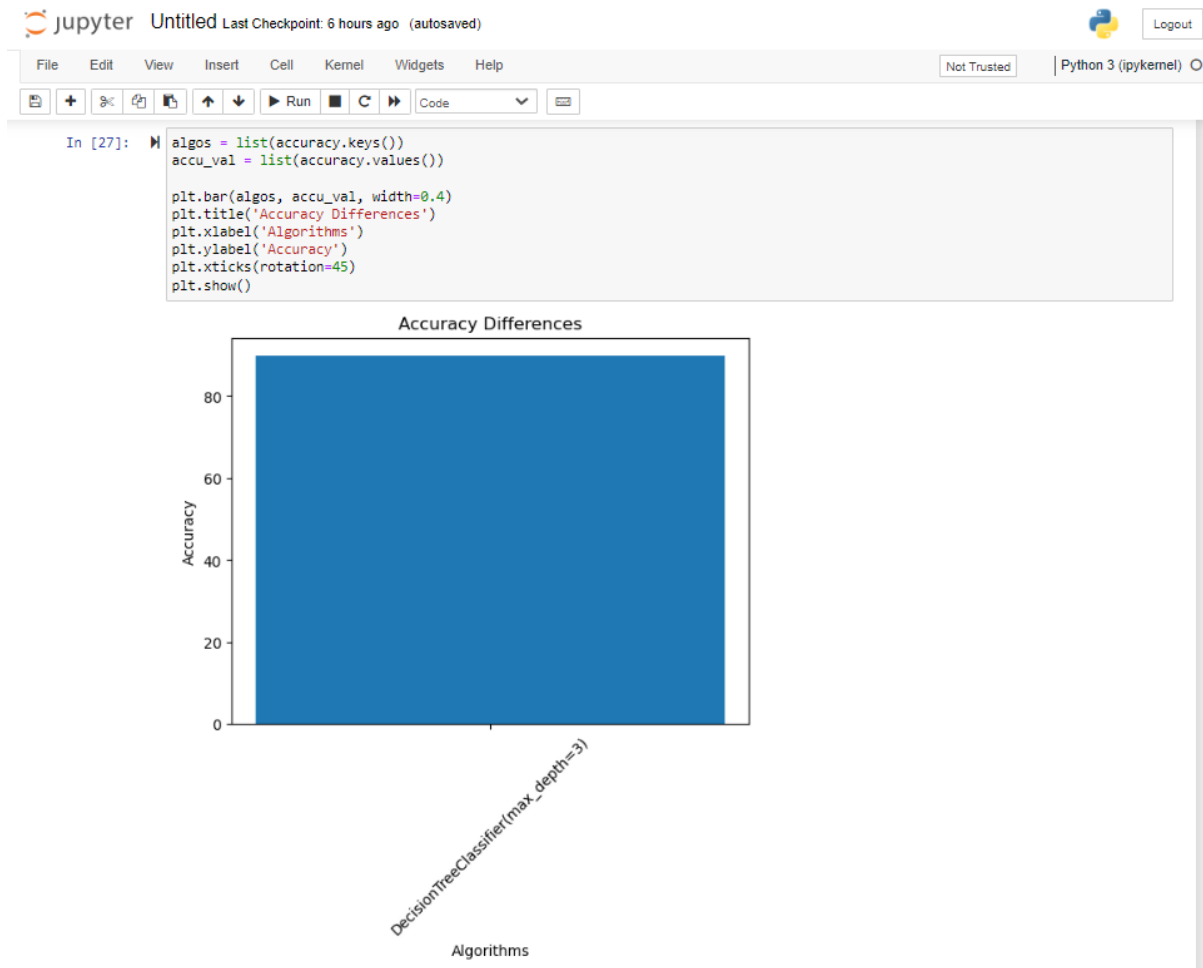


When scoping a project, you want to be as close to the actual workload as possible. Defining the scope means that you and your client are figuring out and documenting a list of specific project goals. That could be features, functionalities, deliverables, deadlines, and ultimately costs of the project. Project scope helps with resource planning and time management of the project. Accuracy and precision are used in the context of measurement, e.g. the size of a project and therefore are both helpful when defining the scoping.

Accuracy and precision are alike only in the fact that they both refer to the quality of measurement, but they are very different indicators of measurement. Accuracy is the degree of closeness to true value. Precision is the degree to which an instrument or process will repeat the same value. In other words, accuracy is the degree of veracity while precision is the degree of reproducibility.

4. RESULT AND DISCUSSION

4.1.1 Results



4.1.2 Discussion

The Decision tree we used are calculated using sklearn wrapper so can be trusted for the model performance. Our end goal was to have a Trained model that could beat the untuned benchmark which it did by a very fine margin. So, the solution described below is satisfactory to our initial expectations. We generated a final model with above list of tuned parameters.

5.IMPLEMENTATION

5.1.1 DECISION TREE CLASSIFIER

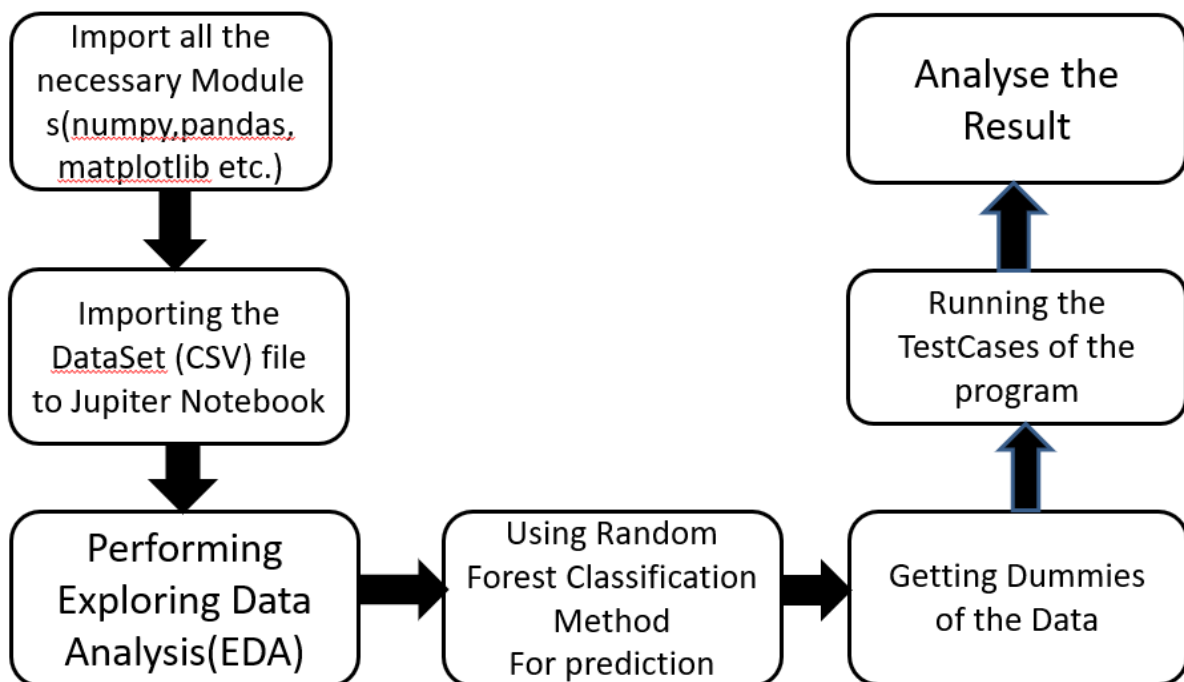
The first part of this project consisted of gathering and analysing the data of the Decision Tree. To achieve this, the programming language Python was used combination with the libraries of NumPy, pandas, matplotlib, seaborn and scikit learn classifier we imported decision tree classifier, used for scientific computing. This program is responsible of three major parts of the system:

1. Understanding the customer potential using supervised learning. This part of the system is also responsible for the data pre-processing and importing the python libraries, in the data set there is a column called Temperature which is int datatype model, we changed the temperature into category datatype as it is an ordinal datatype
2. After the pre-processing the model we consider various factors in our data set that is age, gender, annual income, and spending score through this data we visualize and fed into the decision tree classifier. This was done using seaborn. The latter was also used in the development stages for splitting data given some split algorithm such as test/train split method. The **Test/Train Split Method** is

one of the most popular methods to train the model. At the initial stage of the development, all the plotting and data visualization was made with the python plotting library matplotlib

3. Finally, the understanding the trained model potential is researched through the accuracy and precision. The better accuracy and precision model can be next implemented and showcase as decision tree.

5.1.2 System Architecture / Ideation Map



In this project, I will demonstrate how to build a model predicting recommender system using decision tree in Python using the following steps:

1. Data preparation
2. Exploratory Data Analysis
3. Analysing the correlation to numerical features
4. Encode categorical features
5. Using Machine Learning Decision Tree Algorithm for getting output

Hardware & Software Requirements:

Anaconda Navigator along with Python 3.8

Operating System

Windows 10, Core i5

4GB RAM

About Dataset:

This data was collected via a survey on Amazon Mechanical Turk. The survey describes different driving scenarios including the destination, current time, weather, passenger, etc., and then ask the person whether he will accept the coupon if he is the driver.

5.1.3 REFERENCES

Here we are providing the reference links which we have followed during the project work

- <https://www.kaggle.com/loganalive/echocardiogram-dataset-uci/report#missing-value-treatment-->
- <https://specialistdirectinc.com/telecardiology-articles/uncategorized/a-guide-to-understanding-echocardiogram-results/>
- <https://code.datasciencedojo.com/datasciencedojo/datasets/tree/master/Echocardiogram>
- <https://erp.bioscientifica.com/view/journals/echo/2/1/G9.xml>

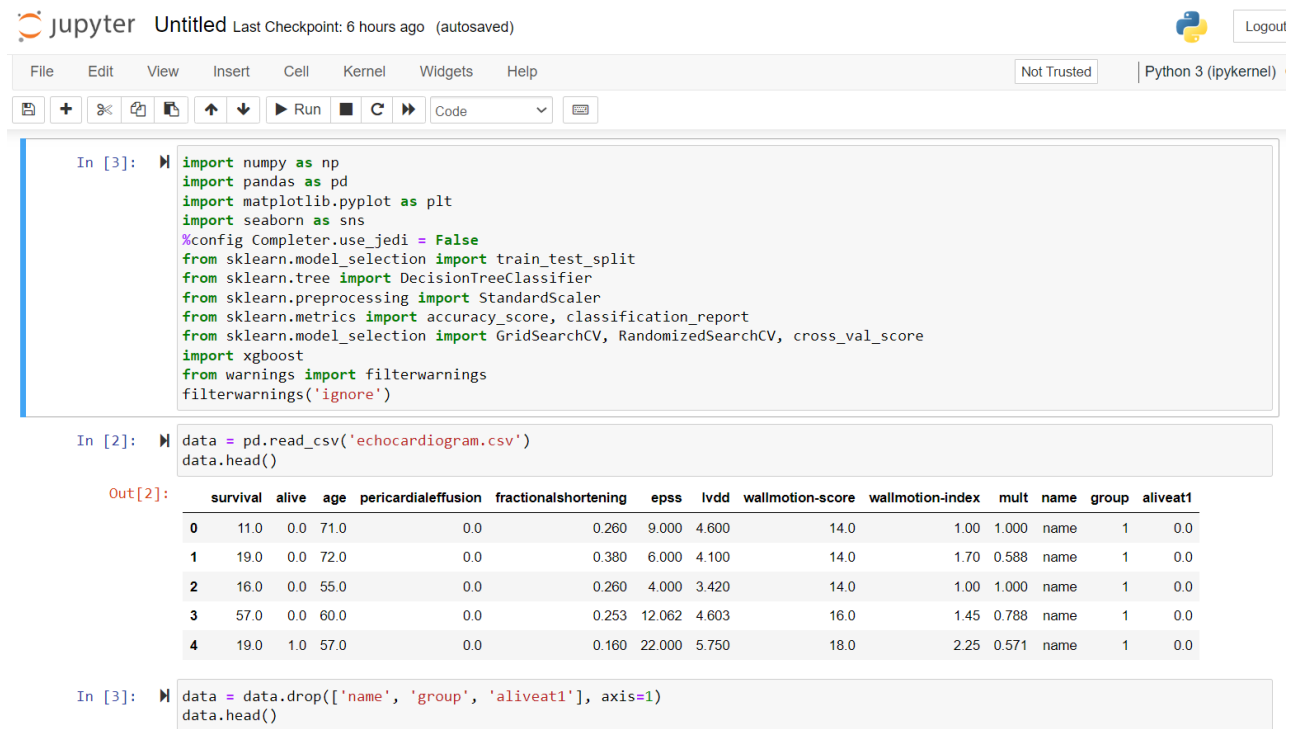
<https://www.nature.com/articles/s41746-018-0065-x>ed more than one model. Thus, more random forests, more the time.

6. CONCLUSION & FUTURE WORK:

The metrics we used are calculated using sklearn wrapper so can be trusted for the model performance. Our end goal was to have a tuned model that could beat the untuned benchmark which it did by a very fine margin. So, the solution described below is satisfactory to our initial expectations. We generated a final model with above list of tuned parameters.

Machine learning techniques are widely used data analysis methods in various business and industrial sectors. The main reason for that because ML can build predictive models to produce better predictions and achieve the desired level of accuracy, leading to better outcomes. Building the models is an easy and straight forward process. The main challenges in data analysis are data preparation and cleaning, the selection of appropriate models and attributes used in their implement at

SOURCE CODE SCREEN SHOTS



The image shows a Jupyter Notebook interface with the following components:

- Header:** Jupyter logo, "Untitled" filename, "Last Checkpoint: 6 hours ago (autosaved)", Python logo, and "Logout" button.
- Menu Bar:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help.
- Toolbar:** Not Trusted, Python 3 (ipykernel), and various icons for file operations and execution.
- Code Cells:**
 - In [3]:** Imports libraries: numpy, pandas, matplotlib.pyplot, seaborn, sklearn (train_test_split, DecisionTreeClassifier, StandardScaler, accuracy_score, GridSearchCV, RandomizedSearchCV, cross_val_score), xgboost, and warnings. It also configures Completer and ignores filterwarnings.
 - In [2]:** Loads 'echocardiogram.csv' and displays the first five rows of the data.
 - In [3]:** Drops 'name', 'group', and 'aliveat1' columns and displays the first five rows of the remaining data.
- Output:** A table showing the first five rows of the 'echocardiogram.csv' data.

	survival	alive	age	pericardialeffusion	fractionalshortening	epss	lvdd	wallmotion-score	wallmotion-index	mult	name	group	aliveat1
0	11.0	0.0	71.0	0.0	0.260	9.000	4.600	14.0	1.00	1.000	name	1	0.0
1	19.0	0.0	72.0	0.0	0.380	6.000	4.100	14.0	1.70	0.588	name	1	0.0
2	16.0	0.0	55.0	0.0	0.260	4.000	3.420	14.0	1.00	1.000	name	1	0.0
3	57.0	0.0	60.0	0.0	0.253	12.062	4.603	16.0	1.45	0.788	name	1	0.0
4	19.0	1.0	57.0	0.0	0.160	22.000	5.750	18.0	2.25	0.571	name	1	0.0

Run Code

```
In [3]: data = data.drop(['name', 'group', 'aliveat1'], axis=1)
data.head()
```

```
Out[3]:
```

	survival	alive	age	pericardialeffusion	fractionalshortening	epss	lvdd	wallmotion-score	wallmotion-index	mult
0	11.0	0.0	71.0	0.0	0.260	9.000	4.600	14.0	1.00	1.000
1	19.0	0.0	72.0	0.0	0.380	6.000	4.100	14.0	1.70	0.588
2	16.0	0.0	55.0	0.0	0.260	4.000	3.420	14.0	1.00	1.000
3	57.0	0.0	60.0	0.0	0.253	12.062	4.603	16.0	1.45	0.788
4	19.0	1.0	57.0	0.0	0.160	22.000	5.750	18.0	2.25	0.571

```
In [4]: data.isnull().sum()
```

```
Out[4]:
```

survival	3
alive	2
age	7
pericardialeffusion	1
fractionalshortening	9
epss	16
lvdd	12
wallmotion-score	5
wallmotion-index	3
mult	4
dtype: int64	

```
In [5]: features_with_null = [features for features in data.columns if data[features].isnull().sum()>0]
for feature in features_with_null:
    print(feature, ': ', round(data[feature].isnull().mean(), 4), '%')
```

Run Code

```
In [5]: features_with_null = [features for features in data.columns if data[features].isnull().sum()>0]
for feature in features_with_null:
    print(feature, ': ', round(data[feature].isnull().mean(), 4), '%')
```

```
survival : 0.0226 %
alive : 0.015 %
age : 0.0526 %
pericardialeffusion : 0.0075 %
fractionalshortening : 0.0677 %
epss : 0.1203 %
lvdd : 0.0902 %
wallmotion-score : 0.0376 %
wallmotion-index : 0.0226 %
mult : 0.0301 %
```

```
In [6]: for feature in features_with_null:
        print(feature, ': ', data[feature].unique())
```

```
survival : [1.10e+01 1.90e+01 1.60e+01 5.70e+01 2.60e+01 1.30e+01 5.00e+01 2.50e+01
1.00e+01 5.20e+01 4.40e+01 5.00e-01 2.40e+01 2.20e+01 1.00e+00 7.50e-01
5.00e+00 4.80e+01 2.90e+01 2.50e-01 3.60e+01 3.00e+00 2.70e+01 3.50e+01
3.10e+01 3.20e+01 4.00e+01 4.60e+01 2.00e+00 3.70e+01 1.95e+01 2.00e+01
nan 7.00e+00 1.20e+01 4.50e+01 5.30e+01 3.80e+01 9.00e+00 4.90e+01
4.70e+01 4.10e+01 3.30e+01 1.50e+01 3.00e-02 2.30e+01 3.40e+01 2.10e+01
5.50e+01 4.00e+00 1.25e+00 2.80e+01 1.70e+01 7.50e+00]
alive : [ 0.  1. nan]
age : [71.  72.  55.  60.  57.  68.  62.  46.  54.  77.
73.  69.  62.529 66.  85.  64.  35.  75.  65.  52.
nan 47.  63.  61.  80.  70.  79.  56.  67.  81.
59.  58.  51.  50.  78.  86.  74.  53.  48.  ]
pericardialeffusion : [ 0.  1. 77. nan]
fractionalshortening : [0.26  0.38  0.253 0.16  0.23  0.33  0.34  0.14  0.13  0.45  0.15  0.12
0.25  0.07  0.09  0.22  0.18  0.17  0.19  0.3  nan 0.21  0.4  0.61
0.06  0.51  0.41  0.35  0.27  0.44  0.03  0.04  0.24  0.01  0.29  0.1
0.187 0.11  0.36  0.225 0.217 0.2  0.05  0.28  0.155 0.344 0.272 0.5
0.258 0.228 0.036 0.43 ]
epss : [ 9.  6.  4. 12.062 22.  5.  31.  8.  0. 13.
16. 10. 23. 12.063 11. 20. 17. 15. 12. 19.
12.733 5.9  7.  nan 4.2 17.2 5.12 13.1 23.6 5.4
9.3  4.7 17.5 21.3 14. 14.8 9.4 24.6 15.6 18.6
9.8 11.9 16.4 10.3 13.2 11.4 9.7 8.8 16.1 12.2
10.2 7.5 30.1 17.9 21.7 19.4 7.1 16.8 25. 19.2
5.5 8.7 11.3 6.6 9.1 16.5 5.6 4.8 8.5 28.9
11.8 6.9 14.3 40.  7.6 12.1 13.6 9.2 28.6 19.1
6.8 25.5 12.9 ]
lvdd : [4.6  4.1  3.42 4.603 5.75 4.31 5.43 5.25 5.09 4.49 4.23 3.6
4.  3.73 5.8  4.29 4.65 5.2  5.819 5.4  5.39 5.46 6.06 3.48
3.85 4.17  nan 4.16 5.05 5.32 3.1  4.07 5.31 3.88 4.36 3.63
4.27 3.59 3.96 6.29 5.  5.26 3.49 5.65 6.15 4.57 4.37 5.3
4.41 5.15 6.78 5.02 4.96 4.68 4.75 5.57 5.78 5.62 4.72 5.95
4.54 4.85 4.77 4.58 6.74 4.48 4.44 6.21 4.14 4.56 5.16 4.04
5.36 4.02 3.87 5.47 6.73 4.69 4.55 4.87 3.52 5.49 4.12 6.23
4.42 3.92 4.38 4.06 6.63 4.79 5.86 2.32 5.04 3.66 4.51 ]
wallmotion-score : [14. 16. 18. 12. 22.5 15.5 11.67 24.  8.  27. 19.5 13.83
7.5 10.  2.  nan 6. 13.  5. 21.5 15. 11. 22. 17.5
9. 17. 23. 39. 12.33 10.5 16.67 17.83 5.5 13.67 16.5 21.
28. 11.5 13.5 12.67 15.67 12.5 26.08 18.16 19. 14.5 ]
wallmotion-index : [1.  1.7  1.45 2.25 1.875 1.14 1.19 1.8  2.33 1.64 2.  1.333
1.625 1.38 1.5  1.11 1.667 1.56 1.17 3.  2.15 1.67 1.222 1.3
1.25 1.31 2.3  nan 1.08 1.37 1.167 1.05 1.39 1.18 2.5 1.1
1.367 2.18 2.39 1.15 2.1  1.09 1.36 1.41 1.83 1.04 1.27 1.06
1.42 2.01 1.51 1.23 1.4  2.2  1.95 1.2  1.375 1.73 1.21 1.409]
mult : [1.  0.588 0.788 0.571 0.857 1.003 0.93  0.714 0.358 0.784 0.429 0.71
0.36 0.57 0.64 0.43 2.  nan 0.14 0.357 0.786 0.428 0.928 0.812
0.642 0.785 0.28 0.643]
```



```
In [7]: data = data.dropna(subset=['alive'])
        data['alive'].isnull().sum()
```

Out[7]: 0

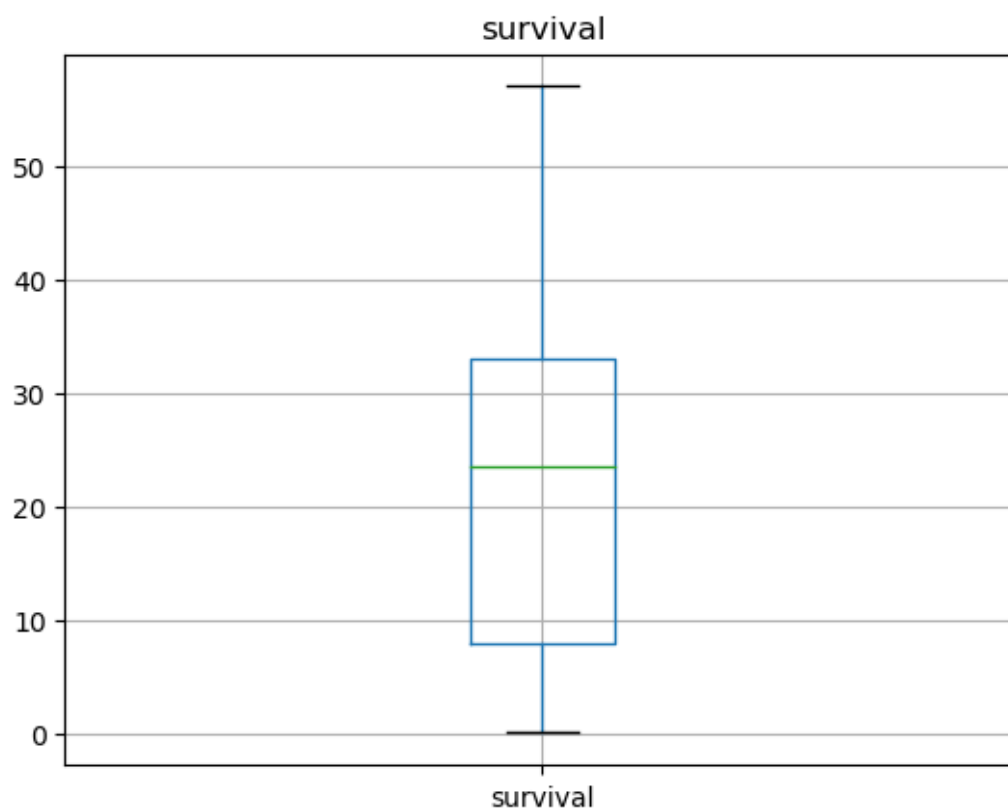
```
In [8]: discrete_features = ['pericardialeffusion']
        continuous_features = data.drop(['pericardialeffusion', 'alive'], 1).columns
        label = ['alive']

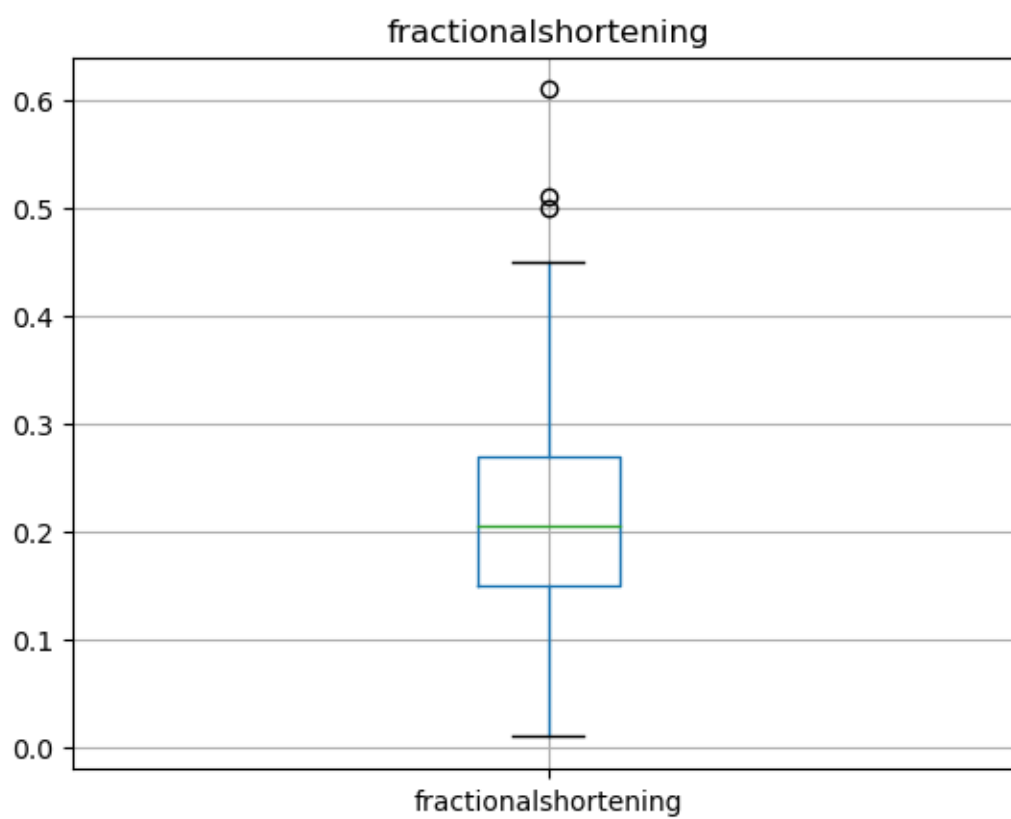
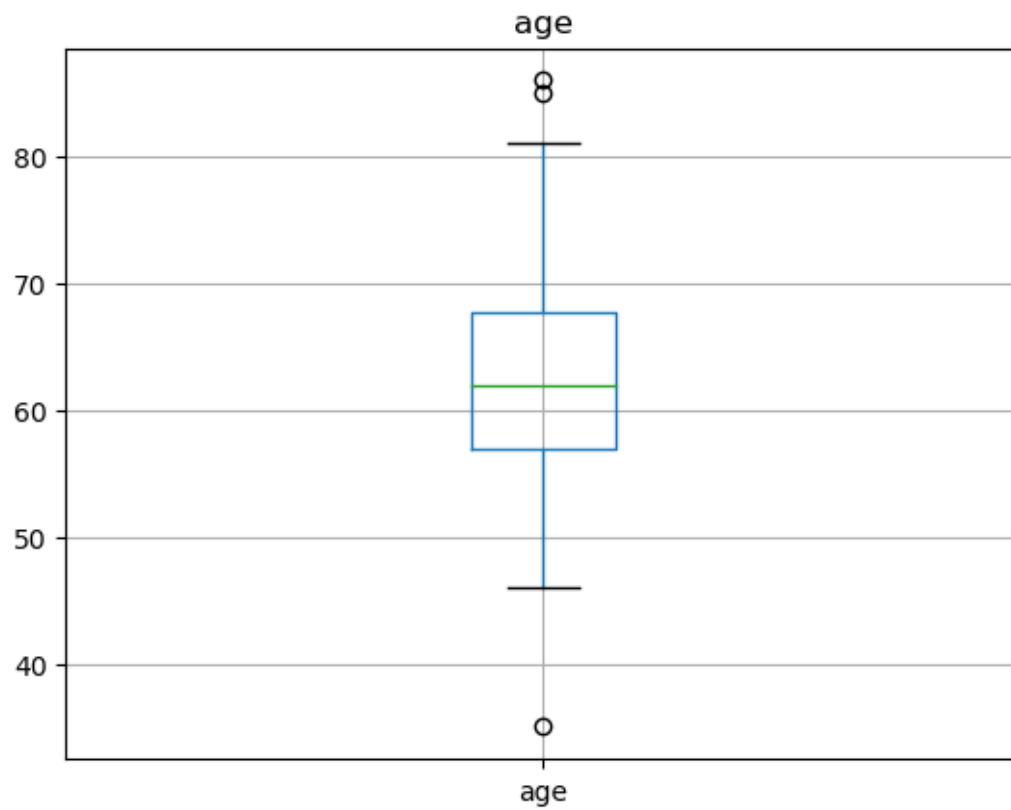
        print(continuous_features)

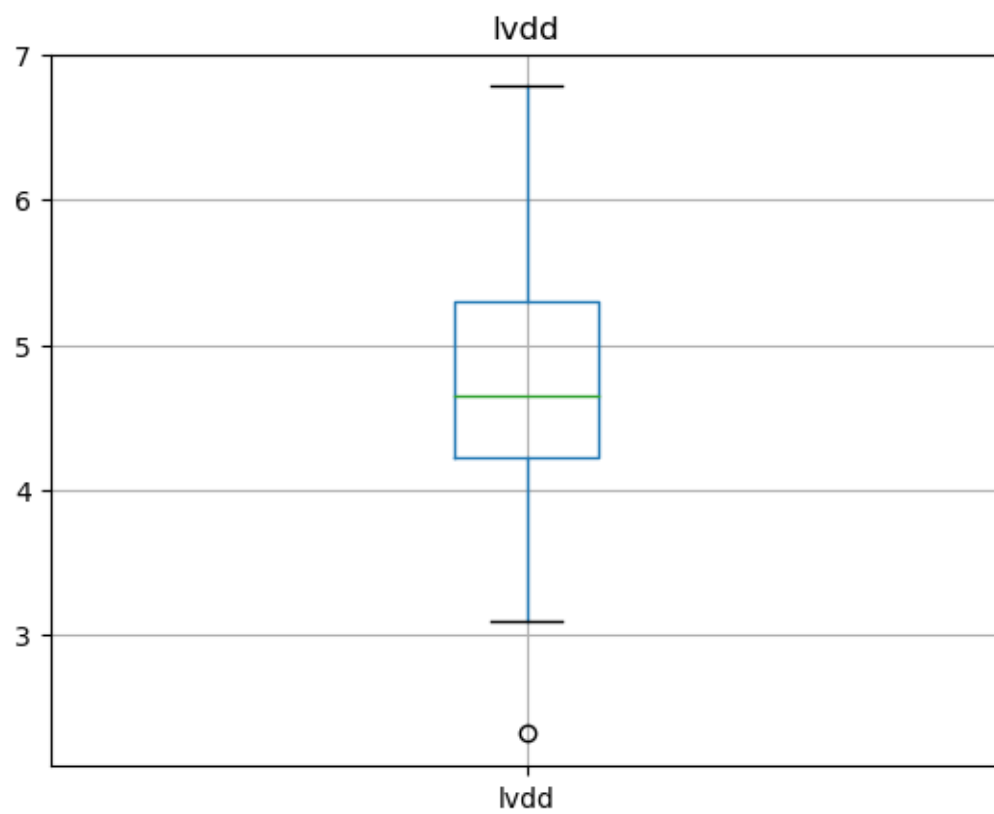
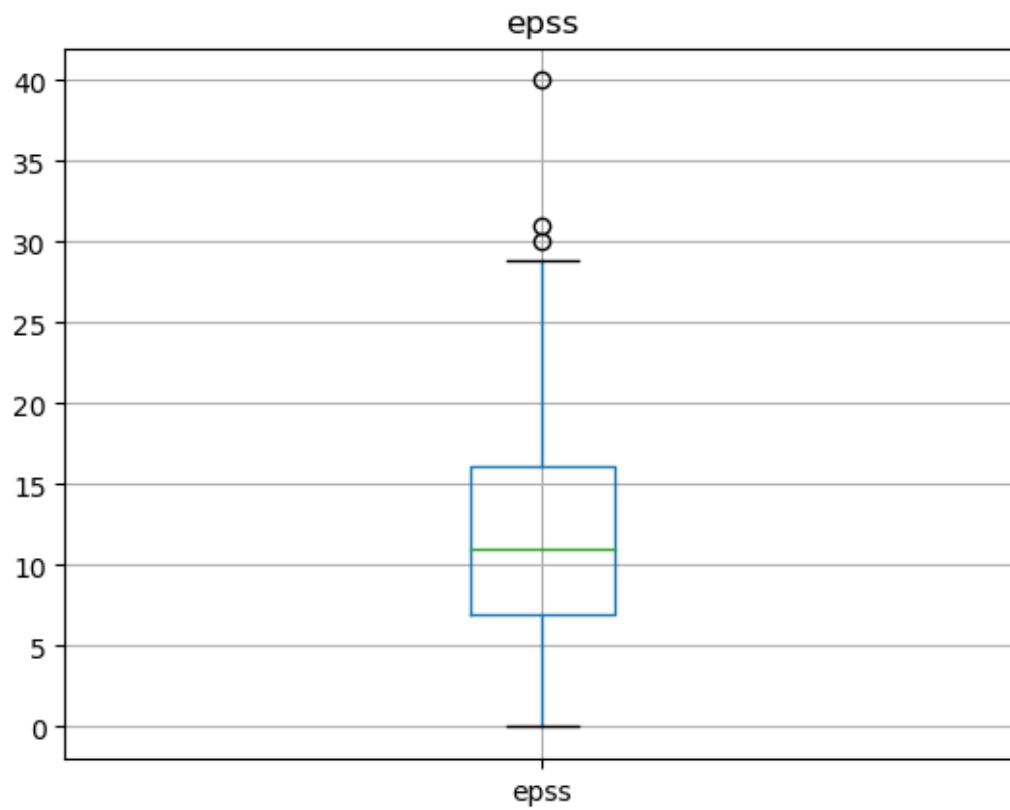
Index(['survival', 'age', 'fractionalshortening', 'epss', 'lvdd',
       'wallmotion-score', 'wallmotion-index', 'mult'],
      dtype='object')
```

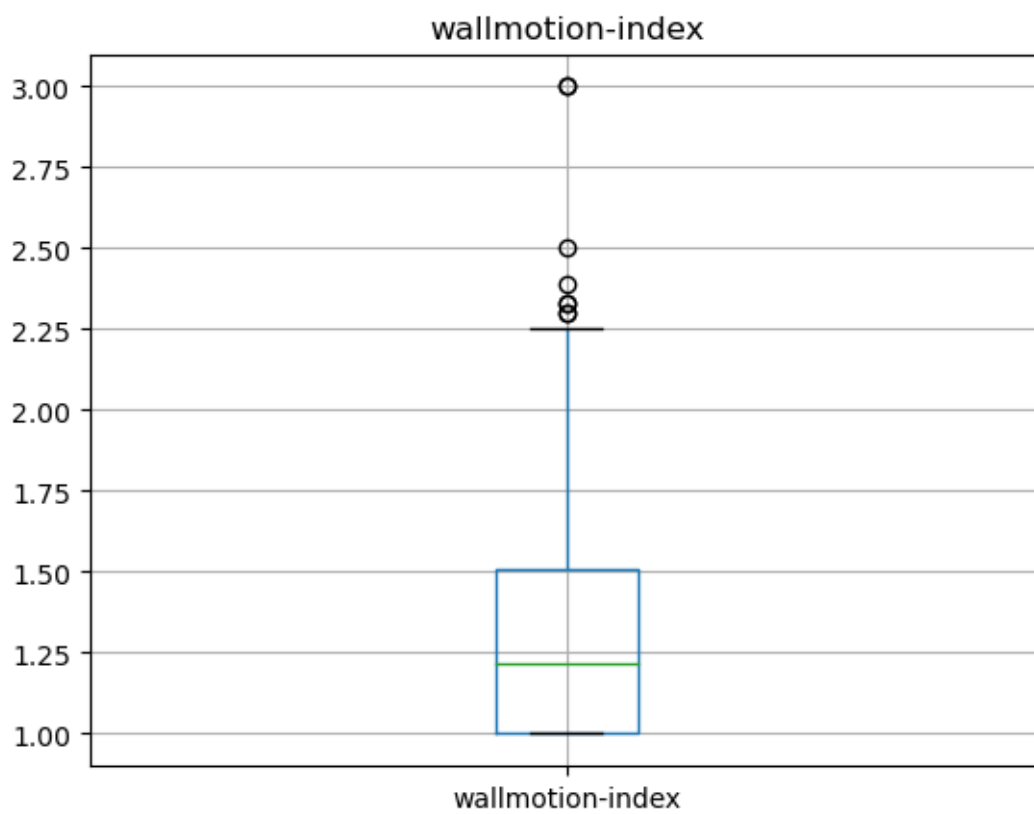
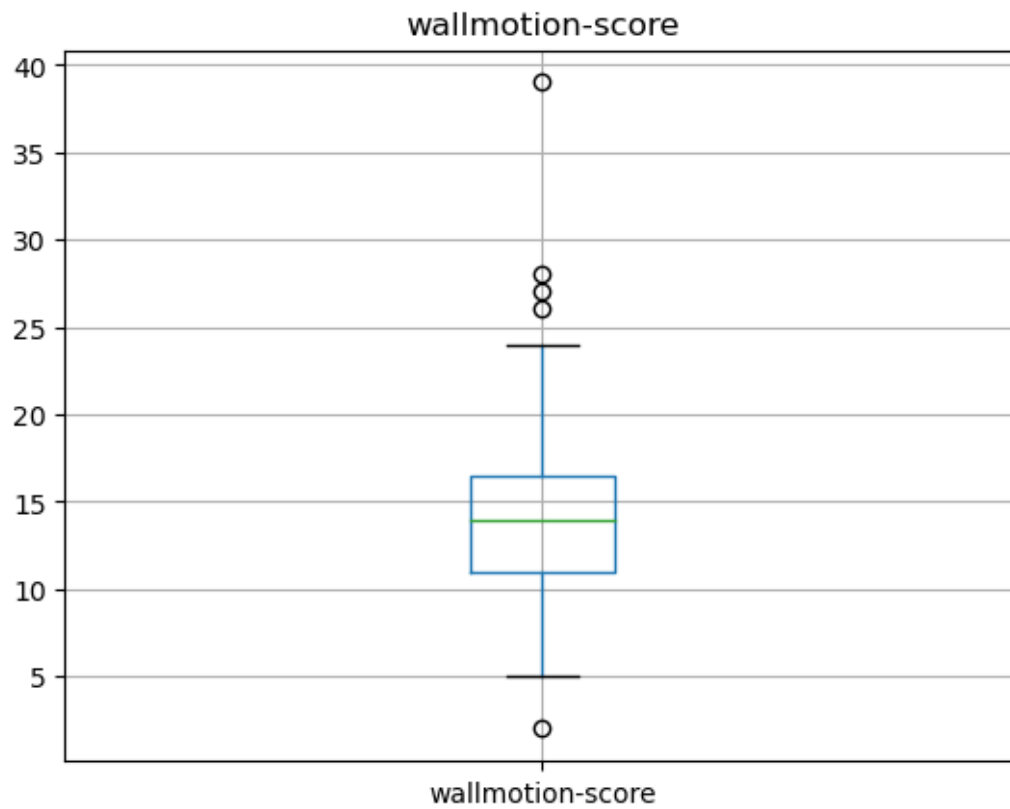
```
In [9]: for feature in discrete_features:
        data[feature] = data[feature].fillna(data[feature].mode()[0])
```

```
In [10]: for feature in continuous_features:
        data.boxplot(feature)
        plt.title(feature)
        plt.show()
```











```
In [11]: features_with_outliers = ['wallmotion-score', 'wallmotion-index', 'mult']
```

```
In [12]: for feature in continuous_features:
          if feature in features_with_outliers:
              data[feature].fillna(data[feature].median(), inplace=True)
          else:
              data[feature].fillna(data[feature].mean(), inplace=True)
```

```
In [13]: from sklearn.neighbors import LocalOutlierFactor
          lof = LocalOutlierFactor()
          outliers_rows = lof.fit_predict(data)
```

```
In [14]: mask = outliers_rows != -1
```

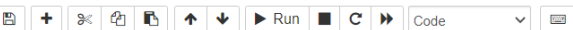
```
In [15]: data.isnull().sum()
```

```
Out[15]: survival      0
         alive         0
         age           0
         pericardialeffusion  0
         fractionalshortening  0
         epss           0
         lvdd           0
         wallmotion-score  0
         wallmotion-index  0
         mult           0
         dtype: int64
```

```
In [16]: data = data[mask]
```

```
In [17]: data1 = pd.get_dummies(data, columns = discrete_features, drop_first = True)
          scaler = StandardScaler()
          data1[continuous_features] = scaler.fit_transform(data1[continuous_features])
```

```
In [18]: data1.head()
```



```
In [18]: data1.head()
```

```
Out[18]:
```

	survival	alive	age	fractionalshortening	epss	lvdd	wallmotion-score	wallmotion-index	mult	pericardialeffusion_1.0
0	-0.729968	0.0	1.039461	0.419602	-0.470229	-0.190540	-0.050728	-0.856557	1.156755	0
1	-0.221716	0.0	1.170502	1.580619	-0.940477	-0.839869	-0.050728	0.769979	-0.939966	0
2	-0.412311	0.0	-1.057203	0.419602	-1.253976	-1.722956	-0.050728	-0.856557	1.156755	0
3	2.192480	0.0	-0.401996	0.351875	0.009737	-0.186644	0.392359	0.189073	0.077860	0
4	-0.221716	1.0	-0.795120	-0.547913	1.567512	1.302915	0.835445	2.047971	-1.026482	0

```
In [19]: X = data1.drop(['alive'], 1)
          y = data1['alive']
```

```
In [20]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
          X_train.shape, X_test.shape
```

```
Out[20]: ((88, 9), (39, 9))
```

```
In [21]: accuracy = {}
```

```
In [22]: model2 = DecisionTreeClassifier(max_depth=3)
          model2.fit(X_train, y_train)
          y_pred2 = model2.predict(X_test)
          print(accuracy_score(y_test, y_pred2))
          accuracy[str(model2)] = accuracy_score(y_test, y_pred2)*100

0.8974358974358975
```

```
In [23]: accuracy
```

```
Out[23]: {'DecisionTreeClassifier(max_depth=3)': 89.74358974358975}
```




```
In [24]: param_combinations = {
        'learning_rate': np.arange(0.05, 0.4, 0.05),
        'max_depth': np.arange(3, 10),
        'min_child_weight': np.arange(1, 7, 2),
        'gamma': np.arange(0.0, 0.5, 0.1),
    }

    XGB = xgboost.XGBClassifier()
    perfect_params = RandomizedSearchCV(XGB, param_distributions=param_combinations, n_iter=6, n_jobs=-1, scoring='roc_auc')

    perfect_params.fit(X, y)
    perfect_params.best_params_
```

```
Out[24]: {'min_child_weight': 1, 'max_depth': 9, 'learning_rate': 0.05, 'gamma': 0.0}
```

```
In [25]: model5 = xgboost.XGBClassifier(min_child_weight=3, max_depth=8, learning_rate=0.05, gamma=0.0)
        score = cross_val_score(model5, X, y, cv=10)
```

```
In [26]: print(score)
        print('Mean: ', score.mean())

[0.92307692 0.92307692 1.          0.76923077 0.84615385 1.
 1.          0.83333333 0.83333333 0.58333333]
Mean: 0.8711538461538464
```