CLASSIFICATION OF ECG SIGNAL USING HYBRID DEEP NEURAL NETWORK

By

MD. ASFI-AR-RAIHAN ASIF (160021079)

NAFEW AHMED (160021125)

MD. MOHI UDDIN KHAN (160021163)

An Undergraduate Thesis Submitted to the Academic Faculty in Partial Fulfillment of the Requirements for the Degree of

BACHELOR OF SCIENCE IN ELECTRICAL AND ELECTRONIC ENGINEERING

Department of Electrical and Electronic Engineering
Islamic University of Technology (IUT)
Gazipur, Bangladesh

March 2021

A dissertation on

CLASSIFICATION OF ECG SIGNAL USING HYBRID DEEP NEURAL NETWORK

Approved by:

_____

Dr. Golam Sarowar

Supervisor and Professor
Department of Electrical and Electronic Engineering (EEE)
Islamic University of Technology (IUT)
Boardbazar, Gazipur-1704, Bangladesh

Date: 10th March, 2021

Declaration of Authorship

This is to certify that the work presented in this thesis paper is the outcome of research carried out by the candidates under the supervision of Dr. Golam Sarowar, Professor, Department of Electrical and Electronic Engineering (EEE), Islamic University of Technology (IUT). It is also declared that neither this thesis paper nor any part thereof has been submitted anywhere else for the reward of any degree or any judgment.

Authors

Md. Asfi-Ar-Raihan Asif
ID-160021079

Nafew Ahmed
ID-160021125

Md. Mohi Uddin Khan
ID-160021163

Dedicated to

Our beloved parents & teachers whose support made it all possible for us

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

KNN	K-Nearest Neighbor
RFC	Random Forest Classifier
SVM	Support Vector Machine
SGD	Stochastic Gradient Descent
AdB	Adaptive Boosting
XGB	Extreme Gradient Boosting
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
DCNN	Deep Convolutional Neural Network
VAE	Variational Auto Encoder
HDNN	Hybrid Deep Neural Network
RSCV	Randomized Search Cross-Validation
GSCV	Grid Search Cross-Validation
HPO	Hyperparameter Optimization
DHP	Default Hyperparameter

ACKNOWLEDGEMENTS

Foremost, we would like to express our sincere gratitude and gratefulness to the Almighty Allah; without His graces and blessings, this study would not have been possible.

Acknowledging all who helped us complete this work, we wish to compliment the university's significant role and the department that has been very amiable to us during the entire period of our research.

We are indebted to our honorable supervisor, Dr. Golam Sarowar sir, for his selfless support, motivation, patience, enthusiasm, and extensive knowledge of the relevant fields. His continuous guidance and careful supervision kept us going even during the hardest of hours.

Lastly, our warmest tribute to our parents, family members, and friends, whose moral support and well wishes benefitted us spiritually in achieving our goals.

ABSTRACT

This dissertation studies a comprehensive approach to evaluating the performance of different machine learning and deep learning algorithms to classify five ECG signal categories. A novel algorithm is also proposed to achieve the same objective efficiently. Cardiovascular disease is responsible for a prominent amount of mortality among humankind is detected by analyzing ECG signals. ECG signal classification is an arduous task since sometimes the abnormal heartbeats are too similar to categorize. Most of the patients with heart diseases come to the doctor when the person is severely attacked. Therefore, doctors or medical persons cannot take much time to start the treatment. The heart is the most sensitive organ of the body, a misapprehension in classification can cause death to the patient. Machine learning and deep learning can be handy tools for the classification of the ECG signal quickly and efficiently. A Famous MIT-BIH ECG signal dataset was utilized to train and test the models. Six machine learning algorithms and five deep learning algorithms were studied with efficient hyperparameter optimization technique, and their performance was evaluated. Finally, a novel Hybrid Deep Neural Network (HDNN) was proposed which provided the best accuracy of 99.23% among all the algorithms studied for the classification of ECG signal. A detailed comparative analysis of performance with all other algorithms was carried out in terms of accuracy, precision, recall, and F-1 score.

Chapter 1

INTRODUCTION

1.1 Introduction

One of the significant illnesses affecting human health is cardiovascular disease. Mortality from cardiovascular diseases (CVDs) ranked first of all reasons of death today, according to the World Health Organization's estimation. More than 17.7 million people have died from CVDs or around 31 percent of all deaths. More than 75% of these deaths have occurred in developing nations. What's more, cardiovascular disease (CVD) prevalence and mortality are still increasing. To control and avoid CVDs, frequent monitoring of heart rhythm has also become a progressively significant and necessary fact. Arrhythmia in coronary diseases is a significant category of diseases. [1]

Arrhythmias are unusual beats of the heart due to the heart's inappropriate electrical activities that may cause severe risks, such as heart disease, stroke, sudden cardiac death, etc. Heartbeats are commonly divided into several different forms, as like normal beat (N), right bundle branch block beat (RBBB), premature ventricular contraction (PVC), atrial premature contraction (APC), etc. Every one of them exhibits varying signs and needed different forms of medication. It's also essential to correctly identify different types of arrhythmias to provide efficient, effective, and timely therapies. [2]

Arrhythmia may happen with other cardiovascular diseases or on its own. The diagnosis of Arrhythmia relies primarily on an electrocardiogram (ECG). The ECG (electrocardiogram) is a significant advanced medical instrument that tracks the heart’s excitability, transmission, and recovery process. A major task for the automated diagnosis of cardiovascular disease is the automatic identification of abnormal heartbeats from ECG signals. [1]

Four different measures form most current rhythm/morphology abnormality detection models: 1) acquisition of ECG signals; 2) analysis of data; 3) extraction of features; 4) classification. Each activity can produce errors and cause false detection. ECG signal analysis has recently been successfully extended to a deep learning-based approach that assembles feature extraction and classification into one operation to overcome those challenges. The ECG deep learning signal processing architecture has a better ability to draw on functionality that can learn deep characteristics from the signals generated and automatically optimize the model for a high degree of accuracy. [3]

For deep learning methods, each model is made of stacked multi-hidden layers. Each hidden layer, meanwhile, includes expanded parameters. The number of parameters that need to be trained in the deep learning model is therefore high. The deep learning model demands a sufficient amount of coaching by sufficient balanced coaching expertise to achieve a high degree of accuracy. Furthermore, in operation, the incidence rates of different anomalies are frequent. Usually, it results in a related degree of imbalanced distribution of numerous abnormalities in collected ECG signals between minor and large cases. This class imbalance prohibits the deep network from being instructed on how to assess the type of minority**.** [3]

1.2 Significance of Research

Automaticity, auto-rhythmicity, excitability, conductivity, contractility, refractory period, all or none law & functional syncytium comprises the properties of cardiac muscles. A healthy person's heart muscle should initiate normal cardiac impulse by SA node without an external stimulus. It should contract and expand due to de/repolarization after a regular interval maintaining a proper rhythm to conduct impulse from one heart muscle cell to another cell. Both sympathetic and parasympathetic nerves supply the heart in the cardiac plexus mainly through β_1, β_2 receptors and SA-AV nodes are supplied via muscarinic (M2) receptors.[4]

Normal heartbeat has the characteristic sequence of P-wave, followed by PR interval, then QRS Complex, ST interval, T wave, and often U wave, which can be viewed in ECG machine. [4]

In the first step, atrial depolarization occurs since action potentials (voltage) start the journey from the SA node towards the AV node. Atrial depolarization invokes atrial systole, which is viewed as the P wave in the ECG machine. Action potentials further spread from the Atrio-Ventricular node sweeping the bundle of HIS, followed by bundle branches and then Purkinje fibers. Transportation of action potential causes ventricular depolarization, which further invokes ventricular systole. These activities are viewed as the QRS complex in the ECG machine. The time surpassed for the cardiac impulse to spread over the atrium is viewed as the PR interval. Ventricular diastole is induced when action potential passes out of the ventricles; this causes Ventricular repolarization viewed as the T wave. [4]

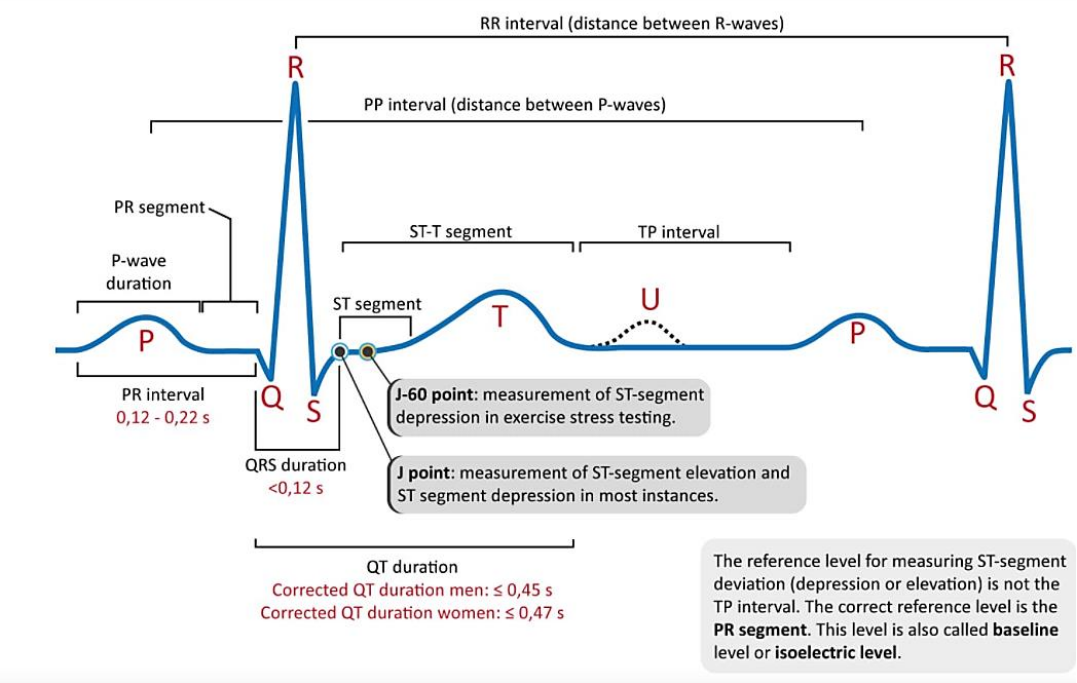


Figure 1.1: The classical ECG curve with waveforms. [5]

Any imbalance or distortion in the timing or pattern of the beat is an indication of heart disease caused by the ill-posed electrical activity of the heart, which can be viewed from ECG. Normal 12 lead ECG recording uses six chest leads (V_1 to V_6), three bipolar standard leads (L_I, L_{II}, L_{III}) and three unipolar limb leads (aVR, aVL, aVF), which collectively work to gather & plot different view parts of the heart's electrical activity. [4]

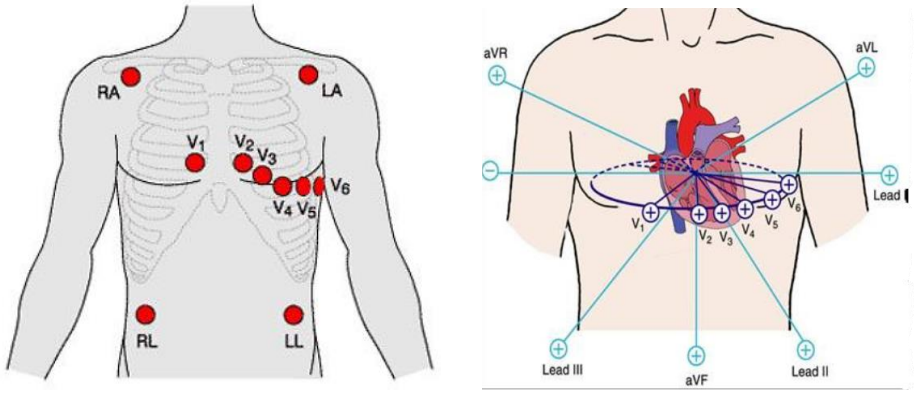


Figure 1.2: 12 lead ECG placement & view parts. [6]

Table 1.1: How different diseases are predicted based on ECG wave pattern [7]

Heartbeat segment	**Nominal characteristic**	**Underlying diseases if nominal property violated**
P wave	Upright except aVR lead Biphasic in L_{II} & V_1 lead P axis is between $0°$ and $75°$	An increase in amplitude occurs for Atrial Enlargement (AE). Right AE: P wave $> 2mm$ in L_{II}, L_{III}, aVF Left AE: Broad & double-peaked P wave
PR interval	100 to 200 milliseconds	Interval time extension → First-degree atrioventricular block
QRS complex	Q wave $<$ 50 milliseconds in all leads, excluding V_1 & V_3	Abnormality indicates a past or current infarction
	Imperfect R wave size or height	R wave height larger → Ventricular hypertrophy
	$90°$ to $-30°$ QRS axis	Axis of $-30°$ to $-90°$ is recognized as Left axis deviation and causes inferior myocardial infarction. Axis of $90°$ to $180°$ is recognized as Right axis deviation and causes right ventricular hypertrophy.
	QRS interval of 70 to 100 milliseconds	Interval $\geq$ 120 milliseconds → Intraventricular conduction delay or Complete bundle branch block. 100 to 110 milliseconds interval → Unspecific intraventricular conduction delay or Incomplete bundle branch block.
QT segment	-	Prolongation → Torsades de pointe ventricular tachycardia
ST-segment	Horizontal with PR (or TP) interval baseline	Elevation → Myocardial ischemia and infarction, Left ventricular hypertrophy, Left ventricular aneurysm, Early repolarization, Pulmonary embolism, Pericarditis, Hypothermia, Hyperkalemia Depression → Subendocardial ischemia, Hypokalemia, Reciprocal changes in acute myocardial infarction, Digoxin

T wave	Smooth, rounded & takes the same direction as QRS complex	Low amplitude → Hypokalemia, Hypomagnesemia Tall & peaked → Left ventricular hypertrophy, Hyper/Hypocalcemia.
U wave	Sometimes present in a healthy person	Commonly found in patients having Hypokalemia, Hypomagnesemia, or Ischemia

It's too cumbersome for a human to memorize properties tabulated above & look for the appropriate pattern to predict the correct disease. Moreover, age, gender, sex, ethnicity, locality, etc., impose a vital role in contributing to the different features of heart diseases.

Machine learning can help solve this enormous problem reducing the burden to physicians & increasing their diagnostic accuracy. Different machine learning & deep learning algorithms can be used to teach the computer a vast dataset to learn the correct ECG pattern corresponding to various diseases; the computer will use this knowledge to diagnose new ECG data of a patient further. Here comes the significance of the research related to Machine Learning for the multiclass classification of different ECG waveforms.

1.3 Objectives of this Research

One of the most publicly available ECG datasets is MIT-BIH arrhythmia database which is being used for decades to develop algorithms for automated arrhythmia detection which can read & analyze diversified features and can classify a right group of ECG wave.

MIT-BIH arrhythmia dataset comprising 87554 instances with 188 attributes consists of 5 categories of ECG data: Normal heartbeats (N), Supraventricular ectopic beats (S), Ventricular ectopic beats (V), Fusion beats (F), Unclassified beats (Q).

The objective & scope of this research work is to select the appropriate set of parameters, to choose sensitive & highly correlated features, and to find out efficient ensemble algorithm to perform multiclass classification of the arrhythmia dataset aiming for the development of technologies that would early detect cardiovascular diseases giving the scope for prevention of the disease.

1.4 Main Contribution

We've analyzed the arrhythmia dataset with six machine learning algorithms (KNN, RFC, SVM, SGD, AdB, XGB) & 5 deep learning algorithms (CNN, RNN-LSTM, Deep CNN, VAE, DBN). We've performed hyperparameter tuning for each of the algorithms in order to select the right set of hyperparameters giving the best accuracy, portrayed Confusion Matrices, calculated individual & overall Performance matrices (Precision, Recall, Accuracy, F-1 score) for finding out the best algorithm among these. Later we've proposed an HDNN algorithm that offers the **highest accuracy of 99.23%** among the recent works performed for multiclass classification of this dataset. The proposed HDNN algorithm is an ensemble of Deep CNN (11 layers), VAE (7 layers), DBN (64 layers).

1.5 Thesis Outline

In chapter – 1, the significance of research on cardiovascular disease classification from ECG data is explained. The necessity of ML & DL algorithms for increasing the accuracy of diagnostics & the main contribution of our research work is also described.

In chapter – 2, relevant research works published recently on ECG data classification using ML & DL algorithms are analyzed comparatively that in-line with our research topic of interest.

In chapter – 3, the central architecture & methodology of our entire work, performance matrices that we calculated are narrated.

In chapter – 4, the Data preprocessing & Feature engineering process that we've performed are visually depicted, which is an integral part of any ML/DL research.

In chapter – 5, the 6-Machine learning & 5 deep learning algorithms & their working procedures are briefly described that we've implemented & optimized to compare the performance matrices with our proposed HDNN algorithm.

Chapter – 6 describes the confusion & performance matrices of all the 11 algorithms & compares the overall result & performance with our proposed HDNN algorithm, which portrays 99.23% accuracy in the classification of the MIT-BIH Arrhythmia ECG dataset.

Finally, the discussion of our research work is concluded in chapter – 7 by mining out the Future-scopes to improve & better implement our study of interest.

Chapter 2

LITERATURE REVIEW

2.1 Relevant Research

X. Zhang et al. extracted the ECG is featured by transform wavelet and builds a vector data set of the key characteristics of the waveform features for each heartbeat and the RR interval [8]. For the identification of Arrhythmia through training and research, the ELM model is used. All ECG function parameters have to be correctly acquired for proper classification. Here QRS complex wavelet transform detection algorithm is used. The model is checked, and the average accuracy of 94.4% is finally obtained using the 10-fold cross-validation process. The result shows, this process is highly accurate and generalized, but the true positive rate for the F and S classes is not high enough because the number of samples is too limited.

Kachuee et al. presented a method for the classification of heartbeat through ECG based on a transferable representation [9]. It has a deep convolutional neural network with a residual connection for Arrhythmia classification, which can be used as a base for training the classifiers for the classification of MI accurately. The accuracy of this method is 95.9%. Here the disadvantage is sample labels are not used in the visualization of the learned representation.

A. Rana and K. K. Kim used the single-layer LSTM model of the time series ECG to classify ECG signals [10]. Three kinds of gates are used here. From the MIT-BIH dataset, ECG data is collected, and then the input is sent to the LSTM model.100 hidden units with time step 10 and batch size 50 are used here. It has an accuracy of 95% with 200 epochs. It has the disadvantage is that the converging starts after 125 epochs.

Pu Wang et al. augmented data by Auxiliary Classifier Generative Adversarial Network (ACGAN) and implemented the algorithm for classifying ECG signal using eight stacked residual blocks connected parallel with 1-layer LSTM [3]. They reported an F-1 score of 0.883 for the performance of their proposed model.

Fajr Ibrahem Alarsan and Mamoon Younes performed machine learning approach for the classification of ECG signals [11]. They used Apache Spark's scalable machine learning library

for the simulation, implemented Gradient Boosting Trees (GBT) and Random Forest Classifier (RFC). In multiclass ECG classification, their proposed Random Forest Classifier with ten trees and max depth of 25 gave 98.03% of accuracy.

S. Chakroborty and M. A. Patil proposed a classification paradigm of coarse-to-fine for Arrhythmia to calculate an extensive database of real-time classification, which seeks to address the computer burden problem without sacrificing the accuracy of the classification [12]. The MIT-BIH Arrhythmia database experimentation has conducted five separated model groups with a maximum speed-up factor of 2.2:1 and with minimal classification accuracy deterioration. A decimation-based approach has been used to minimize the beat length, and the use of the MSVQ method reduces the number of beats. The work also integrates these two approaches and demonstrates a 2.2:1 time-complexity reduction relative to traditional public database classification methods. In small databases, such as MIT-BIH, the speed-up factor might not be very promising, but in a more extensive database, classes and beats are massive, the speed-up factor would dramatically increase. Although this is more effective for large datasets and not wise to use for small datasets, it showed an average accuracy of 98.1321 for each class.

Rekh Ram Janghel and Saroj Kumar Pandey used machine learning techniques for the classification of ECG signals [13]. They implemented Support Vector Machine (SVM) with linear and RBF kernel, Decision Tree (DT), Naïve Bayes (NB), K-Nearest Neighbour (KNN), AdaBoost (AdB), and Random Forest (RF). The proposed Decision Tree (DT) classifier for the detection of ECG signal having an accuracy of 93.4%

R. Banerjee et al. used a hybrid CNN-LSTM structure with a combination of the hand-crafted features to bring both CAD markers in a single frame of architecture for the classification of disease [14]. It has been applied to the datasets of two hospitals, and it was successful, which proves its efficiency. A low-cost sensor was used in one of them. But both the biomarkers that are considered in this paper fail to guarantee the onset of CAD. It also shows the failed approach in the detection of a few borderline patients. However, it showed sensitivity, specificity values of 0.94 & 0.92 respectively for test set-1 and 0.90 and 0.85 for test set-2.

2.2 Comparative Analysis of Relevant Research

Table 2.1: Comparative tabulation for a better understanding of previous relevant research

Work	**Model Used**	**Dataset Used**	**Measures Reported**	**Best Performance Reported**
[8]	Extreme Learning Machine	MIT-BIH	Accuracy	94.4%
[9]	Deep-CNN	MIT-BIH	Accuracy	95.9%
[10]	LSTM	MIT-BIH	Accuracy	95%
[3]	1-layer LSTM with 8 residual blocks	MIT-BIH	F-1 Score	0.889
[11]	Random Forest	MIT-BIH	Accuracy	98.03%
[12]	Coarse-to-fine classification techniques	MIT-BIH	Accuracy	98.1321%
[13]	Decision Tree	MIT-BIH	Accuracy	93.4%
[14]	CNN-LSTM	MIT-BIH	Sensitivity Specificity	0.94 & 0.90 (TS-1&2) 0.92 & 0.85 (TS-1&2)

Chapter 3

METHODOLOGY

3.1 Basic Methodology

The basic working flow of our research is depicted by a flowchart below:

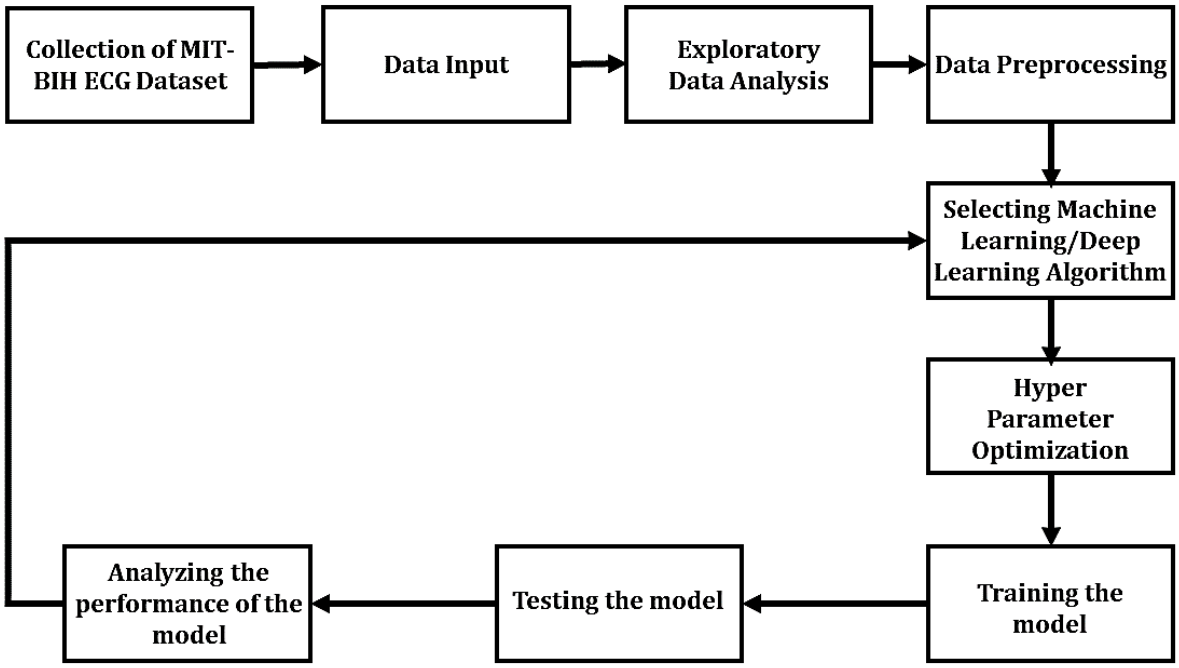


Figure 3.1: Basic Methodology Flowchart

3.2 Description of the Basic Methodology

Firstly MIT-BIH ECG dataset was collected. The description of the dataset will be discussed in the next chapter. Then the dataset was given input to the programming platform "Jypyter Notebook" of Anaconda Navigator, which is a very famous platform for Machine Learning and Deep Learning [15-16]. Exploratory Data Analysis was performed for the visualization from the dataset. Data preprocessing and feature extraction were done to make the dataset perfect for the Machine learning/Deep Learning models. Then, Data were split into train and test sets. After that, a particular ML/DL algorithm was selected for the implementation of the model. Hyperparameter optimization was done to make the models more efficient and compact. Then the model was trained with the training dataset, and later on, they were tested with the test dataset. Respective confusion matrices were portrayed. Finally, the performance of each model was analyzed using various performance matrices like accuracy, precision, recall & F-1 score.

3.3 Description of Performance Matrices in ML/DL

Performance Matrices of ML and DL come from the confusion matrices. Confusion matrices give the visualization of the performance of the ML/DL models. Each row of the matrix presents the instances of the actual class, where each column represents the instances of the predicted class or vice-versa [17]. One example of the confusion matrix is mentioned below:

Table 3.1: Confusion Matrices in ML/DL

Confusion Matrix		**Predicted**	
		True	**False**
Actual	**True**	True Positive (TP)	False Negative (FN)
	False	False Positive (FP)	True Negative (TN)

Here, we can see four terms like True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). Their description is given below:

a) **True Positive (TP):** A class was predicted Positive, which is True actually.
b) **True Negative (TN):** A class was predicted Negative, which is True actually.
c) **False Positive (FP):** A class was predicted Positive, which is False actually.
d) **False Negative (FN):** A class was predicted Negative, which is False actually.

3.4 Formula of Performance Matrices

Formulae that were used to measure the performance matrices of ML and DL models in our research are mentioned below [18]:

a) $$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

b) $$Precision = \frac{TP}{TP + FP}$$

c) $$Recall = \frac{TP}{TP + FN}$$

d) $$F1 - Score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

3.5 Detailed Methodology of our Research

The detailed working flow of our research is illustrated below:

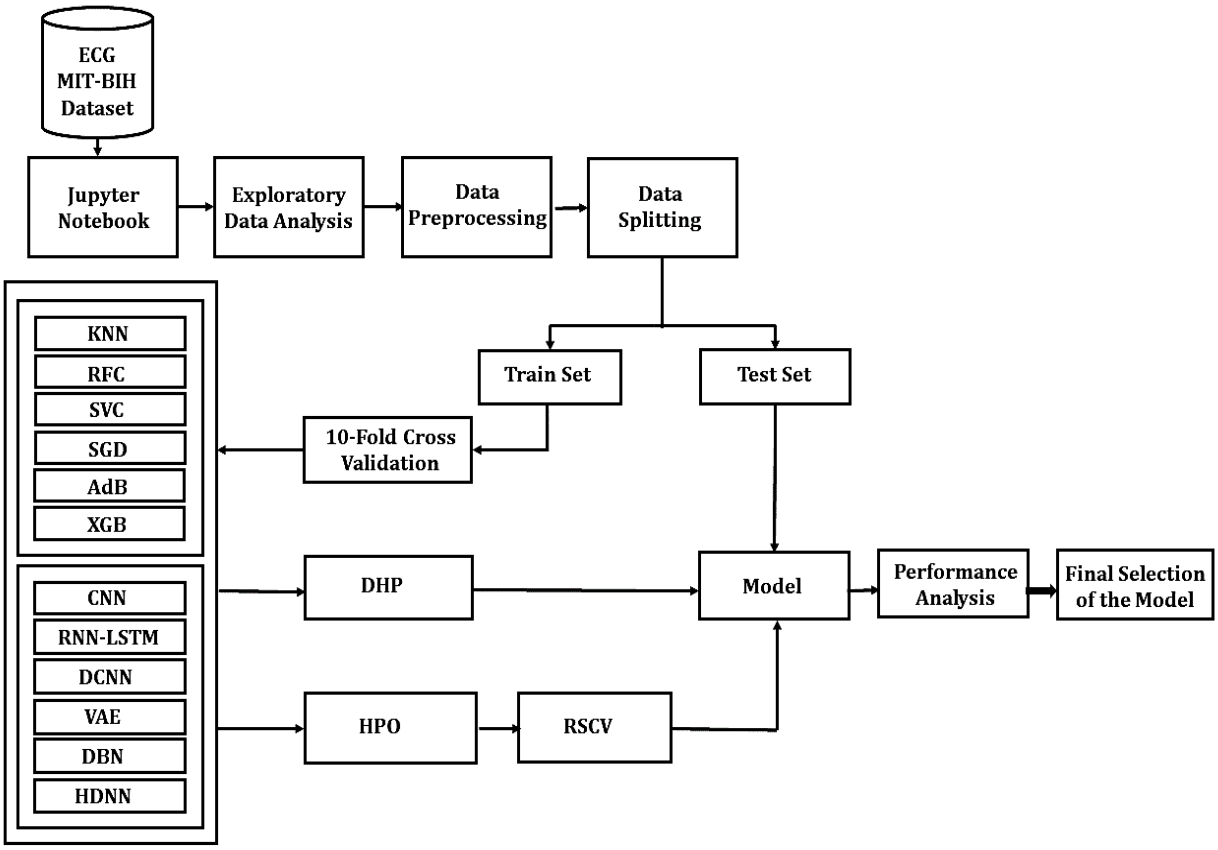


Figure 3.2: Detailed Working Flow of our Research

Six Machine Learning Algorithms (KNN, RFC, SVC, SGD, AdB, XGB) and five Deep Learning Algorithms (CNN, RNN-LSTM, DCNN, VAE, DBN) were implemented with Default Hyperparameter (DHP) and later on with Hyperparameter Optimization (HPO) in our research. In the end, our proposed Hybrid Deep Neural Network (HDNN) was implemented. Their description and mathematical background will be discussed in chapter five.

Chapter 4

DATA PREPROCESSING

4.1 Importance of Feature Engineering

The most informative property or attribute related to any dataset to classify different dataset patterns is known as features.

Feature Engineering is an integral part of any machine learning and deep learning model. A good feature engineering technique can increase the efficiency & overall performance of the model effectively [19].

The raw data collected from patient history & diagnostics are processed to transform them into useful features that are compatible with algorithm requirements & better represent the underlying & root cause of the disease to the predictive machine learning & deep learning models. The procedure is known as Feature Engineering. Feature engineering converts the data inputs into the most valuable assets that the classifier algorithm can understand & work on.

Different feature engineering techniques are [20]:

a) Data visualization
b) Finding missing values & Data imputation
c) Outlier mining & Handling outliers
d) Histogram Plotting & Data Binning
e) Data resampling
f) Plotting Correlation Heatmap
g) Dataset splitting

4.2 Dataset Description

In our research, we have used MIT-BIH Arrhythmia Dataset from The PhysioNet repository, which is a popular research resource for complex physiological signals [21]. The dataset contains 87554 instances with 188 attributes. The sampling frequency is 125 Hz, and there are five target categories. A short description of the categories is given below [22]:

Table 4.1: ECG data categories of MIT-BIH Arrhythmia dataset

Category	Annotation
N (Normal Beats)	• Normal • Left/Right bundle branch block • Atrial escape • Nodal escape
S (Supraventricular ectopic beats)	• Atrial premature • Aberrant atrial premature • Nodal premature • Supra-ventricular premature
V (Ventricular ectopic beats)	• Premature ventricular contraction • Ventricular escape
F (Fusion beats)	• Fusion of ventricular and normal
Q (Unclassified beats)	• Paced • Fusion of paced and normal • Unclassifiable

4.3 Feature Engineering

4.3.1 Data Visualization

Exploratory Data visualizations of the dataset are pivotal for realizing the pattern among the features, especially whenever the dataset is high dimensional [23].

After taking input the data into the programming platform, several data visualization were performed.

4.3.1.1 Data Distribution Plotting

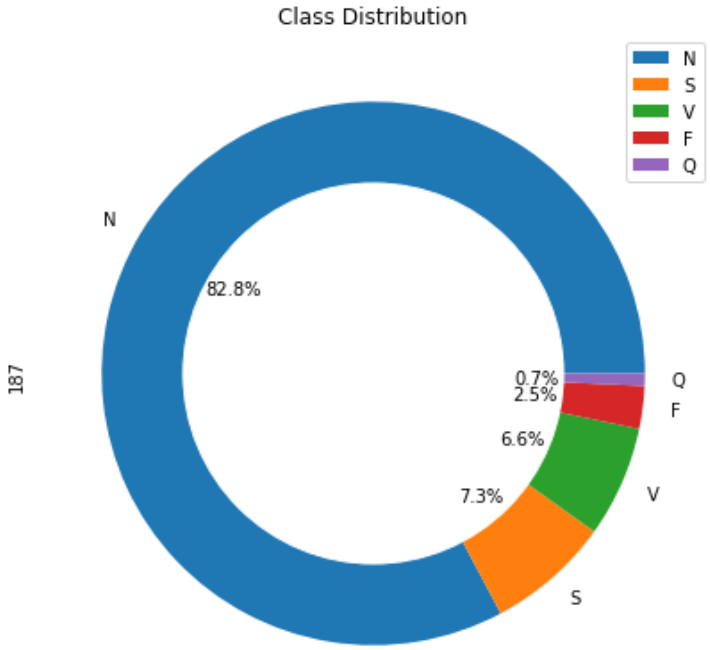


Figure 4.1: Data Distribution

From the data distribution, it is seen that the classes are unevenly distributed like 82.8% of the class is N, 7.3% is of S, 6.6% is of V, 2.5% is of F, and 0.7% is of Q category.

4.3.1.2 ECG Signal plotting from the dataset:

Five categories of ECG signal were normalized to one and plotted up to 1750 ms:

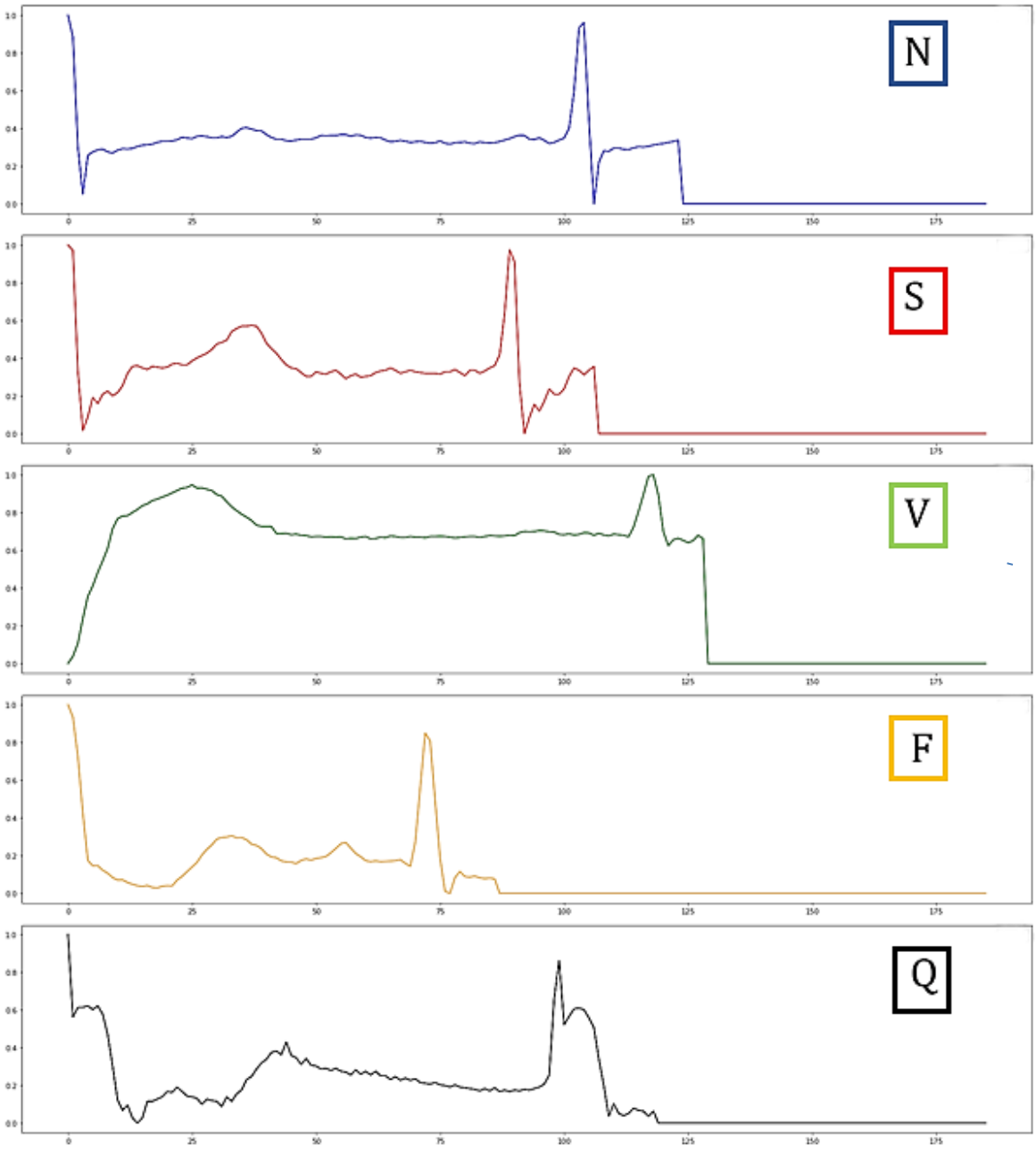


Figure 4.2: Five categories of ECG Signal

4.3.2 Finding missing values & outliers:

Some data might be found missing in the dataset due to technical interruptions during data flow, human error due to lack of consciousness or proper training & deleted data for maintaining patient data privacy.

For data size preservation, to keep the performance of the model unaffected, most occurred value may be imputed in place of missing value, or it may be valued as NaN.

The data due to measurement or execution error that deviates significantly from the rest of the dataset such that it's reasonable to be discarded are called outliers. Most of the machine learning & deep learning algorithms are very sensitive to outliers. Therefore, data visualization through the graph in terms of Standard Deviation or in terms of Percentiles gives a better opportunity to detect & take the decision for handling the outliers. [20]

Our MIT-BIH ECG Arrhythmia Dataset was checked thoroughly to find any missing values & outliers since missing values & outliers may create problems while training the model [24].

There was no missing value & outliers in our dataset of interest.

4.3.3 Histogram Plotting & Data Binning:

Even though this process costs the performance a little bit by making data a bit regularized sacrificing information, but dataset binning gives much more advantage by preventing overfitting & making the model robust.[20]

The color-coded 2D histogram is used to differentiate two categories of ECG data from a multi-categorized dataset where the intensity range of the image is divided into bins of color codes varying from green to red. The green color represents low counts of the target data group while the red represents high counts of the target data group. [25]

From these, we can clearly distinguish the heartbeat pattern of the target group we want to visualize from the other groups.

N-type:

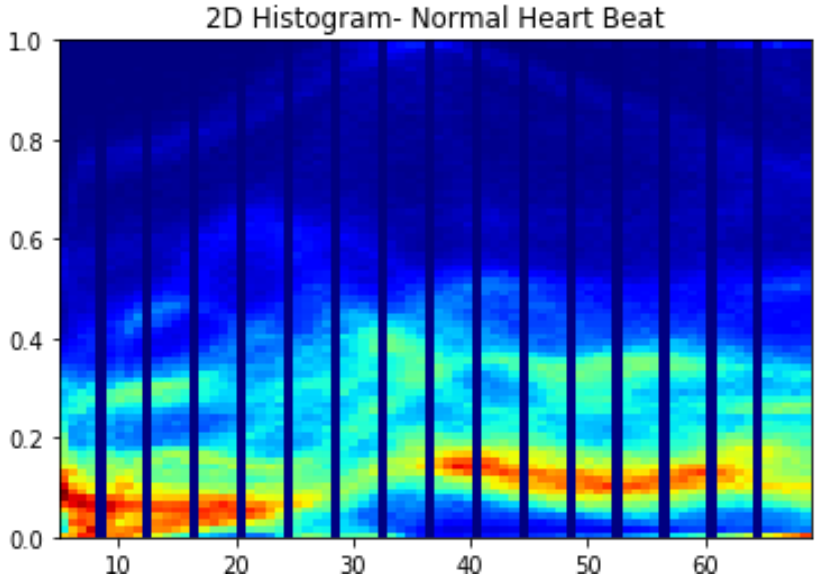


Figure 4.3: Normal Heartbeat

S-type:

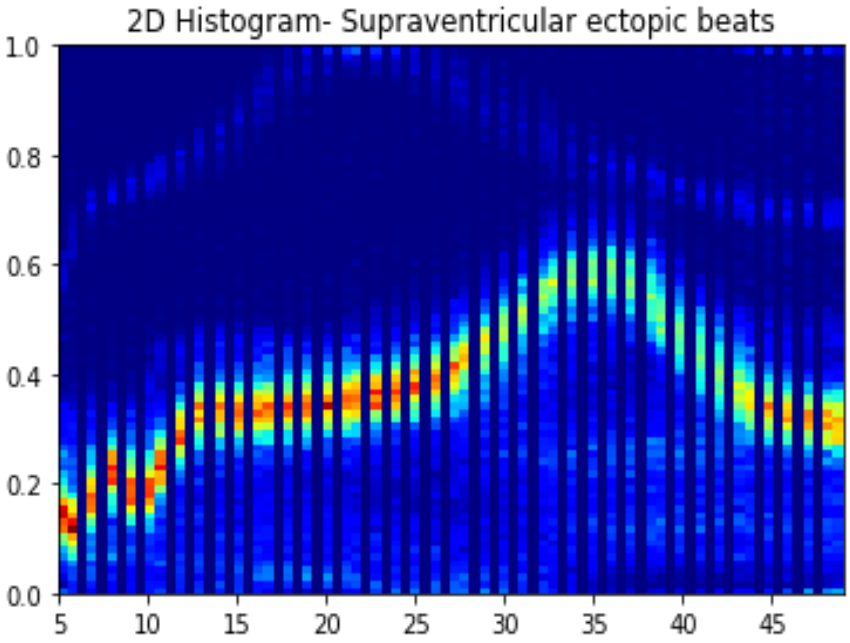


Figure 4.4: Supraventricular Ectopic Beats

V-type:

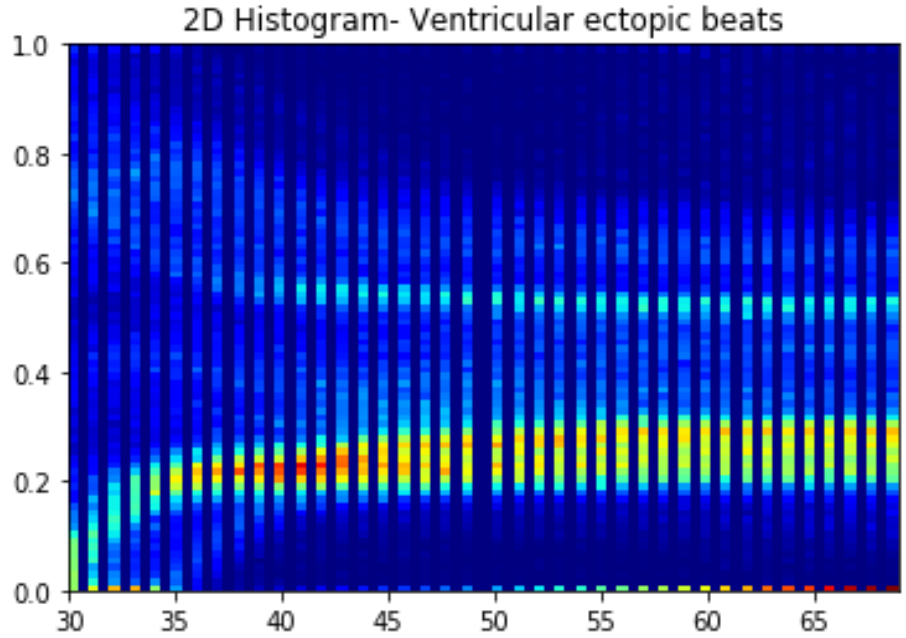


Figure 4.5: Ventricular Ectopic Beats

F-type:

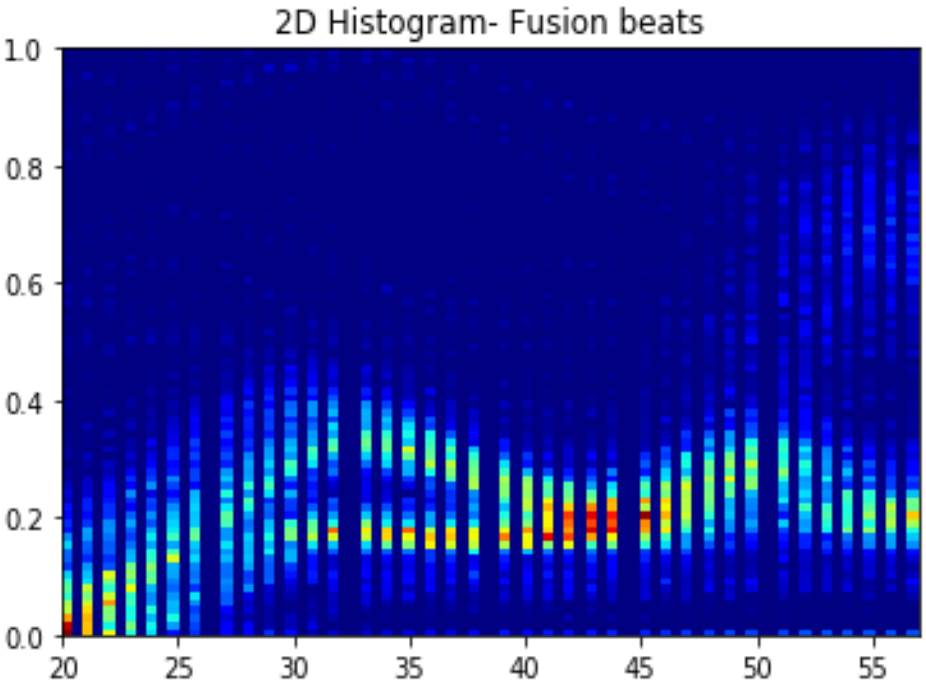


Figure 4.6: Fusion Beats

Q-type:

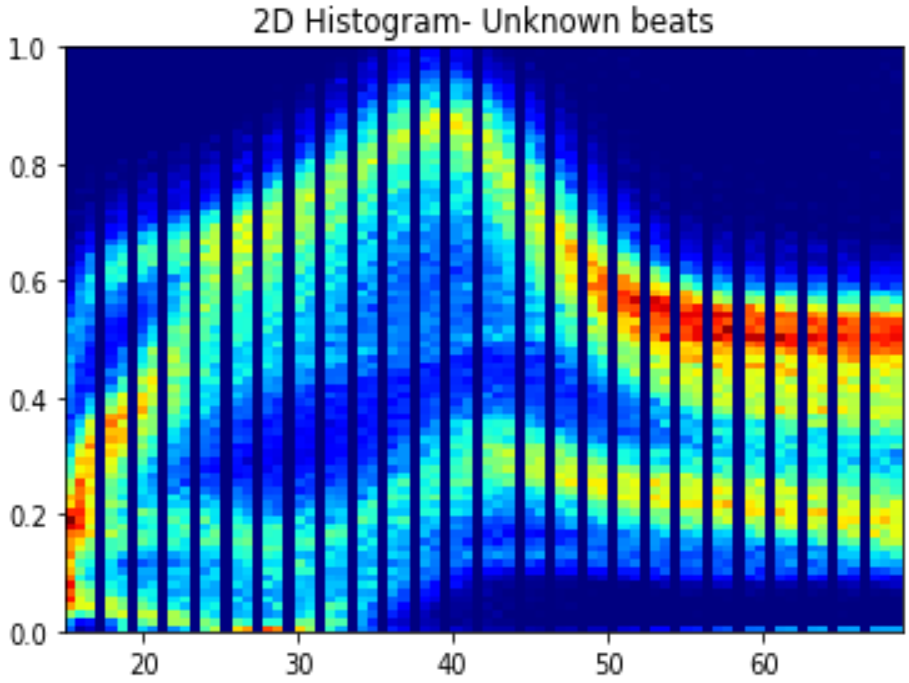


Figure 4.7: Unknown Beats

4.3.4 Data Resampling:

From the data distribution (Figure. 4.1), it is clearly seen that the classes are unevenly distributed. So, if we train the model on this Data without preprocessing, the model may get biased [26]. Resampling was done to each category of data so that the whole Data becomes well balanced. After resampling, the data distribution becomes like the following figure:

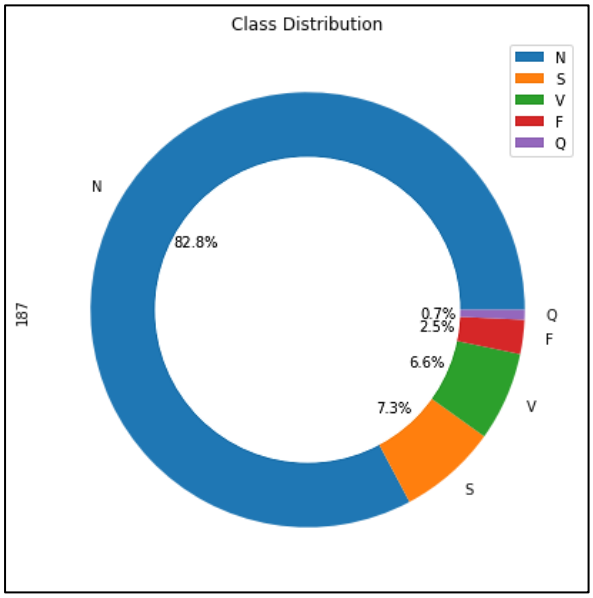


Figure 4.8: Class Distribution before Resampling

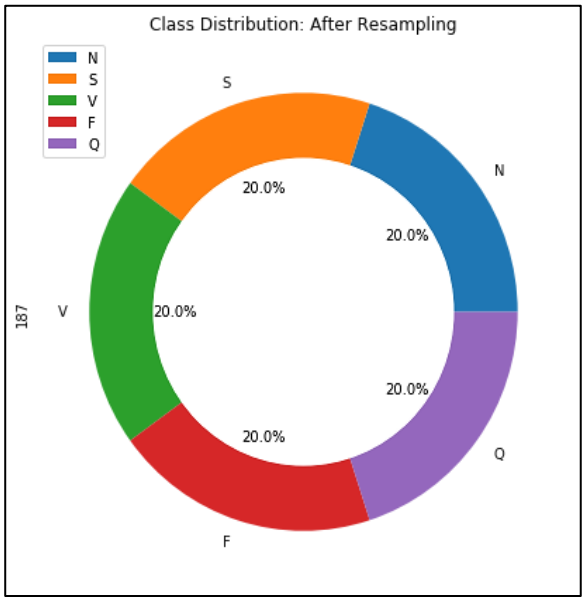


Figure 4.9: Class Distribution after Resampling

4.3.5 Plotting Correlation Heatmap:

To portray the same dataset in a visually appealing way, it's an essential step in exploratory data analysis, which graphically shows correlating variables, degree & direction of correlation and notify us about multicollinearity problems.

The degree of correlation between variables varies from -1 to +1. Correlating values near to -1 defines the correlation between two variables as more negative (as one value increases, the other value decreases); the closer to -1, the stronger the relationship is. Correlating values near 0 define no linear trend between the two variables. Correlating values near 1 represent the positive linear trend between the two variables.

A correlation heatmap was plotted to view highly correlated dataset features with the target output variable.

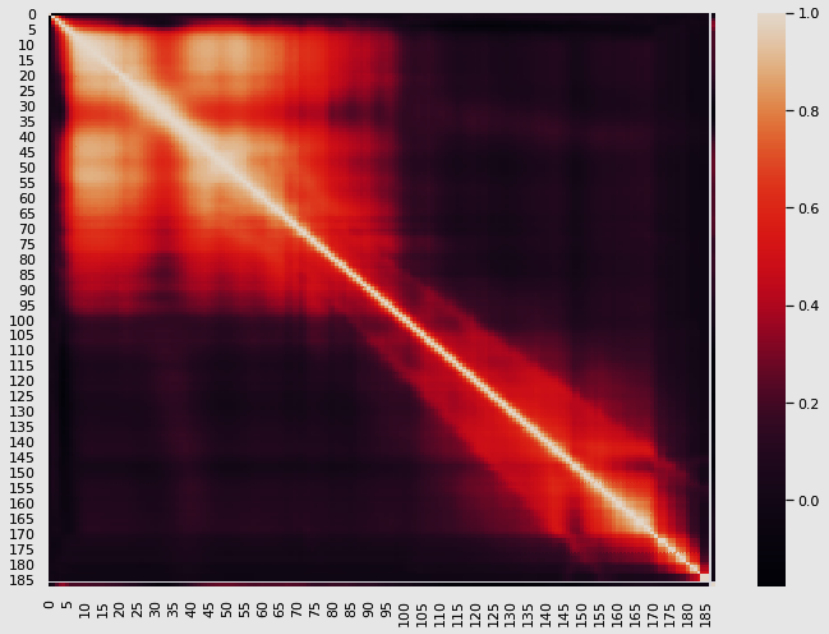


Figure 4.10: Correlation Heatmap

It was seen that from five to ninety-five and from one fifty to one seventy-five, the features were highly correlated with the target output variables.

4.3.6 Dataset Splitting:

Data (87554 instances) were randomly shuffled & then split into 80:20 ratio for training (70043 instances) and testing (17511 instances), respectively, as this ratio shows the least bias and variance for the machine learning and deep learning algorithms we implemented.

Chapter 5

INTRODUCTION TO ALGORITHMS

5.1 K-Nearest Neighbors [27,28]

K-nearest neighbors (KNN) is used for both regression & classification problems, but it's termed as typically simple, nonparametric & lazy supervised algorithm.

It has the simplest working procedure based on distance measurement:

Step 1 – At first, the training & test dataset is being loaded.

Step 2 – Then nearest 'k' (any integer) number of data points are chosen.

Step 3 – Next, the following procedures are performed on every data point in the test dataset –

a − Distance between each row of training data point and test data point is being measured.

b – The test data points are sorted in ascending order based on the measured distance.

c – Then, from the sorted array, upper K rows are being chosen in order to assign a class to the test data point depending on the most vibrant class of those sorted rows.

Step 4 – End of the algorithm

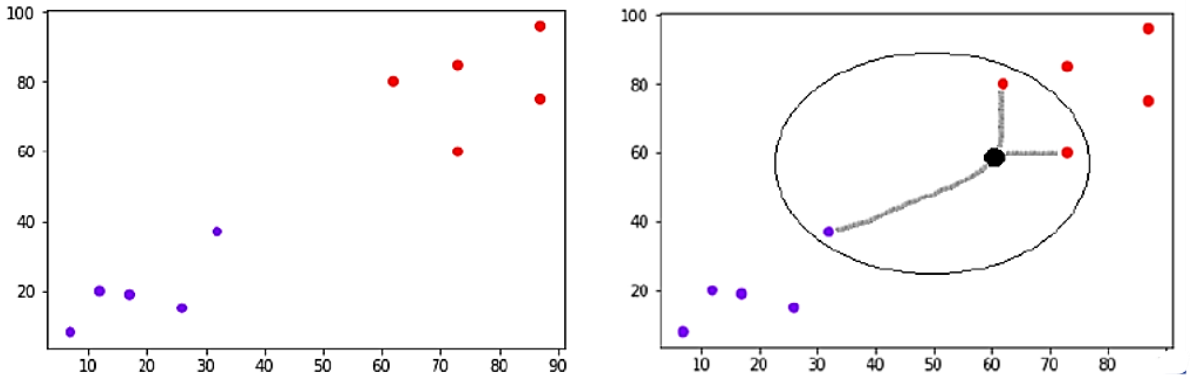


Figure 5.1: KNN Architecture [28]

Distance functions:

i. Euclidean distance:

$$\sqrt{\sum_{j=1}^{k}\left(p_j - q_j\right)^2}$$

ii. Manhattan distance:

$$\sum_{j=1}^{k}\left|p_j - q_j\right|$$

iii. Minkowski distance:

$$\left(\sum_{j=1}^{k}\left(\left|p_j - q_j\right|\right)^n\right)^{\frac{1}{n}}$$

iv. Hamming distance:

$$D_H = \sum_{j=1}^{k}\left|p_j - q_j\right|$$

$$p = q \Rightarrow D = 0$$

$$p \neq q \Rightarrow D = 1$$

e.g.:

p	q	Distance
Vanilla	Vanilla	0
Vanilla	Chocolate	1

5.2 Random Forest Classifier [29]

Introduction: Branched-trees are the indistinguishable property of a forest; the more trees, the more robust the forest is. Random Forest algorithm is another type of supervised learning technique applied in the tasks where Regression or Classification of data are required. Random Forest classifier contains a couple of decision trees composed of different sub-groups of the input dataset, and output from each tree is taken average in order to boost the predictive precision of that dataset. A greater number of decision trees increases accuracy & prevents overfitting.

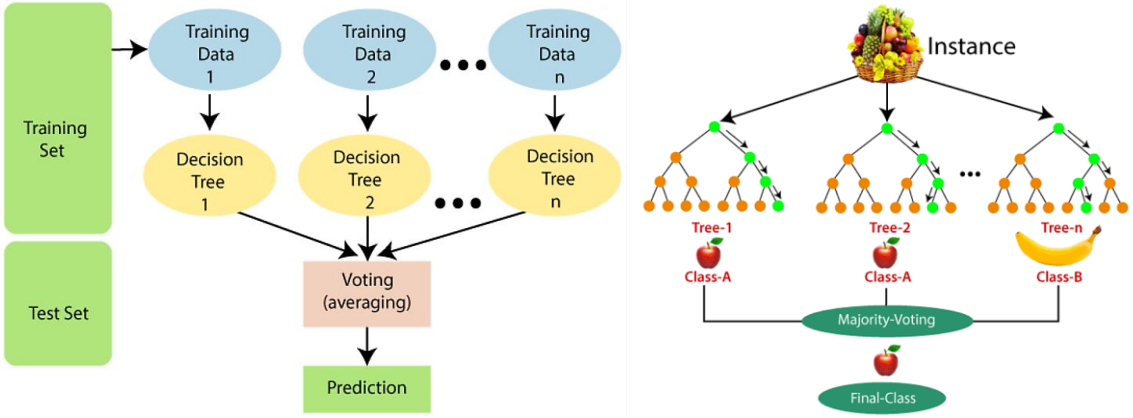


Figure 5.2: Random Forest Classifier Architecture [29]

Working Procedure:

Step-1: Random 'P' data points are being picked from the given training dataset.

Step-2: Using those chosen data, the decision trees are built.

Step-3: Calculate how many ('D' number of) decision trees you want to construct.

Step-4: Loop on Step 1 & 2.

Step-5: Get majority vote by averaging predictions of from all decision the tree for new data points & that should be the final prediction.

5.3 Support Vector Machine [30, 31]

Introduction: SVM is regarded as a few of the foremost prevalent supervised algorithms in order to use in Classification or Regression problems. The algorithm portrays the hyperplane (decision boundary) along the n-dimensional training dataset to form different classes so that test data can be put in the appropriate sub-group in the future. SVM uses the extreme points/vectors called support vectors to draw the hyperplane; hence the name Support Vector Machine.

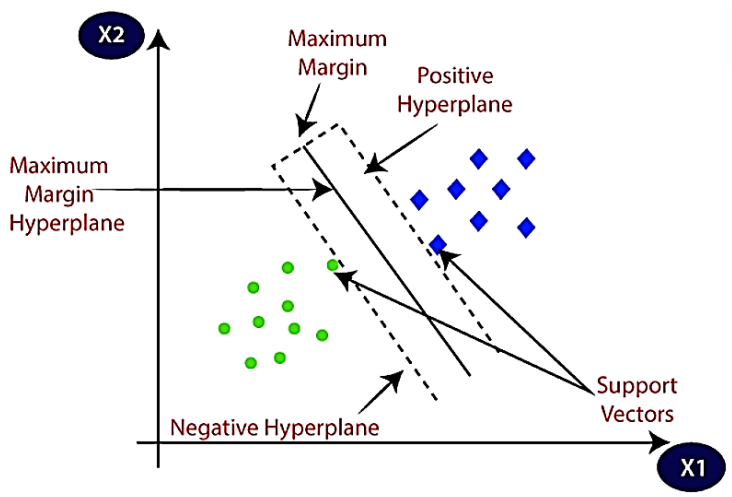


Figure 5.3: Support Vector Machine Architecture [31]

SVM is of 2 types based on hyperplane: Linear SVM & Non-linear SVM

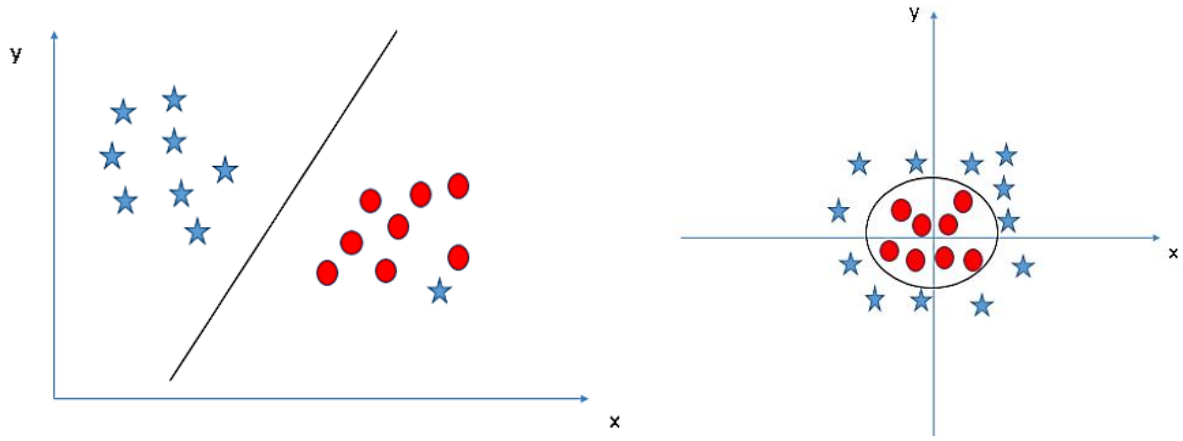


Figure 5.4: Linear & Non-linear SVM architecture [31]

Algorithm:

Hyperplane on a set of points' z' can be equated as,

$$\theta^T z - a = 0$$

Hard Margin: It's implemented on a linearly separable training dataset so as to draw two hyperplanes parallelly at the maximum distance separating two classes of data.

$$\text{Hard margin} = \begin{cases} \theta^T z_i - a \geq 1, & if \;\; y_i = 1 \\ \theta^T z_i - a \leq -1, & if \;\; y_i = -1 \end{cases}$$

Optimization problem: "Minimize $\|\theta\|$ of $y_i\left(\theta^T z_i - a\right) \geq 1$ where, $i = 1, \ldots, n$."

Soft Margin: It's implemented in the case of a non-linearly separable training dataset.

Hinge loss function: $\max\left(0, 1 - y_i\left(\theta^T z_i - a\right)\right)$

The optimization problem is to minimize:

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i\left(\theta^T z_i - a\right)\right)\right] + \lambda\|\theta\|^2$$

5.4 Stochastic Gradient Descent [32]

Introduction: It is an optimization machine learning algorithm. Gradient or function slope is changing rate of a variable with respect to that of another variable. Numerically, the convex function called Gradient Descent calculates the partial derivative of given input parameters. The steepness of the slope depends on the larger value of the gradient. Gradient Descent iteratively works to reduce cost function & finds the optimum cost value utilizing the input dataset & parameters in order to predict the proper output.

Three varieties of Gradient Descent exist (Stochastic, Batch, Mini-batch).

The word 'stochastic' means distribution or pattern that may be examined statistically but may not be predicted precisely. In case of the smaller dataset, batch or mini-batch gradient descent can be useful & accurate where iteration goes through each & every input sample. But in case of a vast dataset, sweeping through every dataset takes enormous computational time, which is impractical. Stochastic Gradient Descent solves this problem where the whole dataset or the batches are randomly shuffled & then a few data are selected from the shuffled dataset as input to the gradient descent algorithm.

Algorithm:

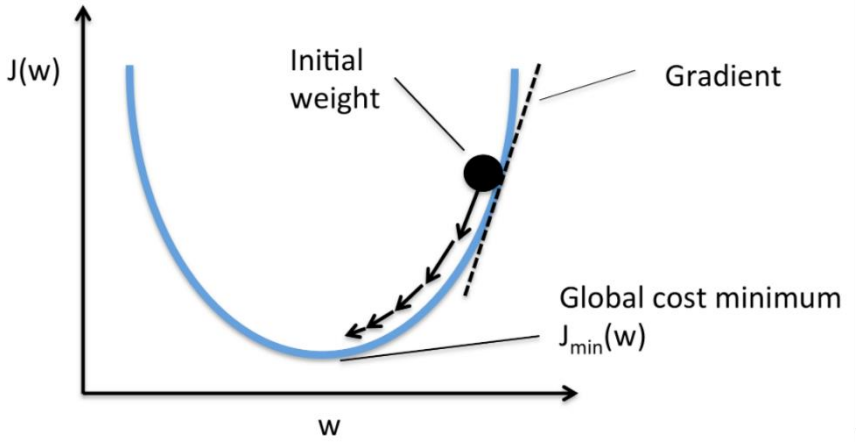
If the objective function is $J(w)$, the number of samples is n, the parameter is w, step size or learning rate is α & regularization term λ; then repeat the following until objective/cost function is minimized:

Minimize the objective function:

$$J(w) = \frac{1}{n}\sum_{i=1}^{n} J_i(w)$$

Update parameter:

$$w := w - \alpha \nabla J(w) + \lambda \Delta w$$



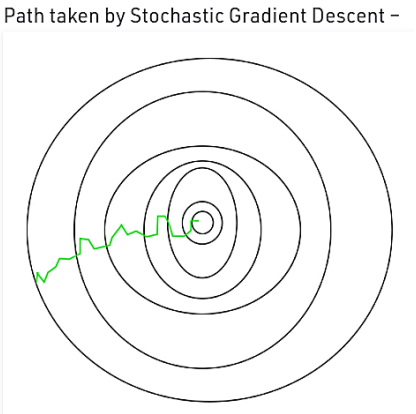


Figure 5.5: Stochastic Gradient Descent Architecture [32]

5.5 AdaBoost [33]

Introduction: From several weak classifiers, an ensemble learning technique called Boosting is used in order to develop a robust classifier that also deals with bias-variance trade-offs. While bagging algorithms work for only high variance in a machine learning model, boosting controls both the bias & variance; thus, boosting works more efficiently. Boosting builds models on resampled data to reduce a model's variance to increase its generalization capability. Moreover, boosting works for both generalization and prediction accuracy.

Few boosting algorithms can be exemplified as AdaBoost, Gradient Boosting, XGBoost, CatBoost, Light GBM

AdaBoost represents Adaptive Boosting which is the first effective & viable boosting algorithm created to work for binary classification. The tweak in this AdaBoost algorithm is, it gradually ensembles the weak classifiers (with decision trees) after each iteration which previously misclassified some data adjusting the weights to focus on misclassified data & eventually, the ensembled weak classifiers work together as strong classifier with higher accuracy. As a result, AdaBoost is sensitive to noisy data and outliers.

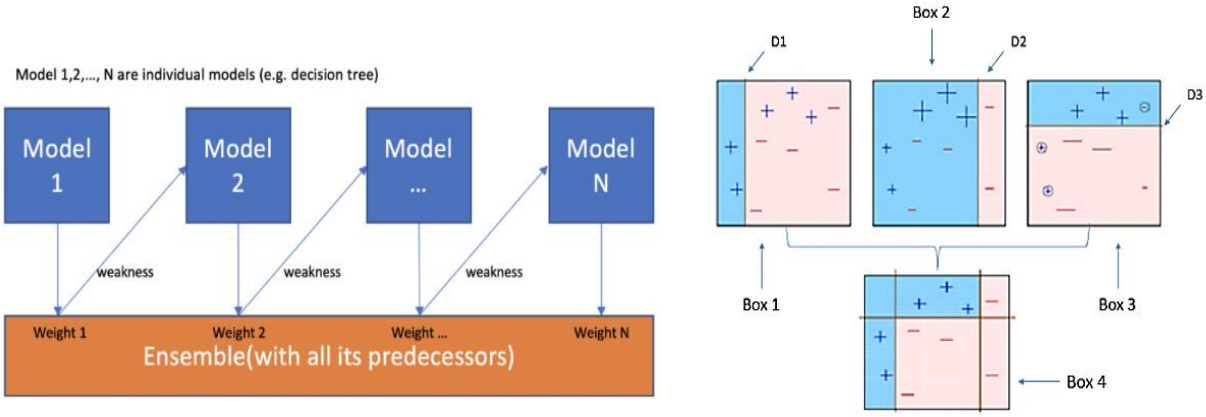


Figure 5.6: AdaBoost Architecture [33]

In the picture above: D1, D2, D3 lines of Box - 1, 2, 3 are weak classifiers trying to separate + & -, but each one does few misclassifications. Together they've become a robust classifier, as stated in Box-4.

Algorithm: At every iteration –

i. Choose a weak classifier k_m.

ii. The classifier k_m minimizes the total weighted error,

$$\sum_{y_i \neq k_m(x_i)} w_i^{(m)}$$

iii. Use minimized total weighted error to calculate the error rate,

$$\epsilon_m = \left(\sum_{y_i \neq k_m(x_i)} w_i^{(m)} \right) \cdot \left(\sum_{i=1}^{N} w_i^{(m)} \right)^{-1}$$

iv. Calculate the weight: $\alpha_m = \frac{1}{2} \ln \left(\frac{1-\epsilon_m}{\epsilon_m} \right)$

v. Improve the boosted classifier from C_{m-1} to C_m:

$$C_m = C_{m-1} + \alpha_m k_m$$

5.6 XGBoost [34, 35, 36]

XGBoost stands for Extreme Gradient Boosting, which uses an optimized distributed gradient boosting (GBM) framework.

Features of XGBoost algorithm:

a) Parallel Computing: This algorithm does parallel processing via using all the cores of a computer processor by default.
b) Regularization: This algorithm performs regularization, which is a technique to avoid overfitting data. It was not available in the GBM framework.
c) Enabled Cross-Validation: XGBoost has an internal CV function, but in many other models, we need to do cross-validation manually.
d) Missing Values: This algorithm can handle missing values internally, even the tendency can be captured by the model.

Model Representation: If M is the number of trees & Q is all possible trees, then,

$$\hat{o}_i = \sum_{m=1}^{M} q_m(x_i) \qquad ; q_m \in \mathcal{Q}$$

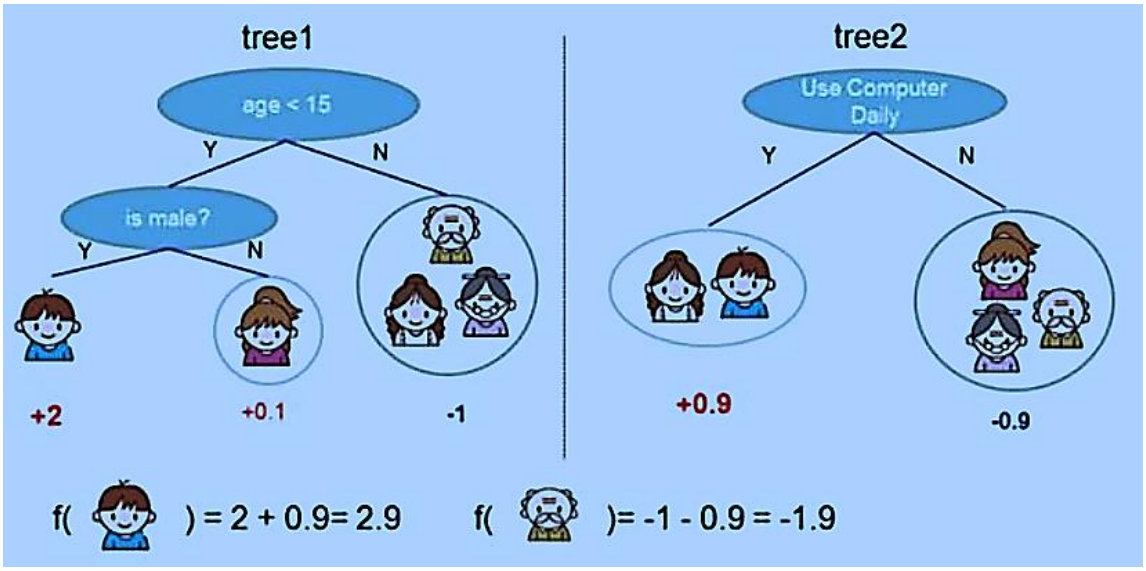


Figure 5.7: XGBoost Architecture [35]

Objective/Loss function:

$$\mathcal{L}(\phi) = \sum_i l(\hat{o}_i, o_i) + \sum_m \Omega(q_m) = \sum_i l(\hat{o}_i, o_i) + \sum_m (\gamma T + \frac{1}{2}\lambda \|\theta\|^2)$$

After regularization & optimization of the above function, the final Objective function becomes:

$$Obj^{(t)} = \sum_{j=1}^{T} \left[\left(\sum_{i \in I_j} u_i \right) \theta_j + \frac{1}{2} \left(\sum_{i \in I_j} v_i + \lambda \right) \theta_j^2 \right] + \gamma T$$

where,

$$u_i = \partial_{\hat{o}_i^{(t-1)l}} \left(o_i, \hat{o}_i^{(t-1)} \right) \qquad \text{and} \qquad v_i = \partial^2_{\hat{o}_i^{(t-1)l}} \left(o_i, \hat{o}_i^{(t-1)} \right)$$

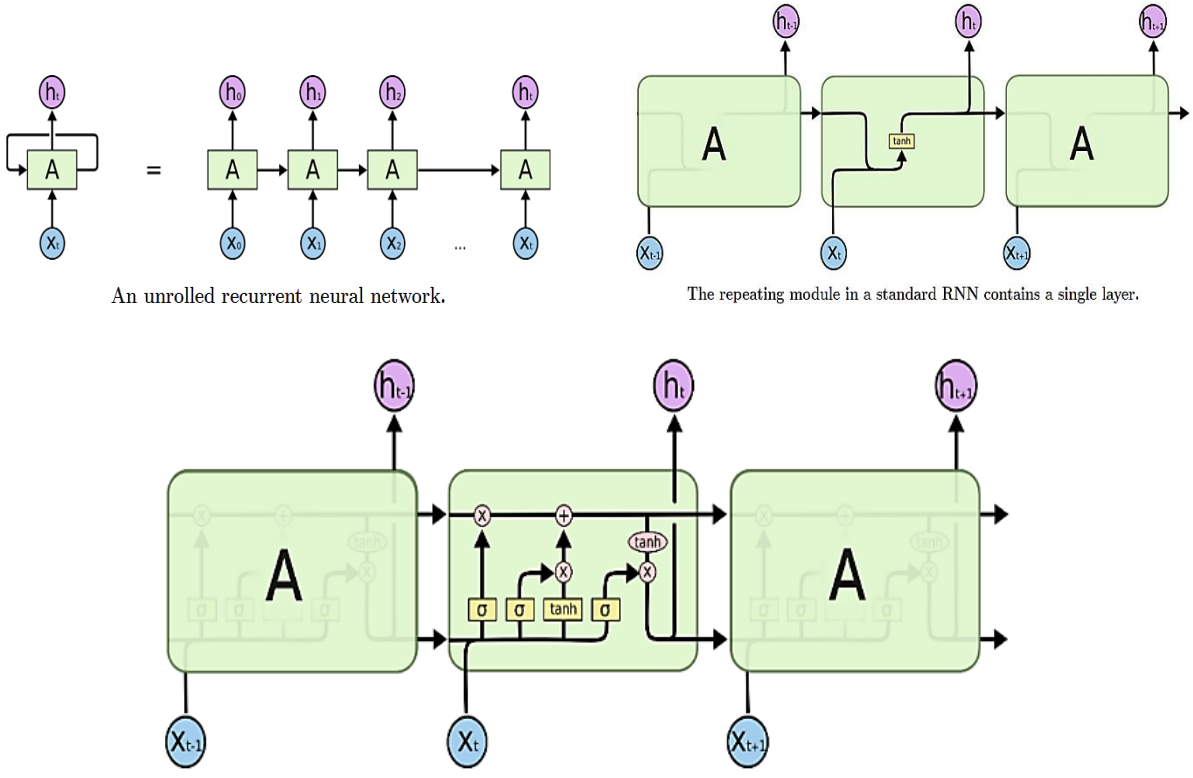
5.7 Recurrent Neural Network (LSTM method) [37, 38]

In a recurrent neural network (RNN), nodal connections form a directed graph along a time-dependent sequence.

RNN has many architectures: Hopfield, Fully recurrent, Echo state, Independently RNN (IndRNN), Second-order RNNs, Bi-directional, Differentiable neural computer, Recursive, Continuous-time, Hierarchical, Elman networks and Jordan networks, Recurrent multilayer perceptron network, Long short-term memory, Gated recurrent unit, Multiple timescales model, Neural history compressor, Neural Turing machines, Neural network pushdown automata, Memristive Networks

We've used the Long short-term memory (LSTM) method in our study.

Introduction: Unlike a typical neural network, RNNs memorize the calculated events in sequential time, make predictions & use the result for calculations in successive time periods (don't start calculation from scratch every second). RNN learns from past data. The specialty of LSTM architecture is, it can remember information for a prolonged time, which resolved the problem of long-term dependent learning & the algorithm can use that information for later predictions. e.g., "I live in Bangladesh & I speak?" – LSTM suggests text as 'Bengali.'



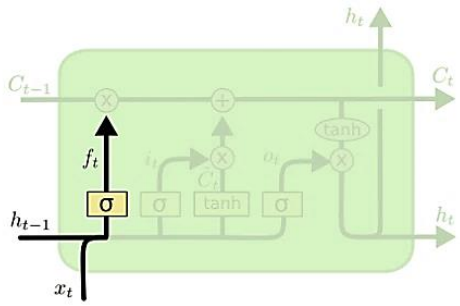
An unrolled recurrent neural network.

The repeating module in a standard RNN contains a single layer.

The repeating module in an LSTM contains four interacting layers.

Figure 5.8: Recurrent Neural Network (LSTM method) Architecture [37]

Architecture: At first, a sigmoid gate decides what information we need to delete from the cell state C_{t-1}. For each number belonging to the cell state, the Sigmoid function calculates a number between 0 and 1. The number 1 tells that the data should exist & 0 tells to delete the data.

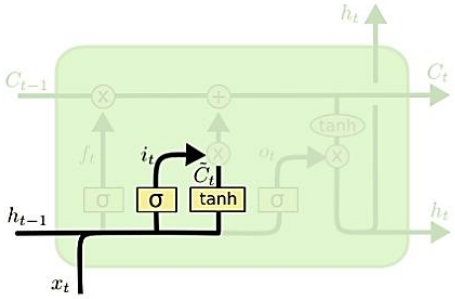


$$f_t = \sigma(b_f + W_f x_t + W_f h_{t-1})$$

The next step decides for the cell state which relevant information should be stacked in it.

(a) The sigmoid layer determines which values we'll update.

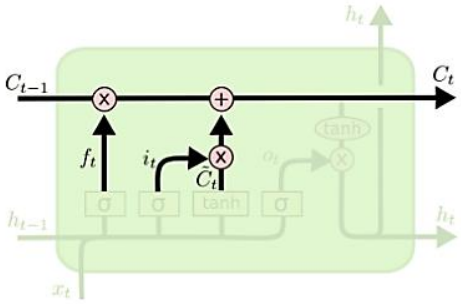
(b) Another layer, called $\tanh()$ produces a vector $\tilde{C}_t$ of new values, and the vector is added to the state.



$$i_t = \sigma\left(b_i + W x_t + W h_{t-1}\right)$$

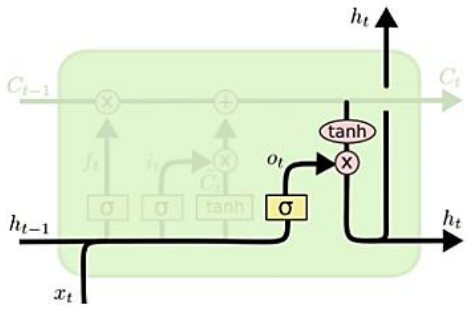
$$\tilde{C}_t = \tanh(b_c + W_c x_t + W_c h_{t-1})$$

In the third step, a new cell state C_t is updated from the previous C_{t-1} state.



$$C_t = i_t \tilde{C}_t + f_t C_{t-1}$$

Finally, the updated version of the cell state is used to calculate output data. Sigmoid & tanh () functions are used like 2nd step & outputs are multiplied so that we only get the parts we decided to.



$$o_t = \sigma(b_o + W_o x_t + W_o h_{t-1})$$

$$h_t = \tanh(C_t) \times o_t$$

We have discussed the simplest form of LSTM architecture; more variations of LSTM architecture are being used in the research & industrial arena.

5.8, 5.9 Convolutional Neural Network & Deep CNN [39]

Introduction: Space/shift-invariant artificial neural networks or convolutional neural network (CNN) is a regularized type of multilayer perceptrons in which neurons of a present layer is connected to each neuron of the previous layer. Data overfitting tendency due to that intense connectivity is resolved using regularization technique using updated weights which minimize the cost/loss function. CNN architectures were modeled after biological processes of human memory cells (neurons).

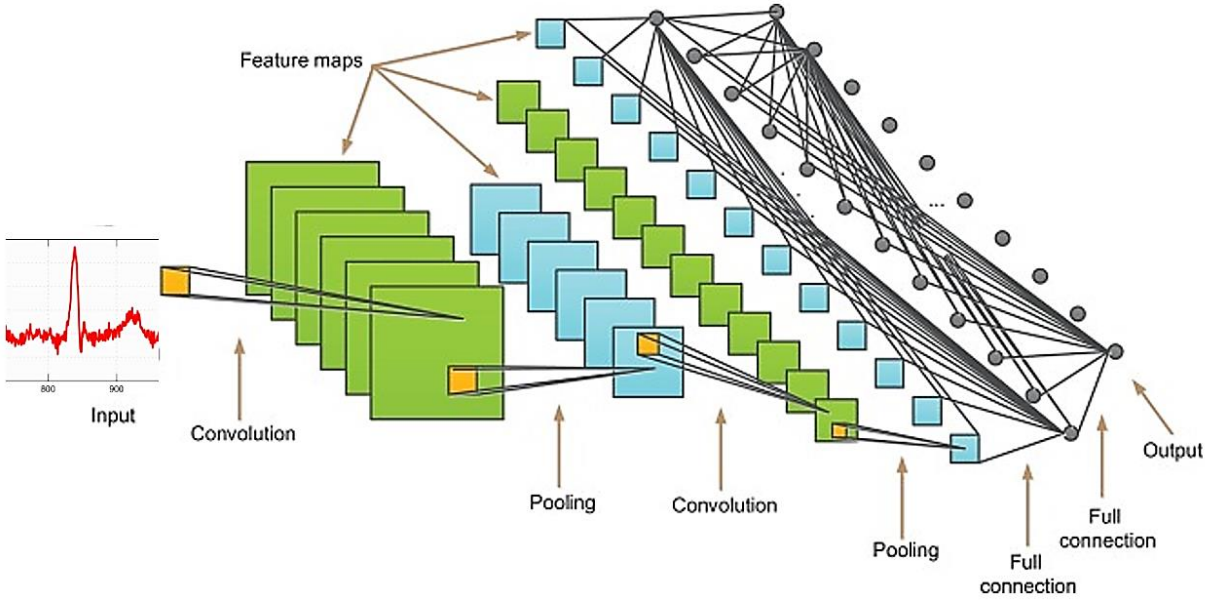


Figure 5.9: Deep CNN Architecture [39]

Architecture:

i. **Convolutional layer:** The core building block of a CNN is referred to as kernels (learnable filters). During forward propagation, discrete convolution occurs in each filter with the zero-padded input.

Though much sophisticated convolutional algorithm is used in practice via TensorFlow, the simplest formula of convolution can be mentioned as:

If original input having 'q' elements is denoted by X to the filter W having 't' filter elements; if the zero-padded input vector is X^P, then the practical convolution formula:

$$Y = X * W \rightarrow Y[j] = \sum_{l=0}^{q-1} X^p[j+q-l]W[l]$$

That's how the convolutional layer gradually learns filtering with a view to detecting particular features in case of test input. This activation stacking method is repeated for all the filters along the whole path of the convolutional layer leading final form of complete output volume of the CNN layer.

Local connectivity: To avoid resource-hungry, intensely connected architecture where each neuron of the present layer is connected to all of the previous layers, in practice, local connectivity is preferred instead while working with a high-dimensional input dataset. Convolutional networks follow a sparse local connectivity pattern where each neuron connects to only a small region of input volume backward & a small region of neuron forward. Such an architecture takes advantage of spatially local correlation.

Spatial agreement: Three hyperparameters (depth, stride and zero-padding) determine the output volume size of the convolutional layer.

The number of neurons (responsible for activating diverse features in the input) in a layer is defined by the depth. For example, if the input of the ECG dataset is taken by the first convolutional layer, then various neurons along the depth may activate when it detects diversified peak amplitudes, spectral frequency.

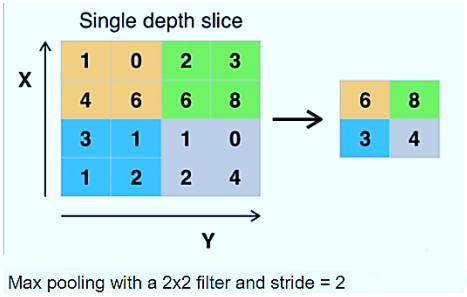
The column depth around height and width is controlled by stride.

Padding defines output volume spatial size.

Parameter sharing: The parameter sharing scheme determines how many parameters are free throughout the convolutional network.

ii. **Pooling layer:** Non-linear down-sampling is referred to as pooling. It can be done by many non-linear functions, but the Max pooling technique is used mostly. This layer works intending to successively shrink the local dimension of the problem set, which in turn lessen the size of memory & parameters that need to be used. This efficient property finally helps to regulate high bias.

$$func_{X,Y}(S) = \max_{a,b=0}^{1} S_{2X+a,2Y+b}$$



iii. **Activation unit:** These units find out the new features at every neuron node.

Sigmoid function: $f(z) = (1 + e^{-z})^{-1}$

Rectified Linear Unit (ReLu): $f(z) = \max(0, z)$

Hyperbolic tangent: $f(z) = |\tanh(z)|$

ReLu unit is mostly used because it shortens the training time of the neural network model, but eventually, accuracy is maintained.

iv. **Fully connected layer:** The layer prior to the final layer of the network model after multiple max-pooling & convolutional layers is called the fully connected layer. Since all the neurons of this final layer connect to each neuron of the previous & successive layer, it ensures high accuracy of the neural network model.

v. **Loss layer:** It's the final layer that calculates the difference between the true output (given in the training set) & predicted output. Use of this layer is dependent on user application:

Softmax is commonly used among Sigmoid cross-entropy loss, Softmax loss, Euclidean loss methods because from N mutually exclusive classes, it has the capability of predicting a single class.

Hyperparameter Tuning: Filter size, amount of filters & max-pooling shape are chosen carefully to achieve higher efficiency from the overall architecture.

Regularization methods: CNN uses the regularization process to prevent data from overfitting or to introduce extra features to remedy the ill-posed problem. Various regularization methods are:

(a) Empirical: Stochastic pooling, Dropout, DropConnect, Artificial Data

(b) Explicit: Max norm constraints, Early stopping, Weight decay, Numbers of parameters

5.10 Variational Auto-encoders (VAEs) [40, 41]

Introduction: VAEs are a deep learning technique in order to learn latent representations. Instead of a single point, these autoencoders give back a distribution over latent space. Moreover, the algorithm adds a regularization parameter to the returned distribution in the loss function. That's how it resolves the problem of the irregularity of the latent space.

Deep Generative Model: Let a directional latent-variable model

$p(x,z) = p(x \mid z)\,p(z)$ with observed $x \in \chi$, on that χ can be either discrete or continuous and latent $z \in \Re^k$.

Consider the ECG data collection and latent factors (not observed during the workout) that clarify characteristics of the ECG continuum. e.g., the frequency & amplified ECG data can be encoded by one coordination, another by the corresponding pattern of disease. The model multilayer can be defined as:

$$p(x \mid z_1)\,p(z_1 \mid z_2)\,p(z_2 \mid z_3) \cdots p(z_{m-1} \mid z_m)\,p(z_m)$$

These are classified as profound generative models and can learn latent representation hierarchies.

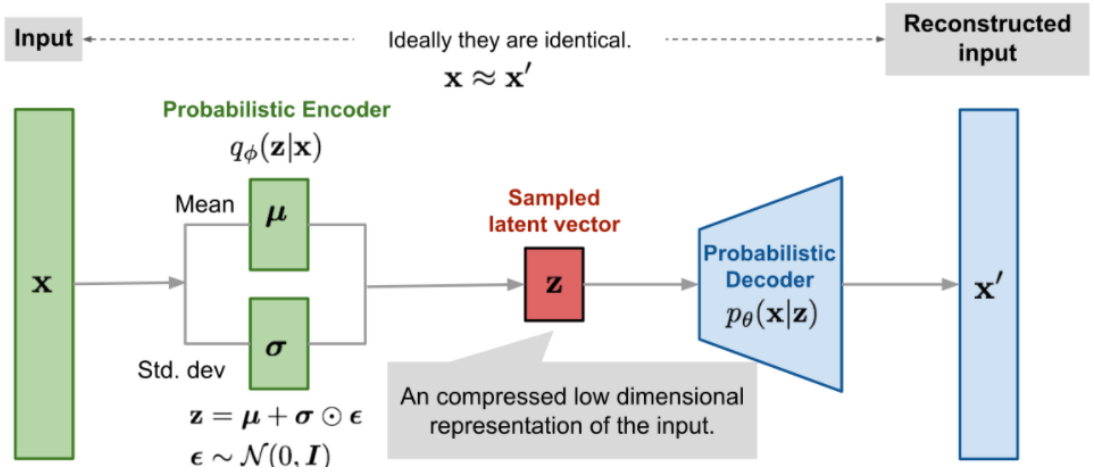


Figure 5.10: Variational Auto-encoders Architecture [40]

Auto-encoding variational Bayes: It's an algorithm based on the ideas of variational inference which can solve three learning and interference tasks efficiently. This algorithm is represented by the variational auto-encoder. Our focus is on optimizing the evidence lower bound (ELBO) in variational inference:

$$\mathcal{L}\left(p_\theta, q_\phi\right) = \mathbb{E}_{q\phi}\left[log\ p_\theta\left(x, z\right) - log\ q_\phi\left(z \mid x\right)\right]$$

The ELBO fulfills the equation:

$$log\ p_\theta\left(x\right) = KL(q_\phi\left(z \mid x\right) \| p\left(z \mid x\right) + \mathcal{L}\left(p_\theta, q_\phi\right)$$

We define $q(z \mid x)$ to be conditioned on x, where x is fixed. That means that we basically are choosing a dissimilar $q(z)$ effectively for each x so as to provide an improved posterior approximation instead of always choosing the identical $q(z)$.

Black-box variational inference: It is designed to optimize q, which works for broad q groups. It suggests that q_ϕ which is used in the equation is differentiable by its parameters ϕ. Rather than just doing inferences, we will be coding to learn by gradient descent on both ϕ and θ simultaneously. Optimizing ϕ will hold ELBO close around $\log p(x)$; optimization of θ continue to drive up the lower bound (and hence $\log p(x)$).

The score function gradient estimator: We need to calculate the gradient by using score function estimator to make the black-box variational inference:

$$\nabla_{\theta,\phi}\ \mathbb{E}_{q\phi}\left[log\ p_\theta(x,z) - log\ q_\phi(z)\right] = \mathbb{E}_{q\phi}\left[\left(log\ p_\theta(x,z) - log\ q_\phi(z)\right)\nabla_\phi log\ q_\phi(z)\right]$$

The SGVB estimator: The reformulation of the ELBO can be equated as,

$$log\ p(x) \geq \mathbb{E}_{q_\phi(z|x)}[log\ p_\theta(x|z) - KL(q_\phi(z|x)||\ p(z))]$$

Assume χ as a data point observed. The right-hand side has two individual terms, both of which include a sample $z \sim q(z|\ x)$, that could be considered as a code that describes χ. The encoder is called q. Given the sampled code z, $log\ p(x|\ z)$ term is the log-likelihood of the observing value of χ. The maximized term $p(x|\ z)$ offers a high probability to the original χ. From code z, it is trying to reconstruct χ. That's why we can say $p(x|\ z)$ is the decoder network, and that term is being called reconstruction error.

The 2nd term, known as the regularization term, is the divergence between the $p(z)$ and $q(z|\ x)$ which should be defined as a unit Normal. Codes z to appear Gaussian are needed. This forces it to learn some more interesting features instead of preventing $q(z|\ x)$ identity mapping from being encoded.

And that is how the optimized goal attempts to suit a $q(z|\ x)$ will map χ in a useful latent room z; that we can reconstruct from χ using $p(x|\ z)$. This form of the target is reminiscent of auto-encoders neural networks.

The Reparameterization tricks: Under some minor conditions, the distribution $q_\phi(z \mid x)$ can be written as a 2-step generative process:

i. Like standard normal $\mathcal{N}(0,1)$, sample a noise variable ϵ from a simple distribution $p(\epsilon)$.

$$\epsilon \approx p(\epsilon)$$

ii. To map the random noise to a more complex distribution, apply a deterministic transformation.

$$z = g_\phi(\epsilon, x).$$

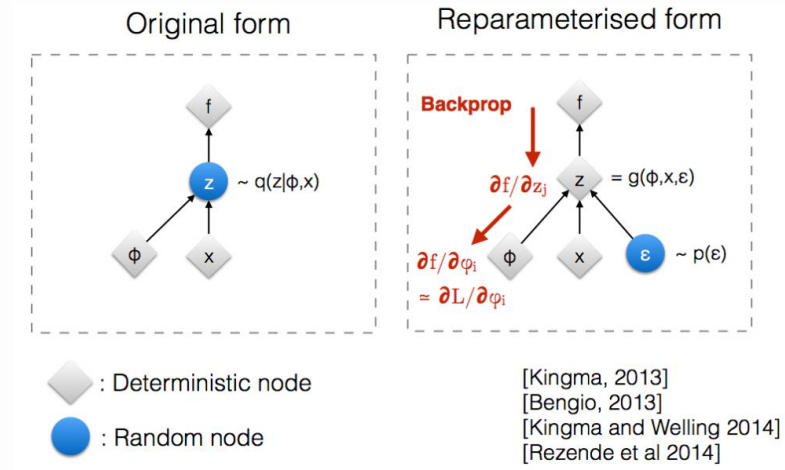


Figure 5.11: Reparameterization Tricks [40]

The variational auto-encoder: Now, let's define the AEVB autoencoder. In its simplest form, the AEVB algorithm is the amalgam of:

(a) Auto-encoding ELBO reformulation

(b) Black-box variational inference approach

(c) Reparameterization-based low-variance gradient estimator.

The task of the AEVB algorithm is to optimize the auto-encoding ELBO with the help of a reparametrized gradient estimator using the black-box variational inference method. This AEVB algorithm can be used for any deep generative model p_θ with latent variables that can be differentiable in θ. The AEVB algorithm is used by a variational autoencoder to learn a particular model p while using an encoder q. The model p & q are parameterized as:

$$p(x,z) = \mathcal{N}\left(x; \vec{\mu}(z), diag\left(\vec{\sigma}(z)\right)^2\right)$$

$$q(z \mid x) = \mathcal{N}\left(z; \vec{\mu}(x), diag\left(\vec{\sigma}(x)\right)^2\right)$$

$$p(z) = \mathcal{N}(z; 0, I)$$

5.11 Deep Belief Network [42, 43]

Introduction: Deep belief networks are a collection of binary latent variables modeled after the human brain that use unsupervised learning and probabilities. DBN contains both the undirected & directed layers. DBN works globally for learning the whole dataset & handle each layer in order. DBN shows a greater ability to process complex information & recognize patterns like the human brain due to these properties.

Architecture: In DBN, the neurons of each layer have a weighted connection to all neurons of the previous & next layer; the top layer connections are undirected & it allows associative memory formation. However, bottom layer connections are directed. DBN can be compared to a pile of Restricted Boltzmann Machines (RBMs); neurons in DBN don't communicate sideways within their layer, unlike RBMs though.

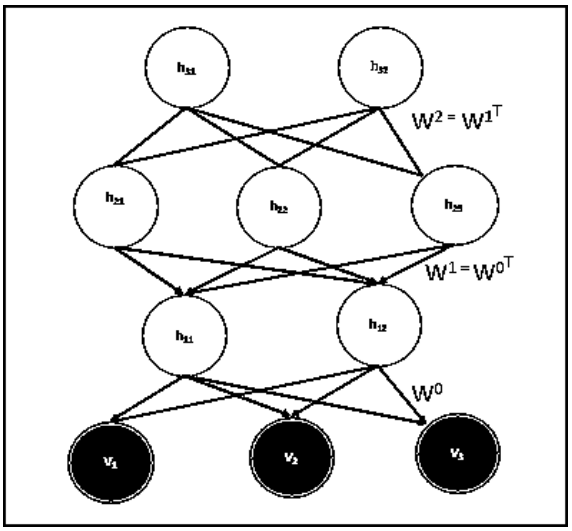


Figure 5.12: Deep Belief Network Architecture [42]

Hidden layer neurons identify the correlations in the data, but they perform like hidden layers to the preceding & as visible layers to the succeeding neurons.

Training Procedure: Deep belief networks are pre-trained by greedy learning algorithms, which start from the bottom but gradually move to the upper layer using layer-by-layer (each layer trained at a time) basis learning, gradually finding optimal value at each layer & finally finds global optimum. Greedy learning algorithms are famous for training deep belief networks since they work to optimize the weights at every layer. Moreover, they are quick and efficient. In the Contrastive Divergence model, gradient descent is used for updating weights using the equations below:

If the probability of a visible vector, $prob(a) = \frac{1}{Z}\sum_{b} e^{-E(a,b)}$

Then, updated weight, $\theta_{ij}(t) = \theta_{ij}(t-1) + \eta \frac{\partial \log(prob(a))}{\partial \theta_{ij}}$

The Contrastive Divergence process:

1. Visible units are initialized as a training vector.
2. If bias of hidden unit b_j is c_j; given the visible training vector, update hidden units in parallel:

$$prob(b_j = 1 | A) = \sigma(c_j + \sum_i a_i \theta_{ij})$$

3. If bias of a_i is d_i; since hidden units are found in the previous step, it's time for parallel upgradation of the visible units:

$$prob(a_i = 1 | B) = \sigma(d_i + \sum_j b_j \theta_{ij})$$

   This is known as the "reconstruction" step.
4. Since reconstructed visible units are found, repeat step 2.
5. Update weight as follows:

$$\Delta\theta_{ij} \propto \langle a_i b_j \rangle_{data} - \langle a_i b_j \rangle_{reconst.}$$

After the end of training of one RBM layer, another RBM layer is readied as a training vector & goes for training, taking the recently completed layer's output as its input. The new training vector is trained using the procedure mentioned above & the overall process is repeated until the stopping criterion is fulfilled.

Chapter 6

RESULT & ANALYSIS

6.1 Implementation of Machine Learning Algorithms

Six machine learning algorithms were implemented to classify five categories of ECG signals. Firstly, they were implemented with the default hyperparameter (DHP) setting. The performance was too low. That is why Randomized Search Cross-Validation (RSCV) hyperparameter tuning technique was introduced. Randomized Search Cross-Validation (RSCV) is an efficient hyperparameter tuning technique that is suitable for a large amount of training data [44]. It is also more time-efficient than another hyperparameter tuning technique called Grid Search Cross-Validation (GSCV) [45]. Every machine learning model was trained with the RSCV hyperparameter optimization technique, and the performance of the model was increased.

6.1.1 K-Nearest Neighbor

K-nearest neighbor was implemented using neighbor size 19 and leaf size six found from Randomized Search Cross-validation hyperparameter optimization. KDTree algorithm was used to calculate the distance between two neighbors. Distance function was used as a weight function. After training and testing the model, the confusion matrix was found like below:

Table 6.1: Confusion Matrix (KNN)

Confusion Matrix (KNN)		**Predicted**				
		N	**S**	**V**	**F**	**Q**
Actual	**N**	12577	619	419	488	215
	S	203	73	155	91	97
	V	490	231	102	178	161
	F	81	61	36	23	19
	Q	282	177	180	183	410

Classification Report: The classification report which is found from the calculation from the above confusion matrix is portrayed below:

Table 6.2: Classification Report (KNN)

Class	Accuracy	Precision	Recall	F-1 Score
N	84.06%	0.92	0.88	0.90
S	90.69%	0.063	0.12	0.082
V	89.46%	0.11	0.088	0.099
F	93.52%	0.024	0.10	0.039
Q	92.51%	0.45	0.33	0.38

Overall Accuracy= 75.12%

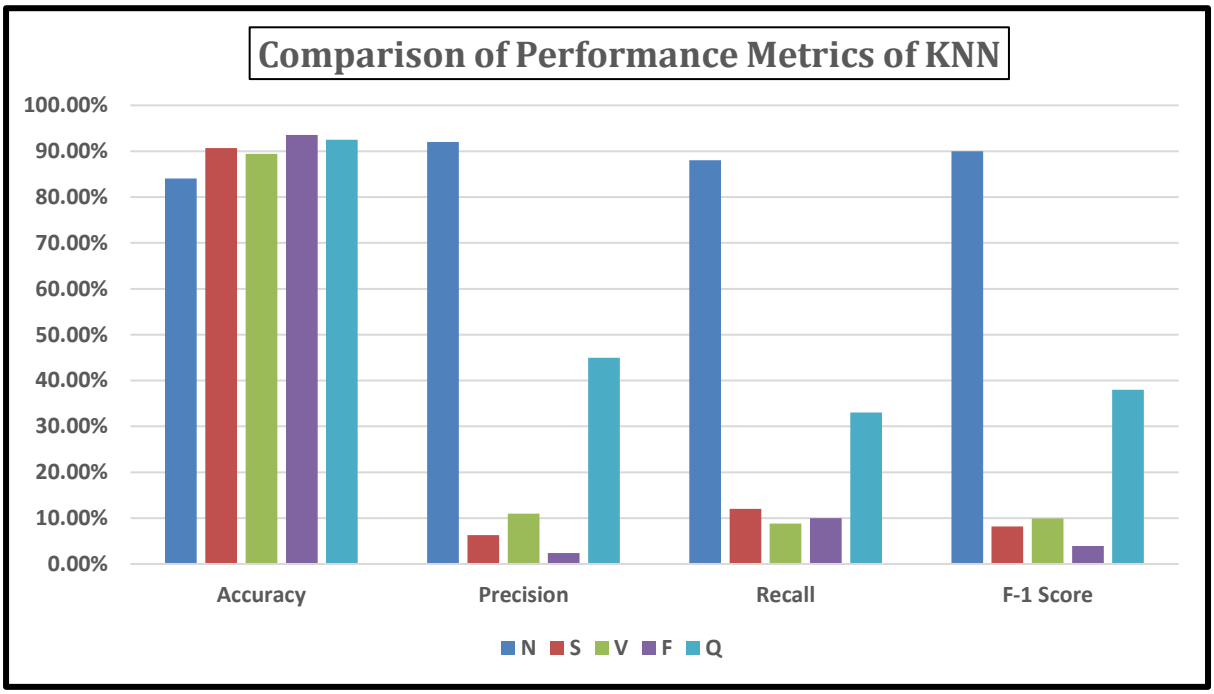


Figure 6.1: Comparison of Performance Metrics of KNN

6.1.2 Random Forest Classifier

Random Forest Classifier was implemented with 'entropy' as the measurement of the quality s plit, max depth of 8. 'Sqrt' was used as max feature function. The number of trees in the forest was taken as 200. After training and testing the model, the confusion matrix was found like be low:

Table 6.3: Confusion Matrix (RFC)

Confusion Matrix (RFC)		**Predicted**				
		N	**S**	**V**	**F**	**Q**
Actual	**N**	14259	141	51	33	10
	S	67	361	16	11	06
	V	73	16	1036	21	21
	F	27	09	11	83	02
	Q	39	47	48	25	1135

Classification Report: The classification report which is found from the calculation from the above confusion matrix is portrayed below:

Table 6.4: Classification Report (RFC)

Class	**Accuracy**	**Precision**	**Recall**	**F-1 Score**
N	83.06%	0.93	0.85	0.89
S	88.52%	0.18	0.25	0.21
V	87.97%	0.15	0.13	0.14
F	91.25%	0.087	0.23	0.13
Q	91.57%	0.34	0.36	0.35

Overall Accuracy= 71.18%

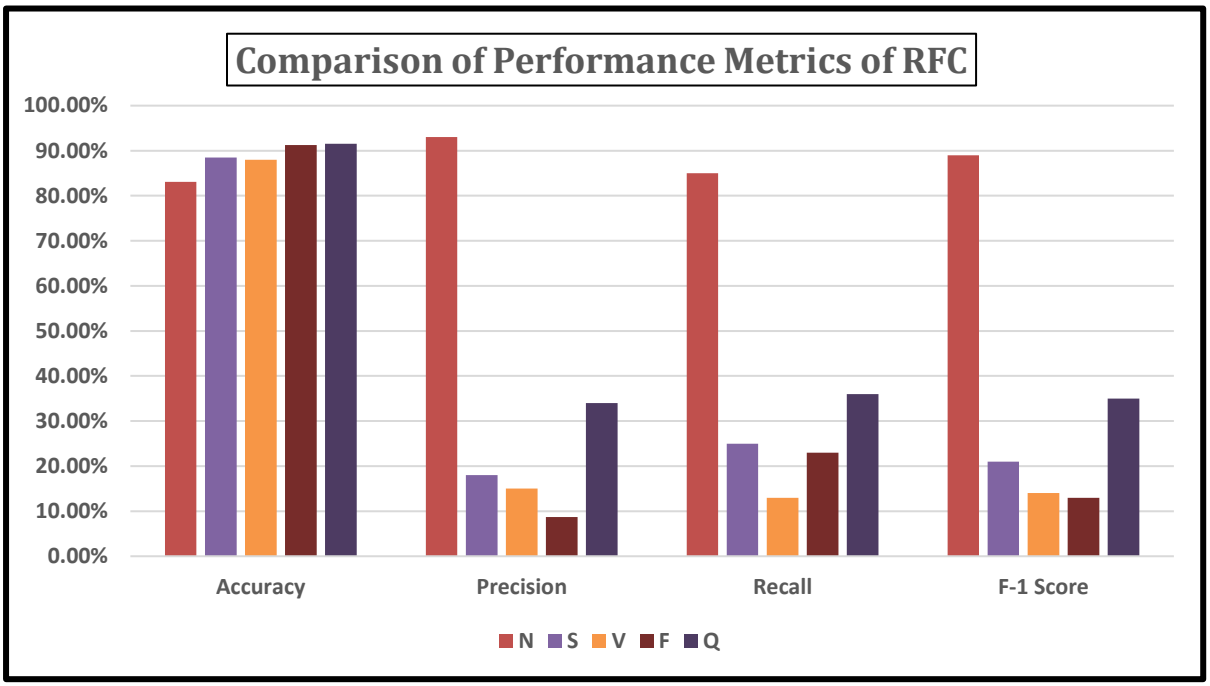


Figure 6.2: Comparison of Performance Metrics of RFC

6.1.3 Support Vector Machine

Support Vector Machine was executed with RBF kernel. 'Scale' was taken as kernel coefficient for RBF. The regularization parameter 'c' was taken as 1. After training and testing the model, the confusion matrix was found like below:

Table 6.5: Confusion Matrix (SVM)

Confusion Matrix (SVM)		**Predicted**				
		N	**S**	**V**	**F**	**Q**
Actual	**N**	13665	348	253	161	114
	S	148	134	103	41	53
	V	381	219	329	124	121
	F	48	9	29	38	12
	Q	241	144	136	140	560

Classification Report: The classification report which is found from the calculation from the above confusion matrix is portrayed below:

Table 6.6: Classification Report (SVM)

Class	Accuracy	Precision	Recall	F-1 Score
N	90.35%	0.94	0.94	0.94
S	93.93%	0.16	0.28	0.20
V	92.22%	0.39	0.28	0.33
F	96.79%	0.075	0.28	0.12
Q	94.52%	0.65	0.46	0.54

Overall Accuracy= 83.9%

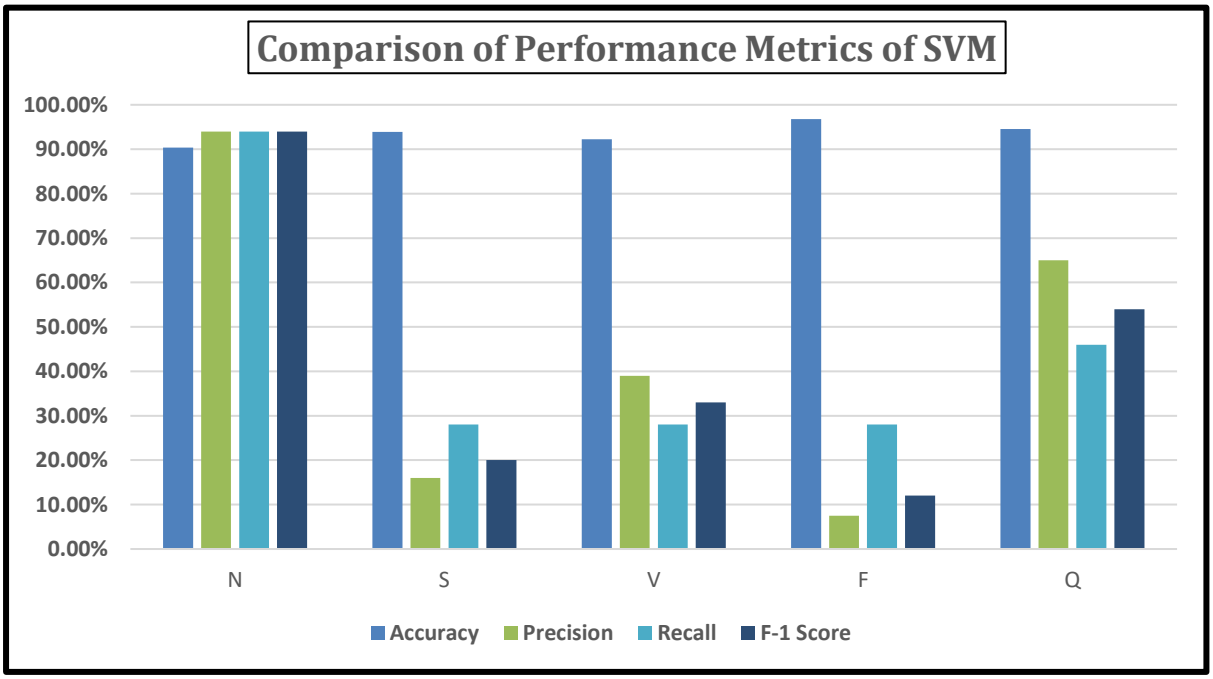


Figure 6.3: Comparison of Performance Metrics of SVM

6.1.4 Stochastic Gradient Descent

Stochastic Gradient Descent is performed with 'perceptron' loss function and l2 regularization. The regularization constant multiplier was taken as 0.001, and the epsilon-insensitive loss function was taken as 'Huber.' 'Adaptive' learning rate was used. After training and testing the model, the confusion matrix was found like below:

Table 6.7: Confusion Matrix (SGD)

Confusion Matrix (SDG)		**Predicted**				
		N	**S**	**V**	**F**	**Q**
Actual	**N**	12455	634	434	481	228
	S	195	58	179	102	106
	V	496	239	95	183	169
	F	87	71	43	32	25
	Q	285	182	191	152	439

Classification Report: The classification report which is found from the calculation from the above confusion matrix is portrayed below:

Table 6.8: Classification Report (SGD)

Class	**Accuracy**	**Precision**	**Recall**	**F-1 Score**
N	83.82	0.92	0.88	0.90
S	90.33%	0.049	0.092	0.064
V	89.04%	0.10	0.080	0.090
F	93.48%	0.034	0.12	0.053
Q	92.38%	0.45	0.35	0.40

Overall Accuracy= 74.52%

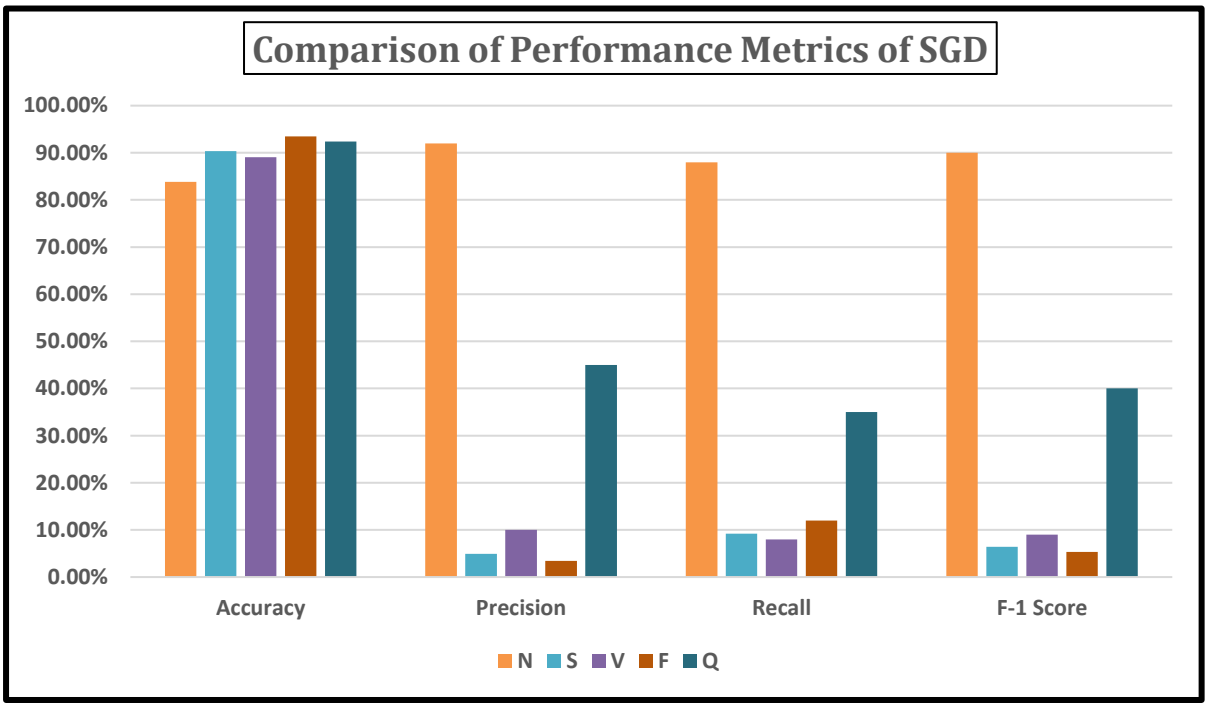


Figure 6.4: Comparison of Performance Metrics of SGD

6.1.5 AdaBoost

Adaptive Boosting algorithm was incorporated with Decision Tree base estimator, the maximum number of estimators of 100 and learning rate 0.01. AdaBoost assigns more weight to the wrongly classified observation so that in the next iteration, it gets a high probability for the classification. The model was iterated 100 times which was the number of maximum estimators. After training and testing the model, the confusion matrix was found like below:

Table 6.9: Confusion Matrix (AdaBoost)

Confusion Matrix (AdaBoost)		**Predicted**				
		N	**S**	**V**	**F**	**Q**
Actual	**N**	13216	508	355	281	159
	S	177	78	126	61	59
	V	463	224	142	149	146
	F	53	15	29	27	12
	Q	261	194	162	164	440

Classification Report: The classification report which is found from the calculation from the above confusion matrix is portrayed below:

Table 6.10: Classification Report (AdaBoost)

Class	Accuracy	Precision	Recall	F-1 Score
N	87.14%	0.93	0.91	0.92
S	91.94%	0.073	0.16	0.099
V	90.29%	0.17	0.12	0.14
F	95.65%	0.040	0.20	0.066
Q	93.41%	0.54	0.36	0.43

Overall Accuracy= 79.21%

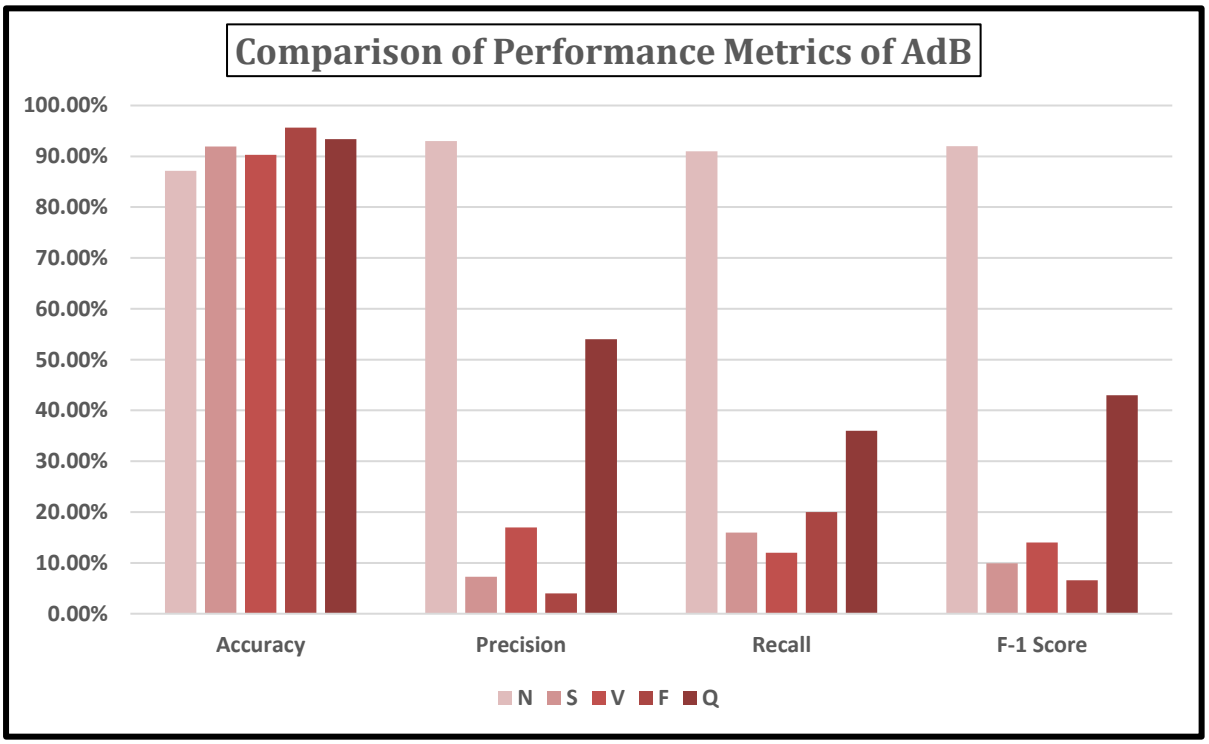


Figure 6.5: Comparison of Performance Metrics of AdB

6.1.6 XGBoost

Extreme Gradient Boosting was implemented where 'gbtree' was used as a booster. The learning rate was taken as 0.2, and the loss reduction parameter 'gamma' was taken as 0.1. The maximum depth of the trees was taken as ten, and gradient-based sampling method was used. After training and testing the model, the confusion matrix was found like below:

Table 6.11: Confusion Matrix (XGBoost)

Confusion Matrix (XGB)		**Predicted**				
		N	**S**	**V**	**F**	**Q**
Actual	**N**	12816	609	420	466	201
	S	185	73	140	73	70
	V	473	211	123	164	156
	F	56	18	31	22	16
	Q	263	201	166	170	428

Classification Report: The classification report which is found from the calculation from the above confusion matrix is portrayed below:

Table 6.12: Classification Report (XGBoost)

Class	**Accuracy**	**Precision**	**Recall**	**F-1 Score**
N	84.77%	0.93	0.88	0.91
S	91.41%	0.066	0.13	0.088
V	89.97%	0.14	0.11	0.12
F	94.34%	0.025	0.15	0.042
Q	92.92%	0.49	0.35	0.41

Overall Accuracy= 76.6%

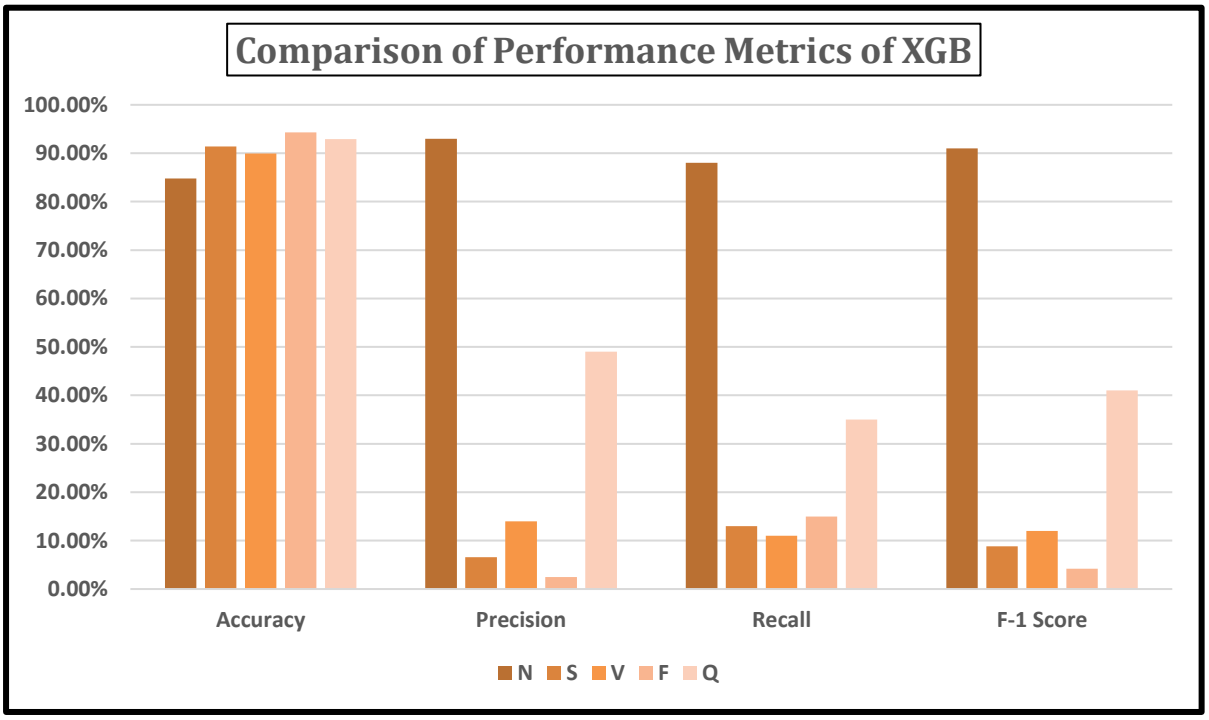


Figure 6.6: Comparison of Performance Metrics of XGB

6.2 Implementation of Deep Learning Algorithms

From the previous section, it was seen that Machine Learning Algorithms were not performing that much good. The reasons behind this are multiclass classification and a considerable amount of training data. So Deep Learning Algorithm implementation was performed, which can handle multiclass classification more efficiently [46]. For hyperparameter optimization, instead of Grid Search Cross-Validation (GSCV), Randomized Search Cross-Validation (RSCV) was also used. RSCV is more efficient than GSCV in the case of deep learning [47].

6.2.1 Convolutional Neural Network

The convolutional neural network was incorporated. Here, four 2D convolutional layers and 2D Global average pooling were used. Put the first convolutional layer; the kernel size was taken 10, 2 and 5, 2 was taken for the rest of the convolutional layers. 'ReLu' was used as an activation function, and 'Batch normalization' was used to standardize the Data while giving input to each layer of the neural network. Drop out function was used with the rate is equal to 0.5 to prevent the overfitting of the training data. The last layer was a fully connected layer for achieving classification results where the softmax function was used as the activation function. The confusion matrix which was found after training and testing of the model is portrayed

below:

Table 6.13: Confusion Matrix (CNN)

Confusion Matrix (CNN)		**Predicted**				
		N	**S**	**V**	**F**	**Q**
Actual	**N**	14259	141	51	33	10
	S	67	361	16	11	06
	V	73	16	1036	21	21
	F	27	09	11	83	02
	Q	39	47	48	25	1135

Classification Report: The classification report which is found from the calculation from the above confusion matrix is portrayed below:

Table 6.14: Classification Report (CNN)

Class	**Accuracy**	**Precision**	**Recall**	**F-1 Score**
N	97.49%	0.99	0.98	0.98
S	98.22%	0.63	0.78	0.70
V	98.54%	0.89	0.89	0.89
F	99.21%	0.48	0.63	0.54
Q	98.87%	0.97	0.88	0.92

Overall Accuracy= 96.16%

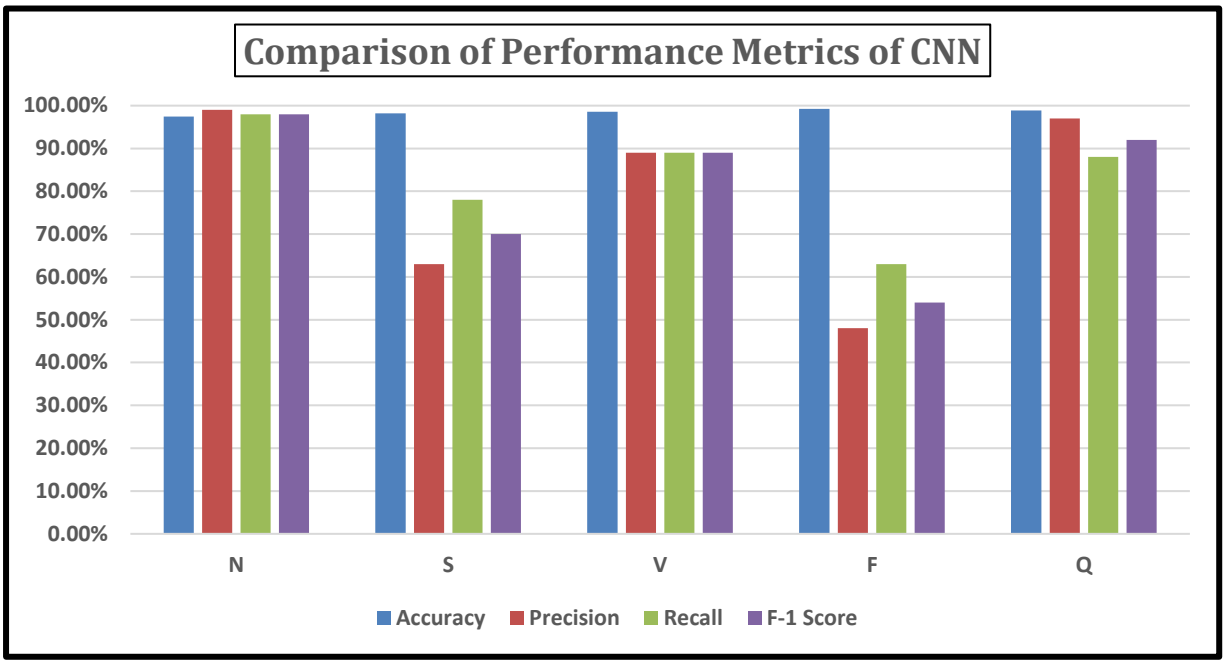


Figure 6.7: Comparison of Performance Metrics of CNN

6.2.2 Recurrent Neural Network (LSTM)

RNN Bidirectional LSTM was performed because rather than other RNN models, LSTM is more efficient since it doesn't have the weight vanishing problem. Also, bidirectional LSTM is used as It preserves both the information of past and future input. 64-layer bidirectional LSTM was performed where 'tanh' and 'sigmoid' was used as activation function respectively. Finally, a dense layer having a 'softmax' activation function was added for getting the classification. After training and testing, this confusion matrix was found:

Table 6.15: Confusion Matrix (LSTM)

Confusion Matrix (RNN)		**Predicted**				
		N	**S**	**V**	**F**	**Q**
Actual	**N**	14241	148	53	37	13
	S	78	345	23	11	21
	V	21	19	1029	23	22
	F	28	09	13	79	07
	Q	41	44	49	27	1110

Classification Report: The classification report which is found from the calculation from the above confusion matrix is portrayed below:

Table 6.16: Classification Report (LSTM)

Class	Accuracy	Precision	Recall	F-1 Score
N	97.27%	0.98	0.98	0.98
S	97.99%	0.61	0.72	0.66
V	98.39%	0.88	0.88	0.88
F	99.12%	0.45	0.58	0.50
Q	98.72%	0.95	0.87	0.91

Overall Accuracy=95.74%

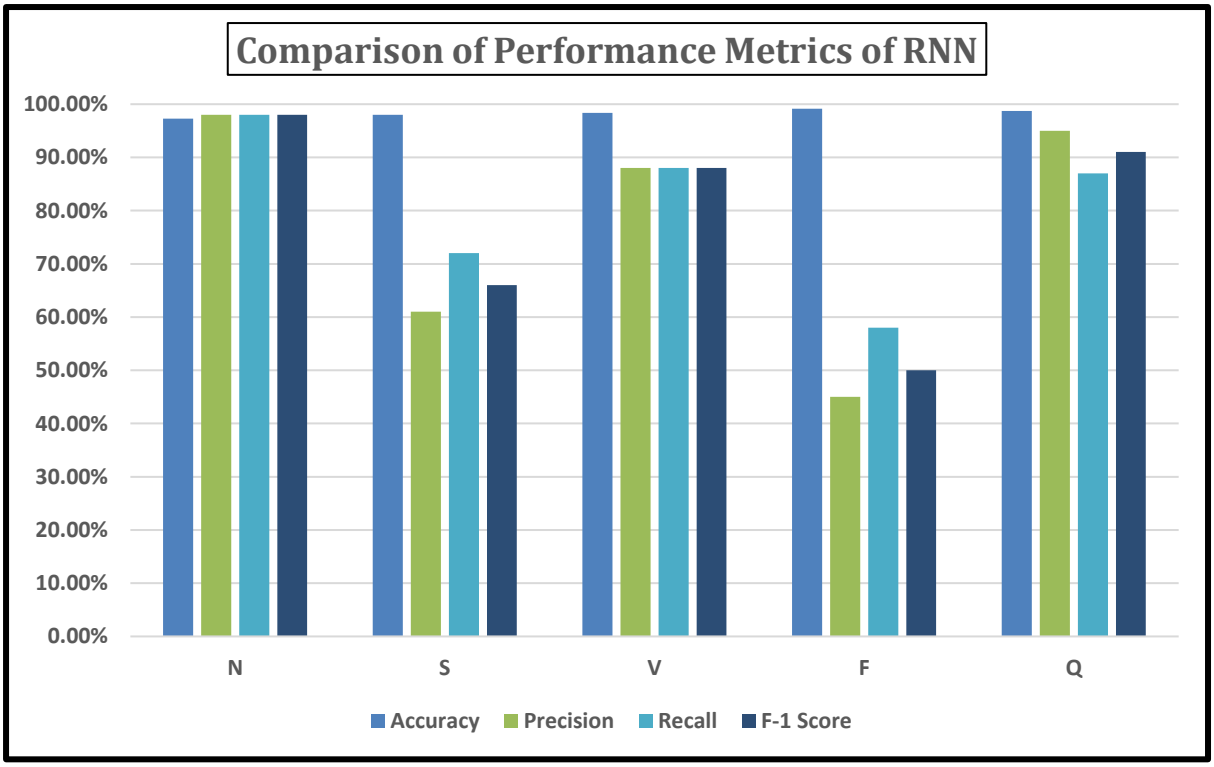


Figure 6.8: Comparison of Performance Metrics of LSTM

6.2.3 Deep Convolutional Neural Network

The deep convolutional neural network was implemented where there were 7 two dimensional convolutional layers. For the first convolutional layers, the kernel size was taken (20,2) and (10,2) for the other layers. 'ReLu' was used as an activation function, and 'Batch normalization,' 'Drop out function' were used like previously implemented CNN. The last layer was fully connected, where ReLu was used as an activation function. After train and testing, the confusion matrix was like below:

Table 6.17: Confusion Matrix (Deep CNN)

Confusion Matrix (Deep CNN)		**Predicted**				
		N	**S**	**V**	**F**	**Q**
Actual	**N**	14302	117	37	30	8
	S	55	386	3	1	0
	V	21	1	1128	5	3
	F	15	1	7	105	0
	Q	7	0	1	0	1278

Classification Report: The classification report which is found from the calculation from the above confusion matrix is portrayed below:

Table 6.18: Classification Report (Deep CNN)

Class	**Accuracy**	**Precision**	**Recall**	**F-1 Score**
N	98.34%	0.99	0.99	0.99
S	98.98%	0.76	0.87	0.81
V	99.55%	0.96	0.97	0.97
F	99.66%	0.74	0.82	0.78
Q	99.89%	0.99	0.99	0.99

Overall Accuracy=98.22%

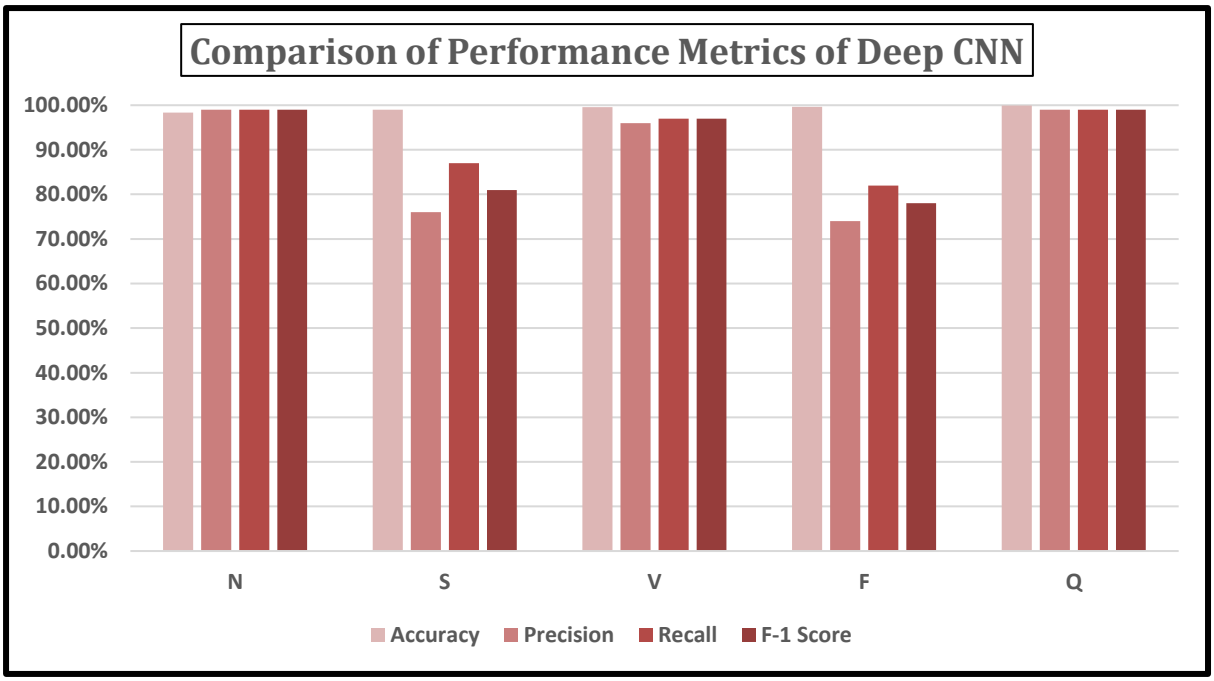


Figure 6.9: Comparison of Performance Metrics of Deep CNN

6.2.4 Variational Autoencoder

Variational autoencoder learns from the distribution of data by calculating the mean and variance of the latent vectors for every sample forcing to follow a standard normal distribution. The autoencoder was formed by repeating multiple convolutional layers and pooling layers in the 'keras' platform. Two fully connected layers were used for calculating mean and log variance from the convoluted features. These mean and log-variance were used to measure the latent encoding for the input data points. Then this latent encoding vector was passed to build the model.

The decoder is also formed by deconvolutional layers and up-sampling layers. The decoder takes the latent encoding vector as input and correlates with the original data. Thus, the decoder model was built.

Finally, the variational autoencoder was formed by combining the encoder and decoder parts. After training and testing, this confusion matrix was found:

Table 6.19: Confusion Matrix (VAE)

Confusion Matrix (VAE)		Predicted				
		N	**S**	**V**	**F**	**Q**
Actual	**N**	14284	128	44	30	8
	S	57	379	10	2	0
	V	26	7	1113	12	4
	F	19	5	9	97	0
	Q	27	7	3	5	1235

Classification Report: The classification report which is found from the calculation from the above confusion matrix is portrayed below:

Table 6.20: Classification Report (VAE)

Class	Accuracy	Precision	Recall	F-1 Score
N	98.06%	0.99	0.99	0.99
S	98.77%	0.72	0.85	0.78
V	99.34%	0.94	0.96	0.95
F	99.53%	0.66	0.75	0.70
Q	99.69%	0.99	0.97	0.98

Overall Accuracy=97.7%

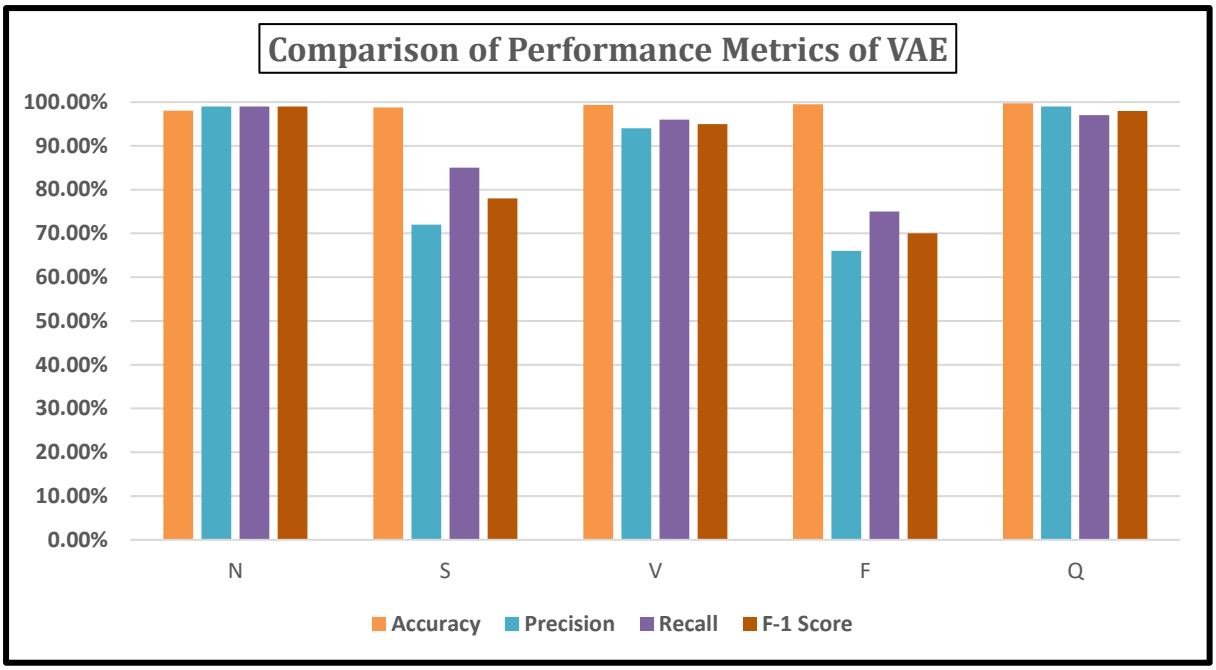


Figure 6.10: Comparison of Performance Metrics of VAE

6.2.5 Deep Belief Network

Deep belief network is based on Restricted Boltzmann Machines (RBM). Deep Belief network was implemented from TensorFlow, Scikit-Learn, and Numpy library with hidden layer structure (50,50), restricted Boltzmann machine learning rate 0.05 and model learning rate 0.1 and 'ReLu' activation function. After train and testing, this confusion matrix was found:

Table 6.21: Confusion Matrix (DBN)

Confusion Matrix (DBN)		**Predicted**				
		N	**S**	**V**	**F**	**Q**
Actual	**N**	14278	130	46	31	9
	S	60	374	11	2	1
	V	29	08	1105	14	6
	F	22	08	9	92	01
	Q	39	20	8	13	1235

Classification Report: The classification report which is found from the calculation from the above confusion matrix is portrayed below:

Table 6.22: Classification Report (DBN)

Class	Accuracy	Precision	Recall	F-1 Score
N	97.91%	0.99	0.99	0.99
S	98.63%	0.69	0.83	0.76
V	99.25%	0.94	0.95	0.94
F	99.43%	0.61	0.70	0.65
Q	99.45%	0.99	0.94	0.96

Overall Accuracy=97.34%

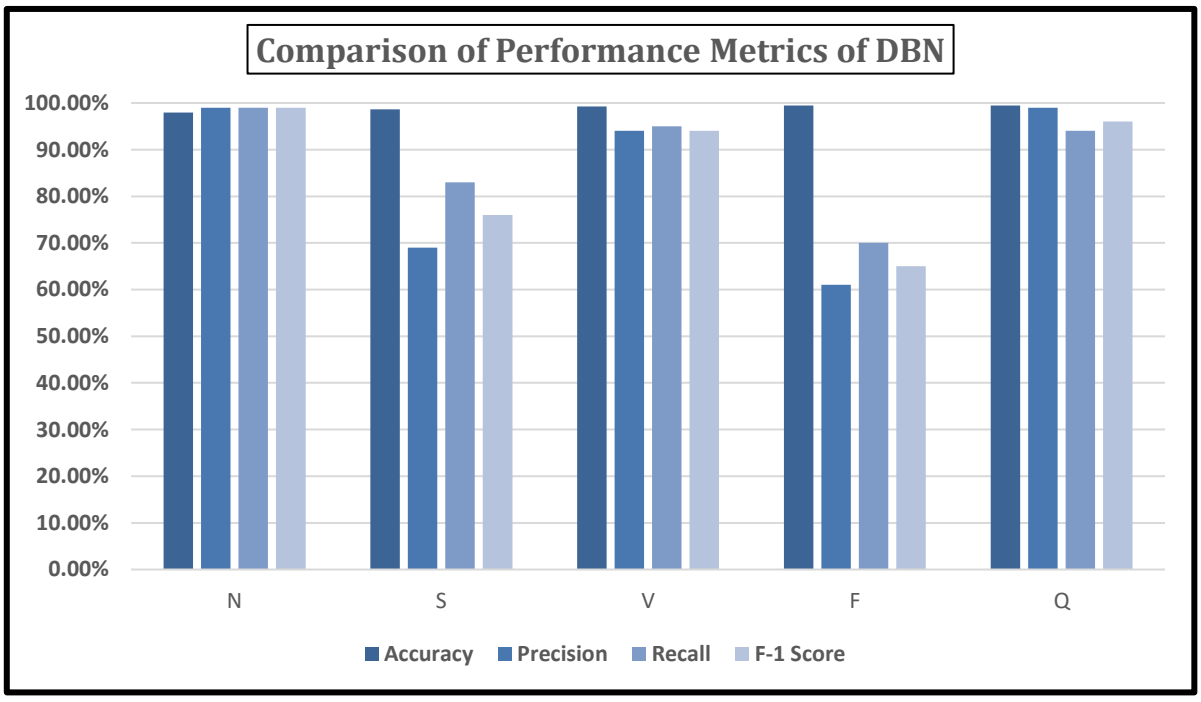


Figure 6.11: Comparison of Performance Metrics of DBN

6.2.6 Hybrid Deep Neural Network

Deep CNN, Variational Autoencoder, and Deep Belief Network are ensembled to form a Deep Hybrid Neural Network. 11-layer deep CNN was connected to a 7-layer variational autoencoder, and then a 64-layer deep belief network was connected to it. At last, a fully connected layer was added for getting the classification result. There are several benefits of merging these three deep neural network architectures together. Data gets efficient nodes to propagate as well as classification gets more perfect. But since it is a bigger network, the mathematical calculation gets more complicated. Though it takes more amount of time for training and testing, it gives more efficient classification. After training & testing, the confusion matrix was found like below:

Table 6.23: Confusion Matrix (HDNN)

Confusion Matrix (HDNN)		**Predicted**				
		N	**S**	**V**	**F**	**Q**
Actual	**N**	14449	30	6	5	5
	S	50	392	2	1	0
	V	10	2	1136	6	3
	F	7	1	3	117	0
	Q	3	0	0	0	1283

Classification Report: The classification report which is found from the calculation from the above confusion matrix is portrayed below:

Table 6.24: Classification Report (HDNN)

Class	**Accuracy**	**Precision**	**Recall**	**F-1 Score**
N	99.34%	1	1	1
S	99.51%	0.92	0.88	0.90
V	99.82%	0.99	0.98	0.99
F	99.87%	0.91	0.91	0.91
Q	99.94%	0.99	1	1

Overall Accuracy=99.23%

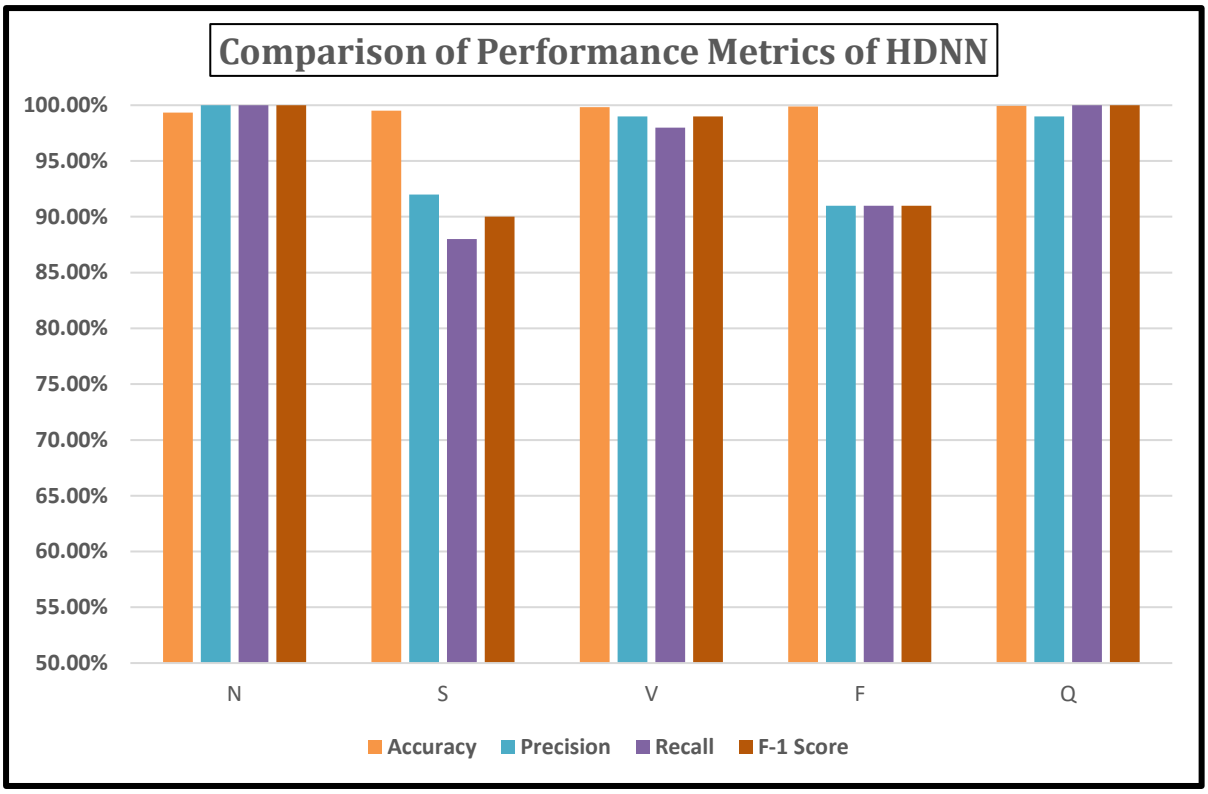


Figure 6.12: Comparison of Performance Metrics of HDNN

Overall Performance of All the Classifiers:

Table 6.25: Overall Accuracy of Classifiers from Machine Learning

Classifier	Accuracy
K-Nearest Neighbour	75.12%
Random Forest Classifier	71.18%
Support Vector Machine	83.9%
Stochastic Gradient Descent	74.52%
AdaBoost	79.21%
XG Boost	76.7%

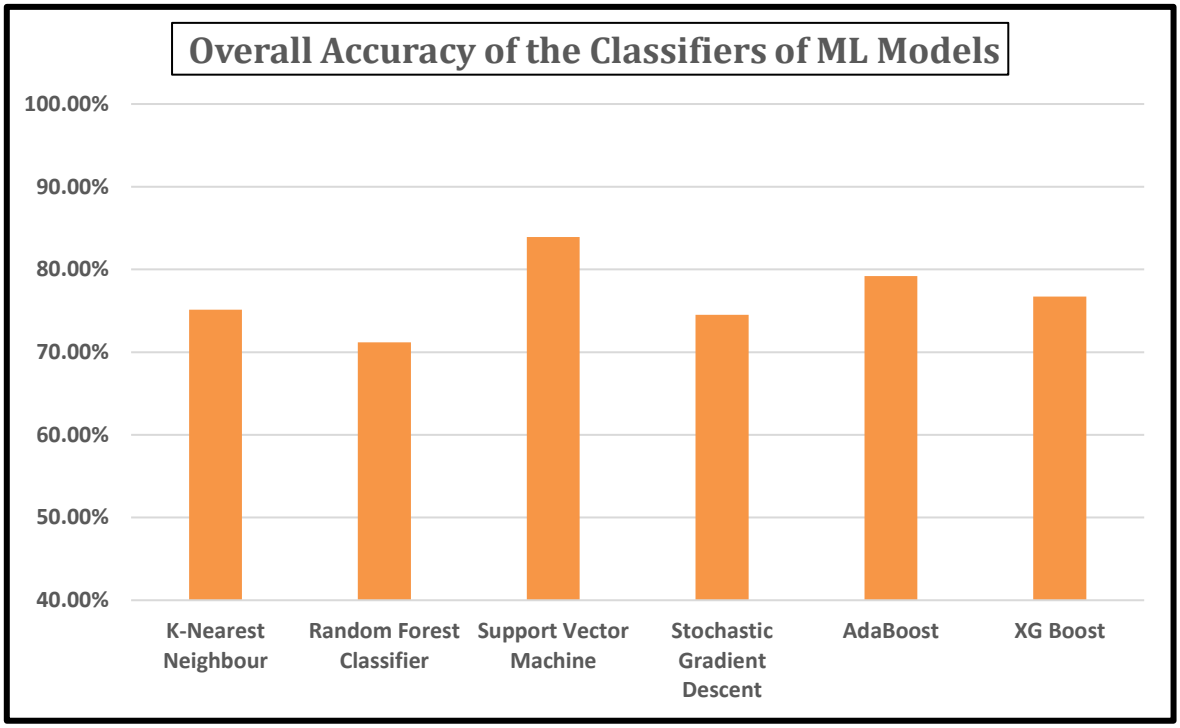


Figure 6.13: Overall Accuracy of Classifiers from ML Models

Table 6.26: Overall Accuracy of Classifiers from Deep Learning

Classifier	Accuracy
Convolutional Neural Network	96.16%
Recurrent Neural Network	95.74%
Deep Convolutional Neural Network	98.22%
Variational Autoencoder	97.7%
Deep Belief Network	97.34%
Hybrid Deep Neural Network (Proposed)	**99.23%**

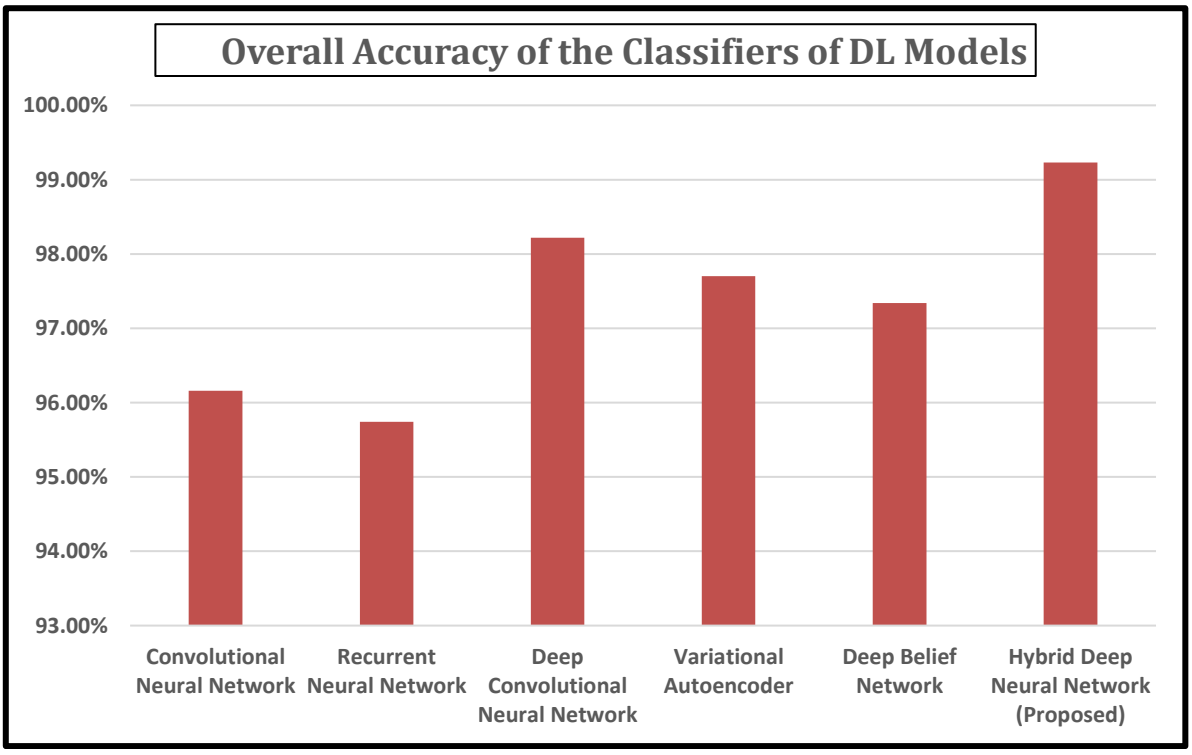


Figure 6.14: Overall Accuracy of Classifiers from DL Models

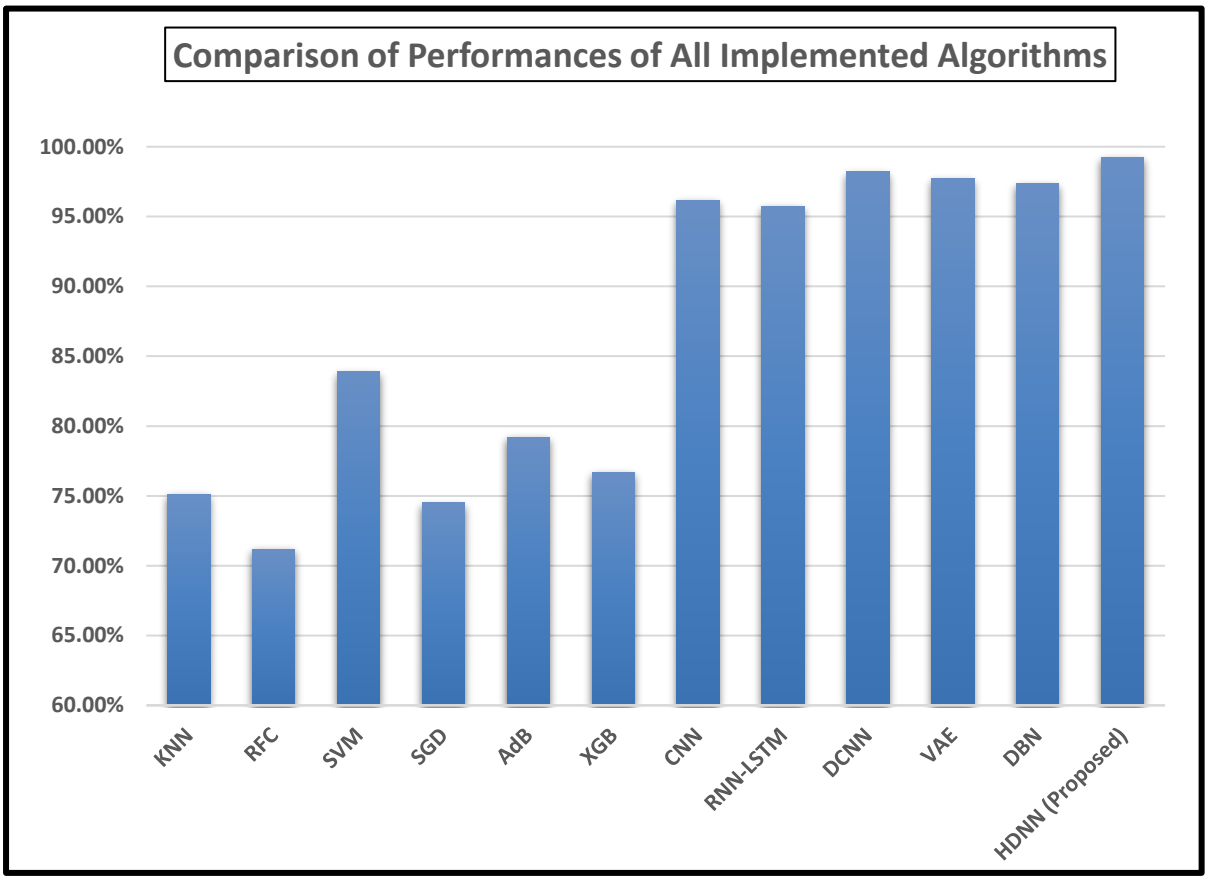


Figure 6.15: Comparison of Performances of All Implemented Algorithms

Here, it is graphically seen that our proposed Hybrid Deep Neural Network (HDNN) model has got the best accuracy among all other algorithms used in this research.

Chapter 7

CONCLUSION & FUTURE SCOPES

7.1 Conclusion

Computer-Aided Diagnostics (CAD) has an extensive dominance in the pathological arena in order to detect the underlying health conditions precisely & accurately. However, to avail the benefits of the ideology "Prevention is better than cure," many biotech giants & medical research institutes have already been started working on early detection of diseases harnessing the early-stage biomarkers. Recent developments unleashed the technology of ECG & heart rate detection even in the smartwatch, which is being used daily; the accuracy & preciseness of the collected data & predicted result isn't appropriate for medical use, though. Nevertheless, these techs can give an alert ringing the bell in the early stage of cardiovascular diseases invoking the patient to consult physicians earlier before the most precious time is lost.

Keeping these potential scopes in mind, this research paper analyzed the MIT-BIH ECG dataset evaluating the performance of eleven different algorithms for multiclass (Five categories of ECG Signal) classification & finally, proposed a Hybrid Deep Neural Network (HDNN). After calculating the individual & overall accuracy, precision, recall & F-1 score from the confusion matrices of all the algorithms, we conclude that the proposed HDNN algorithm outperforms the other eleven algorithms & other pieces of literature we studied, giving overall accuracy of 99.23%. Thus, our proposed model can help doctors, diagnostic centers, and medical professionals to classify ECG signals quickly and so that it can prevent cardiovascular diseases early and efficiently bringing welfare to the human being.

7.2 Future Scopes

The proposed model holds the future scope of upgradation by incorporating few non-invasive features like family disease history, patient demographics & lifestyle; qualitative analysis of different parts (P, PR, QRS, ST, QT, T, U) of the beat; implementation of software or application for smartwatches & smartphones. Though the proposed model seems very efficient on the MIT-BIH data set, it might fluctuate slightly on the practical data set despite doing cross-validation. So, in the future, the proposed model can be trained with more practical data collecting from the hospitals and diagnostic centers so that this fluctuation is as less as possible. Also, training with regionally collected data will help to provide better results to the regional people.

References

1. J. Huang, B. Chen, B. Yao and W. He, "ECG Arrhythmia Classification Using STFT-Based Spectrogram and Convolutional Neural Network," in IEEE Access, vol. 7, pp. 92871-92880, 2019, doi: 10.1109/ACCESS.2019.2928017.
2. F. Liu, X. Zhou, J. Cao, Z. Wang, H. Wang and Y. Zhang, "A LSTM and CNN Based Assemble Neural Network Framework for Arrhythmias Classification," ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, United Kingdom, 2019, pp. 1303-1307, doi: 10.1109/ICASSP.2019.8682299
3. Wang, Pu & Hou, Borui & Shao, Siyu & Yan, Ruqiang. (2019). ECG Arrhythmias Detection Using Auxiliary Classifier Generative Adversarial Network and Residual Network. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2930882.)
4. Abdullah, A. (2014). ECG in Medical Practice (4th ed.). Jaypee Brothers Medical Pub.
5. ECG & Echo Waves. (2021, February 22). ECG interpretation: Characteristics of the normal ECG (P-wave, QRS complex, ST segment, T-wave) –. ECG & ECHO. https://ecgwaves.com/topic/ecg-normal-p-wave-qrs-complex-st-segment-t-wave-j-point/
6. Furst, J. (2017, February 13). Recording a 12 lead ECG/EKG. First Aid for Free. https://www.firstaidforfree.com/recording-a-12-lead-ecgekg/
7. Electrocardiography. (n.d.). MSD Manual Professional Edition. https://www.msdmanuals.com/professional/cardiovascular-disorders/cardiovascular-tests-and-procedures/electrocardiography#v931549
8. X. Zhang et al., "Classification of Arrhythmia Based on Extreme Learning Machine," 2018 10th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), Hangzhou, China, 2018, pp. 123-126, doi: 10.1109/IHMSC.2018.10135.
9. Kachuee, Mohammad & Fazeli, Shayan & Sarrafzadeh, Majid. (2018). ECG Heartbeat Classification: A Deep Transferable Representation.
10. Rana and K. K. Kim, "ECG Heartbeat Classification Using a Single Layer LSTM Model," *2019 International SoC Design Conference (ISOCC)*, Jeju, Korea (South), 2019, pp. 267-268, doi: 10.1109/ISOCC47750.2019.9027740

11. Alarsan, Fajr & Younes, Mamoon. (2019). Analysis and classification of heart diseases using heartbeat features and machine learning algorithms. Journal of Big Data. 6. 10.1186/s40537-019-0244-x.
12. S. Chakroborty and M. A. Patil, "Real-time arrhythmia classification for large databases," 2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Chicago, IL, USA, 2014, pp. 1448-1451, doi: 10.1109/EMBC.2014.6943873.
13. R. R. Janghel and S. k. Pandey, "Classification and Detection of Arrhythmia in ECG Signal Using Machine Learning Techniques," 2019 16th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), Pattaya, Thailand, 2019, pp. 101-104, doi: 10.1109/ECTI-CON47248.2019.8955208.
14. R. Banerjee, A. Ghose and K. Muthana Mandana, "A Hybrid CNN-LSTM Architecture for Detection of Coronary Artery Disease from ECG," 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 2020, pp. 1-8, doi: 10.1109/IJCNN48605.2020.9207044.
15. “Jupyter Notebook” (https://www.jupyter.org/)
16. “Anaconda Navigator” (https://docs.anaconda.com/anaconda/navigator/)
17. “Confusion Matrix” (https://en.wikipedia.org/wiki/Confusion_matrix/)
18. H. Dalianis, Clinical Text Mining: Secondary Use of Electronic Patient Records Springer Open, Cham Switzerland (2018), p. 47, 10.1007/978-3-319-78503-5
19. Duboue, Pablo. (2020). The Art of Feature Engineering: Essentials for Machine Learning. 10.1017/9781108671682.
20. A. (2020, October 5). 7 Feature Engineering Techniques in Machine Learning You Should Know. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/10/7-feature-engineering-techniques-machine-learning/
21. https://www.physionet.org/content/mitdb/1.0.0/
22. M. Kachuee, S. Fazeli and M. Sarrafzadeh, "ECG Heartbeat Classification: A Deep Transferable Representation," 2018 IEEE International Conference on Healthcare Informatics (ICHI), New York, NY, 2018, pp. 443-444, doi: 10.1109/ICHI.2018.00092.
23. Butcher, Brandon & Smith, Brian. (2020). Feature Engineering and Selection: A Practical Approach for Predictive Models: by Max Kuhn and Kjell Johnson. Boca Raton, FL: Chapman & Hall/CRC Press, 2019, xv + 297 pp., $79.95(H), ISBN: 978-1-

13-807922-9.. The American Statistician. 74. 308-309. 10.1080/00031305.2020.1790217.

24. Q.McCallum. Bad Data Hand Book: Cleaning Up The Data So You Can Get Back To Work. O'Reilly Media, 2013
25. Two Channel Histogram. (n.d.). Scientific Volume Imaging. https://svi.nl/TwoChannelHistogram/
26. https://towardsdatascience.com/deep-learning-unbalanced-training-data-solve-it-like-this-6c528e9efea6/
27. https://www.saedsayad.com/k_nearest_neighbors.htm/
28. https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_knn_algorithm_finding_nearest_neighbors.htm/
29. https://www.javatpoint.com/machine-learning-random-forest-algorithm/
30. https://en.wikipedia.org/wiki/Support-vector_machine/
31. https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm/
32. https://www.geeksforgeeks.org/ml-stochastic-gradient-descent-sgd/
33. https://en.wikipedia.org/wiki/AdaBoost/
34. https://www.mygreatlearning.com/blog/xgboost-algorithm/
35. https://www.programmersought.com/article/16143908973/
36. https://www.hackerearth.com/practice/machine-learning/machine-learning-algorithms/beginners-tutorial-on-xgboost-parameter-tuning-r/tutorial/
37. https://colah.github.io/posts/2015-08-Understanding-LSTMs/
38. https://en.wikipedia.org/wiki/Recurrent_neural_network/
39. https://en.wikipedia.org/wiki/Convolutional_neural_network/
40. https://blog.bayeslabs.co/2019/06/04/All-you-need-to-know-about-Vae.html/
41. https://ermongroup.github.io/cs228-notes/extras/vae/
42. https://en.wikipedia.org/wiki/Deep_belief_network/
43. https://missinglink.ai/guides/neural-network-concepts/deep-belief-networks-work-applications/
44. Asif, Md. Asfi-Ar-Raihan, Mirza Muntasir Nishat, Fahim Faisal, Rezuanur Rahman Dip, Mahmudul Hasan Udoy, Md. Fahim Shikder , and Ragib Ahsan. “Performance Evaluation and Comparative Analysis of Different Machine Learning Algorithms in Predicting Cardiovascular Disease.” *IAENG Engineering Letters*, 2021-In Press.
45. Géron, Aurélien. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. O'Reilly

Media, 2019.

46. Maxwell, A., Li, R., Yang, B. *et al.* Deep learning architectures for multi-label classification of intelligent health risk prediction. *BMC Bioinformatics* 18**,** 523 (2017). https://doi.org/10.1186/s12859-017-1898-z/
47. Floydhub: Practical Guide to Hyperparameters Optimization for Deep Learning Models (https://blog.floydhub.com/guide-to-hyperparameters-search-for-deep-learning-models/)