**ABSTRACT**

An electrocardiogram, or ECG, is a straightforward, non-invasive test used to identify abnormal conditions like irregular heartbeats. Notwithstanding the way that man-made reasoning and AI are used in a wide assortment of medical care related applications and datasets, various heartbeat classifiers utilizing profound and AI methods have been proposed lately. However, there are no widely explained public ECG datasets, and the datasets that are available for building and testing AI models frequently have very small sizes. A machine and deep transfer learning framework for classifying a small training dataset is presented in this paper. In order to predict the likelihood of developing a problem related to the heartbeat, a number of machine learning (ML) models were developed. Using a variety of physiological indicators and machine learning techniques like Decision Tree, Random Forest (RF) Classification, Support Vector Machine (SVM), and Naïve Bayes, this study trains four alternative models for accurate prediction. With an accuracy of almost 99 percent, the Random Forest method performed the best on this assignment. The open-access heartbeat categorization dataset was utilized in the development of the strategy. The fact that the models used in this study have a much higher percentage of accuracy than those used in previous studies indicates that these models are more reliable. The scheme can be found in the research's analysis, and numerous model comparisons have demonstrated their strength.

**Table of Content**

|  |  |  |
| --- | --- | --- |
|  |  | Page |
|  | **Chapter 1: Introduction**……………………………………………... | 11 |
| 1.1  1.2 | Introduction…………………………………………………………………………  Background………………………………………………………………………… | 12 |
| 1.3 | Problem Statement…………………………………………………………………. | 13 |
| 1.4  1.5 | Literature Review.…………………………………………………………………..  Project Goal………………………………………………………………………… | 13  14 |
| 1.6 | Summary…………………………………………………………………………… | 15 |
|  |  |  |
|  | **Chapter 2: Motivation**………………………………………………. | 16 |
| 2.1 | Introduction………………………………………………………………………… | 17 |
| 2.2 | Motivation towards our project……………………………………………………. | 17 |
| 2.3 | Summary…………………………………………………………………………… | 17 |
|  |  |  |
|  |  |  |
|  | **Chapter 3: Methodology** ………………………………………... | 18 |
| 3.1 | Introduction………………………………………………………………………… | 19 |
| 3.2 | Proposed System…………………………………………………………………… | 19 |
| 3.3 | Dataset…………………………………………………………………………… | 19 |
| 3.4 | Preprocessing …………………………………………………………………… | 20 |
| 3.5  3.6 | Algorithms………………………………………………………………………….  Summary……………………………………………………………………………  **Chapter 4: Technical Design** ………………………………………... | 21  27  28 |
| 4.1 | Introduction………………………………………………………………………… | 29 |
| 4.2 | Technical Design: System Level…………………………………………… | 29 |
| 4.3 | Summary…………………………………………………………………………… | 30 |
|  |  |  |
|  | **Chapter 5: Software Design**……………………………………... | 31 |
| 5.1 | Introduction……………………………………………………………… | 32 |
| 5.2 | Software Design: Module Level………….………………………………………… | 32 |
| 5.3 | Summary………………………………………………………………………….. | 33 |
|  |  |  |
|  | **Chapter 6: Skills**……………………………………………………... | 34 |
| 6.1 | Introduction……………………………………………………………… | 35 |
| 6.2 | Skills obtained……………………………………………………………………... | 35 |
| 6.3 | Summary………………………………………………………………… | 36 |
|  | **Chapter 7: Essential parts and Devices**…………………………….. | 37 |
| 7.1 | Introduction………………………………………………………………………… | 38 |
| 7.2 | Design Requirements………………………………………………………………. | 38 |
| 7.3 | Test Requirements………………………………………………………………… | 40 |
| 7.4 | Summary…………………………………………………………………………… | 40 |
|  |  |  |
|  | **Chapter 8: Working Sheets**…………………………………………. | 41 |
| 8.1 | Introduction………………………………………………………………………… | 42 |
| 8.2 | Work Breakdown Structure………………………………………………………… | 42 |
| 8.3 | Feasibility Study…………………………………………………………………… | 43 |
| 8.4 | Summary…………….…………………………………………………………… | 45 |
|  |  |  |
|  | **Chapter 9: Future Work**……………………………………………. | 46 |
| 9.1 | Introduction………………………………………………………………................ | 47 |
| 9.2 | Future scope of work………………………………………………………….......... | 47 |
| 9.3 | Summary…………………………………………………………............................ | 47 |
|  |  |  |
|  | **Chapter 10: Design Impact**………………………………………… | 48 |
| 10.1 | Introduction………………………………………………………………………… | 49 |
| 10.2 | Environmental Impact……………………………………………………………… | 49 |
| 10.3 | Economic Impact…………………………………………………………………… | 49 |
| 10.4 | Social Impact……………………………………………………………………….. | 49 |
| 10.5 | Sustainability……………………………………………………………………… | 49 |
| 10.6 | Summary…………………………………………………………………………… | 50 |
|  |  |  |
|  | **Chapter 11: Compliance with Standards**…………………………... | 51 |
| 11.1 | Introduction………………………………………………………………………… | 52 |
| 11.2 | Compliance with IEEE standard…………………………………………………… | 52 |
| 11.3 | Compliance with US standard……………………………………………………… | 52 |
| 11.4 | Summary…………………………………………………………………………… | 53 |
|  |  |  |
|  | **Chapter 12: Result**…………………………………………………… | 54 |
| 12.1 | Introduction………………………………………………………………………… | 55 |
| 12.2  12.3  12.4  12.5 | Result Analysis………………………………………………………………….  Data Visualization………………………………………………………………….  Evaluation of the Model……………………………………………………………  Model Comparison………………………………………………………………… | 55  55  56  63 |
| 12.6 | Summary………………………………………………………………………….. | 64 |
|  | **Chapter 13: Conclusion**……………………………………………… | 65 |
|  |  |  |
|  | **Bibliography**…………………………………………………………... | 67 |
|  | **Appendix: Software Listing**………………………………..………… | 70 |

**List of Figures**

|  |  |  |
| --- | --- | --- |
| Figure No. | Figure caption | Page No. |
|  |  |  |
| 1 | The block diagram of the proposed system | 19 |
| 2 | Statistical Description of the Dataset | 20 |
| 3 | Target Column Data | 21 |
| 4 | Block Diagram of Random Forest Classifier | 22 |
| 5 | Basic structure of Support Vector Machine | 23 |
| 6 | Block Diagram of voting Classifier | 24 |
| 7 | Block Diagram of Logistic Regression Classifier | 25 |
| 8 | Block diagram of Confusion Matrix | 26 |
| 9 | Block Diagram of System Flow | 29 |
| 10 | CNN System Architecture | 32 |
| 11 | Work Breakdown Structure | 42 |
| 12 | Gantt chart | 43 |
| 13 | Visualization of some features of the dataset | 25 |
| 14 | Classification result of Random Forest | 56 |
| 15 | Confusion Matrix of Random Forest | 57 |
| 16 | Classification Report of SVM | 58 |
| 17 | Confusion Matrix of SVM Classifier | 58 |
| 18 | Classification Report of Voting Classifier | 59 |
| 19 | Confusion Matrix of Voting Classifier | 60 |
| 20 | Classification Report of LR Classifier | 61 |
| 21 | Confusion Matrix of Logistic Regression | 61 |
| 22 | CNN Classifier accuracy | 62 |
| 23 | CNN Confusion Matrix | 63 |

**List of Tables**

|  |  |  |
| --- | --- | --- |
| Table No. | Table caption | Page No. |
|  |  |  |
| 1 | Algorithm Performance Comparison | 63 |

**CHAPTER 1**

**Introduction**

**1.1 Introduction**

Our project can detect heartbeat categorizations using an ECG dataset and deep learning and machine learning methods. It will add significant value for all types of people and modern medical science, and it also has some positive effects on socio-economics. Our system will be simple to use, and many people with non-technical backgrounds will be able to use it.

**1.2 Background**

Cardiologists and other medical professionals frequently use electrocardiograms (ECG) to monitor cardiac health. Similar to many other time-series data, the difficulty of identifying and classifying various waveforms and morphologies in the signal is the primary issue with manual ECG signal analysis. This is a task that takes a lot of time and is easy to make mistakes for a human. Due to the fact that cardiovascular diseases account for approximately one third of all deaths worldwide, accurate diagnosis is critical [1]. For example, a large number of individuals experience sporadic pulses which can be deadly at times. As a result, it is highly desirable to have a diagnosis of arrhythmias that is both accurate and affordable [2]. Numerous studies in the literature looked into using machine learning techniques to accurately detect anomalies in the signal in order to address the issues raised by manual ECG signal analysis [3, 4]. The majority of these strategies involve a phase of preprocessing to prepare the signal (such as running it through band-pass filters, for example). The handcrafted features, which are primarily statistical summaries of signal windows, are then extracted from these signals and utilized in subsequent classification analysis. Support Vector Machines, multi-layer perceptron, decision trees, and other conventional machine learning approaches are utilized for ECG analysis's inference engine. 5], [6], [7]. The most common symptoms are: seizures, headaches, vision issues, mental shifts, vomiting, and other symptoms other symptoms include difficulty speaking or moving, diminished sensations, and occasionally a loss of consciousness [4-5]. People frequently ignore these symptoms, which leads to an increase in their classification and worsens their condition. Gradually, the classification begins to change into heartbeat disease, which can even kill you. Therefore, a categorization of a heartbeat ought to be found early on. Numerous lives could be saved with the proper treatment of the disease. Adult heartbeat classifications are typically benign (non-diseases). However, malignant classifications are frequently observed in children. A CT (Computed Tomography) scan and an MRI (Magnetic Resonance Imaging) scan are the medical tests that are most frequently suggested for the purpose of determining the classifications of heartbeats. A CT examine is essentially a X-beam of the head. Because they harm the cells that make up our heartbeats, X-rays are known to be harmful to our bodies. Therefore, MRI is the most secure diagnosis and better at detecting categorizations than the previous tests. A heartbeat categorization cell and a normal cell can also be distinguished more clearly using deep learning techniques on MR images. In addition, this may speed up the process of categorizing heartbeats and identifying them while maintaining higher precision.

**1.3 Problem Statement**

We are aware that cardiovascular disease is currently the leading cause of death in both men and women. With ECG Heartbeat Categorization, the ECG can identify the state of the heart. Due to irregular heartbeats, many people nearby experience breathing difficulties and chest pain. In our project, we will assess the activity and health of the human heart and determine if it is normal or unhealthy based on signals obtained from the human body. An ECG signal shows whether a patient has heart disease (called an abnormal signal) or doesn’t have heart disease (called a normal signal). Besides, it can also detect if the heartbeat is missing at any time and identify the weaknesses of the existing ECG and eliminate them. The major problem when we try to create an ECG circuit is that the baseline is not stable, thus creating noise. The correct method to amplify and filter the signal was discovered to rectify this situation. Therefore, it is very important to recognize the signs of a heart attack and immediately seek medical attention. The ECG is an important device, yet it still produces noise that can distort the original waveform, leading to an inaccurate signal waveform. This inaccurate signal is not good for the analysis of the patient’s condition, where it may indicate the patient has a slightly better or worse condition.

**1.3 Literature Review**

Basically, deep learning (DL) is a type of machine learning (ML) that uses AI to work with images. By distinguishing between images that a regular human cannot detect using standard methods, people can gain certain knowledge in this way. By combining a variety of layers of linear process parameters, the DL algorithms extract features. Data extraction becomes easier as we move deeper into the network because the result of each layer becomes the source for the next layer. A type of deep learning known as deep convolutional neural networks (CNNs) is designed to be easy to use and is frequently utilized in visual image analysis. It is based on the biological activities of the human heartbeat and has been used to order data in various arrays [6]. Over the years, a variety of machine learning algorithms have been used to identify heartbeat categorization. Research [7] was directed utilizing AI calculations, Multi-facet Perceptron (MLP) and Innocent Bayes to identify the heartbeat categorization from MR pictures, weighted-F1 scores were around 98.6% and 91.6% individually. In [8], the heartbeat classification was detected with a precision of 91% and 95%, respectively, using the C4.5 algorithm and MLP. With the help of a Deep Neural Network and a Machine Learning Algorithm, the researchers in the study [9] discovered an effective classification of heartbeats. They achieved incremental accuracies of 98.67%, 97.34%, and 94.24% by utilizing Softmax's Fully Connected layer, CNN with the Radial Basis Function (RBF) classifier, and the Decision Tree (DT) classifier. In [10], the authors looked at various approaches taken by other authors and discovered that MRI images were more effective at identifying categorizations of heartbeats. The creators in [11] zeroed in on convolutional brain organizations (CNN) for the location of heartbeat categorization MR pictures. Using the CNN algorithm, they were able to get a weighted-F1 score of around 94 percent. In [12], CNN and VGG-16 were used to identify categorizations of heartbeats with 91.6 percent and 91.9 percent, respectively. The K-NN classification algorithm for categorizing heartbeats, which had a weighted-F1 score of 86%, was the subject of the study in [13]. Using UTSU's method, MATLAB was utilized to detect images for heartbeat categorization detection in [14]. The level of accuracy was 95 percent. CNN was utilized for distinguishing heartbeat categorizations in [15], and had a general precision of 97.87%. Deep transfer learning is used in this study to demonstrate a method for categorizing heartbeats from MR images. The most common diagnosis at hand is an MRI, taking into account all of the effects of a CT scan. This paper will probably give a productive framework that recognizes heartbeat categorizations at a beginning phase with minimal expense and high effectiveness, which will decrease passings and experiencing because of heartbeat categorizations. The deep learning (DL) process is crucial to our ability to distinguish between cells that are affected by heartbeat categorization and cells that are normal heartbeat cells. Furthermore, deep convolutional neural networks (CNN) are extremely useful for the study because the differences between the images are so small that traditional methods are unable to detect them. Most studies had an accuracy rate of 90–95 percent, which was considered excellent. However, the unique aspect of our study is that we employed a combination of well-known machine learning techniques to get the greatest results. With 97, 98, 97, and 96% F1-scores, the most effective algorithms were Random Forest (RF), Support Vector Machine (SVM), Voting Classifier (VC), and Logistic Regression (LR). The accuracy percentage of the models employed in this study is much higher than that of earlier studies, indicating that the models utilized in this study are more reliable. They've been proved to hold up in a variety of model comparisons, and the system might be derived from the study's findings.

**1.3 Project Goals**

There are some major goals that we want to focus on with our project. The main goals are treating heart disease to cure people of the heart. Protecting kidney function is especially important for patients who has age more than fifty. For some patients, surgery will never be needed. By using this it can be highly ensured that the detection of the disease will be well planned and it would yield efficient results in the years to come. The system has been built in such a way that is it easy to use and it will work properly without any complications. So, it can be said that the main goal of our project is to detect Kidney Cancerous Tumor using Deep Learning Technique.

**1.4 Summary**

This chapter provided us with knowledge about the modules that are included in the suggested system. With the aid of the project goals being elaborated, this chapter gave readers a clear image of how this system uses Deep Learning and Machine Learning to effectively detect heart problems.

**CHAPTER 2**

**Motivation**

**2.1 Introduction**

In this chapter, we discuss why we considered implementing this system. This chapter will also cover the reasons why we decided to work in this area rather than any other field.

**2.2 Motivation towards our project**

Health problems in the human race are becoming more prevalent by the day, and cardiac problems are one of the most common diseases that have been observed in recent decades. Therefore, machine learning has played an important role in the area of computer-aided diagnosis (CAD). From the above discussion, we expected that various algorithms or methods have performed pretty well in the field of cardiovascular disease detection. This indicates that machine learning in cardiac signal processing has an unbounded scope in the research field for enhancing CAD and getting more accurate, cost-effective, and fast output. For now, we understand the strengths and weaknesses of the system in general, give an idea of how the system will behave in the future, and perhaps teach how to correct the system's mistakes Now, when introducing a new system, people think about timing, cost, and dependability. Taking everything into account, we created a system that will automate a business and be effective, affordable, user-friendly, and reliable.

**2.3 Summary**

This chapter provided the idea about the motivation towards our project which aims to systematically automate the entire work structure of a medical health care.

**CHAPTER 3**

**Methodology**

**3.1 Introduction**

Dataset description, block diagram, flow diagram, and evaluation matrices are included in this part, as well as the study's procedure and methodology.

**3.2 Proposed System**

The block diagram of the proposed system is shown in Figure 1.

Diagram

Description automatically generated

After the data have been analyzed, they can be used to build models. A preprocessed dataset and machine learning algorithms are required for model creation. LR, SVM Classification, RF Classification, and a Voting Classifier are some of the methods used. Using the accuracy metrics Accuracy Score, Precision Score, Recall Score, and F1 Score, four alternative models are compared. The following subsections go over each one of the block diagram's components.

**3.3 Dataset**

The MIT-BIH Arrhythmia Dataset and The PTB Analytic ECG Data set, two notable heartbeat grouping datasets, contain the two assortments of heartbeat signals in this dataset [17]. A profound brain organization can be prepared involving the huge number of tests in the two assortments. This dataset was used to investigate heartbeat classification and observe some transfer learning capabilities by utilizing architectures of deep neural networks. The shapes of the heartbeats on the electrocardiogram (ECG) match the signals in both cases of normal heartbeat and cases of various arrhythmias and myocardial infarction. These signals have been preprocessed so that each segment can be compared to a heartbeat.

Table

Description automatically generated

**Figure 2:** Statistical Description of the Dataset

On the other hand, the fact that some of the values in the dataset are missing indicates that some questions have been left blank due to privacy concerns raised by several patients. So, this study only looks at things that are easy to get to and have a less than 15% missing value for detecting heartbeat problems.

**3.4 Preprocessing**

The current heartbeat problem dataset will have many imbalances and missing values fixed by this step. Due to privacy concerns, many patients did not respond to questions, resulting in this lack of data. As a result, between 12 and 14 percent of all attributes lack data. The sample mean and mode were used as a data imputation strategy to fill in the values that were missing from the impacted characteristics. In addition, the dataset was subjected to the synthetic minority oversampling method (SMOTE) following the correction of missing data in order to correct the dataset's imbalance and generate new instances of the under sampled target class. After preprocessing, the data for the target column are shown in Figure 3.

A picture containing logo

Description automatically generated

**Figure 3:** Target Column Data

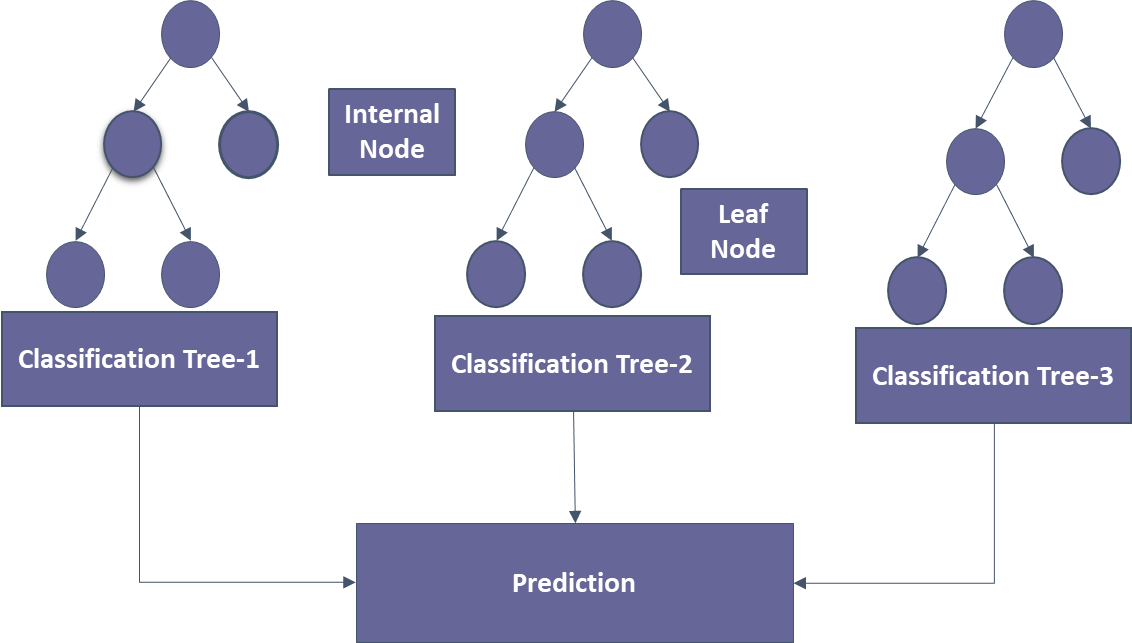
Following data preparation and control of the unbalanced dataset, the model's design is the next step. In order to make this work more accurate and efficient, the data are divided into training and testing segments in an 80/20 ratio. After the model splits, a variety of classification algorithms are used to train it. This work employs Logistic Regression, Support Vector Machine, Random Forest, and Voting Classifier as classification techniques.

**3.5 Algorithms**

This part of the process uses algorithms like Random Forest Decision Tree (RF), Support Vector Machine (SVM), Logistic Regression (LR), and Voting Classifier (VC) to classify the data. A confusion matrix is used to figure out how well the algorithms did in terms of accuracy, precision, and recall.

**3.5.1 Random Forest**

Relapse and characterization related issues can be settled with the assistance of the irregular backwoods’ calculation, a managed learning strategy. Due to its adaptability and ease of use, it is one of the machine learning algorithms that is used the most. Overfitting, which could be a major issue with such a complex algorithm, is reduced and accuracy is improved through the use of randomization. These methods construct decision trees from randomly selected data samples and derive predictions from each tree. After that, they vote on the option that is best for them. It is utilized, among other everyday tasks, in feature selectors, recommender systems, and image classifiers. Misrepresentation location, advance application order, and sickness forecast are just a few of its legitimate uses. The Boruta algorithm, which is used to find important data points, is based on it. Figure 4 depicts a block diagram of random forest categorization.

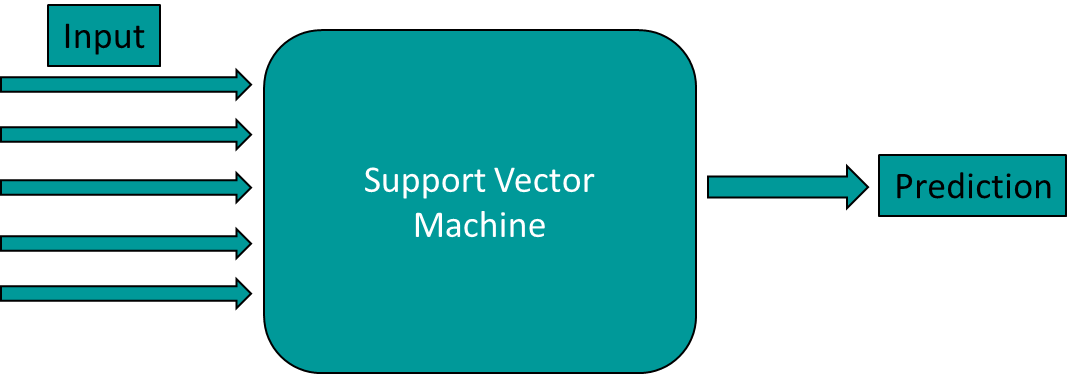


**Figure 4**: Block Diagram of Random Forest Classifier

The majority of non-linear classifiers fall short of the performance of the random forest method. Additionally, the fact that it draws its conclusion from a number of decision trees makes this approach quite resilient. Since it averages all predictions, the random forest classifier prevents overfitting by eliminating biases and resolving the overfitting issue. Missing values are not a problem for random forests. They could address this problem by either calculating the proximity-weighted average of the missing data or substituting median values for continuous variables. By comparing the importance of each characteristic, you can select the most crucial ones for your classifier.

**3.5.2 Support Vector Machine**

In N-dimensional space, where n is the number of features in a dataset, an SVM is a discriminative classifier that creates hyperplanes. Future data inputs can be distinguished with the help of these hyperplanes. The boundaries that aid in the classification of data points are called hyperplanes, decision boundaries, or decision planes. The side of the hyperplane where a new data point lands may belong to more than one class. The number of features assigned to a dataset is inversely proportional to the hyperplane's dimension. The hyperplane might be a straight line if the dataset only has two features. The hyperplane is a two-dimensional plane when there are three distinct characteristics in a dataset. The data points closest to the hyperplane that have an effect on its location are called support vectors. The term "Support Vector Machine Algorithm" comes from the fact that these vectors have an effect on where the hyperplane is positioned. Figure 5 depicts a block diagram of SVM categorization.

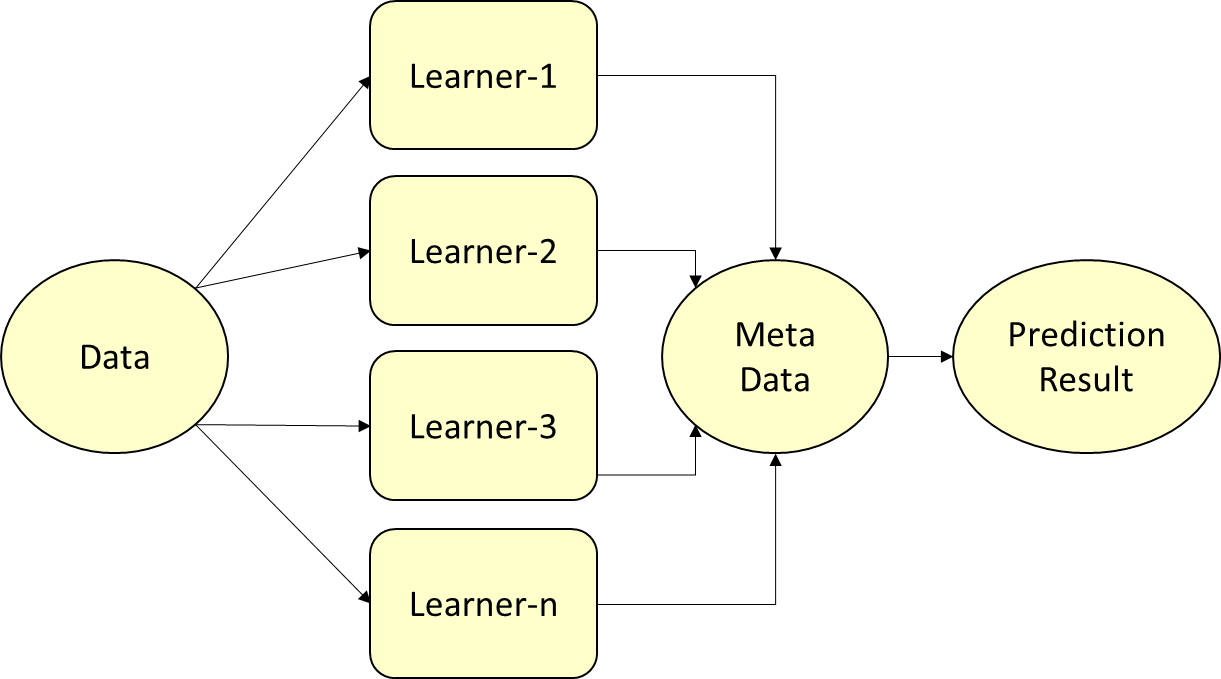


**Figure 5**: Basic structure of Support Vector Machine

Regression and classification problems can all be solved with SVM algorithms. It works well for getting non-linear data out of linear data. SVMs should not be used with large datasets. The task of selecting the best kernel for SVM is challenging. SVM also performs poorly when the number of features in each data set is less than the number of training data samples. The level of classification accuracy that the SVM algorithm must achieve is indicated by the C parameter. In a nutshell, the C parameter tells your model how much to penalize for each misclassified point on a particular curve. The model has an easier time correctly classifying all of its training examples when its C is low. The model is able to select a greater number of samples as support vectors thanks to its high C, which aids in the classification of all of its training examples.

**3.5.3 Voting Classifier**

A voting classifier is a type of machine learning model that predicts an output (class) based on the class with the greatest likelihood of being chosen as the output and trains on an ensemble of multiple models. It forecasts the output class based on the class with the largest voting majority by simply combining the results of each classifier that are fed into the voting classifier. We create a single model that trains on these models and predicts output based on the overall majority of votes for each output class rather than creating discrete models for each output class and determining their accuracy. The block diagram for the voting classifier model is shown in Figure 6.



**Figure 6**: Block Diagram of Voting Classifier

The mechanism we will use to compare alternative training models is summarized by voting. There are two ways to vote:

* Soft Voting: For each model, the expected probability slopes are added and averaged in this step. The content of the category with the highest value is printed and declared the winner. Even though this seems like a good strategy, it should only be used if each subcategory is calibrated correctly. With the exception of the fact that the various models each contribute in a different way to the final output vector, this is similar to determining the weighted average of a group of integers.
* Hard Voting: The mode value of the final output is determined by combining the outputs of the various categorization models in this step. This method is comparable to finding the arithmetic mean of a set of integers because it ignores the specific probability values associated with each model. Only the output of each model is taken into consideration when evaluating it.

**3.5.4 Logistic Regression**

A discriminative model based on binary classification is the logistic regression classifier. The logistic sigmoid function that was utilized in the prediction model is referred to as "logistic regression." Additionally, the sigmoid function is frequently utilized as an activation function in neural networks. The Logistic Regression model's logic flowchart can be seen in Figure 7. In the context of supervised learning, LR is one of the machine learning algorithms that is utilized the most frequently. It is a modeling technique that uses a set of independent variables to forecast a categorical response variable.

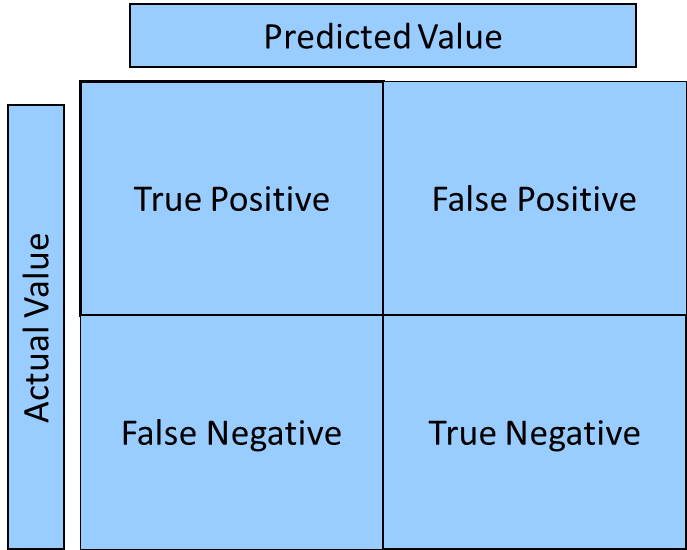


**Figure 7**: Block Diagram of Logistic Regression Classifier

When predicting the outcome of a categorical dependent variable, logistic regression is utilized. The output format must therefore be discrete or categorical. It could be true or false, yes or no, 0 or 1, true or false, or any other number between 0 and 1, but only probabilities between 0 and 1 are specified. Linear regression and logistic regression share a lot of similarities. Linear Regression is used to solve regression issues, whereas Logistic Regression is used to solve classification issues. We use a logistic function with an "S" shape that predicts two maximum values instead of a regression line.

**3.5.5 Evaluation Matrix**

Figure 8 depicts the evaluation matrix, also known as the confusion matrix. A metric for evaluating the efficiency of machine learning algorithms is the confusion matrix. Each model's efficacy was evaluated with the help of the confusion matrix. The confusion matrix depicts the frequency with which our models correctly predict and the frequency with which they underestimate. False positives and false negatives have been ascribed to values that were incorrectly predicted, whilst true positives and true negatives have been given to values that were correctly predicted. After grouping all the predicted values into a matrix, the accuracy, precision-recall trade-off, and AUC of the model were used to see how well it worked.



**Figure 8**: Block diagram of the confusion matrix

The classifier's errors and the kinds of errors they make are detailed in the confusion matrix. While making predictions, it demonstrates how disorganized and confusing a categorization model is. The limitations imposed by relying solely on categorization accuracy can be circumvented with the assistance of this feature. It is used when one class dominates over others and the categorization problem is extremely unbalanced. Recall, precision, specificity, accuracy, and the AUC-ROC Curve can all be calculated using the confusion matrix in an extremely efficient manner.

**3.6 Summary**

Considering all aspects, our proposed solution is an optimal one. The project is usable in a reasonable cost. It is reasonable because we are using numerous customs designed and self-programmed frameworks that are going to help us to develop our project. The system interface is easy to operate and can be maintained by a non-IT professional as well. This system has been developed in such a way that it gives a feeling of user friendliness and there will not arise any issues with regards to complications. Therefore, this chapter gives the idea about the current systems that are available in the market and about the motivation towards developing this dynamic project.

.

**CHAPTER 4**

**TECHNICAL DESIGN**

**4.1 Introduction**

We go over the technical design of our system in this chapter. It would be simpler to conceptualize the entire system's data flow by going through the system level design.

**4.2 Technical Design: System Level**

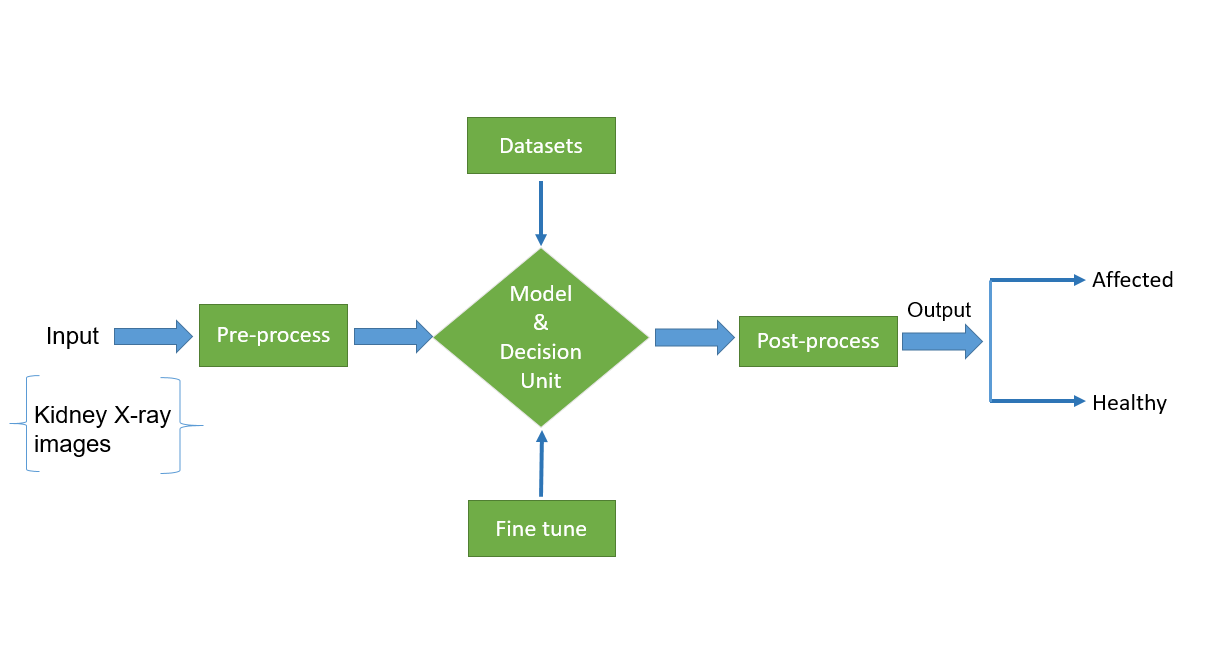
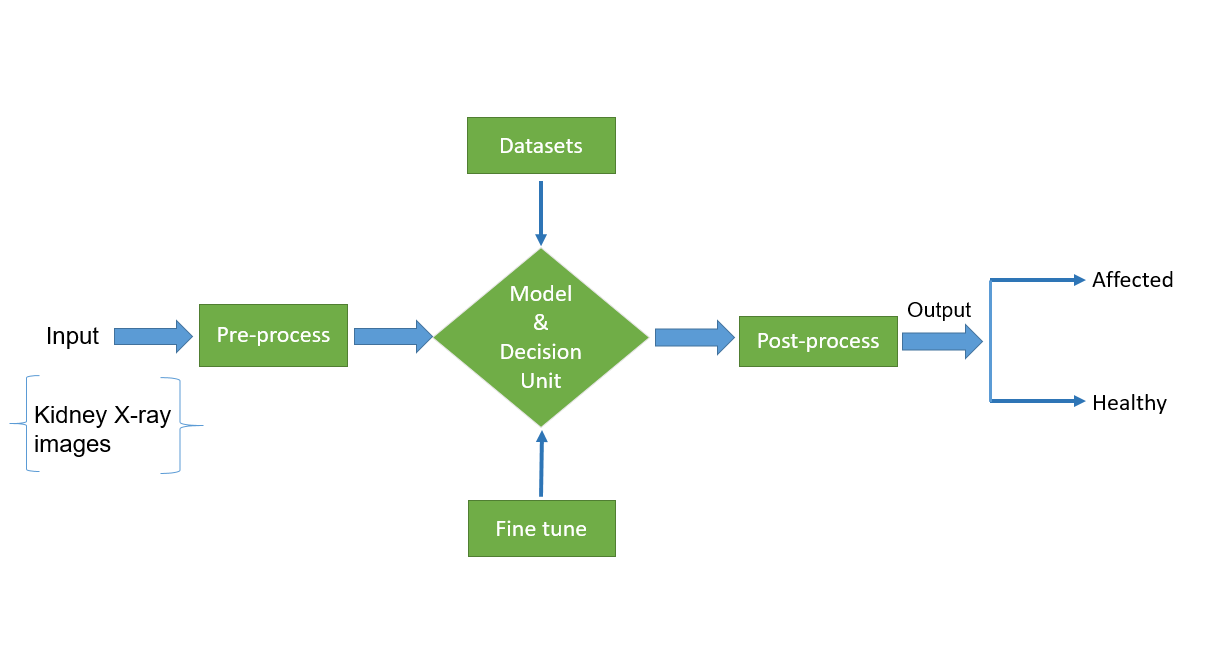


Figure 9: Block Diagram

**4.3 Summary**

As was already noted, the technical design has given us a clear understanding of how our system is functioning. We can therefore understand how our system operates by taking into consideration the data flow diagram above.

**CHAPTER 5**

**SOFTWARE DESIGN**

**5.1 Introduction**

In this chapter we discuss the software design of our system. One can consider this part as the essential structure of the system as it will provide as the necessary features that our system entails.

**5.2 Software Design: Module Level**

Diagram

Description automatically generated

Figure 10: CNN System architecture

The framework engineering provides you with an outline of the entire thing. The information is a CTpicture, and the result is an expectation of the picture in this engineering. It will decide whether the image is affected in this present circumstance. There are three channels and the info shape is 224x224. The channel size is 32 with cushioning, the portion size is 3, and the enactment capability is ReLU in the principal layer of the proposed engineering. The main max-pooling layer follows, which has a pool size of 2 and steps of 1. The following layer is a level layer that joins each of the pooled highlights into one segment. Two thick layers were created eventually. The initiation capability for the primary layer is ReLU, though the enactment capability for the most un-thick layer is soft max. The features are added to the network once they have been preprocessed. Figure 1 illustrates the architecture.

**5.3 Summary**

This chapter provided all the necessary details on the software design framework of our system.

**CHAPTER 6**

**SKILLS**

**6.1 Introduction**

In this chapter we discuss the skills that we have obtained in order to develop this massive sophisticated system.

**6.2 Skills obtained**

Through this project the following skills have been developed:

* **Skill in Programming & Tools**
* **Anaconda Navigator**

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows you to launch applications and easily manage anaconda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository. It is available for Windows, macOS, and Linux.

* **Python 3**

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. [30] Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library. [31] [32] Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. [33] Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a cycle-detecting garbage collection system (in addition to reference counting). Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible. Python 2 was discontinued with version 2.7.18 in 2020. [34]

**6.3 Summary**

In this chapter we discussed the list of skills that have been obtained throughout the process of developing and materializing this system.

**CHAPTER 7**

**ESSENTIAL PARTS AND DEVICES**

**7.1 Introduction**

In this chapter, we shed light on the tools that we used to develop this sophisticated system and we also discuss what tools will be required if one wants to test this system in there one spheres.

**7.2 Design Requirements**

The For designing the project we needs to have a good software that not only makes the project user-friendly but also makes it easier to detect the kidney cancerous tumor. Similarly, the project must be flexible enough to adjust to any structural changes. We used the following tools/materials for our system:

Python is an undeniable level programming language with a large number of utilizations, including information examination and the formation of profound learning calculations. Python libraries are utilized to make AI, information science, information perception, picture and information control, and numerous different applications. In view of Python's broad library access, profound learning-based issues are especially effective with Python programming. For dataset preprocessing, Boa constrictor Guide and Jupiter Journal were utilized. As well as, for dealing with the huge datasets and preparing the model, Google Collab was utilized. Every one of the codes, information, charts, and works were saved to the neighborhood machine as well as online GitHub with the goal that it tends to be recovered whenever utilizing any GPU.

**1. Model:**

* VGG16
* MobileNetV2
* CNN
* InceptionV3
* Resnet50

**2. Dataset:**

* Kaggle

**3. Framework:**

In general, a framework is a real or conceptual structure intended to serve as a support or guide for the building of something that expands the structure into something useful.

In computer systems, a framework is often a layered structure indicating what kind of programs can or should be built and how they would interrelate. Some computer system frameworks also include actual programs, specify programming interfaces, or offer programming tools for using the frameworks. A framework may be for a set of functions within a system and how they interrelate; the layers of an operating system; the layers of an application sub-system; how communication should be standardized at some level of a network; and so forth. A framework is generally more comprehensive than a [protocol](http://searchnetworking.techtarget.com/definition/protocol) and more prescriptive than a [structure](http://searchnetworking.techtarget.com/definition/protocol). [5]

**4. Software:**

* + GitHub
  + Jupitar Notebook
  + Google Colab

**5. Operating System:**

* Win-XP, Win-7, Win-8 or higher version
* Linux or any other higher version.

**6. Language:**

* Python

**7. Memory Requirements:**

For a small diagnostic center, it would require 5-10MB per day to store the information for an average of 50-100 patients. But the main concern is to take everyday data and coding it in order to back up the sensitive information to overcome the unexpected mishaps. So it requires total 100-120 MB memory space per day. A memory of 1TB+ hard disk can take one-year data backup.

**7.3 Test Requirements**

In order to test the project, the following tools are necessary:

**1. Internet Connectivity**

**2. Domain/Server**

**3. Hardware:**

* + Processor (Core i3, Core i5 or higher)
  + RAM (4 GB or Higher)
  + Disk Space (500GB or Higher)

**4. Operating System:**

* Win-XP, Win-7, Win-8 or higher version
* Linux or any other higher version

**7.4 Summary**

In this chapter we described what tools we have taken benefit of in order to build this sophisticated system. Furthermore, we have also learnt what tools we need in order to conduct a thorough testing of this system from both the user and the admin’s perspective.

**CHAPTER 8**

**WORKING SHEETS**

**8.1 Introduction**

In this chapter, we observe the entire work structure, meaning how the scheduling was maintained throughout the developmental phase. We shall also see the financial foundation of this project and furthermore the feasibility study should be also discussed.

**8.2 Work Breakdown Structure**

We have divided the work breakdown structure into two parts. In order to develop this system, we gave ample importance to scheduling because we believed if we want to provide the best of quality then we must give due importance to scheduling which helped us to garner better results. The figure below focuses the weekly work we had accomplished.

|  |  |
| --- | --- |
| **Week No.** | **Proposed Work** |
| Week-1 | Project Proposal Report and Presentation |
| Week-2 | Collecting Dataset |
| Week-3 | Reading the existing paper |
| Week-4 | Start preprocessing data |
| Week-5 | Feature selection of the data |
| Week-6 | Exploratory data analysis and preprocessing done |
| Week-7 | Model implementation |
| Week-8 | Report Generator |
| Week-9 | Technical and software designing |
| Week-10 | Design results and analysis |
| Week-11 | Testing & problem solving |
| Week-12 | Final Report and Presentation |

Figure 11: Work Breakdown Structure

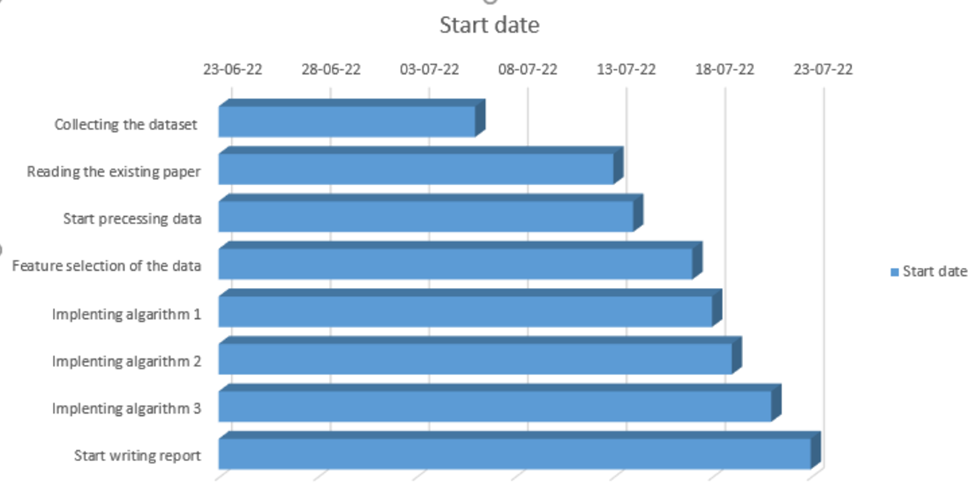


Figure 12: Work breakdown structure illustrated in Gantt chart

**8.3 Feasibility Study**

Depending on the results of the initial investigation the survey is now expanded to a more detailed feasibility study. “**FEASIBILITY STUDY**” is a test of system proposal according to its workability, impact of the organization, ability to meet needs and effective use of the resources. It focuses on these major questions:

1. What are the user’s demonstrable needs and how does a candidate System meets them?

2. What resources are available for given candidate system?

3. What are the likely impacts of the candidate system on the organization?

4. Whether it is worth to solve the problem?

During feasibility analysis for this project, following primary areas of interest are to be considered. Investigation and generating ideas about a new system does the following steps:

**Steps in feasibility analysis**

1. Form a project team and appoint a project leader.

2. Enumerate potential proposed system.

3. Define and identify characteristics of proposed system.

4. Determine and evaluate performance and cost effectively of each proposed system.

5. Weight system performance and cost data.

6. Select the best-proposed system.

7. Prepare and report final project directive to management.

**Technical feasibility**

A study of available resource that may affect the ability to achieve an acceptable system. This evaluation determines whether the technology needed for the proposed system is available or not.

* Can the work for the project be done with current equipment existing software technology & available personal?
* Can the system be upgraded if developed?
* If new technology is needed then what can be developed?

This is concerned with specifying equipment and software that will successfully satisfy the user requirement. The technical needs of the system may include:

**Economic feasibility**

Economic justification is generally the “Bottom Line” consideration for most systems. Economic justification includes a broad range of concerns that includes cost benefit analysis. In this we weight the cost and the benefits associated with the candidate system and if it suits the basic purpose of the organization i.e. profit making, the project is making to the analysis and design phase.

The financial and the economic questions during the preliminary investigation are verified to estimate the following:

* The cost to conduct a full system investigation.
* The cost of hardware and software for the class of application being considered.
* The benefits in the form of reduced cost.
* The proposed system will give the minute information, as a result the performance is improved which in turn may be expected to provide increased profits.
* This feasibility checks whether the system can be developed with the available funds.

Our system does not require enormous amount of money to be developed. This can be done economically if planned judicially, so it is economically feasible. The cost of project depends upon the number of man-hours required.

**8.4 Summary**

To conclude, we discussed the scheduling processes of developing this system. Additionally, we have also identified how feasible the system is through the lens of evaluating using various feasibility studies.

**CHAPTER 9**

**FUTURE WORK**

**9.1 Introduction**

This chapter discusses the future scope or the implementation of this system. As our system is a web-based, various forms of new features can be incorporated to this system as per the requirements.

**9.2 Future Scope of Work**

The main objective of developing this system is to provide a basic platform for startup clinics and diagnostic centers. The system can be more improved than the current form. In the context of securing this system further work can be done by including firewalls and other means of securing this system. Additionally, the interface of the system can be a work in progress to look more user-friendly. The sign up section can be updated by including the popular social media sites. When a user wants to sign up for an account it can be made easier by incorporating the mainstream social media sites such as signing with one’s Facebook or Twitter profile. The help tab can be further enhanced through a virtual tour inside the diagnostic center with the location of important junctions such as the chamber room of a particular doctor or a lab test room.

**9.3 Summary**

This chapter has described the possible future applications of the design. But there are a lot of possibilities with the designed system. The system may need some research for different applications, though the principle of the designed system will remain as it is.

**CHAPTER 10**

**DESIGN IMPACT**

**10.1 Introduction**

In this chapter, we discuss about the various impacts that our system has been able to generate.

**10.2 Environmental Impact**

Large amounts of paper can be saved by implementing this system at a diagnostic facility because all duties will be computerized and as a result, the daily amount of paper wasted in official activities will be much reduced. Software and computational data are other project components. There won't be much of an environmental impact as a result.

**10.3 Economic Impact**

Because the system is automated and takes less human labor, the economic impact of using it in an organization is that employee salaries can be decreased dramatically because fewer people are needed to complete a given activity.

**10.4 Social Impact**

People can easily detect irregular heartbeats since they can check their heart rate. As a result, people can quickly identify their illness. They can recover more quickly if they receive early treatment for their illness. It will help people to prevent the disease at an early stage.

**10.5 Sustainability**

Any of our models won't need to be adjusted in the future. Therefore, our project is sustainable.  This project is software-based, therefore if the software changes in the future, it will update automatically. As a result, we do not need to think about the project's long-term viability.

.

**10.6 Summary**

This chapter has covered the different types of impacts that our system offers and those has been described and discussed. From the above given impacts, we can conclude that our designed system is good enough to use under any circumstance.

**CHAPTER 11**

**COMPLIANCE WITH IEEE STANDARDS**

**11.1 Introduction**

In this section we discuss about the consistence of our task with diverse standards. There are a few distinct standards, amongst which the IEEE standards, US standards and European standards are talked about in this part.

**11.2 Compliance with IEEE standard**

There are a few distinct guidelines put forward by IEEE Stanards affiliation. The majority of them however are not material for our framework. We have included idea of operation as for the IEEE standard. A conference paper has additionally been submitted and affirmed by IEEE standards entitled "MediPro – A Cost Efficient and User Friendly Medical Information System" that points of interest out our work on this task.

**11.3. Compliance with US standard**

ANSI recommends that copyrighted software should only be included for informational purposes, or in forms which do not mandate particular implementations of the standard. Object code should never be included in a standard as a normative requirement. While ANSI opposes use of software standards to mandate particular implementations and believes that use of software in standards should be avoided to the extent possible, ANSI recognizes that there may be circumstances in which inclusion of some software, provided it is accompanied by adequate legal permissions, may facilitate development of multiple, competing and interoperable implementations of the standard. Examples of such software could include: ·

* Pseudo Code (code that is human readable and similar to programming languages but cannot be directly processed or compiled directly to be processed by hardware that manipulates data according to instructions);
* Schema examples;
* Data structure definitions;
* ASN.1 structure definitions;
* ABNF grammar specifications;
* Example programming instructions that are sufficiently limited in scope that they do not, either singularly or in the aggregate, perform a complete or a substantial part of a function and are illustrative, at most, of limited sections of an independent fully described specification; or
* Sample programming instructions provided solely for conformance testing purposes.

Our project has been established based on the above ANSI principles and it completely relies upon it.

**11.4 Summary**

In this section we have examined the different compliant standards and made sure that we are in accordance with. These standards have been put without hesitation so as to control things, guarantee well-being and ensure there are no well-being dangers to the use of distinctive segments. It is imperatively essential to maintain these measures and we have done as such over the span of our task work.

**CHAPTER 12**

**RESULTS**

**12.1 Introduction**

This chapter of the report contains the results that we achieved throughout the course of using this system.

**12.2 Result Analysis**

The experiments' objective is to determine the optimal classification system for heatbeat problem diagnosis. The performance of random forest, logistic regression, voting classifier, and support vector machines is examined and compared in terms of accuracy, precision, and recall using a heartbeat problem risk factor dataset and classification in machine learning.

**12.3 Data Visualization**

The graphical representation of a frequency distribution with categorized continuous classes is known as a data visualization. Figure 9 shows the visualization of the heartbeat problem dataset.

Diagram

Description automatically generated

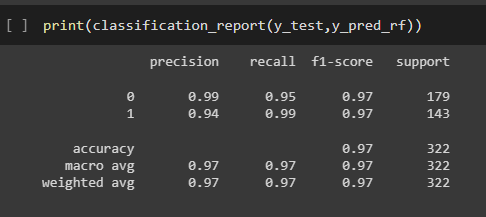
**Figure 13**: Visualization of some features of the dataset

It is known as an area diagram, and its definition states that it is a collection of rectangles with areas that correspond to frequencies in the associated classes and bases and intervals between class borders. In such representations, each rectangle is contiguous because the base spans the spaces between class borders. The matching frequencies of similar classes determine the heights of rectangles, whereas the matching frequency densities of dissimilar classes determine the heights of rectangles.

**12.4 Evaluation of The Model**

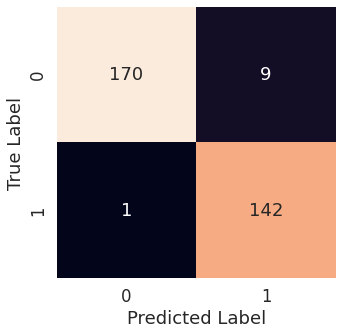
**12.4.1 Random Forest (RF)**

The classification result for the RF model is shown in Figure 10.



**Figure 14**: Classification result of Random Forest

The overall F1 score, in this case, is 97 percent. Individual F1 scores range from 97 percent for healthy people to 97 percent for people who have had heartbeat problem. Precision for 0 is 99 percent and for 1 is 94 percent. Prior to fine-tuning, the model had an accuracy of 93 percent. Figure 11 displays the Random Forest model's estimate.

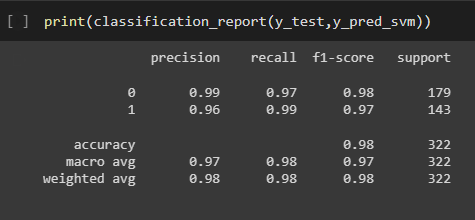


**Figure 15**: Confusion Matrix of Random Forest

The confusion matrix displays the projected result as well as the estimated performance of the model. There were 312 correct predictions and 10 incorrect ones.

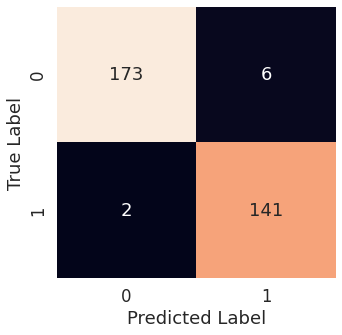
**12.4.2 Support Vector Machine**

Figure 12 shows the SVM Classifier classification report.



**Figure 16**: Classification Report of SVM

In this case, the final F1 score is 98 percent of the total. According to the study, the F1 score of a healthy individual is 98 percent, but the score of a person who has had heatbeat problem is 97 percent. Figure 13 also depicts the precision and recall of the test results. An SVM model that has been fine-tuned has also been incorporated. However, even after fine-tuning, the accuracy remained unchanged.

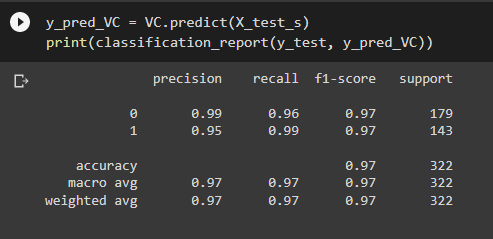


**Figure 17**: Confusion Matrix of SVM classifier

The forecast made by the SVM model is seen in Figure 14. There were 314 correct predictions and just eight incorrect predictions.

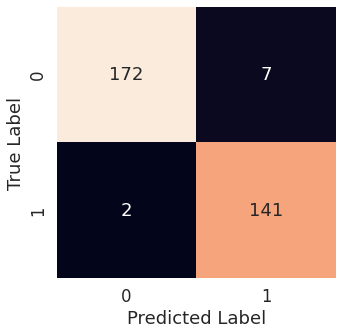
**12.4.3 Voting Classifier**

Figure 14 shows the categorization report that the Voting Classifier makes for use in voting.



**Figure 18**: Classification Report of Voting Classifier

In this case, the final F1 score is 97 percent of the total. According to the study, the F1 score of a healthy individual is 97 percent, but the score of a person who has had heatbeat problem is 97 percent. Figure 15 also depicts the precision and recall of the test results. A voting classifier model that has been fine-tuned has also been incorporated. However, just like SVM, even after fine-tuning, the accuracy remained unchanged.

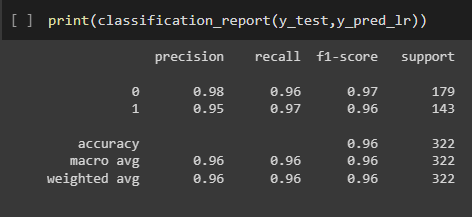


**Figure 19**: Confusion Matrix of Voting Classifier

The forecast made by the Voting classifier model is seen in Figure 16. There were 313 correct predictions and just 9 incorrect predictions.

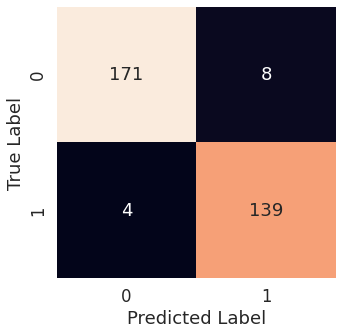
**12.4.4 Logistic Regression Classifier**

Figure 17 shows the categorization report that the Logistic Regression classifier.



**Figure 20**: Classification Report of Logistic Regression Classifier

In this case, the final F1 score is 96 percent of the total. According to the study, the F1 score of a healthy individual is 97 percent, but the score of a person who has had heatbeat problem is 96 percent. Figure 17 also depicts the precision and recall of the test results. A Logistic Regression classifier model that has been fine-tuned has also been incorporated. However, just like SVM and Voting classifier, even after fine-tuning, the accuracy remained unchanged.



**Figure 21:** Confusion Matrix of Logistic Regression Classifier

The forecast made by the Logistic Regression classifier model is seen in Figure 17. There were 310 correct predictions and just 12 incorrect predictions.

**12.4.5 CNN**

The accuracy of the CNN model is shown in Figure 18. This model has the 99% accuracy in classifying objects.

Calendar

Description automatically generated

**Figure 22**: CNN Classifier Accuracy

In this example, the overall accuracy is 99 percent. Individual f1-score is 99 and 97 percent for 0 and 1, respectively. Figure 19 depicts the CNN model's final prediction number.

Graphical user interface

Description automatically generated with low confidence

**Figure 23**: CNN Confusion Matrix

The confusion matrix illustrates the expected result as well as the estimated performance of the model. For a total of 99 percent accuracy, there were 1134 right predictions and 16 wrong forecasts

**12.5 Model Comparison**

The graphic clearly illustrates that the CNN model is the most effective model among the numerous models contained in the framework. Additionally, it has greater precision, better memory, and accuracy, in addition to having a higher F1 score.

**Table 1:** Algorithm Performance Comparison

|  |  |  |  |
| --- | --- | --- | --- |
| This Paper (Model Name) | Accuracy (%) | Reference Paper (Model Name) | Accuracy (%) |
| Random Forest | 97 | Ref [10] Random Forest | 80.18 |
| **Support Vector Machine** | **98** | Ref [9] Support Vector Machine | 90.0 |
| Voting Classifier | 97 | Ref [11] Artificial Neural Network | 54.0 |
| Logistic Regression | 96 | Ref [15] Logistic Regression | 74.56 |
| CNN | 99 | Ref [18] ANN | 83 |

All of the algorithms have an acceptable level of accuracy, as shown in Table 1, but the DL method is the preferred option due to its higher level of accuracy than the others. The accuracy of this study's CNN approach was 99 percent, whereas the authors of [9] achieved only 90 percent with their algorithm. The Random Forest method was used in this study to achieve 97 percent accuracy, while the same algorithm was used by the authors of [10] to achieve 80.18 percent accuracy. A voting classifier was used in this study to achieve 97 percent accuracy, while artificial neural networks were used in the Artificial Neural Network ref [11] to achieve 54 percent accuracy. On the other hand, this article achieved a 96% accuracy rate with a Logistic Regression classifier, while reference [15] achieved a 74% accuracy rate with the same approach.

**13.6 Summary**

This chapter has covered the different types of results that we have managed to obtain throughout the course of using this system.

**CHAPTER 13**

**CONCLUSION**

In the field of machine learning-based illness prediction, medical data are essential. Data detection and diagnostics applications rely on machine learning to learn from previous data and make predictions based on that knowledge. The review shows that the CNN method and random forest could be used to better predict seizures caused by epilepsy. These findings could be added to the current research in the area of heatbeat problem prediction utilizing emerging machine learning technologies. The investigation revealed that their individual accuracy was almost 100% and 95%. The fact that the precision percentage of the models used in this request is significantly higher than that of the models used in previous studies suggests that the models used in this study are more trustworthy than those used in previous studies. When cross-endorsement assessments are used in the assumption for epileptic seizures, the CNN system beats various cycles. Future research could build on this work by developing a web application that uses these calculations and a larger dataset than the one used in this review. This will help improve outcomes as well as the accuracy and efficacy with which medical professionals can anticipate epilepsy issues. This will be beneficial to the system's appearance as well as the structure's unwavering quality. The goal is to get people to get treatment for epilepsy early and make certain changes to their daily lives. On the other hand, the fact that we utilized a variety of well-known machine learning algorithms to achieve the best results makes our work distinctive. With F1 ratings of 97, 98, 97, and 96 percent, Random Forest (RF), Support Vector Machine (SVM), Voting Classifier (VC), and Logistic Regression (LR) were the most successful algorithms (LR). The fact that the models used in this investigation have a much higher percentage of reliability than the models used in previous studies indicates that these models are more reliable than their predecessors. They were found to be reliable in a number of model comparisons. The study's findings could serve as the basis for the system. Utilizing a larger dataset and machine learning algorithms like AdaBoost and Bagging to enhance the framework models are potential future directions for this study. This increase in dependability will have a positive impact on the framework's presentation as well as its dependability. In exchange for providing some basic information, the machine-learning architecture may be able to assist the general public in determining the likelihood of an adult patient with a heartbeat problem developing. It would, in an ideal world, aid patients in receiving prompt treatment for heartbeat issues and reestablishing their lives following a tragedy.

**BIBLIOGRAPHY**

1. W. H. O. R. Health, W. H. O. C. Diseases, and H. Promotion, Comprehensive Heartbeat problem Control: A Guide to Essential Practice, World Health Organization, Geneva, Switzerland, 2006.
2. Heartbeat problem: ESMO Clinical Practice Guidelines for diagnosis, treatment and follow up N Colombo and others Annals of Oncology, 2012. Volume 23, Supplement 7
3. I. C. Scarinci et al., “Heartbeat problem Prevention: New Tools and Old Barriers,” Disease, vol. 116, no. 11, pp. 2531–2542, 2011.
4. John Doorbar, “Molecular biology of human papillomavirus infection and heartbeat problem”, Clinical Science,May 01, 2006,110(5)525-541;DOI: 10.1042/CS20050369.
5. D. Saslow et al., “American disease society, American society for colposcopy and heartbeat problem pathology, and American society for clinical pathology screening guidelines for the prevention and early detection of heartbeat problem,” CA-Disease J. Clin., vol. 62, pp. 147–172, 2012.
6. Igor Kononenko,“Machine learning for medical diagnosis: history, state of the art and perspective”, Artificial Intelligence in Medicine Volume 23, Issue 1, August 2001, Pages 89-109.
7. NasserH. Sweilama , A.A.Tharwatb , N.K.Abdel Moniemc , ”Support vector machine for diagnosis disease disease: A comparative study”, Egyptian Informatics Journal,Volume 11, Issue 2, December 2010, Pages 81-92.
8. Y. Weng, C. Wu, Q. Jiang, W. Guo and C. Wang, "Application of support vector machines in medical data," 2016 4th International Conference on Cloud Computing and Intelligence Systems (CCIS), Beijing,2016,pp.200-204.
9. W. Wu and H. Zhou, "Data-Driven Diagnosis of Heartbeat problem With Support Vector Machine-Based Approaches," in IEEE Access, vol. 5, pp. 25189-25195, 2017.
10. Y. E. Kurniawati, A. E. Permanasari and S. Fauziati, "Comparative study on data mining classification methods for heartbeat problem prediction using pap smear results," 2016 1st International Conference on Biomedical Engineering (IBIOMED), Yogyakarta, 2016, pp. 1-5.
11. Priyanka K Malli , Dr. Suvarna Nandyal, “Machine learning Technique for detection of Heatbeat problem using k-NN and Artificial Neural Network”, International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), JulyAugust 2017.
12. R. Vidya1\* and G. M. Nasira2, “Prediction of Heartbeat problem using Hybrid Induction Technique: A Solution for Human Hereditary Disease Patterns”, Indian Journal of Science and Technology, August 2016.
13. D. Kashyap et al., "Heartbeat problem detection and classification using Independent Level sets and multi SVMs," 2016 39th International Conference on Telecommunications and Signal Processing (TSP), Vienna, 2016, pp. 523-528.
14. E. Njoroge, S. R. Alty, M. R. Gani and M. Alkatib, "Classification of Heartbeat problem Cells using FTIR Data," 2006 International Conference of the IEEE Engineering in Medicine and Biology Society, New York, NY, 2006, pp. 5338-5341.
15. J. Hyeon, H. J. Choi, K. N. Lee and B. D. Lee, "Automating Papanicolaou Test Using Deep Convolutional Activation Feature," 2017 18th IEEE International Conference on Mobile Data Management (MDM), Daejeon, 2017, pp. 382-385.
16. K. Teeyapan, N. Theera-Umpon and S. Auephanwiriyakul, "Application of support vector based methods for heatbeat problem cell classification," 2015 IEEE International Conference on Control System,Computing and Engineering (ICCSCE), George Town, 2015, pp. 514-519.
17. Heatbeat problem (Risk Factors) Data Set. Available on: https://www.kaggle.com/datasets/shayanfazeli/heartbeat
18. Auria, Laura and Moro, R. A., Support Vector Machines (SVM) as a Technique for Solvency Analysis (August 1, 2008). DIW Berlin Discussion Paper No. 811. Available at SSRN: https://ssrn.com/abstract=1424949 or http://dx.doi.org/ 10.2139/ssrn.1424949

**APPENDIX**

**SOFTWARE LISTING**

**Deep Learning:**

import numpy as np

import pandas as pd

import os

for dirname, \_, filenames in os.walk('/data/input'):

    for filename in filenames:

        print(os.path.join(dirname, filename))

from matplotlib import pyplot as plt

import seaborn as sns

import math

import random

from scipy.signal import resample

train\_df = pd.read\_csv('/data/input/heartbeat/mitbih\_train.csv',header=None)

test\_df = pd.read\_csv('/data/input/heartbeat/mitbih\_test.csv',header=None)

train\_df.sample(10)

#Classes:['N': 0, 'S': 1, 'V': 2, 'F': 3, 'Q': 4]

Classes = {

    0:'N',

    1:'S',

    2:'V',

    3:'F',

    4:'Q'

}

Classes\_index = {

    'N': 0,

    'S': 1,

    'V': 2,

    'F': 3,

    'Q': 4

}

train\_df.info()

test\_df.info()

feat\_cols\_name = [i for i in range(train\_df.shape[1]-1)]

train\_df.columns = feat\_cols\_name+["label"]

test\_df.columns = feat\_cols\_name+["label"]

train\_df["label"] = train\_df["label"].astype(int)

test\_df["label"] = test\_df["label"].astype(int)

train\_df.columns[train\_df.isnull().sum() != 0]

test\_df.columns[train\_df.isnull().sum() != 0]

train\_df.describe()

fig, axes= plt.subplots(math.ceil(len(Classes\_index.keys())/2), 2, figsize=(20,30), constrained\_layout=True)

X = np.arange(0, len(feat\_cols\_name))\*8/1000

i = 0

for key, value in Classes\_index.items():

    ax = axes[int(i/2), i%2]

    ax.plot(X, train\_df[train\_df["label"] == key][feat\_cols\_name].sample(1).values.reshape(-1,1))

    ax.legend()

    ax.set(title="1-beat ECG for Cat {}".format(key))

    ax.set(ylabel="Amplitude")

    ax.set(xlabel="Time (ms)")

    i+=1

def stretch(x):

    l = int(len(feat\_cols\_name) \* (1 + (random.random()-0.5)/3))

    y = resample(x, l)

    if l < len(feat\_cols\_name):

        y\_ = np.zeros(shape=(len(feat\_cols\_name), ))

        y\_[:l] = y

    else:

        y\_ = y[:len(feat\_cols\_name)]

    return y\_

def amplify(x):

    alpha = (random.random()-0.5)

    factor = -alpha\*x + (1+alpha)

    return x\*factor

def gaussian\_noise(signal):

    noise=np.random.normal(0,0.01, len(feat\_cols\_name))

    return (signal+noise)

def augment(x):

    result = np.zeros(shape= (4, len(feat\_cols\_name)))

    for i in range(3):

        if random.random() < 0.25:

            new\_X = gaussian\_noise(x)

        elif random.random() < 0.50:

            new\_X = stretch(x)

        elif random.random() < 0.75:

            new\_X = amplify(x)

        else:

            new\_X = gaussian\_noise(x)

            new\_X = stretch(new\_X)

            new\_X = amplify(new\_X)

        result[i, :] = new\_X

    return result

train\_df["label"] = train\_df["label"].map(Classes\_index)

test\_df["label"] = test\_df["label"].map(Classes\_index)

X\_train = train\_df[feat\_cols\_name].values

y\_train = train\_df["label"].values

X\_test = test\_df[feat\_cols\_name].values

y\_test = test\_df["label"].values

plt.plot(X\_train[0])

plt.plot(gaussian\_noise(X\_train[0]))

plt.plot(amplify(X\_train[0]))

plt.plot(stretch(X\_train[0]))

plt.show()

from sklearn.model\_selection import train\_test\_split

X\_train, X\_val, y\_train, y\_val = train\_test\_split(X\_train, y\_train, test\_size=0.1, random\_state=2, stratify=y\_train)

X\_train.shape, y\_train.shape, X\_val.shape, y\_val.shape

tmp\_X\_train = X\_train.copy()

tmp\_y\_train = y\_train.copy()

X\_train = []

y\_train = []

for i in range(0, len(tmp\_y\_train)):

    augment\_X = augment(tmp\_X\_train[i])

    augment\_y = np.ones(shape=(augment\_X.shape[0],), dtype=int)\*tmp\_y\_train[i]

    X\_train.extend(augment\_X)

    y\_train.extend(augment\_y)

X\_train = np.array(X\_train)

y\_train = np.array(y\_train)

X\_train.shape, y\_train.shape

X\_train = X\_train.reshape((-1, len(feat\_cols\_name), 1))

X\_val = X\_val.reshape((-1, len(feat\_cols\_name), 1))

X\_test = X\_test.reshape((-1, len(feat\_cols\_name), 1))

y\_train = y\_train.reshape((-1, 1))

y\_val = y\_val.reshape((-1, 1))

y\_test = y\_test.reshape((-1, 1))

input\_shape = X\_train.shape[1:]

nb\_classes = len(Classes\_index.keys())

input\_shape, nb\_classes

from tensorflow.keras.utils import to\_categorical

y\_train = to\_categorical(y\_train, num\_classes=nb\_classes)

y\_val = to\_categorical(y\_val, num\_classes=nb\_classes)

y\_test = to\_categorical(y\_test, num\_classes=nb\_classes)

from sklearn.metrics import accuracy\_score, precision\_recall\_fscore\_support,confusion\_matrix, classification\_report, precision\_score, recall\_score

from sklearn.metrics import f1\_score as f1\_score\_rep

import seaborn as sn

def print\_score(y\_pred, y\_real, label\_encoder):

    print("Accuracy: ", accuracy\_score(y\_real, y\_pred))

    print("Precision:: ", precision\_score(y\_real, y\_pred, average="micro"))

    print("Recall:: ", recall\_score(y\_real, y\_pred, average="micro"))

    print("F1\_Score:: ", f1\_score\_rep(y\_real, y\_pred, average="micro"))

    print()

    print("Macro precision\_recall\_fscore\_support (macro) average")

    print(precision\_recall\_fscore\_support(y\_real, y\_pred, average="macro"))

    print()

    print("Macro precision\_recall\_fscore\_support (micro) average")

    print(precision\_recall\_fscore\_support(y\_real, y\_pred, average="micro"))

    print()

    print("Macro precision\_recall\_fscore\_support (weighted) average")

    print(precision\_recall\_fscore\_support(y\_real, y\_pred, average="weighted"))

    print()

    print("Confusion Matrix")

    cm = confusion\_matrix(y\_real, y\_pred)

    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    df\_cm = pd.DataFrame(cm, index = [i for i in label\_encoder],

                  columns = [i for i in label\_encoder])

    plt.figure(figsize = (10,10))

    sn.heatmap(df\_cm, annot=True)

    print()

    print("Classification Report")

    print(classification\_report(y\_real, y\_pred, target\_names=label\_encoder))

from keras.layers import Conv1D, MaxPool1D, Concatenate, BatchNormalization, Activation, Input, Add, \

                         GlobalAveragePooling1D, Dense

from keras.models import Model

from tensorflow.keras.optimizers import Adam

from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlyStopping

from livelossplot import PlotLossesKeras

from tensorflow.keras.metrics import Recall, Precision

import keras

import time

import keras.backend as K

def f1\_score(y\_true, y\_pred):

    true\_positives = K.sum(K.round(K.clip(y\_true \* y\_pred, 0, 1)))

    possible\_positives = K.sum(K.round(K.clip(y\_true, 0, 1)))

    predicted\_positives = K.sum(K.round(K.clip(y\_pred, 0, 1)))

    precision = true\_positives / (predicted\_positives + K.epsilon())

    recall = true\_positives / (possible\_positives + K.epsilon())

    f1\_val = 2\*(precision\*recall)/(precision+recall+K.epsilon())

    return f1\_val

class Classifier\_INCEPTION:

    def \_\_init\_\_(self, weights\_directory, input\_shape, nb\_classes, verbose=False, build=True, batch\_size=64,

                 nb\_filters=32, use\_residual=True, use\_bottleneck=True, depth=10, kernel\_size=41, nb\_epochs=100):

        self.weights\_directory = weights\_directory

        self.nb\_filters = nb\_filters

        self.use\_residual = use\_residual

        self.use\_bottleneck = use\_bottleneck

        self.depth = depth

        self.kernel\_size = kernel\_size - 1

        self.callbacks = None

        self.batch\_size = batch\_size

        self.bottleneck\_size = 32

        self.nb\_epochs = nb\_epochs

        if build == True:

            self.model = self.build\_model(input\_shape, nb\_classes)

            if (verbose == True):

                self.model.summary()

            self.verbose = verbose

    def \_inception\_module(self, input\_tensor, stride=1, activation='linear'):

        if self.use\_bottleneck and int(input\_tensor.shape[-1]) > 1:

            input\_inception = Conv1D(filters=self.bottleneck\_size, kernel\_size=1,

                                     padding='same', activation=activation, use\_bias=False)(input\_tensor)

        else:

            input\_inception = input\_tensor

        kernel\_size\_s = [self.kernel\_size // (2 \*\* i) for i in range(3)]

        conv\_list = []

        for i in range(len(kernel\_size\_s)):

            conv\_list.append(Conv1D(filters=self.nb\_filters, kernel\_size=kernel\_size\_s[i],

                                    strides=stride, padding='same', activation=activation, use\_bias=False)(

                input\_inception))

        max\_pool\_1 = MaxPool1D(pool\_size=3, strides=stride, padding='same')(input\_tensor)

        conv\_6 = Conv1D(filters=self.nb\_filters, kernel\_size=1,

                        padding='same', activation=activation, use\_bias=False)(max\_pool\_1)

        conv\_list.append(conv\_6)

        x = Concatenate(axis=2)(conv\_list)

        x = BatchNormalization()(x)

        x = Activation(activation='relu')(x)

        return x

    def \_shortcut\_layer(self, input\_tensor, out\_tensor):

        shortcut\_y = Conv1D(filters=int(out\_tensor.shape[-1]), kernel\_size=1,

                            padding='same', use\_bias=False)(input\_tensor)

        shortcut\_y = BatchNormalization()(shortcut\_y)

        x = Add()([shortcut\_y, out\_tensor])

        x = Activation('relu')(x)

        return x

    def build\_model(self, input\_shape, nb\_classes):

        input\_layer = Input(input\_shape)

        x = input\_layer

        input\_res = input\_layer

        for d in range(self.depth):

            x = self.\_inception\_module(x)

            if self.use\_residual and d % 3 == 2:

                x = self.\_shortcut\_layer(input\_res, x)

                input\_res = x

        gap\_layer = GlobalAveragePooling1D()(x)

        output\_layer = Dense(nb\_classes, activation='softmax')(gap\_layer)

        model = Model(inputs=input\_layer, outputs=output\_layer)

        model.compile(loss='categorical\_crossentropy',

                      optimizer=Adam(),

                      metrics=['accuracy', Precision(), Recall(), f1\_score])

        reduce\_lr = ReduceLROnPlateau(monitor='val\_accuracy',

                                      factor=0.5,

                                      patience=int(self.nb\_epochs/20),

                                      min\_lr=0.0001)

        file\_path = os.path.join(self.weights\_directory,"best\_weights.h5")

        model\_checkpoint = ModelCheckpoint(filepath=file\_path,

                                           monitor='val\_accuracy',

                                           mode="max",

                                           save\_best\_only=True)

        early\_stopping = EarlyStopping(monitor="val\_accuracy",

                                       mode="max",

                                       verbose=1,

                                       patience=int(self.nb\_epochs/10))

        plotlosses = PlotLossesKeras()

        self.callbacks = [reduce\_lr, model\_checkpoint, early\_stopping, plotlosses]

        return model

    def fit(self, x\_train, y\_train, x\_val, y\_val, class\_weights=None):

        if self.batch\_size is None:

            mini\_batch\_size = int(min(x\_train.shape[0] / 10, 16))

        else:

            mini\_batch\_size = self.batch\_size

        start\_time = time.time()

        hist = self.model.fit(x\_train, y\_train,

                              batch\_size=mini\_batch\_size,

                              epochs=self.nb\_epochs,

                              verbose=self.verbose,

                              validation\_data=(x\_val, y\_val),

                              callbacks=self.callbacks)

        duration = time.time() - start\_time

        keras.backend.clear\_session()

        print("Model take {} S to train ".format(duration))

        return hist

weights\_directory = "./"

inception = Classifier\_INCEPTION(weights\_directory, input\_shape, nb\_classes, 1,

                                 batch\_size=256, build=True)

from tensorflow.keras.utils import plot\_model

plot\_model(inception.model, to\_file="model\_fig.jpg", show\_shapes=True)

history = inception.fit(X\_train, y\_train, X\_val, y\_val)

inception.model.load\_weights("./best\_weights.h5")

inception.model.evaluate(X\_val, y\_val)

inception.model.evaluate(X\_test, y\_test)

y\_hat = inception.model.predict(X\_test)

from sklearn.metrics import roc\_curve, auc

from itertools import cycle

def ROC\_plot(y\_true\_ohe, y\_hat\_ohe, label\_encoder, n\_classes):

    lw = 2

    fpr = dict()

    tpr = dict()

    roc\_auc = dict()

    for i in range(n\_classes):

        fpr[i], tpr[i], \_ = roc\_curve(y\_true\_ohe[:, i], y\_hat\_ohe[:, i])

        roc\_auc[i] = auc(fpr[i], tpr[i])

    all\_fpr = np.unique(np.concatenate([fpr[i] for i in range(n\_classes)]))

    mean\_tpr = np.zeros\_like(all\_fpr)

    for i in range(n\_classes):

        mean\_tpr += np.interp(all\_fpr, fpr[i], tpr[i])

    mean\_tpr /= n\_classes

    fpr["macro"] = all\_fpr

    tpr["macro"] = mean\_tpr

    roc\_auc["macro"] = auc(fpr["macro"], tpr["macro"])

    fpr["micro"], tpr["micro"], \_ = roc\_curve(y\_true\_ohe.ravel(), y\_hat\_ohe.ravel())

    roc\_auc["micro"] = auc(fpr["micro"], tpr["micro"])

    plt.figure(figsize=(20,20))

    plt.plot(

        fpr["micro"],

        tpr["micro"],

        label="micro-average ROC curve (area = {0:0.2f})".format(roc\_auc["micro"]),

        color="deeppink",

        linestyle=":",

        linewidth=4,

    )

    plt.plot(

        fpr["macro"],

        tpr["macro"],

        label="macro-average ROC curve (area = {0:0.2f})".format(roc\_auc["macro"]),

        color="navy",

        linestyle=":",

        linewidth=4,

    )

    colors = cycle(["aqua", "darkorange", "cornflowerblue"])

    for i, color in zip(range(n\_classes), colors):

        plt.plot(

            fpr[i],

            tpr[i],

            color=color,

            lw=lw,

            label="ROC curve of class {0} (area = {1:0.2f})".format(list(label\_encoder.keys())[i], roc\_auc[i]))

    plt.plot([0, 1], [0, 1], "k--", lw=lw)

    plt.xlim([0.0, 1.0])

    plt.ylim([0.0, 1.05])

    plt.xlabel("False Positive Rate")

    plt.ylabel("True Positive Rate")

    plt.title("multiclass characteristic")

    plt.legend(loc="lower right")

    plt.show()

ROC\_plot(y\_test, y\_hat, Classes\_index, nb\_classes)

y\_hat = np.argmax(y\_hat, axis=1)

y\_true = np.argmax(y\_test, axis=1)

print\_score(y\_hat, y\_true, Classes\_index)

Machine Learning:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sb

%matplotlib inline

import os

for dirname, \_, filenames in os.walk('/kaggle/input'):

    for filename in filenames:

        print(os.path.join(dirname, filename))

train\_data = pd.read\_csv('/input/heartbeat/mitbih\_train.csv',header = None)

test\_data = pd.read\_csv('/input/heartbeat/mitbih\_train.csv', header = None)

train\_target = train\_data[187].value\_counts()

train\_target

plt.bar(train\_target.index,train\_target.values, color = sb.color\_palette()[4]);

np.random.seed(2018)

sample = np.random.choice(train\_data.shape[0], 200, replace = False)

subset = train\_data.loc[sample]

subset

percentages = [count / subset.shape[0] \* 100 for count in subset[187].value\_counts()]

percentages[0]

for i in np.arange(len(percentages)):

    print(f'the precent of {int(train\_target.index[i])} is : {np.round(percentages[i], 2) } %')

base\_color = sb.color\_palette()[0]

type\_orderion = subset[187].value\_counts().index

sb.countplot(data = subset, y = 187, color = base\_color, order = type\_orderion );

plt.xticks(rotation = 90);

from sklearn.utils import resample

target1=train\_data[train\_data[187]==1]

target2=train\_data[train\_data[187]==2]

target3=train\_data[train\_data[187]==3]

target4=train\_data[train\_data[187]==4]

target0=(train\_data[train\_data[187]==0]).sample(n=20000,random\_state=42)

target1\_sample=resample(target1,replace=True,n\_samples=20000,random\_state=0)

target2\_sample=resample(target2,replace=True,n\_samples=20000,random\_state=0)

target3\_sample=resample(target3,replace=True,n\_samples=20000,random\_state=0)

target4\_sample=resample(target4,replace=True,n\_samples=20000,random\_state=0)

train\_data=pd.concat([target0,target1\_sample,target2\_sample,target3\_sample,target4\_sample])

train\_target = train\_data[187].value\_counts()

train\_target

plt.bar(train\_target.index,train\_target.values, color = 'green');

sorted\_counts = train\_data[187].value\_counts();

plt.figure(figsize = [12,5.01])

plt.pie(sorted\_counts, labels = sorted\_counts.index, startangle = 90,

        counterclock = False);

plt.axis('square');

test\_target = test\_data[187].value\_counts()

test\_target

plt.bar(test\_target.index,test\_target.values, color = sb.color\_palette()[0]);

np.random.seed(2020)

sample2 = np.random.choice(test\_data.shape[0], 200, replace = False)

subset2 = test\_data.loc[sample2]

percentages = [count / subset2.shape[0] \* 100 for count in subset2[187].value\_counts()]

percentages[0]

for i in np.arange(len(percentages)):

    print(f'the precent of {int(test\_data.index[i])} is : {np.round(percentages[i], 2) } %')

from scipy import stats

pearson\_coef, p\_value = stats.pearsonr(train\_data[1], train\_data[187])

l = [('1', 1),('2',2),('3',3)]

if max(l):

    print(l)

max(l)

effictive\_list = []

for i in train\_data.columns:

    if i != 187:

        pearson\_coef, p\_value = stats.pearsonr(train\_data[i], train\_data[187])

        effictive\_list.append((f'column number : {i}', f'Pearson Correlation {pearson\_coef}', f'P-value : {p\_value}'))

    else: break

print('\*' \* 75)

print(max(effictive\_list))

print('\*' \* 75)

def add\_guassian\_noise(signal):

    noise = np.random.normal(0,0.05,186)

    return(signal + noise)

noise\_data = add\_guassian\_noise(train\_data.iloc[0,:186])

plt.figure(figsize = [12,4.02])

plt.plot(train\_data.iloc[0,:186], color = 'green')

plt.title('data without noise')

plt.xlabel('ECG in mili volts')

plt.ylabel('time in seconds')

plt.figure(figsize = [12,4.02])

plt.plot(noise\_data, color = 'red')

plt.title('data with noise')

plt.xlabel('ECG in mili volts')

plt.ylabel('time in seconds')

plt.show()

len(train\_data.columns)/60

df = train\_data.copy()

counter = 0

while counter <= len(df.columns):

    if counter != 186:

        df[counter] = df[counter] + df[counter + 1] + df[counter + 2]

        counter += 3

    else: break

df.head()

plt.figure(figsize = [12,4.02])

plt.plot(df.iloc[0,:186], color = 'green')

plt.title('data without noise');

plt.ylabel('ECG in mili volts');

plt.xlabel('time in minutes');

def add\_guassian\_noise2(signal):

    noise = np.random.normal(0,500,186)

    return(signal + noise)

plt.figure(figsize = [12,4.02])

plt.plot(add\_guassian\_noise2(df.iloc[0,:186]), color = 'red')

plt.title('data with noise')

plt.ylabel('ECG in mili volts')

plt.xlabel('time in minutes')

plt.show()

df.head()

x = train\_data.iloc[0,:].values

from numpy.fft import fft, ifft

x = train\_data.iloc[0,:]

X = fft(x)

plt.plot(X)

X = fft(train\_data.iloc[0,:186])

N = len(X)

n = np.arange(N)

# get the sampling rate

sr = 1 / (60\*60)

T = N/sr

freq = n/T

# Get the one-sided specturm

n\_oneside = N//2

# get the one side frequency

f\_oneside = freq[:n\_oneside]

plt.figure(figsize = (12, 6))

plt.plot(f\_oneside, np.abs(X[:n\_oneside]), 'b')

plt.xlabel('Freq (Hz)')

plt.ylabel('FFT Amplitude |X(freq)|')

plt.show()

N = len(X)

n = np.arange(N)

T = N/sr

freq = n/T

plt.stem(freq, np.abs(X), 'b', \

         markerfmt=" ", basefmt="-b")

plt.xlabel('Freq (Hz)')

plt.ylabel('FFT Amplitude |X(freq)|')

plt.xlim(0, 10);

plt.plot(ifft(X), 'r')

plt.xlabel('Time (s)')

plt.ylabel('Amplitude')

plt.tight\_layout()

plt.show()

t\_h = 1/f\_oneside / (60 \* 60)

plt.figure(figsize=(12,6))

plt.plot(t\_h, np.abs(X[:n\_oneside])/n\_oneside)

plt.xticks([12, 24, 84, 168])

plt.xlim(0, 200)

plt.xlabel('Period (hour)')

plt.show()

from sklearn.model\_selection import train\_test\_split

X\_train = train\_data.drop(187, axis=1)

y\_train = train\_data[187]

X\_test = test\_data.drop(187, axis=1)

y\_test = test\_data[187]

X\_train = X\_train.values

y\_train = y\_train.values

X\_test = X\_test.values

y\_test = y\_test.values

unique, counts = np.unique(y\_train, return\_counts=True)

pd.DataFrame(np.asarray((unique, counts)).T)

val = X\_train[0][1]

from sklearn.preprocessing import Normalizer

from sklearn.preprocessing import StandardScaler

norm = Normalizer()

std = StandardScaler()

val1 = norm.fit\_transform([[val]])

val2 = std.fit\_transform([[val]])

from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from sklearn.naive\_bayes import GaussianNB

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from xgboost import XGBClassifier

from sklearn.metrics import confusion\_matrix, accuracy\_score, precision\_score, recall\_score, f1\_score, fbeta\_score

models = {

    "LR": LogisticRegression(solver='liblinear'),

    "KNN": KNeighborsClassifier(),

    "DT": DecisionTreeClassifier(),

    "RF": RandomForestClassifier(),

    "XGB": XGBClassifier(),

    "Naive Bayes": GaussianNB(),

    "SVC": SVC()

}

type(X\_train)

pip install scikit-learn  -U

for name, model in models.items():

    print(f'Training Model {name} \n--------------')

    model.fit(X\_train, y\_train)

    y\_pred = model.predict(X\_test)

    print(f'Training Accuracy: {accuracy\_score(y\_train, model.predict(X\_train))}')

    print(f'Testing Accuracy: {accuracy\_score(y\_test, y\_pred)}')

    print(f'Testing Confusion Matrix: \n{confusion\_matrix(y\_test, y\_pred)}')

    print("Recall Score : ",recall\_score(y\_test, y\_pred, pos\_label='positive', average='micro'))

    print("Precision Score : ",precision\_score(y\_test, y\_pred,  pos\_label='positive',average='micro'))

    print(f"Testing F-1:", f1\_score(y\_test, y\_pred, pos\_label='positive', average='micro' )        )

    print(f"Testing F Beta:", fbeta\_score(y\_test, y\_pred, beta=0.5, pos\_label='positive', average='micro'))

    print('-'\*30)

model = XGBClassifier()

model.fit(X\_train, y\_train)

print(model)

# make predictions for test data

y\_pred = model.predict(X\_test)

predictions = [round(value) for value in y\_pred]

# evaluate predictions

accuracy = accuracy\_score(y\_test, predictions)

print("Accuracy: %.2f%%" % (accuracy \* 100.0))

rounded\_y = np.round(y\_pred)

unique = pd.DataFrame(y\_test)

unique[0].unique()

from sklearn.metrics import multilabel\_confusion\_matrix

y\_unique = unique[0].unique()

mcm = multilabel\_confusion\_matrix(y\_test, y\_pred, labels = y\_unique)

mcm

from sklearn.model\_selection import  GridSearchCV

params = {

        'min\_child\_weight': [1, 5, 10],

        'gamma': [0.5, 1, 1.5, 2, 5],

        'subsample': [0.6, 0.8, 1.0],

        'colsample\_bytree': [0.6, 0.8, 1.0],

        'max\_depth': [3, 4, 5]

}

import tqdm

import time

# for i in time:

# #   tqdm(grid\_search = grid\_search.fit(X\_train, y\_train))

# best\_accuracy = grid\_search.best\_score\_

# bets\_params = grid\_search.best\_params\_

# print(f'the best accuracy is {best\_accuracy} % while the recommanded parameters is : {bets\_params}')

# import warnings

# warnings.filterwarnings('ignore')

# import numpy as np

# import pandas as pd

# from datetime import datetime

# from sklearn.model\_selection import RandomizedSearchCV, GridSearchCV

# from sklearn.metrics import roc\_auc\_score

# from sklearn.model\_selection import StratifiedKFold

# from xgboost import XGBClassifier

# def timer(start\_time=None):

#     if not start\_time:

#         start\_time = datetime.now()

#         return start\_time

#     elif start\_time:

#         thour, temp\_sec = divmod((datetime.now() - start\_time).total\_seconds(), 3600)

#         tmin, tsec = divmod(temp\_sec, 60)

#         print('\n Time taken: %i hours %i minutes and %s seconds.' % (thour, tmin, round(tsec, 2)))

# folds = 3

# param\_comb = 5

# skf = StratifiedKFold(n\_splits=folds, shuffle = True, random\_state = 1001)

# random\_search = RandomizedSearchCV(model, param\_distributions=params, n\_iter=param\_comb, scoring='roc\_auc', n\_jobs=-1 cv=skf.split(X\_train,y\_train), verbose=3, random\_state=1001 )

# # Here we go

# start\_time = timer(None) # timing starts from this point for "start\_time" variable

# random\_search.fit(X\_train, y\_train)

# timer(start\_time)

# print('\n All results:')

# print(random\_search.cv\_results\_)

# print('\n Best estimator:')

# print(random\_search.best\_estimator\_)

# print('\n Best normalized gini score for %d-fold search with %d parameter combinations:' % (folds, param\_comb))

# print(random\_search.best\_score\_ \* 2 - 1)

# print('\n Best hyperparameters:')

# print(random\_search.best\_params\_)

# results = pd.DataFrame(random\_search.cv\_results\_)

# results.to\_csv('xgb-random-grid-search-results-01.csv', index=False)