PERFORMANCE ENHANCEMENT OF KNN ALGORITHM USING 8-BIN HASHING AND FEATURE WEIGHTING

A PROJECT REPORT

Submitted by

BASKARA PRABHU A
14ITR012

HARIHARAN RAJAGOPAL
14ITR031

MADHANKUMAR C
14ITR043

in partial fulfilment of the requirements

for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMMUNICATION AND COMPUTER SCIENCES

Estd : 1984

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI ERODE – 638 060

OCTOBER 2017

DEPARTMENT OF INFORMATION TECHNOLOGY

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI ERODE – 638060

OCTOBER 2017

BONAFIDE CERTIFICATE

This is to certify that the Project Report entitled **PERFORMANCE ENHANCEMENT OF KNN ALGORITHM USING 8-BIN HASHING AND FEATURE WEIGHTING** is the bonafide record of project work done by **BASKARA PRABHU A** (14ITR012), **HARIHARAN RAJAGOPAL** (14ITR031) and **MADHANKUMAR C** (14ITR043) in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Information Technology of Anna University, Chennai during the year 2017- 2018.

SUPERVISOR **HEAD OF THE DEPARTMENT**

(Signature with seal)

Date:

Submitted for the end semester viva voce examination held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DEPARTMENT OF INFORMATION TECHNOLOGY

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI ERODE - 638060

OCTOBER 2017

DECLARATION

We affirm that the Project Report titled **PERFORMANCE ENHANCEMENT OF KNN ALGORITHM USING 8-BIN HASHING AND FEATURE WEIGHTING** being submitted in partial fulfilment of the requirements for the award of Bachelor of Technology is the original work carried out by us. It has not formed the part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

BASKARA PRABHU A
14ITR012

HARIHARAN RAJAGOPAL
14ITR031

Date:

MADHANKUMAR C
14ITR043

I certify that the declaration made by the above candidates is true to the best of my knowledge.

Date:

Name and Signature of the Supervisor with seal

ABSTRACT

The K Nearest Neighbor (KNN) algorithm is an instance based learning method that has been widely used in many pattern classification tasks due to its simplicity, effectiveness and robustness. But the Standard KNN algorithm fails to work satisfactorily due to many limitations.This project dealt with improvements of KNN over two major limitations.First efficiency of KNN classification is improved by eliminating the instances which are too far away from the query instances.So,the computational time can also be improved.Second, a novel feature weighting algorithm is introduced based on positive instances in order to improve the accuracy of classification by reducing the number of features. Finally,this project gives validation of improvement theories through experimental results.

ACKNOWLEDGEMENT

First and foremost we acknowledge the abundant grace and presence of Almighty throughout different phases of the project and its successful completion.

We wish to express our extreme gratefulness to our beloved Correspondent **Thiru.A.VENKATACHALAM B.Sc,** and all the trust members of Kongu Vellalar Institute of Technology Trust for providing all the necessary facilities to complete the project successfully.

We express our deep sense of gratitude to our beloved Principal **Prof.S.KUPPUSWAMI B.E., M.Sc(Engg)., Dr.Ing (France),** for providing us an opportunity to complete the project.

We are highly indebted to express our sincere thanks to **Dr.S.VARADHAGANAPATHY M.S., M.E., Ph.D.,** Head of the Department, Department of Information Technology, for his valuable suggestions.

We are thankful to our project coordinators **Dr.C.NALINI M.E., Ph.D.,** and **Ms.S.ANITHA M.E,** for their valuable guidance and support to complete our project successfully.

We are thankful to **Mr.A.P.PONSELVAKUMAR M.E.,** Department of Information Technology for his valuable supervision and advice for the fruitful completion of the project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

KNN	K Nearest Neighbors
HR	Human Resources
IW	Initial Weight
FW	Final Weight
IDE	Integrated Development Environment
GUI	Graphical User Interface

CHAPTER 1

INTRODUCTION

1.1 DATAMINING

Data mining, also referred to as data or knowledge discovery, is the process of analyzing data and transforming it into insight that informs business decisions. Data mining refers to the systematic software analysis of groups of data in order to uncover previously unknown patterns and relationships. Data mining software enables organizations to analyze data from several sources in order to detect patterns. With the volume of data available today, organizations turn to Big Data management solutions and customer experience management solutions capable of advanced data mining for translating raw data into actionable insights.

1.2 DATAMINING TECHNIQUES

1.2.1 Association

Association is one of the best-known data mining techniques. In association, a pattern is discovered based on a relationship between items in the same transaction. That is the reason why association technique is also known as relation technique. The association technique is used in market basket analysis to identify a set of products that customers frequently purchase together.

1.2.2 Classification

Classification is a classic data mining technique based on machine learning. Basically, classification is used to classify each item in a set of data into one of a predefined set of classes or groups. Classification method makes use of mathematical techniques such as decision trees, linear programming, neural network and statistics. In classification, a software is developed to learn how to classify the data items into groups. For example, classification can be applied in the application that given all records of employees who left the company, predict who will probably leave the company in a future period. In this case, the records of employees are divided into two groups that named leave and stay.Then,the datamining software is asked to classify the employees into separate groups.

1.2.3 Clustering

Clustering is a data mining technique that makes a meaningful or useful cluster of objects which have similar characteristics using the automatic technique. The clustering technique defines the classes and puts objects in each class, while in the classification techniques, objects are assigned into predefined classes. For example in a library, there is a wide range of books on various topics available. The challenge is how to keep those books in a way that readers can take several books on a particular topic without hassle. By using the clustering technique,books are kept based on the kinds of similarities in one cluster or one shelf and label it with a meaningful name. If readers want to grab books in that topic, they would only have to go to that shelf instead of looking for the entire library.

1.2.4 Prediction

Prediction is one of the data mining techniques that discovers the relationship between independent variables and relationship between dependent and independent variables. For instance, the prediction analysis technique can be used in the sale to predict profit for the future if sale is an independent variable, profit could be a dependent variable.

Then, based on the historical sale and profit data, a fitted regression curve can be drawn that can be used for profit prediction.

1.2.5 Sequential Patterns

Sequential patterns analysis is one of the data mining techniques that seeks to discover or identify similar patterns, regular events or trends in transaction data over a business period. In sales, with historical transaction data, businesses can identify a set of items that customers buy together different times in a year. Then, businesses can use this information to recommend customers buy it with better deals based on their purchasing frequency in the past.

1.2.6 Decision Trees

Decision tree is one of the most commonly used data mining techniques because, its model is easy to understand for users. In decision tree technique, the root of the decision tree is a simple question or condition that has multiple answers. Each answer then leads to a set of questions or conditions that helps to determine the data to make the final decision.

1.3 THE K NEAREST NEIGHBOR ALGORITHM

K-nearest neighbor classifier is one of the introductory supervised classifiers, which every data science learner should be aware of. Fix and Hodges proposed K Nearest Neighbor classifier algorithm in the year of 1951 for performing pattern classification task. For simplicity, this classifier is called as KNN Classifier. To be surprised k-nearest neighbor classifier is mostly represented as KNN, even in many research papers too. KNN address the pattern recognition problems and also the best choices for addressing some of the classification related tasks.

The simple version of the K Nearest Neighbor classifier algorithm is to predict the target label by finding the nearest neighbor class. The closest class will be identified using the distance measures like Euclidean distance.

1.4 OBJECTIVE

To improve the performance efficiency and accuracy of K Nearest Neighbor classification algorithm.

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING METHOD

Pattern classification is used in the field of artificial intelligence and data mining to recognize the class of a given unknown instance. Generally, a set of training instances with their class labels are given. A pattern classification algorithm must able to predict the class of an unknown instance based on set of training instances.In traditional KNN, distance between query instance and all training instances are calculated and then voting criterian is applied to get the class of an givenn query instance.

The K Nearest Neighbor algorithm finds out k closest instances to given query instance and then carries out voting among k neighbors to get resultant output label of an unknown query instance . Algorithm assumes that all instances are located in n dimensional space where n is the number of features by which an instance is defined. Different distance metrics are used for measuring distance between instances. Most common of them are Manhattan distance and Euclidean distance. More precisely, if x and y are two instances with feature vector represented as <f1, f2…fn> then Manhattan and Euclidean distance between them are defined as d1(x, y) and d2(x, y), where,

$$d1(x,y)=\sum_{i=1}^{n} |f_i(x)-f_i(y)| \quad (2.1)$$

$$d2(x,y)=\sqrt{\sum_{i=1}^{n} | f_i(x) - f_i (y)|^2} \quad (2.2)$$

Both the above are forms of Minkowski distance d3(x, y) where

$$d3(x,y)=(\sum_{i=1}^{n} |f_i(x)-f_i(y)|^p)^{1/p} \quad (2.3)$$

where p=1 for Manhattan distance and p=2 for Euclidean distance. One of the most important parameter that has significant influence on accuracy is parameter k. k is the number of nearest neighbors of query instance that take part in the voting process. Smaller values of k may lead to inaccurate classification results whereas larger values of k make boundaries between classes less distinct .The best choice of k is still a good field for research.

2.2 STANDARD KNN ALGORITHM

Algorithm: Standard K-NN (T, q , k)

Input: Training instances T, query instance q, value of nearest neighbors k.

Output: The class label of query instance

1. For each instance t in T

    a. Find the distance d(x, y) between query instance q and training instance t.

2. End For

3. Sort these distances in ascending order keeping track of which instance is associated with each distance.

4. Return instances associated with first k distances of sorted list.

5. Use voting criterion to find out the label of query instance.

2.3 PROBLEM STATEMENT

But standard KNN fails to give desired results due to its inherent weaknesses. One of the limitations of KNN algorithm is that it takes lots of time for training if dataset is too large . During training period it has to calculate distance of query instance from all training instances and then sort them out which takes lot of computational cost which is not desirable where fast classification is a priority. Second limitation is KNN does not consider the importance of one feature over the others which can affect final accuracy of classification.

2.4 PROPOSED METHOD

1.Efficiency of classification is improved by dividing the dataset into 8 parts (bins) based on single centroid instance which serves as a key for hashing.

2.Accuracy of classification is improved by using a novel feature weighting algorithm based on positive instances. Normalization also carried out to get final weight set.

CHAPTER 3

SYSTEM REQUIREMENTS

3.1 HARDWARE REQUIREMENTS

3.1.1 Hardware System Configuration

Table 3.1 Hardware Configuration

Hardware Components	**Specifications**
Processor	Intel core i3
Speed	2.00Ghz
RAM	8GB
Hard disk	1TB
Key board	Standard windows keyboard
Mouse	Two button mouse
Monitor	SVGA

3.2 SOFTWARE REQUIREMENTS

3.2.1 Software Configuration

Table 3.2 Software Configuration

Software Components	**Specifications**
Operating System	Windows 8.1
Programming Language	R
Tool	Rstudio

3.2.2 LANGUAGE SPECIFICATION -R

R is an open source programming language and software environment for statistical computing and graphics that is supported by the R Foundation for Statistical Computing.The R language is widely used among statisticians and data miners for developing statistical software and data analysis. Polls, surveys of data miners, and studies of scholarly literature databases show that R's popularity has increased substantially in recent years.

R is a GNU package. The source code for the R software environment is written primarily in C. R is freely available under the GNU General Public License, and pre-compiled binary versions are provided for various operating systems. While R has a command line interface, there are several graphical front-ends are available.

3.2.3 TOOL SPECIFICATION-R STUDIO

RStudio is a free and open source integrated development environment (IDE) for R a programming language. RStudio was founded by JJ Allaire, creator of the programming language ColdFusion. Hadley Wickham is the Chief Scientist at R Studio. R Studio is written in C++ programming language and uses the Qt framework for its Graphical User Interface.

R Studio is available in two editions: R Studio Desktop, where the program is run locally as a regular desktop application and R Studio Server, which allows accessing R Studio using a web browser while it is running on a remote Linux server. Prepackaged distributions of R Studio Desktop are available for Windows, macOS, and Linux.

R Studio is available in open source and commercial editions and runs on the desktop (Windows, macOS, and Linux) or in a browser connected to R Studio Server or R Studio .Work on R Studio started at around December 2010, and the first public beta version (v 0.92) was officially announced in February 2011. Version 1.0 was released in November 2016.

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 8-BIN HASHING METHOD

Hashing technique is widely used to reduce the search in many computer science algorithms. Simple rule of basic hashing technique is the use of a key to carry out the search in particular bin where result is guaranteed at reduced computational cost.

In this project a novel approach used for efficiency improvement using 8-bin hashing method.In this proposed method top three features f1, f2, f3 having highest weights are selected. Which are weighted through a feature weighting algorithm mentioned in section 4.4. Now an imaginary instance is founded and named as centroid instance X with values x1, x2 and x3. Values of this instance in 3-dimensional space is mean of features f1, f2 and f3. Now consider an instance from training set whose values of 3 selected features are f1' , f2' and f3'.f1',f2' and f3' are subtracted from x1,x2 and x3 respectivly. Difference will either be positive or negative.0 is denoted for positive and 1 is for negative. By using this ,8 bins of (2^3) of training set instances are created namely 000, 001, 010, 011,100, 101, 110 and 111.

For any query instance Q with feature values q1, q2 and q3 (features selected using feature weighting algorithm) subtraction is carried out from the centroid instance X and based on differences one bin is selected out of available eight. Next,KNN algorithm is carried with reduced training set which consist of instances only from selected bin. Voting criterion is then applied on k nearest neighbors to find out target label of query instance Q.

4.2 BOUNDARY INSTANCE PROBLEM

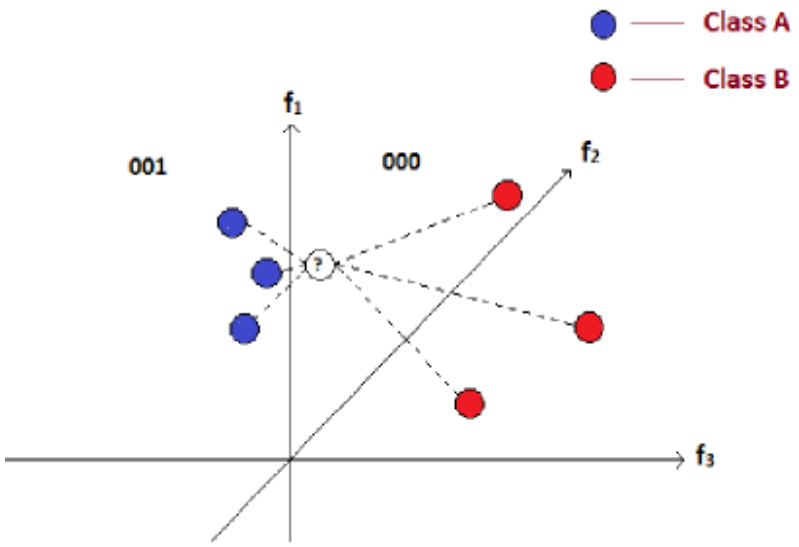

Figure 4.1 Boundary Instance Problem

The figure 4.2 shows that the query instance falls in bin 000 by using proposed 8-Bin hashing methodology. So it carries out nearest neighbor search (3-NN) on instances of bin 000.Since 3 nearest neighbors of query instance are of class B so it is given target label of B. But,the actual 3 nearest neighbors, which are more closer to query instance fall in bin 001.Since they belong to class A, actual target label of query instance through standard KNN is A. It is understood from the figure 4.2 that boundary instance problem can lead to inaccurate classification results.

4.3 EFFICIENCY IMPROVED 8-BIN HASHING METHOD

To deal with the boundary instance problem the algorithm is modified to include 8 different centroid instances for creation of 8 bin sets. First the error margins e1, e2 and e3 for features f1 , f2 and f3 are calculated. Error margin e for a particular feature f can be defined as probable distance in which k instances exist and can be calculated using formula

$$e = \left(\frac{\max(f) - \min(f)}{n} \right) \times k \tag{4.1}$$

Where n is number is number of instances in training set and k is value of nearest neighbors required.The values of 8 centroids are calculated for partitioning the dataset.To calculate the centroid of a particular bin either addition or subtractraction is performed on error margins e_1, e_2 and e_3 from values of original centroid X (x_1, x_2 , x_3) depending upon binary value of bin. For a bin with binary value b_1,b_2,$b_3$and centroid for that bin be O with values o_1, o_2 and o_3, which are calculated using formula

$$o_i = x_i - e_i \quad \text{if} \quad b_i=0 \tag{4.2}$$

$$o_i = x_i + e_i \quad \text{if} \quad b_i=1 \tag{4.3}$$

For each centroid o_i of bin b_i single pass is carried out through entire training dataset including those instances in bin b_i which satisfy the required condition. For example in the figure 4.2 for bin 000 all differences should be positive using bin-specific centroid o_1 rather than original centroid X. So total there will be 8 passes through training dataset, one for each bin. Any instance that satisfies the condition of one or more bins exists in all of those bins. For query instance Q ,the same condition is performed to check with the original centroid X to select 1 out of 8 bins to carry out nearest neighbor search.

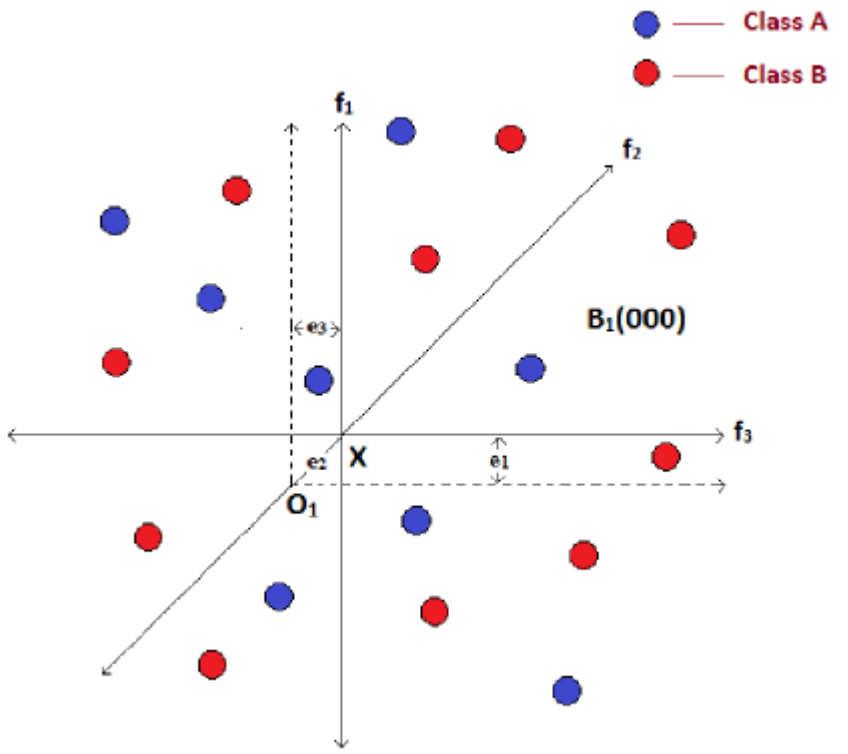

Figure 4.2 Efficiency Improved 8- bin Hashing

4.4 FEATURE WEIGHTING AND NORMALIZATION

The main goal of this proposed feature weighting algorithm is to give stable and accurate classification results. Many times irrelevant features tend to degrade the classification performance. In most of the feature weighting algorithms relevant features are given higher weights compared to irrelevant features which are given lower weights. For a particular feature a positive instance is an instance which is surrounded by instances of its own class. First the dataset is divided into f subparts where f is the number of features. Numeric features have more impact on classification than categorical, so numeric features are weighted higher. All categorical features are given weight of 1 and numeric features are weighted more than 1. For each numeric feature subpart f′ K Nearest Neighbor algorithm is carried out on each instance of training set. If class returned by K Nearest Neighbor matches with the actual class of the instance then that instance is regarded as a positive instance, and if not it is called

negative instance. For feature f whose values vary from x_1 to x_2 and k=7 positive and negative instances are depicted below:

Figure 4.3 Positive and NegativeInstances

The number of positive instances are calculated for each numeric feature of the training set. Next initial weighting of features is carried out. Numeric features are weighted from 2 to (f′+1) where f′ is number of numeric features. Feature with least number of positive instances is weighted 2 and feature with highest number of positive instances will be given initial weight f′. If two features are having same number of positive instances then both of them are given same weights based on which highest weight will be reduced. Basic theory behind this idea is that the feature that brings an instance closer to the members of its own class is a better feature and is weighted higher.

4.5 NEW NORMALIZATION METHOD

Data or Feature Normalization is used in many classifiers to normalize the features before classification . Since data values for different features may vary widely, some machine learning algorithms will not perform accurately. In feature normalization all features are normalized so that their values fall in a certain range. This is carried out such that features with higher ranges do not have negative impact on classification accuracy. In this proposed

algorithm a new normalization formula is introduced in which normalization is carried out by weighting the features rather than more direct approach of first normalizing the feature in range 0 to1. For feature fi with initial weight of IW(fi), final weight FW(fi) is calculated as given in formula

$$FW(f_i)=\frac{IW(f_i)-\text{Relative Mean}}{\text{Mean}(f_i)} \tag{4.4}$$

Where Relative Mean is the highest mean among all features and Mean(fi) is the mean of feature f_i.

4.6 FEATURE WEIGHTING ALGORITHM

Algorithm: Feature Weighting Algorithm (T, m[ ], r, k)
Input: Training instances T, mean set of features m[ ], Highest mean r, value of nearest neighbors k
Output: weight set w [ ]

1. For each feature f in training set T:
    a. If f is categorical:
        W[f] =1
    b. else
        I. for each instance i:
            a. class=StandardK-NN(T,i,k)
            b. If class=label[i]
                increment PositiveCount[f]
        II. End for
2. End For
3. Sort Numeric features based on Positive Count[f]
4. IW[f] is equal to sorted index +1
5. For each Numeric feature f in training set T:

a. W[f] = (IW[f]*r)/m[f]

6. .End For

4.7 TIME COMPLEXITY IMPROVEMENT

Next time complexity analysis is carried for our proposed algorithm. For query instance Q, training dataset T with n instances and each instance having f number of features, it takes O(f*n) to calculate distances between q and all other instances and O(n*log(n)) to sort distances using quick sort. So total time complexity of standard KNN algorithm for each new query instance is O(f*n + n*log(n)).In this proposed algorithm if it is assumed that all instances are uniformly distributed in three dimensional space, approximate number of instances in single bin n′ will be

$$n' \approx \left(\frac{n}{8}\right) + \alpha \tag{4.5}$$

where α is number of instances that are included into each bin by using error margins. So time complexity of this proposed algorithm will be O(f*n′+n′*log(n′)). Since value of n′ is smaller than n, proposed algorithm is more efficient than standard KNN.

4.8 RESULTS

Standard and proposed methodologies are applied on three different datasets. Details of all the datasets are shown in Table 4.1

Table 4.1 Datasets

Dataset name	**No. of. instances**	**No. of. classes**	**No .of .features**
Spine dataset	310	2	12
Iris dataset	151	3	4
HR dataset	2000	2	5

4.8.1 Spine Dataset Results

In table 4.2 and figure 4.4 experimental results for spine dataset are shown. The dataset consists of 310 instances out of which 217 are used for training and 93 for testing. There are in total 12 features and two classes.

Table 4.2 Spine Dataset Results

K-Value	Accuracy	
	Standard KNN	Improved KNN
1	0.767	0.79
3	0.757	0.811
5	0.769	0.83
7	0.794	0.834
11	0.8	0.87

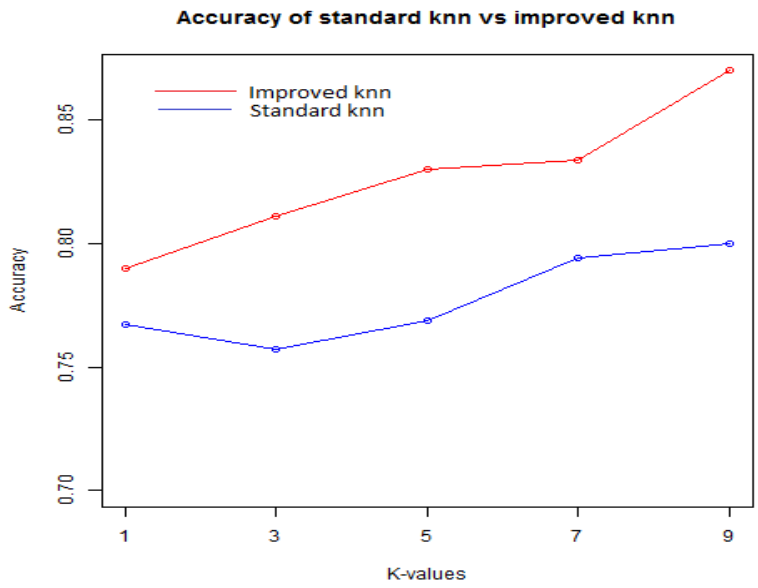

Figure 4.4 Accuracy Comparison for Spine Dataset

4.8.2 HR Dataset Results

In table 4.3 and figure 4.5 experimental results for HR dataset are shown. The data set consists of 2000 instances out of which 1400 are used for training and 600 for testing. There are in total 5 features and two classes.

Table 4.3 HR Dataset Results

K-Value	Accuracy	
	Standard KNN	Improved KNN
1	0.69	0.79
3	0.752	0.80
5	0.73	0.81
7	0.80	0.83
11	0.74	0.82

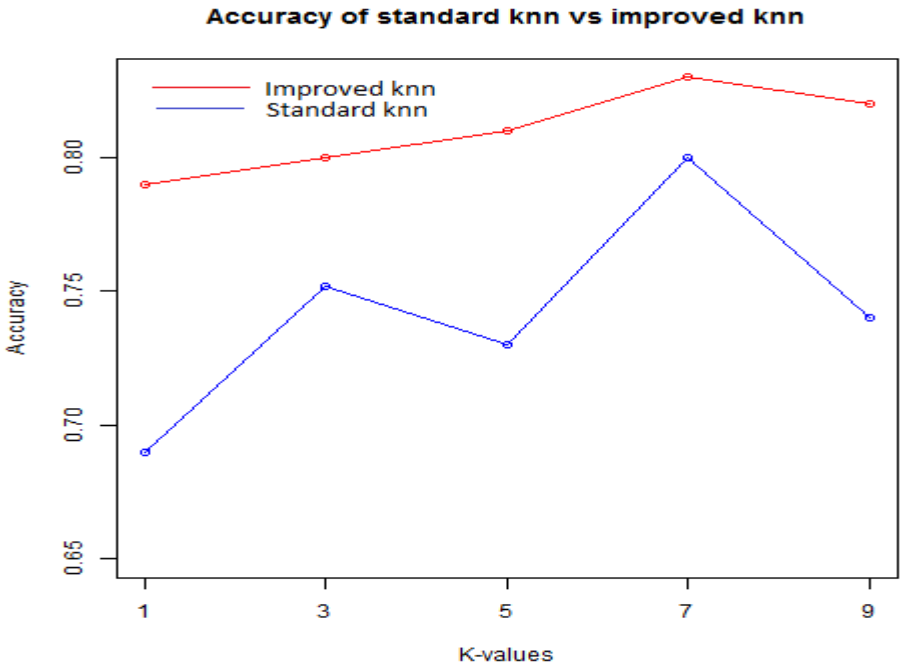

Figure 4.5 Accuracy Comparison for HR Dataset

4.8.3 Iris Dataset Results

In table 4.4 and figure 4.6 experimental results for iris dataset are shown. The dataset consists 151 instances out of which 106 are used for training and 45 for testing. There are in total 4 features and three classes.

Table 4.4 Iris Dataset Results

K-Value	Accuracy	
	Standard KNN	Improved KNN
1	0.69	0.76
4	0.72	0.767
7	0.71	0.75
10	0.75	0.74
13	0.74	0.79

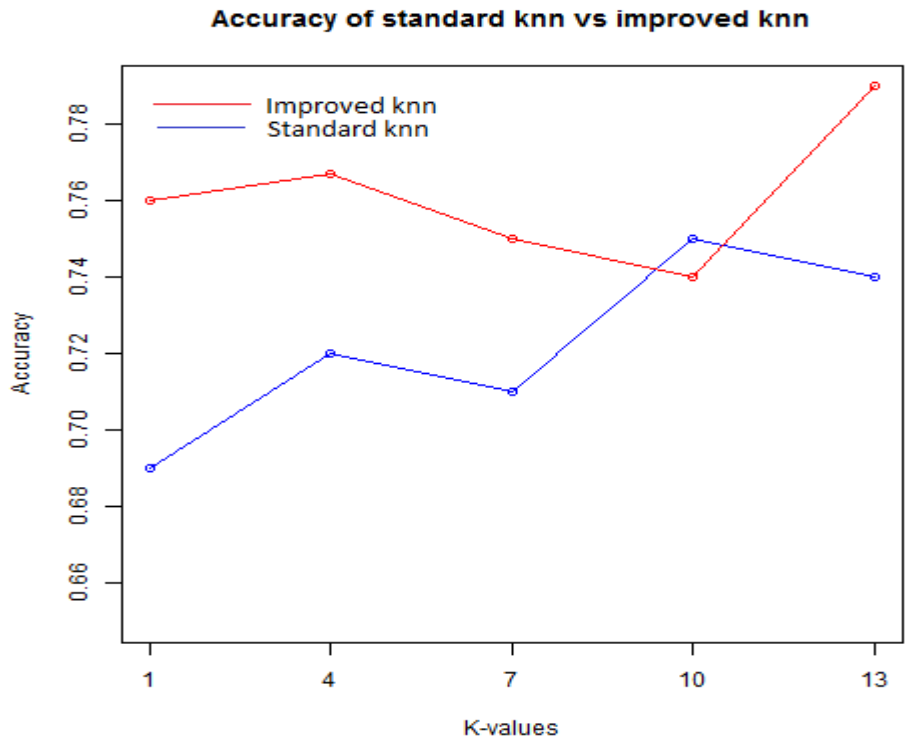

Figure 4.6 Accuracy Comparison for Iris Dataset

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

5.1 CONCLUSION

The K Nearest Neighbor algorithm is an instance based learning method that is widely used in pattern classification. Based on limitations of the standard K Nearest Neighbor algorithm, this project proposes novel improvements for the following two aspects. First, efficiency of standard classifier is improved by dividing the entire dataset into 8 bins and then using hash key to search only one bin for new query instance. Second, accuracy is improved by weighting each feature based on number of positive instances. Normalization of features also carried out .So, higher valued features do not degrade performance of the classifier.

5.2 FUTURE SCOPE

The experimental results shows that the improved K NN has following merits: a remarkable decrease in computational time and a stable increase in classification accuracy. The research can be continued by taking hyperplanes into consideration which will further reduce the computational cost. Also, more effective weighting algorithm is needed to weight categorical features.

APPENDIX -1

CODING

```
setwd("E:/KNN-improved")
library(class)
#Function for finding the bin number
findno<-function(o,xcc)
{
        ff1=o[1]-xcc[1,1]
        ff2=o[2]-xcc[1,2]
        ff3=o[3]-xcc[1,3]
        binnum=0
        if(ff1>0)
        binnum=binnum+4
        if(ff2>0)
        binnum=binnum+2
        if(ff3>0)
        binnum=binnum+1
        binnum=binnum+1
```

```
        return(binnum)
}
#method for the  improved algorithm
myfunction<-function(b,c)
{
        #geting 30% of indexes randomly
        indexes = sample(1:nrow(b), size=0.3*nrow(b))
        # Split training data
        test = b[indexes,]
        # Split testing data
        train = b[-indexes,]
        columns<-names(b)
        nocol=ncol(b)
        cvec<-train[,nocol]
        kfff<-knn(train,test,cvec,c)
        acc1=0
        for(i in 1:nrow(test))
        {
                integer(kfff[i])
                if(kfff[i]==test[i,nocol])
        {
```

```
acc1=acc1+1
}
acc1=acc1/nrow(test)
zo=nocol-1
countr=integer(zo)
meann=integer(zo)
for(i in 1:zo)
{
        x<-data.frame(train[,i])
        x$c2=train[,nocol]
        names(x)[1]="c1"
        kf1<-knn(x,x,x$c2,c)
        for(j in 1:nrow(x))
        {
                integer(kf1[j])
                if(kf1[j]==x[j,2])
                {
                        countr[i]=countr[i]+1
                }
        }
```

```
        meann[i]=mean(x$c1)
}
meanss=c(meann)
maxmi=max(meann)
counters=data.frame(names(train))
counters=data.frame(counters[1:zo,])
countr=c(countr)
counters$b=countr
counters$c=meanss
counters$d=c(1:zo)
names(counters)[1:4]=c("column name","Positive count","Mean ","Column number")
counterss=counters[order(counters$'Positive count'),]
counterss$iw=c(1:zo)
counterss$fw=counterss$iw*maxmi/counterss$Mean
countersss=counterss[order(counterss$fw,decreasing=TRUE),]
fno=integer(3)
fno[1]=countersss[1,4]
fno[2]=countersss[2,4]
fno[3]=countersss[3,4]
x=data.frame(train[,fno[1]])
x$c2=train[,fno[2]]
```

```
x$c3=train[,fno[3]]

x$c4=train[,nocol]

names(x)[1]<-c("c1")

y=data.frame(test[,fno[1]])

y$c2=test[,fno[2]]

y$c3=test[,fno[3]]

y$c4=test[,nocol]

names(y)[1]<-c("c1")

ran=sample(1:nrow(x),1)

c1=x[ran,1]

c2=x[ran,2]

c3=x[ran,3]

oo=matrix(double(24),nrow=8,ncol=3,byrow = TRUE)

e<-double(3)

rr=max(x$c1)-min(x$c1)

rr=rr*c

rr=rr/nrow(x)

#finding error margins

e[1]=(max(x$c1)-min(x$c1))*c/nrow(x)

e[2]=(max(x$c2)-min(x$c2))*c/nrow(x)

e[3]=(max(x$c3)-min(x$c3))*c/nrow(x)
```

```
# calculating centroid for each bin

oo[1,1]=c1-e[1]

oo[1,2]=c2-e[2]

oo[1,3]=c3-e[3]


oo[2,1]=c1-e[1]

oo[2,2]=c2-e[2]

oo[2,3]=c3+e[3]


oo[3,1]=c1-e[1]

oo[3,2]=c2+e[2]

oo[3,3]=c3-e[3]


oo[4,1]=c1-e[1]

oo[4,2]=c2+e[2]

oo[4,3]=c3+e[3]


oo[5,1]=c1+e[1]

oo[5,2]=c2-e[2]

oo[5,3]=c3-e[3]
```

```
oo[6,1]=c1+e[1]

oo[6,2]=c2-e[2]

oo[6,3]=c3+e[3]


oo[7,1]=c1+e[1]

oo[7,2]=c2+e[2]

oo[7,3]=c3-e[3]


oo[8,1]=c1+e[1]

oo[8,2]=c2+e[2]

oo[8,3]=c3+e[3]


#initialize bins

binn<-list()

binn[[1]]<-0

binn[[2]]<-0

binn[[3]]<-0

binn[[4]]<-0

binn[[5]]<-0

binn[[6]]<-0

binn[[7]]<-0
```

```
binn[[8]]<-0
#8 bin hashing
for(i in 1:8)
{
	binn[[i]]<-data.frame()
	tempp<-data.frame()
	rowss=1
	for(j in 1:nrow(x))
	{
		binnoo<-findno(oo[i,],x[j,])
		x[j,5]=binnoo
		names(x)[5]<-"binn"
	}
	binn[[i]]=subset(x,x$binn==i)
	binn[[i]]=binn[[i]][,1:4]
	x=x[,1:4]
}
acc=0
for(i in 1:nrow(y))
{
	ff1=c1-y[i,1]
```

```
ff2=c2-y[i,2]

ff3=c3-y[i,3]


binnum=0

if(ff1>0)

        binnum=binnum+4

if(ff2>0)

        binnum=binnum+2

if(ff3>0)

        binnum=binnum+1

binnum=binnum+1


kf1<-knn(binn[[binnum]],y,binn[[binnum]]$c4,k=c)

resi=y[i,4]

integer(resi)

as.integer(kf1)

zv=(kf1[2]==resi)

if(zv)

{

        acc=acc+1

}
```

```
        }
        acc=acc/nrow(y)
        out<-c(acc1,acc)
        return out
}
#read dataset in a dataframe
dataset <- read.csv("lower.csv")
#renaming columns
names(dataset)[1:13]<-c('Pelvic Incidence','Pelvic Tilt','Lumbar Lordosis Angle'
,'Sacral Slope','Pelvic Radius', 'Spondylolisthesis Degree',
'Pelvic Slope', 'Direct Tilt', 'Thoracic Slope', 'Cervical Tilt',
'Sacrum Angle', 'Scoliosis Slope','Outcome')
#removing column X
dataset$X<- NULL
#set class labels normal and abnormal  to 1 and 0
levels(dataset$Outcome)[1]<-0
levels(dataset$Outcome)[2]<-1
a1<-myfunction(dataset,1)
dataset2 <- read.csv("hr.csv")
indeces = sample(1:nrow(dataset2), size=2000)
dataset2<-dataset2[indeces,]
```

```
a2<-myfunction(dataset2,1)

#read dataset in a dataframe

dataset3 <- read.csv("iriss.csv")

levels(dataset3$Species)[1]<-1

levels(dataset3$Species)[2]<-2

levels(dataset3$Species)[3]<-3

a3<-myfunction(dataset3,1)

print("Backpain dataset :")

print("Improved knn:")

print("Accuracy:")

print(a1[1])

print("Standard knn:")

print("Accuracy:")

print(a1[2])

print("Human Resources dataset :")

print("Improved knn:")

print("Accuracy:")

print(a2[1])

print("Standard knn:")

print("Accuracy:")

print(a2[2])
```

```
print("Iriss dataset :")

print("Improved knn:")

print("Accuracy:")

print(a3[1])

print("Standard knn:")

print("Accuracy:")

print(a3[2])
```

APPENDIX- 2

SNAPSHOTS

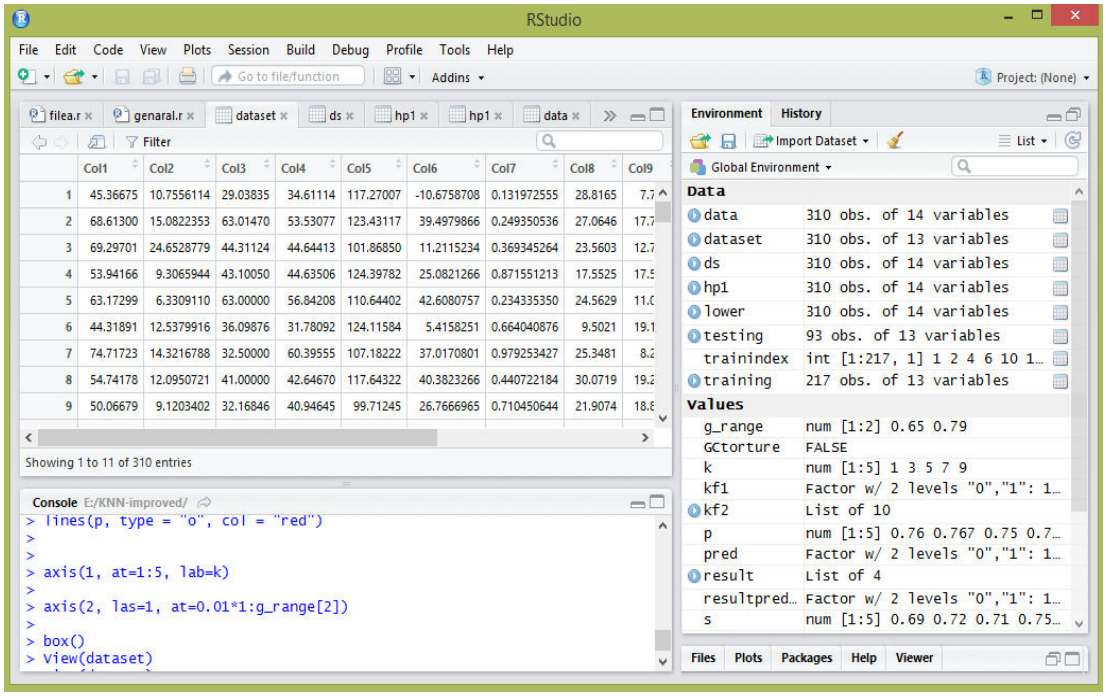

Figure A2.1 View Dataframe in R Studio

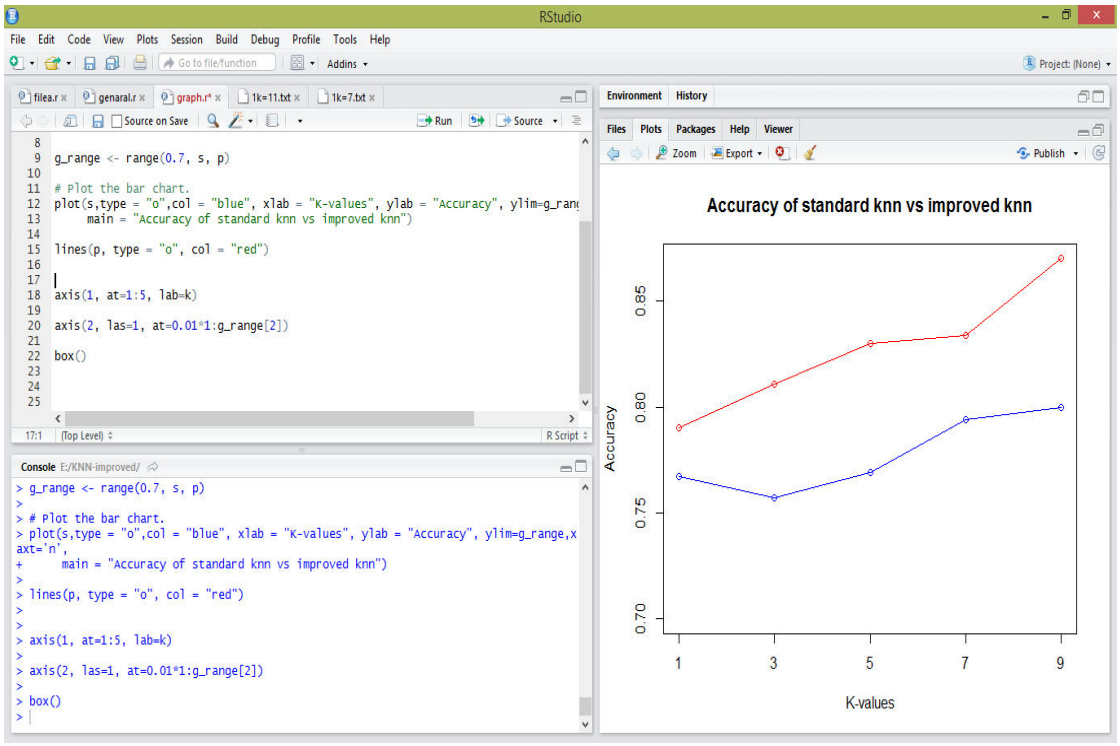

Figure A2.2 Plot Graph in R Studio

Console E:/KNN-improved/

```
+        if(ff1>0)
+          binnum=binnum+4
+        if(ff2>0)
+          binnum=binnum+2
+        if(ff3>0)
+          binnum=binnum+1
+        binnum=binnum+1
+
+        kf1<-knn(binn[[binnum]],y,binn[[binnum]]$c4,k=c)
+        resi=y[i,4]
+
+        integer(resi)
+        as.integer(kf1)
+
+        zv=(kf1[2]==resi)
+       if(zv)
+       {
+         acc=acc+1
+       }
+
+
+
+
+
+     }
+     acc=acc/nrow(y)
+
+
+     out<-c(acc1,acc,time.taken1,time.taken2)
+
+
```

Figure A2.3 R Studio Console

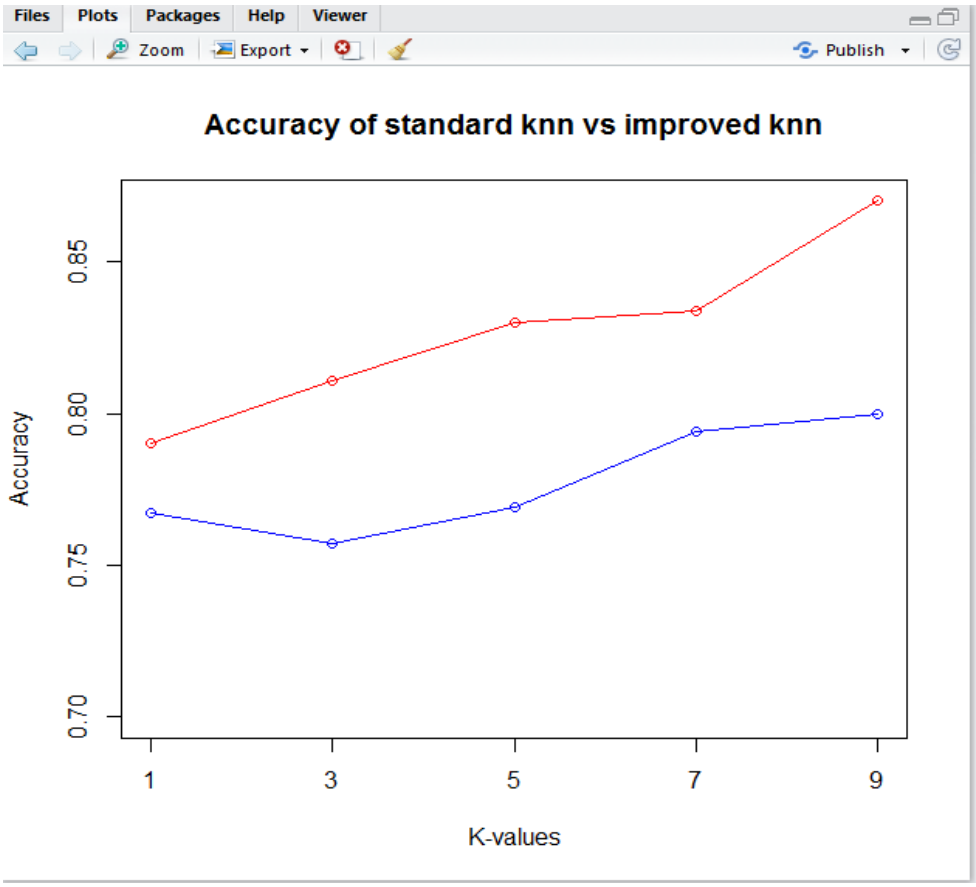

Figure A2.4 Graph generated from R Studio

REFERENCES

1. Akhil Rane , Dr. J.A. Laxminarayana and Nitesh Naik (2014),"Performance Enhancement of K Nearest Neighbor Classification Algorithm Using 8-Bin Hashing and Feature Weighting", International Conference on Interdisciplinary Advances in Applied Computing", Coimbatore.

2. Jiawei Han, Micheliene Kamber and Jian Pei ,"Datamining concepts and techniques",3rd edition, Elsevier, 2012.

3. https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/

4. https://ragrawal.wordpress.com/2012/01/14/dividing-data-into-training-and-testing-dataset- in-r/