

EFFECTIVE HEART DISEASE PREDICTION USING HYBRID MACHINE LEARNING TECHNIQUES

Submitted in partial fulfillment of the requirements for the
award of Bachelor of Computer Science

By

K.Madhavan (Reg.No : 38290043)

A.Ephean Raj (Reg.No : 38290017)



DEPARTMENT OF COMPUTER SCIENCE
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAR NAGAR, RAJIV GANDHI SALAI , CHENNAI -600 119

APRIL-2021



SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

JEPPIAR NAGAR, RAJIV GANDHI SALAI, CHENNAI-600 119.

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bonafide work of K.Madhavan (Reg.No : 38290043) and A.Ephean Raj (Reg.No : 38290017) , who carried out the project entitled "EFFECTIVE HEART DISEASE PREDICTION USING HYBRID MACHINE LEARNING TECHNIQUES " under my Supervision from November 2020 - April 2021.

Internal Guide

Mrs.J.REFONAA ,M.E.,(Ph.D)

Head Of the Department

Dr.S.VIGNESHWARI,M.E.,Ph.D

Submitted for Viva voce presentation held on

Internal Examiner

External Examiner

DECLARATION

I K.Madhavan (Reg.No : 38290043) hereby declare that the Professional training project entitled “ EFFECTIVE HEART DISEASE PREDICTION USING HYBRID MACHINE LEARNING TECHNIQUES” done by me under the guidance of Mrs. J. REFONAA, M.E., Ph.D is submitted in the partial fulfilment of the requirements for the award of Bachelor of Science degree in Computer Science.

DATE:

PLACE:CHENNAI

SIGNATURE OFCANDIDATE:

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management Of SATHYABAMA** for their kind encouragement in doing this project and for completing successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA ,M.E., Ph.D., Dean** school of Computer Science and **Dr. S. Vigneshwari., M.E., Ph.D** and **Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** , for providing me necessary support and details at the right time during theprogressivereviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Mrs.J. REFONAA, M.E, Ph.D.,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-Teaching staff members of **DEPARTMENT OF COMPUTER SCIENCE** who were helpful in many ways for the completion of the project.

ABSTRACT:

Heart disease is one of the most significant causes of mortality in the world today. Prediction of cardiovascular disease is a critical challenge in the area of clinical data analysis. Machine learning (ML) has been shown to be effective in assisting in making decisions and predictions from the large quantity of data produced by the healthcare industry. We have also seen ML techniques being used in recent developments in different areas of the Internet of Things (IoT). Various studies give only a glimpse into predicting heart disease with ML techniques. In this paper, we propose a novel method that aims at finding significant features by applying machine learning techniques resulting in improving the accuracy in the prediction of cardiovascular disease. The prediction model is introduced with different combinations of features and several known classification techniques. We produce an enhanced performance level with an accuracy level of 88.7% through the prediction model for heart disease with the hybrid random forest with a linear model (HRFLM).

LIST OF ABBREVIATIONS

- 1.** KNN – K NearestNeighbor
- 2.** DT – DecisionTree
- 3.** NB - NaïveBayes
- 4.** LBBB – Left Bundle BranchBlock
- 5.** RBBB – Right Bundle BranchBlock
- 6.** PVC – Premature VentricularContraction
- 7.** PSO – Particle SwarmOptimization
- 8.** R-CNN – Regional based ConvolutionalNetwork
- 9.** SVM – Support VectorMachine

TABLE OF CONTENTS

CHAPTERNO	TITLE	PAGE NO
	ABSTRACT	I
	LIST OF ABBREVIATIONS	II
1	INTRODUCTION	1
	About the Project	3
2	LITERATURE SURVEY	
	Review of Literature work	8
	Existing System	10
	Proposed System	10
3	AIM & SCOPE OF THE PROJECT	12
4	EXPERIMENTAL METHODS AND	
	ALGORITHM	
	Hardware &Software Requirements	13
	Technologies used	13
	Algorithms	14
5	DESIGN &IMPLEMENTATION	

	Design &Implementation Constraint	62
	Non-Functional Requirements	63
6	SYSTEM DESIGN	
	Architecture Diagram	65
	Sequence Diagram	65
	UseCase Diagram	66
	Activity Diagram	68
	Collaboration Diagram	69
7	MODULES	
	Module Description	71
8	CODING AND TESTING	
	Coding	73
	Coding Standards	73
	Test Procedure	75
	Test Data and Output	76
	Testing Techniques	79
9	RESULTS&DISCUSSION	85
10	CONCLUSION &FUTUREWORK	91
	REFERENCES	93
	APPENDIX	
	A.SOURCECODE	96

CHAPTER – 1

INTRODUCTION

It is difficult to identify heart disease because of several contributory risk factors such as diabetes, high blood pressure, high cholesterol, abnormal pulse rate and many other factors. Various techniques in data mining and neural networks have been employed to find out the severity of heart disease among humans. The severity of the disease is classified based on various methods like K-Nearest Neighbor Algorithm (KNN), Decision Trees (DT), Genetic algorithm (GA), and Naïve Bayes (NB).

SYNOPSIS:

The nature of heart disease is complex and hence, the disease must be handled carefully. Not doing so may affect the heart or cause premature death. The perspective of medical science and data mining are used for discovering various sorts of metabolic syndromes. Data mining with classification plays a significant role in the prediction of heart disease and data investigation. We have also seen decision trees be used in predicting the accuracy of events related to heart disease. Various methods have been used for

knowledge abstraction by using known methods of data mining for prediction of heart disease. In this work, numerous readings have been carried out to produce a prediction model using not only distinct techniques but also by relating two or more techniques. These amalgamated new techniques are commonly known as hybrid methods. We introduce neural networks using heart rate time series. This method uses various clinical records for prediction such as Left bundle branch block (LBBB), Right bundle branch block (RBBB), Atrial Fibrillation (AFIB), Normal Sinus Rhythm (NSR), Sinus bradycardia (SBR), Atrial Flutter (AFL), Premature Ventricular Contraction (PVC) and Second degree block (BII) to find out the exact condition of the patient in relation to heart disease. The dataset with a radial basis function network (RBFN) is used for classification, where 70% of the data is used for training and the remaining 30% is used for classification. We also introduce Computer Aided Decision Support System (CADSS) in the field of medicine and research. In previous work, the usage of data mining techniques in the healthcare industry has been shown to take less time for the prediction of disease with more accurate results. We propose the diagnosis of heart disease using the GA. This method uses effective association rules inferred with the GA for tournament selection, crossover and the mutation which results in the new proposed fitness function. For experimental validation, we use the well-known Cleveland dataset which is collected from a UCI machine learning repository. We will see later on how our results prove to be prominent when compared to some of the known supervised learning techniques. The most powerful evolutionary algorithm Particle Swarm Optimization (PSO) is introduced and some rules are generated for heart disease. The rules

have been applied randomly within coding techniques which result in improvement of the accuracy overall Heart disease is predicted based on symptoms namely, pulse rate, sex, age, and many others

About Python:

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

Uses of Python:

- Web development(server-side)
- Software development
- Mathematics
- Systems scripting.

What can Python do?

- ◆ Python can be used on a server to create web applications.
- ◆ Python can be used alongside software to create workflows.

- ◆ Python can connect to database systems. It can also read and modify files.
- ◆ Python can be used to handle big data and perform complex mathematics.
- ◆ Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-oriented way or a functional way.

Good to know

The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.

Python 2.0 was released in 2000, and the 2.x versions were the prevalent releases until December 2008. At that time, the development team made the decision to release version 3.0, which contained a few relatively small but significant changes that were not backward compatible with the 2.x versions. Python 2 and 3 are very similar, and some features of Python 3 have been back ported to Python 2. But in general, they remain not quite compatible.

Both Python 2 and 3 have continued to be maintained and developed, with periodic release updates for both. As of this writing, the most recent versions available are 2.7.15 and 3.6.5. However, an official End of Life date of January 1, 2020 has been established for Python 2, after which time it will no longer be maintained.

Python is still maintained by a core development team at the Institute, and Guido is still in charge, having been given the title of BDFL (Benevolent Dictator For Life) by the Python community. The name Python, by the way, derives not from the snake, but from the British comedy troupe Monty Python's Flying Circus, of which Guido was, and presumably still is, a fan. It is common to find references to Monty Python sketches and movies scattered throughout the Python documentation.

It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

Python Syntax compared to other programming languages

Python was designed to for readability, and has some similarities to the English language with influence from mathematics. Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses. Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Python An Interpreter:

Many languages are compiled, meaning the source code you create needs to be translated into machine code, the language of your computer's processor, before it can be run. Programs written in an interpreted language are passed straight to an interpreter that runs them directly. This makes for a quicker development cycle because you just type in your code and run it, without the intermediate compilation step.

One potential downside to interpreted languages is execution speed. Programs that are compiled into the native language of the computer processor tend to run more quickly than interpreted programs. For some applications that are particularly computationally intensive, like graphics processing or intense number crunching, this can be limiting.

In practice, however, for most programs, the difference in execution speed is measured in milliseconds, or seconds at most, and not appreciably noticeable to a human user. The expediency of coding in an interpreted language is typically worth it for most applications. For all its syntactical simplicity, Python supports most constructs that would be expected in a very high-level language, including complex dynamic data types, structured and functional programming, and object-oriented programming. Additionally, a very extensive library of classes and functions is available that provides capability well beyond what is built into the language, such as database manipulation or GUI programming.

Python accomplishes what many programming languages don't: the language itself is simply designed, but it is very versatile in terms of what you can accomplish with it.

CHAPTER - 2

LITERATURE

SURVEY

The objective of the project is to improve the accuracy in the prediction of heart disease using hybrid machine learning techniques.

Review on literatureWork

P.Sivakumar,D.Anand,V.Uday kumar,Debnath Bhattach aryya:2016 Data Mining algorithm such as Neural networks,Naive Bayes, generic alogirthm. The Effective way to tackle the risks,integrates the strength of various techniques. Only 80% of accuracycan be achieved, lack of data security. In that case, patient may beinter-related with the symptoms in medical diagnosis may not able to diagnose.[1]

Kaan Uyar,Ahmet Ilhan: 2017 Filtering process and fuzzy cluster algorithm. Validates the principles of embedded system and promotes the maximum possible efficiency. Hardware limitations i.e : lack of memory management, absence of operational system. [2]

Reddy Prasad, Pidaparthi,Anjali,S. Adil,N.Deepa: 2019 Logistic Regression (LR) using Machine Learning algorithm. Solve privacy violation problem with high accuracy. Takes more speed, time for training and testing, discreate data will lead to other problem.[3]

Yii-Ching Lee,Chih-Hsuan,Yi-chen Lin,Hsin-Hung: 2017 Apriori algorithm applying WEKA and MATLAB software. High performance which offers better accuracy that influence the degree

of burnout between the demographic variables. Very slow and desires candidate generation overtime. Direct contact with Patient feels like stressful to work all the days that may also result infection to others.[4]

Madhura Patio,Rims Jadhav,Vishakha Patio, Aditi Bhawar,Mrs.Geeta Chillarge :2019

Support Vector Machine (SVM) Maximises the prediction accuracy, sensitivity, specificity, avoids over-fitting problem, classifies and diagnosis effectively. Needs more tuning parameters and deep study is needed. Can't perform using statistical analysis, group attention selection process needs more attention. [5]

K.Susmitha,B.Senthil Kumar :2018

K-means Clustering algorithm with MAFIA method. Easy to implement with large number of variable, K-means may be computationally faster than the Hierarchical clustering. Difficult to predict the number of clusters (k-value). The order of data has impact on final results.[6]

VijayKumar. V. Nagree, Vidyasagar. G, Kamble, Pravin.L.Devdakar,Vishwajeet Malusare, Mr.S.M.Jawahire (Asst. Professor).

Naive Bayes and WAL(Weighted Associated Classifier)
Very simple and easy to use, highly scalable, probabilistic predictions. Features in output class are Independent, Scarcity of Data. [7]

J.Vijayashree,N.Ch.Sriman Narayanalyengar :2016 Data mining and Hybrid Intelligent Techniques. Automatic Intelligent trusted system that predicts based on gender /age and domain at a low cost. Blood circulation of body is Inefficient, it affects both Heart and brain. Cardiovascular Heart Disease (CAD) is the major reason for death. [8]

Indu Yekkala, Vardhaman :2018 Principal Component Analysis (PCA) Low-noise sensitivity, decreased requirements for capacity and memory, Increased efficiency. The original variables are correlated and Homogeneous.[9]

EXISTING SYSTEM

The obtained results are compared with the results of existing models within the same domain and found to be improved. The data of heart disease patients collected from the UCI laboratory is used to discover patterns with NN, DT, Support Vector machines SVM, and Naive Bayes. The results are compared for performance and accuracy with these algorithms. The proposed hybrid method returns results of 86:8% for F-measure, competing with the other existing methods.

PROPOSED SYSTEM

We have used python and pandas operations to perform heart disease classification of the Cleveland UCI repository. It provides an

easy-to-use visual representation of the dataset, working environment and building the predictive analytics. ML process starts from a pre-processing data phase followed by feature selection based on data cleaning, classification of modeling performance evaluation, and the results with improved accuracy.

CHAPTER - 3

AIM AND SCOPE OF THE PROJECT

AIM OF THE PROJECT:

To apply machine learning techniques resulting in improving the accuracy in the prediction of cardiovascular disease

SCOPE OF THE PROJECT:

Heart disease is one of the most significant causes of mortality in the world today. Prediction of cardiovascular disease is a critical challenge in the area of clinical data analysis. Machine learning (ML) has been shown to be effective in assisting in making decisions and predictions from the large quantity of data produced by the healthcare industry. We have also seen ML techniques being used in recent developments in different areas of the Internet of Things (IoT). Various studies give only a glimpse into predicting heart disease with ML techniques. In this paper, we propose a novel method that aims at finding significant features by applying machine learning techniques resulting in improving the accuracy in the prediction of cardiovascular disease. The prediction model is introduced with different combinations of features and several known classification techniques.

CHAPTER - 4

EXPERIMENTAL METHODS AND ALGORITHM

HARDWARE AND SOFTWARE SPECIFICATION

HARDWARE REQUIREMENTS

Processor	13 and Above
HardDisk	500GB and Above
Ram	4GB andAbove

SOFTWARE REQUIREMENTS

- ✓ OperatingSystem: Windows 7 , 8, 10 (64bit)
- ✓ Software : Python 3.7
- ✓ Tools : Anaconda(JupyterNoteBookIDE)

TECHNOLOGIES USED

- Python
- Machinelearning

ALGORITHMS

K-NEARESTNEIGHBOR

Introduction

In four years of the analytics built more than 80% of classification models and just 15-20% regression models. These ratios can be more or less generalized throughout the industry. The reason of a bias towards classification models is that most analytical problem involves making a decision. 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. This analysis is more insightful and directly links to an implementation roadmap. In this article, we will talk about another widely used classification technique called K-nearest neighbors (KNN). Our focus will be primarily on how does the algorithm work and how does the input parameter effect the output/prediction.

KNN algorithm

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:

1. Ease to interpret output
2. Calculation time
3. Predictive Power

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

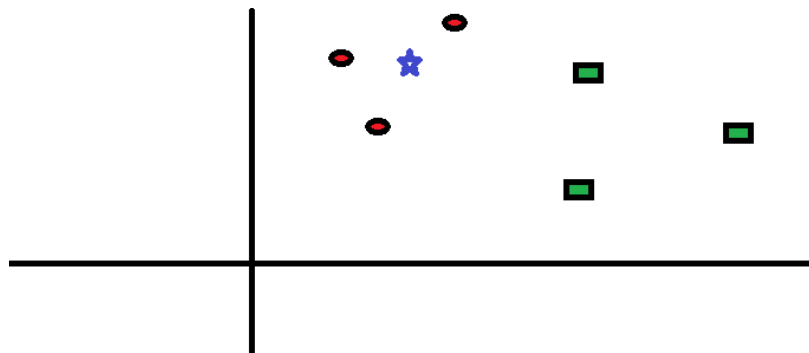
	Logistic Regression	CART	Random Forest	KNN
1. Ease to interpret output	2	3	1	3
2. Calculation time	3	2	1	3
3. Predictive Power	2	2	3	2

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

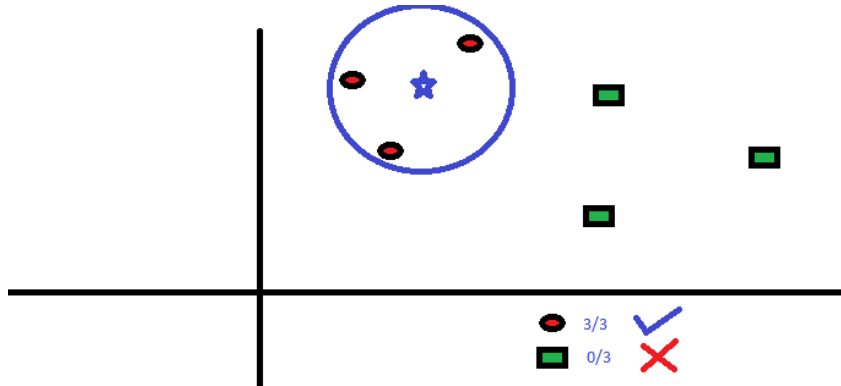
The KNN algorithm work

Let's take a simple case to understand this algorithm.

Following is a spread of red circles (RC) and green squares (GS):



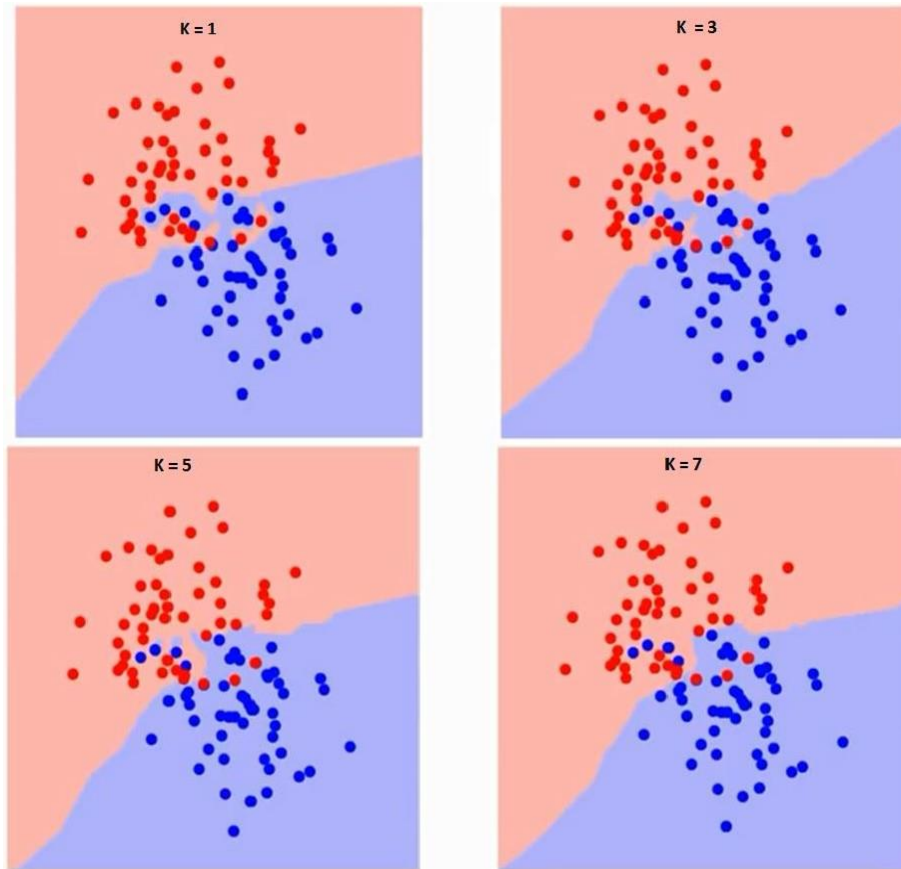
You intend to find out the class of the blue star (BS). BS can either be RC or GS and nothing else. The "K" in KNN algorithm is the nearest neighbors we wish to take vote from. Let's say $K = 3$. Hence, we will now make a circle with BS as center just as big as to enclose only three data points on the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



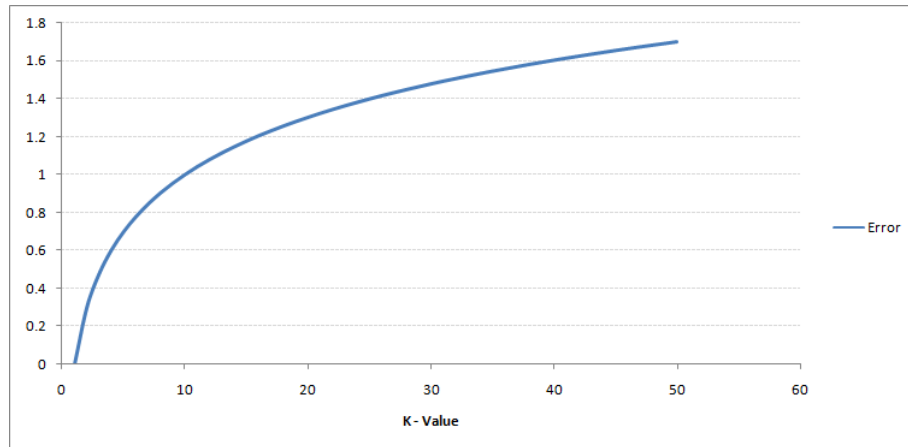
The three closest points to BS is all RC. Hence, with good confidence level we can say that the BS should belong to the class RC. Here, the choice became very obvious as all three votes from the closest neighbor went to RC. The choice of the parameter K is very crucial in this algorithm.

How do we choose the factor K?

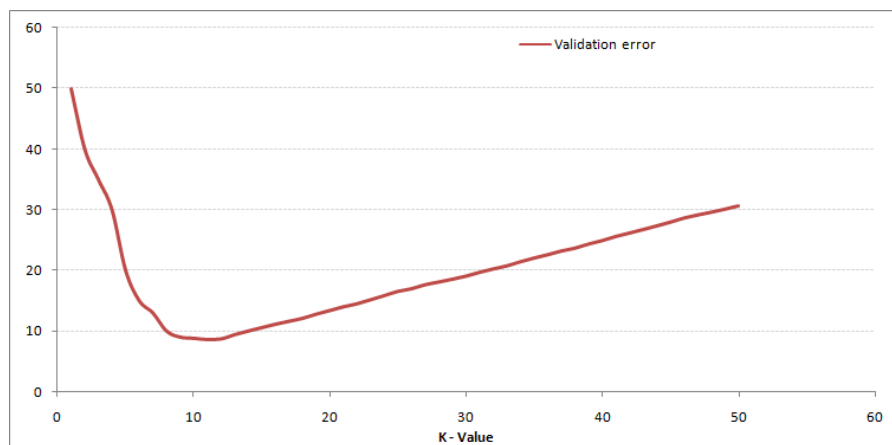
First let us try to understand what exactly does K influence in the algorithm. If we see the last example, given that all the 6 training observation remain constant, with a given K value we can make boundaries of each class. These boundaries will segregate RC from GS. The same way, let's try to see the effect of value "K" on the class boundaries. 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 on different K-value. Following is the curve for the training error rate with 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 over fitting the boundaries. Hence, error rate initially decreases and reaches a minimal. After the minima point, it then increases 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.

Breaking it down – Pseudo Code of KNN

We can implement a KNN model by following the below steps:

1. Load the data
2. Initialize the value of k
3. For getting the predicted class, iterate from 1 to total number of training data points
 1. Calculate the distance between test data and each row of training data. Here we will use Euclidean distance as our distance metric since it's the most popular method. The other metrics that can be used are Chebyshev, cosine, etc.
 2. Sort the calculated distances in ascending order based on distance values
 3. Get top k rows from the sorted array
 4. Get the most frequent class of these rows
 5. Return the predicted class

Conclusion

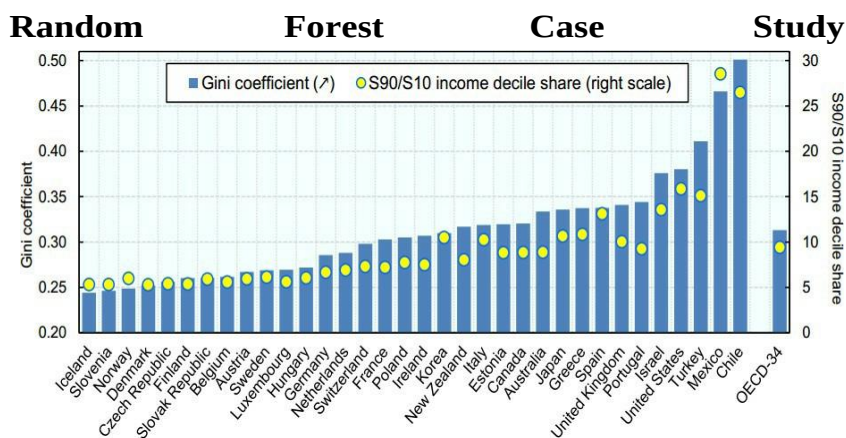
KNN algorithm is one of the simplest classification algorithms. Even with such simplicity, it can give highly competitive results. KNN algorithm can also be used for regression problems. The only difference from the discussed methodology will be using averages of nearest neighbors rather than voting from nearest neighbors.

RANDOMFOREST

Introduction

With increase in computational power, we can now choose algorithms which perform very intensive calculations. One such algorithm is “ Random Forest”, which we will discuss in this article. While the algorithm is very popular in various competitions (e.g. like the ones running on Kaggle), the end output of the model is like a black box and hence should be used judiciously.

Before going any further, here is an example on the importance of choosing the best algorithm.



Following is a distribution of Annual income Gini Coefficients across different countries:

Mexico has the second highest Gini coefficient and hence has a very high segregation in annual income of rich and poor. Our task is to come up with an accurate predictive algorithm to estimate annual income bracket of each individual in Mexico. The brackets of income are as follows:

1. below \$40,000
2. \$40,000 – 150,000
3. More than \$150,000

Following are the information available for each individual:

1. Age, 2. Gender, 3. Highest educational qualification, 4. Working in Industry, 5. Residence in Metro/Non-metro

We need to come up with an algorithm to give an accurate prediction for an individual who has following traits:

1. Age: 35 years, 2. Gender: Male, 3. Highest Educational Qualification: Diploma holder, 4. Industry: Manufacturing, 5. Residence: Metro

We will only talk about random forest to make this prediction in this article.

The algorithm of Random Forest

Random forest is like bootstrapping algorithm with Decision tree (CART) model. Say, we have 1000 observation in the complete population with 10 variables. Random forest tries to build multiple CART model with different sample and different initial variables. For instance, it will take a random sample of 100 observation and 5 randomly chosen initial variables to build a CART model. It will repeat the process (say) 10 times and then make a final prediction on each observation. Final prediction is a function of each prediction. This final prediction can simply be the mean of each prediction.

Back to Case study

Disclaimer: The numbers in this article are illustrative

Mexico has a population of 118 MM. Say, the algorithm Random forest picks up 10k observation with only one variable (for simplicity) to build each CART model. In total, we are looking at 5 CART models being built with different variables. In a real life problem, you will

have more number of population sample and different combinations of input variables.

Salary

bands: Band 1: Below

\$40,000

Band 2: \$40,000 – 150,000

Band 3: More than \$150,000

Following are the outputs of the 5 different CART model.

CART 1: Variable Age

	Salary Band	1	2	3
Age	Below 18	90%	10%	0%
	19-27	85%	14%	1%
	28-40	70%	23%	7%
	40-55	60%	35%	5%
	More than 55	70%	25%	5%

CART 2: Variable Gender

	Salary Band	1	2	3
Gender	Male	70%	27%	3%
	Female	75%	24%	1%

CART 3: Variable Education

	Salary Band	1	2	3
Residence	Metro	70%	20%	10%
	Non-Metro	65%	20%	15%

CART 4: Variable Residence

	Salary Band	1	2	3
Education	<=High School	85%	10%	5%
	Diploma	80%	14%	6%
	Bachelors	77%	23%	0%
	Master	62%	35%	3%

CART 5: Variable Industry

	Salary Band	1	2	3
Industry	Finance	65%	30%	5%
	Manufacturing	60%	35%	5%
	Others	75%	20%	5%

Using these 5 CART models, we need to come up with single set of probability to belong to each of the salary classes. For simplicity, we will just take a mean of probabilities in this case study. Other than simple mean, we also consider vote method to come up with the final prediction. To come up with the final prediction let 's locate the following profile in each CART model:

1. Age: 35 years, 2. Gender: Male, 3. Highest Educational Qualification: Diploma holder, 4. Industry: Manufacturing, 5. Residence: Metro

For each of these CART model, following is the distribution across salary bands:

CART	Band	1	2	3
Age	28-40	70%	23%	7%
Gender	Male	70%	27%	3%
Education	Diploma	80%	14%	6%
Industry	Manufacturing	60%	35%	5%
Residence	Metro	70%	20%	10%
Final probability		70%	24%	6%

The final probability is simply the average of the probability in the same salary bands in different CART models. As you can see from this analysis, that there is 70% chance of this individual falling in class 1 (less than \$40,000) and around 24% chance of the individual

falling in class2.

Conclusion

Random forest gives much more accurate predictions when compared to simple CART/CHAID or regression models in many scenarios. These cases generally have high number of predictive variables and huge sample size. This is because it captures the variance of several input variables at the same time and enables high number of observations to participate in the prediction.

REGIONAL-CNN

In this chapter we will talk about the Region based methods for object detection. In particular, R-CNN (Regional CNN), the original application of CNNs to this problem, along with its descendants Fast R-CNN, and Faster R-CNN. In classification, there 's generally an image with a single object as the focus and the task is to say what that image is. But when we look at the world around us, we carry out far more complex tasks. We see complicated sights with multiple overlapping objects, and different backgrounds and we not only classify these different objects but also identify their boundaries, differences, and relations to one another. The goal of a Region based CNN is to take an image input to be detected and localize the object in the image. All three above mentioned architectures were used for object detection and localization for a driving scenario to detect cars, pedestrians and traffic signs etc. and performed a time study to conclude which of the three architectures can be used as a detector for a real-world driving scenario for detection of objects present on

road. We will also discuss their architecture and describe how they perform the task of object detection and localization given the input image.

R-CNN

R-CNN was first introduced by Ross Girshick, Jeff Donahue, Trevor Darrell and Jitendra Malik in 2014. Region based Convolutional Network(R-CNN) combines two ideas: (1) one can apply high-capacity Convolutional Networks (CNN ' S) to bottom-up region proposals in order to localize and segment objects and (2) When labelling data is scarce, supervised pre-training for an auxiliary task, followed by domain-specific-fine-tuning, boost performance significantly. They combined region proposals with CNNs and called the resulting model an R-CNN or Region-based Convolutional Network.

R-CNN Architecture.

In R-CNN we input an image and then using the basic feature detection techniques such as edge detection etc. we get Region Proposals Also referred as Regions of interest. This process is also known as selective search.

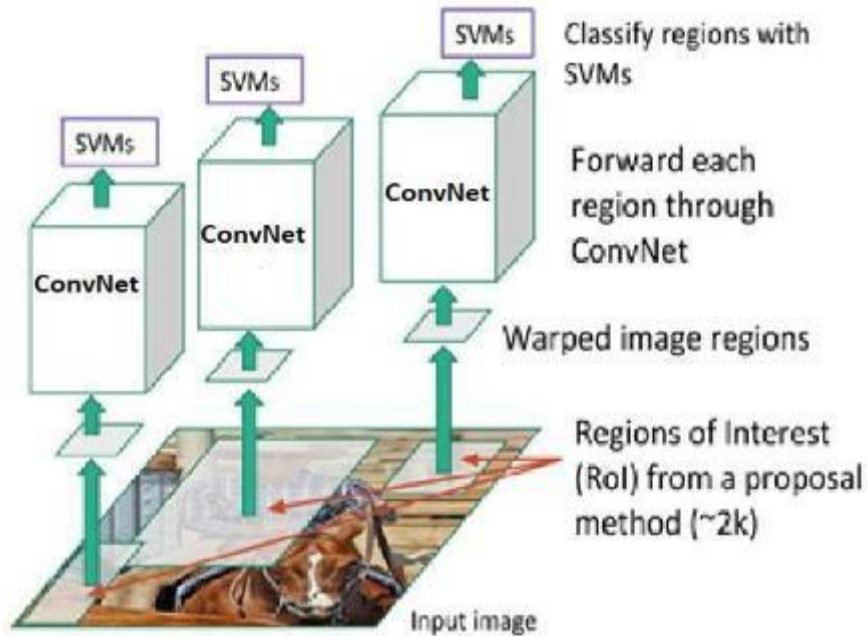
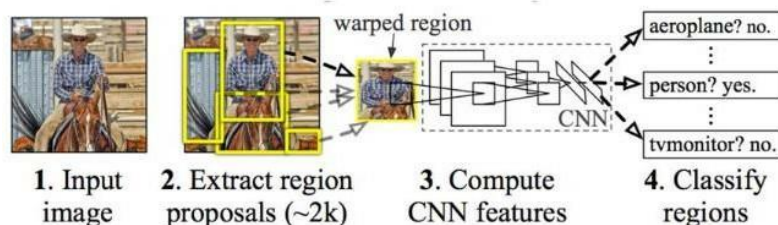


Figure 4.3.3 (i) R-CNN Architecture

In the next step after these region proposals are extracted these wrapped image regions go through a trained CNN in our implementation AlexNet and then on the final layer the classification is done Using a Support Vector Machine(SVM) classifier which classifies whether this is an object and if so what. Pictorial representation of above process is shown in Figure.



Procedure for Object Detection and Localization in R-CNN

R-CNN works really well but is really quite slow for implementation in real time object detection.

1. It requires a forward pass of the CNN (AlexNet) for every single region proposal for every single image that's around 2000 forward passes per image.
2. It must train three different models separately -the CNN to generate image features, the classifier that predicts the class and the regression model to tighten the bounding boxes. This makes the pipeline extremely hard to train

CONVOLUTIONAL NEURAL NETWORKS(CNN)

Introduction

Convolutional neural networks (CNN) sounds like a weird combination of biology and math with a little CS sprinkled in, but these networks have been some of the most influential innovations in the field of computer vision. 2012 was the first year that neural nets grew to prominence as Alex Krizhevsky used them to win that year's Image Net competition (basically, the annual Olympics of computer vision), dropping the classification error record from 26% to 15%, an astounding improvement at the time. Ever since then, a host of companies have been using deep learning at the core of their services. Facebook uses neural nets for their automatic tagging algorithms, Google for their photo search, Amazon for their product

recommendations, Pinterest for their home feed personalization, and Instagram for their search infrastructure.

However, the classic, and arguably most popular, use case of these networks is for image processing. Within image processing, let's take a look at how to use these CNNs for image classification.

The Problem Space

Image classification is the task of taking an input image and outputting a class (a cat, dog, etc) or a probability of classes that best describes the image. For humans, this task of recognition is one of the first skills we learn from the moment we are born and is one that comes naturally and effortlessly as adults. Without even thinking twice, we're able to quickly and seamlessly identify the environment we are in as well as the objects that surround us. When we see an image or just when we look at the world around us, most of the time we are able to immediately characterize the scene and give each object a label, all without even consciously noticing. These skills of being able to quickly recognize patterns, generalize from prior knowledge, and adapt to different image environments are ones that



What We See

```
08 02 22 97 38 15 00 40 00 75 04 05 07 76 52 12 50 77 91 08
49 49 99 40 17 01 18 57 60 87 17 40 98 43 49 48 04 56 42 00
81 49 31 73 55 79 14 29 99 71 40 47 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91
22 31 14 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 34 84 20 35 17 12 50
32 98 81 29 46 23 67 10 24 38 40 47 59 94 70 46 18 38 64 70
67 24 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21
24 55 58 05 64 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72
21 34 23 09 75 00 76 44 20 45 35 14 00 41 33 97 34 31 33 95
78 17 53 28 22 75 31 47 15 94 03 80 04 62 14 14 09 53 56 92
14 39 05 42 94 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 40 21 58 51 54 17 58
19 80 81 48 05 94 47 49 28 73 92 13 86 52 17 77 04 89 55 40
04 52 08 83 97 35 99 14 07 97 57 32 14 24 24 79 33 27 98 44
88 34 68 87 57 62 20 72 03 46 33 47 46 55 12 32 63 93 53 49
04 42 14 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36
20 49 34 41 72 30 23 88 34 42 99 49 82 47 59 85 74 04 34 14
20 73 35 29 78 31 90 01 74 31 49 71 48 84 81 14 23 57 05 94
01 70 54 71 83 51 54 49 14 92 33 48 41 43 52 01 89 19 47 48
```

What Computers See

we do not share with our fellow machines.

Inputs and Outputs

When a computer sees an image (takes an image as input), it will see an array of pixel values. Depending on the resolution and size of the image, it will see a 32 x 32 x 3 array of numbers (The 3 refers to RGB values). Just to drive home the point, let's say we have a color image in JPG format and its size is 480 x 480. The representative array will be 480 x 480 x 3. Each of these numbers is given a value from 0 to 255 which describe the pixel intensity at that point. These numbers, while meaningless to us when we perform image classification,

are the only inputs available to the computer. The idea is that you give the computer this array of numbers and it will output numbers that describe the probability of the image being a certain class (.80 for cat, .15 for dog, .05 for bird, etc).

What We Want the Computer to do

Now that we know the problem as well as the inputs and outputs, let's think about how to approach this. What we want the computer to do is to be able to differentiate between all the images it's given and figure out the unique features that make a dog a dog or that make a cat a cat. This is the process that goes on in our minds subconsciously as well. When we look at a picture of a dog, we can classify it as such if the picture has identifiable features such as paws or 4 legs. In a similar way, the computer is able to perform image classification by looking for low level features such as edges and

curves, and then building up to more abstract concepts through a series of convolutional layers. This is a general overview of what a CNN does. Let's get into the specifics.

Biological Connection

But first, a little background. When you first heard of the term convolutional neural networks, you may have thought of something related to neuroscience or biology, and you would be right. Sort of. CNNs do take a biological inspiration from the visual cortex. The visual cortex has small regions of cells that are sensitive to specific regions of the visual field. This idea was expanded upon by a fascinating experiment by Hubel and Wiesel in 1962 (Video) where they showed that some individual neuronal cells in the brain responded (or fired) only in the presence of edges of a certain orientation. For example, some neurons fired when exposed to vertical edges and some when shown horizontal or diagonal edges. Hubel and Wiesel found out that all of these neurons were organized in a columnar architecture and that together, they were able to produce visual perception. This idea of specialized components inside of a system having specific tasks (the neuronal cells in the visual cortex looking for specific characteristics) is one that machines use as well, and is the basis behind CNNs.

Structure

Back to the specifics. A more detailed overview of what CNNs do would be that you take the image, pass it through a series of convolutional, nonlinear, pooling (down sampling), and fully

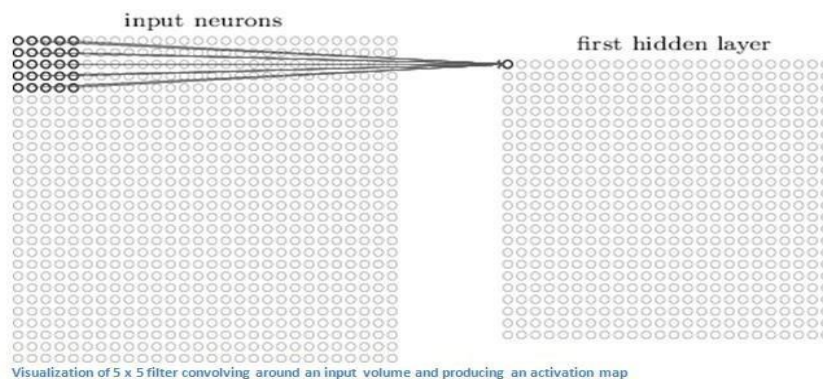
connected layers, and get an output. As we said earlier, the output can be a single class or a probability of classes that best describes the image. Now, the hard part understands what each of these layers do. So let's get into the most important one.

First Layer – Math Part

The first layer in a CNN is always a Convolutional Layer. First thing to make sure you remember is what the input to this conv (I ' ll be using that abbreviation a lot) layer is. Like we mentioned before, the input is a $32 \times 32 \times 3$ array of pixel values. Now, the best way to explain a conv layer is to imagine a flashlight that is shining over the top left of the image. Let ' s say that the light this flashlight shines covers a 5×5 area. And now, let ' s imagine this flashlight sliding across all the areas of the input image. In machine learning terms, this flashlight is called a filter (or sometimes referred to as a neuron or a kernel) and the region that it is shining over is called the receptive field. Now this filter is also an array of numbers (the numbers are called weights or parameters). A very important note is that the depth of this filter has to be the same as the depth of the input (this makes sure that the math works out), so the dimension of this filter is $5 \times 5 \times 3$. Now, let's take the first position the filter is in for example. It would be the top left corner. As the filter is sliding, or convolving, around the input image, it is multiplying the values in the filter with the original pixel values of the image (aka computing element wise multiplications). These multiplications are all summed up (mathematically speaking, this would be 75 multiplications in total). So now you have a single number. Remember, this number is just representative of when the filter is at

the top left of the image. Now, we repeat this process for every location on the input volume. (Next step would be moving the filter to the right by 1 unit, then right again by 1, and so on). Every unique location on the input volume produces a number. After sliding the filter over all the locations, you will find out that what you're left with is a $28 \times 28 \times 1$ array of numbers, which we call an activation map or feature map. The reason you get a 28×28 array is that there are 784 different locations that a 5×5 filter can fit on a 32×32 input image. These 784 numbers are mapped to a 28×28 array.

Let's say now we use two $5 \times 5 \times 3$ filters instead of one. Then our output volume would be $28 \times 28 \times 2$. By using more filters, we are able to preserve the spatial dimensions better. Mathematically, this is what's going on in a convolutional layer.



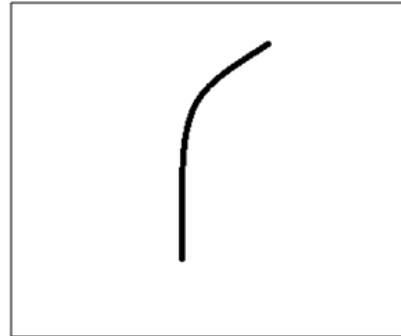
First Layer – High Level Perspective

However, let's talk about what this convolution is actually doing from a high level. Each of these filters can be thought of as **feature identifiers**. When I say features, I'm talking about things like straight edges, simple colors, and curves. Think about the simplest characteristics that all images have in common with each other. Let's say our first filter is $7 \times 7 \times 3$ and is going to be a curve detector. In

this section, let's ignore the fact that the filter is 3units deep and only consider the top depth slice of the filter and the image, for simplicity.)As a curve detector, the filter will have a pixel structure in which there will be higher numerical values along the area that is a shape of a curve (Remember, these filters that we're talking about as just numbers!).

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

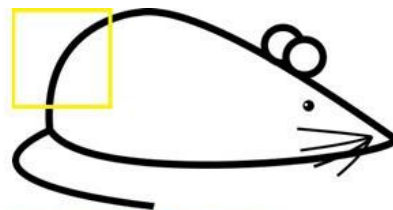


Visualization of a curve detector filter

Now, let ' s go back to visualizing this mathematically. When we have this filter at the top left corner of the input volume, it is computing multiplications between the filter and pixel values at that region. Now let ' s take an example of an image that we want to classify, and let's put our filter at the top left corner.

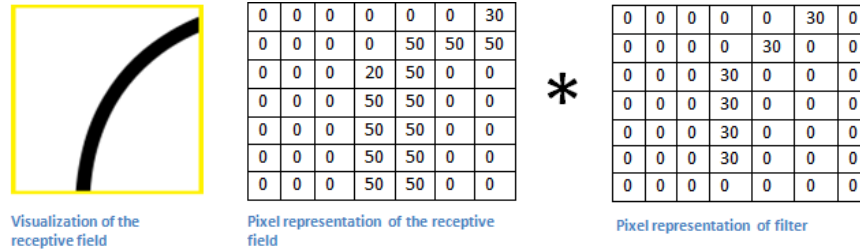


Original image



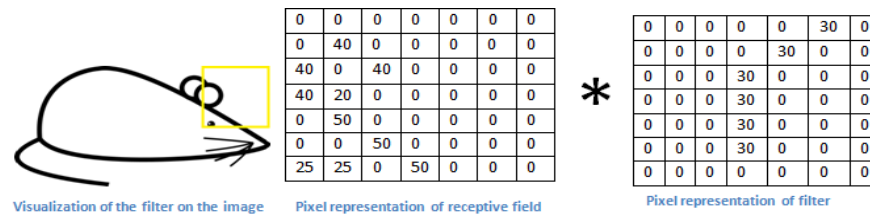
Visualization of the filter on the image

Remember, what we have to do is multiply the values in the filter with the original pixel values of the image.



Multiplication and Summation = $(50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600$ (A large number!)

Basically, in the input image, if there is a shape that generally resembles the curve that this filter is representing, then all of the multiplications summed together will result in a large value! Now let's see what happens when we move our filter.



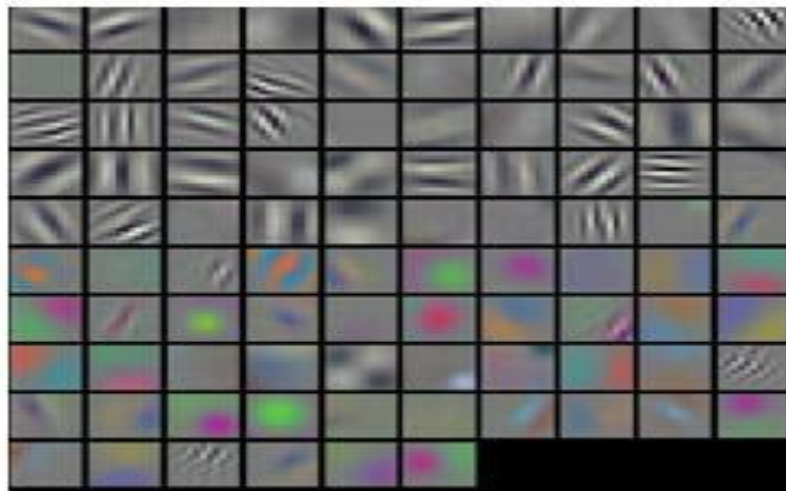
Multiplication and Summation = 0

The value is much lower! This is because there wasn't anything in the image section that responded to the curve detector filter. Remember, the output of this conv layer is an activation map. So, in the simple case of a one filter convolution (and if that filter is a curve detector), the activation map will show the areas in which there are mostly likely to be curves in the picture. In this example, the top left value of our 26 x 26 x 1 activation map (26 because of the 7x7 filter instead of 5x5) will be 6600. This high value means that it is likely that there is some sort of curve in the input volume that caused the filter to activate. The top right value in our activation map will be 0 because there wasn't anything in the input volume that caused the filter to activate or more simply said, there wasn't a curve in that

region of the original image). Remember, this is just for one filter. This is just a filter that is going to detect lines that curve outward and to the right. We can have other filters for lines that curve to the left or for straight edges. The more filters, the greater the depth of the activation map, and the more information we have about the input volume.

Disclaimer:

The filter I described in this section was simplistic for the main purpose of describing the math that goes on during a convolution. In the picture below, you'll see some examples of actual visualizations of the filters of the first conv layer of a trained network. Nonetheless, the main argument remains the same. The filters on the first layer convolve around the input image and "activate" (or compute high values) when the specific feature it is looking for is in the input



Visualizations of filters

volume.

Going Deeper Through the Network

Now in traditional convolutional neural network architecture, there are other layers that are interspersed between these conv layers. I'd strongly encourage those interested to read up on them and understand their function and effects, but in a general sense, they provide nonlinearities and preservation of dimension that help to improve the robustness of the network and control over fitting. A classic CNN architecture would look like this.

Input -> Conv -> ReLU -> Conv -> ReLU -> Pool -> ReLU -> Conv -> ReLU -> Pool -> Fully Connected
--

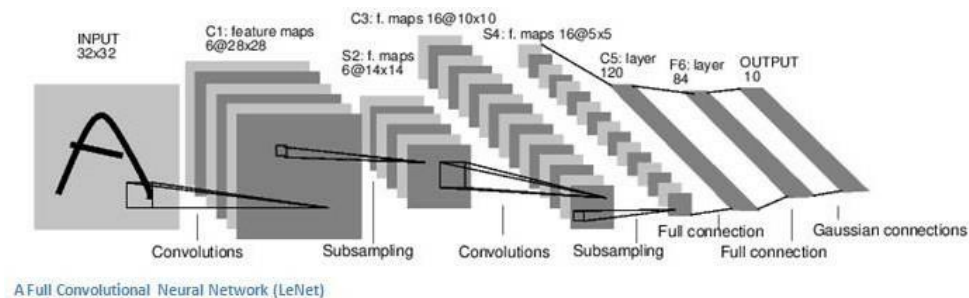
The last layer, however, is an important one and one that we will go into later on. Let's just take a step back and review what we've learned so far. We talked about what the filters in the first conv layer are designed to detect. They detect low level features such as edges and curves. As one would imagine, in order predicting whether an image is a type of object, we need the network to be able to recognize higher level features such as hands or paws or ears. So let's think about what the output of the network is after the first conv layer. It would be a $28 \times 28 \times 3$ volume (assuming we use three $5 \times 5 \times 3$ filters). When we go through another conv layer, the output of the first conv layer becomes the input of the 2nd conv layer. Now, this is a little bit harder to visualize. When we were talking about the first layer, the input was just the original image. However, when we're talking about the 2nd conv layer, the input is the activation map(s) that result from the first layer. So each layer of the input is basically describing the locations in the original image for where certain low

level features appear. Now when you apply a set of filters on top of that (pass it through the 2nd conv layer), the output will be activations that represent higher level features. Types of these features could be semicircles (combination of a curve and straight edge) or squares (combination of several straight edges). As you go through the network and go through more conv layers, you get activation maps that represent more and more complex features. By the end of the network, you may have some filters that activate when there is handwriting in the image, filters that activate when they see pink objects, etc. If you want more information about visualizing filters in ConvNets, Matt Zeiler and Rob Fergus had an excellent research paper discussing the topic. Jason Yosinski also has a video on YouTube that provides a great visual representation. Another interesting thing to note is that as you go deeper into the network, the filters begin to have a larger and larger receptive field, which means that they are able to consider information from a larger area of the original input volume (another way of putting it is that they are more responsive to a larger region of pixelspace).

Fully Connected Layer

Now that we can detect these high level features, the icing on the cake is attaching a **fully connected layer** to the end of the network. This layer basically takes an input volume (whatever the output is of the conv or ReLU or pool layer preceding it) and outputs an N dimensional vector where N is the number of classes that the program has to choose from. For example, if you wanted a digit classification program, N would be 10 since there are 10 digits. Each number in this N dimensional vector represents the probability of a

certain class. For example, if the resulting vector for a digit classification program is $[0.1 \ 0.1 \ 0.75 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0.05]$, then this represents a 10% probability that the image is a 1, a 10% probability that the image is a 2, a 75% probability that the image is a 3, and a 5% probability that the image is a 9 (Side note: There are other ways that you can represent the output, but I am just showing the softmax approach). The way this fully connected layer works is that it looks at the output of the previous layer (which as we remember should represent the activation maps of high level features) and determines which features most correlate to a particular class. For example, if the program is predicting that some image is a dog, it will have high values in the activation maps that represent high level features like a paw or 4 legs, etc. Similarly, if the program is predicting that some image is a bird, it will have high values in the activation maps that represent high level features like wings or a beak, etc. Basically, a FC layer looks at what high level features most strongly correlate to a particular class and has particular weights so that when you compute the products between the weights and the previous layer, you get the correct probabilities for the different classes.



Training (AKA: What Makes this Stuff Work)

Now, this is the one aspect of neural networks that I purposely haven't mentioned yet and it is probably the most

important part. There may be a lot of questions you had while reading. How do the filters in the first conv layer know to look for edges and curves? How does the fully connected layer know what activation maps to look at? How do the filters in each layer know what values to have? The way the computer is able to adjust its filter values (or weights) is through a training process called **back propagation**.

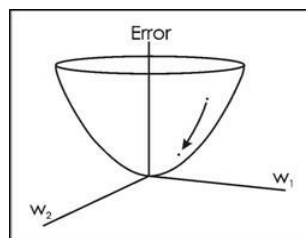
Before we get into back propagation, we must first take a step back and talk about what a neural network needs in order to work. At the moment we all were born, our minds were fresh. We didn't know what a cat or dog or bird was. In a similar sort of way, before the CNN starts, the weights or filter values are randomized. The filters don't know to look for edges and curves. The filters in the higher layers don't know to look for paws and beaks. As we grew older however, our parents and teachers showed us different pictures and images and gave us a corresponding label. This idea of being given an image and a label is the training process that CNNs go through. Before getting too into it, let's just say that we have a training set that has thousands of images of dogs, cats, and birds and each of the images has a label of what animal that picture is. Back to backprop.

So back propagation can be separated into 4 distinct sections, the forward pass, the loss function, the backward pass, and the weight update. During the **forward pass**, you take a training image which as we remember is a $32 \times 32 \times 3$ array of numbers and pass it through the whole network. On our first training example, since all of the weights or filter values were randomly initialized, the output will probably be something like $[.1 \ .1 \ .1 \ .1 \ .1 \ .1 \ .1 \ .1 \ .1 \ .1]$, basically an output that doesn't give preference to any number in particular. The

network, with its current weights, isn't able to look for those low level features or thus isn't able to make any reasonable conclusion about what the classification might be. This goes to the **loss function** part of back propagation. Remember that what we are using right now is training data. This data has both an image and a label. Let's say for example that the first training image inputted was a 3. The label for the image would be [0 0 0 1 0 0 0 0 0]. A loss function can be defined in many different ways but a common one is MSE (mean squared error), which is $\frac{1}{2}$ times (actual - predicted) squared.

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

Let ' s say the variable L is equal to that value. As you can imagine, the loss will be extremely high for the first couple of training images. Now, let's just think about this intuitively. We want to get to a point where the predicted label (output of the ConvNet) is the same as the training label (This means that our network got its prediction right). In order to get there, we want to minimize the amount of loss we have. Visualizing this as just an optimization problem in calculus, we want to find out which inputs (weights in our case) most directly contributed to the loss (or error) of the network.



One way of visualizing this idea of minimizing the loss is to consider a 3-D graph where the weights of the neural net (there are obviously more than 2 weights, but let's go for simplicity) are the independent variables and the dependent variable is the loss. The task of minimizing the loss involves trying to adjust the weights so that the loss decreases. In visual terms, we want to get to the lowest point in our bowl shaped object. To do this, we have to take a derivative of the loss (visual terms: calculate the slope in every direction) with respect to the weights.

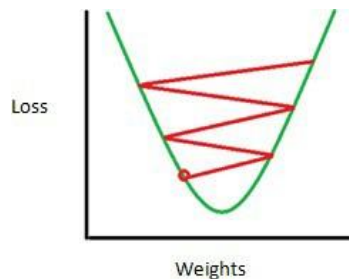
This is the mathematical equivalent of a $\frac{dL}{dW}$ where W are the weights at a particular layer. Now, what we want to do is perform

a **backward pass** through the network, which is determining which weights contributed most to the loss and finding ways to adjust them so that the loss decreases. Once we compute this derivative, we then go to the last step which is the **weight update**. This is where we take all the weights of the filters and update them so that they change in the opposite direction of the gradient.

$$w = w_i - \eta \frac{dL}{dW}$$

w = Weight
 w_i = Initial Weight
 η = Learning Rate

The **learning rate** is a parameter that is chosen by the programmer. A high learning rate means that bigger steps are taken in the weight updates and thus, it may take less time for the model to converge on an optimal set of weights. However, a learning rate that is too high could result in jumps that are too large and not precise enough to reach the optimal point.



Consequence of a high learning rate where the jumps are too large and we are not able to minimize the loss.

The process of forward pass, loss function, backward pass, and parameter update is one training iteration. The program will repeat this process for a fixed number of iterations for each set of training images (commonly called a batch). Once you finish the parameter update on the last training example, hopefully the network

should be trained well enough so that the weights of the layers are tuned correctly.

Testing

Finally, to see whether or not our CNN works, we have a different set of images and labels (can't double dip between training and test!) and pass the images through the CNN. We compare the outputs to the ground truth and see if our network works!

How Companies Use CNNs

Data, data, data. The companies that have lots of this magic 4 letter word are the ones that have an inherent advantage over the rest of the competition. The more training data that you can give to a network, the more training iterations you can make, the more weight updates you can make, and the better tuned to the network is when it goes to production. Facebook (and Instagram) can use all the photos of the billion users it currently has, Pinterest can use information of the 50 billion pins that are on its site, Google can use search data, and Amazon can use data from the millions of products that are bought every day. And now you know the magic behind how they use it.

Disclaimer

While this post should be a good start to understanding CNNs, it is by no means a comprehensive overview. Things not discussed in this post include the nonlinear and pooling layers as well as hyper parameters of the network such as filter sizes, stride, and padding.

Topics like network architecture, batch normalization, vanishing gradients, dropout, initialization techniques, non-convex optimization, biases, and choices of loss functions, data augmentation, regularization methods, computational considerations, modifications of back propagation, and more were also not discussed (yet).

NAIVE BAYESALGORITHM

Introduction

What will you do? You have hundreds of thousands of data points and quite a few variables in your training data set. In such situation, if I were at your place, I would have used ' **Naive Bayes** ', which can be extremely fast relative to other classification algorithms. It works on Bayes theorem of probability to predict the class of unknown data set.

So that next time when you come across large data sets, you can bring this algorithm to action.

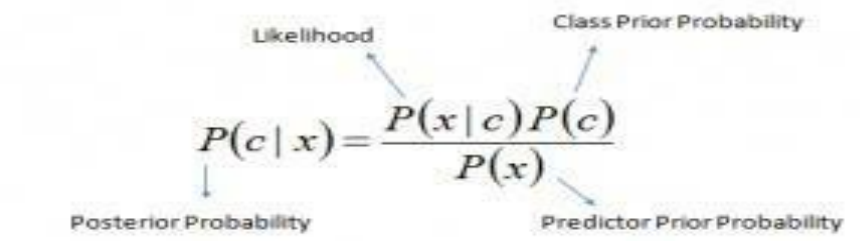
Naive Bayes algorithm

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this

fruit is an apple and that is why it is known as 'Naive'.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability $P(c|x)$ from $P(c)$, $P(x)$ and $P(x|c)$. Look at the equation below:


$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$
$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c)$$

Above,

- $P(c|x)$ is the posterior probability of class (c, target) given predictor (x, attributes).
- $P(c)$ is the prior probability of class.
- $P(x|c)$ is the likelihood which is the probability of predictor given class.
- $P(x)$ is the prior probability of predictor.

Naive Bayes algorithm works

Let's understand it using an example. Below I have a training data set of weather and corresponding target variable 'Play' (suggesting possibilities of playing). Now, we need to classify whether players will play or not based on weather condition. Let's

follow the below steps to perform it.

Step 1: Convert the data set into a frequency table

Step 2: Create Likelihood table by finding the probabilities like
Overcast probability = 0.29 and probability of playing is 0.64.

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
All	5	9
	=5/14	=9/14
	0.36	0.64

Step 3: Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

Problem:

Players will play if weather is sunny. Is this statement is correct?

We can solve it using above discussed method of posterior probability.

$$P(\text{Yes} | \text{Sunny}) = P(\text{Sunny} | \text{Yes}) * P(\text{Yes}) / P(\text{Sunny})$$

Here we have $P(\text{Sunny} | \text{Yes}) = 3/9 = 0.33$, $P(\text{Sunny}) = 5/14 = 0.36$,
 $P(\text{Yes}) = 9/14 = 0.64$

Now, $P(\text{Yes} | \text{Sunny}) = 0.33 * 0.64 / 0.36 = 0.60$, which has higher probability.

Naive Bayes uses a similar method to predict the probability of different class based on various attributes. This algorithm is mostly used in text classification and with problems having multiple classes.

The Pros and Cons of Naive Bayes

Pros:

- It is easy and fast to predict class of test data set. It also perform well in multi class prediction
- When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
- It performs well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

Cons:

- If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as “Zero Frequency”. To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.
- On the other side naive Bayes is also known as a bad estimator, so the probability outputs from predict probability are not to be taken too seriously.
- Another limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

Four Applications of Naive Bayes Algorithms

- **Real time Prediction:** Naive Bayes is an eager learning classifier and it is sure fast. Thus, it could be used for making predictions in realtime.
- **Multi class Prediction:** This algorithm is also well known for multi class prediction feature. Here we can predict the probability of multiple classes of target variable.
- **Text classification/ Spam Filtering/ Sentiment Analysis:** Naive Bayes classifiers mostly used in text classification (due to better result in multi class problems and independence rule) have higher success rate as compared to other algorithms. As a result, it is widely used in Spam filtering (to identify spam e-mail) and Sentiment Analysis (in social media analysis, to identify positive and negative customer sentiments)
- **Recommendation System:** Naive Bayes Classifier and Collaborative Filtering together builds a Recommendation System that uses machine learning and data mining techniques to filter unseen information and predict whether a user would like a given resource or not

Tips to improve the power of Naive Bayes Model

Here are some tips for improving power of Naive Bayes Model:

- If continuous features do not have normal distribution, we should use transformation or different methods to convert it in normal distribution.

- If test data set has zero frequency issue, apply smoothing techniques “Laplace Correction” to predict the class of test dataset.
- Remove correlated features, as the highly correlated features are voted twice in the model and it can lead to over inflating importance.
- Naive Bayes classifiers has limited options for parameter tuning like `alpha=1` for smoothing, `fit_prior=[True|False]` to learn class prior probabilities or not and some other options (look at detail here). I would recommend to focus on your pre-processing of data and the featureselection.
- You might think to apply some classifier combination technique like ensembling, bagging and boosting but these methods would not help. Actually, “ensembling, boosting, bagging” won’t help since their purpose is to reduce variance. Naive Bayes has no variance to minimize.

SUPPORT VECTOR MACHINE(SVM)

Introduction:

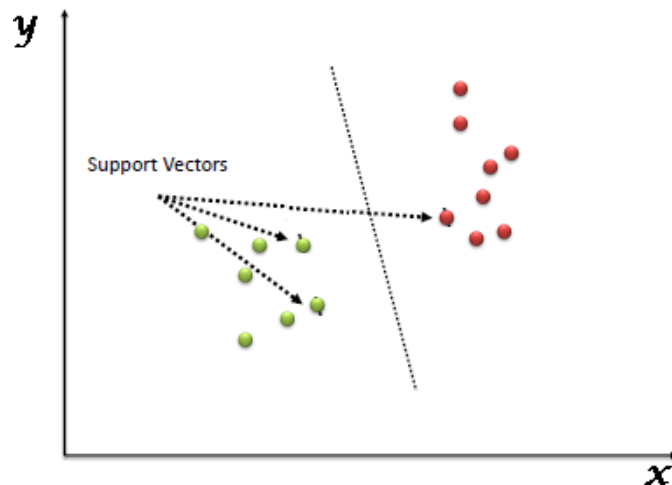
Mastering machine learning algorithms isn ’ t a myth at all. Most of the beginners start by learning regression. It is simple to learn and use, but does that solve our purpose? Of course not! Because, you can do so much more than justRegression!

Think of machine learning algorithms as an armory packed with axes, sword, blades, bow, dagger etc. You have various tools, but you ought to learn to use them at the right time. As an analogy, think of ‘Regression’ as a sword capable of slicing and dicing data

efficiently, but incapable of dealing with highly complex data. On the contrary, 'Support Vector Machines' is like a sharpknife – it works on smaller datasets, but on them, it can be much stronger and powerful in building models.

Support Vector Machine

“ Support Vector Machine ” (SVM) is a supervised machine learning algorithm which can be used for either classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well (look at the below snapshot).



Support Vectors are simply the co-ordinates of individual observation. Support Vector Machine is a frontier which best segregates the two classes (hyper-plane/line).

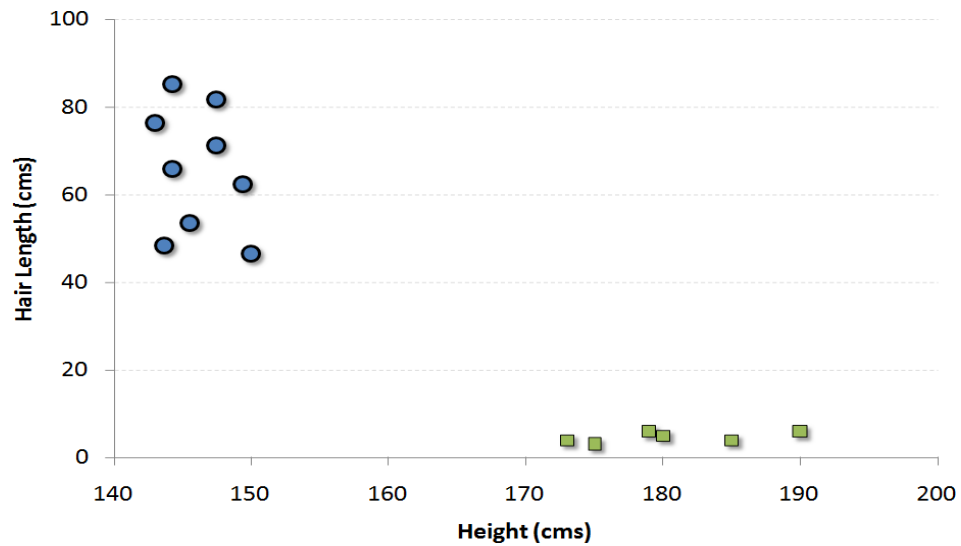
You can look at definition of support vectors and a few examples of its working here.

Support Vector Machine – Simplified

Let ' s talk about how support vector machine works. **This is suitable for readers who do not know much about this algorithm and have a curiosity to learn a new technique.** In following explore the technique in detail and analyze cases where such techniques are stronger than other techniques.

Classification analysis

Let's consider an example to understand these concepts. We have a population composed of 50%-50% Males and Females. Using a sample of this population, you want to create some set of rules which will guide us the gender class for rest of the population. Using this algorithm, we intend to build a robot which can identify whether a person is a Male or a Female. This is a sample problem of classification analysis. Using some set of rules, we will try to classify the population into two possible segments. For simplicity, let ' s assume that the two differentiating factors identified are: Height of the individual and Hair Length. Following is a scatter plot of the sample.



The blue circles in the plot represent females and green squares represents male. A few expected insights from the graph are:

1. Males in our population have a higher average height.
2. Females in our population have longer scalp hairs.

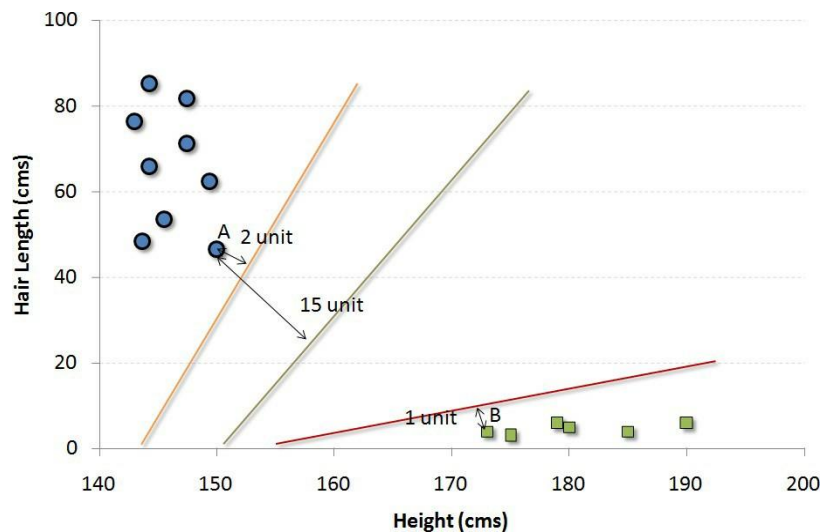
If we were to see an individual with height 180 cm's and hair length 4 cm's, our best guess will be to classify this individual as a male. This is how we do a classification analysis.

A Support Vector and SVM

Support Vectors are simply the co-ordinates of individual observation. For instance, (145, 85) is a support vector which corresponds to a female. Support Vector Machine is a frontier which best segregates the Male from the Females. In this case, the two classes are well separated from each other; hence it is easier to find a SVM.

How to find the Support Vector Machine for case in hand?

There are many possible frontiers which can classify the problem in hand. Following are the three possible frontiers.

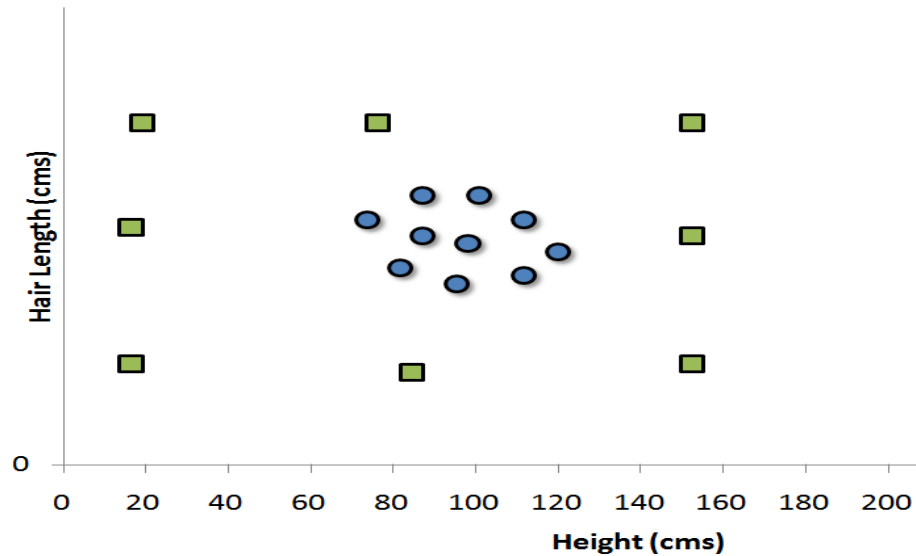


How do we decide which is the best frontier for this particular problemstatement?

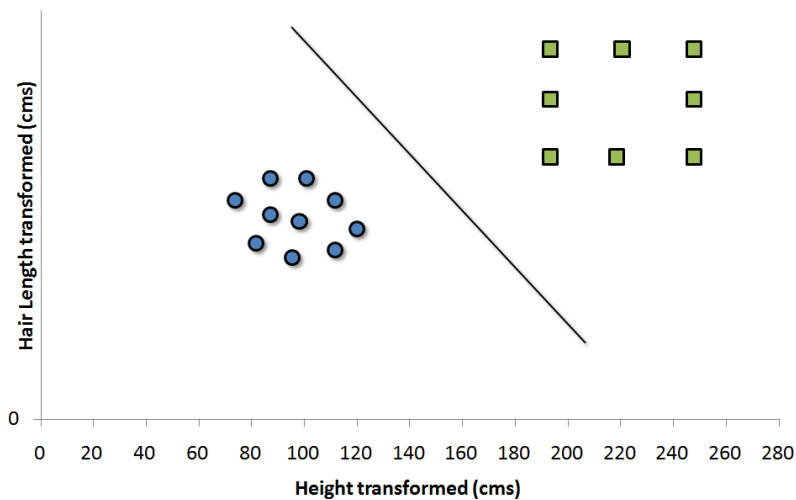
The easiest way to interpret the objective function in a SVM is to find the minimum distance of the frontier from closest support vector (this can belong to any class). For instance, orange frontier is closest to blue circles. And the closest blue circle is 2 units away from the frontier. Once we have these distances for all the frontiers, we simply choose the frontier with the maximum distance (from the closest support vector). Out of the three shown frontiers, we see the black frontier is farthest from nearest support vector (i.e. 15units).

What if we do not find a clean frontier which segregates the classes?

Our job was relatively easier finding the SVM in this business case. What if the distribution looked something like as follows?



In such cases, we do not see a straight line frontier directly in current plane which can serve as the SVM. In such cases, we need to map these vectors to a higher dimension plane so that they get segregated from each other. Such cases will be covered once we start with the formulation of SVM. For now, you can visualize that such transformation will result into following type of SVM.



Each of the green squares in original distribution is mapped on a transformed scale. And the transformed scale has clearly

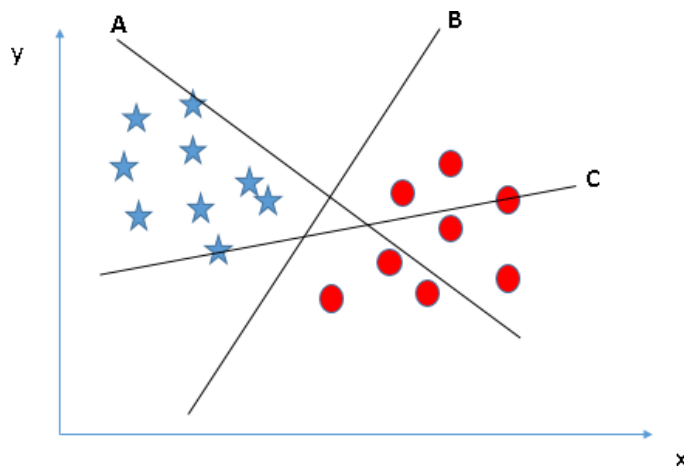
segregated classes. Many algorithms have been proposed to make these transformations.

How does it work?

Above, we got accustomed to the process of segregating the two classes with a hyper-plane. Now the burning question is “How can we identify the right hyper-plane?” Don’t worry; it’s not as hard as you think!

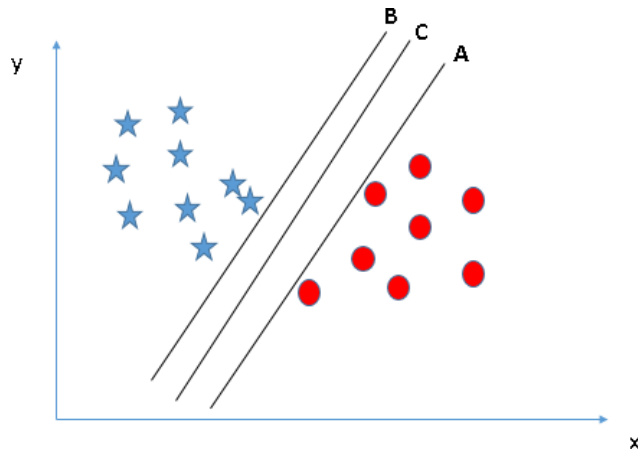
Let’s understand:

- **Identify the right hyper-plane (Scenario-1):** Here, we have three hyper-planes (A, B and C). Now, identify the right hyper-plane to classify star and circle.

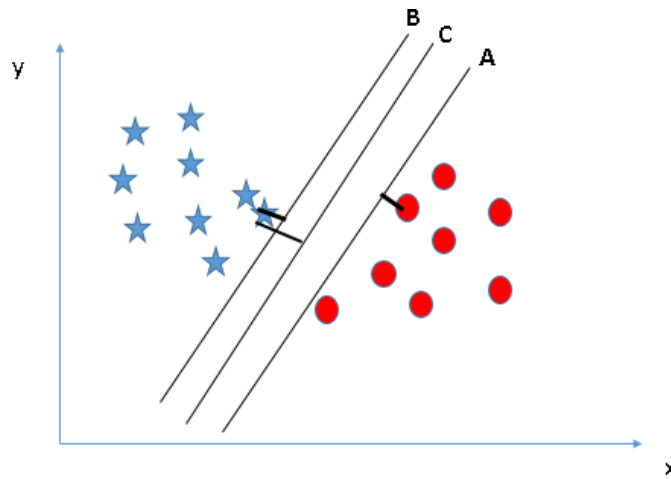


You need to remember a thumb rule to identify the right hyper-plane: “Select the hyper-plane which segregates the two classes better”. In this scenario, hyper-plane “B” has excellently performed this job.

- **Identify the right hyper-plane (Scenario-2):** Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Now, how can we identify the right hyper-plane?

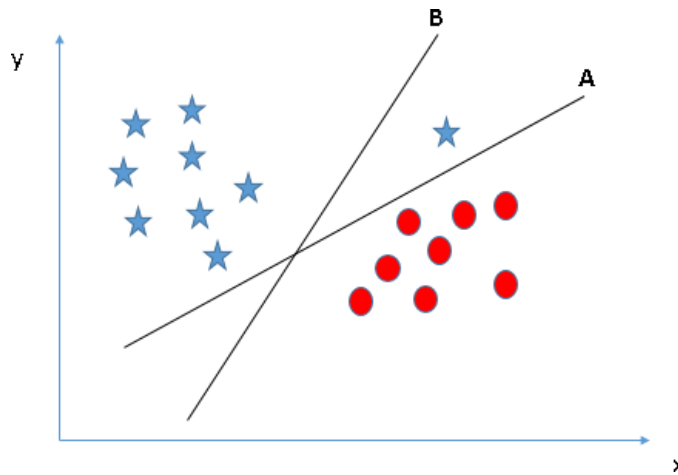


Here, maximizing the distances between nearest data points (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as **Margin**. Let's look at the below snapshot:



Above, you can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is high chance of miss-classification.

- **Identify the right hyper-plane (Scenario-3):** Hint: Use the rules as discussed in previous section to identify the right hyper-plane

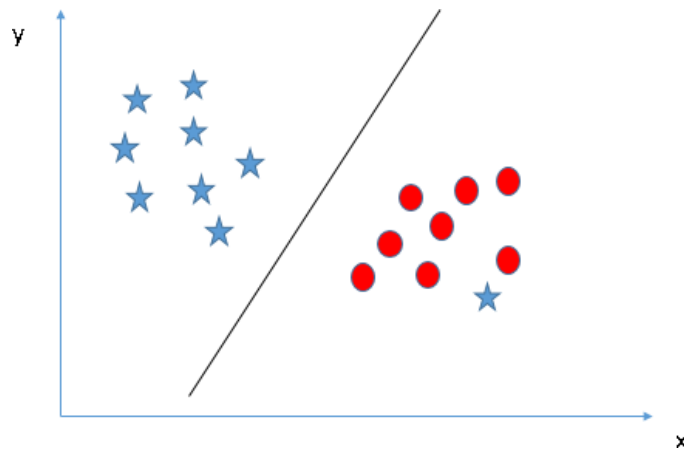


Some of you may have selected the hyper-plane **B** as it has higher margin compared to **A**. But, here is the catch; SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane **B** has a classification error and **A** has classified all correctly. Therefore, the right hyper-plane is **A**.

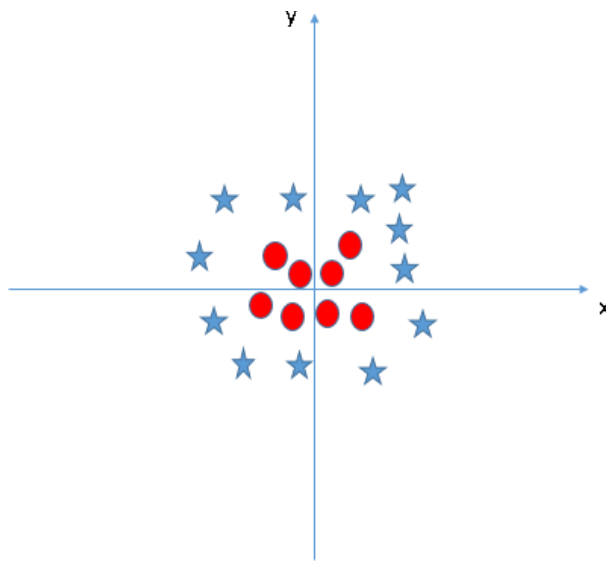
- **Can we classify two classes (Scenario-4)?:** Below, I am unable to segregate the two classes using a straight line, as one of star lies in the territory of other(circle) class as an outlier.



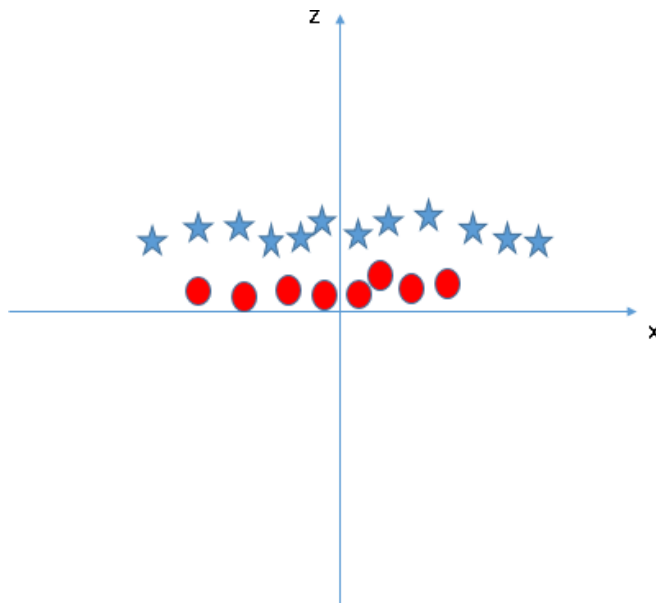
As I have already mentioned, one star at other end is like an outlier for star class. SVM has a feature to ignore outliers and find the hyper-plane that has maximum margin. Hence, we can say, SVM is robust to outliers.



- **Find the hyper-plane to segregate to classes (Scenario-5):** In the scenario below, we can ' t have linear hyper-plane between the two classes, so how does SVM classify these two classes? Till now, we have only looked at the linearhyper-plane.



SVM can solve this problem. Easily! It solves this problem by introducing additional feature. Here, we will add a new feature $z = x^2 + y^2$. Now, let's plot the data points on axis x and z:

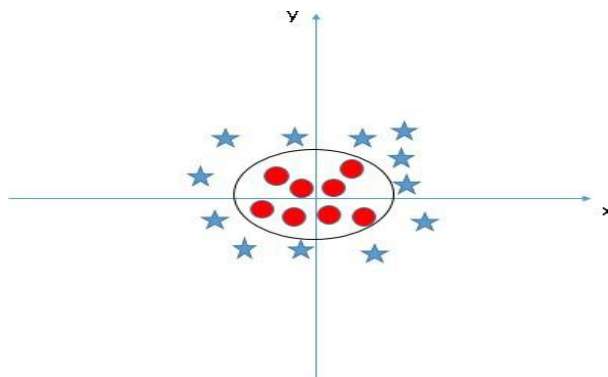


In above plot, points to consider are:

- o All values for z would be positive always because z is the squared sum of both x and y
- o In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z .

In SVM, it is easy to have a linear hyper-plane between these two classes. But, another burning question which arises is, should we need to add this feature manually to have a hyper-plane. No, SVM has a technique called the **kerneltrick**. These are functions which takes low dimensional input space and transform it to a higher dimensional space i.e. it converts not separable problem to separable problem, these functions are called kernels. It is mostly useful in non-linear separation problem. Simply put, it does some extremely complex data transformations, then find out the process to separate the data based on the labels or outputs you've defined.

When we look at the hyper-plane in original input space it looks like a circle:



Pros and Cons associated with SVM

- **Pros:**
 - o It works really well with clear margin of separation

- o It is effective in high dimensional spaces.
- o It is effective in cases where number of dimensions is greater than the number of samples.
- o It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- **Cons:**
 - o It doesn't perform well, when we have large data set because the required training time is higher
 - o It also doesn't perform very well, when the data set has more noise i.e. target classes are overlapping
 - o SVM doesn't directly provide probability estimates, these are calculated using an expensive five-fold cross-validation. It is related SVC method of Python scikit-learn library

CHAPTER 5

DESIGN AND IMPLEMENTATION

Design and Implementation Constraints

Constraints in Analysis

- ◆ Constraints as Informal Text
- ◆ Constraints as Operational Restrictions
- ◆ Constraints Integrated in Existing Model Concepts
- ◆ Constraints as a Separate Concept
- ◆ Constraints Implied by the Model Structure

Constraints in Design

- ◆ Determination of the Involved Classes
- ◆ Determination of the Involved Objects
- ◆ Determination of the Involved Actions
- ◆ Determination of the Require Clauses
- ◆ Global actions and Constraint Realization

Constraints in Implementation

A hierarchical structuring of relations may result in more classes and a more complicated structure to implement. Therefore it is advisable to transform the hierarchical relation structure to a

simpler structure such as a classical flat one. It is rather straightforward to transform the developed hierarchical model into a bipartite, flat model, consisting of classes on the one hand and flat relations on the other. Flat relations are preferred at the design level for reasons of simplicity and implementation ease. There is no identity or functionality associated with a flat relation. A flat relation corresponds with the relation concept of entity-relationship modeling and many object oriented methods.

OTHER NONFUNCTIONAL REQUIREMENTS

Performance Requirements

The application at this side controls and communicates with the following three main general components.

- embedded browser in charge of the navigation and accessing to the web service;
- Server Tier: The server side contains the main parts of the functionality of the proposed architecture. The components at this tier are the following.

Web Server, Security Module, Server-Side Capturing Engine, Preprocessing Engine, Database System, Verification Engine, Output Module.

Safety Requirements

1. The software may be safety-critical. If so, there are issues associated with its integrity level
2. The software may not be safety-critical although it forms part of a safety-critical system. For example, software may simply log transactions.
3. If a system must be of a high integrity level and if the software is shown to be of that integrity level, then the hardware must be at least of the same integrity level.
4. There is little point in producing 'perfect' code in some language if hardware and system software (in widest sense) are not reliable.
5. If a computer system is to run software of a high integrity level then that system should not at the same time accommodate software of a lower integrity level.
6. Systems with different requirements for safety levels must be separated.
7. Otherwise, the highest level of integrity required must be applied to all systems in the same environment.

CHAPTER 6

SYSTEM DESIGN

ARCHITECTURE DIAGRAM:

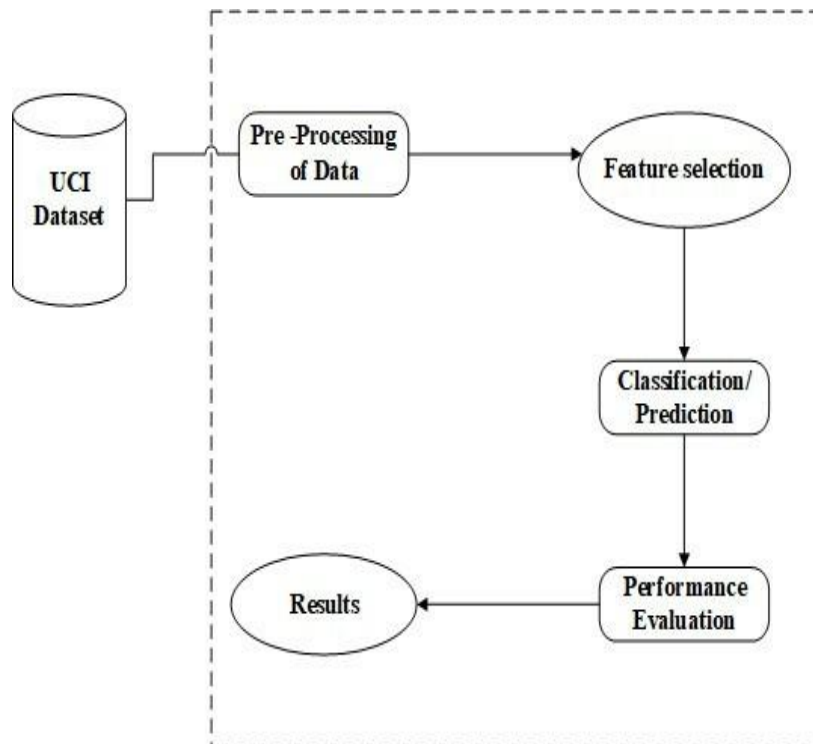


Figure: 6.1 Proposed Architecture

SEQUENCE DIAGRAM:

A Sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of Message Sequence diagrams are sometimes called

event diagrams, event sceneries and timing diagram.

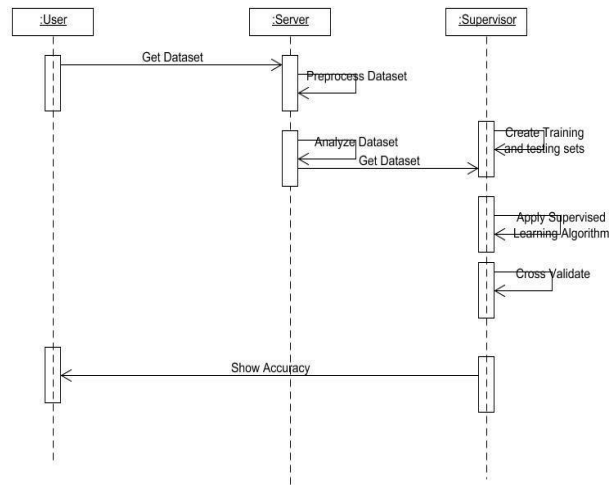


Figure: 6.2 Sequential Diagram

USE CASE DIAGRAM:

Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of software engineering. The standard is managed and was created by the Object Management Group. UML includes a set of graphic notation techniques to create visual models of software intensive systems. This language is used to specify, visualize, modify, construct and document the artifacts of an object oriented software intensive system under development.

USECASEDIAGRAM

A Use case Diagram is used to present a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those use cases. Use case diagram consists of two parts:

Use case: A use case describes a sequence of actions that provided something of measurable value to an actor and is drawn as a horizontal ellipse.

Actor: An actor is a person, organization or external system that plays a role in one or more interaction with the system.

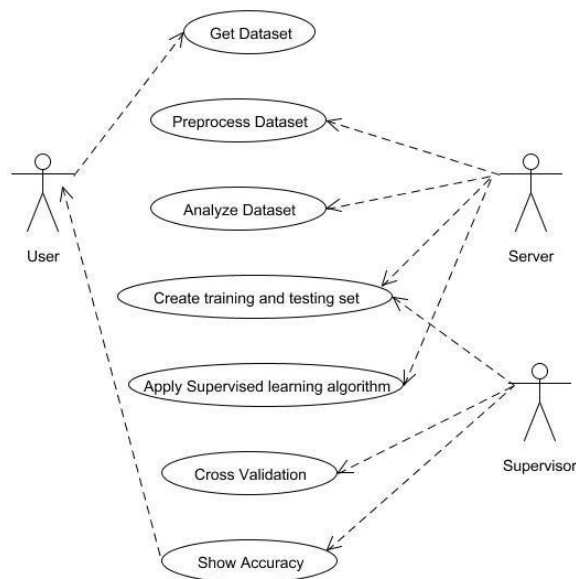


Figure : 6.3.1 Use case Diagram

ACTIVITYDIAGRAM:

Activity diagram is a graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. An activity diagram shows the overall flow of control.

The most important shape types:

- Rounded rectangles represent activities.
- Diamonds represent decisions.
- Bars represent the start or end of concurrent activities.
- A black circle represents the start of the workflow.
- An encircled circle represents the end of the workflow.

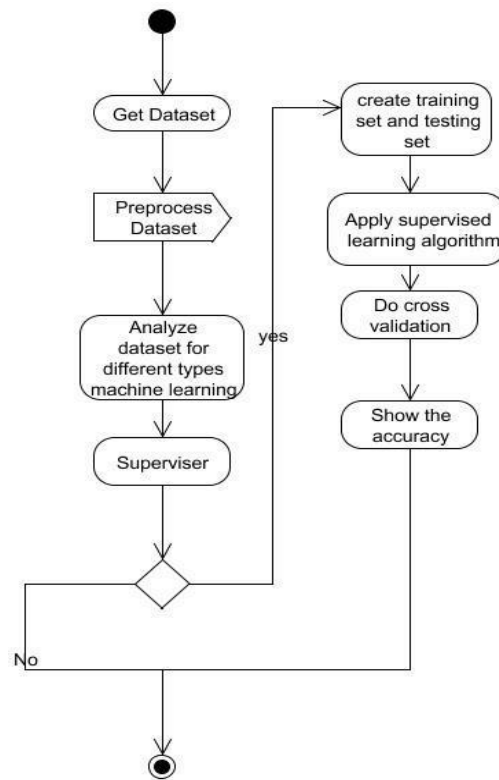


Figure :6.4 Activity Diagram

COLLABORATIONDIAGRAM:

UML Collaboration Diagrams illustrate the relationship and interaction between software objects. They require use cases, system operation contracts and domain model to already exist. The collaboration diagram illustrates messages being sent between classes and objects.

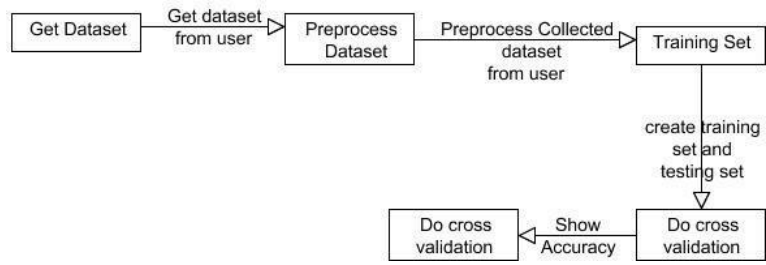


Figure :6.5 Collaboration Diagram.

CHAPTE R 7 MODULE S

- DataPre-Processing
- Feature Selection And Reduction
- Classification Modeling
- Performance Measures

MODULE DESCRIPTION:

(1) Data Pre-Processing

Heart disease data is pre-processed after collection of various records. The dataset contains a total of 303 patient records, where 6 records are with some missing values. Those 6 records have been removed from the dataset and the remaining 297 patient records are used in pre-processing.

(2) Feature Selection and Reduction

From among the 13 attributes of the data set, two attributes pertaining to age and sex are used to identify the personal information of the patient. The remaining 11 attributes are considered important as they contain vital clinical records. Clinical

records are vital to diagnosis and learning the severity of heart disease.

(3) Classification Modeling

The clustering of datasets is done on the basis of the variables and criteria of Decision Tree (DT) features. Then, the classifiers are applied to each clustered dataset in order to estimate its performance. The best performing models are identified from the above results based on their low rate of error.

- Decision Trees Classifier
- Support Vector Classifier
- Random Forest Classifier
- K- Nearest Neighbour

(4) Performance Measures:

Several standard performance metrics such as accuracy, precision and error in classification have been considered for the computation of performance efficacy of this model.

CHAPTER 8

CODING AND TESTING

CODING

Once the design aspect of the system is finalized, the system enters into the coding and testing phase. The coding phase brings the actual system into action by converting the design of the system into the code in a given programming language. Therefore, a good coding style has to be taken whenever changes are required; it is easily screwed into the system.

CODING STANDARDS

Coding standards are guidelines to programming that focus on the physical structure and appearance of the program. They make the code easier to read, understand, and maintain. This phase of the system actually implements the blueprint developed during the design phase. The coding specification should be in such a way that any programmer must be able to understand the code and can bring about changes whenever felt necessary. Some of the standards needed to achieve the above-mentioned objectives are as follows:

- ✓ Program should be simple, clear, and easy to understand.
- ✓ Naming conventions
- ✓ Value conventions

✓ Script and comment procedure

- ✓ Message box format
- ✓ Exception and error handling

NAMING CONVENTIONS

Naming conventions of classes, data member, member functions, procedures etc., should be **self-descriptive**. One should even get the meaning and scope of the variable by its name. The conventions are adopted for **easy understanding** of the intended message by the user. So it is customary to follow the conventions. These conventions are as follows:

Class names

Class names are problem domain equivalence and begin with capital letter and have mixed cases.

Member Function and Data Member name

Member function and data member name begins with a lowercase letter with each subsequent letters of the new words in uppercase and the rest of letters in lowercase.

VALUE CONVENTIONS

Value conventions ensure values for variable at any point of time. This involves the following:

- Proper default values for the variables.
- Proper validation of values in the field.
- Proper documentation of flag values.

SCRIPT WRITING AND COMMENTING STANDARD

Script writing is an art in which indentation is utmost important. Conditional and looping statements are to be properly aligned to facilitate easy understanding. Comments are included to minimize the number of surprises that could occur when going through the code.

MESSAGE BOX FORMAT

When something has to be prompted to the user, he must be able to understand it properly. To achieve this, a specific format has been adopted in displaying messages to the user. They are as follows:

- X – User has performed illegal operation.
- ! – Information to the user.

TEST

PROCEDURE

SYSTEM TESTING

Testing is performed to identify errors. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example the design must not have any logic faults in the design is detected before coding commences, otherwise the cost of fixing the faults will be

considerably higher as reflected. Detection of design faults can be

achieved by means of inspection as well as walkthrough.

Testing is one of the important steps in the software development phase. Testing checks for the errors, as a whole of the project testing involves the following testcases:

- Static analysis is used to investigate the structural properties of the Source code.
- Dynamic testing is used to investigate the behavior of the source code by executing the program on the testdata.

TEST DATA AND OUTPUT

UNIT TESTING

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing.

FUNCTIONAL TESTS

Functional test cases involved exercising the code with nominal input values for which the expected results are known, as well as boundary values and special values, such as logically related inputs, files of identical elements, and empty files.

Three types of tests in Functionaltest:

- PerformanceTest
- Stress Test
- StructureTest

PERFORMANCE TEST

It determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization by the programunit.

8..4.4 STRESS TEST

Stress Test is those test designed to intentionally break the unit. A Great deal can be learned about the strength and limitations of a program by examining the manner in which a programmer in which a program unitbreaks.

STRUCTURED TEST

Structure Tests are concerned with exercising the internal logic of a program and traversing particular execution paths. The way in which White-Box test strategy was employed to ensure that the test cases could Guarantee that all independent paths within a module have been have been exercised at leastonce.

- Exercise all logical decisions on their true or falsesides.
- Execute all loops at their boundaries and within their operational bounds.
- Exercise internal data structures to assure theirvalidity.

- Checking attributes for their correctness.
- Handling end of file condition, I/O errors, buffer problems and textual errors in output information

INTEGRATION TESTING

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which another module is added and tested in combination. And so on. The advantages of this approach are that, interface dispenses can be easily found and corrected.

The major error that was faced during the project is linking error. When all the modules are combined the link is not set properly with all support files. Then we checked out for interconnection and the links. Errors are localized to the new module and its intercommunications. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

TESTING TECHNIQUES / TESTING STRATEGIES

TESTING

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet – undiscovered error. A successful test is one that uncovers an as-yet- undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of program hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing.

The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise the program or the project is not said to be complete. Software testing is the critical element of software quality assurance and represents the ultimate the review of specification design and coding. Testing is the process of executing the program with the intent of finding the error. A good test case design is one that as a probability of finding an yet undiscovered error. A successful test is one that uncovers an yet undiscovered error. Any engineering product can be tested in one of the two ways:

WHITE BOX TESTING

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

Basis path testing:

- Flow graphnotation
- Cyclometriccomplexity
- Deriving testcases
- Graph matricesControl

BLACK BOX TESTING

In this testing by knowing the internal operation of a product, test can be conducted to ensure that “ all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

- Graph based testing methods
- Equivalencepartitioning

- Boundary value analysis
- Comparison testing

SOFTWARE TESTING STRATEGIES:

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically. For this reason a template for software testing a set of steps into which we can place specific test case design methods should be strategy should have the following characteristics:

- Testing begins at the module level and works “outward” toward the integration of the entire computer based system.
- Different testing techniques are appropriate at different points in time.
- The developer of the software and an independent test group conduct testing.
- Testing and Debugging are different activities but debugging must be accommodated in any testing strategy.

INTEGRATION TESTING:

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors, should not be

assumed to work instantly when we put them together. The problem of course, is “putting them together”- interfacing. There may be the chances of data lost across on another 's sub functions, when combined may not produce the desired major function; individually acceptable impression may be magnified to unacceptable levels; global data structures can present problems.

PROGRAM TESTING:

The logical and syntax errors have been pointed out by program testing. A syntax error is an error in a program statement that in violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted keywords are common syntax error. These errors are shown through error messages generated by the computer. A logic error on the other hand deals with the incorrect data fields, out-off-range items and invalid combinations. Since the compiler s will not deduct logical error, the programmer must examine the output. Condition testing exercises the logical conditions contained in a module. The possible types of elements in a condition include a Boolean operator, Boolean variable, a pair of Boolean parentheses A relational operator or on arithmetic expression. Condition testing method focuses on testing each condition in the program the purpose of condition test is to deduct not only errors in the condition of a program but also other a errors in the program.

SECURITY TESTING:

Security testing attempts to verify the protection mechanisms built in to a system well, in fact, protect it from improper penetration. The system security must be tested for invulnerability from frontal attack must also be tested for invulnerability from rear attack. During security, the tester places the role of individual who desires to penetrate system.

VALIDATION TESTING

At the culmination of integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test-validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the customer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirement. After validation test has been conducted, one of two conditions exists.

- * The function or performance characteristics confirm to specifications and are accepted.
- * A validation from specification is uncovered and a deficiency created.

Deviation or errors discovered at this step in this project is corrected prior to completion of the project with the help of the by negotiating to establish a method for resolving deficiencies. Thus the

proposed system under consideration has been tested by using validation testing and found to be working satisfactorily. Though there were deficiencies in the system they were not catastrophic

USER ACCEPTANCE TESTING

User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required.

This is done in regarding to the following points.

- ✓ Input screendesign
- ✓ Output screendesign.

.

CHAPTER 9

RESULTS AND DISCUSSION

The below stated figures, plots gives a detailed explanation of the output of the project.

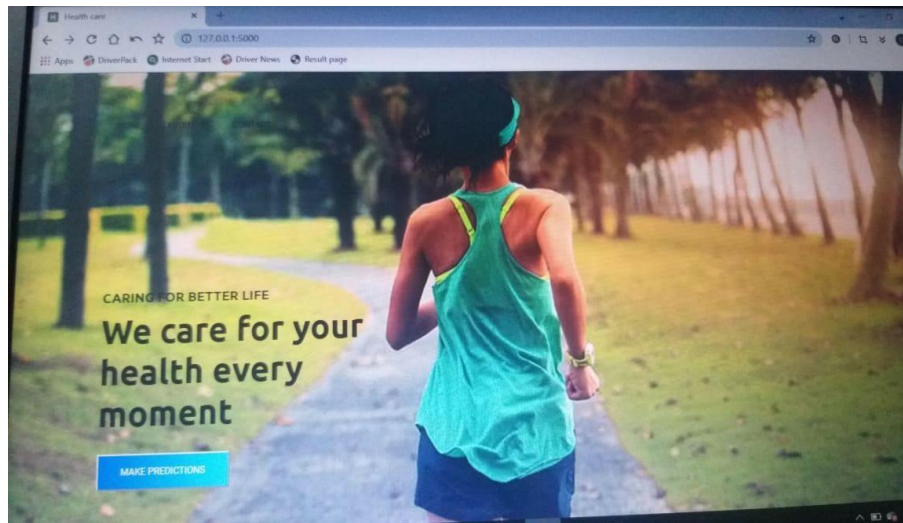


Figure9.1:Health Care measures against Heart Disease

The above figure 9.1 is the home page or main page to find the accuracy by predicting heart disease. In this page we can just fill the required details of a patient by clicking on the Make Predictions Button so that we can find whether the patient is with Heart disease or Not.

Once you click the Make predictions button there comes an Heart Diseases Prediction Form there we must fill the age, sex, chest pain, Resting Blood pressure, serum cholesterol , Fasting blood sugar, resting electro graphic results, maximum heart rate achieved, exercise induced angina, ST depression induced by exercise ,slope of the ST segment, number of major vessels

coloured by flours copy and defect type. Now after adding up the details click predicts normality shows No Heart Disease or else it shows as Heart disease. Once we have predicted, with the given data set it shows as givenbelow.

result page

Heart Disease Prediction Form

Age
55

Sex
male

Chest Pain
asymptomatic

Resting blood pressure
150

Serum cholestrol
160

Fasting Blood Sugar

it page

Fasting Blood Sugar
≤ 120 mg/dl

Resting electrographic results
Normal

Maximum heart rate achieved
150

Exercise induced angina -(Exang)
no

ST depression induced by exercise
0

Slope of the ST segment
Unknown

Result page

no

ST depression induced by exercise

0

Slope of the ST segment

Unknown

Number of major vessels colored by flourscopy

Unknown

Defect type

Unknown

PREDICT RESET

Figure 9.2: Heart Disease Prediction Form with dataset to diagnose

Once we have predicted with the given data set as mentioned above in the Figure:9.2 there shows all the general and most important data for an individual person.

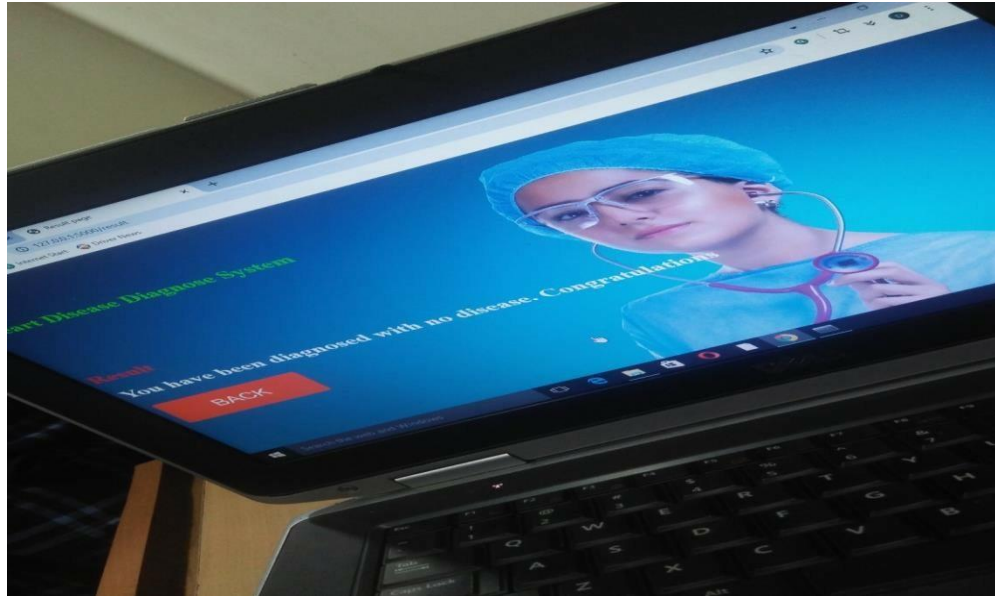


Figure:9.3 Heart Disease is Diagnosed for thatpatient.



Figure:9.4 Heart disease is not diagnosed for thatPatient

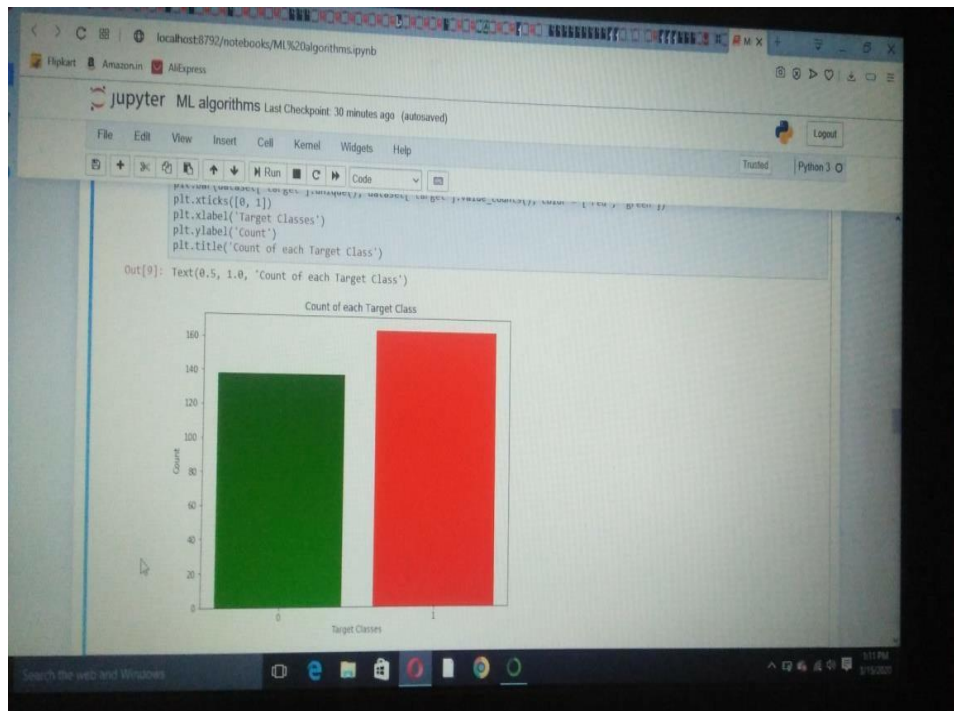
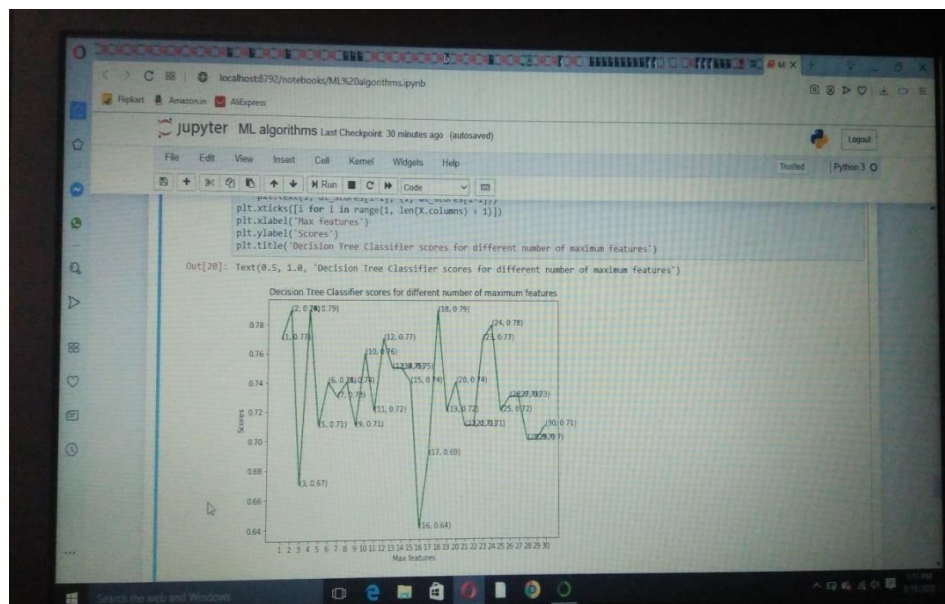


Figure:9.5 Target 0 shows No Heart

Disease(GREEN)Target 1 shows Heart



Disease(RED)

Figure:9.6 Plot Chart using Decision Tree Classifier

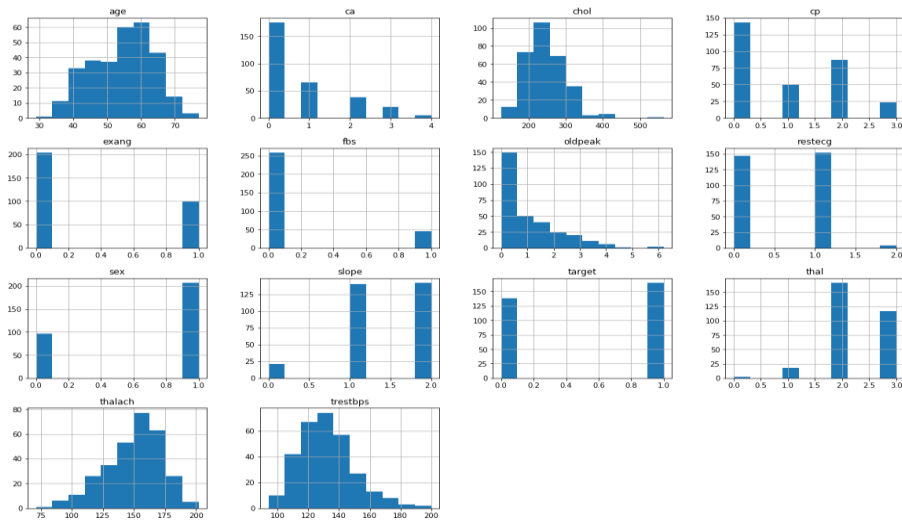


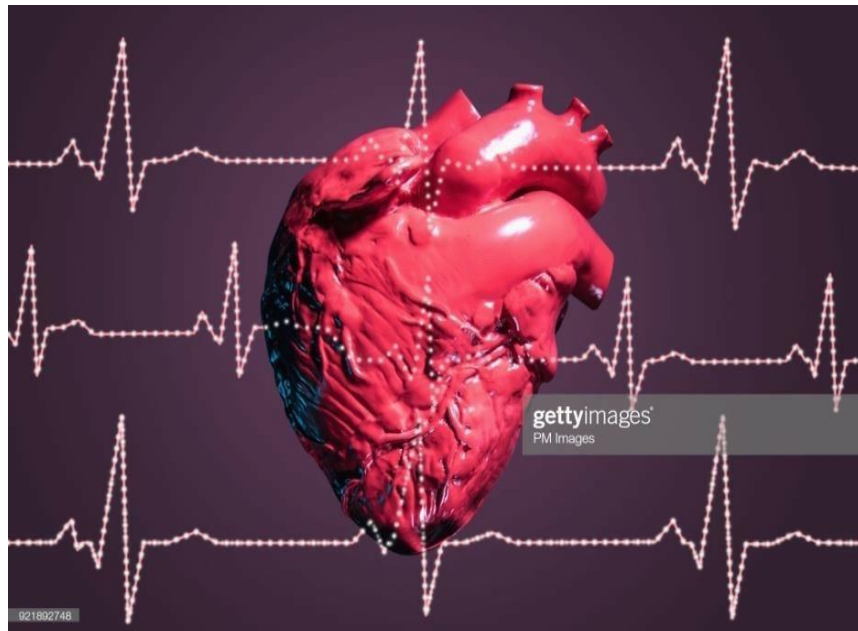
Figure:9.7 Bar Representation.

CHAPTER 10

CONCLUSION AND FUTURE WORK

CONCLUSION

According to a 2010 survey, for every \$6 spent on health care, \$1 is for Cardio vascular diseases. Coronary Heart Disease (CHD) is the leading cause of death of 3,70,000 people annually. But the cost associated with their treatment is estimated to be around \$444 billion (US) dollars. We produce an enhanced performance level with an accuracy level of 88:7% through the prediction model for heart disease with the hybrid random forest with a linear model(HRFLM).



FUTUREWORK

In future, an interactive platform can be built between the doctors and the patients to communicate about risks and factors. This is an idea to calculate the accuracy by predicting the heart disease using some techniques. It would be really helpful in this world to lead a healthy and happy life.



REFERENCES

[1] A. S. Abdullah and R. R. Rajalaxmi, "A data mining model for predicting the coronary heart disease using random forest classifier," in Proc. Int. Conf. Recent Trends Comput. Methods, Commun. Controls, Apr. 2012, pp.22_25.

[2] A. H. Alkeshuosh, M. Z. Moghadam, I. Al Mansoori, and M. Abdar, "Using PSO algorithm for producing best rules in diagnosis of heart disease," in Proc. Int. Conf. Comput. Appl. (ICCA), Sep. 2017, pp. 306_311.

[3] N. Al-milli, "Backpropagation neural network for prediction of heart disease," J. Theor. Appl. Inf. Technol., vol. 56, no. 1, pp. 131_135, 2013.

[4] C. A. Devi, S. P. Rajamhoana, K. Umamaheswari, R. Kiruba, K. Karunya, and R. Deepika, "Analysis of neural networks based heart disease prediction system," in Proc. 11th Int. Conf. Hum. Syst. Interact. (HSI), Gdansk, Poland, Jul. 2018, pp.233_239.

[5] P. K. Anooj, "Clinical decision support system: Risk level prediction of heart disease using weighted fuzzy rules," J. King Saud Univ.-Comput. Inf. Sci., vol. 24, no. 1, pp. 27_40, Jan. 2012. doi: 10.1016/j.jksuci.2011.09.002.

[6] L. Baccour, "Amended fused TOPSIS-VIKOR for classification (ATOVIC) applied to some UCI data sets," *Expert Syst. Appl.*, vol. 99, pp. 115_125, Jun. 2018. doi:10.1016/j.eswa.2018.01.025.

[7] C.-A. Cheng and H.-W. Chiu, "An artificial neural network model for the evaluation of carotid artery stenting prognosis using a national-wide database," in *Proc. 39th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Jul. 2017, pp.2566_2569.

[8] H. A. Esfahani and M. Ghazanfari, "Cardiovascular disease detection using a new ensemble classifier," in *Proc. IEEE 4th Int. Conf. Knowl.- Based Eng. Innov. (KBEI)*, Dec. 2017, pp.1011_1014.

[9] F. Dammak, L. Baccour, and A. M. Alimi, "The impact of criterion weights techniques in TOPSIS method of multi-criteria decision making in crisp and intuitionistic fuzzy domains," in *Proc. IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE)*, vol. 9, Aug. 2015, pp.1_8.

[10] R. Das, I. Turkoglu, and A. Sengur, "Effective diagnosis of heart disease through neural networks ensembles," *Expert Syst. Appl.*, vol. 36, no. 4, pp. 7675_7680, May 2009. doi: 10.1016/j.eswa.2008.09.013.

[11] M. Durairaj and V. Revathi, "Prediction of heart disease using back propagationMLPalgorithm," *Int. J. Sci. Technol. Res.*, vol. 4, no. 8, pp. 235_239,2015.

[12] M. Gandhi and S. N. Singh, "Predictions in heart disease using techniques of data mining," in Proc. Int. Conf. Futuristic Trends Comput. Anal. Knowl. Manage. (ABLAZE), Feb. 2015, pp.520_525.

[13] A. Gavhane, G. Kokkula, I. Pandya, and K. Devadkar, "Prediction of heart disease using machine learning," in Proc. 2nd Int. Conf. Electron., Commun. Aerosp. Technol. (ICECA), Mar. 2018, pp.1275_1278.

[14] B. S. S. Rathnayakc and G. U. Ganegoda, "Heart diseases prediction with data mining and neural network techniques," in Proc. 3rd Int. Conf. Conver. Technol. (I2CT), Apr. 2018, pp.1_6.

[15] N. K. S. Banu and S. Swamy, "Prediction of heart disease at early stage using data mining and big data analytics: A survey," in Proc. Int. Conf. Elect., Electron., Commun., Comput. Optim. Techn. (ICEECCOT), Dec. 2016, pp. 256_261.81

APPENDIX

A. SOURCE CODE

`# In[1]:`

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import
rcParams from matplotlib.cm
import trainbow
get_ipython().run_line_magic('matplotlib', 'inline')
import warnings
warnings.filterwarnings('ignore')
```

`# In[2]:`

```
from sklearn.model_selection import
train_test_split from sklearn.preprocessing import
StandardScaler
```

`# In[3]:`

```
from sklearn.svm import SVC
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
# In[4]:

dataset=pd.read_csv('C:/Users/gts/Downloads/Heart-
Disease-
Prediction-master/dataset.csv') dataset

# In[5]:

dataset.info()

# In[6]:

dataset.describe()

# In[7]:

rcParams['figure.figsize'] = 20, 14
plt.matshow(dataset.corr())
plt.yticks(np.arange(dataset.shape[1]),
dataset.columns)
plt.xticks(np.arange(dataset.shape[1]),
dataset.columns) plt.colorbar()
```

```
# In[8]:
```

```
dataset.hist()
```

```
# In[9]:
```

```
rcParams['figure.figsize'] = 8,6  
plt.bar(dataset['target'].unique(),  
dataset['target'].value_counts(), color = ['red', 'green'])  
plt.xticks([0, 1])  
plt.xlabel('Target Classes')  
plt.ylabel('Count')  
plt.title('Count of each Target Class')
```

```
# In[10]:
```

```
dataset = pd.get_dummies(dataset, columns = ['sex',  
'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal'])
```

```
# In[11]:
```

```
standardScaler = StandardScaler()  
columns_to_scale = ['age', 'trestbps',  
'chol', 'thalach', 'oldpeak']
```

```

dataset[columns_to_scale] =
standardScaler.fit_transform(dataset[columns_to_scale])

# In[12]:

y = dataset['target']
X = dataset.drop(['target'], axis = 1)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.33, random_state = 0)

# In[13]:

svc_scores = []
kernels = ['linear', 'poly', 'rbf', 'sigmoid']
for i in range(len(kernels)):
    svc_classifier = SVC(kernel = kernels[i])
    svc_classifier.fit(X_train, y_train)
    svc_scores.append(svc_classifier.score(X_test, y_test))

# In[14]:

colors = rainbow(np.linspace(0, 1, len(kernels)))
plt.bar(kernels, svc_scores, color = colors)
for i in range(len(kernels)):
    plt.text(i, svc_scores[i], svc_scores[i])
plt.xlabel('Kernels')

```

```
plt.ylabel('Scores')
plt.title('Support Vector Classifier scores for different
kernels')
```

```
# In[15]:
```

```
print("The score for Support Vector Classifier is {}%with
{}
kernel.".format(svc_scores[0]*100, 'linear'))
```

```
# In[16]:
```

```
dt_scores = []
for i in range(1, len(X.columns) + 1):
    dt_classifier = DecisionTreeClassifier(max_features = i,
    random_state = 0)
```

```
dt_classifier.fit(X_train, y_train)
```

```
dt_scores.append(dt_classifier.score(X_test, y_test))
```

```
# In[17]:
```

```
plt.plot([i for i in range(1, len(X.columns) + 1)], dt_scores,
```

```

color = 'green')
for i in range(1, len(X.columns) + 1):
    plt.text(i, dt_scores[i-1], (i, dt_scores[i-1]))
plt.xticks([i for i in range(1, len(X.columns) + 1)])
plt.xlabel('Max features')
plt.ylabel('Scores')
plt.title('Decision Tree Classifier scores for different
number of maximum features')

```

```
# In[18]:
```

```

print("The score for Decision Tree Classifier is {}% with
maximum features.".format(dt_scores[17]*100, [2,4,18]))

```

```
# In[19]:
```

```

rf_scores = []
estimators = [10, 100, 200, 500,
1000] for i in estimators:
    rf_classifier = RandomForestClassifier(n_estimators = i,
random_state = 0)
    rf_classifier.fit(X_train, y_train)
    rf_scores.append(rf_classifier.score(X_test, y_test))#

```

```
# In[20]:
```

```
colors=rainbow(np.linspace(0,1,len(estimators)))
plt.bar([i for i in range(len(estimators))], rf_scores, color=
colors, width=0.8)
for i in range(len(estimators)):
    plt.text(i, rf_scores[i], rf_scores[i])
plt.xticks(ticks=[i for i in range(len(estimators))], labels=
[str(estimator) for estimator in estimators])
plt.xlabel('Number of estimators')
plt.ylabel('Scores')
plt.title('Random Forest Classifier scores for different
number of estimators')
```

```
# In[21]:
```

```
print("The score for Random Forest Classifier is {}% with {}
estimators.".format(rf_scores[1]*100, [100, 500]))
```

```
# In[22]:
```

B. SCREENSHOTS:

```

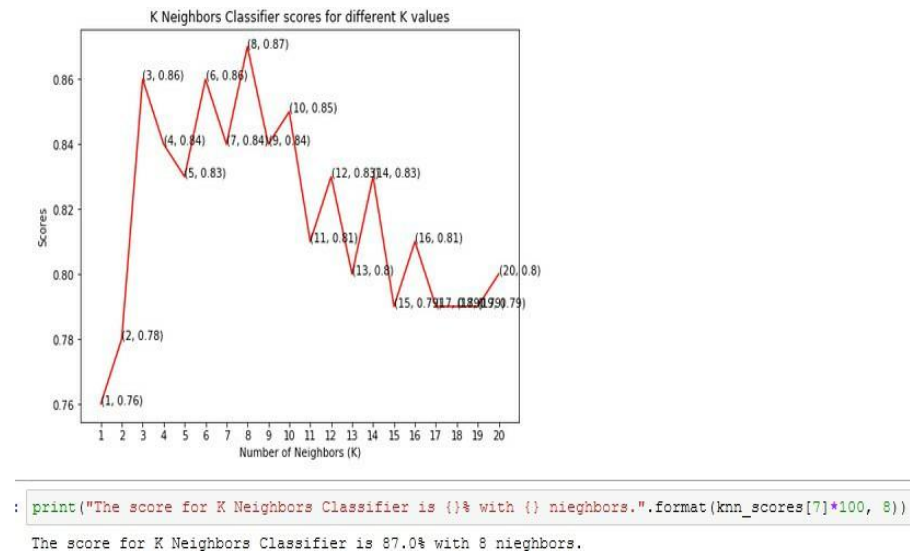
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

E:\Python_19-20_Projects\Heart_Diseases\Module - 1>python preprocess.py
Cleveland data. Size=(302, 14)
Number of missing values
age          0
sex          0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          4
thal        2
num         0
dtype: int64

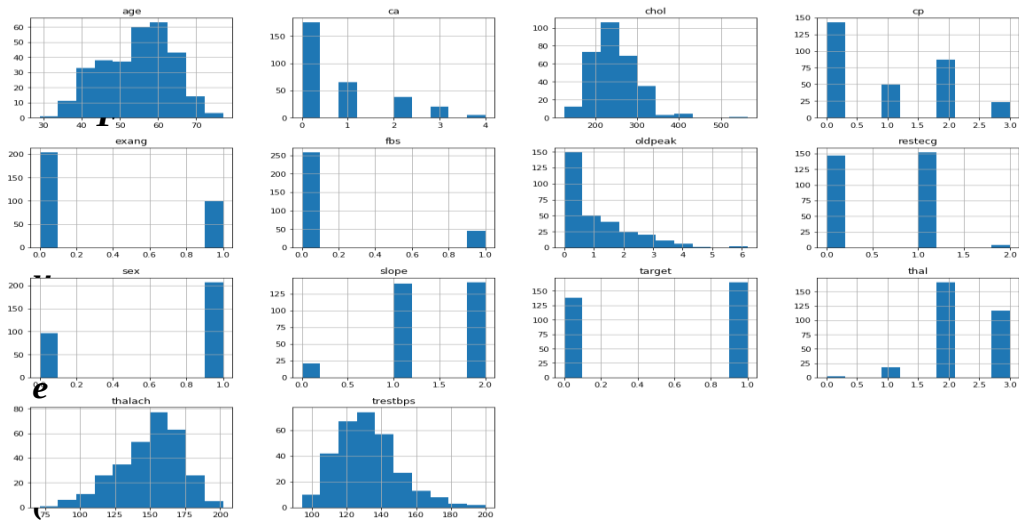
Hungary data.: Size=(293, 14)
Number of missing values
age          0

```

Figure B(1): Preprocessing Data set



Figure(B)2 : K neighbor classifier



: **Figure (B)3 : BarRepresentation**

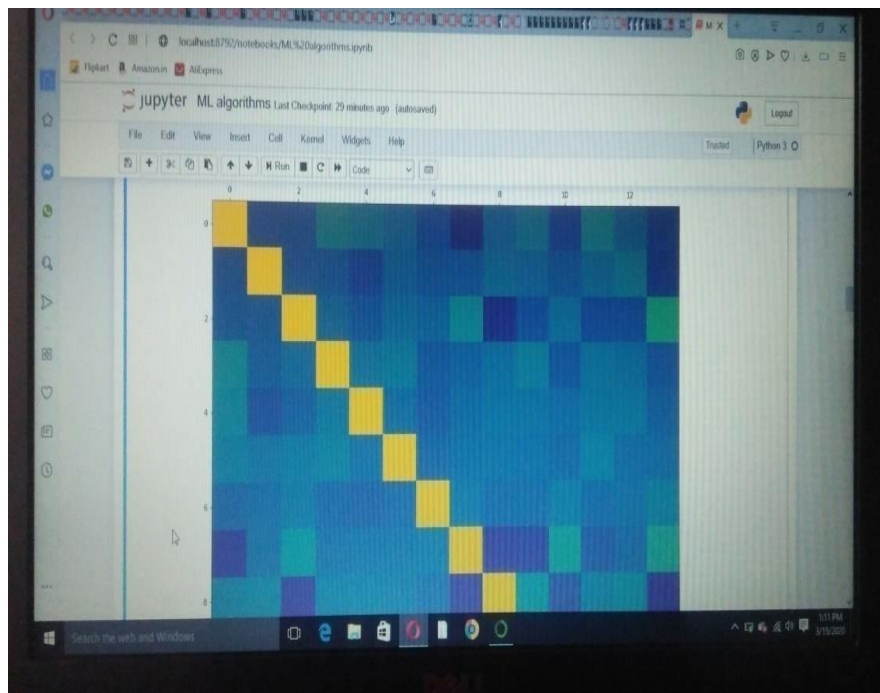


Figure (B)4 : Feature Selection



Figure B(5) : Health Care Agency



Figure (B)6 : Medicines to prevent Heart Disease