**K NEAREST NEIGHBOUR ESTIMATOR**

Submitted in partial fulfillment of the requirements for the award of the degree of

***Bachelor of Engineering***

***(Computer Science and Engineering)***

**Submitted to**

**M.S Ramaiah Institute of Technology , Bangalore**

**Submitted by**

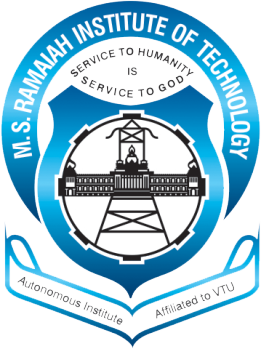
**Aman Mehta (1MS13CS021)**

**Ayush Kumar (1MS13CS034)**

**Bineet Kumar (1MS13CS036)**

**Supervised by:-**

**Prof. Srinidhi H**



**M.S Ramaiah Institute Of Technology ,**

**Autonomous Institute (Affiliated under VTU)**

**Contents**

* Certificate
* Acknowledgement
* Introduction
* Literature
* Code
* Screenshots
* Reference

**Certificate**

I hereby certify that the work that is being presented in B.E mini project report entitled ***"K nearest neighbor estimator"*** , in partial fulfillment of the requirements for the award of ***Bachelor of Engineering in Computer Science and Engineering*** and submitted to the department of CS of MSRIT , an authentic record of our own work carried out during the period of Dec-9 to Dec-11 (5 sem) , Under the supervision of Srinidhi H , Astt Prof CSE Dept .

**Signature of Students**

Aman Mehta (1MS13CS021)

Ayush Kumar (1MS13CS034)

Bineet Kumar (1MS13CS036)

This is to certify that the above statements made by the student's is correct to best of my knowledge.

**Signature of Supervisor**

**Acknowledgement**

Being totally unware about the subject and then completing a mini project on that wouldn't be possible without guidance.

We express our sincere gratitude to professor Srinidhi Sir , Dept of CSE of MSRIT , for his guidance , continuous encouragement and supervision throughout the course of the present work .

***Signature of the Students***

Aman Mehta (1MS13CS021)

Ayush Kumar (1MS13CS034)

Bineet Kumar (1MS13CS036)

**Introduction**

KNN is an “*non parametric lazy learning”*algorithm. That is a pretty concise statement. When you say a technique is non parametric , it means that it does not make any assumptions on the underlying data distribution. This is pretty useful , as in the real world , most of the practical data does not obey the typical theoretical assumptions made (eg gaussian mixtures, linearly separable etc) . Non parametric algorithms like KNN come to the rescue here.

It is also a lazy algorithm. What this means is that it does not use the training data points to do any *generalization*. In other words, there is *no explicit training phase*or it is very minimal. This means the training phase is pretty fast . Lack of generalization means that KNN keeps all the training data. More exactly, all the training data is needed during the testing phase. (Well this is an exaggeration, but not far from truth). This is in contrast to other techniques like SVM where you can discard all non support vectors without any problem. Most of the lazy algorithms – especially KNN – makes decision based on the entire training data set (in the best case a subset of them).

**Literature Review**

We use “R” for finding the accuracy of comparing the training set with the data set and displaying the result.

The data set we use is **Waveform and Digit Recognition** data set.

The **Waveform** data set has **21** fields,which we use to find the minimum distance of a test data set.

The **Waveform** data set has **41** fields,which we use to find the minimum distance of a test data set.

We Applied the above mentioned algorithm on 2 different data-sets

The **Digit Recognition** data set has **785** fields,which we use to find the minimum distance of a test data set.

**CODE**

1)Waveform without noise R scipt :

data<- read.csv("waveform.csv",header=FALSE)

normalize <- function(x)

(

return((x-min(x))/(max(x)-min(x)))

)

data\_feature<-data[,1:21]

data\_n <- as.data.frame(lapply(data\_feature[,1:21],normalize))

train\_data <- data\_n[1:4499,]

test\_data <-data\_n[4500:5000,]

train\_data\_class <- data[1:4499,22]

test\_data\_class <- data[4500:5000,22]

require(class)

error<- NULL

#for(j in 1:200)

#{

m1 <- knn(train=train\_data,test=test\_data,cl=train\_data\_class,k=71)

x<-table(test\_data\_class,m1)

sum\_diag <- sum(diag(x))

sum<-sum(x)

error <- c(error,1 - sum\_diag/sum)

#}

#cat("minimum error at k=",which.min(error)," and the minimum value is : ",min(error))

#y<-(1:length(error))

#plot(y,error)

2)Waveform with noise R scipt :

data<- read.csv("waveform\_with\_noise.csv",header=FALSE)

normalize <- function(x)

(

return((x-min(x))/(max(x)-min(x)))

)

data\_feature<-data[,1:40]

data\_n <- as.data.frame(lapply(data\_feature[,1:40],normalize))

train\_data <- data\_n[1:4499,]

test\_data <-data\_n[4500:5000,]

train\_data\_class <- data[1:4499,41]

test\_data\_class <- data[4500:5000,41]

require(class)

library(class)

error<-NULL

m2 <- knn(train=train\_data,test=test\_data,cl=train\_data\_class,k=71)

x<-table(test\_data\_class,m2)

sum\_diag <- sum(diag(x))

sum<-sum(x)

error <- c(error,1 - (sum\_diag/sum))

cat("Error : ",error\*100,"%")

3) Digit recognition R script :

data<- read.csv("train\_digit.csv",header=TRUE)

data\_feature<-data[,2:785]

data\_n<-data\_feature

train\_data <- data\_n[1:1100,]

test\_data <-data\_n[1101:1199,]

train\_data\_class <- data[1:1100,1]

test\_data\_class <- data[1101:1199,1]

require(class)

m2 <- knn(train\_data,test\_data,train\_data\_class,k=18)

x<-table(test\_data\_class,m2)

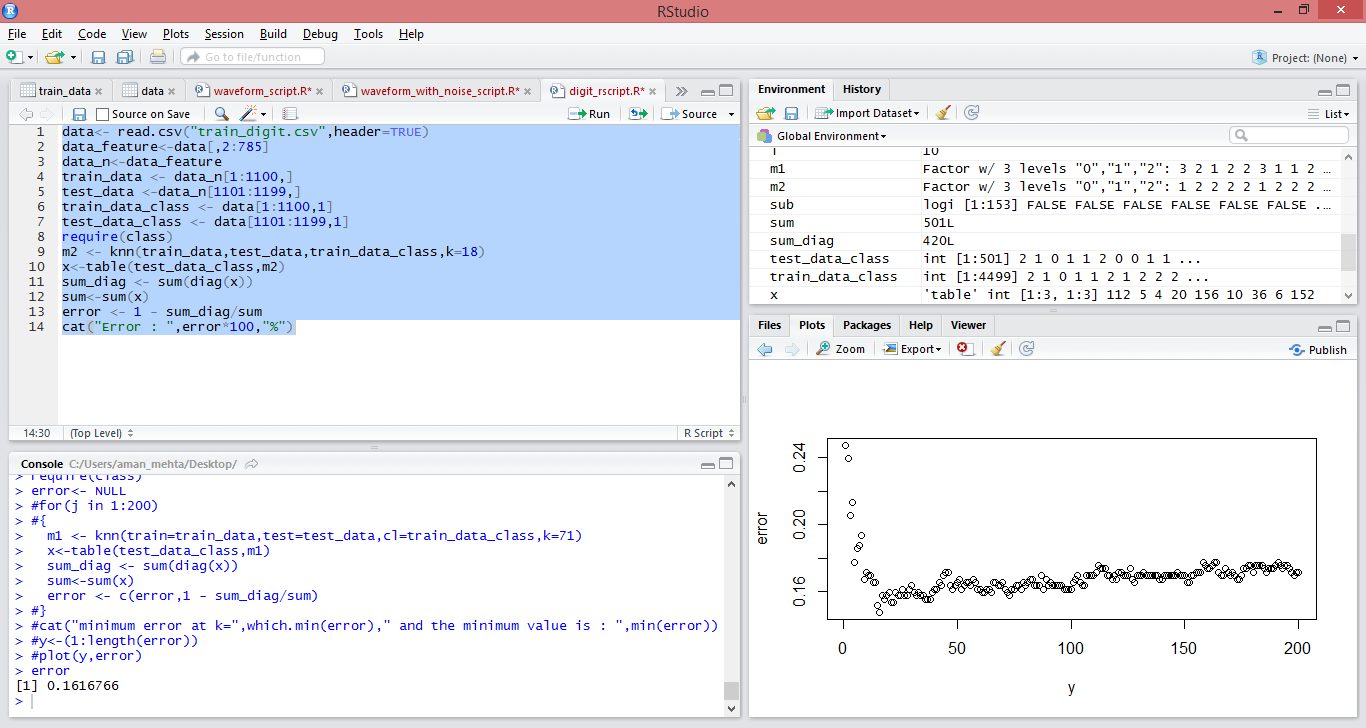
sum\_diag <- sum(diag(x))

sum<-sum(x)

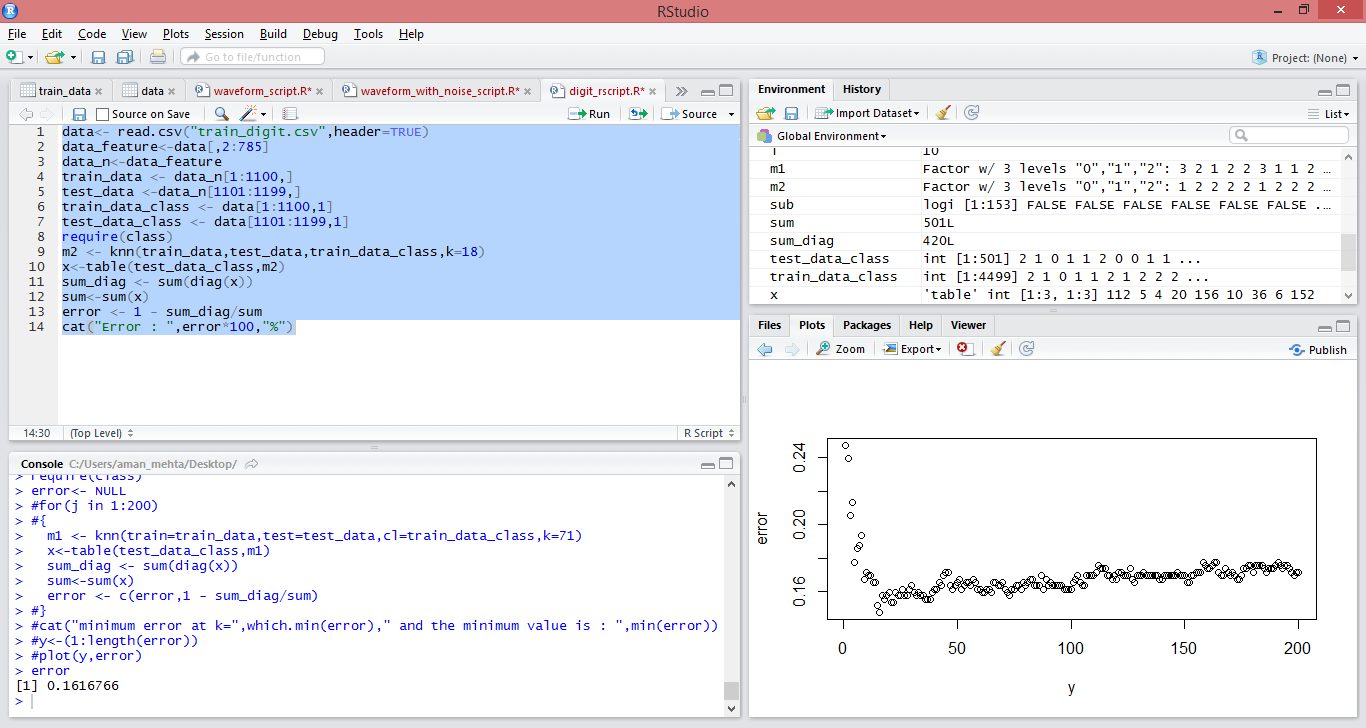
error <- 1 - sum\_diag/sum

cat("Error : ",error\*100,"%")

**Screen - Shots**

****

**Comparing error values for different values of k in knn algorithm for waveform data set without noise**



**References**

* Stackoverflow
* Thales Seign koting (Youtube Channel)