Early Detection of In-Hospital Cardiac Arrest Using Machine Learning

Submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Shashank Kansal

18BCE0860

Vikas Mani Tripathi

18BCE0615

Under the guidance of

Prof. Manjula R

SCOPE

VIT, Vellore

June,2022

DECLARATION

I hereby declare that the thesis entitled “Early Detection In-Hospital Cardiac Arrest Based On Machine Learning” submitted by me, for the award of the degree of Bachelor of Technology in Computer Science and Engineering to VIT is a record of bonafide work carried out by me under the supervision of Prof. Manjula R.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place : Vellore

Date : 03 June 2022

CERTIFICATE

This is to certify that the thesis entitled "Early Detection In-Hospital Cardiac Arrest Based On Machine Learning" submitted by Shashank Kansal & Reg. No 18BCE0860, SCOPE, VIT University, for the award of the degree of Bachelor of Technology in Computer Science and Engineering, is a record of bonafide work carried out by him under my supervision during the period, 03. 02. 2022 to 03.06.2022, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place : Vellore

Date : 3 June 2022

Executive Summary

Over the last few decades, heart disease is the most common cause of global death. So early detection of cardiac arrest and continuous monitoring can reduce the mortality rate. The exponential growth of data from different sources such as wearable sensor devices used in Internet of Things health monitoring, streaming system and others have been generating an enormous amount of data on a continuous basis. The combination of streaming big data analytics and machine learning is a breakthrough technology that can have a significant impact in healthcare field especially early detection of cardiac arrest. This technology can be more powerful and less expensive. To overcome this issue, this paper propose a cardiac arrest prediction system based on machine learning with django framework which stand as a strong large scale distributed computing platform that can be used successfully for streaming data event against machine learning through in-memory computations. The system consists of two main sub parts, namely streaming processing and data storage and visualization. The first uses jupyter for python language and applies classification model like RF and KNN on data events to predict heart disease. The seconds uses Django framework for visualizing the machine learning models.

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION

1.2Jupyter

Jupyter, previously known as IPython Notebook, is a web-based, interactive development environment. Originally developed for Python, it has since expanded to support over 40 other programming languages including Julia and R.

Jupyter allows for *notebooks*to be written that contain text, live code, images, and equations. These notebooks can be shared, and can even be hosted on GitHubfor free.

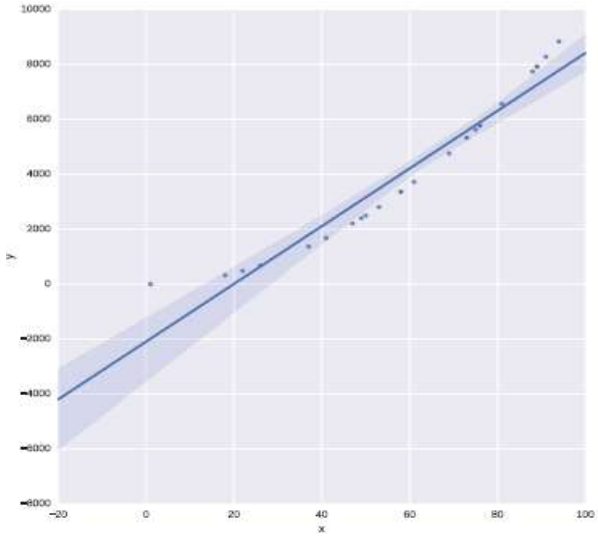
For each section of this tutorial, you can download a Juypter notebook that allows you to edit and experiment with the code and examples for each topic. Jupyter is part of the Anaconda distribution; it can be started from the command line using the jupyter command:

```
$ jupyter notebook
```

1.3 Machine Learning

We will now move on to the task of machine learning itself. In the following sections we will describe how to use some basic algorithms, and perform regression, classification, and clustering

on some freely available medical datasets concerning breast cancer and diabetes, and we will also take a look at a DNA microarray dataset.



SciKit-Learn

SciKit-Learn provides a standardised interface to many of the most commonly used machine learning algorithms, and is the most popular and frequently used library for machine learning for Python. As well as providing many learning algorithms, SciKit-Learn has a large number of convenience functions for common preprocessing tasks (for example, normalisation or *k*-fold cross validation).

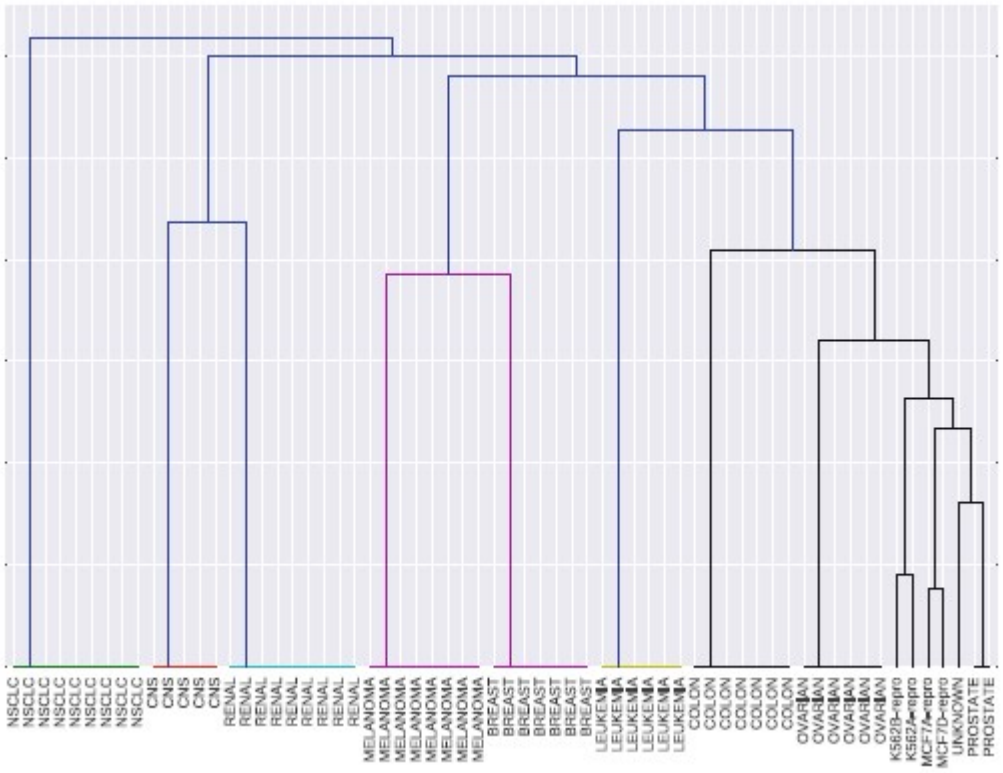
SciKit-Learn is a very large software library.

Clustering

Clustering algorithms focus on ordering data together into groups. In general clustering algorithms are unsupervised—they require no **y** response variable as input. That is to say, they attempt to find groups or clusters within data where you do not know the label for each sample. SciKit-Learn have many clusteringalgorithms, but in this section we will demonstrate hierarchical clustering on a DNA expression microarray dataset using an algorithm from the SciPy library.

We will plot a visualisation of the clustering using what is known as a dendrogram, also using the SciPy library.

The goal is to cluster the data properly in logical groups, in this case into the cancer types represented by each sample's expression data. We do this using agglomerative hierarchical clustering, using Ward's linkage method:



1.4Classification

weanalysed data that was **unlabelled**—we did not know to what class a sample belonged (known as unsupervised learning). In contrast to this, a supervised problem deals with **labelled** data where are aware of the discrete classes to which each sample belongs. When we wish to predict which class a sample belongs to, we call this a classification problem. SciKit-Learn has a number of algorithms for classification, in this section we will look at the Support Vector Machine.

We will work on the Wisconsin breast cancer dataset, split it into a training set and a test set, train a Support Vector Machine with a linear kernel, and test the trained model on an unseen dataset. The Support Vector Machine model should be able to predict if a new sample is malignant or benign based on the features of a new, unseen sample:

```
>>> from sklearn import cross_validation
>>> from sklearn import datasets
>>> from sklearn.svm import SVC
>>> from sklearn.metrics import classification_report
>>> X = datasets.load_breast_cancer().data
>>> y = datasets.load_breast_cancer().target
>>> X_train, X_test, y_train, y_test = cross_validation.
    train_test_split(X, y, test_size=0.2)
>>> svm = SVC(kernel="linear")
>>> svm.fit(X_train, y_train)
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape=None, degree=3, gamma="auto",
    kernel="linear", max_iter=-1, probability=False,
    random_state=None, shrinking=True,  tol=0.001, verbose=
    False)
>>> svm.score(X_test, y_test)
0.95614035087719296
>>> y_pred = svm.predict(X_test)
>>> classification_report(y_test, y_pred)

             precision    recall  f1-score   support

  malignant       1.00      0.89      0.94        44
     benign       0.93      1.00      0.97        70

avg / total       0.96      0.96      0.96       114
```

You will notice that the SVM model performed very well at predicting the malignancy of new, unseen samples from the test set—this can be quantified nicely by printing a number of metrics using the classification report function. Here, the precision, recall, and $F1$ score ($F1 = 2 \cdot \text{precision} \cdot \text{recall} / \text{precision} + \text{recall}$) for each class is shown. The support column is a count of the number of samples for each class.

Support Vector Machines are a very powerful tool for classification. They work well in high dimensional spaces, even when the number of features is higher than the number of samples. However, their running time is quadratic to the number of samples so large datasets can become difficult to train. Quadratic means that if you increase a dataset in size by 10 times, it will take 100 times longer to train.

Last, you will notice that the breast cancer dataset consisted of 30 features. This makes it difficult to visualize or plot the data. To aid in visualization of highly dimensional data, we can apply a technique called dimensionality reduction.

Dimensionality Reduction

Another important method in machine learning, and data science in general, is dimensionality reduction. For this example, we will look at the Wisconsin breast cancer dataset once again. The dataset consists of over 500 samples, where each sample has 30 features. The features relate to

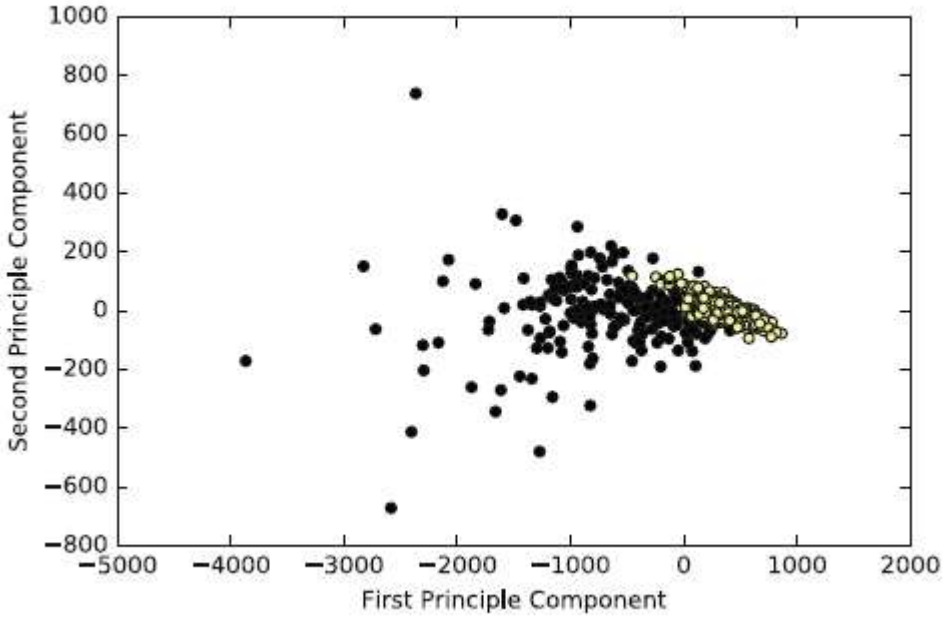
images of a fine needle aspirate of breast tissue, and the features describe the characteristics of the cells present in the images. All features are real values. The target variable is a discrete value (either malignant or benign) and is therefore a classification dataset.

You will recall from the Iris example in Sect. 7.3 that we plotted a scatter matrix of the data, where each feature was plotted against every other feature in the dataset to look for potential correlations (Fig. 3). By examining this plot you could probably find features which would separate the dataset into groups. Because the dataset only had 4 features we were able to plot each feature against each other relatively easily. However, as the numbers of features grow, this becomes less and less feasible, especially if you consider the gene expression example in Sect. 9.4 which had over 6000 features.

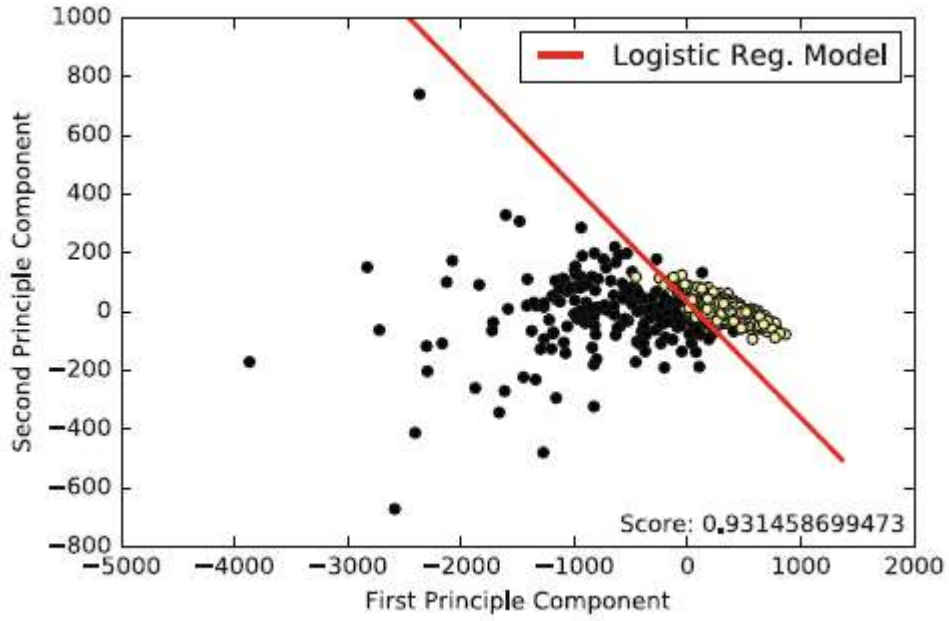
One method that is used to handle data that is highly dimensional is Principle Component Analysis, or PCA. PCA is an unsupervised algorithm for reducing the number of dimensions of a dataset. For example, for plotting purposes you might want to reduce your data down to 2 or 3 dimensions, and PCA allows

you to do this by generating components, which are combinations of the original features, that you can then use to plot your data.

PCA is an unsupervised algorithm. You supply it with your data, **X**, and you specify the number of components you wish to reduce its dimensionality to. This is known as transforming the data:



Again, you would not use this model for new data—in a real world scenario, you would, for example, perform a 10-fold cross validation on the dataset, choosing the model parameters that perform best on the cross validation. This model would be much more likely to perform well on new data. At the very least, you would randomly select a subset, say 30% of the data, as a test set and train the model on the remaining 70% of the dataset. You would evaluate the model based on the score on the test set and not on the training set



1.4.1 NEURAL NETWORKS AND DEEP LEARNING

While a proper description of neural networks and deep learning is far beyond the scope of this chapter, we will however discuss an example use case of one of the most popular frameworks for deep learning: Keras4.

In this section we will use Keras to build a simple neural network to classify theWisconsin breast cancer dataset that was described earlier. Often, deep learning algorithms and neural networks are used to classify images—convolutional neural networks are especially used for image related classification. However,

they can of course be used for text or tabular-based data as well. In this we will build a standard feed-forward, densely connected neural network and classify a text-based cancer dataset in order to demonstrate the framework'susage.

In this example we are once again using the Wisconsin breast cancer dataset, which consists of 30 features and 569 individual samples. To make it more challenging for the neural network, we

will use a training set consisting of only 50% of the entire dataset, and test our neural network on the remaining 50% of the data.

Note,Keras is not installed as part of the Anaconda distribution, to install it use pip:

```
$sudo pip install keras
```

Keras additionally requires either Theano or TensorFlow to be installed. In the examples in this chapter we are using Theano as a backend, however the code will work identically for either backend. You can install Theano using pip, but it has a number of dependencies that must be installed first. Refer to the Theano and TensorFlow documentation for more information [12].

Keras is a modular API. It allows you to create neural networks by building a stack of modules, from the input of the neural network, to the output of the neural network, piece by piece until you have a complete network. Also, Keras can be configured to use your Graphics Processing Unit, or GPU. This makes training neural networks far faster than if we were to use a CPU. We begin by importing Keras:

```
>>> import keras
Using Theano backend.
Using gpu device 0: GeForce GTX TITAN X (CNMeM is enabled
    with initial size: 90.0 % of memory, cuDNN 4007)
```

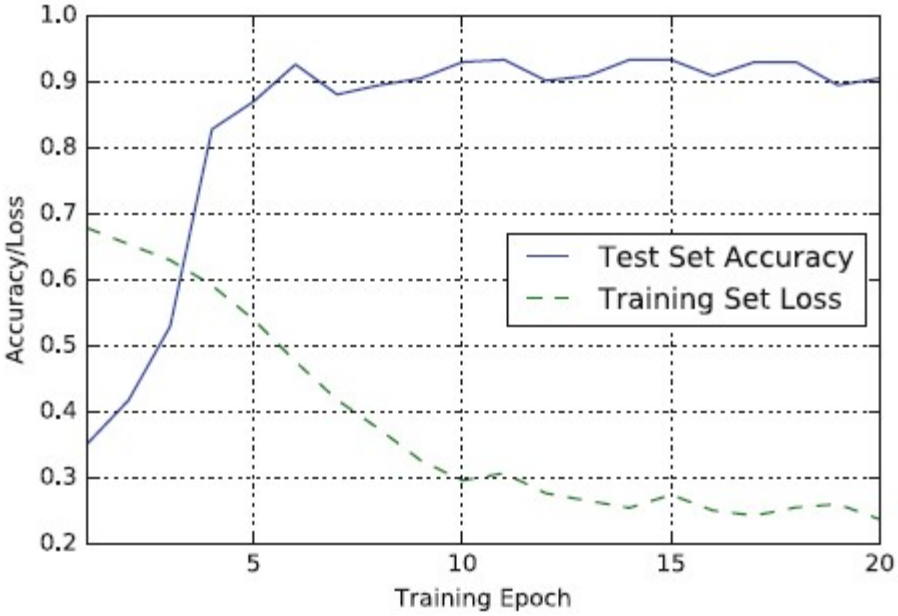
We may want to view the network's accuracy on the test (or its loss on the training set) over time (measured at each epoch), to get a better idea how well it is learning. An epoch is one complete cycle through the training data.

Fortunately, this is quite easy to plot as Keras' fit function returns a history object which we can use to do exactly this:

```
>>> plt.plot(h.history["val_acc"])
>>> plt.plot(h.history["loss"])
>>> plt.show()
```

This will result in a plot similar to that shown. Often you will also want to plot the loss on the test set and training set, and the accuracy on the test set and training set.

Plotting the loss and accuracy can be used to see if you are over fitting (you experience tiny loss on the training set, but large loss on the test set) and to see when your training has plateaued.



PROBLEM STATEMENT:

The only shortcoming faced during the implementation of the algorithm to predict the cardiac arrest is the unavailability of the larger dataset than used. If a larger dataset was available then the neural network would have been trained more accurately and the results would have been more precise than they presently are. When a much larger dataset is available then CNN could be applied and the epochs and hidden layers could be increased which would increase the accuracy and precision of the outcome provided by the neural network.

1.5 OBJECTIVE STATEMENT:

Cardiac arrest is a life-threatening cessation of activity in the heart. Early prediction of cardiac arrest is important, as it allows for the necessary measures to be taken to prevent or intervene during the onset. Artificial intelligence (AI) technologies and big data have been increasingly used to enhance the ability to predict and prepare for the patients at risk.This study aims to explore the use of AI technology in predicting cardiac arrest as reported in the literature.

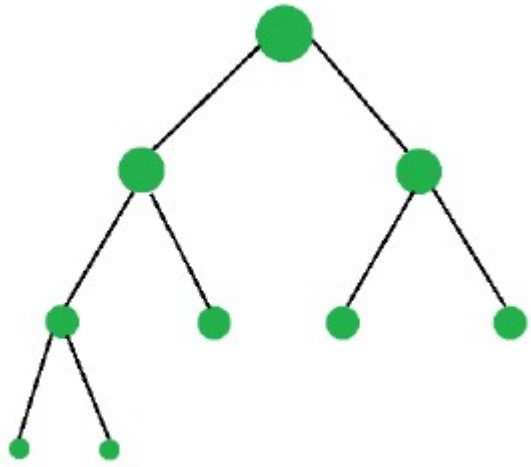
SCOPE OF THE PROJECT:

cardiac arrest and other heart diseases are the most common problem in majority of people, and there are various factors that act as backbone of this problem like people are not paying attention towards health mainly because of work stress, laziness, substandard quality of food that results in increasing cholesterol and untimely diagnosis of heart disease due to lack of technology, methods used in diagnosing these diseases and consequently having a lot of tests. A lot of research and medical supporting systems are developing day by day, however, every system have its various features or advantages and limitations which are unknown to either side. This paper aims to study different machine learning algorithms on dataset to predict the possibility of a cardiac arrest based on various controlled and uncontrolled variables.

1.6 EXISTING SYSTEM

Decision tree algorithm is a supervised learning algorithm that is developed to solve the problems of regression and classification. So, the main advantage of decision trees is that they can handle both numerical and categorical data. Like other conventional algorithms decision tree algorithm creates a training model and that training model is used to predict the value or class of the target label/variable but here this is done by learning decision rules inferred from previous training dataset. This algorithm makes use of tree structure in which the internal nodes also known as decision node refers to an attribute and each internal node has two or more leaf nodes which corresponds to a class label. The topmost node known as root node corresponds to the best predictor i.e. best attribute of the dataset. This algorithm splits the whole data-frame into parts or subsets and simultaneously a decision tree is developed and the end result of this is a tree with

leaf nodes, internal nodes and a root node. As the tree becomes more deep and more complex, then the model becomes more and more fit.



1.6.1 DISADVANTAGES OF EXISTING SYSTEM

- Several risk scoring systems are used to identify patients at high risk of serious adverse events including unexpected inpatient death
- Accuracy low
- Cardiovascular disease and cancer are leading causes of morbidity and mortality, and can both be present in one patient. In patients with simultaneous disease, the most threatening disease should be treated first.
- Improved depression scores , and decreased health care utilization
- It may increase death rate .

1.6.2 LITERATURE SURVEY

1.TITLE: Prediction of In-Hospital Cardiac Arrest Using Shallow and Deep Learning

Author: Minsu Chae , Sangwook Han , Hyowook Gil , Namjun Cho and Hwamin Lee

Year:2021

Abstract:

Sudden cardiac arrest can leave serious brain damage or lead to death, so it is very important to predict before a cardiac arrest occurs. However, early warning score systems including the National Early Warning Score, are associated with low sensitivity and false positives. We applied shallow and deep learning to predict cardiac arrest to overcome these limitations. We evaluated the performance of the Synthetic Minority Oversampling Technique Ratio. We evaluated the performance using a Decision Tree, a Random Forest, Logistic Regression, Long Short-Term Memory model, Gated Recurrent Unit model, and LSTM–GRU hybrid models. Our proposed Logistic Regression demonstrated a higher positive predictive value and sensitivity than traditional early warning systems.

2.TITLE: Prediction of cardiac arrest in critically ill patients based on bedside vital signs monitoring

Author: LiYijinga,YeWenyua,YangKangaZhang,Shengyua,HeXianlianga,JinXinglianga

Year:2021

Abstract:

Despite advances in healthcare, the incidence of in-hospital cardiac arrest (IHCA) has continued to rise for the past decade. Identifying those patients at risk has proven challenging. Our objective was to conduct a systematic review of the literature to compare the IHCA predictive performance of machine learning (ML) models with the Modified Early Warning Score (MEWS). Design: The systematic review was conducted following the Preferred Reporting Items of Systematic Review and Meta-Analysis guidelines and registered on Method: Data extraction

was completed using the Critical Appraisal and Data Extraction for Systematic Reviews of Prediction Modeling Studies checklist. The risk of bias and applicability were evaluated using the Prediction model Risk of Bias Assessment Tool. Results: Nine articles were included in this review that developed or validated IHCA prediction models and compared them with the MEWS. The studies by Jang et al and Kim et al showed that their ML models outperformed MEWS to predict IHCA with good to excellent predictive performance. Conclusions: The ML models presented in this systematic review demonstrate a novel approach to predicting IHCA. All included studies suggest that ML models had similar or better predictive performance compared with MEWS. However, there is substantial variability in performance measures and concerns for risk of bias.

3.TITLE: An Algorithm Based on Deep Learning for Predicting In-Hospital Cardiac Arrest

Author: Joon-Myoung Kwon, Youngnam Lee

Year:2021

Abstract:

In-hospital cardiac arrest is a major burden to public health, which affects patient safety. Although traditional track-and-trigger systems are used to predict cardiac arrest early, they have limitations, with low sensitivity and high false-alarm rates. We propose a deep learning-based early warning system that shows higher performance than the existing track-and-trigger systems. Methods and results: This retrospective cohort study reviewed patients who were admitted to 2 hospitals from June 2010 to July 2017. A total of 52 131 patients were included. Specifically, a recurrent neural network was trained using data from June 2010 to January 2017. The result was tested using the data from February to July 2017. The primary outcome was cardiac arrest, and the secondary outcome was death without attempted resuscitation. As comparative measures, we used the area under the receiver operating characteristic curve (AUROC), the area under the precision-recall curve (AUPRC), and the net reclassification index. Furthermore, we evaluated sensitivity while varying the number of alarms. The deep learning-based early warning system (AUROC: 0.850; AUPRC: 0.044) significantly outperformed a modified early warning score (AUROC: 0.603; AUPRC: 0.003), a random forest algorithm (AUROC: 0.780; AUPRC: 0.014),

and logistic regression (AUROC: 0.613; AUPRC: 0.007). Furthermore, the deep learning-based early warning system reduced the number of alarms by 82.2%, 13.5%, and 42.1% compared with the modified early warning system, random forest, and logistic regression, respectively, at the same sensitivity. Conclusions: An algorithm based on deep learning had high sensitivity and a low false-alarm rate for detection of patients with cardiac arrest in the multicenter study.

4.TITLE: Machine Learning Models for Survival and Neurological Outcome Prediction of Out-of-Hospital Cardiac Arrest Patients

Author: Chi-Yung Cheng, I-Min Chiu, Wun-Huei Zeng, Chih-Min Tsai and Chun-Hung Richard Lin

Year:2021

Abstract:

Out-of-hospital cardiac arrest (OHCA) is a major health problem worldwide, and neurologic injury remains the leading cause of morbidity and mortality among survivors of OHCA. The purpose of this study was to investigate whether a machine learning algorithm could detect complex dependencies between clinical variables in emergency departments in OHCA survivors and perform reliable predictions of favorable neurologic outcomes. *Methods*. This study included adults (≥18 years of age) with a sustained return of spontaneous circulation after successful resuscitation from OHCA between 1 January 2004 and 31 December 2014. We applied three machine learning algorithms, including logistic regression (LR), support vector machine (SVM), and extreme gradient boosting (XGB). The primary outcome was a favorable neurological outcome at hospital discharge, defined as a Glasgow-Pittsburgh cerebral performance category of 1 to 2. The secondary outcome was a 30-day survival rate and survival-to-discharge rate.

5.TITLE: Machine learning model for predicting out-of-hospital cardiac arrests using meteorological and chronological data

Author: Takahiro Nakashima, Soshiro ,Teruo Noguchi, Yoshio Tahara

Year:2020

Abstract:

To evaluate a predictive model for robust estimation of daily out-of-hospital cardiac arrest (OHCA) incidence using a suite of machine learning (ML) approaches and high-resolution meteorological and chronological data.In this population-based study, we combined an OHCA nationwide registry and high-resolution meteorological and chronological datasets from Japan. We developed a model to predict daily OHCA incidence with a training dataset for 2005–2013 using the eXtreme Gradient Boosting algorithm. A dataset for 2014–2015 was used to test the predictive model. The main outcome was the accuracy of the predictive model for the number of daily OHCA events, based on mean absolute error (MAE) and mean absolute percentage error (MAPE). In general, a model with MAPE less than 10% is considered highly accurate.Among the 1 299 784 OHCA cases, 661 052 OHCA cases of cardiac origin (525 374 cases in the training dataset on which fourfold cross-validation was performed and 135 678 cases in the testing dataset) were included in the analysis. Compared with the ML models using meteorological or chronological variables alone, the ML model with combined meteorological and chronological variables had the highest predictive accuracy in the training (MAE 1.314 and MAPE 7.007%) and testing datasets (MAE 1.547 and MAPE 7.788%). Sunday, Monday, holiday, winter, low ambient temperature and large interday or intraday temperature difference were more strongly associated with OHCA incidence than other the meteorological and chronological variables.

1.7PROPOSED SYSTEM

Cardiac arrest (CA) is the most serious death-related event in critically ill patients and the early detection of CA is beneficial to reduce mortality according to clinical research. This study aims to develop and verify a real-time, interpretable machine learning model, namely cardiac arrest prediction index (CAPI), to predict CA of critically ill patients based on bedside vital signs monitoring.

The cleaned data is split into 80% training and 20% testing for training and testing purposes. The same dataset is tested with different machine learning classifiers such as RF & KNN. In this

paper, we proposed a RF & KNN to predict the accuracy of whether a patient had a cardio arrest or not. 89.91 training accuracy and 86.83% testing accuracy

1.7.1 PROPOSED SYSTEM ADVANTAGES

1. Increased accuracy for effective cardiac arrest diagnosis.
2. Handles roughest (enormous) amount of data using random forest algorithm and feature selection.
3. Reduce the time complexity of doctors.

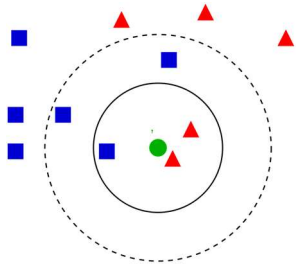
Cost effective for patients

KNN:

Algorithms:

KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry. To evaluate any technique we generally look at 3 important aspects

For instance, will a customer attrite or not, should we target customer X for digital campaigns, whether customer has a high potential or not, etc. These analysis are more insightful and directly linked to an implementation roadmap.



In this article, we will talk about another widely used machine learning classification technique called K-nearest neighbors (KNN). Our focus will be primarily on how does the algorithm work and how does the input parameter affects the output/prediction.

1. Ease to interpret output

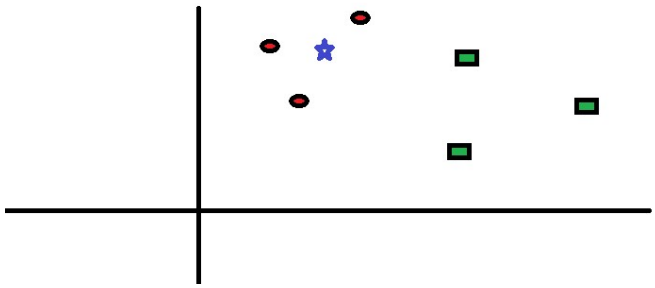
2. Calculation time

3. Predictive Power

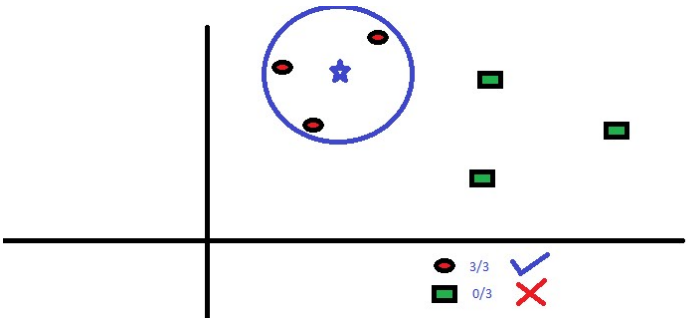
Let us take a few examples to place KNN in the scale :

KNN algorithm fairs across all parameters of considerations. It is commonly used for its easy of interpretation and low calculation time.

How does the KNN algorithm work?
Let's take a simple case to understand this algorithm. Following is a spread of red circles (RC) and green squares (GS) :



You intend to find out the class of the blue star (BS). BS can either be RC or GS and nothing else. The "K" is KNN algorithm is the nearest neighbor we wish to take the vote from. Let's say K = 3. Hence, we will now make a circle with BS as the center just as big as to enclose only three datapoints on the plane. Refer to the following diagram for more details:

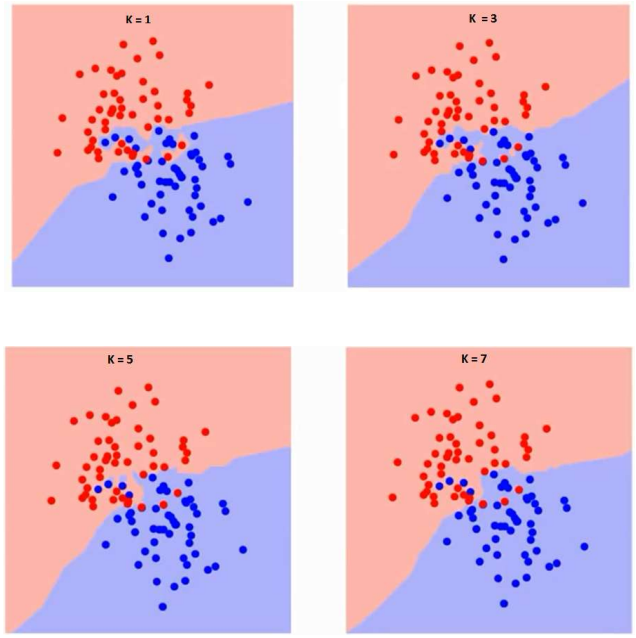


The three closest points to BS is all RC. Hence, with a good confidence level, we can say that the BS should belong to the class RC. Here, the choice became very obvious as all three votes from

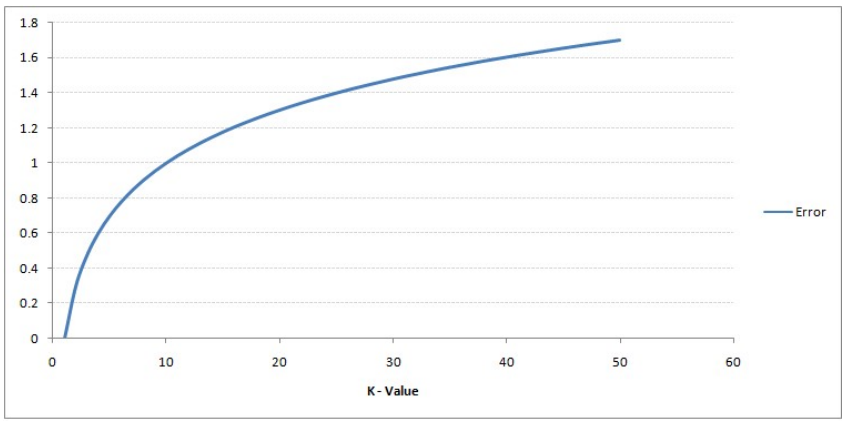
the closest neighbor went to RC. The choice of the parameter K is very crucial in this algorithm. Next, we will understand what are the factors to be considered to conclude the best K.

How do we choose the factor K?

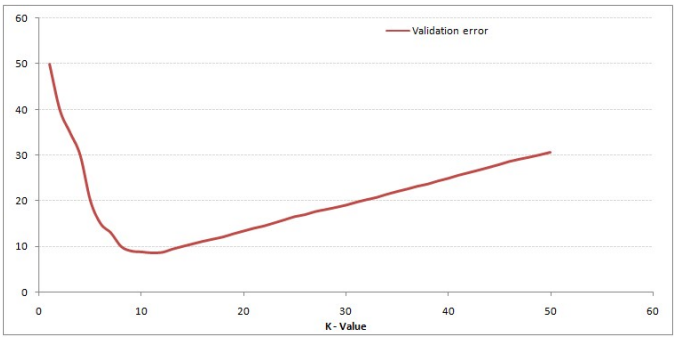
First let us try to understand what exactly does K influence in the algorithm. If we see the last example, given that all the 6 training observation remain constant, with a given K value we can make boundaries of each class. These boundaries will segregate RC from GS. In the same way, let's try to see the effect of value "K" on the class boundaries. The following are the different boundaries separating the two classes with different values of K.



If you watch carefully, you can see that the boundary becomes smoother with increasing value of K. With K increasing to infinity it finally becomes all blue or all red depending on the total majority. The training error rate and the validation error rate are two parameters we need to access different K-value. Following is the curve for the training error rate with a varying value of K :



As you can see, the error rate at K=1 is always zero for the training sample. This is because the closest point to any training data point is itself.Hence the prediction is always accurate with K=1. If validation error curve would have been similar, our choice of K would have been 1. Following is the validation error curve with varying value of K:



This makes the story more clear. At K=1, we were overfitting the boundaries. Hence, error rate initially decreases and reaches a minima. After the minima point, it then increase with increasing K. To get the optimal value of K, you can segregate the training and validation from the initial dataset. Now plot the validation error curve to get the optimal value of K. This value of K should be used for all predictions.

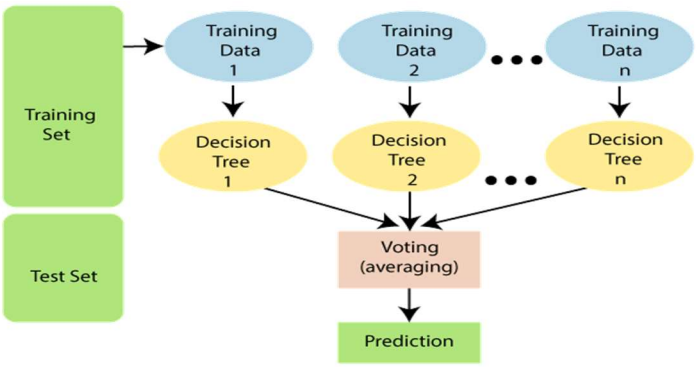
RANDOM FOREST ALGORITHM

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

The below diagram explains the working of the Random Forest algorithm:



Assumptions for Random Forest

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

Why use Random Forest?

Below are some points that explain why we should use the Random Forest algorithm:

<=”” li=””>

- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

How does Random Forest algorithm work?

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

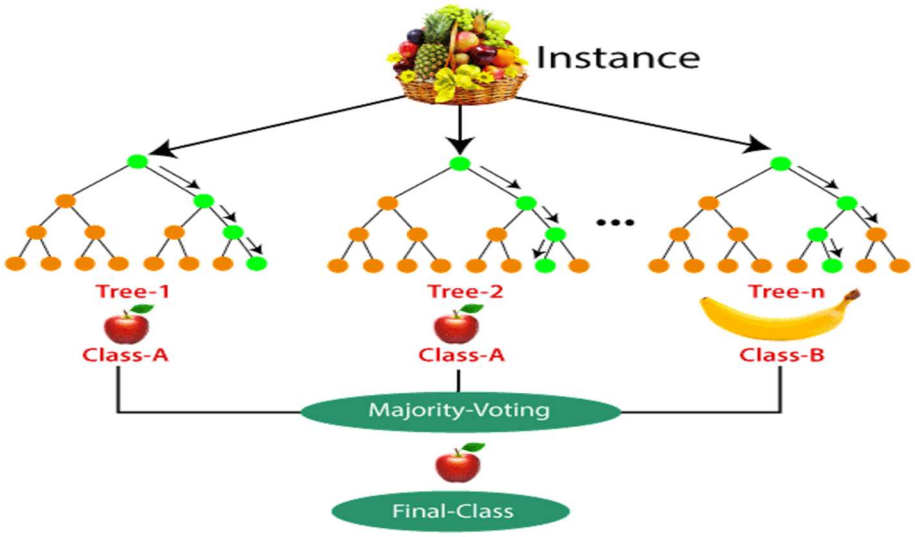
Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

The working of the algorithm can be better understood by the below example:

Example: Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision. Consider the below image:


Instance
Tree-1
Tree-2
Tree-n
Class-A
Class-A
Class-B
Majority-Voting
Final-Class

CHAPTER 2

2.1 INTRODUCTION

Cardiac arrest, also known as sudden cardiac death, is the cessation of the ability of the heart to pump blood. This acute cessation requires immediate intervention, as vital organs, such as the brain and the heart itself, are deprived of blood flow. A delay in intervention can lead to lifelong complications and even death. The global rate of mortality after cardiac arrest is significantly high—78% of out-of-hospital cardiac arrest (OHCA) cases die before they reach the hospital. For those who do receive advanced care, the survival rate remains low. The survival rate for OHCA from the time of cardiac arrest to the time of discharge ranges from 2% to 11% worldwide. The number of cardiac arrest deaths that occur within an in-hospital setting is also significant. In the United States alone, over 290,000 in-hospital cardiac arrests occur annually, with survival rates varying from as low as 0% to 36.2%, out of which a small percentage have favorable neurological prognoses.

Artificial intelligence (AI) is reforming health care every day. AI technologies have the perfect platform to thrive and mature with the growing adoption of electronic health records, development in computational power, continuous monitoring systems, and availability of big data .It has become an important clinical decision-making tool that allows for personalized diagnoses, solutions, prognoses, and predictions of future health outcomes, guiding clinicians and other stakeholders in doing what is best for their patients. AI technology is also rapidly progressing in cardiology, like in any other field of medicine .AI-guided diagnosis and therapy selection have allowed for advancement in research, clinical practice, and population health in cardiovascular medicine. Machine learning (ML) models have also been shown to outperform traditional statistical models in detecting sex differences in cardiovascular disease, further enhancing individualized medicine.AI also plays a major role in improving care for cardiac arrest. AI technologies are being used to prevent cardiac arrest through early identification of risk factors, early detection ,improved management (eg, effective cardiopulmonary resuscitation) and prognosis determination for patients post cardiac arrest. A large part of cardiac arrest research is the prediction of cardiac arrest before its occurrence, as it gives clinicians time to prepare and achieve better patient outcomes.

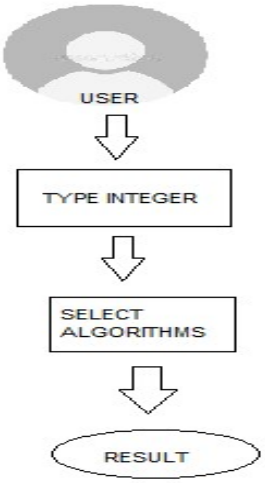
Thus, what are AI technologies and their counterparts in this context? AI refers to the field of science revolving around building computational systems and algorithms that facilitate the

ability of a machine to mimic human behavior to learn and find solutions to tasks autonomously . ML is a subset of AI. ML algorithms focus on building smart solutions after learning from patterns and experiences provided by a structured sample of training data. Deep learning (DL) is a class of ML. It consists of a complex, interconnected, multilayered neural network, resembling a human brain. The aim of DL is to learn and understand patterns from a large amount of unstructured data. In short, the more information it is fed, the more accurate the outcome.

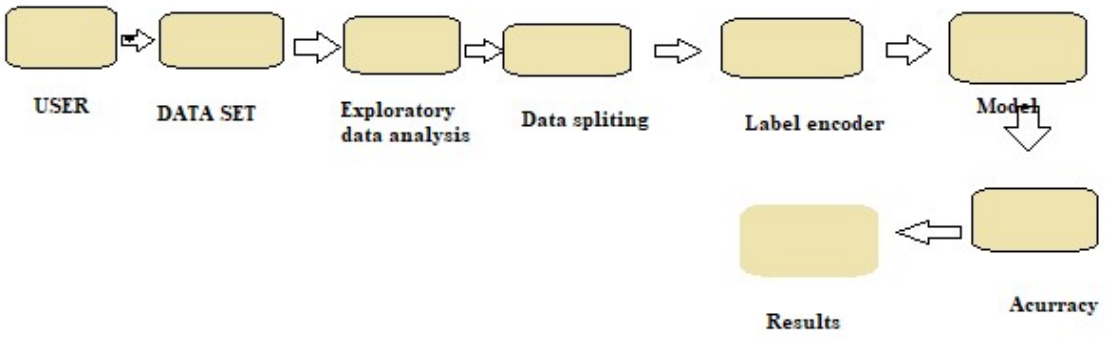
The ability of AI technologies to process and evaluate patient data to generate predictions is important to support clinicians in making critical decisions, provide effective management, and, ultimately, improve patient outcomes in cardiac arrest cases. Therefore, we believe it is crucial to explore the use of AI technology in predicting cardiac arrest and report our findings to help clinicians and researchers.

2.2 DETAILED DIAGRAM

2.2.1Front End Module Diagrams:



2.2.2 Back End Module Diagrams:



2.3 SYSTEM SPECIFICATION:

2.3.1 HARDWARE REQUIREMENTS:

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It shows what the system does and not how it should be implemented

PROCESSOR	:	Intel I5
RAM	:	4GB
HARD DISK	:	40 GB

2.3.2 SOFTWARE REQUIREMENTS:

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the team's and tracking the team's progress throughout the development activity.

PYTHON IDE : Anaconda Jupyter Notebook

PROGRAMMING LANGUAGE : Python

2.4 MODULE DESCRIPTION

2.4.1 DATASET

Patient data collected from 2011 to 2017 at the cardiovascular ward of the Mostar hospital were used to create the model. The data set contained 507 patients with cardiovascular disease, of whom 123 died and 384 survived over a 12-month period. The dataset included 71 attributes and two specials, namely the patient ID and the label attribute Vital status (12 Month FU), which contains the two classes Alive and Dead.

The rest of the attributes are: Date visit hospital, Type of patient, Date of birth, Age, Sex, Height, Weight, Body Mass Index (BMI), Home situation, Heart Failure History (HFH), Heart Rate (HR), Systolic Blood Pressure (SBP), Diastolic Blood Pressure (DBP), Primary Etiology (PE), Smoking, Previous Atrial Fibrillation (PAF), Diabetes, Alcohol, Physical Activities (PA), Myocardial Infarction Angina (MI/Angina), Coronary Artery Bypass Graft (CABG), Percutaneous Coronary Intervention (PCI), Transient Ischemic Attack (Stroke/TIA), Peripheral Vascular Disease (PVD)

DATA CLEANING:

In this module the data is cleaned. After cleaning of the data, the data is grouped as per requirement. This grouping of data is known as data clustering. Then check if there is any missing value in the data set or not. It there is some missing value then change it by any default value. After that if any data need to change its format, it is done. That total process before the prediction is known is data pre-processing.

2.4.2 DATA PREPROCESSING

In this step, the pre-processed data is taken for the prediction. This prediction can be done in any process which are mentioned above. For that, the pre-processed data is splitted for the train and test purpose. Then a predictive object is created to predict the test value which is trained by the trained value. Then the object is used to forecast data for next few years.

2.4.3 DATA SPLITTING:

For each experiment, we split the entire dataset into 70% training set and 30% test set. We used the training set for resampling, hyper parameter tuning, and training the model and we used test set to test the performance of the trained model. While splitting the data, we specified a random seed (any random number), which ensured the same data split every time the program executed.

2.4.4 TRAINING AND TESTING:

Algorithms learn from data. They find relationships, develop understanding, make decisions, and evaluate their confidence from the training data they're given. And the better the training data is, the better the model performs.

In fact, the quality and quantity of your training data has as much to do with the success of your data project as the algorithms themselves.

Now, even if you've stored a vast amount of well-structured data, it might not be labeled in a way that actually works for training your model. For example, autonomous vehicles don't just need pictures of the road, they need labeled images where each car, pedestrian, street sign and more are annotated; sentiment analysis projects require labels that help an algorithm understand when someone's using slang or sarcasm; chatbots need entity extraction and careful syntactic analysis, not just raw language.

In other words, the data you want to use for training usually needs to be enriched or labeled. Or you might just need to collect more of it to power your algorithms. But chances are, the data you've stored isn't quite ready to be used to train your classifiers.

Because if you're trying to make a great model, you need great training data. **Classifier Training**:

A classifier is a function that takes features as input and generates a class label prediction. Based on the learning function and underlying assumptions, different types of classifiers can be developed.

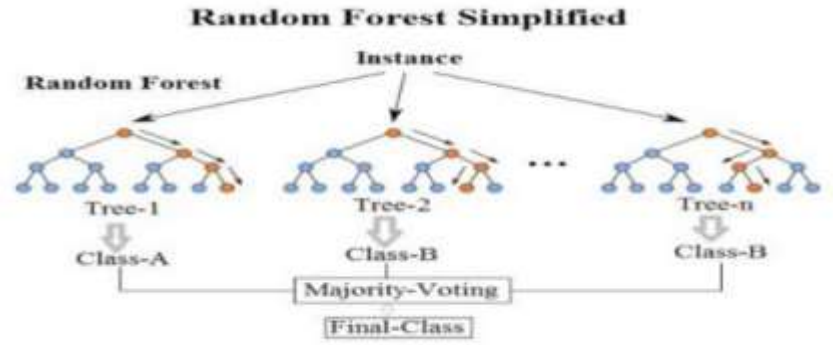
Neuroimaging studies have applied various classifiers for mental illness prediction. The dimensionality issue associated with the relatively large number of features and the small number of samples should be

Random Forest:

Random forests are the classifiers that combine many tree possibilities, where each trees are depends on the values of a random vector sampled independently. then, all trees in the forest will have same allotment. To construct a tree, we assume that n is the number of training observations and p is the number of variables (features) in a training set. To determine the decision node at a tree we choose k « p as the number of variables to be selected. We select a bootstrap sample from the n observations in the training set and use the rest of the observations to estimate the error of the tree in testing phase. hence, we randomly choose 'k' variables as a decision at certain node in the tree and calculate the best split based on the k variables in the training set. Trees are always grown and never pruned compared to other tree algorithms. Random forests can handle large number of variables in a data set. Also, during the forest building process they generate an internal unbiased estimate of the generalization error. additionally, they can estimate missing data closely. A major disadvantage of random forests algorithm is it does not gives precise continous forecast.

Random Forest algorithm is a machine learning based algorithm that combines multiple decision trees together for obtaining efficient outcome. Decision trees are created by random forest algorithm based on data samples and selects the best solution by means of voting

Random Forest algorithms are used for classification as well as regression. It creates a tree for the data and makes prediction based on that. Random Forest algorithm can be used on large datasets and can produce the same result even when large sets record values are missing. The generated samples from the decision tree can be saved so that it can be used on other data. In random forest there are two stages, firstly create a random forest then make a prediction using a random forest classifier created in the first stage.



The random forest is a supervised learning algorithm that randomly creates and merges multiple decision trees into one "forest." The goal is not to rely on a single learning model, but rather a collection of decision models to improve accuracy. The primary difference between this approach and the standard decision tree algorithm is that the root nodes feature splitting nodes are generated randomly.

K- NEAREST NEIGHBOUR (KNN):

K-nearest neighbors (KNN) algorithm uses 'feature similarity' to predict the values of new data points which further means that the new data point will be assigned a value based on how closely it matches the points in the training set. We can understand its working with the help of following steps −

Step 1 − For implementing any algorithm, we need data-set. So during the first step of KNN, we must load the training as well as test data.

Step 2 − Next, we need to choose the value of K i.e. the nearest data points. K can be any integer.

Step 3 − For each point in the test data do the following

3.1 − Calculate the distance between test data and each row of training data with the help of any of the method namely:

Euclidean, Manhattan or Hamming distance. The most commonly used method to calculate distance is Euclidean.

3.2 − Now, based on the distance value, sort them in ascending order.

3.3 − Next, it will choose the top K rows from the sorted array.

3.4 − Now, it will assign a class to the test point based on most frequent class of these rows.

Step 4 - End

PERFORMANCE MATRICES:

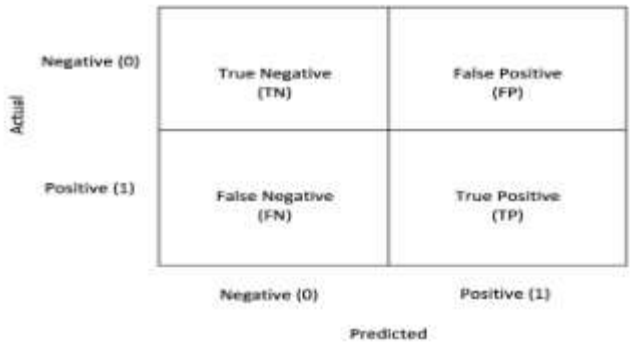
Data was divided into two portions, training data and testing data, both these portions consisting 70% and 30% data respectively. All these six algorithms were applied on same dataset using Enthought Canaopy and results were obtained.

$$\textbf{Accuracy} = (TP+TN) / (P + N)$$

Predicting accuracy is the main evaluation parameter that we used in this work. Accuracy can be defied using equation. Accuracy is the overall success rate of the algorithm.

CONFUSION MATRIX:

It is the most commonly used evaluation metrics in predictive analysis mainly because it is very easy to understand and it can be used to compute other essential metrics such as accuracy, recall, precision, etc. It is an NxN matrix that describes the overall performance of a model when used on some dataset, where N is the number of class labels in the classification problem.



All predicted true positive and true negative divided by all positive and negative. True Positive (TP), True Negative (TN), False Negative (FN) and False Positive (FP) predicted by all algorithms are presented in table.

True positive (TP) indicates that the positive class is predicted as a positive class, and the number of sample positive classes was actually predicted by the model.

False negative indicates (FN) that the positive class is predicted as a negative class, and the number of negative classes in the sample was actually predicted by the model.

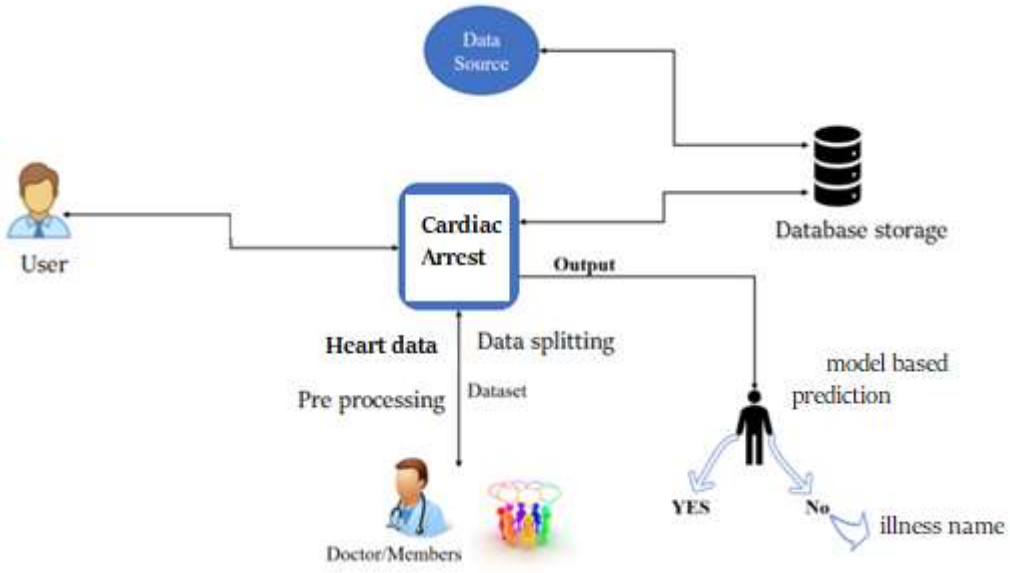
False positive (FP) indicates that the negative class is predicted as a positive class, and the number of positive classes of samples was actually predicted by the model.

True negative (TN) indicates that the negative class is predicted as a negative class, and the number of sample negative classes was actually predicted by the model.

2.5 SYSTEM DESIGN:

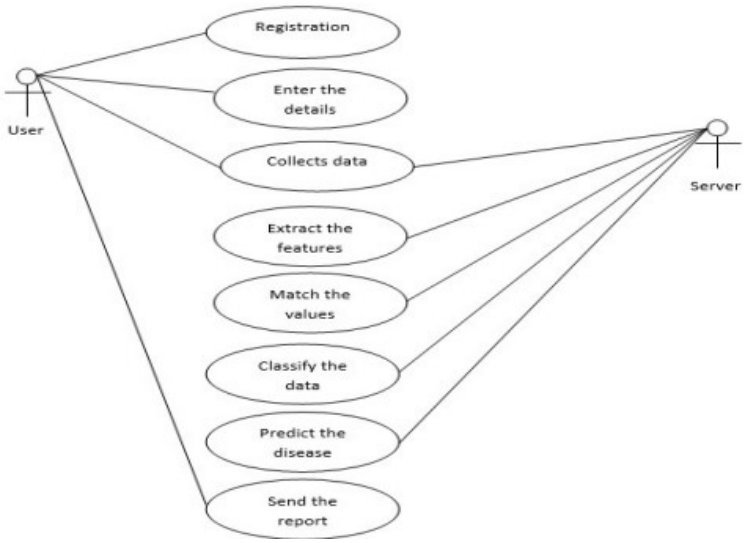
Designing of system is the process in which it is used to define the interface, modules and data for a system to specified the demand to satisfy. System design is seen as the application of the system theory. The main thing of the design a system is to develop the system architecture by giving the data and information that is necessary for the implementation of a system.

2.5.1 SYSTEM ARCHITECTURE:



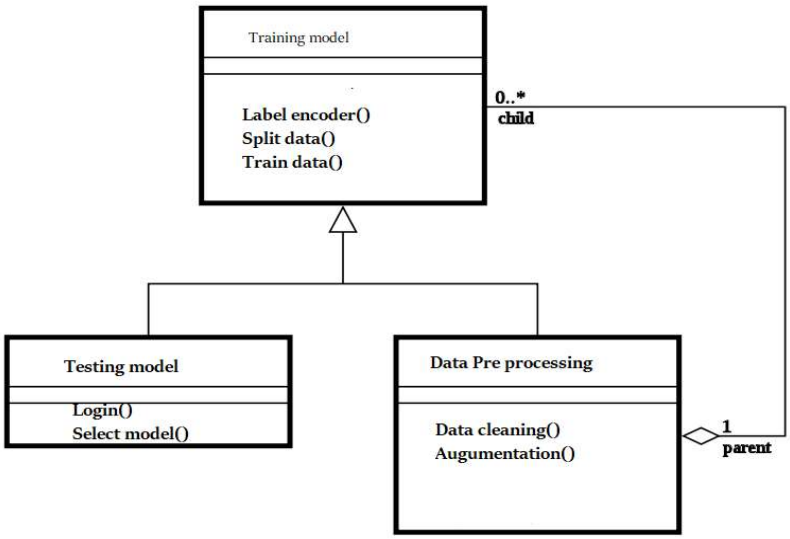
2.5.2 Use case Diagram:

Use case diagrams identify the functionalities provides by the use cases, the actors who interact with the system and the association between the actors and the functionalities.



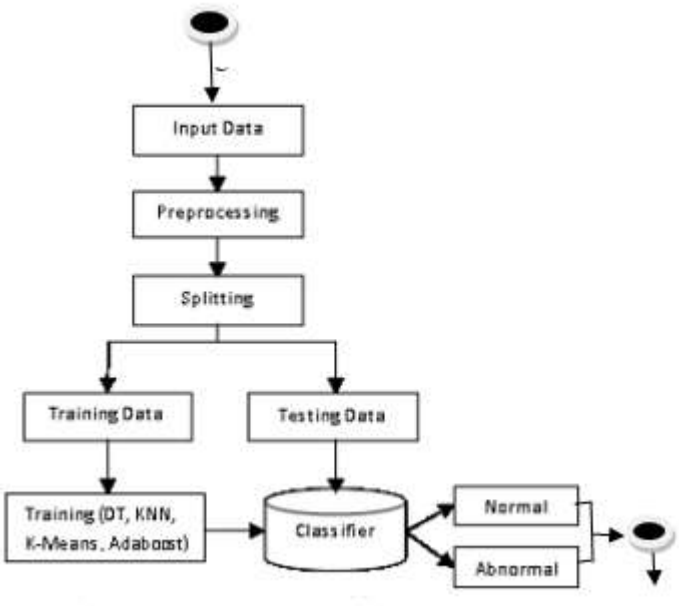
2.5.3 CLASS DIAGRAM:

The class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing and documenting different aspects of a system but also for constructing executable code of the software application

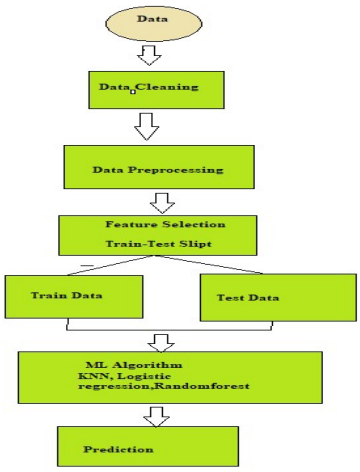


2.5.4 Activity of Diagram:

The Activity Diagram forms effective while modeling the functionality of the system. Hence this diagram reflects the activities, the types of flows between these activities and finally the response of objects to these activities.

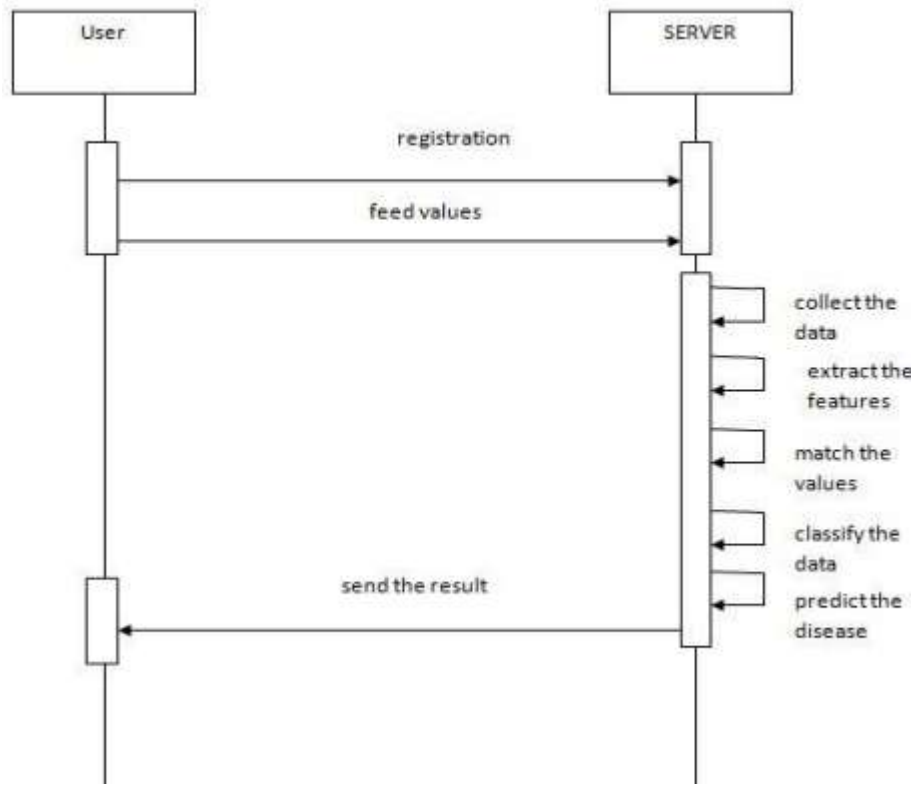


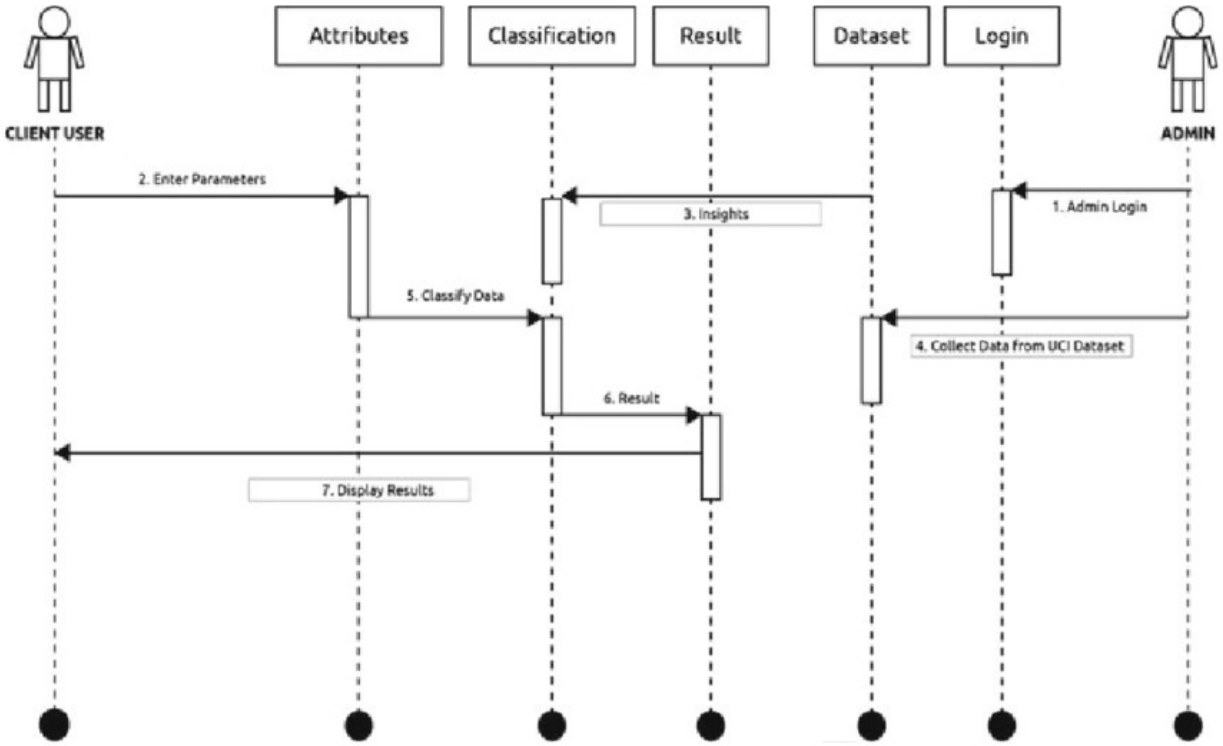
DATA FLOW DIAGRAM:



2.5.5 SEQUENCE Diagram:

The sequence diagram of a system shows the entity interplay are ordered in the time order level. So, that it drafts the classes and object that are imply in the that plot and also the series of message exchange take place betwixt the body that need to be carried out by the purpose of that scenario.




Attributes
Classification
Result
Dataset
Login
CLIENT USER
ADMIN
2. Enter Parameters
3. Insights
1. Admin Login
5. Classify Data
4. Collect Data from UCI Dataset
6. Result
7. Display Results

CHAPTER 3

SOFTWARE SPECIFICATION

3.1 GENERAL

3.2ANACONDA

It is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.

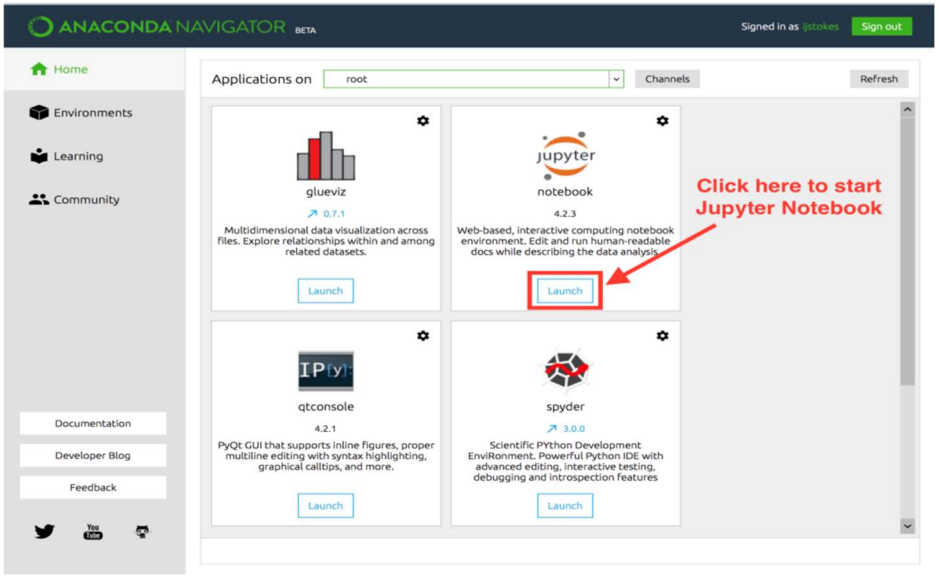
Anaconda distribution comes with more than 1,500 packages as well as the Conda package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the Command Line Interface (CLI).

The big difference between Conda and the pip package manager is in how package dependencies are managed, which is a significant challenge for Python data science and the reason Conda exists. Pip installs all Python package dependencies required, whether or not those conflict with other packages you installed previously.

So your working installation of, for example, Google Tensorflow, can suddenly stop working when you pip install a different package that needs a different version of the Numpy library. More insidiously, everything might still appear to work but now you get different results from your data science, or you are unable to reproduce the same results elsewhere because you didn't pip install in the same order.

Conda analyzes your current environment, everything you have installed, any version limitations you specify (e.g. you only want tensorflow>= 2.0) and figures out how to install compatible dependencies. Or it will tell you that what you want can't be done. Pip, by contrast, will just install the thing you wanted and any dependencies, even if that breaks other things.Open source packages can be individually installed from the Anaconda repository, Anaconda Cloud (anaconda.org), or your own private repository or mirror, using the conda install command. Anaconda Inc compiles and builds all the packages in the Anaconda repository itself, and provides binaries for Windows 32/64 bit, Linux 64 bit and MacOS 64-bit. You can also install

anything on PyPI into a Conda environment using pip, and Conda knows what it has installed and what pip has installed. Custom packages can be made using the conda build command, and can be shared with others by uploading them to Anaconda Cloud, PyPI or other repositories.The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, you can create new environments that include any version of Python packaged with conda.



Anaconda Navigator is a desktop Graphical User Interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, macOS and Linux.

The following applications are available by default in Navigator:

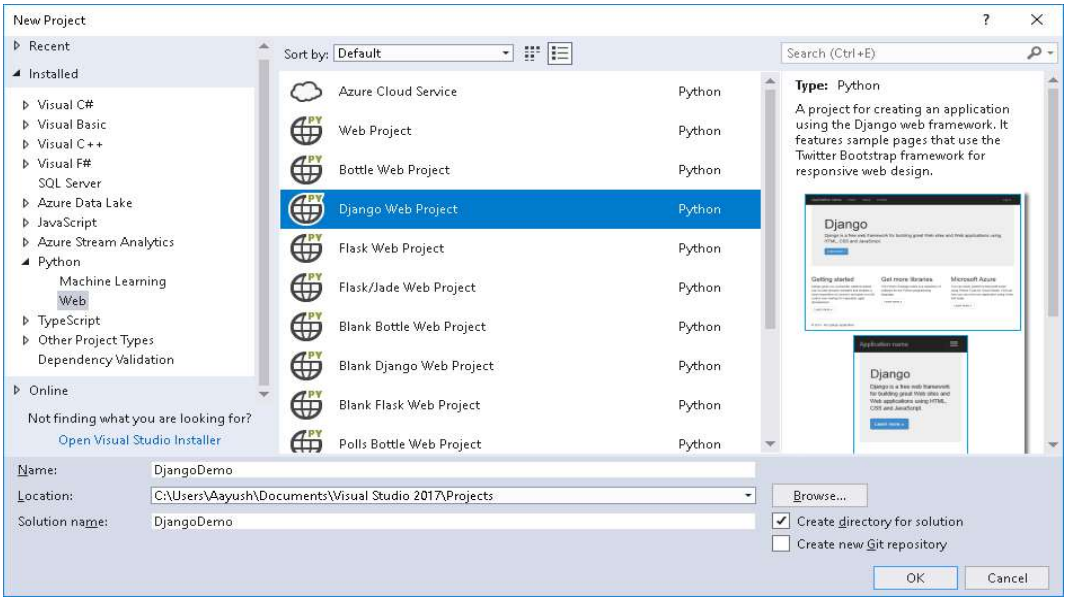
- JupyterLab
- Jupyter Notebook
- QtConsole
- Spyder

- Glueviz
- Orange
- Rstudio
- Visual Studio Code

Microsoft .NET is a set of Microsoft software technologies for rapidly building and integrating XML Web services, Microsoft Windows-based applications, and Web solutions. The .NET Framework is a language-neutral platform for writing programs that can easily and securely interoperate. There's no language barrier with .NET: there are numerous languages available to the developer including Managed C++, C#, Visual Basic and Java Script. The .NET framework provides the foundation for components to interact seamlessly, whether locally or remotely on different platforms. It standardizes common data types and communications protocols so that components created in different languages can easily interoperate.

".NET" is also the collective name given to various software components built upon the .NET platform. These will be both products (Visual Studio.NET and Windows.NET Server, for instance) and services (like Passport, .NET My Services, and so on).

Microsoft VISUAL STUDIO is an Integrated Development Environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps.



Python is a powerful multi-purpose programming language created by Guido van Rossum. It has simple easy-to-use syntax, making it the perfect language for someone trying to learn computer programming for the first time. Python features are:

- Easy to code
- Free and Open Source
- Object-Oriented Language
- GUI Programming Support
- High-Level Language
- Extensible feature
- Python is Portable language
- Python is Integrated language
- Interpreted
- Large Standard Library
- Dynamically Typed Language

3.3 PYTHON:

- Python is a powerful multi-purpose programming language created by Guido van Rossum.
- It has simple easy-to-use syntax, making it the perfect language for someone trying to learn computer programming for the first time.

Features Of Python :

1.Easy to code:
Python is high level programming language. Python is very easy to learn language as compared to other language like c, c#, java script, java etc. It is very easy to code in python language and anybody can learn python basic in few hours or days. It is also developer-friendly language.

2. Free and Open Source:
Python language is freely available at official website and you can download it from the given download link below click on the Download Python keyword.
Since, it is open-source, this means that source code is also available to the public. So you can download it as, use it as well as share it.

3.Object-Oriented Language:
One of the key features of python is Object-Oriented programming. Python supports object oriented language and concepts of classes, objects encapsulation etc.

4. GUI Programming Support:
Graphical Users interfaces can be made using a module such as PyQt5, PyQt4, wxPython or Tk

in python.
PyQt5 is the most popular option for creating graphical apps with Python.

5. High-Level Language:
Python is a high-level language. When we write programs in python, we do not need to remember the system architecture, nor do we need to manage the memory.

6.Extensible feature:
Python is a Extensible language. we can write our some python code into c or c++ language and also we can compile that code in c/c++ language.

7. Python is Portable language:
Python language is also a portable language. for example, if we have python code for windows and if we want to run this code on other platform such as Linux, Unix and Mac then we do not need to change it, we can run this code on any platform.

8. Python is Integrated language:
Python is also an Integrated language because we can easily integrated python with other language like c, c++ etc.

9. Interpreted Language:
Python is an Interpreted Language. because python code is executed line by line at a time. like other language c, c++, java etc there is no need to compile python code this makes it easier to debug our code. The source code of python is converted into an immediate form called bytecode.

10. Large Standard Library
Python has a large standard library which provides rich set of module and functions so you do not have to write your own code for every single thing.There are many libraries present in python for such as regular expressions, unit-testing, web browsers etc.

11. Dynamically Typed Language:
Python is dynamically-typed language. That means the type (for example- int, double, long etc) for a variable is decided at run time not in advance.because of this feature we don't need to specify the type of variable.

APPLICATIONS OF PYTHON :

WEB APPLICATIONS

- You can create scalable Web Apps using frameworks and CMS (Content Management System) that are built on Python. Some of the popular platforms for creating Web Apps are:Django, Flask, Pyramid, Plone, Django CMS.

- Sites like Mozilla, Reddit, Instagram and PBS are written in Python.

3.3.1SCIENTIFIC AND NUMERIC COMPUTING

- There are numerous libraries available in Python for scientific and numeric computing. There are libraries like:SciPy and NumPy that are used in general purpose computing. And, there are specific libraries like: EarthPy for earth science, AstroPy for Astronomy and so on.
- Also, the language is heavily used in machine learning, data mining and deep learning.

3.3.2 CREATING SOFTWARE PROTOTYPES

- Python is slow compared to compiled languages like C++ and Java. It might not be a good choice if resources are limited and efficiency is a must.
- However, Python is a great language for creating prototypes. For example: You can use Pygame (library for creating games) to create your game's prototype first. If you like the prototype, you can use language like C++ to create the actual game.
-

3.3.3GOOD LANGUAGE TO TEACH PROGRAMMING

- Python is used by many companies to teach programming to kids
- It is a good language with a lot of features and capabilities. Yet, it's one of the easiest language to learn because of its simple easy-to-use system.

CHAPTER 4

IMPLEMENTATION

4.1 GENERAL

Python is a program that was originally designed to simplify the implementation of numerical linear algebra routines. It has since grown into something much bigger, and it is used to implement numerical algorithms for a wide range of applications. The basic language used is very similar to standard linear algebra notation, but there are a few extensions that will likely cause you some problems at first.

4.2 CODE IMPLEMENTATION

Back End Code:-

CHAPTER 5

5.1CONCLUSION AND REFERENCES

Machine learning is one of the most exciting technologies that one would have ever come across. It can be explained as automating and improving the learning process of computers based on their experiences without any human assistance. We apply classifiers to detect IHCA by combining static and dynamic features. We use two tasks to test all models. For The best classifier is random forest totally and the AUROC value is 0.88.

5.2APPLICATION:

It is use hospital sectors

It is use to make a advance technology for future purposes in Medical Farm

5.3REFERENCES:

[1] J. M. Kwon, Y. Lee, Y. Lee, S. Lee, and J. Park, "An Algorithm Based on Deep Learning for Predicting In - Hospital Cardiac Arrest," *Journal of the American Heart Association*, vol. 7, no. 13, e008678, 2018.

[2] L. J. Blackhall, "Must we always use CPR," *New England Journal Medicine*, vol. 317, no. 20, pp. 1281-1285, 1987.

[3] W. B. Kouwenhoven, J. R. Jude, and G. G. Knickerbocker, "Closedchest cardiac massage," *JAMA*, vol. 173, no. 10, pp. 1064-1067, 1960.

[4] D. L. Atkins, et al., "Part 11: pediatric basic life support and cardiopulmonary resuscitation quality: 2015 American Heart Association guidelines update for cardiopulmonary resuscitation and emergency cardiovascular care," *Circulation*, 132(18 suppl 2), pp. S519-S525.

[5] D. Kindermann, R. Mutter, and J. M. Pines, "Emergency Department Transfers to Acute Care Facilities," *HCUP Statistical Brief* #157, 2013.

[6] C. J. Ng, Z. S. Yen, J. C. H. Tsai, L. C. Chen, S. J. Lin, Y. Y. Sang, and J. C. Chen, "Validation of the Taiwan triage and acuity scale: a new computerised five-level triage system," *Emerg Med J*, vol. 28, no. 12, pp. 1026-1031, 2011.

[7] C. Lin, Y. Zhangy, J. Ivy, M. Capan, R. Arnold, J. M. Huddleston, and M. Chi, "Early Diagnosis and Prediction of Sepsis Shock by Combining Static and Dynamic Information Using Convolutional- LSTM," in *Healthcare Informatics (ICHI), 2018 IEEE International Conference on.* IEEE, 2018, pp. 219-228.

[8] C. Esteban, O. Staeck, S. Baier, Y. Yang, and V. Tresp, "Predicting clinical events by combining static and dynamic information using recurrent neural networks," in *Healthcare Informatics (ICHI), 2016 IEEE International Conference on.* IEEE, 2016, pp. 93-101.

[9] X. Guo, Y. Yin, C. Dong, G. Yang, and G. Zhou, "On the class imbalance problem," in *Natural Computation, 2008. ICNC'08. Fourth International Conference on (Vol. 4, pp. 192-201)*. IEEE.

[10] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas, "Handling imbalanced datasets: A review," *GESTS International Transactions on Computer Science and Engineering*, vol. 30, no. 1, pp. 25-36, 2006.

[11] S. Visa and A. Ralescu, "Issues in mining imbalanced data sets-a review paper," in *Proceedings of the sixteen midwest artificial intelligence and cognitive science conference* (Vol. 2005, pp. 67-73). sn.

[12] Y. Sun, A. K. Wong, and M. S. Kamel, "Classification of imbalanced data: A review," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 23, no.04, pp. 687-719, 2009.