National College of Ireland

Prediction and Classification of Electrocardiogram-Signals using Machine Learning using Apache Spark

MSc Research Project
Data Analytics

Rishab Rao Gauravaram
Student ID: x19202504

School of Computing
National College of Ireland

Supervisor: Dr. Catherine Mulwa

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Rishab Rao Gauravaram
Student ID:	x19202504
Programme:	Data Analytics
Year:	2020-2021
Module:	MSc Research Project
Supervisor:	Dr. Catherine Mulwa
Submission Due Date:	16/08/2021
Project Title:	Prediction and Classification of Electrocardiogram-Signals using Machine Learning using Apache Spark
Word Count:	7367
Page Count:	25

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Rishab Rao Gauravaram
Date:	23rd September 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Prediction and Classification of Electrocardiogram-Signals using Machine Learning using Apache Spark

Rishab Rao Gauravaram
x19202504

Abstract

The Electrocardiogram (ECG) is a common diagnostic system to identify cardiovascular diseases (CDVs). The aim of this research project is to develop data mining models using Support Vector Machines, Random Forest and Deep Neural Networks in Apache Spark that can be applied on large ECG data to extract and classify signals and predict cardiac Arrhythmia. This will help medical practitioners' timely diagnosis of diseases and help provide timely treatment to the patients. This research paper mainly focuses on classifying the ECG signal into five different classes of signals in the MIT-BIH dataset using Apache Spark. The performance of the models will be evaluated using Accuracy, Specificity, Sensitivity, and computation time. The Deep Neural Network model (local implementation) achieved a classification accuracy of 98.69% (sensitivity 97.01% and Specificity 99.29%). The Deep Neural Network model on Apache Spark achieved an accuracy of 93.94% (sensitivity 53.14%, specificity 93.89%). Overall, all the models created achieved high accuracy and specificity.

1 Introduction

Cardiovascular diseases (CVDs) are the number one cause of deaths across the globe; claiming 17.9 million lives every year. Recent studies show that one person dies every 36 seconds in the United States due to CVDs. Heart related diseases cost the United States $219 billion every year. To detect various heart related diseases, the heartbeat signals recorded by the electrocardiogram tests must be manually analyzed.

Electrocardiogram (ECG) signals are recordings of the electrical activity of the heart. It is a common and a standard test to monitor the electric activity and functioning of the heart and helps detect arrhythmia and other cardiovascular diseases. The ECG for a healthy heart has a characteristic shape or has a rhythm as shown in Figure 1. Arrhythmia is a term that is commonly used to define an abnormal or an irregular heartbeat. Irregularities in the heartbeat signals could mean that the patient is suffering from either arrhythmia or a cardiac disease.

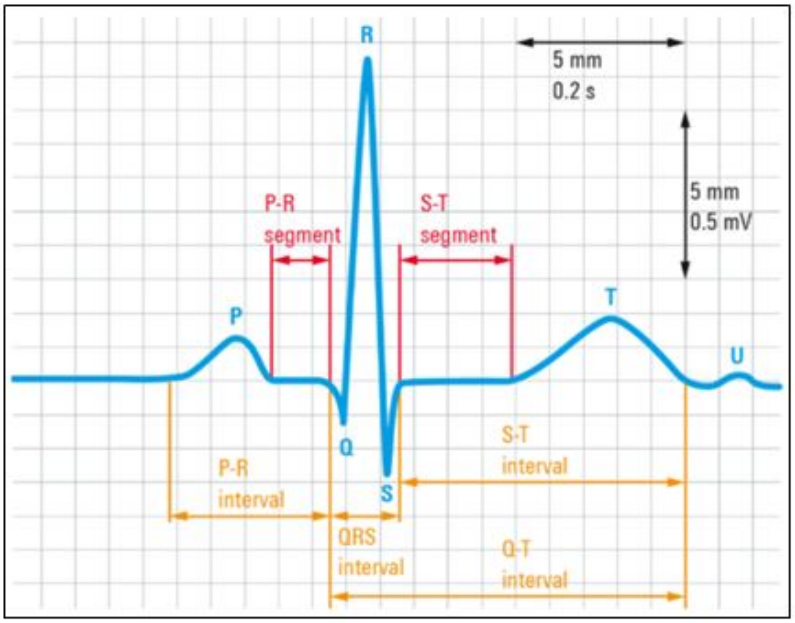


Figure 1: ECG waveform

Medical data is increasing day by day so, the variations and variety of signals for different people changes drastically, which makes it difficult for researchers to analyze. Many attempts to analyze ECG signals and research in the medical field are being carried out extensively, giving rise to many accurate and efficient algorithms to correctly identify arrhythmia.

The aim of this research paper is to accurately classify heartbeats into five categories - supraventricular ectopic (S), non-ectopic (N), ventricular ectopic (V), fusion ectopic (F), and unknown beats (Q). using machine learning techniques and apache spark. This helps providing early treatment to the patients suffering from Arrhythmia and prevent serious heart complications in the future.The motivation behind this project is the increasing risk factor in human health. The stressful life has drastically affected the human life, especially the human heart. The late detection of CVDs has resulted in 31% of human death rates according to the World Health Organization. Early detection and prediction of Arrhythmia using data Mining techniques will help the practitioners deal with large volumes of heart related data effectively thus helps in providing better medical services to the patients.

Big data analysis plays an important role in managing large amount of data in the healthcare field and is constantly improving the quality of the services provided to the patients. One of the major problems lies in accurately classifying large medical data. This can be resolved with the effective use of distributed systems, and data mining technologies. Therefore, a big data methodology has been introduced in this work to tackle the challenges faced while classifying the electrocardiogram signals.

The following research questions and objectives help detecting and predicting arrhythmia in human beings at early stages to provide better treatment and reduce the burden and

the time taken to analyze various ECG signals.

1.1 Research Questions and Objectives

RQ: *"To what extent do Machine Learning Techniques (Support Vector Machines, Random Forest and Convolutional Neural Network) in Apache Spark platform, enhance the identification and classification of heartbeat signals to reduce the efforts put by practitioners to analyze the ECG signals"*

Sub-RQ1: *"Can the developed classification model (RQ) be able to correctly classify various ECG signals in patients?"*

Sub-RQ2: *"To correctly predict Arrhythmia in patients using the proposed model"*

Sub-RQ3: *"To evaluate the performance of machine learning algorithms in Apache Spark"*

Table 1 summarises the research objectives in detail.

Table 1: Research Objectives

ID	NAME	DESCRIPTION	EVALUATION
1	Literature Review	To identify and critically evaluate peer-reviewed literature to classify ECG signals and detect arrhythmia	Review per-reviewed literature
2	Data Processing and Exploratory Data Analysis	Prepare the data for analysis. Perform signal augmentation, denoising and normalization techniques using various libraries.	Exploratory Data Analysis. Plot waveforms and wavelets,
3	Configure the Environments	Download and configure Anaconda, Hadoop, Apache Spark, to run Spark jobs using PySpark.	Check whether all the files have been installed or not
4	Implementation of Data Modelling	Implement a range of classification and predictive models to classify ECG signals and detect arrhythmia	
4.1	Experiment 1		
4.1.1	Support Vector Machines	Implement Support Vector Machine with hyper parameter tuning	Accuracy, Sensitivity, Specificity
4.1.2	Random Forest	Implement Random Forest	Accuracy, Sensitivity, Specificity
4.1.3	Convolutional Neural Networks	Implement Convolutional Neural Network using TensorFlow	Accuracy, Sensitivity, Specificity
4.2	Experiment 2		

4.2.1	Support Vector Machines	Implement Support Vector Machine on Apache Spark	Accuracy, Sensitivity, Specificity,
4.2.2	Random Forest	Implement Random Forest on Apache Spark	Accuracy, Sensitivity, Specificity,
4.2.3	Convolutional Neural Networks	Implement Convolutional Neural Network on Apache Spark	Accuracy, Sensitivity, Specificity,
5	Evaluation and Results	Evaluate and compare the models to implement the research objectives	Accuracy, Sensitivity, Specificity, Find the model that performs the best and that takes the least time to implement. Calculate the time taken to run machine learning algorithms in Apache Spark

The contribution of this research project is to implement, evaluate and predict Arrhythmia from a patient's ECG data using Machine Learning and Deep Learning techniques using apache spark. A minor contribution of this research project is the implementation of the classification and prediction models using Apache Spark to help detect Arrhythmia in the early stages so that appropriate treatment can be provided to the patients by the practitioners.

The flow of the technical report is divided and explained in various sections. Section 2 mainly deals with the literature review of related work and research done in the same field by previous researchers. Section 3 talks about the research methodology and the experiments carried out. Section 4 presents the design specifications and the design architecture used to implement the project. Section 5 explains the implementation and evaluation of the algorithms and models. Section 6 provides a discussion of the implementation and the results of the models and, the problems faced during implementation Section 7 concludes the project.

2 Related Work

2.1 Introduction

This literature review investigates the classification of 5 types of ECG signals and the detection of arrhythmia. This section is divided into multiple sub-sections – i) Machine Learning Techniques for Classification and Detection of Arrhythmia, ii) Literature Review on Pre-processing and Denoising Techniques, iii) Investigation on Apache Spark, iv) Identified Gaps in the existing models and v) Conclusion

2.2 Critique on Existing Models on Cardiovascular Diseases prediction

This section investigates different techniques and approaches implemented by other researchers and previous work done.

In the current age, healthcare is facing big data processing to support medical practitioners using decision making algorithms and tools. Electrocardiogram signals need to be pre-processed to generate insights and help detect various heart related disorders. Carnevale et al. (2017) proposed to implement Menard algorithm using apache spark to address the problem of ECG signals in a distributed environment. The use of spark streaming enabled the authors to perform continuous and real time processing of signals. The methodology was able to address the problem of ECG signals that are processed in a distributed environment. The ECG analysis was performed based on R-R intervals.

Qu et al. (2020) emphasise the need for ECG signals to be stored properly before they can be analysed. There is also a pressing need to process HRV analysis which are based on ECG signals. This helps processing the data timely and generate accurate results. A cloud computing approach using HRV-Spark (Heart Rate Variability) was proposed to compute HRV measures parallelly by utilising Apache Spark and QRS detection Algorithm. It was run on amazon web services using large-scale datasets. Apache Spark is one of the leading cluster computing frameworks use to process data and analyse them. This is because it can store large amount of data in the memory when the jobs are unique. The flexible storage mechanism addresses the storage and processing issues while handling big data. Experiments show that the scalability of the HRV-Spark is efficient and outperforms the Hadoop MapReduce in the same.

Physicians use the ECG records to monitor the patients' heart and detect Arrhythmia. Sometimes, symptoms do not always appear which could mislead the physician and can cause serious problems in the future. To solve this problem, Ilbeigipour et al. (2021). proposed a real-time arrhythmia detection using Apache Spark Streaming pipeline. The data is collected from MIT-BIH database, and the signals are classified into 3 classes – normal beats, RBBB (Right Bundle Branch Block) and atrial fibrillation arrhythmias using decision trees, random forest, and logistic regression models. The results show that the model scales very well and was able to reduce the runtime while using more class labels compared to previous work done in the same field. The pipeline created is portable, can analyse various biological signals such as ECG signals and EEG signals, and is able to accommodate new worker nodes to further increase the performance of the pipeline and is fault tolerant.

Electrocardiogram signals generally consists of unwanted noise. To analyse them, unwanted noise must be removed using various pre-processing and denoising techniques. In another study, Varatharajan et al. (2018) proposed to use linear discriminant analysis (LDA) to reduce the number of features present in the input ECG signal. This helps reduce unwanted frequency components and making it easier to select features and make the model more accurate. The authors also explain the need of Support Vector Machine (SVM) with a weighted kernel function to classify a greater number of features as compared to the traditional SVM to classify LBBB (Left Bundle Branch Block), RBBB (Right

Bundle Branch Block), PVC (Premature Ventricular Contraction), and PACs (Premature Atrial Contractions). The proposed LDA coupled with the enhanced kernel SVM produced low RMSE, MAPE, MAE, R2 and Q2 values compared to LDA with MLP, PCA with SVM and LDA with SVM.

Cardiovascular diseases are the number one cause of mortality rate in Indonesia (Ma'Sum et al.; 2016). A study was proposed that utilises a tele-ecg system built on hadoop framework (to deal with big data processing) to detect heart diseases and monitor the activity of the heart. The system was built on a cluster of 4 nodes (4 machines), which enabled it to handle 60 requests at a time. The use of decision trees and random forest on this framework improved the accuracy between 97.14% and 98.92%. The training process for random forest was noted to be faster than decision tree. However, the testing process in decision trees was faster than random forest. The copying time of data to the HDFS (Hadoop Data File System) for both the algorithms was similar. The authors intend to improvise the model by compressing the module, which in turn reduces the input size sent to the file system, thus handling a greater number of requests at a given time. Another enhancement is the introduction of multi servers.

2.3 Literature Review on Denoising Techniques

Noise is one of the main factors that can degrade the quality of an ECG signal. Unwanted signals and noise are generally caused by power line interferences, electromagnetic fields, random body movements and muscle contraction during respiration. Noise removal is a complicated process due to time variation in ECG signals. Joshi et al. (2013) studied various de-noising and filtering processes- FIR filtering, Fuzzy logic, Empirical Mode Decomposition, and wavelet transform. From the survey, it was clear that Equiripple notch filter is a good choice to remove power-line interference, Discrete Meyer wavelet is the best choice to remove motion artifact and electromagnetic noise, and empirical mode decomposition is the best approach to deal with baseline wander.

Signal-Noise residue algorithm for ECG signal denoising was proposed by Khan et al. (2011). The methodology was based on wavelet theory, the algorithm assumes that the ECG signal is a linear combination of noise and ECG signal. Using multiscale decomposition, noise can be easily removed with very less computation as it enables estimation of noise very accurate. In another study based on wavelet transform, Chmelka and Kozumplik (2005) used Wiener filters to suppress ECG signals that contained Electromagnetic noise. Depending upon the estimated noise level, the coefficients of wavelet transform are modified. This technique is feasible and generates good results for short impulse response.

Zhang and Ge (2008) proposed an algorithm to extract weak ECG signals in a noisy environment. It is a hybrid approach that is based on sub-band decomposition and the property of adaptation filter. This algorithm extracts the ECG signals with high precision and is highly stable. Cornelia and Romulus (2005) also proposed to use a wavelet transform to filter unwanted noise and signals in ECG signals. Three-level decomposing wavelets – db1, db3, sym2 and coifi were used for analysis, and it was found that db3 wavelets produced the best outputs, while sym2 generated the worst outputs.

Signal processing algorithms provide high performance and eliminate noise between 2

beats in an ECG signal. However, these algorithms often attenuate the characteristics of the ECG signals. To solve this problem, Singh and Tiwari (2006) proposed a selection procedure of the mother wavelet (optimal wavelet) for denoising the ECG signal, which helps retaining the signal peak close to the maximum amplitude. The selection function has proved to not only preserve the peaks of the ECG signal, but also is able to generate low Root Mean Square Errors. This makes the model more accurate and contains valuable information that can be used for diagnosis purposes.

2.4 Investigation on Apache Spark

Apache Spark is a data processing engine that can compute and process large datasets in a short period of time. It can distribute data processing tasks across multiple nodes/computers on its own or by utilising other distributed computing tools such as Hadoop. It is reliable, robust and the fastest distributed computing system. Unlike its predecessor (Hadoop), Spark uses memory computing rather than traditional disk computing. Apache Spark is used to perform extensive data analysis tasks; it has in-built libraries such as MLlib, Spark SQL which helps perform data analytics tasks. Spark also allows users to write scalable applications using Java, Scala, Python, R and SQL. Various tests and research show that Apache Spark performs 100x times faster than Hadoop while performing Machine Learning tasks. It can also perform well in data streaming tasks, an important feature its predecessor – Hadoop could not perform. Most machine learning tasks are iterative and requires data to be reloaded into the memory after each iteration. Apache Spark however, relies on RDD (Resilient Distributed Dataset), which allows Spark to store and read data only when needed. Unlike Hadoop, Spark is also designed to handle real time data streaming. Spark is also very flexible, it can be run on cluster mode, local mode, on Hadoop YARN (HBase, HDFS, Hive and Cassandra), Apache Mesos and even on cloud such as Amazon Web Services and Databricks.

2.5 Identified Gaps in the existing models

The above literature review explains the need for a "robust" and a "quick" heartbeat classification and cardiovascular disease detection model. Most of the models created can accurately classify the heartbeat signals and can accurately predict Arrhythmia. However, the biggest drawback in most of the approaches is that the machine learning models take a lot of time to train and generate results. Many research articles and journals have identified that the use of a Big Data tool like Apache Spark or Hadoop MapReduce can be used to solve this problem. However, Hadoop MapReduce is an old technology and there are significantly faster tools like Apache Spark. Combining the robustness and parallel computing power of Apache Spark with these machine learning algorithms can help accurately predict Arrhythmia.

2.6 Conclusion

Many research articles have been published which utilises Deep learning methodologies. The biggest drawback of using Deep Learning is the training time. It takes a lot of time to train the model to generate accurate results. Rana and Kim (2019) implemented a single layer neural network with 200 epochs. This approach has multiple flaws- the number of epochs is very high and could possibly lead to overfitting, and the time it takes to run

200 epochs is significantly high. However, it is possible to run the same model using a big data tool to significantly reduce the run time of the model.

Secondly, ECG signals require extensive of pre-processing. From the literature it is noticed that different waveform pre-processing methods and feature extraction play an important role in accurately classifying the signals and predicting Arrhythmia.

Based on literature, the gaps identified are rectified by implementing Machine Learning algorithms using Apache Spark. The following chapter presents a scientific methodology for a classification and a prediction model to support various cardio practitioners.

3 Research Methodology

Typically, data mining research methodologies are done in either KDD or CRISP-DM. In this case, a modified version CRISP-DM methodology has been adopted. This methodology has been modified and is implemented in the following stages – i) Understanding the Business. ii) Data selection - the MIT-BIH dataset has been used. The data consists primarily of 5 types of heartbeats. iii) The data for each patient is parsed and preprocessed using denoising and normalising techniques. The features and labels are then extracted from the preprocessed data. The data is then transformed to fit the models for classification and prediction. iv) Implementation - ECG classification and prediction of Arrhythmia (abnormal heartbeat) is performed using Deep Neural Networks, Random Forest, and Support Vector Machines in Apache Spark. v) The performance of the model is evaluated using accuracy, sensitivity, and specificity. vi) The final model is then deployed to various hospitals and clinics. Figure 2 represents the research methodology implemented.

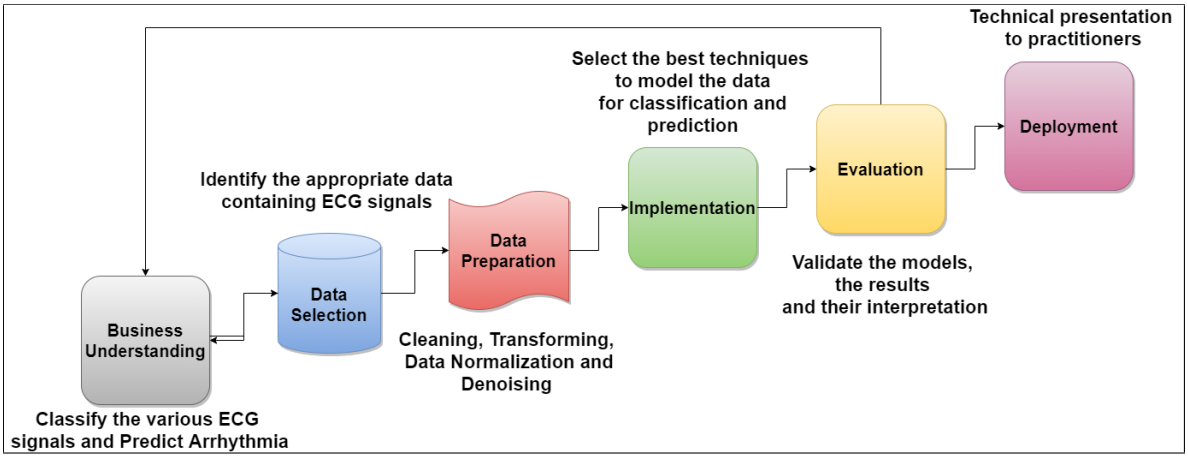


Figure 2: Research Methodology

3.1 Business Understanding

Deaths caused due to heart related diseases have been increasing at an alarming rate over the past few years, thus creating the necessity to develop robust and quick heartbeat

classification models. If cardiovascular diseases such as Arrhythmia are predicted at an early stage, it can potentially help doctors and practitioners saving lives by giving proper treatment.

3.2 Dataset Selection

For this research, the MIT-BIH dataset [1] (Massachusetts-Institute of Technology – Beth Israel Hospital) which is publicly available has been selected. It consists of ECG records from 47 subjects studied by the BIH laboratory between 1975 and 1979. Out of which, 23 records contain data for in-patients and out-patients. The remaining data consists of patients with severe Arrhythmia. The recordings are digitalized at 360 samples per second (360Hz frequency). More than two cardiologists have annotated the data. The data consists of approximately 110,000 annotations in total. For this research all 110,000 annotation will be used.

3.3 Data Processing and Transformation

Data processing is the first step of any data mining or machine learning process. The data needs to be processed and transformed so that it can be compatible with the models. In this case the signals are preprocessed using various denoising - signal Decomposition (3 levels of decomposition) using Discrete Wavelet Transform (DWT) and reconstructed using Inverse Discrete Wavelet Transform (IDWT) and normalizing techniques. The data (features and labels) is then converted into the necessary format to fit the machine learning model. The transformed data is stored in two different formats – Pickle file containing NumPy arrays and a Comma Separated Value file (.csv) containing the same data.

3.4 Modelling

After the data is processed and converted to the required format, data modelling takes place. In this stage, the models are created, trained, tested, and then evaluated. During the modelling stage, two separate experiments are conducted, and the data is transformed and stored in different formats for both the experiments. For the first experiment, Random Forest, Convolutional Neural Networks and Support Vector Machine are run locally on the system using python libraries. Whereas in the second experiment the same models are run on Apache Spark in Distributed environment using PySpark libraries. The task of both experiments is the same, i.e., to classify the ECG signals and predict Arrhythmia.

3.5 Evaluation Metrics

Following are the common terms used extensively and their meanings –

- True Positive (TP) - It is when the model correctly predicts the a positive class.
- True Negative (TN) - It is when the model correctly predicts a negative class.
- False Positive (FP) - It is when the model incorrectly predicts a positive class.

[1] `https://physionet.org/content/mitdb/1.0.0/`

- False Negative (FN) - It is when the model incorrectly predicts a negative class.
- Sensitivity – The ability of a model to correctly test the number of patients suffering from Arrhythmia.
  $$\text{Sensitivity} = \frac{TP}{TP + FN}$$
- Specificity – The ability of the model to correctly test the number of patients who do not suffer from Arrhythmia.
  $$\text{Specificity} = \frac{TN}{TN + FP}$$
- Accuracy – It is defined as the number of correct predictions made by the model for the total number of predictions.
  $$\text{Accuracy} = \frac{TN + TP}{TN + TP + FP + FN}$$

To test the classification and prediction performance of the models, Accuracy, Sensitivity, Specificity, and most importantly the computation time are investigated. Confusion matrix is one of the key elements while evaluating a model; using a confusion matrix, metrics such as sensitivity, specificity and accuracy are computed.

3.6 Deployment

Using Apache Spark, the models implemented can either be integrated using web integration or using the cloud platform. This can be made possible by leveraging Apache Spark's ability to stream data. Thus, real time patients' ECG data can be directly streamed into Apache Spark for classifying the heartbeats and predicting Arrhythmia.

4 Design Specification

This section will provide a detailed description of the design flow and the design specification of the models implemented.

The project design consists of 3 main layers - Data Tier, Application Tier, and Presentation Tier. In the data tier, the raw data is collected and processed extensively to remove redundant data and noise. The data is then stored on Hadoop Distributed File System (HDFS) which will then be imported into Apache Spark.

The application layer (logic layer) mainly works on the business logic which can help achieve the goal of the model. In this case, the business logic is to classify the ECG signals into their respective classes and predict Arrhythmia (therefore, a classification and a prediction model must be created). Using the business logic, the models that are created must achieve the goal to accurately classify the ECG signals and correctly predict arrhythmia. Two experiments are conducted - the first experiment is a local implementation of the classification and prediction models and the second experiment is the implementation of the classification and prediction models in Apache Spark. Using these two models, the various features of ECG signals are identified, extracted and then analysed.

In the presentation tier (client side), the results of the implementation are discussed with the stakeholders or clients (hospitals and clinics in this case). The model is then deployed if the clients are satisfied with the results and if they are not, then the the process will restart from the application.

The 3-tier implementation is summarised in Figure 3.

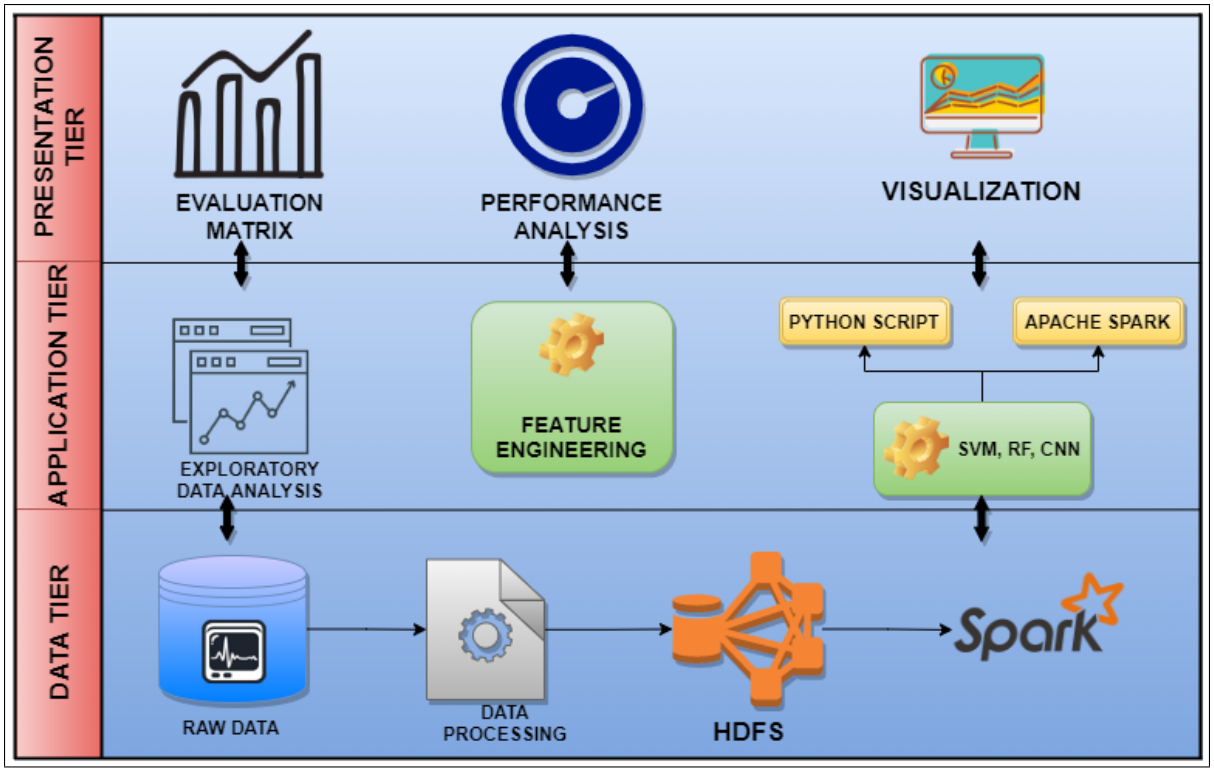


Figure 3: 3-tier Architecture Implementation

5 Implementation, Evaluation and Results of Electrocardiogram classification and prediction model

5.1 Introduction

This section will provide a detailed analysis of the implementation, evaluation and results of the models used to classify heartbeats and predict Arrhythmia. The implementation will be divided into two experiments – first experiment to implement the models using regular python libraries, and the second experiment to implement the models in Apache Spark in Distributed mode. Overall, the flow of the implementation, evaluation and results are depicted in Figure 4

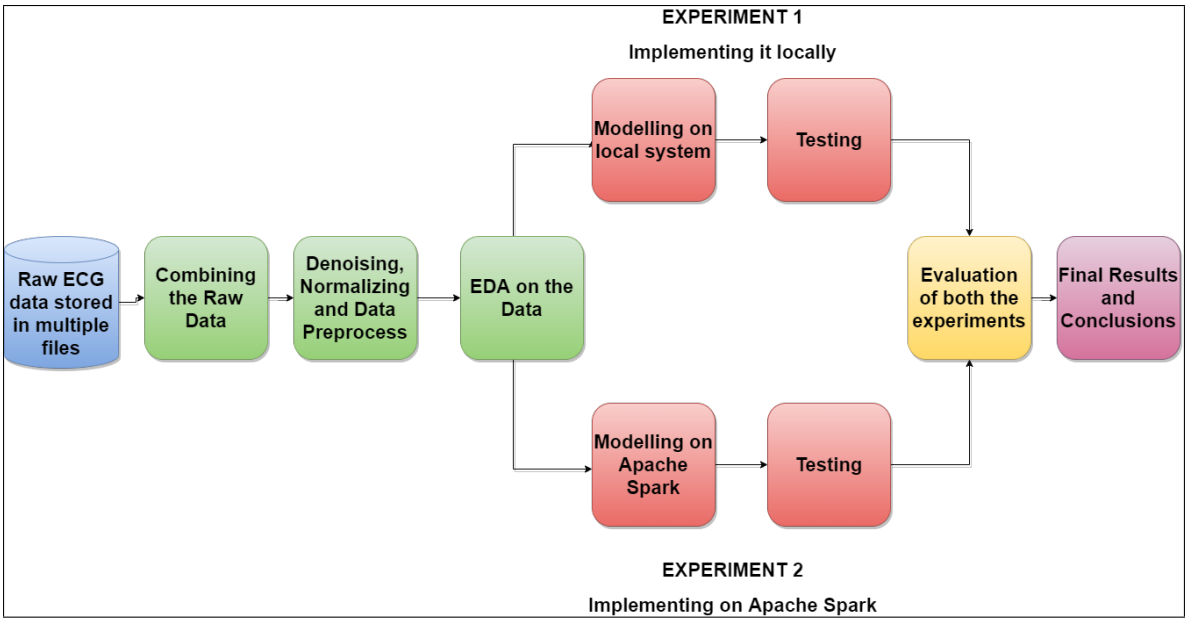


Figure 4: Flow of the implementation

5.2 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is the primary step in a data mining methodology that is used to perform initial investigations and analysis on the data to discover patterns, abnormalities and check for assumptions in the form of visual representations. It is necessary to get a good understanding and before processing the data and implementing a model on it.

The ECG data is investigated by counting the number of values in each of the five classes of signals to understand the data as shown in Figure 5

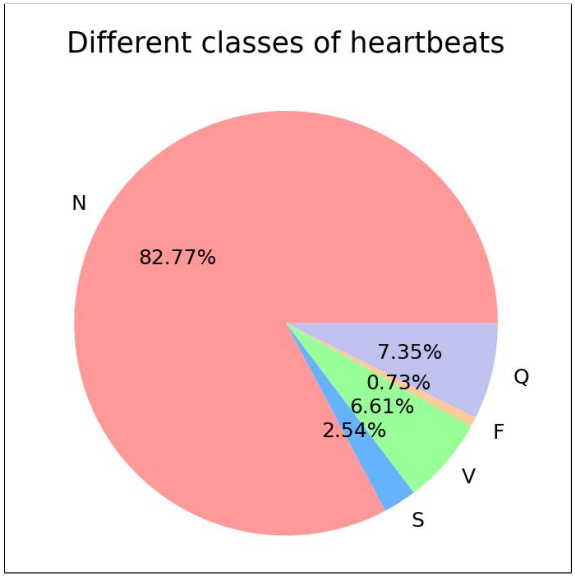


Figure 5: Count of values in each class

From the figure, it is seen that the N class has the most values compared to other classes. This is known as data imbalance problem. To solve this problem, the data must be augmented (adding values to the data in such a way that all the classes are balanced) either artificially or by adding real data. It is not advisable to artificially augment medical data, as it could mislead the model and generate incorrect results. Thus, the ideal way to deal with this is to augment the data using real data. However, this MIT-BIH dataset was created and annotated by the MIT university, hence it is hard to find data that could match the dataset being used for this research project. Therefore, no action will be taken in regards to data augmentation.

5.3 Data Processing, Data Transformation and Feature Extraction

The ECG data for 48 records is stored in 48 different files. The data in these files are parsed and are extracted using the 'wfdb library' as per the database guide [2]. A dictionary for the annotation of the beats is created as described in detail in the database [3]. Using this, different types of beats can be classified in to five major categories of heartbeats - Supraventricular Ectopic (S), Non-ectopic (N), Ventricular Ectopic (V), Fusion Ectopic (F), and Unknown beats (Q).

The raw extracted signals need to be processed and transformed before creating and testing a model. There are two types of wavelet transforms – Continuous Wavelet Transform (CWT) and Discrete Wavelet Transform (DWT). CWT uses all the wavelets over a scale of ranges, whereas the DWT uses a finite set of wavelets. For this research, DWT is used as it is easier to handle and analyze a small set of wavelets. The signals are decomposed and reconstructed using 3-levels of signal Discrete Wavelet Transform (DWT) and Inverse Discrete Wavelet Transform (IDWT) respectively. This is implemented using the bior1.3 wavelet transform.

Bior1.3 belongs to the biorthogonal class of waves, and it is symmetric in nature. Since the ECG signals are finite, display slow oscillations and are not localized w.r.t time and frequency, DWT was chosen instead of CWT (Ravindrakumar et al.; 2013).

According to the Nyquist rule, if the original signal has a frequency of f_{max}, then its sampled frequency must be equal to $f_s > 2f_{max}$. Using this rule, at each decomposition layer, the frequency axis is recursively divided into halves. The basic idea behind decomposition and reconstruction of a wavelet is low-pass (allows low frequency to pass through the filter) and high-pass (allows high frequency to pass through the filters) filtering respectively (Kumar and Singh; 2015).

After 3 levels of wavelet transform (DWT + IDWT), the signals are normalised. Normalizing is performed to bring all the data points in the data to a common scale without distorting the actual difference in the ranges of values. The features and labels are then selected and transformed to be tested on the machine learning models.

[2] `https://archive.physionet.org/faq.shtml#readfiletypes`
[3] `https://archive.physionet.org/physiobank/annotations.shtml`

Figure 6 shows the sample of signal Class N extracted from the ECG signals after signal processing and feature extraction.

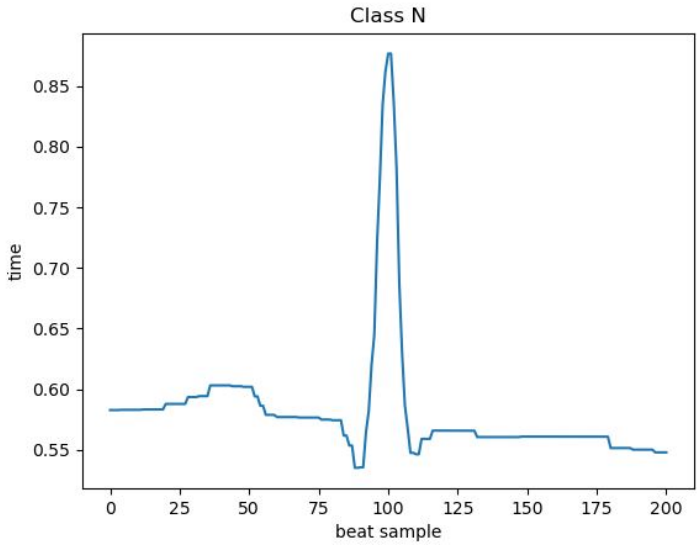


Figure 6: Signals of Class N

Figure 7 shows the sample of signal Class S extracted from the ECG signals after signal processing and feature extraction.

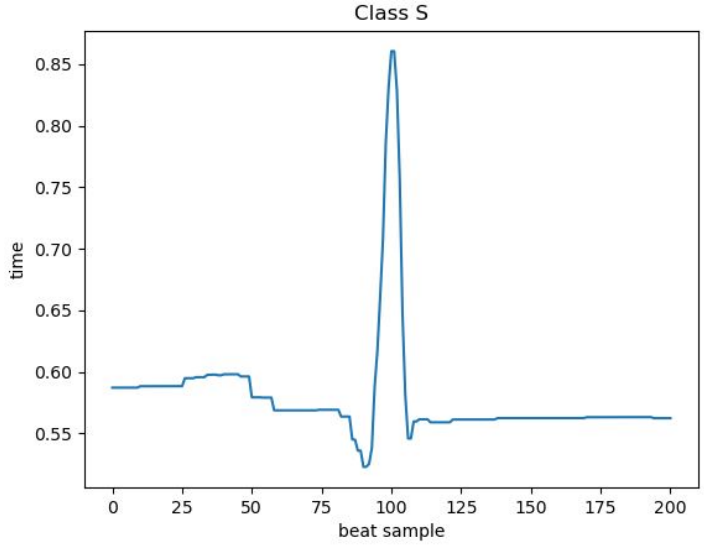


Figure 7: Signals of Class S

Figure 8 shows the sample of signal Class V extracted from the ECG signals after signal processing and feature extraction.

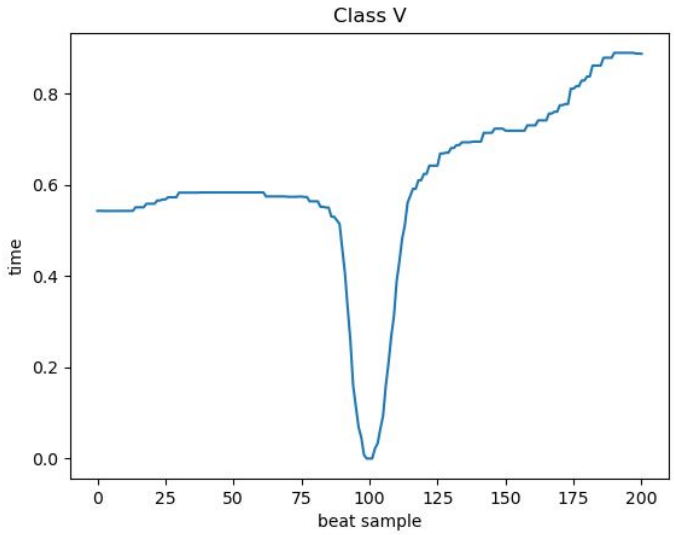


Figure 8: Signals of Class V

Figure 9 shows the sample of signal Class F extracted from the ECG signals after signal processing and feature extraction.

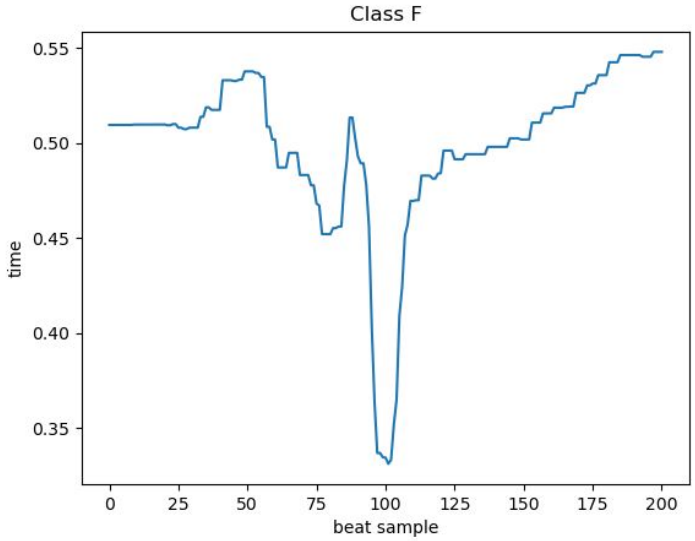


Figure 9: Signals of Class F

Figure 10 shows the sample of signal Class Q extracted from the ECG signals after signal processing and feature extraction.

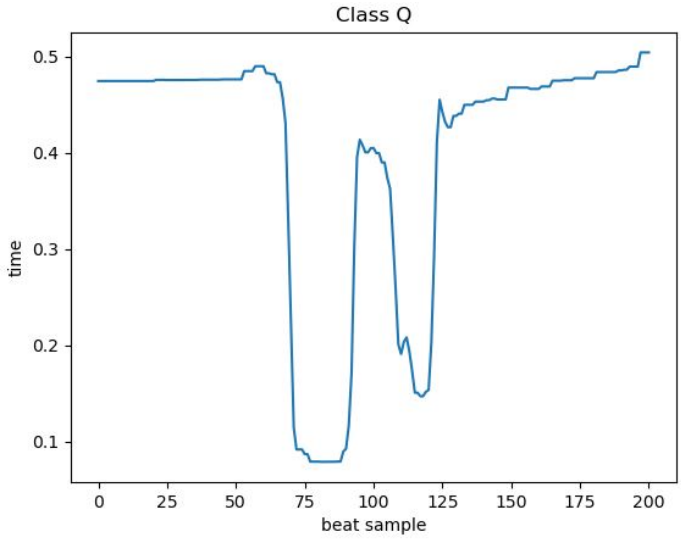


Figure 10: Signals of Class Q

5.4 Experiment 1 – Local Implementation of Classification and Prediction Models

5.4.1 Implementation, evaluation, and results of Deep Neural Network

Implementation: A Deep Neural Network (DNN) is built using 4 layers, 2 of them being Conv1D layers and the remaining 2 being Dense layers. In 3 layers, 'relu' activation layer is used while the last layer (the output layer) uses 'softmax' activation layer. The loss function used in this case is 'categorical crossentropy' and the optimizer used is 'adam'. Dense layers are used when association can possibly exist among the features in the data. The Conv1D layers are used to classify and detect various features into their respective groups. Since images are not used and the wavelets are one dimensional, Conv1D layer is used. The DNN model is trained on the Training data for 50 epochs. Accuracy, Specificity and Sensitivity are the metrics used to evaluate the model.

Evaluation and Results:
From the results, the model can accurately classify 98.69% of the ECG signals. Sensitivity (97.01%) is used to test the probability of patients suffering from Arrhythmia. In contrast, Specificity (99.29%) is used to test the probability of patients who do not suffer from Arrhythmia. Overall, the DNN model performed well. However, the computation time of the model was quite high, it took 1057.75 seconds to execute. Figure 11 represents the output of the model.

```
Epoch: 50 | SE: 0.9701 | ACC 0.9869 | SP: 0.9929
Final Result:

SE: 0.9701 | ACC: 0.9869 | SP: 0.9929
--- 1057.7538135051727 seconds ---
```

Figure 11: Results of Deep Neural Network

5.4.2 Implementation, evaluation, and results of Support Vector Machine

Implementation: The hyper tuned SVM model is implemented using "sci-kit learn" library in python. 3 folds of cross validation is used to split the train data and the model is trained over 10 iterations. The gamma parameter of the radial bias function controls the influence of a single data point. Higher value of gamma could lead to overfitting of the model. Typically, the gamma value is between 0.0001 and 10 and the c parameter 0.1 to 100. In this case, the gamma parameter is between 0.001 and 0.1 and c parameter is between 1 and 10. The evaluation metrics used to evaluate the model are accuracy, specificity and sensitivity.

Evaluation and Results:
Like the DNN model, the SVM model also performed well. This model was able to accurately classify 96.37% of the signals into their respective classes and predict Arrhythmia. The model was able to test positive for 74.41% of patients suffering from Arrhythmia and test negative for 96.54% patients free of Arrhythmia. However, this model took 6156.77 seconds to implement which is significantly higher compared to the DNN model. This model took the longest time to execute. Figure 12 represents the output of the model.

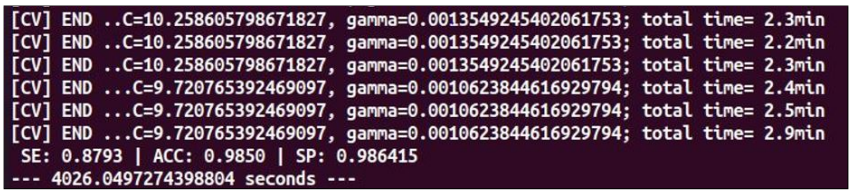


Figure 12: Results of Support Vector Machine

5.4.3 Implementation, evaluation, and results of Random Forest

Implementation: Random forest (RF) is based on ensemble algorithm which can be used to solve both classification and regression problems. In this case it is used to solve a classification problem by combining multiple classifiers to produce a solution to a complex problem. Random forest is a combination of multiple Decision Trees. The random forest classification model is implemented using "sci-kit learn" library in python. A total of 100 trees are used for classifying the features. The data is split in the ratio of 80:20 for Training and Testing respectively.

The evaluation metrics used to evaluate the model are accuracy, specificity and sensitivity.

Evaluation and Results:
The random forest model was able to accurately classify 98.09% of the signals and predict Arrhythmia. It correctly tested 83.62% of the patients suffering from Arrhythmia and 98.13% patients not suffering from Arrhythmia. With its computation time of 359.751 seconds, it was the fastest model to execute in the first experiment. Figure 13 represents the output of the model.

Figure 13: Results of Random Forest

5.5 Experiment 2 – Implementation of Classification and Prediction Models on Apache Spark

5.5.1 Implementation, evaluation, and results of Deep Neural Network

Implementation: Creating and implementing a deep neural network (DNN) in Apache Spark is different from implementing Deep Neural Networks locally. In this case, a DNN is built using 3 Dense layers, 2 of which use 'relu' activation and the last one (output layer) uses 'softmax' activation. The loss function used in this case is 'categorical crossentropy' and the optimizer used is 'adam'. However, a Hyper optimization library is used to enable the DNN to be implemented parallelly in the Apache Spark Distributed environment. HyperOpt is an open-source framework that works using Bayesian optimization. It is designed for large scale models with customizable parameters that can enable experiments to be scaled across multiple nodes. The model was implemented over five epochs, with each epoch iterated five times. The model is evaluated using accuracy, sensitivity and specificity.

Evaluation and Results:
The classification and prediction accuracy of the deep neural network model was 93.94%. With a Specificity of 93.89%, it can test negative for most of the patients that are free from Arrhythmia. However, the Sensitivity was low. With a Sensitivity of 53.55%, up to 47% of the patients suffering from Arrhythmia did not test positive. With an execution time of 827 seconds, it is the slowest to execute compared to the models in Experiment 2. Figure 14 represents the output of the model.

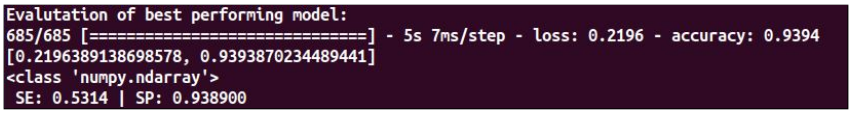


Figure 14: Results of DNN in Apache Spark

5.5.2 Implementation, evaluation, and results of Support Vector Machine

Implementation: The SVM model is built on Apache Spark using "PySpark ML" library. In this case, one vs the rest classifier is used to attach a binary classification algorithm for multiclass classification on the linear model because the features are classified under five classes of predictors. Basic transformation of the data (combining all the features into a single vector or a list using Vector Assembler) is required before the model can be trained and tested on it. The evaluation metrics employed to evaluate the model are accuracy, sensitivity and specificity.

Evaluation and Results:
After examining the results, the SVM model had mediocre performance. The classification and prediction accuracy of the model was 93.08%. The sensitivity of the model was extremely low;the model was only able to test correctly 20.01% of patients suffering from Arrhythmia. In simple terms, up to 80% of the patients suffering from Arrhythmia were not tested positive by the model. However, it was able to test 80.05% of the patients that did not suffer from Arrhythmia.

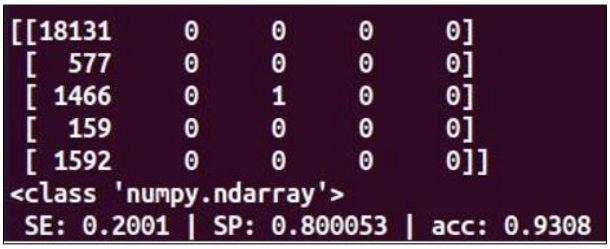


Figure 15: Results of SVM on Apache Spark

5.5.3 Implementation, evaluation, and results of Random Forest

Implementation: Random forest is implemented using built-in machine learning libraries. One of the advantages of using an RF model is that the number of trees assigned does not result in overfitting or underfitting of the model. Unlike other machine learning models, it does not require tuning of parameters. The only important parameter to be included is the number of trees used for the classification model. In this case a total of 200 trees are used in the model. To evaluate the model, accuracy, sensitivity and specificity are used.

Evaluation and Results:
The random forest model had a classification and prediction accuracy of 91%. Like the SVM model, the RF model struggled to test positive for patients suffering from Arrhythmia. The model's sensitivity was 42.85%. This implies that up to 58% of the patients suffering from Arrhythmia were not tested positive. However, the model's specificity was 89.92%, which means that only 11% of the healthy patients were not tested negative.

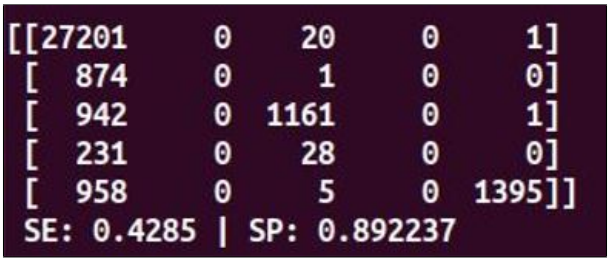


Figure 16: Results of RF on Apache Spark

6 Discussion

The objective to develop various model to classify the Electrocardiogram signals and predict Arrhythmia was accomplished. All the models are compared w.r.t Accuracy, Sensitivity and Specificity. Apart from the evaluation metrics, the computation time for each model is also considered for comparison.

At the first glance, accuracy seems to be one of the primary metrics to evaluate a machine learning model. However, accuracy alone is not enough while working with class imbalance data sets and medical data as seen in Experiment 2. Sensitivity and Specificity are given more importance as these metrics test whether the model can correctly test whether the patient is suffering from a disease or not. The following section will provide a detailed discussion of the results and their interpretations.

6.1 Experiment 1 – Local Implementation Using Python

From the results and evaluations, the Deep Neural performed the best statistically. It had the highest Accuracy, Sensitivity and Specificity. With a sensitivity and specificity of 97.01% and 99.29% respectively, the model will fail to test positive for only 3% of the patients suffering from Arrhythmia. And with a Specificity of 99.70%, the model will fail to test negative for only 0.30% of the patients. These numbers are considered very good compared to other screening tests. In terms of execution time, this Model beat the Support Vector Machine model by a huge margin, however, the execution time for this model is still high, with an execution time of 1057.75 seconds this is the second slowest model to execute compared to all the models.

The second-best model in this experiment is the Random Forest model. It had a Specificity and Sensitivity of 98.13%, and 83.62% respectively. Although the Specificity is high, it is noticed that the Sensitivity is lower than the DNN model. Although these statistical values are high, in real life while diagnosing Arrhythmia, 16% of the patients suffering from Arrhythmia will not be tested positive and will not receive proper treatment which could lead to serious complications and even result in death. The Specificity, however, is not suspiciously high and it can be concluded that there is a less chance for the RF model to over predict compared to the DNN model and correctly test negative for patients free of Arrhythmia. This model was the third fastest model to execute.

The Support Vector Machine model had good performance metrics as well. However, the sensitivity of the model was significantly lower than DNN and RF models. In terms of accuracy and specificity, its performance is comparable to the RF and DNN models. It's biggest drawback is the execution time. The model took around 6200 seconds (5x slower than DNN model).

Overall, all the models in the experiment performed well. While analysing the models purely on Accuracy, and Sensitivity, they are ranked as follows – i) DNN, ii) RF and iii) SVM. Upon improving the Random Forest model further, it could potentially even outperform the DNN model. But due to the machine constraints and memory constraint only 200 trees of random forest were used. The number of trees does not result in overfitting or underfitting of the model.

6.2 Experiment 2 – Apache Spark in Distributed Mode

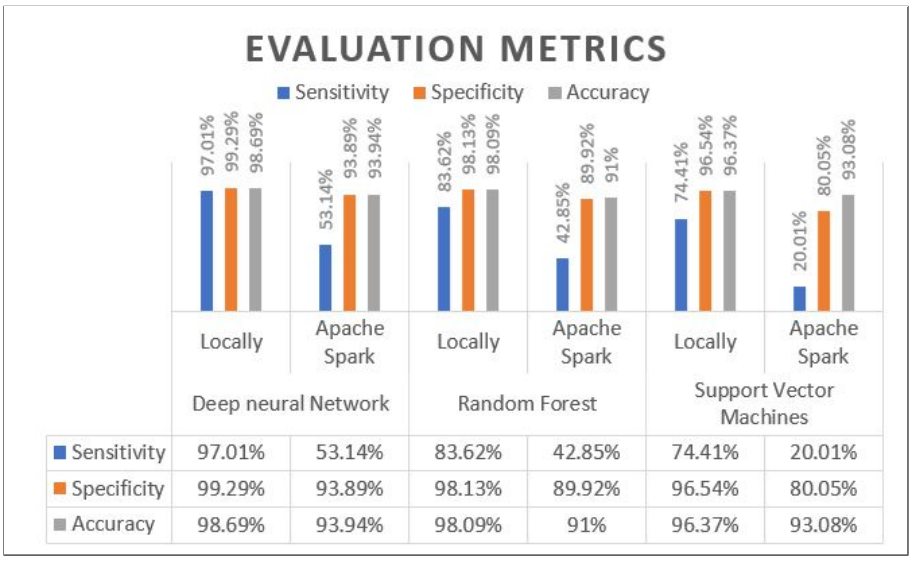
From the results and evaluation, the Deep Neural Network model clearly performed significantly better than the Random Forest and the Support Vector Machine models statistically. Not only did it have a high classification accuracy, it also had a high Specificity value and a mediocre Sensitivity value. This model will only fail to test negative for up to 7% of patients who do not actually suffer from the disease which is a good sign. However, the Sensitivity was surprisingly low, it fails to test positive for 47% of the patient who suffer from Arrhythmia.

The Support Vector machine model had a high classification accuracy of 93.08%. However, the sensitivity of the model was alarmingly low. With a Sensitivity of only 20.01%, it was the worst performing model, as it is not able to test positive for even 50% of patients suffering from Arrhythmia. This could potentially lead to the death of many patients. However, it's specificity is quite high and is able to test negative for patients free from arrhythmia.

The Random Forest model performed better than the SVM model. The model had high Accuracy and Specificity and a low Sensitivity making it one of the worst performing models alongside the SVM model. However, it had one of the fastest execution times, making it the fastest model to execute. Although the RF model and SVM model completed execution in a short amount of time, the biggest trade-off was the Sensitivity.
Overall, both the RF and the SVM models could be implemented to only classify the ECG signals and test patients for negative result of arrhythmia.

6.3 Comparison between the developed models

All the developed models – DNN, SVM, RF from both the experiments are compared below in the Figure 17



	Deep neural Network		Random Forest		Support Vector Machines	
	Locally	Apache Spark	Locally	Apache Spark	Locally	Apache Spark
Sensitivity	97.01%	53.14%	83.62%	42.85%	74.41%	20.01%
Specificity	99.29%	93.89%	98.13%	89.92%	96.54%	80.05%
Accuracy	98.69%	93.94%	98.09%	91%	96.37%	93.08%

Figure 17: Comparison of Developed Models

It is observed that all the models had high accuracies (90% and above) and specificities(80% and above). However, only the Random Forest and Support Vector machine models implemented in Apache Spark had extremely low Sensitivities followed by DNN model implemented on Apache Spark.

The computation time for all the models are shown in the Figure 18

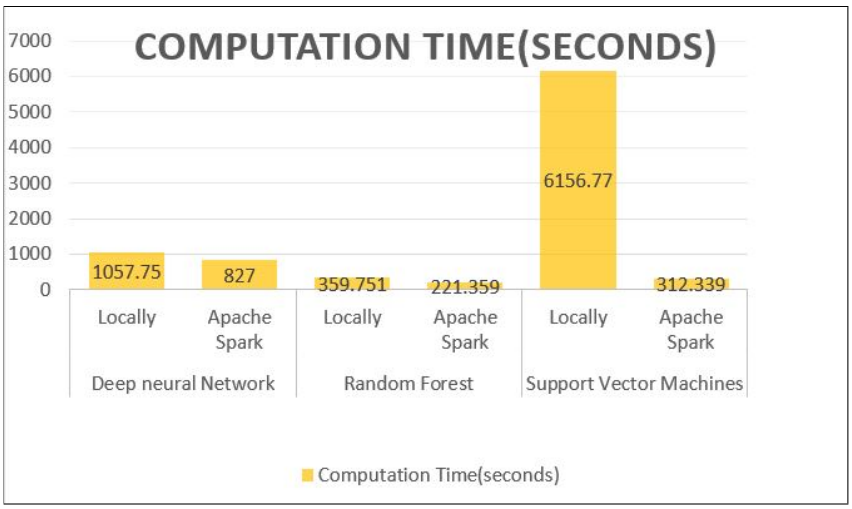


Figure 18: Execution time of the Developed Models

It is observed that the models performed in Apache Spark were faster than their counterparts implemented on local system.

6.4 Conclusion

Sensitivity and Specificity play a crucial role in identifying patients with and without diseases, thus more focus must be given to these metrics. Ideally the Sensitivity and Specificity of a model must be high. However, there is a chance that an extremely high specificity percentage will not only capture patients suffering from Arrhythmia, but also patients that suffer from Arrhythmia. This could unnecessarily cause concerns and confusion to patients who suffer from the disease. This problem has raised many concerns in the medical field as wrong predictions and tests could lead to serious complications. Similarly, a lower percentage of Sensitivity could also mean that a large population of patients suffering from diseases would not test positive for the same, thus preventing them from the required treatment and could also result in death. Ideally, a model should be able to provide high Sensitivity and Specificity, but at the same time must avoid over detection as it could raise some flags regarding the treatment.

Overall, the Apache Spark models were fast to execute but had low sensitivity values.

7 Conclusion and Future Work

Cardiac Arrhythmia could lead to serious heart complications and could also result in death. Early detection of Arrhythmia helps save patients' lives. And the same has been

addressed in this research paper. The heartbeat signals for 48 records were processed using wavelet transforms to extract features and normalising techniques, however the imbalance problem was not solved using artificially augmented data as it did not cause any problems during the implementation. The cleaned heartbeat signals were then classified into five different classes – supraventricular ectopic (S), non-ectopic (N), ventricular ectopic (V), fusion ectopic (F), and unknown beats (Q) using machine learning and deep learning techniques in both local and apache spark environment. The implemented models were evaluated based on accuracy, sensitivity and specificity. From the results, all the models had a high classification accuracy and specificity, four of the six models showed high performance based on sensitivity, specificity and accuracy.

This methodology has resulted in creating models that perform well, however, following are the changes that could be incorporated to the implemented model to obtain better results –

1. Handling Data processing – The data processing methodology addressed to the issued related to wavelet transform, unwanted noise removal and transformation of data. It did not address the unbalanced data in the dataset. Unbalanced data is a problem that arises when the data is not distributed equally across all the classes. This, however, did not cause any problems during the modelling stage. Unbalanced data can be handled in two ways – Artificially augmenting the data and augmenting the data using real life data. The original idea was to augment the data artificially. This idea was scraped because imbalanced data can mislead the model if not done right. And since the data is related to medical data, it is unwise to artificially augment it before testing it thoroughly.

2. To solve the data imbalance problem using modelling algorithms, cost sensitive models such as Cost-sensitive Logistic Regression, Cost-sensitive Decision Trees and Cost-sensitive Support Vector Machines can be used. These algorithms pay more attention to the minority classes and reduce the number of false positives and false negatives, thus improving the performance of the model.

3. Fine tuning the parameter for the models implemented in Apache Spark – The problem with implementing SVM and RF on Apache Spark is the flexibility to tune the parameters. It is much easier to tune the parameters using k-folds of cross validation for binary classification on Apache Spark. However, when implementing a multiclass classification on Apache Spark, the cross-validation metrics must be manually re-written for the specific model. This could be one of the reasons why SVM in apache spark resulted in low sensitivity.

Based on the implementation and results, the project has completely answered all the research questions in Section 1.1. Additionally, all the research objectives in Section 1.1 have been achieved.

Overall, the deep neural network, random forest and support vector machines in the local implementation had the highest performances, and the deep neural network using apache spark showed potential to be one of the best models. These four models can be applied in real-life scenarios to help practitioners predict Arrhythmia and provide necessary treatment.

Future Work:
This research work can be implemented for real-life streaming data, as the Apache Spark model can handle data streaming very well. All the models used for this research paper were supervised models. It would be interesting to see the performance of an unsupervised model like K-means clustering algorithm and time series-based algorithm to classify the signals and predict Arrhythmia.

References

Carnevale, L., Celesti, A., Fazio, M., Bramanti, P. and Villari, M. (2017). Heart disorder detection with menard algorithm on apache spark, *European Conference on Service-Oriented and Cloud Computing*, Springer, pp. 229–237.

Chmelka, L. and Kozumplik, J. (2005). Wavelet-basedwiener filter for electrocardiogram signal denoising, *Computers in Cardiology, 2005*, IEEE, pp. 771–774.

Cornelia, G. and Romulus, R. (2005). Ecg signals processing using wavelets, *University of Oradea: Electronics Department, Oradea, Romania* .

Ilbeigipour, S., Albadvi, A. and Akhondzadeh Noughabi, E. (2021). Real-time heart arrhythmia detection using apache spark structured streaming, *Journal of Healthcare Engineering* **2021**.

Joshi, S. L., Vatti, R. A. and Tornekar, R. V. (2013). A survey on ecg signal denoising techniques, *2013 International Conference on Communication Systems and Network Technologies*, IEEE, pp. 60–64.

Khan, M., Aslam, F., Zaidi, T. and Khan, S. A. (2011). Wavelet based ecg denoising using signal-noise residue method, *2011 5th International Conference on Bioinformatics and Biomedical Engineering*, IEEE, pp. 1–4.

Kumar, A. and Singh, M. (2015). Optimal selection of wavelet function and decomposition level for removal of ecg signal artifacts, *Journal of Medical Imaging and Health Informatics* **5**(1): 138–146.

Ma'Sum, M. A., Jatmiko, W. and Suhartanto, H. (2016). Enhanced tele ecg system using hadoop framework to deal with big data processing, *2016 international workshop on big data and information security (IWBIS)*, IEEE, pp. 121–126.

Qu, X., Wu, Y., Liu, J. and Cui, L. (2020). Hrv-spark: Computing heart rate variability measures using apache spark, *2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, IEEE, pp. 2235–2241.

Rana, A. and Kim, K. K. (2019). Ecg heartbeat classification using a single layer lstm model, *2019 International SoC Design Conference (ISOCC)*, IEEE, pp. 267–268.

Ravindrakumar, S., Shruthilaya, K., Rekha, M., Rajapriya, B. and Saranya, M. (2013). An hybrid method using wavelet transform and slope thresholding for cardiogram denoising and delineation, *Proceedings of the International Conference on Mathematical Computer Engineering-ICMCE*, Vol. 1, pp. 29–30.

Singh, B. N. and Tiwari, A. K. (2006). Optimal selection of wavelet basis function applied to ecg signal denoising, *Digital signal processing* **16**(3): 275–287.

Varatharajan, R., Manogaran, G. and Priyan, M. (2018). A big data classification approach using lda with an enhanced svm method for ecg signals in cloud computing, *Multimedia Tools and Applications* **77**(8): 10195–10215.

Zhang, W. and Ge, L. (2008). Application of adaptive matched filter to ecg signal detection, *2008 7th World Congress on Intelligent Control and Automation*, IEEE, pp. 7960–7964.