Baruch College, STA-CIS 3920, Exercise#5 Anil Poonai 37

23 July 2020

**Warm-Up**

Code is in the appendix.

This shows where knn was predicting the classification for cat and dog.

ProbeKnn(TrainX, ProbeX, TrainY, 1)

A close up of a screen

Description automatically generated

Had to standardize is so that they would both have the same range.

**A close up of a logo

Description automatically generated** NewProbeKnn( StTrainX, StProbeX, TrainY, 1 )

Same thing just with 3 clusters now.

**A picture containing game

Description automatically generated**NewProbeKnn( StTrainX, StProbeX, TrainY, 3 )

**A close up of a logo

Description automatically generated**NewProbeKnnBig(TrainStLnX.IBM,StProbeX,TrainY.IBM,25)

Used the log transformation of the data now.

NewProbeKnnBig(TrainStLnX.IBM,StProbeX,TrainY.IBM,5)

5 clusters now.

**A picture containing pencil

Description automatically generated**

plot(junk,type="l",ylab="Rate",xlab="k")

**A close up of a map

Description automatically generated**This shows the correct classification rate foe each k value on the y axis.

plot(junk,type="l",ylab="Rate",xlab="k")

We used cross validation here in order to make it more random. This gives the possibility of the best choice happening more.

**A screenshot of a cell phone

Description automatically generated**

**5.2**

I ran into a problem here when scaling the log ranges. So, the images below are the logarithms but I then normalized the data between 0 and 1 and they ended up being the exact same so I’m fairly certain the scaling wouldn’t have done much. For some reason when it got to the scaling part, the values ended up all becoming nan’s. The graphs below are pretty integrated into each other, so having them is not too practical. The mistakes are not easily identifiable here and I am not entirely sure if it is the functions mistake or the lagged values part. I would assume the values as excel did not seem to process them clearly as shown in your lecture earlier.

A close up of a map

Description automatically generated

A close up of text on a white background

Description automatically generated

**A close up of a map

Description automatically generated5.2**

The graph below shows the percentage that correctly classifies each data point at the k values in the x axis.

A close up of a map

Description automatically generated**5.3**

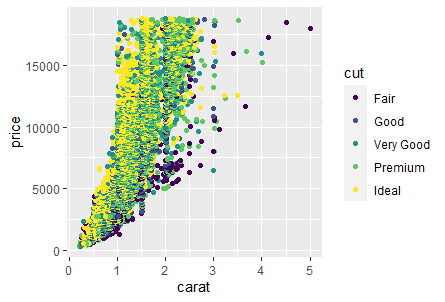
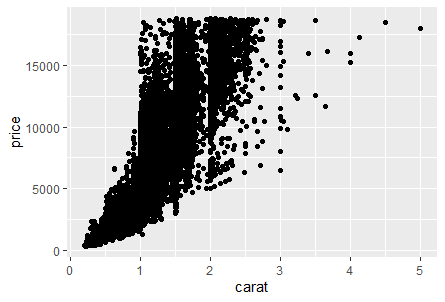
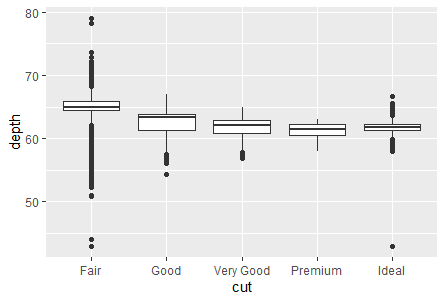
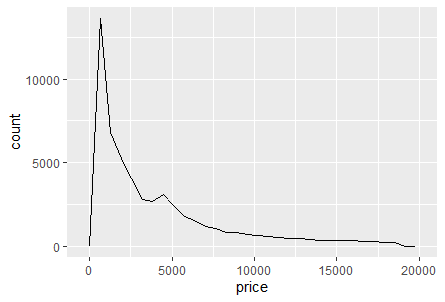
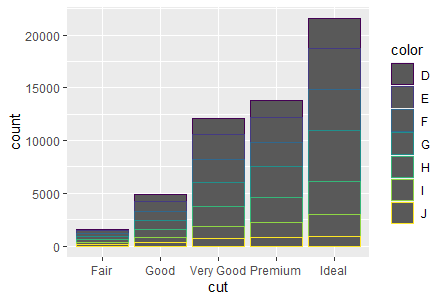
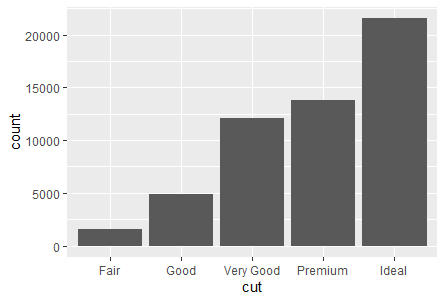
The graph below shows the same thing as the previous graph but with less random variation as this one incorporates taking the average of the correct classification rates that happen in each k.

**A picture containing flying, boat, kite, water

Description automatically generated5.5**

**A.**

So for this part, I’m going to link mt github: <https://github.com/DevonARP>. As I have two projects there that involve different ways to incorporate graphs. I also use ggplot from the Tidyverse as a more efficient and productive way to graph. I’m also going to add an old homework assignment I had that I thought was very elaborate in data visualization. Code is below the exercises section in the appendix.



The graphs will make more sense when looking at the code. It’s analyzing diamond characteristics.

**Appendix**

**Warm-Up**

R version 4.0.0 (2020-04-24) -- "Arbor Day"

Copyright (C) 2020 The R Foundation for Statistical Computing

Platform: x86\_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.

You are welcome to redistribute it under certain conditions.

Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.

Type 'contributors()' for more information and

'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or

'help.start()' for an HTML browser interface to help.

Type 'q()' to quit R.

[Previously saved workspace restored]

> ProbeKnn(TrainX,ProbeX,TrainY,1)

NULL

> StTrainX=apply(TrainX,2,scale)

> StProbe=apply(ProbeX,2,scale)

> NewProbeKnn(StTrainX,StProbeX,TrainY,3)

NULL

> StProbeX=apply(ProbeX,2,scale)

> NewProbeKnn(StTrainX,StProbeX,TrainY,3)

NULL

> StProbeX=apply(ProbeX,3,scale)

Error in apply(ProbeX, 3, scale) : 'MARGIN' does not match dim(X)

> NewProbeKnn(StTrainX,StProbeX,TrainY,1)

NULL

> StProbeX=apply(ProbeX,1,scale)

> NewProbeKnn(StTrainX,StProbeX,TrainY,1)

Error in knn(TrialX, ProbeX, TrialY, k) : no missing values are allowed

> StProbeX=apply(ProbeX,2,scale)

> NewProbeKnn(StTrainX,StProbeX,TrainY,1)

NULL

> NewProbeKnn(StTrainX,StProbeX,TrainY,3)

NULL

> LnX.IBM=log(X.IBM)

> StLnX.IBM=apply(LnX.IBM,2,scale)

> TrainStLnX.IBM=StLnx.IBM[InSample,]

Error: object 'StLnx.IBM' not found

> TrainStLnX.IBM=StLnX.IBM[InSample,]

> TestStLnX.IBM=StLnX.IBM[OutSample,]

> NewProbeKnnBig(TrainStLnX.IBM,StProbeX,TrainY.IBM,25)

NULL

> NewProbeKnnBig(TrainStLnX.IBM,StProbeX,TrainY.IBM,5)

NULL

> junk = kSearch(X.IBM,Y.IBM,1200,50)

Error in kSearch(X.IBM, Y.IBM, 1200, 50) :

could not find function "kSearch"

> kSearch =

+ function (X,Y,split=100, d=25)

+ {

+ if(class(Y)!= "factor") stop('Y is not factor type')

+ if(class(X)!= "data.frame") stop('X is not data.frame type')

+ rows = nrow(X)

+ Hold = rep(NA,d)

+ Shuffle=sample(rows,rows)

+ for(k in 1:d) {

+ InSample=Shuffle[1:split]

+ OutSample=Shuffle[(split+1):rows]

+ TrainX = X[InSample,]

+ TrainY = Y[InSample]

+ TestX = X[OutSample,]

+ TestY = Y[OutSample]

+ knn.pred = knn(TrainX, TestX, TrainY, k)

+ table.out = table(knn.pred,TestY)

+ Hold[k] = (table.out[1,1]+table.out[2,2])/sum(table.out)

+ }

+ return(Hold)

+ }

> junk = kSearch(X.IBM,Y.IBM,1200,50)

> plot(junk,type="l",ylab="Rate",xlab="k")

> kcvSearch =

+ function (X,Y,split=100, d=25, m=10)

+ {

+ if(class(Y)!= "factor") stop('Y is not factor type')

+ if(class(X)!= "data.frame") stop('X is not data.frame type')

+ rows = nrow(X)

+ Hold = rep(NA,d)

+

+ for(k in 1:d) {

+ Store = rep(NA, m)

+

+ for(j in 1:m) {

+ Shuffle=sample(rows,rows)

+ InSample=Shuffle[1:split]

+ OutSample=Shuffle[(split+1):rows]

+ TrainX = X[InSample,]

+ TrainY = Y[InSample]

+ TestX = X[OutSample,]

+ TestY = Y[OutSample]

+ knn.pred = knn(TrainX, TestX, TrainY, k)

+ table.out = table(knn.pred,TestY)

+ Store[j] = (table.out[1,1]+table.out[2,2])/sum(table.out)

+ }

+ Hold[k] = mean(Store)

+ }

+ return(Hold)

+ }

> junk = kcvSearch(X.IBM,Y.IBM,1200,50,30)

> plot(junk,type="l",ylab="Rate",xlab="k")

**Exercises**

R version 4.0.0 (2020-04-24) -- "Arbor Day"

Copyright (C) 2020 The R Foundation for Statistical Computing

Platform: x86\_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.

You are welcome to redistribute it under certain conditions.

Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.

Type 'contributors()' for more information and

'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or

'help.start()' for an HTML browser interface to help.

Type 'q()' to quit R.

[Previously saved workspace restored]

> setwd("C:\Users\poona\Desktop\School")

Error: '\U' used without hex digits in character string starting ""C:\U"

> setwd("C:/Users/poona/Desktop/School")

> AOS.CSV=read.csv("AOS.CSV")

> tail(AOS.CSV

+ )

Date Open High Low Close Adj.Close Volume LGFet Lag1 Lag2 Lag3 Lag4 Lag5 Daily.Range Direction RangeL1 RangeL2

3646 7/7/2020 47.57 48.30 47.54 47.90 47.90 962700 -0.2914217 1.5215575 2.2030260 -1.7402377 1.4642550 4.4769358 0.759998 DOWN 0.6814685 3.9432637

3647 7/8/2020 48.05 48.22 47.51 47.84 47.84 874200 -0.1252651 -0.2914217 1.5215575 2.2030260 -1.7402377 1.4642550 0.710003 DOWN 1.8129792 0.6814685

3648 7/9/2020 47.66 48.07 47.22 47.57 47.57 1013400 -0.5643813 -0.1252651 -0.2914217 1.5215575 2.2030260 -1.7402377 0.849999 DOWN 0.1661566 1.8129792

3649 7/10/2020 47.80 48.71 47.69 48.27 48.27 866200 1.4715157 -0.5643813 -0.1252651 -0.2914217 1.5215575 2.2030260 1.020000 UP 0.4391161 0.1661566

3650 7/13/2020 48.52 49.02 48.34 48.47 48.47 1043000 0.4143381 1.4715157 -0.5643813 -0.1252651 -0.2914217 1.5215575 0.680000 UP 2.0358969 0.4391161

3651 7/14/2020 48.39 49.94 48.10 49.93 49.93 729800 3.0121704 0.4143381 1.4715157 -0.5643813 -0.1252651 -0.2914217 1.840001 UP 1.0571776 2.0358969

> str(AOS.CSV)

'data.frame': 3651 obs. of 17 variables:

$ Date : chr "1/11/2006" "1/12/2006" "1/13/2006" "1/17/2006" ...

$ Open : num 6.68 6.6 6.47 6.48 6.53 ...

$ High : num 6.71 6.66 6.5 6.58 6.74 ...

$ Low : num 6.57 6.44 6.45 6.46 6.51 ...

$ Close : num 6.6 6.46 6.47 6.53 6.74 ...

$ Adj.Close : num 2.82 2.75 2.76 2.79 2.87 ...

$ Volume : int 1474800 1913200 1042800 1850400 2258800 1537200 6706000 3603600 1592400 916800 ...

$ LGFet : num -1.493 -2.172 0.207 0.979 3.164 ...

$ Lag1 : num 2.395 -1.493 -2.172 0.207 0.979 ...

$ Lag2 : num 3.398 2.395 -1.493 -2.172 0.207 ...

$ Lag3 : num 3.43 3.4 2.39 -1.49 -2.17 ...

$ Lag4 : num -0.945 3.433 3.398 2.395 -1.493 ...

$ Lag5 : num 0.543 -0.945 3.433 3.398 2.395 ...

$ Daily.Range: num 0.143 0.228 0.06 0.123 0.227 ...

$ Direction : chr "DOWN" "DOWN" "UP" "UP" ...

$ RangeL1 : num 1.003 3.888 0.679 2.379 0.773 ...

$ RangeL2 : num 0.0349 1.0034 3.8878 0.6794 2.3788 ...

> Shuffle=sample(3651,3651)

> InSample=Shuffle[1:2000]

> OutSample=Shuffle[2001:3651]

> X.AOS=AOS.CSV[,16:17]

> Y.AOS=AOS.CSV[,14]

> median[Y.AOS]

Error in median[Y.AOS] : object of type 'closure' is not subsettable

> median(Y.AOS)

[1] 0.365

> Y.AOS[Y.AOS>.365]="HighRisk"

> Y.AOS[Y.AOS<=.365]="LowRisk"

> Y.AOS=as.factor(Y.AOS)

> TrainX.AOS=X.AOS[InSample,]

> TrainY.AOS=Y.AOS[InSample]

> TestX.AOS=X.AOS[OutSample,]

> TestY.AOS=Y.AOS[OutSample]

> LnX.AOS=log(X.AOS)

> StLnX.AOS=apply(LnX.AOS,2,scale)

> TrainStLnX.AOS=StLnX.AOS[InSample,]

> TestStLnX.AOS=StLnX.AOS[OutSample,]

> NewProbeKnnBig(TrainStLnX.AOS,StProbeX,TrainY.AOS,25)

Error in knn(TrialX, ProbeX, TrialY, k) : no missing values are allowed

> NewProbeKnnBig(TrainStLnX.AOS,TestStLnX.AOS,TrainY.AOS,25)

Error in knn(TrialX, ProbeX, TrialY, k) : no missing values are allowed

> Knn(TrainStLnX.AOS,TestStLnX.AOS,TrainY.AOS,25)

Error in Knn(TrainStLnX.AOS, TestStLnX.AOS, TrainY.AOS, 25) :

could not find function "Knn"

> library(class)

> NewProbeKnnBig(TrainStLnX.AOS,TestStLnX.AOS,TrainY.AOS,25)

Error in knn(TrialX, ProbeX, TrialY, k) : no missing values are allowed

> Knn(TrainStLnX.AOS,TestStLnX.AOS,TrainY.AOS,25)

Error in Knn(TrainStLnX.AOS, TestStLnX.AOS, TrainY.AOS, 25) :

could not find function "Knn"

> knn(TrainStLnX.AOS,TestStLnX.AOS,TrainY.AOS,25)

Error in knn(TrainStLnX.AOS, TestStLnX.AOS, TrainY.AOS, 25) :

no missing values are allowed

> testing=TrainStLnX.AOS

> testing=na.omit(testing)

> length(testing)

[1] 0

> testing=TrainStLnX.AOS

> class(testing)

[1] "matrix" "array"

> testing[2,2]

RangeL2

NaN

> TrainStLnX.AOS[2,2]

RangeL2

NaN

> TrainStLnX.AOS[2]

[1] NaN

> X.AOS[2,2]

[1] 1.00342

> StlnX.AOS[2,2]

Error: object 'StlnX.AOS' not found

> StLnX.AOS[2,2]

RangeL2

NaN

> StLnX.AOS[2]

[1] NaN

> LnX.AOS[2,2]

[1] 0.003414104

> TrainStLnX.AOS=LnX.AOS[InSample,]

> TestStLnX.AOS=LnX.AOS[InSample,]

> NewProbeKnnBig(TrainStLnX.AOS,TestStLnX.AOS,TrainY.AOS,25)

NULL

> NewProbeKnnBig(TrainStLnX.AOS,TestStLnX.AOS,TrainY.AOS,55)

NULL

> NewProbeKnnBig(TrainStLnX.AOS,TestStLnX.AOS,TrainY.AOS,5)

NULL

> NewProbeKnnBig(TrainStLnX.AOS,TestStLnX.AOS,TrainY.AOS,55)

NULL

> normalize <- function(x) {(x-min(x))/(max(x)-min(x))}

> n1=normalize(TrainStLnX.AOS)

> n2=normalize(TestStLnX.AOS)

> NewProbeKnnBig(n1,n2,TrainY.AOS,55)

NULL

> NewProbeKnnBig(n1,n2,TrainY.AOS,25)

NULL

> NewProbeKnnBig(n1,n2,TrainY.AOS,5)

NULL

> kSearch =

+ function (X,Y,split=100, d=25)

+ {

+ if(class(Y)!= "factor") stop('Y is not factor type')

+ if(class(X)!= "data.frame") stop('X is not data.frame type')

+ rows = nrow(X)

+ Hold = rep(NA,d)

+ Shuffle=sample(rows,rows)

+ for(k in 1:d) {

+ InSample=Shuffle[1:split]

+ OutSample=Shuffle[(split+1):rows]

+ TrainX = X[InSample,]

+ TrainY = Y[InSample]

+ TestX = X[OutSample,]

+ TestY = Y[OutSample]

+ knn.pred = knn(TrainX, TestX, TrainY, k)

+ table.out = table(knn.pred,TestY)

+ Hold[k] = (table.out[1,1]+table.out[2,2])/sum(table.out)

+ }

+ return(Hold)

+ }

> junk=kSearch(X.AOS,Y.AOS,1200,50)

> plot(junk,type="1",tlab="Rate",xlab="k"

+ )

Error in plot.xy(xy, type, ...) : invalid plot type '1'

In addition: Warning messages:

1: In plot.window(...) : "tlab" is not a graphical parameter

2: In plot.xy(xy, type, ...) : "tlab" is not a graphical parameter

> plot(junk,type="1",ylab="Rate",xlab="k")

Error in plot.xy(xy, type, ...) : invalid plot type '1'

In addition: Warning message:

"tlab" is not a graphical parameter

> plot(junk,type="I",ylab="Rate",xlab="k")

Error in plot.xy(xy, type, ...) : invalid plot type 'I'

> plot(junk,type="l",ylab="Rate",xlab="k")

> kcvSearch =

+ function (X,Y,split=100, d=25, m=10)

+ {

+ if(class(Y)!= "factor") stop('Y is not factor type')

+ if(class(X)!= "data.frame") stop('X is not data.frame type')

+ rows = nrow(X)

+ Hold = rep(NA,d)

+

+ for(k in 1:d) {

+ Store = rep(NA, m)

+

+ for(j in 1:m) {

+ Shuffle=sample(rows,rows)

+ InSample=Shuffle[1:split]

+ OutSample=Shuffle[(split+1):rows]

+ TrainX = X[InSample,]

+ TrainY = Y[InSample]

+ TestX = X[OutSample,]

+ TestY = Y[OutSample]

+ knn.pred = knn(TrainX, TestX, TrainY, k)

+ table.out = table(knn.pred,TestY)

+ Store[j] = (table.out[1,1]+table.out[2,2])/sum(table.out)

+ }

+ Hold[k] = mean(Store)

+ }

+ return(Hold)

+ }

> junk=kcvSearch(X.AOS,Y.AOS,1200,50,30)

> plot(junk,type="l",ylab="Rate",xlab="k")

>

**5.5**

library(ggplot2)

head(diamonds)

#1

ggplot(diamonds, aes(cut)) +

geom\_bar()

#2

ggplot(diamonds, aes(x=cut,color=color))+

geom\_bar()

#3

ggplot(diamonds, aes(price))+

geom\_freqpoly()

#Most diamonds sell at less than $2500.

#4

ggplot(diamonds, aes(x = clarity,y = price))+

geom\_point()

#SI and VS clarity seems to have the highest prices while I1 seems to have the lowest.

#5

ggplot(diamonds, aes(x = cut, y = depth))+

geom\_boxplot()

#The boxplot interquartile range seems to get smaller as you go from fair to ideal cut and the ideal depth seems to be 63 in depth.

#6

ggplot(diamonds, aes(x = carat,y = price))+

geom\_point()

#7

ggplot(diamonds, aes(x = carat,y = price, color = cut))+

geom\_point()