T.C.

ALTINBAS UNIVERSITY

Institute of Graduate Studies

Electrical and Computer Engineering

OPTIMIZATION OF HEART DISEASE PREDICTION BY IMPROVING MACHINE LEARNING RESULTS WITHOUT NEED MORE DATA

Mohammed Ghassan ADNAN

Master of Science

Supervisor

Asst. Prof. Dr. Abdullahi Abdu IBRAHIM

Istanbul, 2021

OPTIMIZATION OF HEART DISEASE PREDICTION BY IMPROVING MACHINE LEARNING RESULTS WITHOUT NEED MORE DATA

by

Mohammed Ghassan ADNAN

Electrical and Computer Engineering

Submitted to the Graduate School of Science and Engineering

in partial fulfillment of the requirements for the degree of

Master of Science

ALTINBAŞ UNIVERSITY

2021

The thesis titled "OPTIMIZATION OF HEART DISEASE PREDICTION BY IMPROVING MACHINE LEARNING RESULTS WITHOUT NEEDING MORE DATA" prepared and presented by "Mohammed Ghassan ADNAN" was accepted as a Master of Science Thesis in Electrical and Computer Engineering.

_____

Asst. Prof. Dr. Abdullahi Abdu IBRAHIM

Supervisor

Thesis Defense Jury Members:

Asst. Prof. Dr. Abdullahi Abdu IBRAHIM	School of Engineering and Natural Sciences, Altinbas University	_____
Asst. Prof. Dr. Sefer KURANZ	School of Engineering and Natural Sciences, Altinbas University	_____
Asst. Prof. Dr. Zeynep ALTAN	School of Engineering and Architecture, Beykent University	_____

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Approval Date of Institute of Graduate Studies: ___/____/____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Mohammed Ghassan ADNAN

DEDICATION

I dedicate this work to my parent who stood by me every step of the way and supported me to the fullest and my supervisor Asst. Prof. Dr.Abdullahi Abdu IBRAHIM's guidance and insight made this work possible.

ACKNOWLEDGEMENTS

I would like to thank very much my supervisor, Assistant Professor Dr. Abdullahi Abdu IBRAHIM for his assistance throughout the research period and for answering many questions and he was the best help for me throughout the study period. Without his time and help, this research would not have been completed and I am very grateful to him.

ABSTRACT

OPTIMIZATION OF HEART DISEASE PREDICTION BY IMPROVING MACHINE LEARNING RESULTS WITHOUT NEED MORE DATA

Mohammed Ghassan ADNAN,

M.Sc, Electrical and Computer Engineering, Altınbaş University,

Supervisor: Asst. Prof. Dr. Abdullahi Abdu IBRAHIM

Date: April/2021

Pages: 66

It is known that cardiologists can accurately predict 80% of heart disease, but the missing 20% is where AI can help, the problem of achieving high accuracy is one of the challenges that are facing any developer in the heart disease prediction field by using machine learning techniques, but sometimes the kind of data they want can be quite costly or it might be hard to come by, so this research spotlight on one of the methods that try to enhance machine learning algorithm without the need of getting more data by using Hyperparameters optimization in Machine Learning technology to classify the ECG data that got it from the physionet2017 dataset with three disease class (ARR, CHR, NSR), the proposed method increased in classification accuracy (88.3%) after using Hyperparameters optimization in Machine Learning without the need of getting new data.

Keywords: Hyperparameters, Machine Learning, Feature Extraction, Physionet, Coarse Tree, Classification.

TABLE OF CONTENTS

Pages

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

LIST OF CHARTS

Pages

LIST OF ABBREVIATIONS

AI	:	Artificial Intelligence
IoT	:	Internet of Things
DL	:	Deep Learning
ML	:	Machine Learning
NN	:	Neural Network
ECG	:	Electrocardiogram
SVM	:	Support Vector Machine
KNN	:	K-Nearest Neighbor
BO	:	Bayesian Optimization
ARR	:	Abnormal Arrhythmia
CHR	:	Congestive Heart Rate
NSR	:	Normal Sinus Rhythm
ROC	:	Receiver Operating Characteristic
TPR	:	True Positive Rates
TPR	:	True Positive Rates
FNR	:	False Negative Rates

1. INTRODUCTION

This chapter provides a detailed background to predicting heart disease through ECG signals, problems with accuracy and lack of data, the motivation behind the method we chose to use in our research, the structure and contribution of our method, and finally we explain the organization of the remainder of the thesis.

1.1 BACKGROUND

Currently, huge funds are being invested in the study and development of artificial intelligence, especially in the fields of medicine and human biology, healthcare, artificial intelligence with all its implications (ML, DL, and NN), Its practical applications are becoming more and more in creating systems capable of processing large and complex data, as well as in the automatic diagnosis of diseases, development of medical properties, as well as in creating and developing systems for analyzing various types of vital signs. Artificial intelligence systems and algorithms have facilitated a lot of researchers' work, as data are analyzed on workdays as long as computing systems unravel their mysteries in a few seconds, as recent years have seen the emergence of research and studies that previously cost huge money and take many years. Through this thesis, the method of predicting heart disease using ECG signal, machine learning, and how to improve the accuracy of results were studied.

ECG signals are records of the vital electrical activities of the heart system, and every part of these signals is altered from regular when heart disease happens [1]. Individuals with similar heart conditions have nearly all ECG signals with similar features. If the ECG signal with nonspecific features has a similar morphological pattern to the ECG signal with arrhythmias, then it can be concluded that this unknown signal has the same arrhythmia. In such cases, heart

disease can be detected by analyzing the pattern of the ECG signals .fast detection of abnormal heart conditions can stop sudden cardiac death and other severe illnesses caused by heart illness. Several scientists must describe the results in investigating ECG signals and identifying their abnormalities [2]. To identify such conditions in the laboratory, continued monitoring of ECG signals is essential for all patients. This method is tedious and time-consuming. Automation of the analysis of the ECG signal based on computer programming is a good system for diagnosing heart disease.

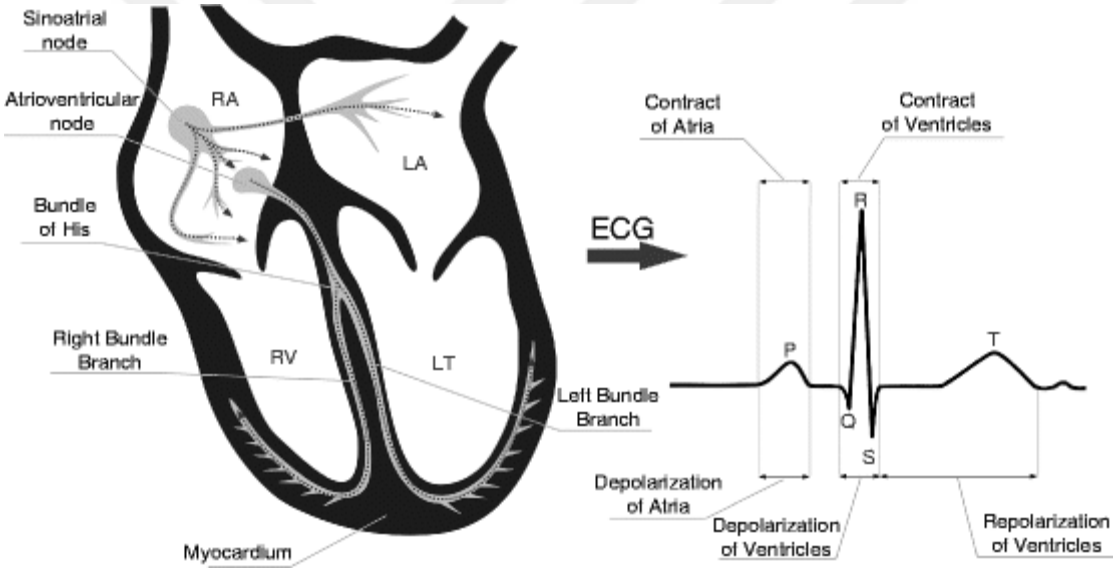


Figure 1.1: ECG signal waveform forms

Source: Adapted from [2]

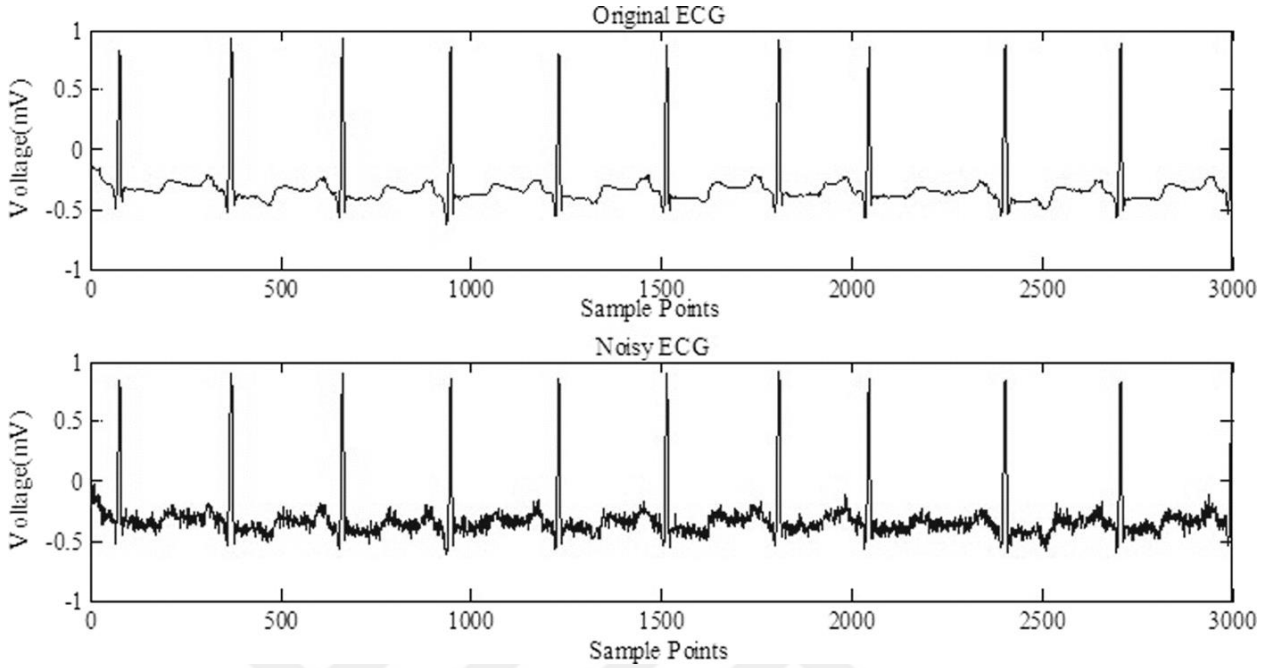


Figure 1.2: ECG signal in the time domain

Source: Adapted from [1]

The high ability of artificial intelligence algorithms to process data and predict appropriate diseases and treatments has made them great sources and references for researchers, as an effective prediction of critical diseases such as cancer and cardiovascular disease can greatly expand a person's survival rate and increase its chances of recovery Cardiovascular diseases, during the past years, Researchers have reached good results in diagnosing and treating diseases of the heart and blood vessels, A diagnosis of myocardial infarction and other serious heart diseases could save hundreds of thousands of people, so the study of artificial intelligence in this sensitive medical significant field, and from what scientists have invented in this regard:

1- Niels Stroudthof and Klas Strudthoff, two German engineers, developing a neural network (NN) that could expose signs of myocardial infarction. They trained an algorithm to works immediately on the (ECG) data and predict the probability of a seizure[3].

2- A team from the Institute of Bioengineering at the University of Auckland has produced a Virtual Heart (3D) that can give a massive penetration in treating more common heart rhythm disorders such as Atrial Fibrillation (AF) [4].

1.2 PROBLEMS

One of the challenges that any developer will face in predicting heart disease using machine learning techniques is the problem of reaching high accuracy. Most developers will follow the default advice to "get extra data" but what if they can't or the type of data they want it can be very expensive. It can be difficult to obtain. And sometimes they get the data, they retrain their model, and they notice that the model isn't improving.

1.3 MOTIVATION

Optimization methods use in machine learning algorithms to solve problems seeking results with high accuracy.

The optimization method has many hyperparameters set before the learning process and affects exactly how the ML algorithms fit the models with the data [2].

With this method, data will be archived (optimization or tuning of hyperparameters) that perform better over time by tuning hyperparameter values.

This method (hyperparameter optimization or tuning) plays an important and effective role by raising the efficiency of machine learning algorithms, thus obtaining higher prediction accuracy.

During this research, more than one model was trained With different combinations of values, the compared performance of models to determine the best model that gives higher prediction accuracy, by applying most of the machine learning algorithms to the data without improving the features, and determining the algorithm with the best accuracy.

With the optimization of hyperparameters(Bayesian) (Grid Search and Random Search) and accurate difference comparison.

The control has been updated by training the model, which in turn is affected by the hyperparameters and thus, developers can find the correct hyperparameters, and the model learns the most optimal weights that it can obtain by using a certain training algorithm and data.

Therefore, this research had two objectives:

The first was how to get high accuracy in classifying and predicting heart disease within machine learning.

• The second was how we improve the accuracy of the algorithm using hyperparameter optimization and the same previously trained data without having to add new data to the dataset, which could cost us time, effort, and cost to collect it.

1.4 CONTRIBUTION

The main contributions to our thesis conclude as follows by conducting training in machine learning algorithms on a dataset of ECG signals and attempting to provide high performance in prediction accuracy by improving machine learning outcomes using the optimization method without requiring more data.

1.5 THESIS STRUCTURE

The remaining thesis classes are organized as follows: In the second chapter, the history and types of machine learning and the machine learning algorithms used to achieve our goal are presented, as well as methods for improving machine learning and all previous studies. Chapter three is our proposed approach and methodology. Our results and analysis are discussed in Chapter four. Finally, we concluded our work and put our future work in Chapter Five

2. THEORY

Through this chapter, a brief background is presented on the history of ML, its types, the ML algorithms used in this thesis, and all the methods used to optimize machine learning models, the reasons and motivations for choosing the method adopted in this thesis are we discussed methods of hyperparameter optimization and its advantages and disadvantages of optimization methods for parameters. Superlative. A summary of the literature review of our study was also presented.

2.1 HISTORY OF MACHINE LEARNING AND TYPES

ML represents the major part of AI that is entrusted with developing and designing algorithms that make computer systems a tool used in machine learning, taking advantage of the features provided by data[5].

Generally speaking, machine learning consists of two parts. The first represents inductive learning and the second represents deductive education for big data.

The term "ML" was first used in 1959 by Arthur Samuel, an American who worked for IBM. In the 1960s, machine learning was mostly used to categorize patterns.

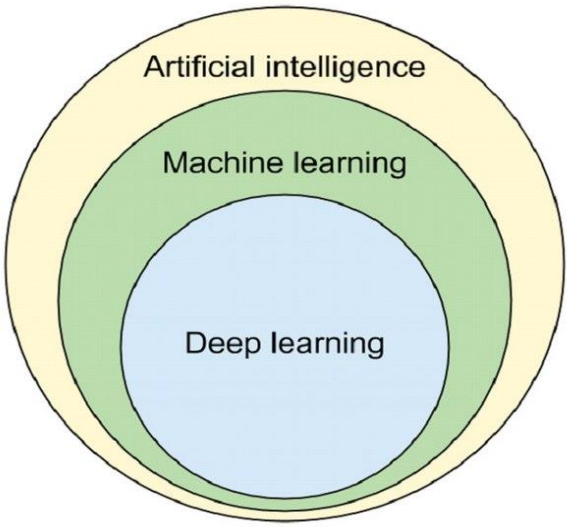


Figure 2.1: Artificial Intelligence

Source: Adapted from [5]

More recently ML is defined by Stanford University as "the science of making computers operate without being explicitly programmed." The ML is now responsible for some of the most important advances in technology, such as the new industry for self-driving vehicles. ML has driven a new set of ideas and knowledge, including supervised and unsupervised learning, novel robotics algorithms, IoT, analysis tools, chatbots, and more.

One of the most significant characteristics of ML algorithms is that they work with advanced computing techniques to increase efficiency and improve scalability. There are several areas in which modern machine learning techniques are used, including predicting disease outbreaks, weather, and economics.

There are several types of ML algorithms that are classified according to the algorithm approach, as well as the type of input and production and the kind of task or problem to be

solved. They are categorized into three main types (supervised, unsupervised, semi-supervised, and reinforced).

2.1.1 SUPERVISED LEARNING

Supervised learning can be considered one of the most important models used in ML[6]. The most important characteristic of this method is the use of relevant data between input and output, which means that the data is classified so that this method helps in clearly disclosing the features to be categorized[7]. After the algorithm determines the relationship between input and output, it can predict the results[8].

When we get more data, the ML algorithm provides more accurate output.

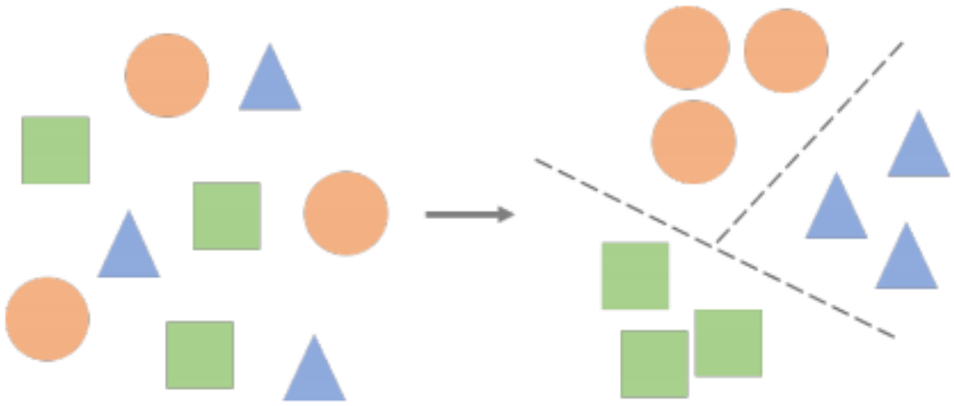


Figure 2.2: Supervised Learning

Source: Adapted from[7]

2.1.2 UNSUPERVISED LEARNING

With the unsupervised learning algorithm, the data are unclassified to different categories, and not designations[9]. A model can learn from data by finding implicit patterns. It defines data according to their similar densities, structures, segments, and other similar features.

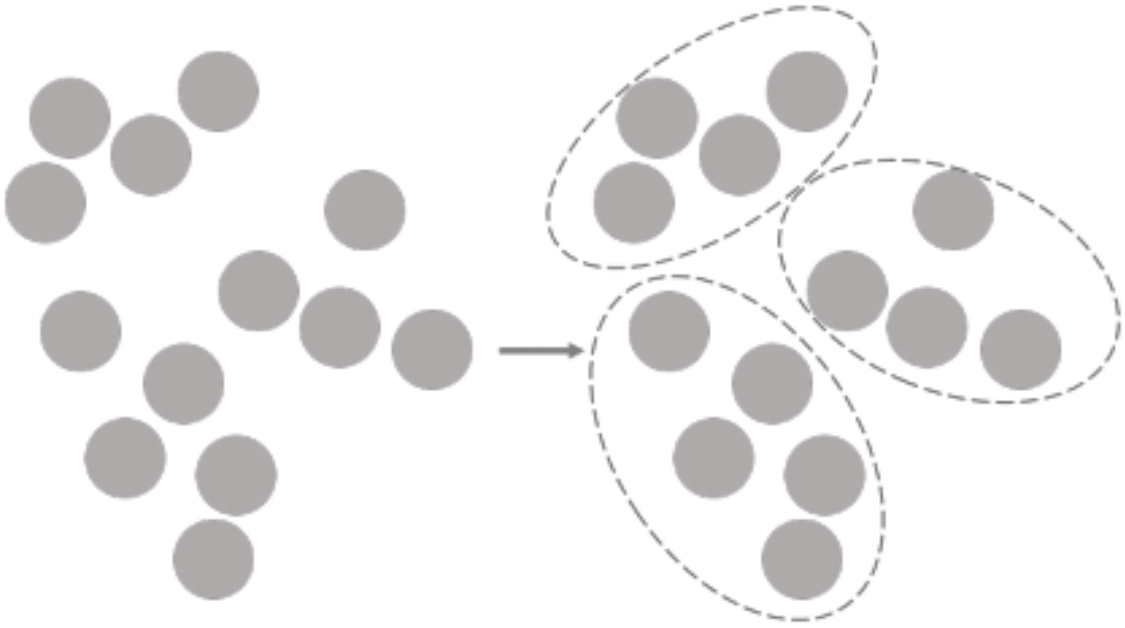


Figure 2.3: Unsupervised Learning

Source: Adapted from[7]

2.1.3 REINFORCEMENT LEARNING

Reinforcement learning covers more parts of artificial intelligence that allow machines to react with their dynamic environment to arrive at their goals[10]. Using this, machines and software agents can evaluate ideal behavior in a given context. With the help of these rewards

notes, agents can learn and improve the behavior in the long run. (See fig 4)

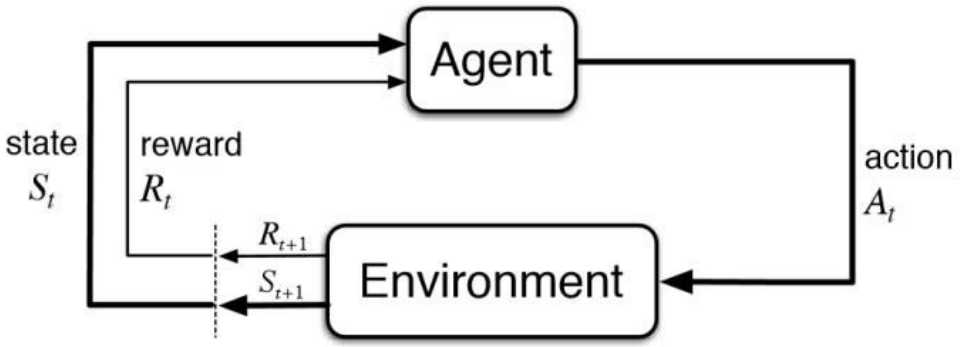


Figure 2.4: Reinforcement Learning

Source: Adapted from[7]

2.2 MACHINE LEARNING ALGORITHMS

Many algorithms are working within the machine learning group, but we briefly explain the most important algorithms that were used in this thesis.

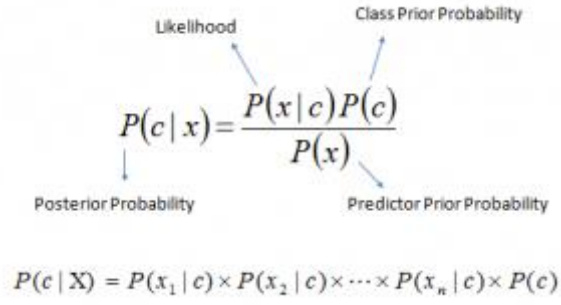
2.2.1 DECISION TREES

It is a kind of supervised ML algorithm, and in the training data the corresponding inputs and outputs are specified, the data is divided according to a specified parameter, and the decision tree contains two entities, namely, decision nodes, papers, and data. They are divided according to the decision node and the papers represent the decisions or results[11].

2.2.2 NAÏVE BAYES

This algorithm is considered one of the classification techniques that were built according to Bayes' theory, and its work is to assume the presence of characteristics in a particular

classification that has nothing to do with the existence of another feature[12]. This algorithm is usually used with big data as it works very efficiently with complex classification[13].



2.2.3 SUPPORT VECTOR MACHINE.

It is a type of supervised ML that can be utilized for classification and regression challenges.

To analyze the statistical classification data and perform the necessary regression analysis, the input to the process is a part of data for the algorithm's training and labeling by its classification into one of the two factions; Whether or not it belongs to the x-type. When the data is plotted on the axes of the characteristics, the algorithm creates a linear frame to separate the characteristics of the two types so that the gap between them is as wide as possible. Then the learning process itself takes place after the training, where another part of data is classified according to the classification found in the training[14].

therefore SVM considers One of the most popular automatic classification methods is machine learning, which relies on finding a curve or super plane that separates the input samples from each other, and is characterized by being limited to classifying problems with binary categories. 1 for positive samples or -1 for negative samples, for example: to classify patient

data samples related to AIDS, if output 1 means that the person has AIDS, and if the output is -1, that is, the person does not have AIDS.

2.2.4 K-NEAREST NEIGHBOR

It is considered a kind of supervised ML algorithm; it is named KNN and is always used for problems of regression and classification. But K- The nearest neighbor is usually applied for classification problems in ML. It works on the precept of assuming that all data points close together fall into the same class. This means that similar things are close together[15].

2.2.5 ENSEMBLE

Ensemble in ML is a method of combining different learning algorithms so that each algorithm supports another algorithm to strengthen the prediction process. The most popular of these methods is to divide the model into a group of decision trees classifiers and random forests by adopting the basic principle that says that a group of Weak decision classifiers behave more powerful than single big decision classifiers; In short, the Ensemble learning process is done by dividing the artificial intelligence model into a group of small models in the learning process and then converting it into one strong model before the prediction stage. Evaluation and voting techniques called genetic algorithms are adopted in this process to get rid of the models that weaken the prediction process[16], (Survival of the fittest or the strongest).

2.2.6 DISCRIMINANT

An algorithm of this type is used in several areas, the most important of which are image identification and predictive with marketing. It's a supervised classification technique, as well as a portion of crafting competitive ML models. The advantage is its reduced dimensions[17].

2.3 Machine-Learning Improving Methods

The development of machine learning models is inherently an iterative process, to achieve the optimal performance you will generally need to apply advanced techniques and iterate.

First, for any given machine learning problem even experts don't know which type of model will perform best. So the first step is to test different models and choose which one performed best.

Other common approaches to improving model performance include:

1. Tuning model parameters. Those are often called "hyperparameters" to distinguish them from the parameters that are learned during normal model training. We will apply this technique in the next section.

2. Transforming existing or extracting new features. If the current feature set doesn't capture all the different characteristics in the data, using different features is likely to assist. Finding the optimal feature set remains one of the aspects of machine learning that require significant expertise.

3. Improving preprocessing and/or adding more training data

4. Addressing imbalances in the data. - For the heart sound data, we have relatively few abnormal heart sounds but very many normal ones. Introducing a bias (with misclassification cost) helps overcome such imbalances.

The first method was adopted during this thesis because our main problem is to improve the performance of predictive heart disease without the need for more data

2.4 HYPERPARAMETER

It is very important for ML algorithms because the hyperparameter directly controls the behavior of the training algorithms and has a noticeable effect on the performance of ML models[19]. Several technologies have been developed and used with success in some areas of application. However, this method requires skilled professional knowledge and experience. Sometimes it resorts to brute force in search[20]. Therefore, if an effective hyperparameter optimization algorithm could be developed to improve any given method of machine learning.

In other words, the ML algorithm turns a problem that needs to be solved into an improvement problem and uses different optimization methods to find a solution to the problem.

The optimization function consists of many hyperparameters placed before the learning process and affects how the ML algorithm fits the model to the data. hyperparameters differ from internal model parameters, such as neural network weights, which can be learned from data during the model training phase[21].

Before the training phase, we would like to find a set of hyperparameter values that would archive the best performance on the data in a reasonable period. This method is called hyperparameter optimization or tuning. It plays a vital role in the prediction accuracy of ML algorithms[22].

In practice, it is necessary to continuously adjust the hyperparameters and train different models with different sets of values, then compare the performance of the model to determine the superior model. Therefore, how to optimize hyperparameters becomes a fundamental problem in ML algorithms[23].

There are two basic types of hyperparameter optimization methods, manual search, and automatic search methods.

2.4.1 MANUAL SEARCH

It is a way to manually try sets of hyperparameter. It is based on the basic intuition and experience of expert users who can determine the important parameters that have the most influence on the results and then determine the relationship between certain parameters and the final results through visualization tools[24]. Manual search requires users to gain more professional background knowledge and practical experiences. And it is difficult to use by non-expert users. The hyperparameter setting cannot be easily repeated. Also, with the increase in the number of hyperparameters and the range of values, it becomes quite difficult to manage them since humans are not very good at handling high-dimensional data[25].

2.4.2 AUTOMATIC SEARCH

To overcome the drawbacks of manual search, automatic search, such as grid search or Cartesian search for hyperparameters, has been suggested[26]. The principle of grid search is a comprehensive search.

Grid search trains the ML model with each of the possible values of the hyperparameters in the training set and evaluates performance against a pre-defined metric on the validation set. As a final point, grid search results in hyperparameters that improve performance[27].

Although this method achieves automatic tuning and can theoretically obtain the global optimum value of the optimization target function, it suffers from a dimensional curse, that is, the performance of the algorithm decreases rapidly as the number of hyperparameters is tuned and the range of values is set to increase the hyperparameters. In other words, this type does not work.

By using a random search algorithm which has been found that for almost all data sets, only a small number of hyperparameters matter The overall efficiency can be improved by reducing the search to hyperparameters that do not matter, and finally, the approximate solution of the optimization function is obtained[29]. A random search attempts random combinations of a set of values. Compared to grid search, the random search is more efficient in a high-dimensional space. However, random research appears to be unreliable for training some complex models[30].

Therefore, how to make the auto-tuning algorithm achieve high accuracy and high efficiency has always been a problem that has not been completely resolved in ML.

It is to experiment with a set of stochastic hyperparameters from a uniform distribution over a given area of hyperparameters and see what works best. It can be easily balanced. Just like Grid search, the performance is slightly better.

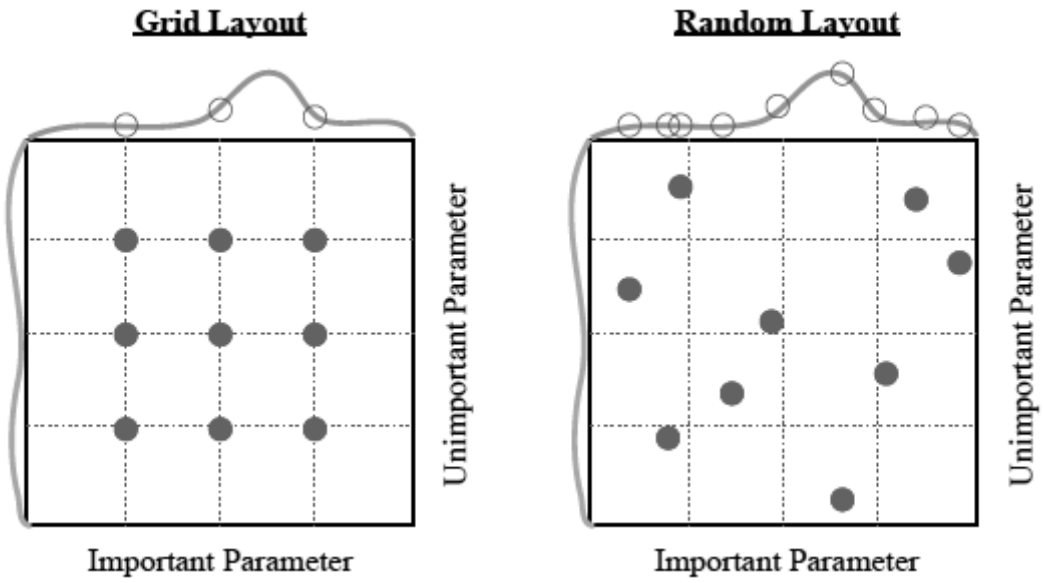


Figure 2.5: Grid and Random Layout

Source: Adapted from[29]

When the researchers tried to compare the random search method and the network search method in terms of performance, they noticed that the random search is better than the network search, it is still computationally intense[31]. If parallelism and simplicity are the most important, then start with this method. But, if we wanted to save time and effort, we would reward big time with Bayesian Optimization.

2.5 BAYESIAN OPTIMIZATION

It is considered a very effective optimization algorithm for solving optimization problems. It collects previous information about the unknown function accompanied by a sample of information, to obtain the following information regarding function distribution using the Bayesian formula. Then, based on the following information, we can deduce where the function gets the optimum value. The experimental results show that the Bayesian optimization algorithm (BO) is superior to the other optimization algorithms. Therefore, Bayesian optimization (BO) depending on the Gaussian process for tuning hyperparameters of ML applies.

(BO) dissimilar to other methods, (BO) uses information from prior iterations for the algorithm.

By (grid search or random search) all hyperparameter guesstimate is independent. Bayesian methods, all-time chose and try out various, and try to build a function with more accuracy and a probability distribution over possible function, that guess how perfect our model might be for a specific choice of hyperparameters[32],[31],[30],[33].

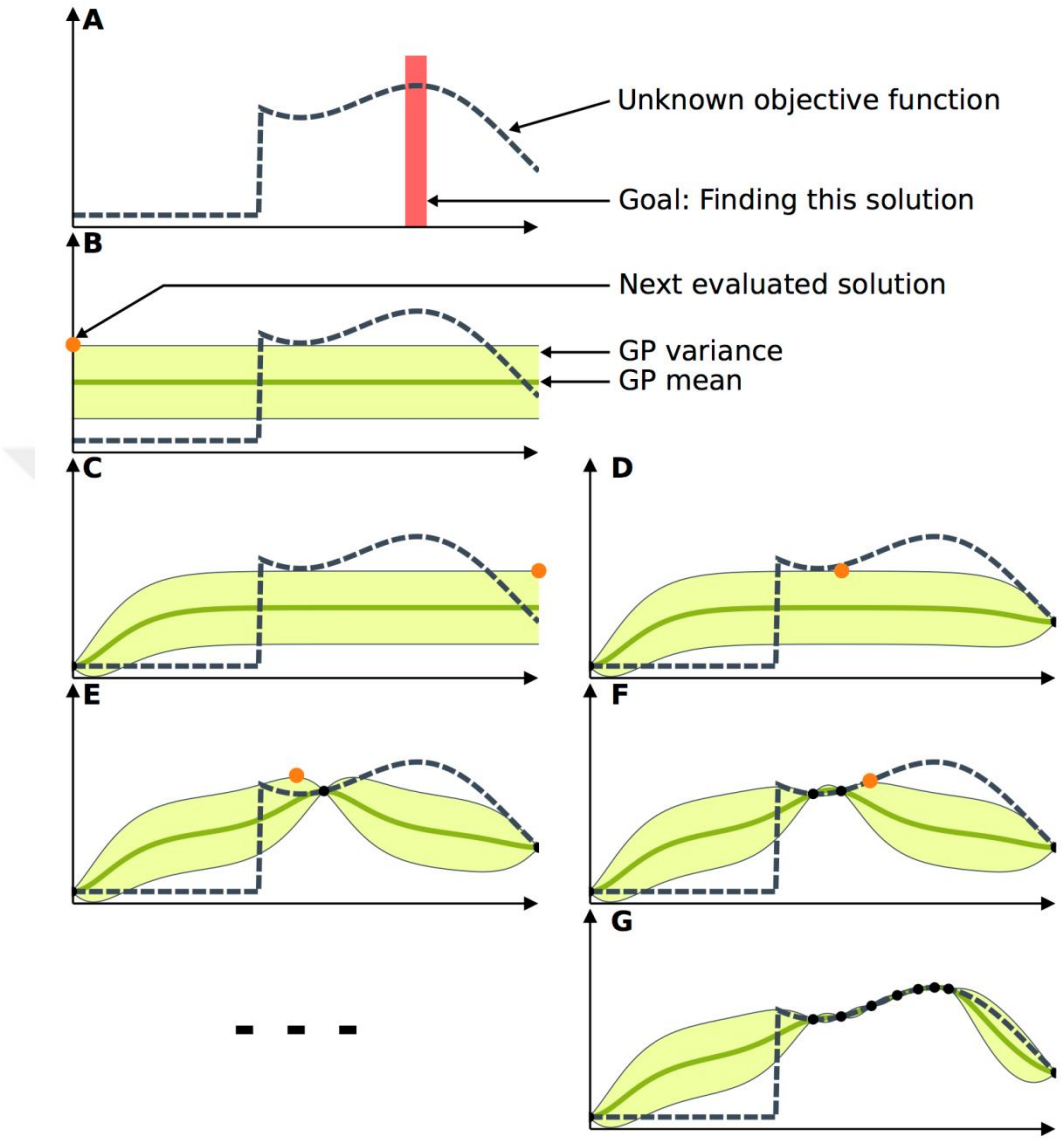


Figure 2.6: BO concept

Source: Adapted from[34]

2.6 LITERATURE REVIEW

Prediction of heart diseases is considered one of the vital topics that several types of research and studies have worked on it during the past few years, and the main goal was to reach high accuracy for prediction and classification, from those researches were as follows:

The classification process is the most significant characteristic of the machine learning method, which is a rule used for the prediction. Some classification algorithms predict with accepted accuracy, but others were given a limited accuracy, by using a method called ensemble classification, which is used to optimize the accuracy of weak algorithms by merging multiple classifiers.

Experimentations with this tool were executed by a heart disease dataset. A comparative analytical process was done to define in what way the ensemble technique can be used for optimizing prediction accuracy for heart diseases. And found The highest increase of 7% accuracy for weak classifiers was given with the help of ensemble classification[35].

Using diverse solutions was suggested by the researchers to discover enhanced accuracy values via the UCI machine learning datasets. In the proposed procedure, an effectual solution is proposed by a dense neural network with hyperparameter tuning for the diagnosis of Coronary artery disease. Investigations were carried out on coronary artery disease data set in a self-regulated method. The result got by the suggested system is 94.91% accuracy, 93.75% recall, and 96.77% precision throughout the timing of the prediction of coronary artery disease[36].

Another suggestion was using different ML techniques which including (DT), (NB), (MLP), (K-NN), Single Conjunctive Rule Learner, RBF, and (SVM) has been performed, one by one

and in a merger, using ensemble ML approach, on the Cleveland Heart Diseases data set to differentiate the performance of every method. In General, the result was Among the seven different techniques, the support vector machine (SVM) outdone all the others when the boosting method was applied [14].

Another research was suggesting a resolution for irregularity and features finding of PCG recordings without segmentation. The suggested method gave the second greatest score in the physionet/cinc Challenge 2016. Several previous methods built on PCG analysis have depended on segmentation which possibly increases the computational load. After the test, The sensitivity was given 86.91% and specificity was given 84.90% on the invisible test data set to give a demonstration of the possibility of enhancement in the future[37].

Saba Bashire et al used an ensemble-based method for the prediction of heart diseases. First data processing and data division are done into the training and testing set. Then model training is done using various classifiers (DT, NB, and SVM). Here various types of classifiers are used in the model to increase the competence of the system. The classification rules of different classifiers are defined during the time of training of the model which will be accumulated as knowledge of the system. Then testing data feed to the classifiers for classification and class of any test instance is identified using a majority voting algorithm[38].

AI procedures can assist ride this issue and foresee a chance at a beginning time. SVM was cognized as the best indicator with 92.1% accuracy, trailed by neural network system precision and choice trees demonstrated a letdown inaccuracy of 89.6% [39].

Gudadhe et al suggested an architecture base with (multilayer perceptron) network and the (support vector machine) approach. This architecture was given the accuracy of (80.41%) in

terms of the classification between two classes (the existing or not existing of heart disease, respectively[40].

Humar Kahramanli and Novruz Allahverdi got an accuracy of (87.4%) by utilizing a hybrid neural network method (HNN) that merges a fuzzy neural network (FNN) algorithm with an artificial neural network (ANN) algorithm [41].

The smart Heart Disease system used a prediction system, This system uses data mining methods, like, (DT, NB, and Neural Network (NN)). The pilot study executed by the authors indicated that the NB model was given the perfect performance accuracy was 86.12%. The second-best model was NN with an accuracy of prediction was 86.12% and the third DT was given accuracy of 80.4% of correct predictions[42].

3. METHODOLOGY

3.1 WORK STEPS FLOW DIAGRAM

The work steps were determined by a flow diagram (Fig.1) and each step is explained in a brief.

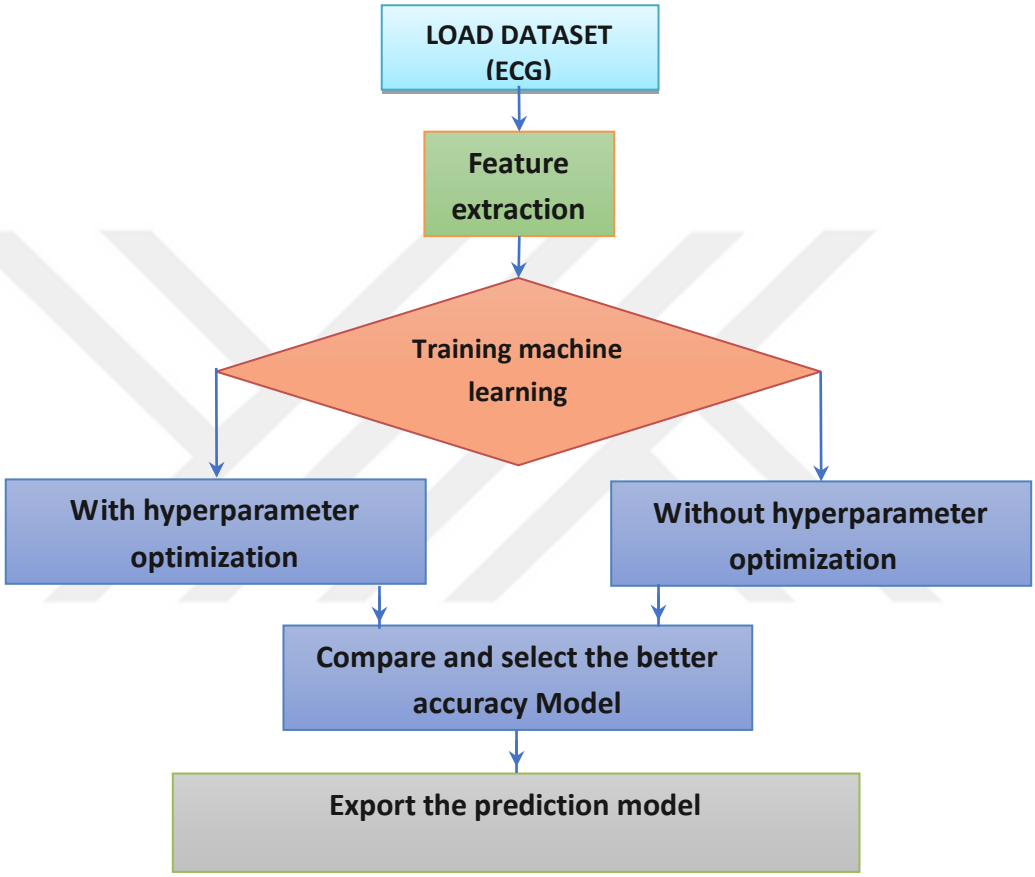


Chart 3.1: Block diagram of work steps

3.2 LOAD DATASET (ECG)

It is a database of heartbeat electrocardiogram (ECG) data from the "physionet2017 Challenge"e web suite, heart signals come from a short ECG lead signal and record with time was (between the 30s - 60s in length), the dataset is a table containing an ECG signal data with

labeling for each signal with three disease class(ARR, CHR, NSR)(See fig 10), this table loaded to the workspace and prepare for next step.

ECGTable

162x2 table

	1 Signal	2 Labels	3	4	5	6	7	8
1	*65536x1 timetable*	'ARR'						
2	*65536x1 timetable*	'ARR'						
3	*65536x1 timetable*	'ARR'						
4	*65536x1 timetable*	'ARR'						
5	*65536x1 timetable*	'ARR'						
6	*65536x1 timetable*	'ARR'						
7	*65536x1 timetable*	'ARR'						
8	*65536x1 timetable*	'ARR'						
9	*65536x1 timetable*	'ARR'						
10	*65536x1 timetable*	'ARR'						
11	*65536x1 timetable*	'ARR'						
12	*65536x1 timetable*	'ARR'						
13	*65536x1 timetable*	'ARR'						
14	*65536x1 timetable*	'ARR'						
15	*65536x1 timetable*	'ARR'						
16	*65536x1 timetable*	'ARR'						
17	*65536x1 timetable*	'ARR'						
18	*65536x1 timetable*	'ARR'						
19	*65536x1 timetable*	'ARR'						

Figure 3.1: Data table containing an ECG signal data with labeling

Code:

```
load ECGTable. Mat
```

in the ECG dataset, there are 3 labels associated with their own signal data.

```
ECGTable_ARR_Example = ECGTable{1,1}{1};
ECGTable_CHR_Example = ECGTable{97,1}{1};
ECGTable_NSR_Example = ECGTable{127,1}{1};
subplot(3,1,1)
plot(ECGTable_ARR_Example.Time(1:1000),ECGTable_ARR_Example.SignalData(1:1000))
;
title('ARR')
grid on
subplot(3,1,2)
plot(ECGTable_CHR_Example.Time(1:1000),ECGTable_CHR_Example.SignalData(1:1000))
;
title('CHR')
grid on
subplot(3,1,3)
plot(ECGTable_NSR_Example.Time(1:1000),ECGTable_NSR_Example.SignalData(1:1000))
;
title('NSR')
grid on
```

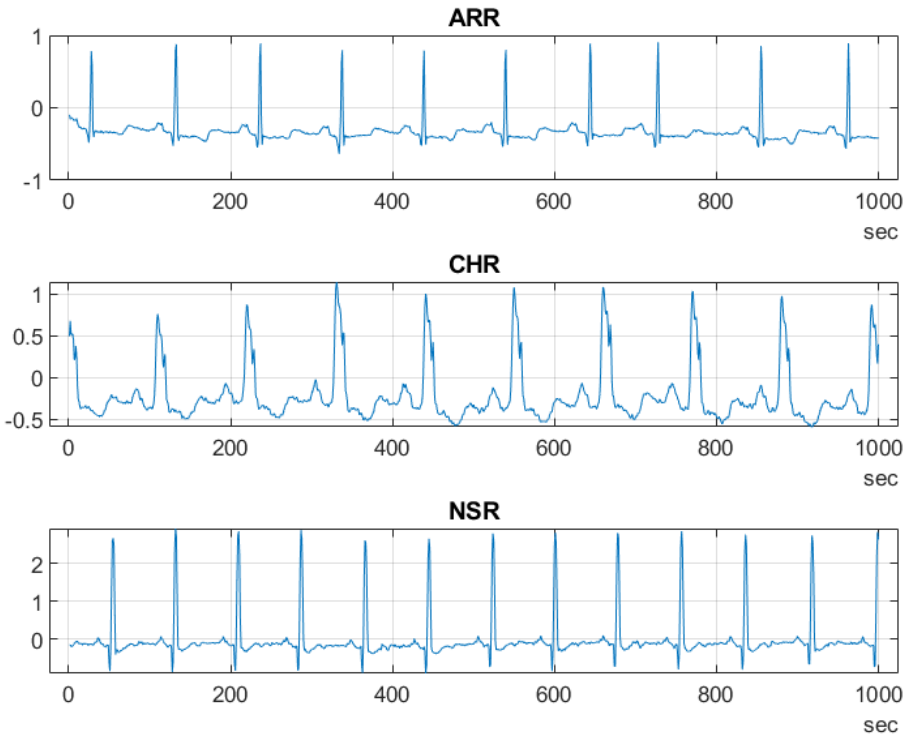


Figure 3.2: signals with three disease class(ARR, CHR, NSR)

3.3 FEATURE EXTRACTION

The process that reduces the dimensions is called feature extraction. When the input (dataset) of an algorithm is so large that it is difficult to process easily, and it is expected that it will form a surplus in data that may lead to high calculation and processing costs and the use of computer memory without a return commensurate with that cost, then the data is changed to a simple form representing original ECG signals data. The process by which data transformed into its characteristics is called feature extraction. This thesis used Extract the Time Domain Signal Features to generate statistics features from the signal by using the Matlab program, it is included

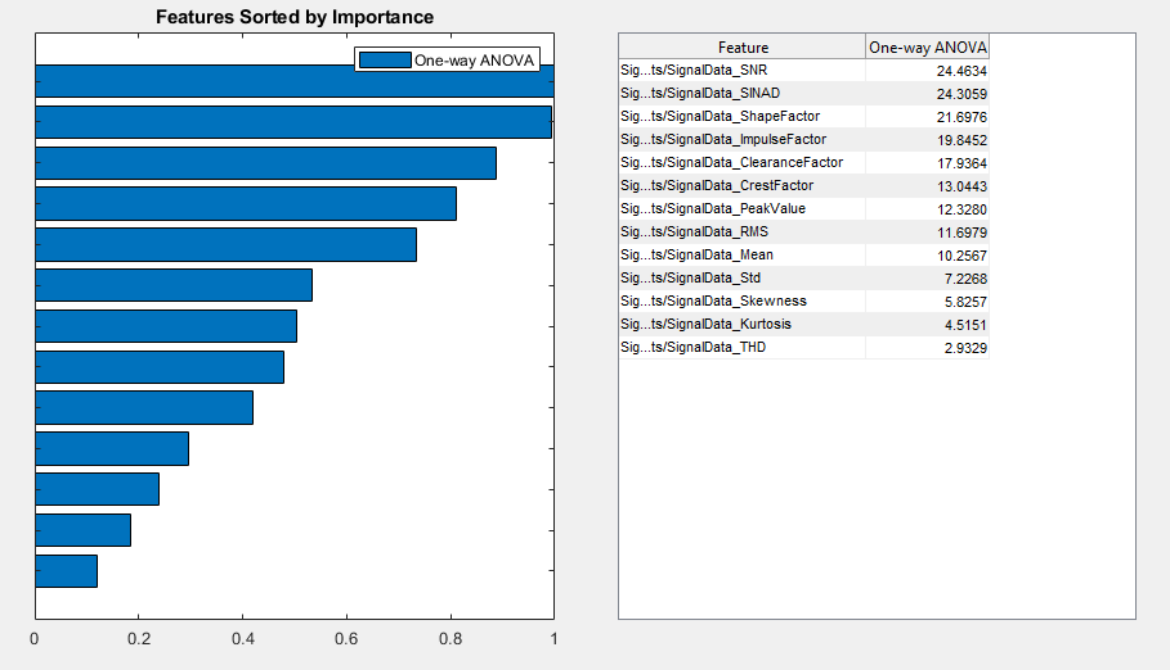
A. *BASIC STATISTICS FEATURES.*

B. *HIGH ORDER STATISTICS FEATURES.*

C. *IMPULSIVE METRICS.*

D. *SIGNAL PROCESSING METRICS*.

This research got a table of features (See fig.10,11,12) features for signals, 13 features for each signal with its label, and after then export this features table to the Matlab workspace to be ready for training.



Feature	One-way ANOVA
Sig...ts/SignalData_SNR	24.4634
Sig...ts/SignalData_SINAD	24.3059
Sig...ts/SignalData_ShapeFactor	21.6976
Sig...ts/SignalData_ImpulseFactor	19.8452
Sig...ts/SignalData_ClearanceFactor	17.9364
Sig...ts/SignalData_CrestFactor	13.0443
Sig...ts/SignalData_PeakValue	12.3280
Sig...ts/SignalData_RMS	11.6979
Sig...ts/SignalData_Mean	10.2567
Sig...ts/SignalData_Std	7.2268
Sig...ts/SignalData_Skewness	5.8257
Sig...ts/SignalData_Kurtosis	4.5151
Sig...ts/SignalData_THD	2.9329

Chart 3.2: Features of signals sorted by importance

	/SignalData_SNR	Signal_stats/SignalData_ShapeFactor	Signal_stats/SignalData_Skewness	Signal_stats/SignalData_Std	Signal_stats/SignalData_THD	Labels
138	-17.9467	1.4964	2.3568	0.2141	-2.6920	NSR
139	-17.5584	1.5197	-1.9328	0.1422	-5.1243	NSR
140	-14.4493	1.8341	-2.0179	0.2340	-8.9379	NSR
141	-14.8547	1.7402	-2.3518	0.2108	-7.6841	NSR
142	-9.8642	1.6386	-2.3587	0.4057	-13.5094	NSR
143	-14.0309	1.6423	-2.9136	0.1620	-5.4063	NSR
144	-13.4818	1.2376	0.5758	0.3752	-13.6549	NSR
145	-16.4633	1.9063	-4.1215	0.1949	-3.3808	NSR
146	-10.5461	1.3548	-0.8226	0.2748	-12.8143	NSR
147	-19.5196	1.6106	-3.5002	0.1375	-2.4052	NSR
148	-17.8440	1.4931	-0.9713	0.2160	-7.1945	NSR
149	-17.9528	1.9378	3.9549	0.5512	-8.2904	NSR
150	-8.6285	1.4835	-1.3702	0.1910	-8.6289	NSR
151	-19.0557	1.4707	0.5655	0.2743	-9.0725	NSR
152	-16.2110	1.3183	-0.6433	0.1397	-3.3475	NSR
153	-13.3083	1.3562	-0.9864	0.1822	-6.8566	NSR
154	-9.0222	1.5025	-0.4262	0.3343	-2.6051	NSR
155	-18.9617	1.8041	-2.6192	0.1838	-7.9966	NSR
156	-15.0041	1.9148	0.0670	0.1360	-8.5531	NSR
157	-8.5645	2.0220	1.8431	0.3868	-17.7915	NSR
158	-15.0987	1.4625	2.3309	0.3155	-8.6941	NSR
159	-21.4202	1.6624	3.6032	0.2838	-7.9022	NSR
160	-16.4806	1.6042	3.0291	0.5575	-8.4512	NSR
161	-20.7733	1.7868	4.3374	0.4714	0.2962	NSR
162	-12.9305	1.4229	-0.0382	0.2564	-7.9870	NSR

Figure 3.3: Excel table of features(162*13)

3.4 TRAINING

Machine learning is some generalized algorithms that can tell us about important things in the data without you programming it. Instead of programming, just feed the data and then applies the method of the algorithm to understanding the data.

Machine learning algorithm creates a model by test many examples and tries to find a model that reduces the loss of change in algorithm parameters with iterations, it is called training to reach the goal or best accuracy. This thesis shows the training step with two-phase(with hyperparameter optimization, without hyperparameter optimization) and makes comparisons between two phases to select a better way. After extracting the features from the dataset during the feature extraction step (See fig.14), and applied it to the machine learning algorithms but with two phases, the first one without optimizing the hyperparameter and the second with optimizing the hyperparameter for the same data to get better accuracy.

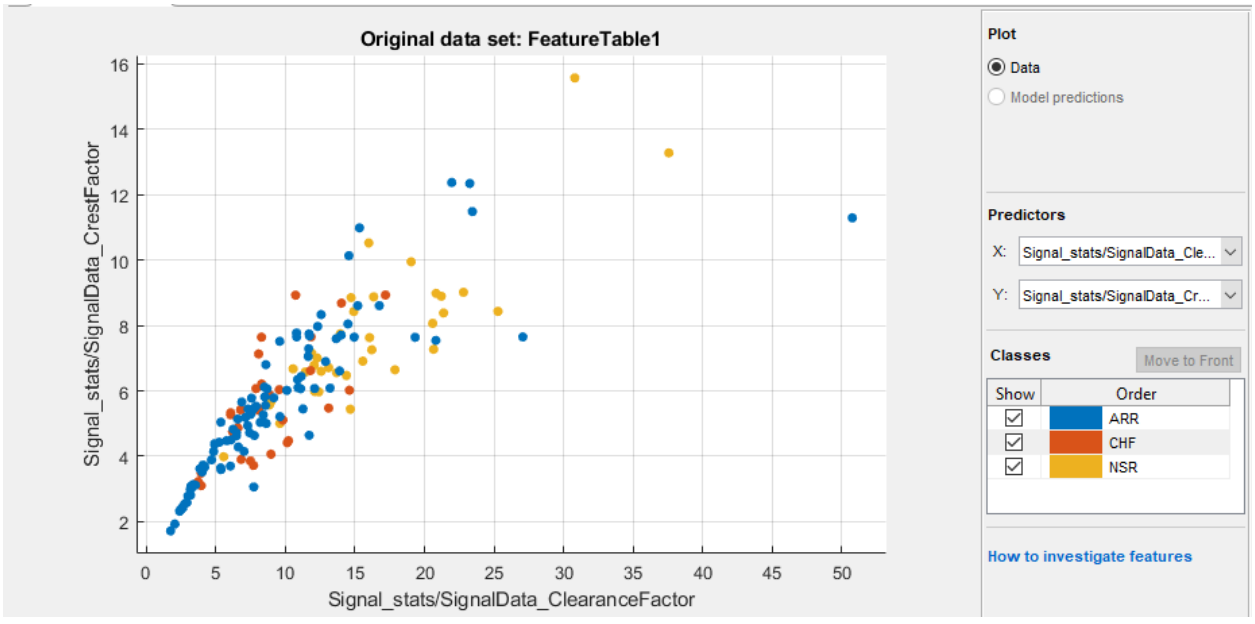


Figure 3.4: original dataset/feature extraction

3.4.1 FIRST PHASE(WITHOUT HYPERPARAMETER OPTIMIZATION)

After applied machine learning algorithm (SVM), boosted and bagged decision trees, KNN, NB, discriminant analysis) on the feature extraction table, this phase got the result with different accuracy (see **Table1)**, but the winning algorithm was a decision tree with type coarse tree (accuracy: 85.8%).

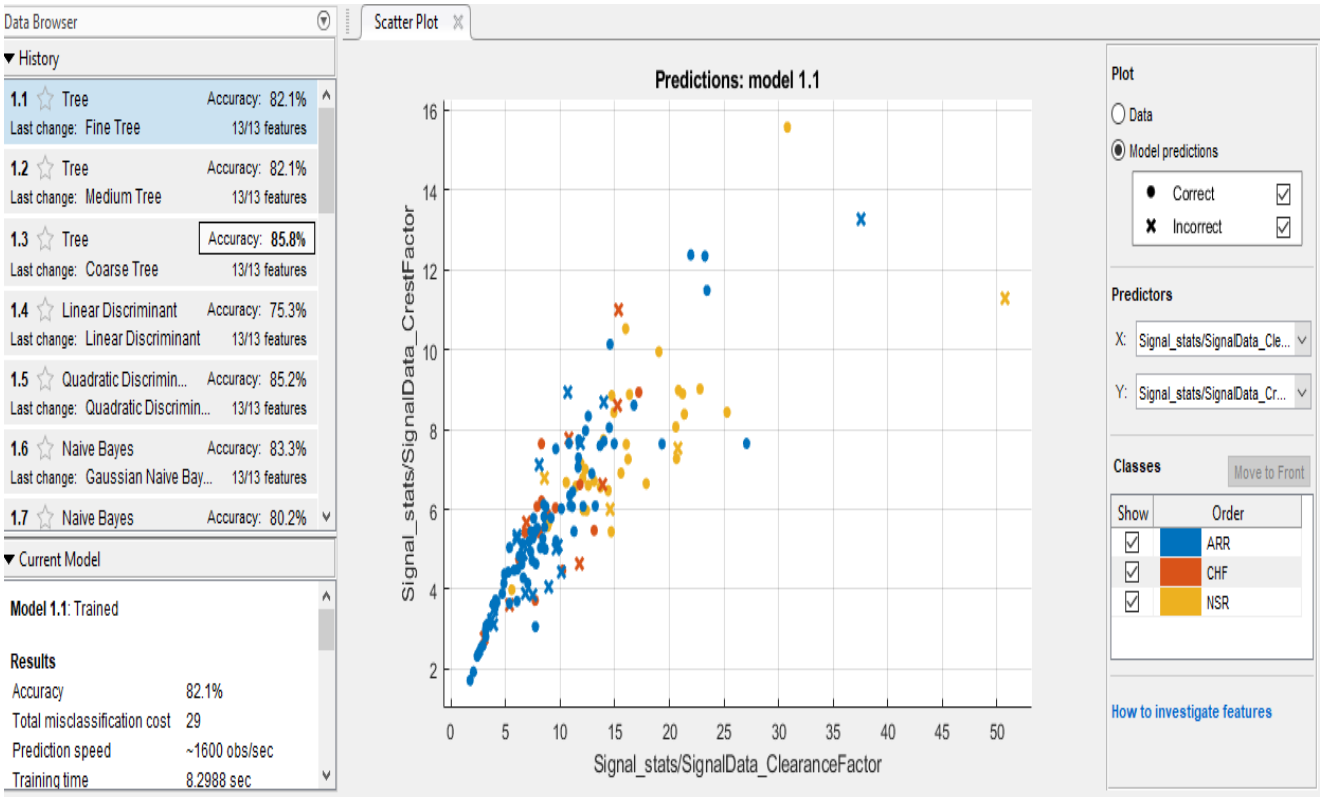


Figure 3.5: Prediction model results(without Hyperparameter Optimization).

3.4.2 THE SECOND PHASE(WITH HYPERPARAMETER OPTIMIZATION)

By using the same feature extraction table and select the algorithm which is given the best accuracy in the first phase (tree acc = 88.3) to apply hyperparameter optimization to reach better accuracy without needing more data of heart signals, after training with this mode the accuracy is increased to 88.3%. The false-positive rate(FPR) shows different values of a threshold with three types of heart diseases (ARR, CHF, NSR).

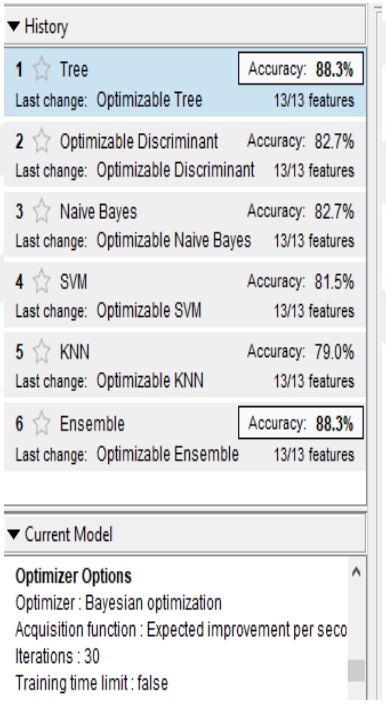


Figure 3.6: Prediction model results(with Hyperparameter Optimization).

4. EXPERIMENTAL RESULTS AND ANALYSIS

4.1 RESULT OF THE FIRST PHASE(without hyperparameter optimization)

After applied machine learning algorithm (SVM), boosted and bagged decision trees, KNN, NB, discriminant analysis) on the feature extraction table, this phase got the result with different accuracy (see **Table1)**, but the winning algorithm was a decision tree with type coarse tree (accuracy: 85.8%).

Model	Algorithm with type	Accuracy
1.1	Tree - fine Tree	82.1%
1.2	Tree-medium Tree	82.1%
1.3	Tree- coarse Tree	85.8%
1.4	Linear Discriminant	75.3%
1.5	Quadratic Discriminant	85.2%
1.6	Naïve Bayes -Gaussian	83.3%
1.7	Naïve Bayes -kernel	80.2%
1.8	SVM – linear SVM	71.0%
1.9	SVM – Quadratic	77.8%
1.10	SVM – Cubic	79.6%
1.11	SVM – Gaussian	61.1%
1.12	SVM – Medium Gaussian	71.0%
1.13	SVM – Coarse Gaussian	60.5%
1.14	KNN - Fine	76.5%
1.15	KNN – Medium	77.8%
1.16	KNN - Coarse	59.3%
1.17	KNN - Cosine	75.9%
1.18	KNN - Weighted	81.5%
1.19	KNN - Cubic	73.5%
1.20	Ensemble –Boosted Trees	59.3%
1.21	Ensemble –Subspace Disc	71.0%
1.22	Ensemble –Bagged Trees	71.6.3%
1.23	Ensemble –Boosted Trees	59.3%

Table1. [machine learning Algorithm result of accuracy] WITHOUT HYPERPARAMETER OPTIMIZATION

ROC curve to evaluate classification models. ROC curves plot (TPR) and (FPR) to shows different values of a threshold for the three types of heart diseases (ARR, CHF, NSR) (See Fig 18,19, and 20).

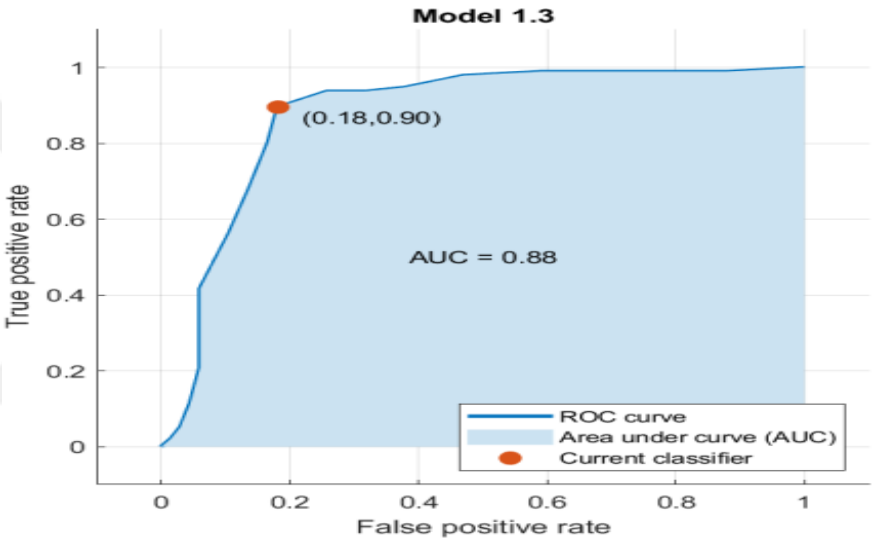


Chart 4.1: [ROC curve of ARR class/ model 1.3 coarse tree]

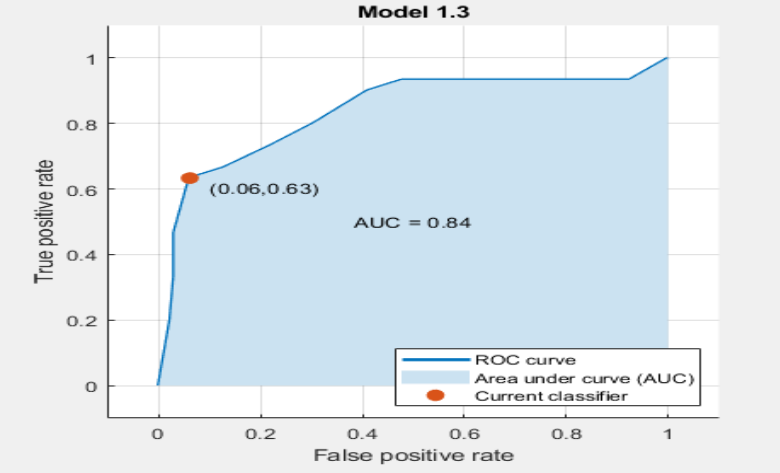


Chart 4.2: [ROC curve of CHR class/ model 1.3 coarse tree]

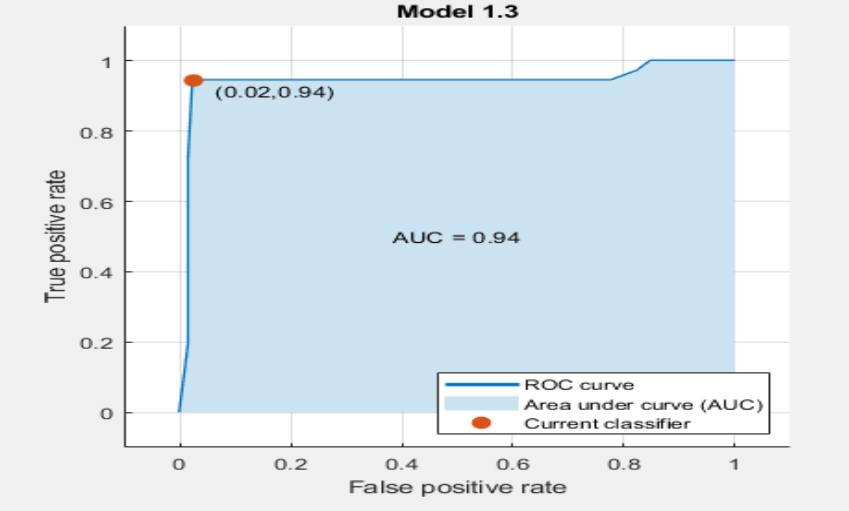


Chart 4.3: [ROC curve of NSR class/ model 1.3 coarse tree]

(Fig.21) it represents the confusion matrix that declares True Positive Rates Denoted by (TPR) and False Negative Rates Denoted by (FNR) for the second phase.

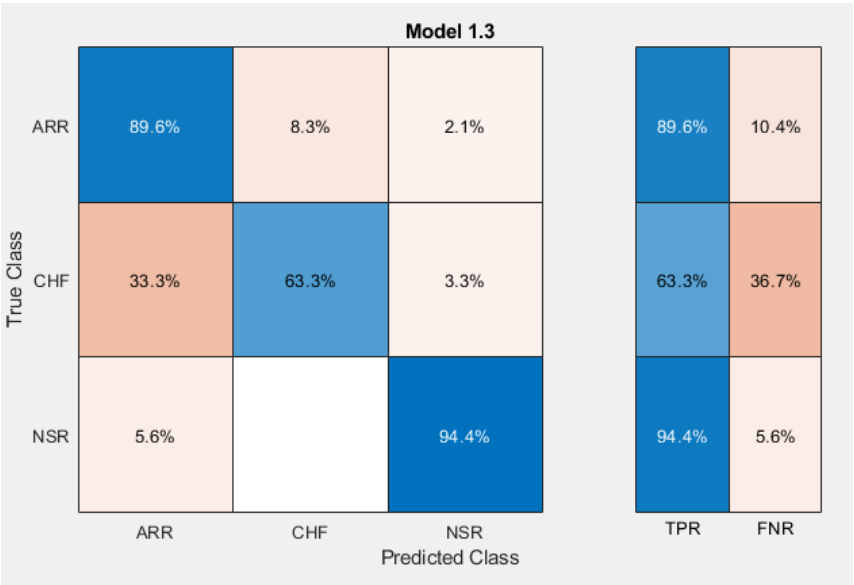


Chart 4.4: [(TPR) and (FNR) of model 1.3 coarse tree]

(Fig. 22) blew of confusion matrix declare the increase of Positive Predictive Values (PPV) and decrease False Discovery Rates (FDR).

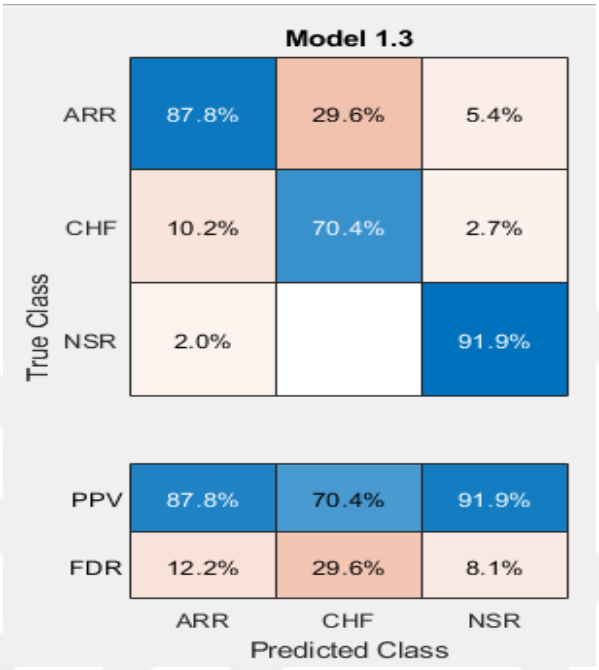


Chart 4.5: [(PPV) and (FDR) of model 1.3 coarse tree]

4.2 RESULT OF THE SECOND PHASE (with hyperparameter optimization)

By using the same feature extraction table and select the algorithm which is given the best accuracy in the first phase (tree-coarse tree acc = 85.8) to apply hyperparameter optimization to reach better accuracy without needing more data of heart signals, after training with this mode the accuracy is increased to 88.3%. (see table 2) the false-positive rate(FPR) shows different values of a threshold with three types of heart diseases (ARR, CHF, NSR) (see fig. 24,25, and 26).

Model	Algorithm with type	Accuracy
1	Optimizable Tree	88.3%
2	Optimizable Discriminant	82.7%
3	Optimizable Naïve Bayes	82.7%
4	Optimizable SVM	81.5%
5	Optimizable KNN - Fine	79.0%
6	Optimizable Ensemble	88.3%

Table2. [machine learning Algorithm result of accuracy] WITH HYPERPARAMETER OPTIMIZATION

By using Bayesian optimization we got a minimum classification error in 30 iterations. (see Fig 23).

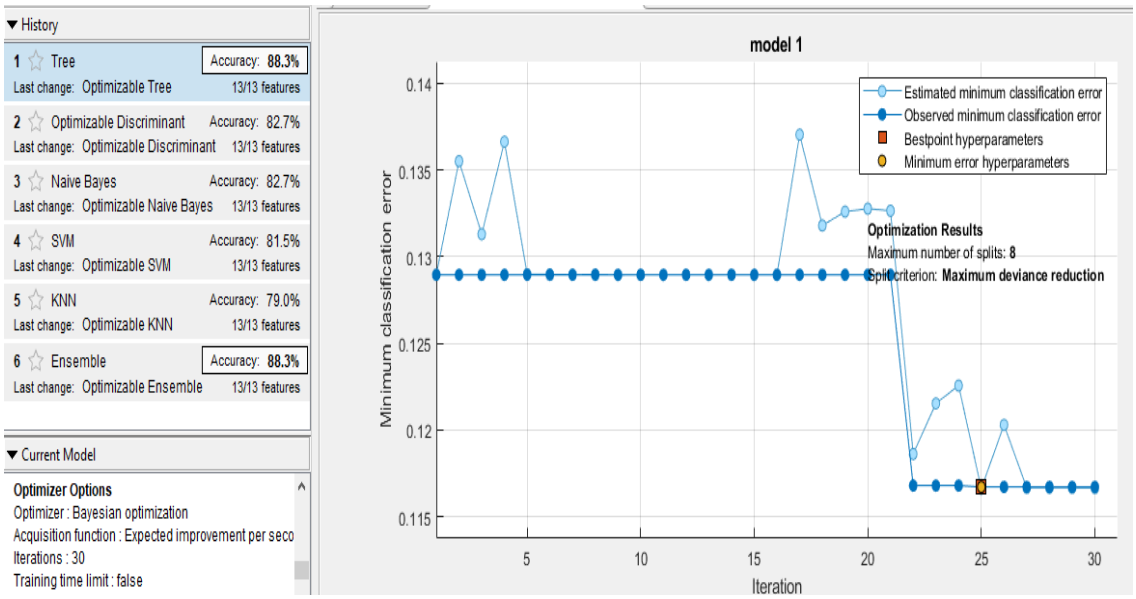


Chart 4.6: Bayesian optimization result

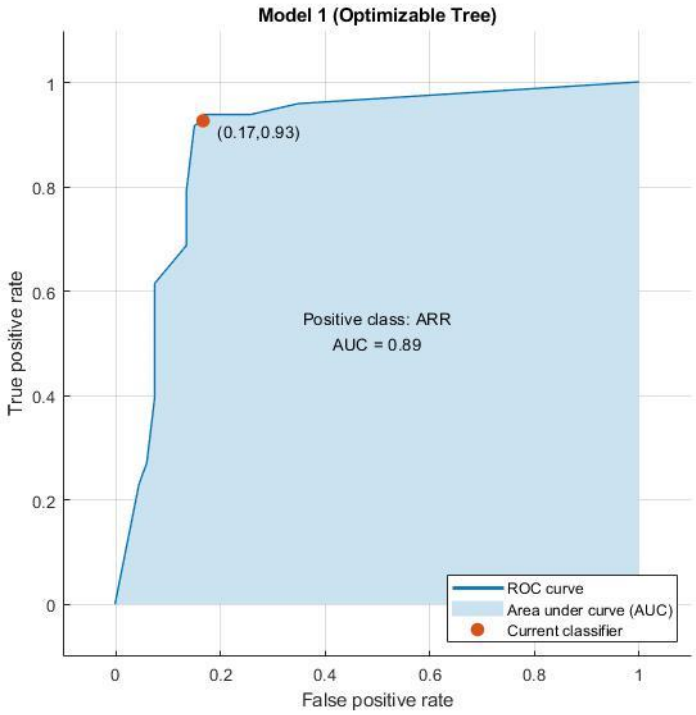


Chart 4.7: [ROC curve of ARR class/ model 1 tree with optimization]

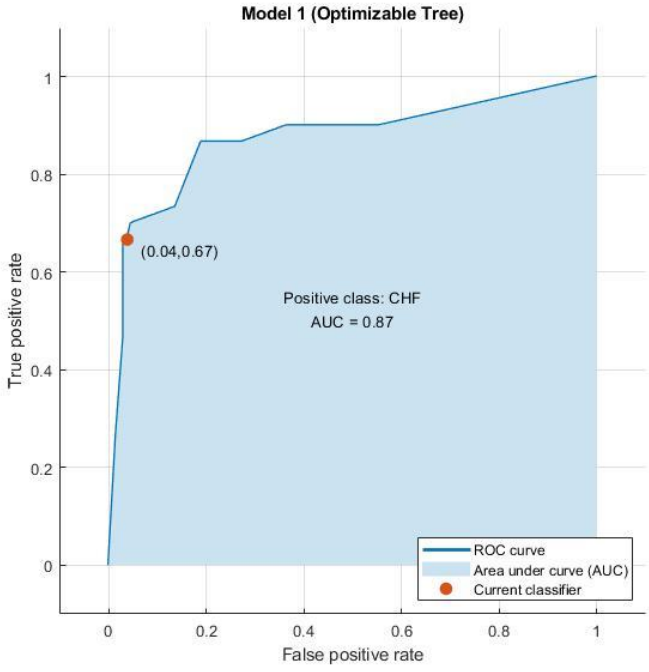


Chart 4.8: [ROC curve of CHR class/ model 1 tree with optimization]

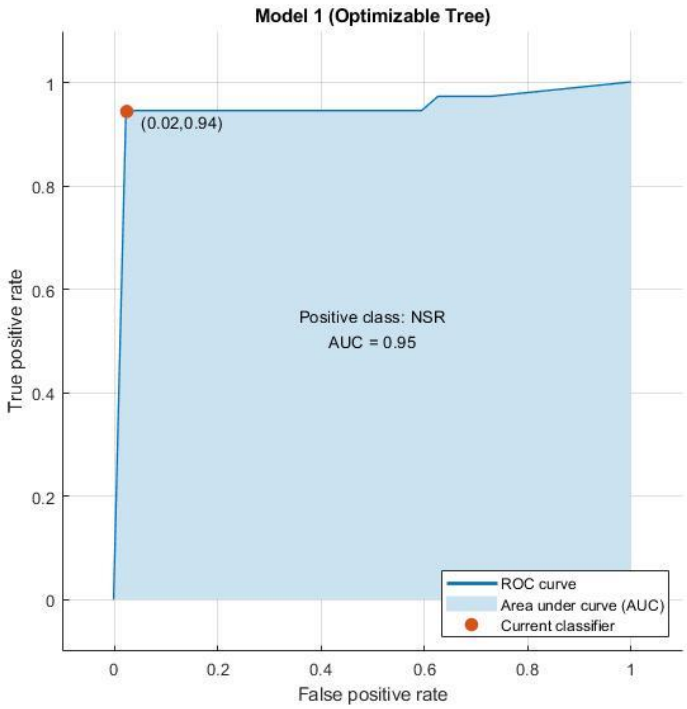


Chart 4.9: [ROC curve of NSR class/ model 1 tree with optimization]

This (Fig.27) blew a confusion matrix declares the increase of True Positive Rates Denoted by (TPR) and decreases False Negative Rates Denoted by (FNR) for the second phase.

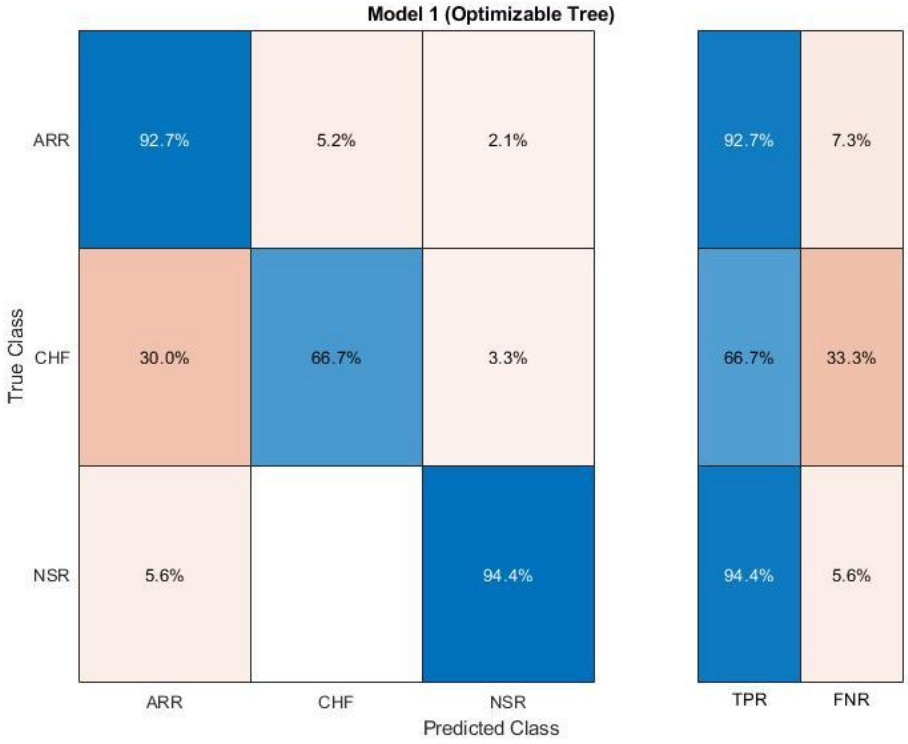


Chart 4.10: [(TPR) and (FNR) of model 1 tree with optimization]

(Fig.28) of the confusion, matrix declares the increase of Positive Predictive Values Denoted by (PPV) and decrease False Discovery Rates Denoted by (FDR).

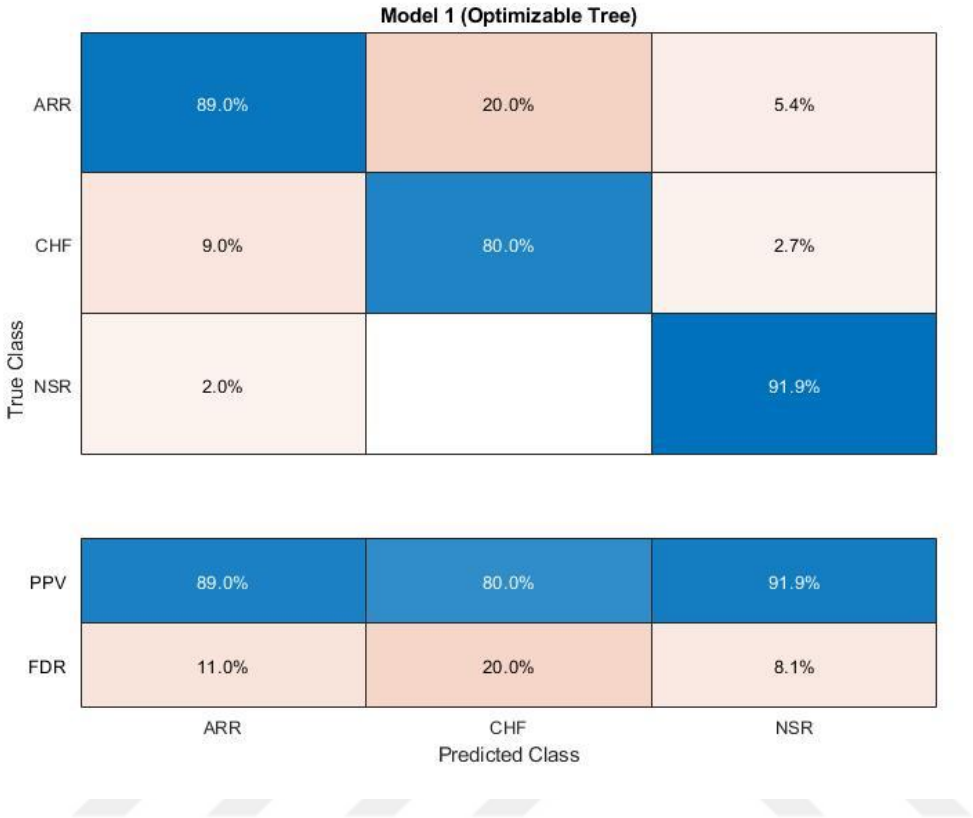


Chart 4.10: [(PPV) and (FDR) of model 1 tree with optimization]

5. CONCLUSION AND PROPOSALS OF FUTURITY WORKS.

The major purpose of the thesis is to develop cardiac disease prediction systems via ML techniques and to highlight one method that attempts to improve ML algorithms without the requirement for further data acquisition, by using hyperparameter tuning using the Bayesian method.

This thesis presented different machine learning algorithms that were applied to the features extraction of ECG signals which were gotten from the “physionet2017 Challenge” web suite with three types of heart disease(ARR, CHF, NSR). The accuracy of prediction is increased to 88.3% by using a decision tree algorithm without adding new data.

As for future work, after it became clear that it is possible to obtain an accurate prediction for ML algorithms using optimization techniques, we will be able to apply these techniques, especially with data that are difficult to obtain, especially in medical fields such as prediction of the heart, brain diseases, cancer, and other diseases and systems. Building. Health provides more exact and dependable information.

REFERENCES

[1] M. H. Vafaie, M. Ataei, and H. R. Koofigar, "Heart diseases prediction based on ECG signals' classification using a genetic-fuzzy system and dynamical model of ECG signals," *Biomed. Signal Process. Control*, vol. 14, pp. 291–296, 2014, doi: 10.1016/j.bspc.2014.08.010.

[2] A. Gacek and W. Pedrycz, "ECG signal processing, classification, and interpretation: A comprehensive framework of computational intelligence," *ECG Signal Process. Classif. Interpret. A Compr. Framew. Comput. Intell.*, vol. 9780857298683, pp. 1–278, 2014, doi: 10.1007/978-0-85729-868-3.

[3] N. Strodthoff and C. Strodthoff, "Detecting and interpreting myocardial infarction using fully convolutional neural networks," *Physiol. Meas.*, vol. 40, no. 1, pp. 1–11, 2019, doi: 10.1088/1361-6579/aaf34d.

[4] J. Bai, A. Lo, P. A. Gladding, M. K. Stiles, V. V. Fedorov, and J. Zhao, "In silico investigation of the mechanisms underlying atrial fibrillation due to impaired Pitx2," *PLoS Comput. Biol.*, vol. 16, no. 2, 2020, doi: 10.1371/journal.pcbi.1007678.

[5] S. V, "an Empirical Science Research on Bioinformatics in Machine Learning," *J. Mech. Contin. Math. Sci.*, vol. spl7, no. 1, pp. 86–94, 2020, doi: 10.26782/jmcms.spl.7/2020.02.00006.

[6] R. Konieczny and R. Idczak, "Mössbauer study of Fe-Re alloys prepared by mechanical alloying," *Hyperfine Interact.*, vol. 237, no. 1, pp. 1–8, 2016, doi: 10.1007/s10751-016-1232-6.

[7] S. Sah, "Machine Learning: A Review of Learning Types," no. July, 2020, doi: 10.20944/preprints202007.0230.v1.

[8] K. El Bouchefry and R. S. de Souza, "Learning in Big Data: Introduction to Machine Learning," *Knowl. Discov. Big Data from Astron. Earth Obs.*, pp. 225–249, 2020, doi: 10.1016/b978-0-12-819154-5.00023-0.

[9] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, 2013, doi: 10.1109/TPAMI.2013.50.

[10] V. François-lavet *et al.*, "An Introduction to Deep Reinforcement Learning. (arXiv:1811.12560v1 [cs.LG]) http://arxiv.org/abs/1811.12560," *Found. trends Mach. Learn.*, vol. II, no. 3–4, pp. 1–140, 2018, doi: 10.1561/2200000071.Vincent.

[11] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*, vol. 9781107057135. 2013.

[12] A. Mccallum, "Graphical Models : Bayesian Network," 2011, [Online]. Available: https://people.cs.umass.edu/~mccallum/courses/gm2011/.

[13] H. M. Abdelaal and H. A. Youness, "Hadith Classification using Machine Learning Techniques According to its Reliability," *Rom. J. Inf. Sci. Technol.*, vol. 22, no. 3–4, pp. 259–271, 2019.

[14] S. Pouriyeh, S. Vahid, G. Sannino, G. De Pietro, H. Arabnia, and J. Gutierrez, "A comprehensive investigation and comparison of Machine Learning Techniques in the domain of heart disease," *Proc. - IEEE Symp. Comput. Commun.*, no. July, pp. 204–207, 2017, doi: 10.1109/ISCC.2017.8024530.

[15] N. Satyanarayana, C. Ramalingaswamy, and Y. Ramadevi, "Survey of Classification Techniques in Data Mining," *IJISET -International J. Innov. Sci. Eng. Technol.*, vol. 1, no. 9, pp. 65–71, 2014, [Online]. Available: www.ijiset.com.

[16] G. Wu, R. Mallipeddi, and P. N. Suganthan, "Ensemble strategies for population-based optimization algorithms – A survey," *Swarm Evol. Comput.*, vol. 44, pp. 695–711, 2019, doi: 10.1016/j.swevo.2018.08.015.

[17] W. Li, F. Feng, H. Li, and Q. Du, "Discriminant Analysis-Based Dimension Reduction for Hyperspectral Image Classification: A Survey of the Most Recent Advances and an Experimental Comparison of Different Techniques," *IEEE Geosci. Remote Sens. Mag.*, vol. 6, no. 1, pp. 15–34, 2018, doi: 10.1109/MGRS.2018.2793873.

[18] Z. Wu *et al.*, "MoleculeNet: A benchmark for molecular machine learning," *Chem. Sci.*, vol. 9, no. 2, pp. 513–530, 2018, doi: 10.1039/c7sc02664a.

[19] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, 2020, doi: 10.1016/j.neucom.2020.07.061.

[20] G. F. Luger, *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*, vol. 5th. 2005.

[21] M. Claesen and B. De Moor, "Hyperparameter Search in Machine Learning," pp. 10–14, 2015, [Online]. Available: http://arxiv.org/abs/1502.02127.

[22] J. Wu, X. Y. Chen, H. Zhang, L. D. Xiong, H. Lei, and S. H. Deng, "Hyperparameter optimization for machine learning models based on Bayesian optimization," *J. Electron. Sci. Technol.*, vol. 17, no. 1, pp. 26–40, 2019, doi: 10.11989/JEST.1674-862X.80904120.

[23] F. Hutter, *Meta-learning*, vol. 498. 2014.

[24] Tianyi Li, G. Convertino, W. Wang, H. Most, T. Zajonc, and Y.-H. Tsai, "HyperTuner: Visual Analytics for Hyperparameter Tuning by Professionals," *Mach. Learn. from User Interact. Vis. Anal. Work.*, no. Ml, 2018, [Online]. Available: https://learningfromusersworkshop.github.io/papers/HyperTuner.pdf.

[25] J. Liu, S. Tripathi, U. Kurup, and M. Shah, "Auptimizer-An Extensible, Open-Source Framework for Hyperparameter Tuning," *Proc. - 2019 IEEE Int. Conf. Big Data, Big Data 2019*, pp. 339–348, 2019, doi: 10.1109/BigData47090.2019.9006330.

[26] S. R. Young, D. C. Rose, T. P. Karnowski, S. H. Lim, and R. M. Patton, "Optimizing deep learning hyper-parameters through an evolutionary algorithm," *Proc. MLHPC 2015 Mach. Learn. High-Performance Comput. Environ. - Held conjunction with SC 2015 Int. Conf. High Perform. Comput. Networking, Storage Anal.*, no. August 2016, 2015, doi: 10.1145/2834892.2834896.

[27] B. Fowler, "A sociological analysis of the satanic verses affair," *Theory, Cult. Soc.*, vol.

17, no. 1, pp. 39–61, 2000, doi: 10.1177/02632760022050997.

[28] D. Chicco, “Ten quick tips for machine learning in computational biology,” *BioData Min.*, vol. 10, no. 1, pp. 1–17, 2017, doi: 10.1186/s13040-017-0155-3.

[29] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, 2012.

[30] Z. Wang, F. Hutter, M. Zoghi, D. Matheson, and N. De Freitas, “Bayesian optimization in a billion dimensions via random embeddings,” *J. Artif. Intell. Res.*, vol. 55, no. 1, pp. 361–367, 2016, doi: 10.1613/jair.4806.

[31] L. Li and A. Talwalkar, “Random search and reproducibility for neural architecture search,” *arXiv*, pp. 1–20, 2019.

[32] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, “Algorithms for hyper-parameter optimization,” *Adv. Neural Inf. Process. Syst. 24 25th Annu. Conf. Neural Inf. Process. Syst. 2011, NIPS 2011*, pp. 1–9, 2011.

[33] L. Franceschi, M. Donini, P. Frasconi, and M. Pontil, “Forward and reverse gradient-based hyperparameter optimization,” *arXiv*, 2017.

[34] J. Mockus, “Bayesian heuristic approach to global optimization and examples,” *J. Glob. Optim.*, vol. 22, no. 1–4, pp. 191–203, 2002, doi: 10.1023/a:1013815314823.

[35] C. B. C. Latha and S. C. Jeeva, “Improving the accuracy of prediction of heart disease risk based on ensemble classification techniques,” *Informatics Med. Unlocked*, vol. 16, no. November 2018, p. 100203, 2019, doi: 10.1016/j.imu.2019.100203.

[36] D. Swain, S. K. Pani, and D. Swain, “An efficient system for the prediction of coronary artery disease using dense neural network with hyper parameter tuning,” *Int. J. Innov. Technol. Explor. Eng.*, vol. 8, no. 6, pp. 689–695, 2019.

[37] M. Zabihi, A. B. Rad, S. Kiranyaz, M. Gabbouj, and A. K. Katsaggelos, “Heart sound anomaly and quality detection using ensemble of neural networks without segmentation,” *Comput. Cardiol. (2010).*, vol. 43, pp. 613–616, 2016, doi: 10.22489/cinc.2016.180-213.

[38] S. Bashir, U. Qamar, and M. Y. Javed, “An ensemble based decision support framework for intelligent heart disease diagnosis,” *Int. Conf. Inf. Soc. i-Society 2014*, no. November, pp. 259–264, 2015, doi: 10.1109/i-Society.2014.7009056.

[39] V. K and J. Singaraju, “Decision Support System for Congenital Heart Disease Diagnosis based on Signs and Symptoms using Neural Networks,” *Int. J. Comput. Appl.*, vol. 19, no. 6, pp. 6–12, 2011, doi: 10.5120/2368-3115.

[40] M. Gudadhe, K. Wankhade, and S. Dongre, “Decision support system for heart disease based on support vector machine and artificial neural network,” *2010 Int. Conf. Comput. Commun. Technol. ICCCT-2010*, pp. 741–745, 2010, doi: 10.1109/ICCCT.2010.5640377.

[41] H. Kahramanli and N. Allahverdi, “Design of a hybrid system for the diabetes and heart diseases,” *Expert Syst. Appl.*, vol. 35, no. 1–2, pp. 82–89, 2008, doi: 10.1016/j.eswa.2007.06.004.

[42] S. Palaniappan and R. Awang, “Intelligent heart disease prediction system using data

mining techniques," *AICCSA 08 - 6th IEEE/ACS Int. Conf. Comput. Syst. Appl.*, no. December, pp. 108–115, 2008, doi: 10.1109/AICCSA.2008.4493524.

APPENDIX A

[Matlab code]

This appendix includes the code for this thesis for the work stages according to the Matlab program, as follows

A.1 Feature extraction code

```
function [featureTable,outputTable] = diagnosticFeatures(inputData)
%DIAGNOSTICFEATURES recreates results in Diagnostic Feature Designer.
%
% Input:
%  inputData: A table or a cell array of tables/matrices containing the
%  data as those imported into the app.
%
% Output:
%  featureTable: A table containing all features and condition variables.
%  outputTable: A table containing the computation results.
%
% This function computes features:
%  Signal_stats/SignalData_ClearanceFactor
%  Signal_stats/SignalData_CrestFactor
%  Signal_stats/SignalData_ImpulseFactor
%  Signal_stats/SignalData_Kurtosis
%  Signal_stats/SignalData_Mean
%  Signal_stats/SignalData_PeakValue
%  Signal_stats/SignalData_RMS
%  Signal_stats/SignalData_SINAD
%  Signal_stats/SignalData_SNR
%  Signal_stats/SignalData_ShapeFactor
%  Signal_stats/SignalData_Skewness
%  Signal_stats/SignalData_Std
%  Signal_stats/SignalData_THD
%
% Organization of the function:
% 1. Compute signals/spectra/features
% 2. Extract computed features into a table
%
% Modify the function to add or remove data processing, feature generation
% or ranking operations.

% Auto-generated by MATLAB on 04-Jan-2021 14:01:48

% Create output ensemble.
outputEnsemble =
workspaceEnsemble(inputData,'DataVariables',"Signal",'ConditionVariables',"L
abels");

% Reset the ensemble to read from the beginning of the ensemble.
reset(outputEnsemble);
```

```
% Append new signal or feature names to DataVariables.
outputEnsemble.DataVariables =
unique([outputEnsemble.DataVariables;"Signal_stats"],'stable');

% Set SelectedVariables to select variables to read from the ensemble.
outputEnsemble.SelectedVariables = "Signal";

% Loop through all ensemble members to read and write data.
while hasdata(outputEnsemble)
    % Read one member.
    member = read(outputEnsemble);

    % Get all input variables.
    Signal = readMemberData(member,"Signal",["Time","SignalData"]);

    % Initialize a table to store results.
    memberResult = table;

    %% SignalFeatures
    try
        % Compute signal features.
        inputSignal = Signal.SignalData;
        SignalData_ClearanceFactor =
max(abs(inputSignal))/(mean(sqrt(abs(inputSignal))))^2);
        SignalData_CrestFactor = peak2rms(inputSignal);
        SignalData_ImpulseFactor =
max(abs(inputSignal))/mean(abs(inputSignal));
        SignalData_Kurtosis = kurtosis(inputSignal);
        SignalData_Mean = mean(inputSignal,'omitnan');
        SignalData_PeakValue = max(abs(inputSignal));
        SignalData_RMS = rms(inputSignal,'omitnan');
        SignalData_SINAD = sinad(inputSignal);
        SignalData_SNR = snr(inputSignal);
        SignalData_ShapeFactor =
rms(inputSignal,'omitnan')/mean(abs(inputSignal),'omitnan');
        SignalData_Skewness = skewness(inputSignal);
        SignalData_Std = std(inputSignal,'omitnan');
        SignalData_THD = thd(inputSignal);

        % Concatenate signal features.
        featureValues =
[SignalData_ClearanceFactor,SignalData_CrestFactor,SignalData_ImpulseFactor,
SignalData_Kurtosis,SignalData_Mean,SignalData_PeakValue,SignalData_RMS,Sign
alData_SINAD,SignalData_SNR,SignalData_ShapeFactor,SignalData_Skewness,Signa
lData_Std,SignalData_THD];

        % Package computed features into a table.
        featureNames =
["SignalData_ClearanceFactor","SignalData_CrestFactor","SignalData_ImpulseFa
ctor","SignalData_Kurtosis","SignalData_Mean","SignalData_PeakValue","Signal
Data_RMS","SignalData_SINAD","SignalData_SNR","SignalData_ShapeFactor","Sign
alData_Skewness","SignalData_Std","SignalData_THD"];
        Signal_stats =
array2table(featureValues,'VariableNames',featureNames);
    catch
```

```
        % Package computed features into a table.
        featureValues = NaN(1,13);
        featureNames =
["SignalData_ClearanceFactor","SignalData_CrestFactor","SignalData_ImpulseFa
ctor","SignalData_Kurtosis","SignalData_Mean","SignalData_PeakValue","Signal
Data_RMS","SignalData_SINAD","SignalData_SNR","SignalData_ShapeFactor","Sign
alData_Skewness","SignalData_Std","SignalData_THD"];
        Signal_stats =
array2table(featureValues,'VariableNames',featureNames);
    end

    % Append computed results to the member table.
    memberResult = [memberResult, ...
        table({Signal_stats},'VariableNames',"Signal_stats")]; %#ok<AGROW>

    %% Write all the results for the current member to the ensemble.
    writeToLastMemberRead(outputEnsemble,memberResult)
end

% Gather all features into a table.
featureTable = readFeatureTable(outputEnsemble);

% Set SelectedVariables to select variables to read from the ensemble.
outputEnsemble.SelectedVariables =
unique([outputEnsemble.DataVariables;outputEnsemble.ConditionVariables;outpu
tEnsemble.IndependentVariables],'stable');

% Gather results into a table.
outputTable = readall(outputEnsemble);
end
```

A.2 The first phase code

```
 function [trainedClassifier, validationAccuracy] =
trainClassifier(trainingData)
% [trainedClassifier, validationAccuracy] = trainClassifier(trainingData)
% Returns a trained classifier and its accuracy. This code recreates the
% classification model trained in Classification Learner app. Use the
% generated code to automate training the same model with new data, or to
% learn how to programmatically train models.
%
%  Input:
%      trainingData: A table containing the same predictor and response
%       columns as those imported into the app.
%
%  Output:
%      trainedClassifier: A struct containing the trained classifier. The
%       struct contains various fields with information about the trained
%       classifier.
%
%      trainedClassifier.predictFcn: A function to make predictions on new
```

```
%       data.
%
%      validationAccuracy: A double containing the accuracy in percent. In
%       the app, the History list displays this overall accuracy score for
%       each model.
%
% Use the code to train the model with new data. To retrain your
% classifier, call the function from the command line with your original
% data or new data as the input argument trainingData.
%
% For example, to retrain a classifier trained with the original data set
% T, enter:
%   [trainedClassifier, validationAccuracy] = trainClassifier(T)
%
% To make predictions with the returned 'trainedClassifier' on new data T2,
% use
%   yfit = trainedClassifier.predictFcn(T2)
%
% T2 must be a table containing at least the same predictor columns as used
% during training. For details, enter:
%   trainedClassifier.HowToPredict

% Auto-generated by MATLAB on 04-Jan-2021 15:40:14


% Extract predictors and response
% This code processes the data into the right shape for training the
% model.
inputTable = trainingData;
predictorNames = {'Signal_stats/SignalData_ClearanceFactor',
'Signal_stats/SignalData_CrestFactor',
'Signal_stats/SignalData_ImpulseFactor', 'Signal_stats/SignalData_Kurtosis',
'Signal_stats/SignalData_Mean', 'Signal_stats/SignalData_PeakValue',
'Signal_stats/SignalData_RMS', 'Signal_stats/SignalData_SINAD',
'Signal_stats/SignalData_SNR', 'Signal_stats/SignalData_ShapeFactor',
'Signal_stats/SignalData_Skewness', 'Signal_stats/SignalData_Std',
'Signal_stats/SignalData_THD'};
predictors = inputTable(:, predictorNames);
response = inputTable.Labels;
isCategoricalPredictor = [false, false, false, false, false, false, false,
false, false, false, false, false, false];

% Train a classifier
% This code specifies all the classifier options and trains the classifier.
classificationTree = fitctree(...
    predictors, ...
    response, ...
    'SplitCriterion', 'gdi', ...
    'MaxNumSplits', 100, ...
    'Surrogate', 'off', ...
    'ClassNames', {'ARR'; 'CHF'; 'NSR'});

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
treePredictFcn = @(x) predict(classificationTree, x);
```

```
trainedClassifier.predictFcn = @(x)
treePredictFcn(predictorExtractionFcn(x));

% Add additional fields to the result struct
trainedClassifier.RequiredVariables =
{'Signal_stats/SignalData_ClearanceFactor',
'Signal_stats/SignalData_CrestFactor',
'Signal_stats/SignalData_ImpulseFactor', 'Signal_stats/SignalData_Kurtosis',
'Signal_stats/SignalData_Mean', 'Signal_stats/SignalData_PeakValue',
'Signal_stats/SignalData_RMS', 'Signal_stats/SignalData_SINAD',
'Signal_stats/SignalData_SNR', 'Signal_stats/SignalData_ShapeFactor',
'Signal_stats/SignalData_Skewness', 'Signal_stats/SignalData_Std',
'Signal_stats/SignalData_THD'};
trainedClassifier.ClassificationTree = classificationTree;
trainedClassifier.About = 'This struct is a trained model exported from
Classification Learner R2020a.';
trainedClassifier.HowToPredict = sprintf('To make predictions on a new
table, T, use: \n  yfit = c.predictFcn(T) \nreplacing ''c'' with the name of
the variable that is this struct, e.g. ''trainedModel''. \n \nThe table, T,
must contain the variables returned by: \n  c.RequiredVariables \nVariable
formats (e.g. matrix/vector, datatype) must match the original training
data. \nAdditional variables are ignored. \n \nFor more information, see <a
href="matlab:helpview(fullfile(docroot, ''stats'', ''stats.map''),
''appclassification_exportmodeltoworkspace'')">How to predict using an
exported model</a>.');

% Extract predictors and response
% This code processes the data into the right shape for training the
% model.
inputTable = trainingData;
predictorNames = {'Signal_stats/SignalData_ClearanceFactor',
'Signal_stats/SignalData_CrestFactor',
'Signal_stats/SignalData_ImpulseFactor', 'Signal_stats/SignalData_Kurtosis',
'Signal_stats/SignalData_Mean', 'Signal_stats/SignalData_PeakValue',
'Signal_stats/SignalData_RMS', 'Signal_stats/SignalData_SINAD',
'Signal_stats/SignalData_SNR', 'Signal_stats/SignalData_ShapeFactor',
'Signal_stats/SignalData_Skewness', 'Signal_stats/SignalData_Std',
'Signal_stats/SignalData_THD'};
predictors = inputTable(:, predictorNames);
response = inputTable.Labels;
isCategoricalPredictor = [false, false, false, false, false, false, false,
false, false, false, false, false, false];

% Perform cross-validation
partitionedModel = crossval(trainedClassifier.ClassificationTree, 'KFold',
5);

% Compute validation predictions
[validationPredictions, validationScores] = kfoldPredict(partitionedModel);

% Compute validation accuracy
validationAccuracy = 1 - kfoldLoss(partitionedModel, 'LossFun',
'ClassifError');
```

A.3 The second phase code

```
function [trainedClassifier, validationAccuracy] =
trainClassifier(trainingData)
% [trainedClassifier, validationAccuracy] = trainClassifier(trainingData)
% Returns a trained classifier and its accuracy. This code recreates the
% classification model trained in Classification Learner app. Use the
% generated code to automate training the same model with new data, or to
% learn how to programmatically train models.
%
%  Input:
%      trainingData: A table containing the same predictor and response
%       columns as those imported into the app.
%
%  Output:
%      trainedClassifier: A struct containing the trained classifier. The
%       struct contains various fields with information about the trained
%       classifier.
%
%      trainedClassifier.predictFcn: A function to make predictions on new
%       data.
%
%      validationAccuracy: A double containing the accuracy in percent. In
%       the app, the History list displays this overall accuracy score for
%       each model.
%
% Use the code to train the model with new data. To retrain your
% classifier, call the function from the command line with your original
% data or new data as the input argument trainingData.
%
% For example, to retrain a classifier trained with the original data set
% T, enter:
%   [trainedClassifier, validationAccuracy] = trainClassifier(T)
%
% To make predictions with the returned 'trainedClassifier' on new data T2,
% use
%   yfit = trainedClassifier.predictFcn(T2)
%
% T2 must be a table containing at least the same predictor columns as used
% during training. For details, enter:
%   trainedClassifier.HowToPredict

% Auto-generated by MATLAB on 04-Jan-2021 18:15:16



% Extract predictors and response
% This code processes the data into the right shape for training the
% model.
inputTable = trainingData;
predictorNames = {'Signal_stats/SignalData_ClearanceFactor',
'Signal_stats/SignalData_CrestFactor',
'Signal_stats/SignalData_ImpulseFactor', 'Signal_stats/SignalData_Kurtosis',
```

```
'Signal_stats/SignalData_Mean', 'Signal_stats/SignalData_PeakValue',
'Signal_stats/SignalData_RMS', 'Signal_stats/SignalData_SINAD',
'Signal_stats/SignalData_SNR', 'Signal_stats/SignalData_ShapeFactor',
'Signal_stats/SignalData_Skewness', 'Signal_stats/SignalData_Std',
'Signal_stats/SignalData_THD'};
predictors = inputTable(:, predictorNames);
response = inputTable.Labels;
isCategoricalPredictor = [false, false, false, false, false, false, false,
false, false, false, false, false, false];

% Train a classifier
% This code specifies all the classifier options and trains the classifier.
classificationTree = fitctree(...
    predictors, ...
    response, ...
    'SplitCriterion', 'deviance', ...
    'MaxNumSplits', 8, ...
    'Surrogate', 'off', ...
    'ClassNames', {'ARR'; 'CHF'; 'NSR'});

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
treePredictFcn = @(x) predict(classificationTree, x);
trainedClassifier.predictFcn = @(x)
treePredictFcn(predictorExtractionFcn(x));

% Add additional fields to the result struct
trainedClassifier.RequiredVariables =
{'Signal_stats/SignalData_ClearanceFactor',
'Signal_stats/SignalData_CrestFactor',
'Signal_stats/SignalData_ImpulseFactor', 'Signal_stats/SignalData_Kurtosis',
'Signal_stats/SignalData_Mean', 'Signal_stats/SignalData_PeakValue',
'Signal_stats/SignalData_RMS', 'Signal_stats/SignalData_SINAD',
'Signal_stats/SignalData_SNR', 'Signal_stats/SignalData_ShapeFactor',
'Signal_stats/SignalData_Skewness', 'Signal_stats/SignalData_Std',
'Signal_stats/SignalData_THD'};
trainedClassifier.ClassificationTree = classificationTree;
trainedClassifier.About = 'This struct is a trained model exported from
Classification Learner R2020a.';
trainedClassifier.HowToPredict = sprintf('To make predictions on a new
table, T, use: \n  yfit = c.predictFcn(T) \nreplacing ''c'' with the name of
the variable that is this struct, e.g. ''trainedModel''. \n \nThe table, T,
must contain the variables returned by: \n  c.RequiredVariables \nVariable
formats (e.g. matrix/vector, datatype) must match the original training
data. \nAdditional variables are ignored. \n \nFor more information, see <a
href="matlab:helpview(fullfile(docroot, ''stats'', ''stats.map''),
''appclassification_exportmodeltoworkspace'')">How to predict using an
exported model</a>.');

% Extract predictors and response
% This code processes the data into the right shape for training the
% model.
inputTable = trainingData;
predictorNames = {'Signal_stats/SignalData_ClearanceFactor',
'Signal_stats/SignalData_CrestFactor',
'Signal_stats/SignalData_ImpulseFactor', 'Signal_stats/SignalData_Kurtosis',
```

```
'Signal_stats/SignalData_Mean', 'Signal_stats/SignalData_PeakValue',
'Signal_stats/SignalData_RMS', 'Signal_stats/SignalData_SINAD',
'Signal_stats/SignalData_SNR', 'Signal_stats/SignalData_ShapeFactor',
'Signal_stats/SignalData_Skewness', 'Signal_stats/SignalData_Std',
'Signal_stats/SignalData_THD'};
predictors = inputTable(:, predictorNames);
response = inputTable.Labels;
isCategoricalPredictor = [false, false, false, false, false, false, false,
false, false, false, false, false, false];

% Perform cross-validation
partitionedModel = crossval(trainedClassifier.ClassificationTree, 'KFold',
5);

% Compute validation predictions
[validationPredictions, validationScores] = kfoldPredict(partitionedModel);

% Compute validation accuracy
validationAccuracy = 1 - kfoldLoss(partitionedModel, 'LossFun',
'ClassifError');
```