CLASSIFYING ELECTROCARDIOGRAM WITH MACHINE LEARNING TECHNIQUES

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Biomedical Engineering

by

Hillal Jarrar

December 2021

COMMITTEE MEMBERSHIP

TITLE:	Classifying Electrocardiogram with Machine Learning Techniques
AUTHOR:	Hillal Jarrar
DATE SUBMITTED:	December 2021
COMMITTEE CHAIR:	Robert Szlavik, Ph.D. Professor of Biomedical Engineering
COMMITTEE MEMBER:	Michael Whitt, Ph.D. Assistant Professor of Biomedical Engineering
COMMITTEE MEMBER:	Ben Hawkins, Ph.D. Assistant Professor of Biomedical and Electrical Engineering

Abstract

Classifying Electrocardiogram with Machine Learning Techniques

Hillal Jarrar

Classifying the electrocardiogram is of clinical importance because classification can be used to diagnose patients with cardiac arrhythmias. Many industries utilize machine learning techniques that consist of feature extraction methods followed by Naive-Bayesian classification in order to detect faults within machinery. Machine learning techniques that analyze vibrational machine data in a mechanical application may be used to analyze electrical data in a physiological application. Three of the most common feature extraction methods used to prepare machine vibration data for Naive-Bayesian classification are the Fourier transform, the Hilbert transform, and the Wavelet Packet transform. Each machine learning technique consists of a different feature extraction method to prepare the data for Naive-Bayesian classification. The effectiveness of the different machine learning techniques, when applied to electrocardiogram, is assessed by measuring the sensitivity and specificity of the classifications. Comparing the sensitivity and specificity of each machine learning technique to the other techniques revealed that the Wavelet Packet transform, followed by Naïve-Bayesian classification, is the most effective machine learning technique.

ACKNOWLEDGMENTS

Thanks to:

- Mohammad Jarrar, M.S., for inspiring this thesis

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ECG	Electrocardiogram
HOS	Higher Order Statistics
SCM	Structural Co-Occurrence Matrix
SA	Sinoatrial
AV	Atrioventricular
RA	Right Atrium
RV	Right Ventricle
LA	Left Atrium
LV	Left Ventricle
SVC	Superior Vena Cava
IVC	Inferior Vena Cava
T	Tricuspid
P	Pulmonic
PA	Pulmonary Artery
PV	Pulmonary Veins
M	Mitral
A	Aortic

Chapter 1
INTRODUCTION

The human body is among the most complex machines in existence. The heart is among the most important organs in the human body. The human heart consists of two to three billion cardiac muscle cells [1]. These cells make up about 99% of the cells in the chambers of the heart. The cardiac muscle cells contract and pump blood throughout the body in response to the myocardial conducting cells. The myocardial conducting cells make up about 1% of the cells in the chambers of the heart [2]. The conducting cells use ion gradients to initiate and maintain their own electrical impulses that are propagated to the contractile muscle cells. The propagation of the electric signal occurs in an extremely coordinated manner to ensure that the different chambers of the heart contract completely and in the proper order. This coordinated communication occurs approximately 60 to 170 times per minute, uninterrupted for the entirety of a human life span [3].

The purpose of this work is to present a method of detecting faults in the electrical communication of the heart and to determine the most effective machine learning technique to obtain the most accurate results of cardiac electrophysiologic fault detection.

1.1 Motivation

475,000 Americans die from cardiac arrest, annually. Globally, more people die from cardiac arrest than from colorectal cancer, breast cancer, prostate cancer, influenza, pneumonia, auto accidents, HIV, firearms, and house fires combined [4].

The feature extraction methods and classification methods used in mechanical industries can be applied to the field of cardiac health. These methods have successfully been applied to motor engines and other mechanical instruments. If the human heart is

treated as an intricate mechanical device, these methods can also be applied to humans. This creates a more effective method of diagnosing cardiac arrhythmias in a rapid and efficient manner.

The feature extraction and classification methods may be applied to a hospital setting in which the healthcare provider will apply these methods while measuring a patient's electrocardiogram (ECG). This can also be applied to the wider population. An ECG measuring device may be supplied in all public buildings, similar to an AED. This ECG measuring device can have the feature extraction and classification algorithms built into the device. Following a provided list of instructions, anyone can apply the device to an individual experiencing chest pain or other signs of cardiac arrest. The device will classify the patient's ECG and alert the user if an ambulance must be called and if CPR must be applied.

Without classification of ECG, a victim of cardiac arrest will not receive medical intervention until cardiac arrest occurs. If the patient has their ECG classified prior to cardiac arrest, they will be more likely to receive medical intervention. Application of machine learning techniques- consisting of feature extraction followed by feature classification- to cardiac health will save innumerable lives.

1.2 Previous Work

In the past several years, numerous studies have applied machine learning techniques to diagnose cardiac health.

In 2019, Leonardo B. Marinho et al. published a novel ECG feature extraction method for arrhythmia classification. This study used four feature extraction algorithms: Fourier transform, Goertzel Extraction, Higher Order Statistics (HOS), and Structural Co-Occurrence Matrix (SCM). The feature extraction algorithms were trained using a segment of a patient's ECG data. The patient's ECG data was then classified using the trained algorithm [5].

The process that was used in the study first filtered the raw ECG by determining a minimum peak prominence. The filtered signal was then segmented. Each segment was analyzed by the trained algorithm to determine amplitudes of each peak. Figure 1.1 depicts the process for preparing ECG data for classification. The features were classified using classification algorithms. Four classification algorithms were used: Naive-Bayes classifier, Multi-Layer Perceptron (MLP), Optimum-Path Forest (OPF), and Support Vector Machine (SVM) [5].

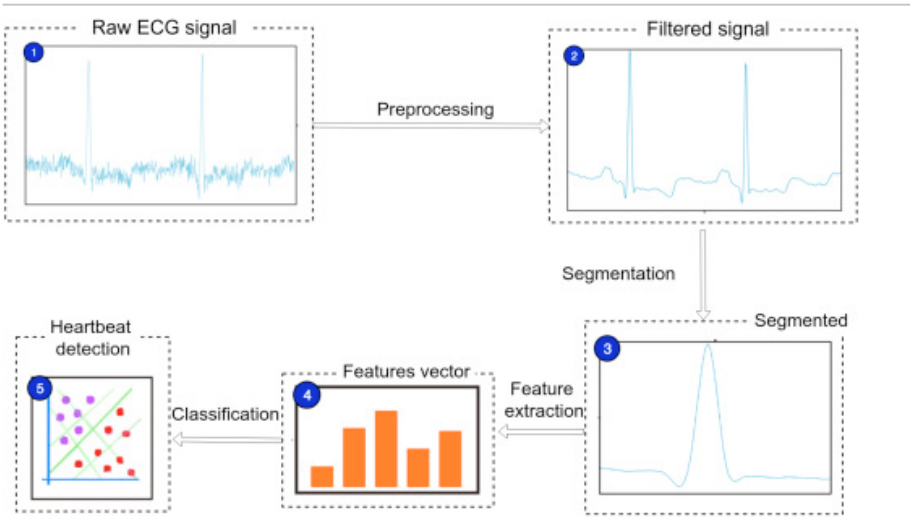

Figure 1.1 Process for Preparing ECG Data for Classification: The process consists of removing noise from the raw ECG, segmenting the ECG, feature extraction, and feature classification [5].

This study identified Naive-Bayesian classifier and HOS as the most effective classifiers [5]. Additionally, the Naive-Bayes classifier is the most commonly used classifier in machine learning techniques. Therefore, this thesis will use the Naive-Bayesian classifier.

1.3 Outline

This document is structured as follows. A background of cardiac electrophysiology is provided. Then the function of three feature extraction methods is described. The feature extraction methods are the Fourier transform, the Hilbert transform, and the Wavelet Packet transform. Next, the Naive-Bayesian classifier is described. These three feature extraction methods, followed by Naïve-Bayesian classification, are the most commonly used machine learning techniques in industry. Therefore, these feature extraction methods will be assessed in this document. Finally, the document describes the results yielded by these feature extraction and classifier methods.

Chapter 2
BACKGROUND

2.1 Cardiac Anatomy

The human heart consists of four chambers: the right atrium (RA), the right ventricle (RV), the left atrium (LA), and the left ventricle (LV). Deoxygenated blood from the venous system enters the RA through two veins called the superior vena cava (SVC) and inferior vena cava (IVC). The blood flows from the RA, through the tricuspid (T) valve and into the RV. The RV contracts and the blood passes through the pulmonic (P) valve and through the pulmonary artery (PA). The PA distributes the blood into the lungs to be reoxygenated. Through four pulmonary veins (PV), oxygenated blood enters the LA. From the LA, blood flows through the mitral (M) valve and into the LV. The left ventricle contracts and blood is pushed through the aortic (A) valve and into the aorta. The aorta delivers the oxygenated blood throughout the arterial system and the body's cells are oxygenated. Figure 2.1 depicts cardiac anatomy [6].

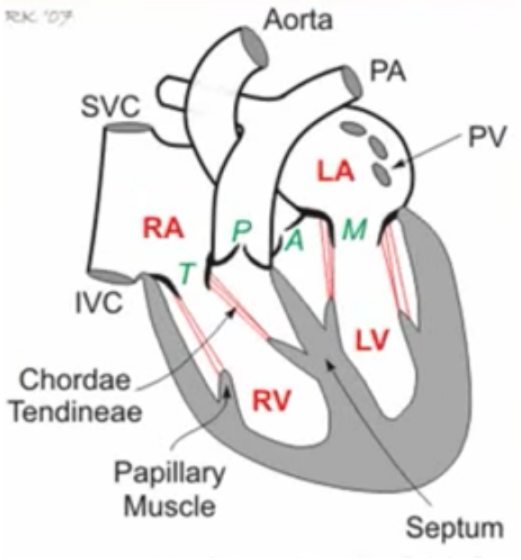

Figure 2.1 Cardiac Anatomy: The human heart consists of four chambers through which deoxygenated blood is delivered to the lungs to be reoxygenated and reoxygenated blood is delivered to the body's muscles [6].

2.2 Cardiac Conduction System

Throughout the walls of the heart's chambers are specialized muscle cells that are capable of transferring electrical signals, as depicted in the figure below. These cells connect into an intricate system consisting of the sinoatrial (SA) node, atrioventricular (AV) node, the Bundle of His, Bundle branches, and the Purkinje Fibers [7]. The cardiac conduction system is depicted in Figure 2.2.

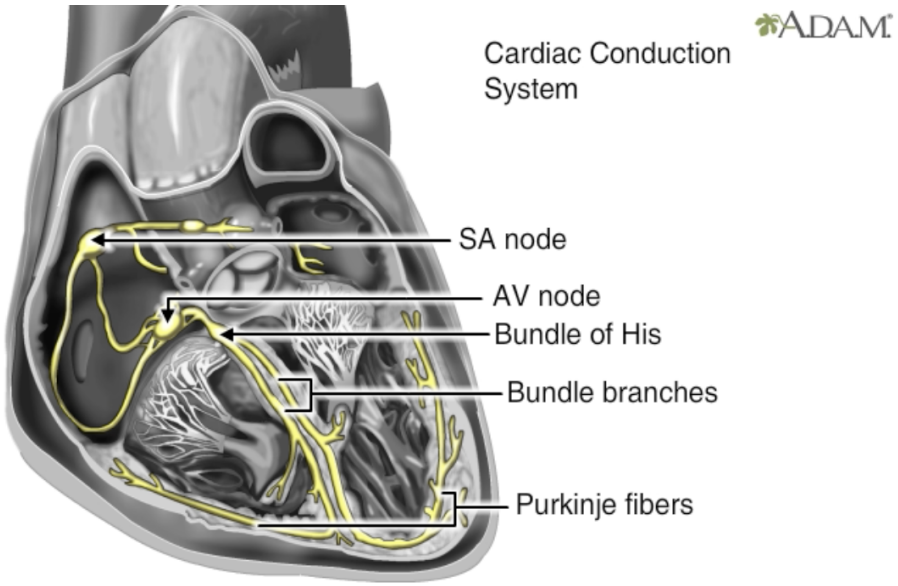

Figure 2.2 Cardiac Conduction System: The electric signal begins in the SA node and travels though the heart's neurons to the AV node, Bundle of His, Bundle branches, and Purkinje fibers [7].

The electric current travels through neurons in the walls of the heart's chambers. The sequence begins with the SA node. The electric signal from the SA node causes the muscle cells in the atria to contract. When the atria contract, blood is pumped into the ventricles. The signal then travels to the AV node, then to the Bundle of His, the Bundle branches, and finally to the Purkinje Fibers. The signal that passes through the Purkinje Fibers causes the ventricles to contract. When the ventricles contract, blood is pumped out of the heart and into the lungs and the rest of the body [7].

Contraction of the cardiac muscle cells causes the contraction of the heart's chambers. Cardiac muscle cells are striated and heavily branched. They are connected into structures called sarcomeres. Cardiac muscle cells contain one nucleus and produce

energy by aerobic respiration. These cells are constantly contracting and consuming energy. Therefore, cardiac muscle cells contain mitochondria and myoglobin to produce the necessary energy in the form of Adenosine triphosphate (ATP) Figure 2.3 depicts the cardiac muscle cells [8].

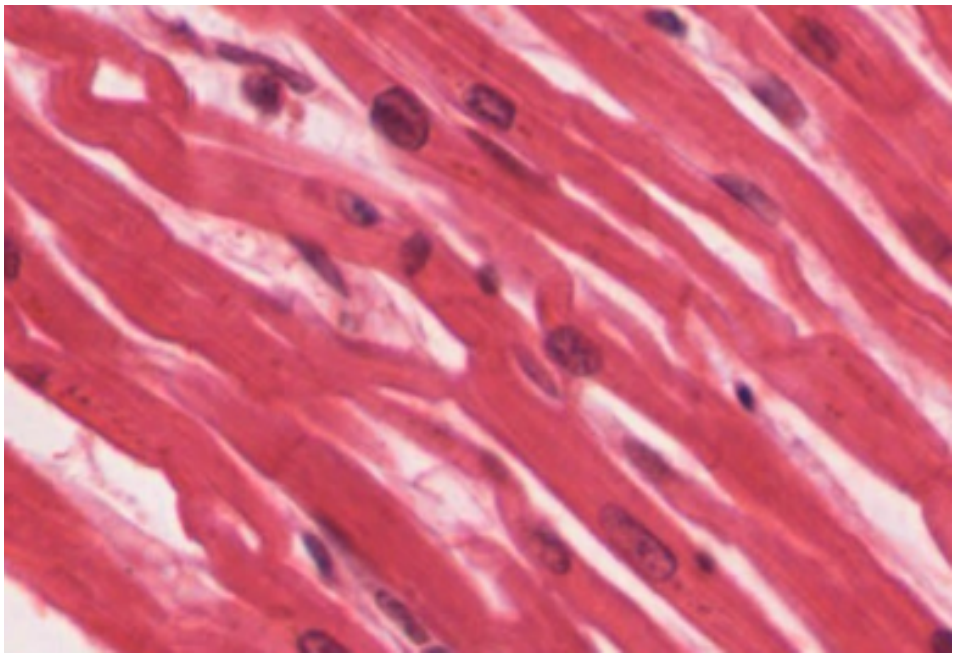

Figure 2.3 Cardiac Muscle Cells: The cardiac muscle cells are striated and heavily branched [8].

T cells are connected together by intercalated disks. Within each intercalated disk are gap junctions and desmosomes. The gap junctions allow the electrical current to travel between cells. Desmosomes are structures that serve as anchors between cardiac muscle cells and ensure that the cells do not disconnect as they contract. Figure 2.4 depicts cardiac muscle cells connected by intercalated disks [8].

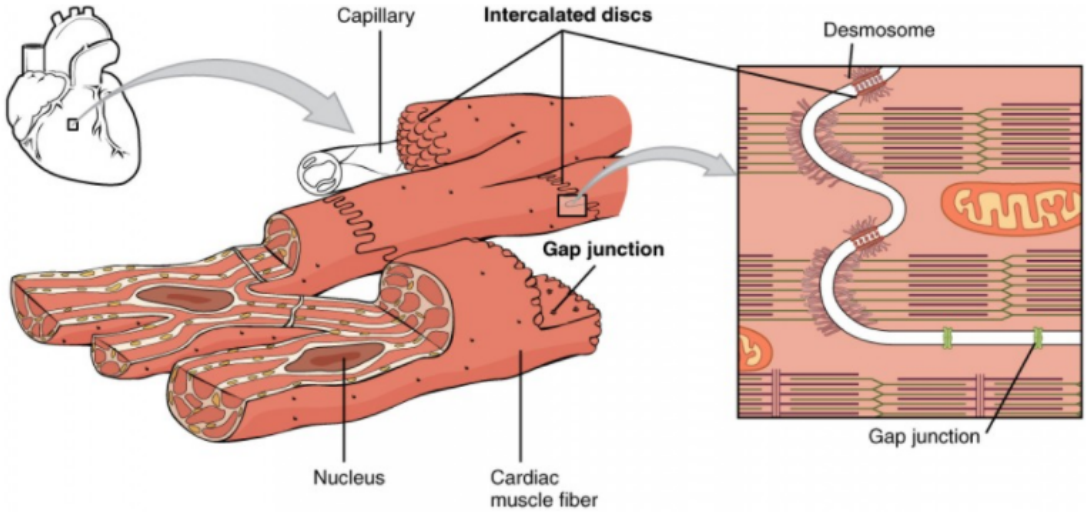

Figure 2.4 Cardiac Muscle Cells Connected by Intercalated Disks: The intercalated disks allow the cardiac muscle cells to transmit electric signals and remain attached during contraction [8].

Cardiac muscle cells contract as their cell walls depolarize. In a non-contracted resting state, cardiac muscle cells are highly polarized with a net negative charge within the cell membrane and a net positive charge outside of the cell membrane. Depolarization occurs with the opening of sodium ion channels in the cell membrane and the rapid influx of sodium ions into the cell. Figure 2.5 depicts ion channels in the membrane of cardiac muscle cells.

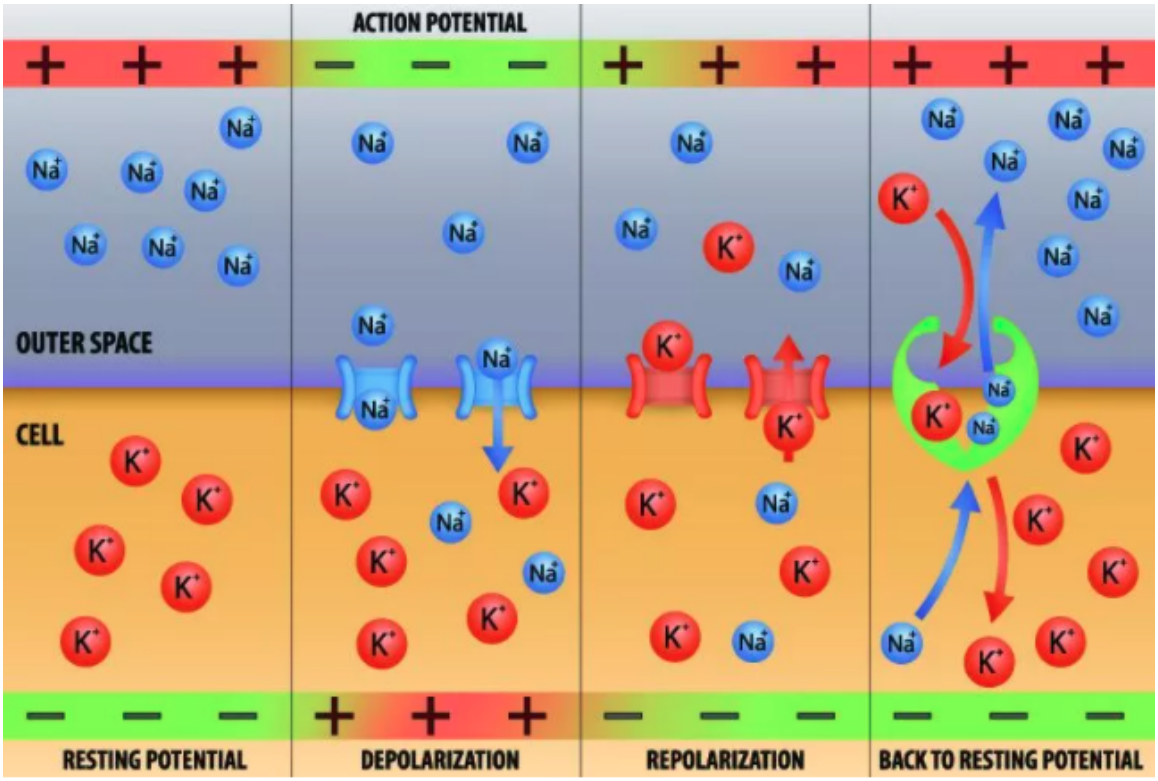

Figure 2.5 Ion Channels in the Membrane of Cardiac Muscle Cells: The ion channels enable the depolarization and repolarization of cardiac muscle cells [9].

The rapid influx of sodium ions results in a net positive charge within the cell relative to the outside the cell. Depolarization of the cell results in the binding of a molecule called myosin to ATP. Muscle fibers are attached to structures called actin filaments. Attached to the actin filaments are myosin molecules. The binding of myosin to ATP pulls the actin filaments towards the center of the sarcomere. This causes the cell to contract. Potassium ion channels in the cell membrane then open. A partial repolarization occurs as relatively small amounts of potassium ions leave the cell. Next, a steady state is achieved with the opening of calcium ion channels which allows for the influx of calcium ions as potassium ions continue to leave the cell membrane. Complete repolarization occurs when the calcium ion channels close and potassium ions continue to

leave the cell. Resting potential is achieved when both sodium and calcium ion channels have closed. Figure 2.6 depicts cardiac muscle cell polarization and depolarization [9].

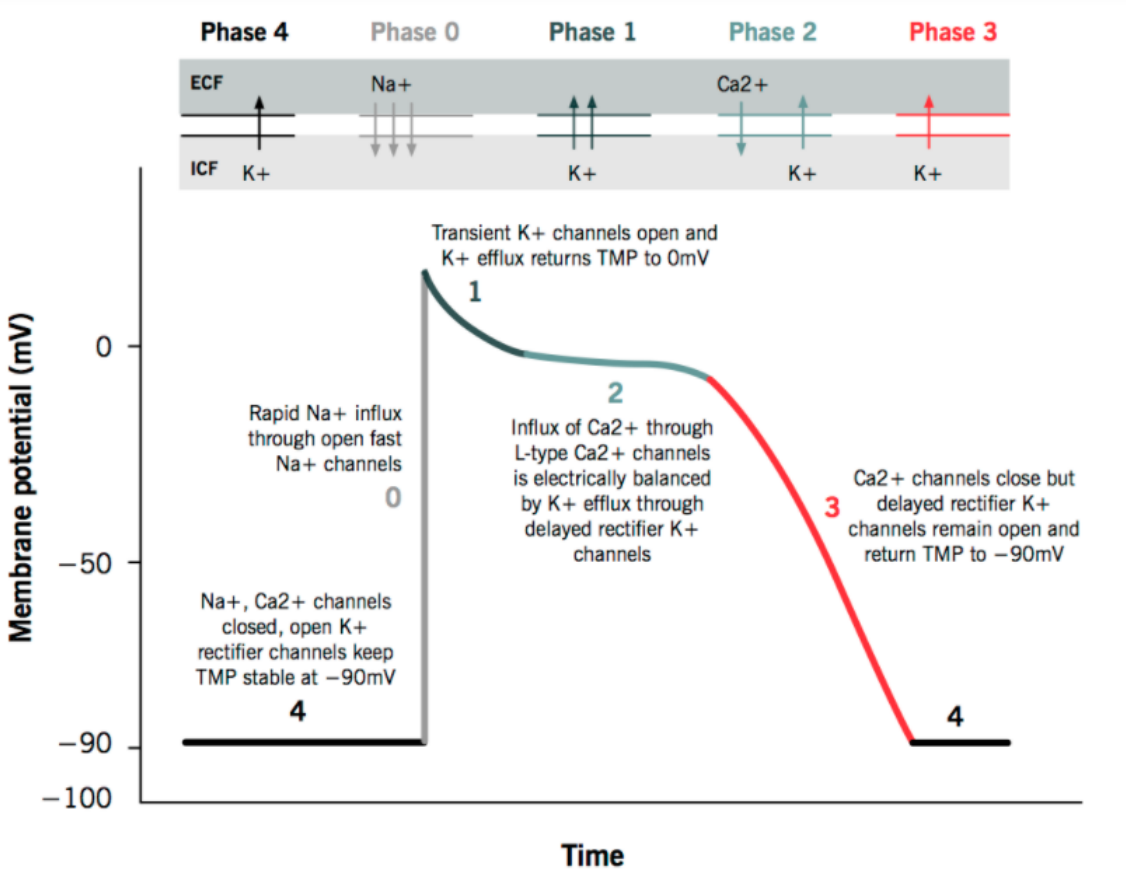

Figure 2.6 Cardiac Muscle Cell Polarization and Depolarization: Depolarization occurs with the inflow of sodium ions into the cell and repolarization occurs with outflow of potassium ions [10].

2.3 Machine Learning

The current method of treating cardiac arrest involves medical intervention when the patient is experiencing extreme chest discomfort immediately before cardiac arrest or

medical intervention immediately after cardiac arrest occurs. The current method of treating cardiac arrest reduces the patient's chance of survival. With machine learning, the patient may have their ECG classified when they first experience chest pain and will be more likely to receive medical intervention well before cardiac arrest. The early detection offered by machine learning may significantly increase the patient's chance of survival.

Machine learning is the development of systems that can analyze data and learn from the data. It consists of two parts: feature extraction and feature classification. The purpose of machine learning is to identify patterns and abnormalities within a dataset and make decisions based on the observations with minimal human intervention. Systems to identify and learn from patterns in a particular dataset are able to adapt and learn from a new dataset without having to be reprogrammed. Machine learning is an integral part of the modern economy, playing a role in online search recommendations, fault detection in manufacturing lines, and fraud detection in financial transactions. As datasets from all industries become larger, the modern economy will only rely more on machine learning [11].

In manufacturing, engineers have a very clear goal, which is to produce more products at a higher quality and with lower cost. Machine learning helps achieve this goal by serving an important role within manufacturing, which is predictive maintenance. Predictive maintenance is the ability to determine when a machine is likely to fail. When the likelihood of failure is detected to be high, the machine will be repaired. Predictive maintenance lowers production costs because it eliminates the need for regular scheduled maintenance. Regular scheduled maintenance results in repairing machines even when no

repair is necessary. Predictive maintenance also lowers cost by detecting faults and eliciting repair prior to the complete and sudden failure of a machine. Production is able to continue without the unexpected interruption of machine failure [12].

These machine learning techniques may be applied to human health. A patient's physiologic data can be collected and analyzed. The detection of potential faults will elicit medical intervention.

2.4 ECG Signal

ECG signals are measured by the placement of electrodes on the surface of the patient's skin. Electrodes are conductive material that connect an electric circuit to a non-metallic material. When measuring ECG, the electrodes typically consist of silver and silver-chloride half-cells. The difference in electric potential between the half-cells enables electrons to flow between the patient's skin and the electrodes. Twelve electrodes are placed throughout the patient's body to measure the electric current of the heart [6].

A healthy ECG produces a PQRST wave when measuring millivolts as a function of time. The P portion of the wave is produced when the SA node emits a current and is indicative of atrial depolarization. The QRS portion of the wave is produced by the AV node and is indicative of ventricular depolarization. The T portion of the wave is indicative of atrial repolarization. Figure 2.7 depicts the PQRST wave of a normal ECG [6].

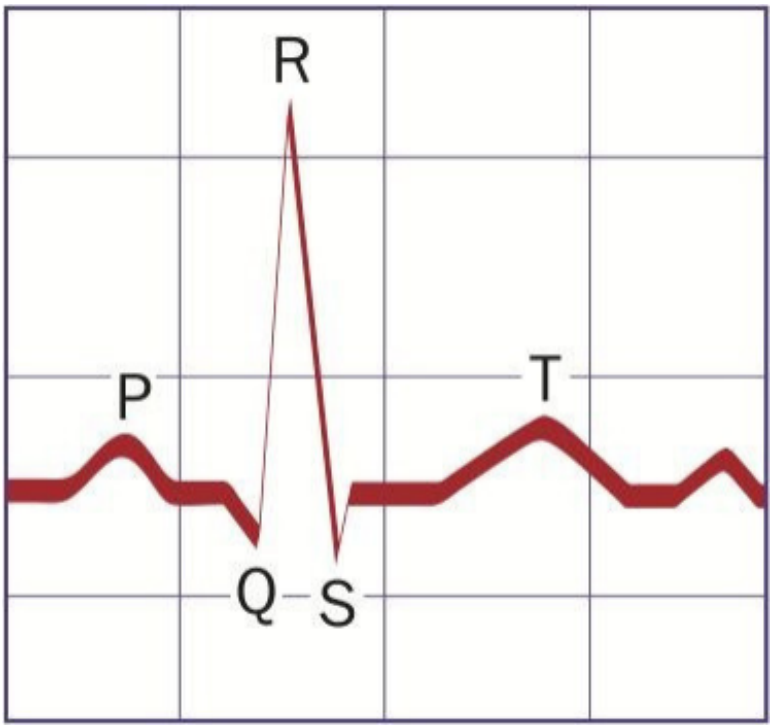

Figure 2.7 PQRST Wave of a Normal ECG: The wave produced by a healthy ECG consists of 5 prominent features: The P, R, and T crests and the Q and S troughs [6].

Discrepancies from the normal waveform of the PQRST wave are referred to as arrhythmias. The most common forms of arrhythmia are bradycardias, tachycardia, supraventricular arrhythmias, and ventricular arrhythmias. Bradycardia occurs when the PQRST wave occurs at a frequency that is too low. Tachycardia occurs when the PQRST wave occurs at a frequency that is too high. Supraventricular arrhythmias are arrhythmias that occur in the atria. Ventricular arrhythmias are arrhythmias that occur in the ventricles [6].

Machine learning algorithms do not diagnose a patient with a specific arrhythmia. Rather, machine learning algorithms detect when any form of arrhythmia occurs and labels the segment of the ECG at which the arrhythmia occurs as a fault.

Common clinical metrics, such as elongated QRS time and ST elevation are two symptoms that may be detected by machine learning. A longer QRS interval indicates a decreased distance between the QRS segment and the P segment preceding it or the T segment proceeding it. ST elevation indicates a peak amplitude that is higher than what is normal. When extracting features from an ECG with these symptoms, the elongated QRS segment will result in a greater instance of extracted features at high frequencies because of the decreased distance between peaks. For example, a Fourier transform plot will display greater peak amplitudes at lower frequencies as compared to a healthy ECG. In the case of ST elevation, there will be a greater instance of extracted features with high amplitudes.

However, if the patient is displaying elongated QRS time and ST elevation in every heart contraction cycle, the machine learning algorithm will not detect these symptoms as faults. The machine learning algorithm will only detect these symptoms as faults if they occur in a portion of the patient's ECG or if it was trained with the patient's ECG at an earlier point in time in which the ECG was healthy.

Machine learning may display similar accuracy when applied to males and females. On average, males and females display different ECG features. Females tend to display a shorter PR interval and shorter QRS duration as compared to males. The Naïve-Bayesian classifier is trained using the patient's own ECG. Machine learning will test the

patient's ECG based on the patterns it learned from a prior portion of the ECG. Therefore, the shorter PR interval and QRS duration in a female patient will be detected as the regular pattern and will not be classified as faults during testing.

Chapter 3
METHODS

3.1 Data Acquisition

The ECG datasets used for this thesis were obtained from Physionet.org, an open source forum that offers large physiologic datasets. The ECG datasets were collected from 1975 to 1979, at the Beth Israel Hospital in Boston, Massachusetts (currently known as the Beth Israel Deaconess Medical Center) and at the Massachusetts Institute of Technology. Data from 47 patients at the Beth Israel Hospital Arrhythmia Laboratory were compiled and published in 1980 as the BIH-MIT Arrhythmia Database. Twenty-three patients' datasets were randomly chosen from a set of 4000 24-hour ambulatory ECG recordings. Of the 4000 recordings, 60% were measured from inpatients at the Beth Israel Hospital and 40% were measured from outpatients. Twenty-five patients' datasets were chosen at random from smaller samples characterized by specific arrhythmias. These patients were separately sampled because their arrhythmias would not have been appropriately represented within a random sample of the 4000 24-hour ambulatory ECG recordings.

The ECGs were measured in 48 half hour excerpts. The measurements were digitized at 360 samples per second over a range of 10 millivolts with 11-bit resolution. The ECG data was published on Physionet.org in two comma-separated value files. One file contains healthy ECG data and the other file contains abnormal ECG data [13]. Figure 3.1 depicts a sample segment of the normal ECG and Figure 3.2 depicts a sample segment of the abnormal ECG.

The abnormal ECG dataset was shrunk to be the same size as the normal ECG dataset because the algorithms in MATLAB required consistent dataset structure and size.

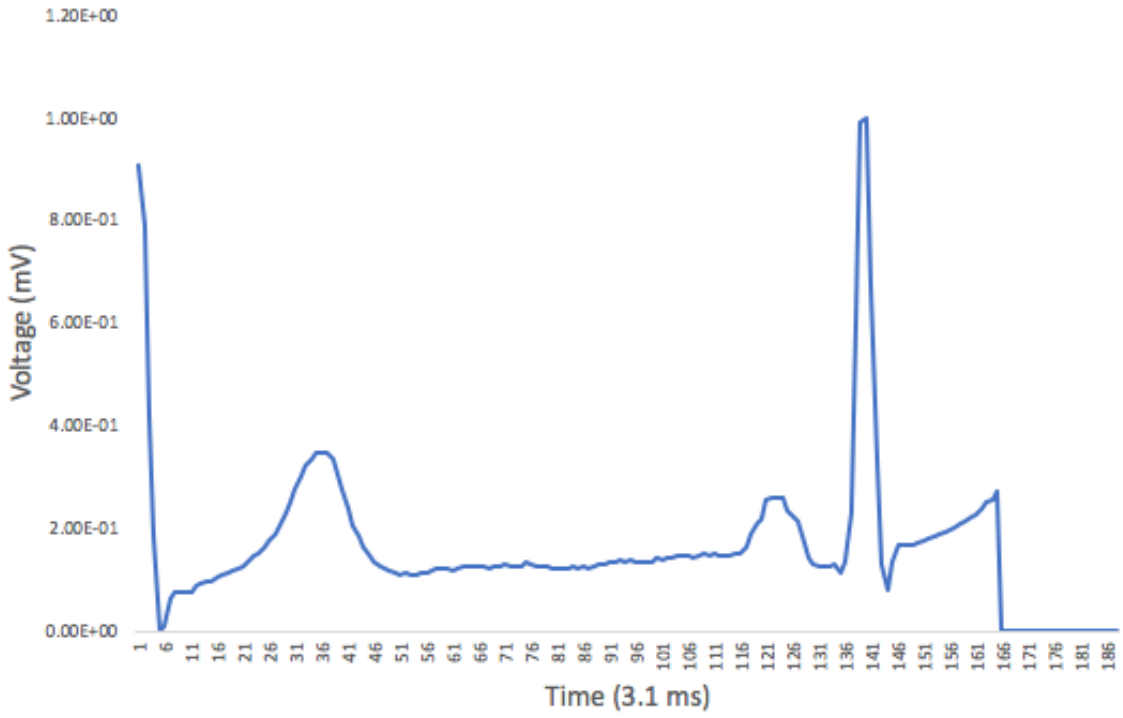

Figure 3.1 Sample Segment of the Normal ECG: The normal ECG displays a clear PQRST wave.

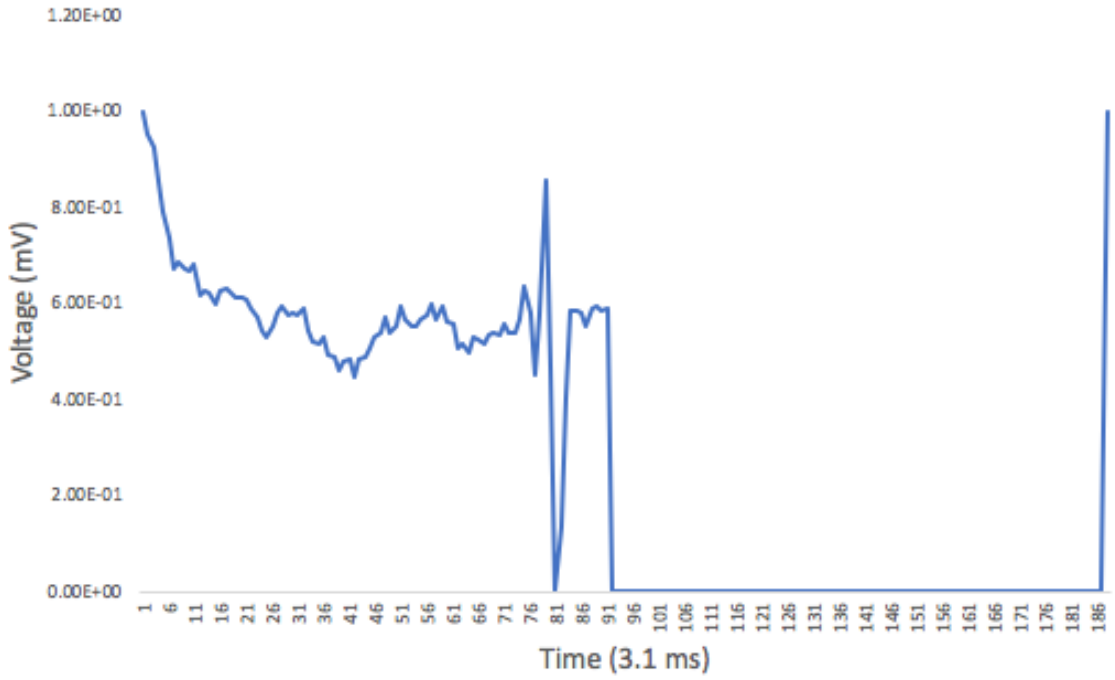

Figure 3.2 Sample Segment of the Abnormal ECG: The abnormal ECG consists of waves that do not display a clear PQRST wave.

3.2 Fourier Transform

The Fourier transform is among the most common tools in engineering. The Fourier transform is a method to break down and analyze any waveform. Almost any measurement in electrophysiology, mechanics, and nature may be represented as a waveform. The Fourier transform describes any of these measurements as a sum of sinusoidal functions. Figure 3.3 depicts a Fourier transform of a signal [14]. The time required for a waveform to repeat itself is the period (P) and the inverse of the period is the frequency (f). The definition of the continuous Fourier transform is given in equation (3.1).

$$F(t) = \int_{-\infty}^{\infty} e^{-2\pi ikt} x(t) dt \quad (3.1)$$

Where F(t) is the Fourier transform function and x(t) is the waveform function in the time domain. Applying the Euler formula to the above equation yields equation (3.2).

$$F(t) = \int_{-\infty}^{\infty} x(t)(\cos(2kt) - isin(2kt))dt \quad (3.2)$$

With these equations, it is possible to decompose a raw signal into its component signals, each with a unique frequency. The resulting Fourier transform function is in the frequency domain, rather than the original time domain [15].

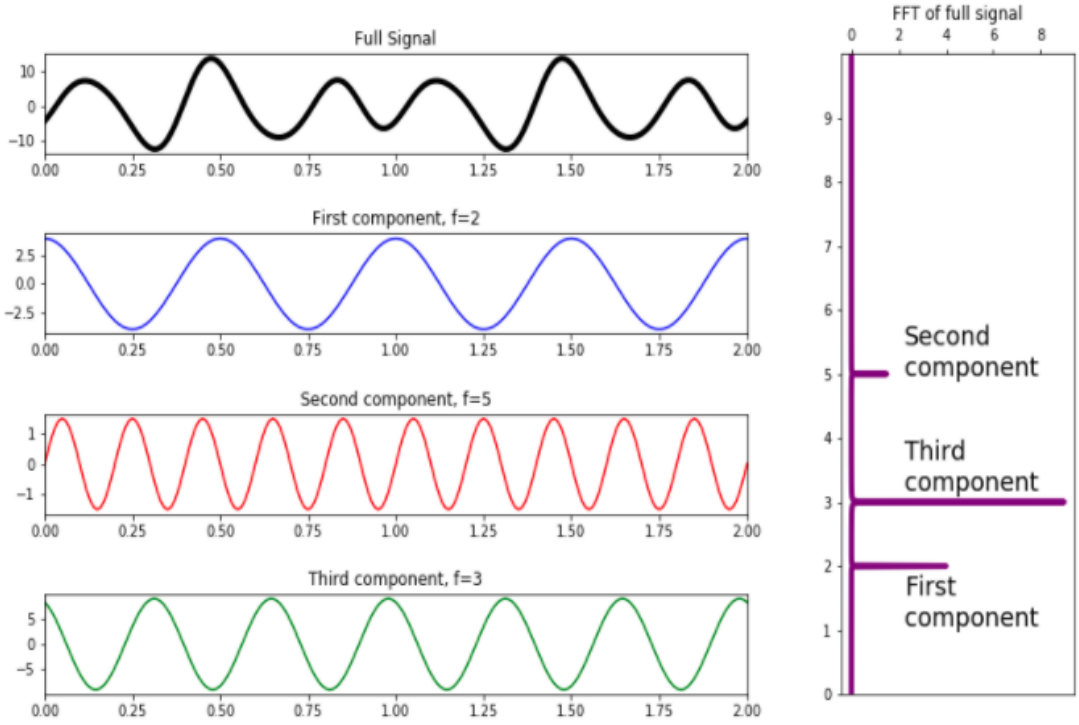

Figure 3.3 Fourier Transform of a Signal: The Fourier transform describes signals as a sum of sinusoidal functions [15].

After measuring the frequencies of the signal components, the signal can then be classified. Classification begins with measuring two characteristics of each of the signal components. The two characteristics are the signal component frequency and the signal component amplitude. With the frequency and amplitude of each signal component, we have the necessary data to begin classification of the raw signal. Figure 3.4 depicts frequency and amplitude of signal components [15]. In the case of this thesis, the phase of the component signals is not necessary for feature extraction of the raw signal.

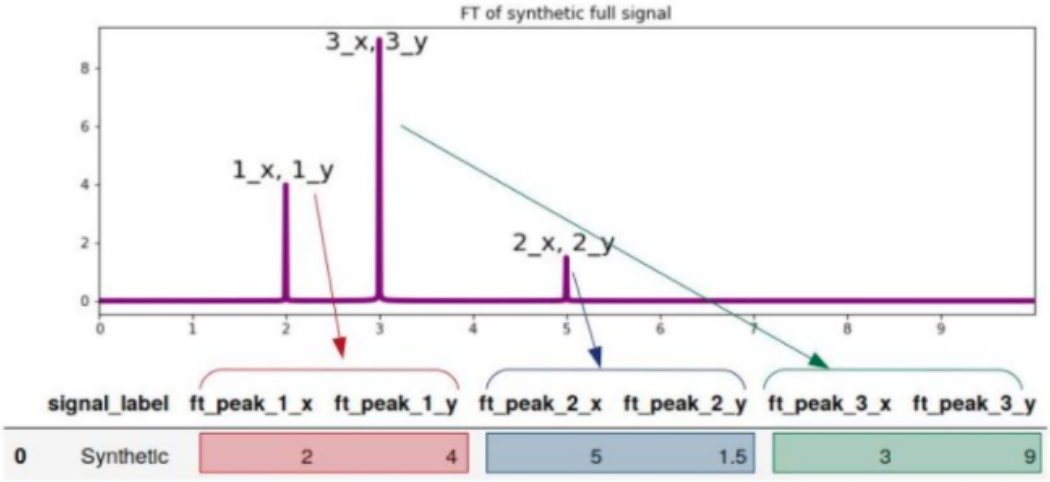

Figure 3.4 Frequency and Amplitude of Signal Components: Frequency and amplitude are the necessary features for classifying the raw signal [15].

3.3 Hilbert Transform

The Hilbert transform is another popular transform. The Hilbert transform is unique from the other feature extraction methods discussed in this document. It does not convert the domain of the raw signal [16]. When the Hilbert transform is applied to a

signal in the time domain, the output is also a waveform in the time domain. The Hilbert transform of a signal, *y(t),* is defined by equation (3.3).

$$u(t) = -\frac{1}{\pi}\int_{-\infty}^{\infty}\frac{y(\eta)}{\eta - t}d\eta = \frac{1}{\pi}\int_{-\infty}^{\infty}\frac{y(\eta)}{t-\eta}d\eta \tag{3.3}$$

Where η is the domain of the transformed function. In machine learning and electronic signals, it is often useful to convert from the Hilbert transform function to an original function, as described by equation (3.4).

$$y(t) = -\frac{1}{\pi}\int_{-\infty}^{\infty}\frac{u(\eta)}{\eta - t}d\eta = \frac{1}{\pi}\int_{-\infty}^{\infty}\frac{u(\eta)}{t-\eta}d\eta \tag{3.4}$$

Figure 3.5 illustrates the Hilbert transform of a signal. The raw signal in the time domain is depicted by the black line. The Hilbert transform deconstructs the raw signal into real and complex (labeled as imaginary in Figure 3.5) components. Both components of the Hilbert transform are in the time domain.

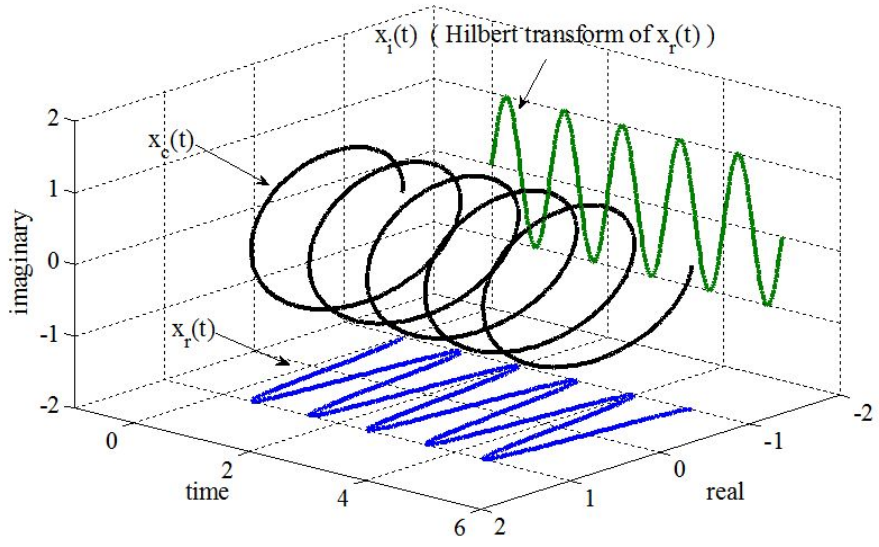

Figure 3.5 Hilbert Transform of a Signal: The Hilbert transform contains both real and complex components [17].

3.4 Wavelet Packet Transform

The Wavelet Packet transform is another popular feature extraction method. This transform is capable of detecting very high frequencies in the time domains. This high resolution makes it particularly useful in decomposing raw signals whose component signals vary in frequency over time [15].

While the Fourier transform deconstructs a raw signal into component frequencies with constant amplitudes, the Wavelet Packet transform deconstructs a raw signal into component frequencies that vary in amplitude. This variation in amplitude is representative of the change in frequency of the component signals over time. The Wavelet Packet transform also differs from the Fourier transform in the domain of its output. While the output of the Fourier transform ranges from negative infinity to positive infinity, the output of the Wavelet Packet transform is finite. Figure 3.6 depicts the output signal of a Wavelet Packet transform. Its domain only ranges in time values at which the raw signal's component frequencies vary. The Wavelet Packet transform outputs signals in both the frequency and time domains, making it a powerful tool for decomposing signals whose component frequencies vary over time [15].

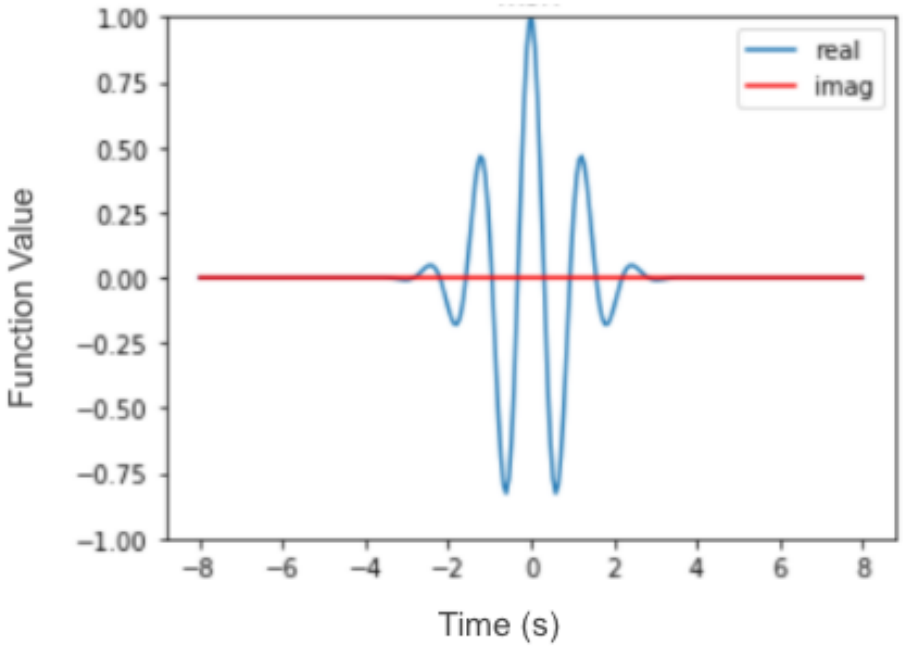

Figure 3.6 Output Signal of a Wavelet Packet Transform: The output of the Wavelet Packet transform is finite in the time domain [15].

The Wavelet Packet transform converts 1-dimensional signals in the time domain into a two-dimensional signal in the frequency domain. *a* is representative of the output's bandwidth parameter and *b* is representative of the central frequency parameter. The central frequency parameter is an array of frequencies that are centered around a base frequency of the Wavelet Packet transform output. The bandwidth parameter determines how much the output signal varies in response to the frequencies around the central frequency parameter. The Wavelet Packet signal is defined by equation (3.5).

$$S(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x(t)\phi\left(\frac{t-b}{a}\right)dt \tag{3.5}$$

The Wavelet Packet transform differs from the Fourier transform in the variety of functions that can appear in its output. While the Fourier transform outputs signals that

only consist of sine and cosine functions, the Wavelet Packet transform can output many: Haar, Daubechies, Symlets, Coiflets, Biorthogonal, Reverse biorthogonal, Discrete Meyer (FIR Approximation), Gaussian, Mexican hat wavelet, Morlet wavelet, Complex Gaussian wavelets, Shannon wavelets, Frequency B-Spline wavelets, and Complex Morlet wavelets [15].

Figure 3.7 illustrates the high resolution of the Wavelet Packet transform of a signal in the levels of decomposition of the raw signal. The raw signal is broken down into component frequencies. Each component frequency is defined by scalar and wavelet components. The wavelet component consists of eigenvalues. In MATLAB, the scalar and wavelet components are referred to as the approximation and detail coefficients, respectively. The Wavelet Packet transform then detects even higher frequencies in the component frequencies and breaks the component frequencies down. The Wavelet Packet transform is capable of breaking down component frequencies several times.

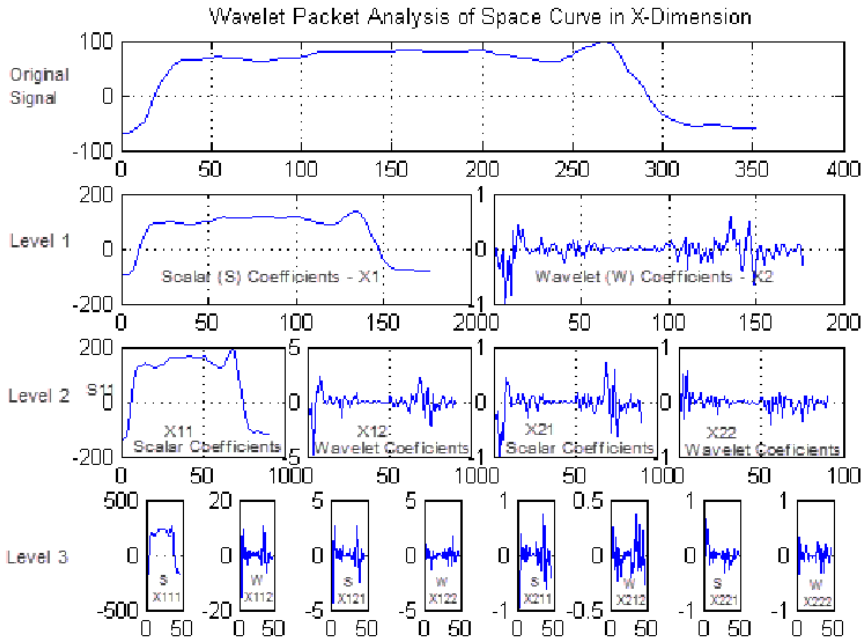

Figure 3.7 Wavelet Packet Transform of a Signal: The component frequencies are in the time domain and the scalar and wavelet coefficients refer to the approximation and detail coefficients, respectively [15].

3.5 Data Preparation

The raw ECG data was prepared prior to being analyzed by each feature extraction function. The Excel files of the normal and abnormal ECGs consisted of 1048 columns. Each column represented a complete heart contraction cycle. The Excel files of the normal and abnormal ECGs consisted of 186 rows. The voltage of the myocardium was measured at intervals of 3.1 milliseconds. Each value in the columns represents the voltage of the myocardium in millivolts at a specific point in time. Each row represented a point in time at which the voltage of the patients' myocardium was measured with electrodes. The rows had units of milliseconds. The title of the Excel files for the normal and abnormal ECGs were "ptbdb_normal_switched_1.xlsx" and

"ptbdb_abnormal_switched_1.xlsx," respectively. These file names were used when importing the datasets into MATLAB.

The Excel files were imported into MATLAB using MATLAB's "readtable" function. The Excel files were then converted into matrices using MATLAB's "table2array" function. The resulting matrices for the normal and abnormal ECGs in MATLAB had dimensions of 186 rows by 1048 columns.

To create placeholders for the extracted features, five empty datasets were defined in MATLAB by creating five empty numeric arrays. Four of the empty numeric arrays will consist of the four extracted features and one empty numeric array will consist of the faults identified from Naïve-Bayesian classification. This will segment the transformed output into short time intervals consisting of four features: the amplitudes of two consecutive peaks and the locations of the two peaks. Peak amplitudes and peak locations are important because they are the most easily identifiable features of a transformed signal. The five empty numeric arrays were then combined into one matrix using MATLAB's "repmat" function with a defined matrix structure of one column by one row. It was necessary to add a sixth empty numeric array when performing the Wavelet-Packet transform to account for the approximation coefficient.

Next, the transformation functions were applied to the matrices of the normal and abnormal ECG datasets. For the Fourier transform, transformation was applied using MATLAB's built-in "fft" function. The Hilbert transform was conducted in three steps: First, MATLAB's built-in "hilbert" function was applied to the matrices of the ECGs. Then, the Fourier transform of the absolute value of the transformed matrices was taken.

Finally, the absolute value of the matrices was taken again. The process for Hilbert transformation is depicted in equation 3.6.

$$H(t) = |fft(|H(t)|)| \quad 3.6$$

The matrices created to contain the four extracted features from the normal and abnormal ECG datasets were filled with the appropriate data using MATLAB's "findpeaks" function, with a minimum peak prominence of zero. The first two features were defined as the peak amplitudes. The last two features were defined as the locations of the peaks. The column for potential faults was assigned values of zero for the entirety of the column. This column will store any faults detected during classification. The extracted features were organized into matrices with four columns for the two peaks and two locations and the fifth column for the faults. The columns of the matrices had the following organization:

```
'Peak Amplitudes1  Peak Amplitudes2  Peak Locations1  Peak Locations2  Faults'
```

The resulting matrices had 37 rows, because 37 features were extracted from each column of the matrices of the normal and abnormal ECGs. This process was embedded within a for loop that extracted the features for every column in the normal and abnormal ECG matrices.

The Wavelet-Packet transform also required a "for" loop statement to carry out the transformation for every column in the normal and abnormal ECG matrices. MATLAB's "wavedec" function was embedded within the "for" loop and was applied to the normal and abnormal ECG matrices with fourth level decomposition. MATLAB's

"wavedec" function carries out 1-dimensional Wavelet Packet transformation on the ECG signal. The detail and approximation coefficients were extracted from the resulting decomposed matrix using MATLAB's "detcoef" function. The coefficients consisted of four detail coefficients and one approximation coefficient. Next, the percentage energy corresponding to the approximation coefficient and the percentage energy corresponding to the detail coefficients was measured using MATLAB's "wenergy" function. Calculating the percentage energy of the approximation and detail coefficients is necessary for determining the size approximation and detail portions of each level of decomposition. The five empty numeric arrays were set equal to the percentage energy of the approximation coefficient and the detail coefficients. The first four features were defined as the four values of the percentage energy corresponding to the detail coefficients and the fifth feature was defined as the percentage energy corresponding to the approximation coefficient. A sixth column was created and assigned values of zero for the entirety of the column. The sixth column was created to store any detected faults during classification. The resulting matrices had 37 rows by six columns.

Prior to classification, the transformed matrices of the normal and abnormal ECG datasets were converted into tables using MATLAB's "struct2table" function. The table of the extracted features of the abnormal ECG was vertically concatenated to the end of the extracted features of the normal ECG using MATLAB's "vertcat" function. Naive-Bayesian classification was conducted three times: once for each of the feature extraction methods.

To use the Naive-Bayesian classifier, an empty data table was created with four columns: iteration number, sensitivity, specificity, and time. This table will contain the

results after running the classifier. The order of the rows of the transformed data tables from the Fourier, Hilbert, and Wavelet-Packet transforms were randomized using MATLAB's "randperm" function. Randomizing the rows of the data tables provides a different training set for every time that the Naive-Bayesian classifier is run and expands the opportunity to assess the effectiveness of the extraction methods. The four extracted features were defined as independent variables. The potential faults were defined as dependent variables. In the case of the Wavelet-Packet transform, the five extracted features were defined as independent variables and the potential faults were defined as dependent variables.

80% of the independent and dependent variables were defined as the training portion of the feature extraction data and 20% of the independent and dependent variables were defined as the testing portion of the feature extraction data. Both the training and testing portions of the feature extraction data consist of normal and abnormal ECG features. The repetitiveness of the normal features provides the Naïve-Bayesian classifier with a regular pattern that it can learn. The irregularity of the abnormal features has no regular pattern that the Naïve-Bayesian classifier can learn and are therefore labeled as faults.

Then the Naive-Bayesian model was applied using MATLAB's "fitcnb" training function. 80% of the variables were used to train the model and 20% of the variables were tested. To predict the number of faults in the data, MATLAB's "predict" function was applied using the results from the "fitcnb" function as the regression model object and the testing portion of the independent variables as the predictor input values. Finally, the accuracy of the fault detection was assessed using MATLAB's "confusionmat"

function. The predicted number of faults was compared to the number of faults in the testing portion of the dependent variables.

The randomization of the extracted feature datasets and the Naive-Bayesian classifier was embedded in a for loop and run for ten iterations for each feature extraction method. Running the Naive-Bayesian classifier for ten iterations with a new dataset in every iteration increased the opportunity to assess the effectiveness of the classifier model. Sensitivity and specificity were calculated with each iteration. The data preparation process is illustrated in Figure 3.8.

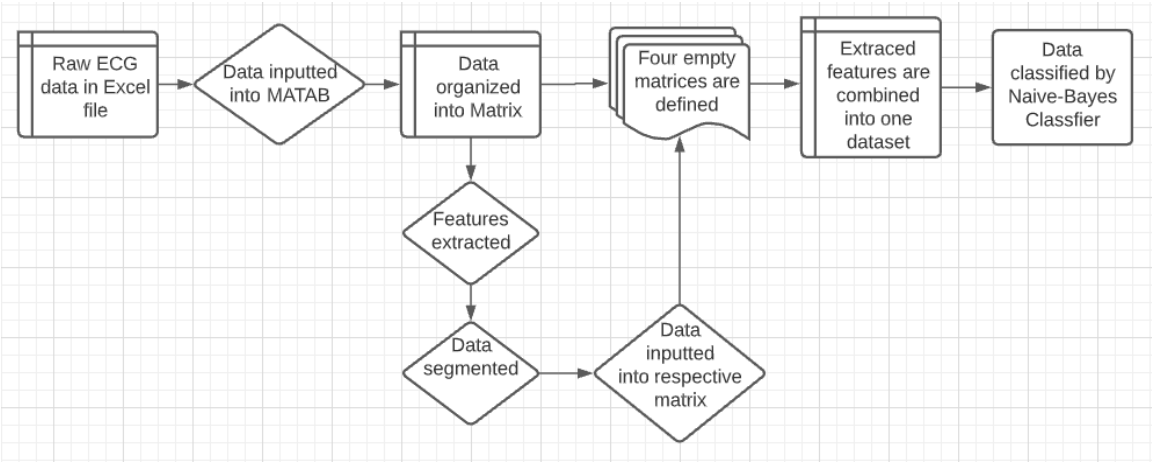

Figure 3.8 Data Preparation Process: Classifying the ECG data consisted of importing the Excel files of the ECG data into MATLAB, transforming the ECG data sets, and then classifying the data sets.

3.6 Naive-Bayesian Classification

The Naive-Bayesian classifier is among the most common classification methods in machine learning. It is a probabilistic classifier that is based on Bayes Theorem. Bayes Theorem describes the probability of an outcome contingent on a previous outcome. The

contingency of an outcome based on another is what distinguishes Bayes Theorem from other probabilistic classifiers. Other probabilistic classifiers describe marginal probability or joint probability. Marginal probability is the probability that an event will occur, regardless of previous events. The marginal probability that event A will occur is defined as P(A).

Joint probability is the probability that two events will occur simultaneously. The joint probability that both event A and event B will occur together is defined as P(A,B).

The conditional probability of Bayes Theorem is described by equation (3.7).

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3.7}$$

This equation describes the probability that event A will occur given that event B has already occurred. This probability is calculated by using the reversed conditional probability. The probability that event B will occur given that event A has already occurred. The reverse conditional probability is multiplied by the marginal probability that event A will occur and divided by the marginal probability that event B will occur. The probability of event A contingent on event B may also be calculated by dividing the joint probability of event A and event B by the marginal probability of event B. The Bayes Theorem is a method to calculate conditional probability without the joint probability.

The reverse probability can be calculated in the same way. The probability that event B will occur given that event A has already occurred is given by rearranging equation (3.7):

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

In machine learning, P(B|A) is referred to as the posterior probability and P(B) is referred to as the prior probability. When calculating the probability of event B, contingent on event A, P(A|B) is referred to as the likelihood and P(A) is the evidence. Therefore, we can describe the probability with the following equation:

$$Posterior = \frac{Likelihood * Prior}{Evidence}$$

For example, Bayes theorem can be used to determine the probability that children will play outdoors if it is a weekend. The probability that children will play outdoors if it is a weekend is the posterior probability. The likelihood probability is the probability that it is a weekend day if children are playing outdoors. The probability that it is a weekend is the evidence. The probability that children will play outdoors is the prior probability. This results in the following equation:

$P(outdoors|weekend) = \frac{P(weekend|outdoor)P(outdoors)}{P(weekend)}$. Each probability has a unique value.

P(weekend) = 2/7 = 0.286

P(outdoors) = 1/2 = 0.5

P(weekend|outdoors) = 3/9 = 0.333

Calculating posterior probability,

$P(outdoors|weekend) = \frac{(0.333)*(0.5)}{(0.286)} = 0.582.$

Bayes Theorem classifies data based on conditional probability and is a useful tool in diagnosing faults in a system. If a feature in a data set is determined to have low probability of occurring based on previous patterns, it will be classified as a fault. Sensitivity is the rate at which the Naive-Bayesian classifier accurately detects faults and specificity is the rate at which the classifier accurately detects the absence of faults [18].

The feature extraction methods prepare the ECG signal for Naive-Bayesian classification. In MATLAB, the feature extraction methods were applied to the normal and abnormal datasets individually. The resulting normal and abnormal datasets were combined into a single dataset before being analyzed by the Naive-Bayesian classifier. The Naive-Bayesian classifier operates in the same manner despite the differences in output between the feature extraction methods. The Naive-Bayesian classifier algorithm consists of two phases: the training phase and the testing phase. The first 80% of the output of a feature extraction method is used to train the Naive-Bayesian classifier algorithm. The algorithm measures two characteristics of the feature extraction output: peaks and locations. The regular pattern of peak amplitudes and the frequency at which peaks occur is measured and "learned" by the algorithm. This system of classification is the same for all feature extraction methods.

The final segment of the feature extraction output is tested by the algorithm. Based on the pattern of peak amplitudes and peak locations that the algorithm learned during training, the final 20% of the output is tested for irregularities. Each segment of the feature extraction output is classified individually. During testing, the classifier may detect a peak amplitude significantly higher or lower than the average amplitude detected during the training phase. When this occurs, it will classify the segment as a fault.

During testing, the Naive-Bayesian classifier will also measure the distances between peaks in the feature extraction output. The distance between peaks is compared to the average distances detected during the training phase. When the classifier detects the distance between two peaks to be significantly larger or smaller in comparison to the average distance calculated during the training phase, it will classify the segment as a fault.

Chapter 4
Results

4.1 Fourier Transform

When the Fourier transform was applied to the normal and abnormal ECG datasets, the ECG signals were converted into the frequency domain. Figure 4.1 depicts a sample Fourier transform of the ECG signals.

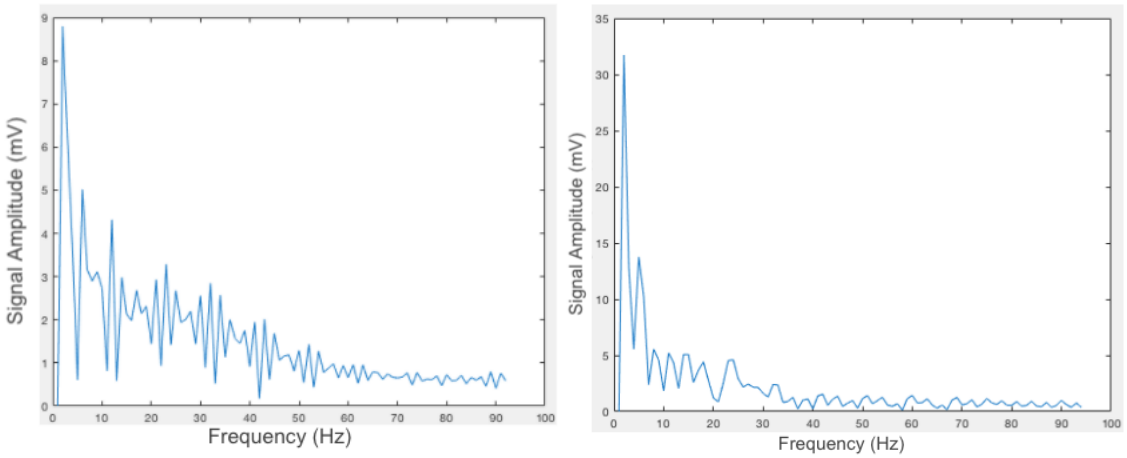

Figure 4.1: Sample Fourier Transform of the ECG Signals: The graph on the left depicts a transformed heart contraction cycle from the normal ECG and the graph on the right depicts a transformed heart contraction cycle from the abnormal ECG.

When the Naïve-Bayesian classifier was applied to the transformed ECG data, MATLAB outputted four values for each iteration: true positives, true negatives, false positives, and false negatives. The MATLAB output differs between each iteration of the Naive-Bayesian classifier. The MATLAB output is used to determine the most effective iteration for each feature extraction method. Table 4.1 depicts the classification results with Fourier transform.

Table 4.1: Classification Results with Fourier Transform

Iteration	True Positives	True Negatives	False Positives	False Negatives
1	674	460	448	129
2	674	430	502	145
3	679	432	506	134
4	696	446	462	147
5	659	460	499	133
6	648	462	496	145
7	663	464	493	131
8	671	410	530	140
9	650	448	498	155
10	658	458	485	150

Each value was compared against other values in the same column. In the True Positives and the True Negatives columns, the red color indicates that the value is below the average value of the column. The green color indicates that the value is above the average value of the column. In the False Positives and False Negatives columns, the red color indicates that the value is above the average value of the column. The green color

indicates that the value is below the average value of the column. Table 4.2 depicts the sensitivity and specificity of classification with Fourier transform.

Table 4.2: Sensitivity and Specificity of Classification with Fourier Transform

Iteration	Sensitivity	Specificity
1	0.8327	0.4565
2	0.8182	0.4789
3	0.8202	0.4866
4	0.8216	0.452
5	0.8424	0.4782
6	0.8177	0.4838
7	0.8039	0.4581
8	0.8427	0.4738
9	0.8223	0.4833
10	0.8102	0.4732

Each value was compared against other values in the same column. The red color indicates that the value is below the average value of the column. The green color indicates that the value is above the average value of the column. Higher values are preferable in both the sensitivity and specificity columns.

4.2 Hilbert Transform

When the Hilbert transform was applied to the normal and abnormal ECG datasets, the ECG signals remained in the time domain. Figure 4.2 depicts a sample Hilbert transform of the ECG signals.

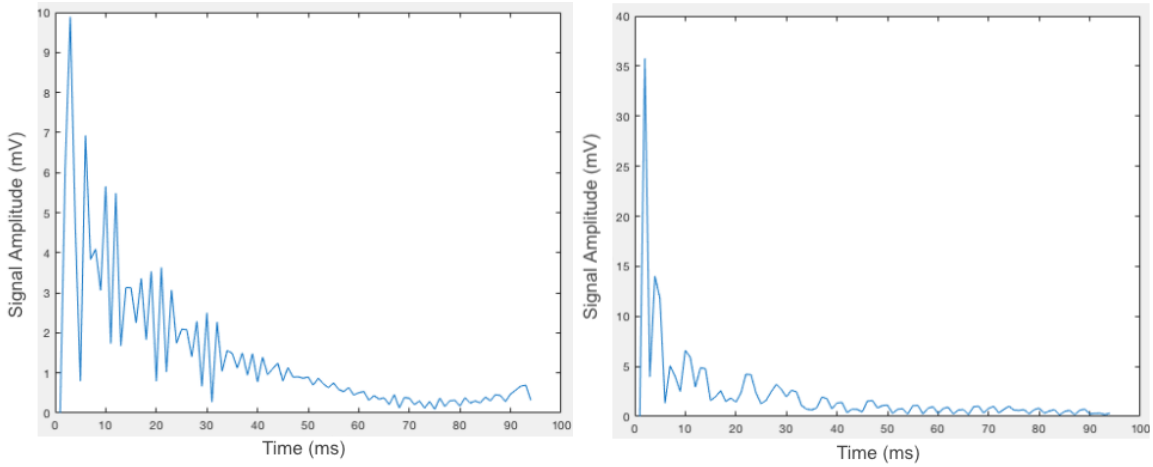

Figure 4.2: Sample Hilbert Transform of the ECG Signals: The graph on the left depicts a transformed heart contraction cycle from the normal ECG and the graph on the right depicts a transformed heart contraction cycle from the abnormal ECG.

When the Naïve-Bayesian classifier was applied to the transformed ECG data, MATLAB outputted four values for each iteration: true positives, true negatives, false positives, and false negatives. The MATLAB output differs between each iteration of the Naive-Bayesian classifier. Table 4.3 depicts the classification results with Hilbert transform.

Table 4.3: Classification Results with Hilbert Transform

Iteration	True Positives	True Negatives	False Positives	False Negatives
1	671	537	401	142
2	654	559	397	141
3	674	546	410	121
4	681	547	379	144
5	665	558	383	145
6	667	550	396	138
7	683	527	393	148
8	680	543	375	153
9	674	537	389	151
10	668	549	389	145

After Naïve-Bayesian Classification, each value of the MATLAB output was compared against other values in the same column. Table 4.4 depicts the sensitivity and specificity of classification with Hilbert transform.

Table 4.4: Sensitivity and Specificity of Classification with Hilbert Transform

Iteration	Sensitivity	Specificity
1	0.8409	0.595
2	0.8242	0.5853
3	0.8176	0.5711
4	0.8258	0.5919
5	0.8302	0.5894
6	0.8294	0.5731
7	0.8142	0.5689
8	0.8133	0.5764
9	0.8252	0.6031
10	0.8439	0.5751

Sensitivity and specificity were also compared to the values of their respective columns.

4.3 Wavelet Packet Transform

When the Wavelet Packet transform was applied to the normal and abnormal ECG datasets, the ECG signals were decomposed into scalar and wavelet portions.

MATLAB did not produce graphs of this transformation. Therefore, sample graphs of the Wavelet Packet transforms are not provided.

When the Naïve-Bayesian classifier was applied to the transformed ECG data, MATLAB outputted four values for each iteration: true positives, true negatives, false positives, and false negatives. The MATLAB output differs between each iteration of the Naive-Bayesian classifier. Table 4.5 depicts the classification results with Wavelet Packet transform.

Table 4.5: Classification Results with Wavelet Packet Transform

Iteration	True Positives	True Negatives	False Positives	False Negatives
1	704	791	168	88
2	732	773	158	88
3	708	804	147	92
4	699	791	165	96
5	708	772	175	96
6	728	772	171	80
7	663	843	162	83
8	707	795	170	79
9	759	741	158	93
10	738	751	179	83

After Naïve-Bayesian classification, each value of the MATLAB output was compared against other values in the same column. Table 4.6 depicts the sensitivity and specificity of classification with Wavelet Packet transform.

Table 4.6: Sensitivity and Specificity of Classification with Wavelet Packet Transform

Iteration	Sensitivity	Specificity
1	0.8868	0.8245
2	0.8793	0.8363
3	0.9166	0.832
4	0.8993	0.8247
5	0.878	0.8434
6	0.8845	0.8409
7	0.8901	0.8536
8	0.9012	0.8182
9	0.8862	0.8343
10	0.8985	0.8144

Sensitivity and specificity were also compared to the values of their respective columns.

4.4 Comparison of the Three Feature Extraction Methods

For the final comparison of the effectiveness of each machine learning technique, the most successful iterations of each feature extraction method were compared against each other. Higher values are preferable in both columns. The red color indicates that the value is low compared to other values in the same column. The green color indicates that the value is high compared to other values in the same column. Table 4.7 depicts the comparison of the three machine learning techniques.

Table 4.7: Comparison of the Three Machine Learning Techniques

Feature Extraction Method	Most Successful Iteration	Sensitivity	Specificity
Fourier Transform	5	0.8424	0.4782
Hilbert Transform	1	0.8409	0.595
Wavelet Packet Transform	7	0.8901	0.8536

Chapter 5
Discussion

The results reveal that the Wavelet Packet transform is the most effective feature extraction method to prepare data for Naive-Bayesian classification. When the Fourier, Hilbert, and Wavelet Packet transforms, followed by Naïve-Bayesian classification, were applied to engine data, all three feature extraction methods yielded results with relatively equal accuracy [19].

5.1 Implications

The three feature extraction methods, followed by Naïve-Bayesian classification, display relatively equal accuracy in classifying engine vibration data. These same machine learning techniques display very unequal accuracy when classifying ECG data. These results indicate a fundamental difference between cardiac physiology and mechanical engines. The fundamental assumption when testing machine learning techniques on ECG data is that the human cardiac system is similar to a mechanical system.

The results of classifying ECG data with machine learning techniques indicate that this fundamental assumption is invalid. ECG data and data from mechanical systems display different results when applied to machine learning. This difference may be a result of the higher resolution of ECG. The Wavelet Packet transform may be more successful in detecting the regular pattern of the PQRST wave than the more sporadic vibrational engine data.

5.2 Limitations and Future Work

This study compares the effectiveness of the three prominent machine learning techniques in classifying ECG data. The human ECG differs significantly in how effectively it can be classified by different machine learning techniques. Further analysis is required to understand why some machine learning techniques are more effective in classifying ECG. This understanding is vital in optimizing ECG classification and predicting which other feature extraction methods may be effective. Further analysis may also uncover the variables that make physiologic systems unique from man-made systems.

5.3 Conclusion

Heart disease is the leading cause of death in the United States. Early detection of heart disease is vital to providing patients with proper care and increasing patients' chance of survival. After obtaining a patient's ECG, feature extraction with Naïve-Bayesian classification will classify the ECG after about 60 seconds. Quickly and effectively diagnosing patients who display symptoms of cardiac arrest may alleviate the catastrophic effect of heart disease on our society. Early detection may be achievable with a design of a device that utilizes effective machine learning techniques. This device may measure the ECG of a patient experiencing chest discomfort and then quickly prepare the ECG data for Naive-Bayesian classification with the effective feature

extraction method. The machine will then determine if the patient requires urgent care. This new technique may save innumerable lives.

BIBLIOGRAPHY

[1] D. Tirziu, F. J. Giordano, and M. Simons, "Cell Communications in the Heart," *Circulation*. 2010 Aug 31; 122(9): 928–937.

[2] "Cardiac Muscle and Electrophysiology," *Lumen,* https://courses.lumenlearning.com/.

[3] R. K. Pai, M. J. Gabica, A. Husney, and G. Philippides, "Electrical System of the Heart," *Michigan Medicine,* August 31, 2021.

[4] "CPR Facts and Stats," *American Heart Association,* https://cpr.heart.org/.

[5] L. B. Marinho, N. D. M. M. Nascimento, J. W. M. Souza, M. V. Gurgel, P. P. R. Filho, and V. H. C. D. Albuquerque. "A novel electrocardiogram feature extraction approach for cardiac arrhythmia classification," *Future Generation Computer Systems,* August 2019, Volume 97 Pages 564 - 577.

[6] R. E. Klabunde, "Cardiovascular Physiology Concepts," https://www.cvphysiology.com/

[7] T. S. Metkus, "Cardiac Conduction System," *Medline Plus,* July, 7, 2020.

[8] "Cardiac Muscle Tissue," *Lumen,* https://courses.lumenlearning.com/.

[9] S. Knapp, "Action Potential," *Biology Dictionary,* June 29, 2020.

[10] G. Ikonnikov and D. Yelle, "PHYSIOLOGY OF CARDIAC CONDUCTION AND CONTRACTILITY," *McMaster Pathophysiology Review,* 2018.

[11] "Machine Learning: What it is and Why it Matters," https://www.sas.com/.

[12] "Machine Learning and AI in Manufacturing," https://www.seebo.com/.

[13] G. Moody and R. Mark, “MIT - BIH Arrhythmia Database,” *Physionet, F*ebruary 24, 2005.

[14] P. Bevel, “Fourier Transforms,” https://www.thefouriertransform.com/.

[15] M. Faraggi and K. Sayadi, “Time series features extraction using Fourier and Wavelet transforms on ECG data,” *Les Blog de Octos,* November 23, 2019.

[16] A. Medoued, A. Lebaroud, and D. Sayad, “Application of Hilbert transform to fault detection in electric machines,” *Advances in Difference Equations,* January 2013.

[17] D.E.T. Romero and G. J. Dolecek, “Digital FIR Hilbert Transformers: Fundamentals and Efficient Design Methods,” *Intechopen.com,* September 26, 2012.

[18] J. Brownlee, “A Gentle Introduction into the Bayes Theorem for Machine Learning,” *Machine Learning Mastery,* October 4, 2019.

[19] M. Jarrar, “COMPARING FEATURE EXTRACTION METHODS TO IMPROVE MACHINERY FAULT DETECTION USING VIBRATIONAL SIGNALS,” *California Polytechnic State University,* August 2019.

APPENDIX A

A.1 FourierTransform_NaiveBayesianClassification.m

```
clc;
clear;
close all;

data = readtable(['ptbdb_normal_switched_1.xlsx']);
% making table into a matrix
data = table2array(data);


%% feature extract w/ fft for normal ECG

%empty_Dataset1.Index = [];
empty_Dataset1.feature1 = [];
empty_Dataset1.feature2 = [];
empty_Dataset1.feature3 = [];
empty_Dataset1.feature4 = [];
empty_Dataset1.fault = [];


Dataset1 = repmat(empty_Dataset1,[1,1]);

[m,n] = size(data);

% making the FFT of each heartbeat
fft1 = abs(fft(data,[],1));
% removing the first row because it seems to be an outlier
fft1(1, :) = zeros(1, n);

%creating the final matrix of the 4 features w/ a column fault = 0
for i=1:n
  %Dataset1(i).Index = i;

  [peaks1,locs1] = findpeaks(fft1(:,i),'MinPeakProminence',0);
  Dataset1(i).feature1 = peaks1(1,1);
  Dataset1(i).feature2 = peaks1(2,1);
  Dataset1(i).feature3 = locs1(1,1);
  Dataset1(i).feature4 = locs1(2,1);

  Dataset1(i).fault = 0;
end
```

```
%% Data with faults

data = readtable('ptbdb_abnormal_switched_1.xlsx');
data = table2array(data);


%% feature extract w/ fft for abnormal heartbeats


% empty_Dataset3.Index = [];
empty_Dataset2.feature1 = [];
empty_Dataset2.feature2 = [];
empty_Dataset2.feature3 = [];
empty_Dataset2.feature4 = [];
empty_Dataset2.fault = [];

Dataset2 = repmat(empty_Dataset2,[1,1]);

[m,n] = size(data);

fft2 = abs(fft(data,[],1));
fft2(1, :) = zeros(1, n);

for i=1:n
  %Dataset3(i).Index = i;

  [peaks2,locs2] = findpeaks(fft2(:,i),'MinPeakProminence',0);
  Dataset2(i).feature1 = peaks2(1,1);
  Dataset2(i).feature2 = peaks2(2,1);
  Dataset2(i).feature3 = locs2(1,1);
  Dataset2(i).feature4 = locs2(2,1);

  Dataset2(i).fault = 1;
end


%% combine datasets

% converting matrix into tables
Dataset1 = struct2table(Dataset1);
Dataset2 = struct2table(Dataset2);

%combining the tables into one
DatasetA = vertcat(Dataset1, Dataset2);
```

```
%removing a row because when we split the data into 80%/20% we have to have an
%odd number of rows
DatasetA([1],:) = [];

%saving the table into a new file
filename = "dataset1TEST.xlsx";
writetable(DatasetA,filename);

%% Naive Bayesian Predict Results
clc;
clear;
close all;

% Read Excel into a table, loading the final dataset
data = readtable('dataset1TEST.xlsx');

%creating empty data table
empty_Result.Index = [];
empty_Result.Sensitivity = [];
empty_Result.Specificity = [];
empty_Result.Time = [];


% this is the NB algorithm code
% I am running it 10 times to find the best model out of the 10
for i = 1:10

%randomize the rows of the dataset
data = data(randperm(size(data, 1)), :);

%Ind_Vs is for the 4 features in my dataset
Ind_Vs = data(:,1:4);
%Dep_V is the Fault or no Fault column in my dataset
Dep_V = data(:,5);

[m,n] = size(data);

%train set is 80% of the data
Train_Ind_Vs = Ind_Vs(1:end-(m*.2),:);
%test set is 20% of the data
Test_Ind_Vs = Ind_Vs(end-(m*.2)+1:end,:);

Train_Dep_V = Dep_V(1:end-(m*.2),:);
Test_Dep_V = Dep_V(end-(m*.2)+1:end,:);

Test_Dep_V = table2array(Test_Dep_V);
```

```
Result(i).Index = i;

% the NB model
NB_Mdl = fitcnb(Ind_Vs,Dep_V);

[Predict_Test,Posterior] = predict(NB_Mdl,Test_Ind_Vs);


C = confusionmat(Test_Dep_V,Predict_Test)

TN = C(2,2);
FN = C(1,2);
FP = C(2,1);
TP = C(1,1);

%Dataset1(i).feature1 = peak_loc1(1,1);
Result(i).Sensitivity = TP / (TP+FN);
Result(i).Specificity = TN / (TN+FP);


end
```

A.2 HilbertTransform_NaiveBayesianClassification.m

```
clc;
clear;
close all;

data = readtable(['ptbdb_normal_switched_1.xlsx']);
% making table into a matrix
data = table2array(data);

%% feature extract w/ hilbert for normal heart beats

%extracting the four features from the healthy ECG data


%empty_Dataset1.Index = [];
empty_Dataset1.feature1 = [];
empty_Dataset1.feature2 = [];
```

```
empty_Dataset1.feature3 = [];
empty_Dataset1.feature4 = [];
empty_Dataset1.fault = [];


Dataset1 = repmat(empty_Dataset1,[1,1]);


[m,n] = size(data);

% Taking the Hilbert transform of each heartbeat
%fft1 = abs(fft(data,[],1));
y= hilbert(data);
env = abs(y);
fft1 = abs(fft(env,[],1));

% removing the first row because it seems to be an outlier
fft1(1, :) = zeros(1, n);


%creating the final matrix of the 4 features w/ a column fault = 0
for i=1:n
  %Dataset1(i).Index = i;

  [peaks1,locs1] = findpeaks(fft1(:,i),'MinPeakProminence',0);
  Dataset1(i).feature1 = peaks1(1,1);
  Dataset1(i).feature2 = peaks1(2,1);
  Dataset1(i).feature3 = locs1(1,1);
  Dataset1(i).feature4 = locs1(2,1);

  Dataset1(i).fault = 0;
end

%% Data with fault

data = readtable('ptbdb_abnormal_switched_1.xlsx');
data = table2array(data);

%% feature extract w/ hilbert for abnormal heartbeats


% empty_Dataset3.Index = [];
empty_Dataset2.feature1 = [];
empty_Dataset2.feature2 = [];
empty_Dataset2.feature3 = [];
empty_Dataset2.feature4 = [];
```

```
empty_Dataset2.fault = [];

Dataset2 = repmat(empty_Dataset2,[1,1]);


[m,n] = size(data);

%fft2 = abs(fft(data,[],1));
y= hilbert(data);
env = abs(y);
fft2 = abs(fft(env,[],1));

fft2(1, :) = zeros(1, n);

for i=1:n

   [peaks2,locs2] = findpeaks(fft2(:,i),'MinPeakProminence',0);
   Dataset2(i).feature1 = peaks2(1,1);
   Dataset2(i).feature2 = peaks2(2,1);
   Dataset2(i).feature3 = locs2(1,1);
   Dataset2(i).feature4 = locs2(2,1);

   Dataset2(i).fault = 1;
end


%% combine datasets

% converting matrix into tables
Dataset1 = struct2table(Dataset1);
Dataset2 = struct2table(Dataset2);

%combining the tables into one
DatasetA = vertcat(Dataset1, Dataset2);

%removing a row because when we split the data into 80%/20% we have to have an
%odd number of rows
DatasetA([1],:) = [];

%saving the table into a new file
filename = "dataset1TEST.xlsx";
writetable(DatasetA,filename);
```

```
%% Naive Bayesian Predict Results
clc;
clear;
close all;

% Read Excel into a table, loading the final dataset
data = readtable('dataset1TEST.xlsx');

%creating empty data table
empty_Result.Index = [];
empty_Result.Sensitivity = [];
empty_Result.Specificity = [];
empty_Result.Time = [];


% this is the NB algorith code
% I am running it 10 times to find the best model out of the 10
for i = 1:10

%randomize the rows of the dataset
data = data(randperm(size(data, 1)), :);

%Ind_Vs is for the 4 features in my dataset
Ind_Vs = data(:,1:4);
%Dep_V is the Fault or no Fault column in my dataset
Dep_V = data(:,5);

[m,n] = size(data);

%train set is 80% of the data
Train_Ind_Vs = Ind_Vs(1:end-(m*.2),:);
%test set is 20% of the data
Test_Ind_Vs = Ind_Vs(end-(m*.2)+1:end,:);

Train_Dep_V = Dep_V(1:end-(m*.2),:);
Test_Dep_V = Dep_V(end-(m*.2)+1:end,:);

Test_Dep_V = table2array(Test_Dep_V);



Result(i).Index = i;
```

```
% the NB model
NB_Mdl = fitcnb(Ind_Vs,Dep_V);

[Predict_Test,Posterior] = predict(NB_Mdl,Test_Ind_Vs);


C = confusionmat(Test_Dep_V,Predict_Test)

TN = C(2,2);
FN = C(1,2);
FP = C(2,1);
TP = C(1,1);

%Dataset1(i).feature1 = peak_loc1(1,1);
Result(i).Sensitivity = TP / (TP+FN);
Result(i).Specificity = TN / (TN+FP);


end
```

A.3 WaveletPacketTransform_NaiveBayesianClassification.m

```
clc;
clear;
close all;

data = readtable(['ptbdb_normal_switched_1.xlsx']);

% making table into a matrix
data = table2array(data);


%% feature extract w/ wavelet-packet for normal heart beats

%extracting the four features from the healthy ECG data


%empty_Dataset1.Index = [];
empty_Dataset1.feature1 = [];
empty_Dataset1.feature2 = [];
empty_Dataset1.feature3 = [];
empty_Dataset1.feature4 = [];
```

```
empty_Dataset1.feature5 = [];
empty_Dataset1.fault = [];


Dataset1 = repmat(empty_Dataset1,[1,1]);


[m,n] = size(data);


for i=1:n
   %Dataset1(i).Index = i;
    [c,l] = wavedec(data(:,i),4,'db2');
    [cd1,cd2,cd3,cd4] = detcoef(c,l,[1 2 3 4]);
    [Ea,Ed] = wenergy(c,l);

   Dataset1(i).feature1 = Ed(1,1);
   Dataset1(i).feature2 = Ed(1,2);
   Dataset1(i).feature3 = Ed(1,3);
   Dataset1(i).feature4 = Ed(1,4);
   Dataset1(i).feature5 = Ea(1,1);

   Dataset1(i).fault = 0;
end
%% Data with fault

data = readtable('ptbdb_abnormal_switched_1.xlsx');
data = table2array(data);

%% feature extract w/ wavelet-packet for abnormal heartbeats


empty_Dataset2.feature1 = [];
empty_Dataset2.feature2 = [];
empty_Dataset2.feature3 = [];
empty_Dataset2.feature4 = [];
empty_Dataset2.feature5 = [];
empty_Dataset2.fault = [];

Dataset2 = repmat(empty_Dataset2,[1,1]);


[m,n] = size(data);

for i=1:n
   %Dataset1(i).Index = i;
```

```
    [c,l] = wavedec(data(:,i),4,'db2');
    [cd1,cd2,cd3,cd4] = detcoef(c,l,[1 2 3 4]);
    [Ea,Ed] = wenergy(c,l);

   Dataset2(i).feature1 = Ed(1,1);
   Dataset2(i).feature2 = Ed(1,2);
   Dataset2(i).feature3 = Ed(1,3);
   Dataset2(i).feature4 = Ed(1,4);
   Dataset2(i).feature5 = Ea(1,1);

   Dataset2(i).fault = 1;
end

%% combine datasets

% converting matrix into tables
Dataset1 = struct2table(Dataset1);
Dataset2 = struct2table(Dataset2);

%combining the tables into one
DatasetA = vertcat(Dataset1, Dataset2);

%removing a row because when we split the data into 80%/20% we have to have an
%odd number of rows
DatasetA([1],:) = [];

%saving the table into a new file
filename = "dataset1TESTB.xlsx";
writetable(DatasetA,filename);




%% Naive Bayesian Predict Results
clc;
clear;
close all;

% Read Excel into a table, loading the final dataset
data = readtable('dataset1TESTB.xlsx');

%creating empty data table
empty_Result.Index = [];
```

```
empty_Result.Sensitivity = [];
empty_Result.Specificity = [];
empty_Result.Time = [];


% this is the NB algorith code
% I am running it 10 times to find the best model out of the 10
for i = 1:10

%randomize the rows of the dataset
data = data(randperm(size(data, 1)), :);

%Ind_Vs is for the 4 features in my dataset
Ind_Vs = data(:,1:5);
%Dep_V is the Fault or no Fault column in my dataset
Dep_V = data(:,6);

[m,n] = size(data);

%train set is 80% of the data
Train_Ind_Vs = Ind_Vs(1:end-(m*.2),:);
%test set is 20% of the data
Test_Ind_Vs = Ind_Vs(end-(m*.2)+1:end,:);

Train_Dep_V = Dep_V(1:end-(m*.2),:);
Test_Dep_V = Dep_V(end-(m*.2)+1:end,:);

Test_Dep_V = table2array(Test_Dep_V);




Result(i).Index = i;

% the NB model
NB_Mdl = fitcnb(Ind_Vs,Dep_V);

[Predict_Test,Posterior] = predict(NB_Mdl,Test_Ind_Vs);


C = confusionmat(Test_Dep_V,Predict_Test)

TN = C(2,2);
FN = C(1,2);
FP = C(2,1);
TP = C(1,1);
```

```
%Dataset1(i).feature1 = peak_loc1(1,1);
Result(i).Sensitivity = TP / (TP+FN);
Result(i).Specificity = TN / (TN+FP);


end
```