DATA 622 - HW1

Michael Y.

3/18/2020

Contents

DATA 622 - HW1

Q1: Naive Bayes

Table 1: Prospect Data

agegroup	networth	status	credit	prospect
youth	high	employed	fair	no
youth	high	employed	excellent	no
middle	high	employed	fair	yes
senior	medium	employed	fair	yes
senior	low	unemployed	fair	yes
senior	low	unemployed	excellent	no
middle	low	unemployed	excellent	yes
youth	medium	employed	fair	no
youth	low	unemployed	fair	yes
senior	medium	unemployed	fair	yes
youth	medium	unemployed	excellent	yes
middle	medium	employed	excellent	yes
middle	high	unemployed	fair	yes
senior	medium	employed	excellent	no

```
df1 <- read.csv('HW1-Q1-40.csv', skip = 1, header = T,
                nrows = 14, stringsAsFactors  = T) %>%
  rename(agegroup = 'age.group',
         credit = 'credit_rating',
         prospect = "class.prospect")

df1 %>%
  kable(caption = "Prospect Data") %>%
  kable_styling(c("bordered","striped"),full_width = F)
```

```
summary(df1)
```

```
##    agegroup   networth         status          credit   prospect
##  middle:4   high  :4   employed  :7   excellent:6   no :5
##  senior:5   low   :4   unemployed:7   fair     :8   yes:9
##  youth :5   medium:6
```

You have been hired by a local electronics retailer and the above dataset has been given to you.

Manager Bayes Jr. 9th wants to create a spreadsheet to predict if a customer is a likely prospect.

To that end,

1) Compute prior probabilities for the Prospect Yes/No

```
#### Number of observations
N <- length(df1$prospect)
N
```

```
## [1] 14
```

```r
#### Tally of Prospect=[yes|no]
Prospect.Prior.Tally <- table(df1$prospect)
Prospect.Prior.Tally
```

```
##
##  no yes
##   5   9
```

```r
#### Probability of Prospect=[yes|no]
Prospect.Prior.Prob <- prop.table(table(df1$prospect))
Prospect.Prior.Prob
```

```
##
##        no       yes
## 0.3571429 0.6428571
```

$P(prospect = no) = 0.3571429$
$P(prospect = yes) = 0.6428571$

2) Compute the conditional probabilities

- $P(agegroup = youth|prospect = yes)$ and
- $P(agegroup = youth|prospect = no)$

```r
library(janitor)
```

where age-group is a predictor variable.

```
##
## Attaching package: 'janitor'

## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```r
#### Conditional Probabilities for agegroup
condprob.agegroup  <- df1 %>%
  tabyl(prospect,agegroup) %>%
  adorn_percentages("row")
rownames(condprob.agegroup) <- t(condprob.agegroup["prospect"])
condprob.agegroup
```

```
##  prospect    middle    senior     youth
##        no 0.0000000 0.4000000 0.6000000
##       yes 0.4444444 0.3333333 0.2222222
```

```r
#### Conditional Probabilities for networth
condprob.networth  <- df1 %>%
  tabyl(prospect,networth) %>%
  adorn_percentages("row")
rownames(condprob.networth) <- t(condprob.networth["prospect"])
condprob.networth
```

```
##  prospect      high       low    medium
##        no 0.4000000 0.2000000 0.4000000
##       yes 0.2222222 0.3333333 0.4444444
```

```r
#### Conditional Probabilities for status
condprob.status  <- df1 %>%
  tabyl(prospect,status) %>%
  adorn_percentages("row")
rownames(condprob.status) <- t(condprob.status["prospect"])
condprob.status
```

```
##  prospect  employed unemployed
##        no 0.8000000  0.2000000
##       yes 0.3333333  0.6666667
```

```r
#### Conditional Probabilities for credit
condprob.credit  <- df1 %>%
  tabyl(prospect,credit) %>%
  adorn_percentages("row")
rownames(condprob.credit) <- t(condprob.credit["prospect"])
condprob.credit
```

```
##  prospect excellent      fair
##        no 0.6000000 0.4000000
##       yes 0.3333333 0.6666667
```

Compute the conditional probabilities for each predictor variable, namely, (age_group,networth,status,credit_rat

Conditional Probabilities:

$$P(agegroup = youth|prospect = yes) = 0.2222222$$
$$P(agegroup = middle|prospect = yes) = 0.4444444$$
$$P(agegroup = senior|prospect = yes) = 0.3333333$$
$$P(agegroup = youth|prospect = no) = 0.6$$
$$P(agegroup = middle|prospect = no) = 0$$
$$P(agegroup = senior|prospect = no) = 0.4$$
$$P(networth = high|prospect = yes) = 0.2222222$$
$$P(networth = low|prospect = yes) = 0.3333333$$
$$P(networth = medium|prospect = yes) = 0.4444444$$
$$P(networth = high|prospect = no) = 0.4$$
$$P(networth = low|prospect = no) = 0.2$$
$$P(networth = medium|prospect = no) = 0.4$$
$$P(status = employed|prospect = yes) = 0.3333333$$
$$P(status = unemployed|prospect = yes) = 0.6666667$$
$$P(status = employed|prospect = no) = 0.8$$
$$P(status = unemployed|prospect = no) = 0.2$$
$$P(credit = fair|prospect = yes) = 0.6666667$$
$$P(credit = excellent|prospect = yes) = 0.3333333$$
$$P(credit = fair|prospect = no) = 0.4$$
$$P(credit = excellent|prospect = no) = 0.6$$

3) Posterior Probabilities

Assuming the assumptions of Naive Bayes are met,

compute the posterior probability $P(prospect|X)$

where X is one of the predictor variable. By Bayes rule, the **posterior** probability is defined as $P(c|x) = \frac{P(x|c) \cdot P(c)}{P(x)}$
where

- $P(x|c)$ is the **likelihood**
- $P(c)$ is the **class prior** probability
- $P(x)$ is the **predictor prior** probability.

Here, $P(prospect|x) = \frac{P(x|prospect) \cdot P(prospect)}{P(x)}$.

Under the Naive Bayes assumption that multiple features $X = (x_1, x_2, x_3, x_4)$ are conditionally independent, given the class, we have

$$P(x_1, x_2, x_3, x_4|\text{prospect}) = P(x_1|\text{prospect}) \cdot P(x_2|\text{prospect}) \cdot P(x_3|\text{prospect}) \cdot P(x_4|\text{prospect})$$

Here,

$$
\begin{aligned}
P(\text{prospect}|x_1,x_2,x_3,x_4) &= \frac{P(x_1,x_2,x_3,x_4|\text{prospect})\cdot P(\text{prospect})}{P(x_1,x_2,x_3,x_4)} \\
&= \frac{P(x_1|\text{prospect})\cdot P(x_2|\text{prospect})\cdot P(x_3|\text{prospect})\cdot P(x_4|\text{prospect}))\cdot P(\text{prospect})}{P(x_1,x_2,x_3,x_4)}
\end{aligned}
$$

where the denominator is

$$
\begin{aligned}
P(x_1,x_2,x_3,x_4) = {} & P(x_1|\text{prospect=yes})\cdot P(x_2|\text{prospect=yes})\cdot P(x_3|\text{prospect=yes})\cdot P(x_4|\text{prospect=yes}))\cdot P(\text{prospect=yes}) \\
& + P(x_1|\text{prospect=no})\cdot P(x_2|\text{prospect=no})\cdot P(x_3|\text{prospect=no})\cdot P(x_4|\text{prospect=no}))\cdot P(\text{prospect=no})
\end{aligned}
$$

Example 1: a poor, unemployed youth with fair credit Consider the following values for the predictor variables:

- agegroup = youth
- networth = low
- status = unemployed
- credit = fair

Then the numererator is

$$
P(agegroup = youth|\text{prospect})\cdot P(networth = low|\text{prospect})\cdot P(status = unemployed|\text{prospect})\cdot P(credit = fair|\text{prospect}))\cdot P
$$

where prospect can be either yes or no.

The denominator is the sum of the two cases.

For **prospect = yes** we have $P(prospect = yes) = 0.6428571$
and

$$
\begin{aligned}
P(agegroup = youth|prospect = yes) &= 0.2222222 \\
P(networth = low|prospect = yes) &= 0.3333333 \\
P(status = unemployed|prospect = yes) &= 0.6666667 \\
P(credit = fair|prospect = yes) &= 0.6666667
\end{aligned}
$$

which computes as

```
posteriorYesNumerator <- Prospect.Prior.Prob["yes"] *
  condprob.agegroup["yes","youth"] *
  condprob.networth["yes","low"] *
  condprob.status["yes","unemployed"] *
  condprob.credit["yes","fair"]
posteriorYesNumerator
```

```
##        yes
## 0.02116402
```

For **prospect = no** we have $P(prospect = no) = 0.3571429$
and

$$P(agegroup = youth|prospect = no) = 0.6$$
$$P(networth = low|prospect = no) = 0.2$$
$$P(status = unemployed|prospect = no) = 0.2$$
$$P(credit = fair|prospect = no) = 0.4$$

which computes as

```
posteriorNoNumerator <- Prospect.Prior.Prob["no"] *
  condprob.agegroup["no","youth"] *
  condprob.networth["no","low"] *
  condprob.status["no","unemployed"] *
  condprob.credit["no","fair"]
posteriorNoNumerator
```

```
##          no
## 0.003428571
```

Therefore, the denominator is

```
evidence = as.numeric(posteriorYesNumerator + posteriorNoNumerator)
evidence
```

```
## [1] 0.02459259
```

so the posterior for **prospect = yes** given the above features is

```
posteriorYes = posteriorYesNumerator / evidence
posteriorYes
```

```
##       yes
## 0.8605852
```

and the posterior for **prospect = no** is

```
posteriorNo = posteriorNoNumerator / evidence
posteriorNo
```

```
##        no
## 0.1394148
```

This does seem rather counterintutive – that an unemployed youth, with low net worth, and credit which is only "fair", should be such a strong "prospect", i.e., 86% yes vs. 14% no.

(Purely by coincidence, this exact case does happen to be in the input dataset – though I did not select it on that basis, and only realized that after performing the above calculations.)

Example 2: a wealthy, employed senior with excellent credit On the other hand, let's consider the following values for the predictor variables:

- agegroup = senior
- networth = high
- status = employed
- credit = excellent

Then the numererator is

$$P(agegroup = senior|\text{prospect}) \cdot P(networth = high|\text{prospect}) \cdot P(status = employed|\text{prospect}) \cdot P(credit = excellent|\text{prospect}$$

where prospect can be either yes or no.

The denominator is the sum of the two cases.

For **prospect = yes** we have $P(prospect = yes) = 0.6428571$
and

$$\begin{aligned}
P(agegroup = senior|prospect = yes) &= 0.3333333 \\
P(networth = high|prospect = yes) &= 0.2222222 \\
P(status = employed|prospect = yes) &= 0.3333333 \\
P(credit = excellent|prospect = yes) &= 0.3333333
\end{aligned}$$

which computes as

```
posteriorYesNumerator2 <- Prospect.Prior.Prob["yes"] *
  condprob.agegroup["yes","senior"] *
  condprob.networth["yes","high"] *
  condprob.status["yes","employed"] *
  condprob.credit["yes","excellent"]
posteriorYesNumerator2
```

```
##         yes
## 0.005291005
```

For **prospect = no** we have $P(prospect = no) = 0.3571429$
and

$$\begin{aligned}
P(agegroup = senior|prospect = no) &= 0.4 \\
P(networth = high|prospect = no) &= 0.4 \\
P(status = employed|prospect = no) &= 0.8 \\
P(credit = excellent|prospect = no) &= 0.6
\end{aligned}$$

which computes as

```
posteriorNoNumerator2 <- Prospect.Prior.Prob["no"] *
  condprob.agegroup["no","senior"] *
  condprob.networth["no","high"] *
  condprob.status["no","employed"] *
  condprob.credit["no","excellent"]
posteriorNoNumerator2
```

```
##         no
## 0.02742857
```

Therefore, the denominator is

```
evidence2 = as.numeric(posteriorYesNumerator2 + posteriorNoNumerator2)
evidence2
```

```
## [1] 0.03271958
```

so the posterior for **prospect = yes** given the above features is

```
posteriorYes2 = posteriorYesNumerator2 / evidence2
posteriorYes2
```

```
##       yes
## 0.1617076
```

and the posterior for **prospect = no** is

```
posteriorNo2 = posteriorNoNumerator2 / evidence2
posteriorNo2
```

```
##        no
## 0.8382924
```

This does seem rather counterintutive – that an employed senior, with high net worth, and excellent credit, should be such a weak "prospect", i.e., 84% no vs. 16% yes.

I'm unsure what sort of electronics they are selling, but their model does seem quite naive.

Q2: Exploratory Data Analysis

You just recently joined a datascience team.

There are two datasets `junk1.txt` and `junk2.csv`

They have two options:

1. They can go back to the client and ask for more data to remedy problems with the data.

2. They can accept the data and undertake a major analytics exercise.

The team is relying on your data science skills to determine how they should proceed.

Can you explore the data and recommend actions for each file, enumerating the reasons?

```
library(ggplot2)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2

##
## Attaching package: 'GGally'

## The following object is masked from 'package:dplyr':
##
##     nasa
```

First dataset ("junk1")

```
junk1 <- read.csv('junk1.txt', header = TRUE, sep = " ", dec = ".")
# Correlation matrix (class is numeric 1|2 )
cor(junk1)
```

```
##                 a           b       class
## a      1.00000000 -0.10215075 -0.06036116
## b     -0.10215075  1.00000000 -0.02453798
## class -0.06036116 -0.02453798  1.00000000
```

```
# Standard Deviation of entire dataset
sapply(X = junk1, FUN = sd)
```

```
##         a         b     class
## 1.2677402 1.4460671 0.5025189
```

```
# Standard Deviation of class=1
sapply(X=junk1[junk1$class==1,], FUN=sd)
```

```
##        a        b    class
## 1.266085 1.457040 0.000000
```

```r
# Standard Deviation of class=2
sapply(X=junk1[junk1$class==2,], FUN=sd)
```

```
##        a        b    class
## 1.277625 1.448926 0.000000
```

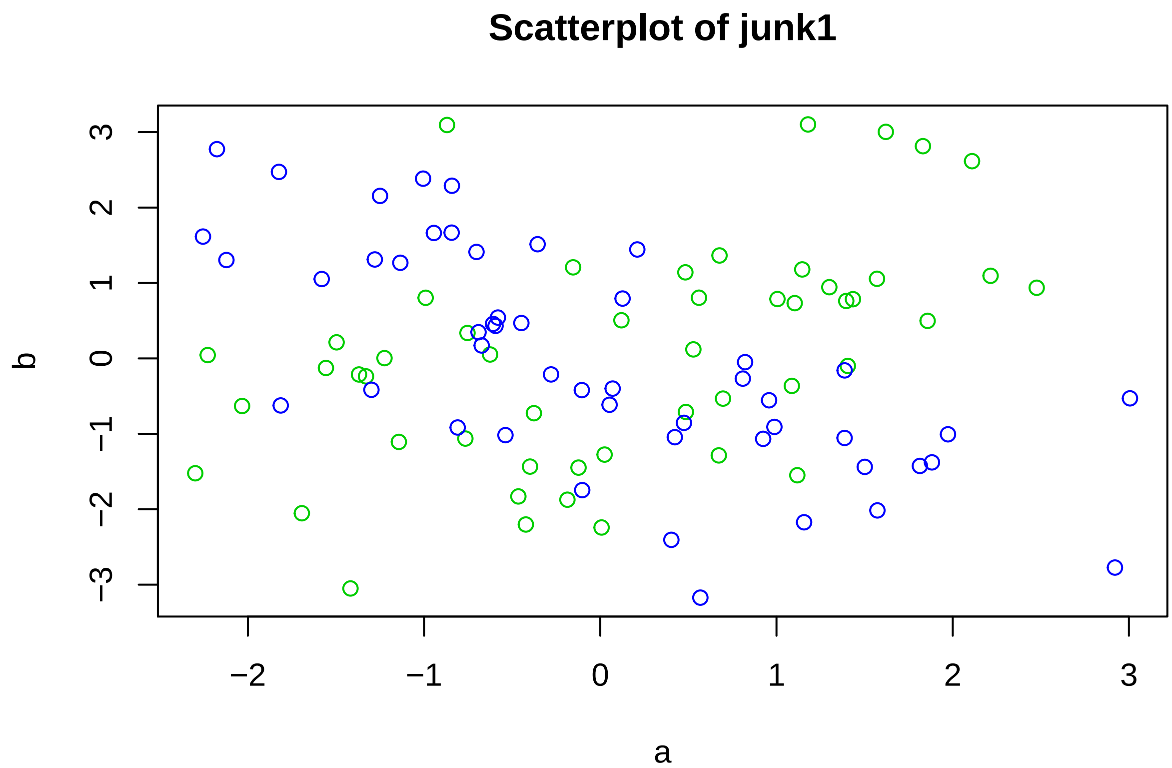
```r
# Make class into a factor
junk1$class <- as.factor(junk1$class)
# summary of junk1
summary(junk1)
```

```
##        a                  b             class
##  Min.   :-2.29854   Min.   :-3.17174   1:50
##  1st Qu.:-0.85014   1st Qu.:-1.04712   2:50
##  Median :-0.04754   Median :-0.07456
##  Mean   : 0.04758   Mean   : 0.01324
##  3rd Qu.: 1.09109   3rd Qu.: 1.05342
##  Max.   : 3.00604   Max.   : 3.10230
```

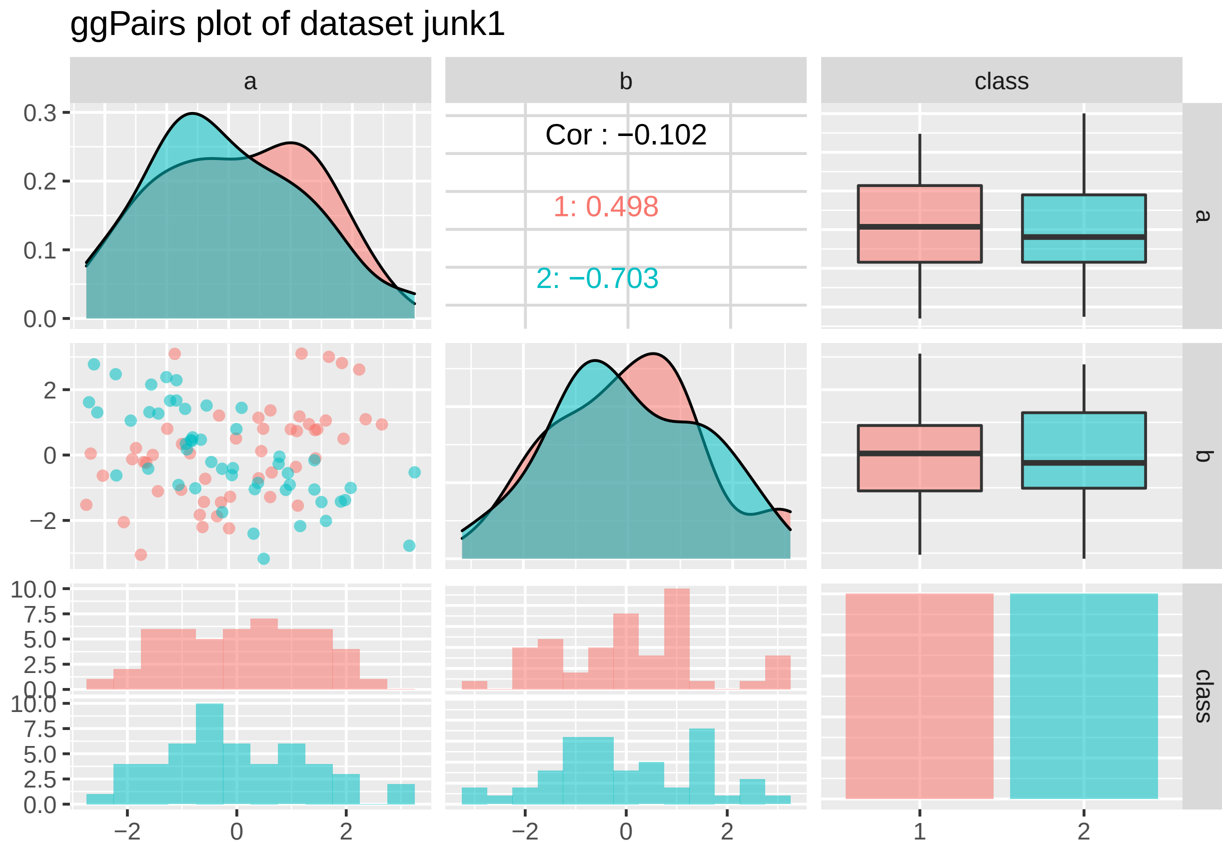
```r
# table of junk1 class
table(junk1$class)
```

```
##
##  1  2
## 50 50
```

```r
# plot the data with classes colored green and blue
plot(b~a,data=junk1,col=as.numeric(class)+2,main="Scatterplot of junk1")
```




```r
ggpairs(junk1, aes(col = class, alpha = 0.25),
        title = "ggPairs plot of dataset junk1",
        lower=list(combo=wrap("facethist",  binwidth=0.5)))
```



This is a small dataset, with only 100 observations.

The dataset is balanced – there are 50 observations in each of the two classes.

Each class appears to be normally distributed, with mean/median close to zero and similar standard deviations.

As we don't know what is the purpose of the data, the mission is unclear.

If the purpose is classification, this would be difficult because the data is overlapping across the space – it doesn't appear that there is sufficient differentiation to enable classification.

Because the dataset is so small, it may be necessary to ask whether additional data may be available.

Second dataset ("junk2")

```r
junk2 <- read.csv('junk2.csv', header = TRUE, sep = ",", dec = ".")

# Correlation matrix (class is numeric 0/1 )
cor(junk2)
```

```
##                a           b      class
## a     1.00000000  0.07392388  0.2151504
## b     0.07392388  1.00000000 -0.2039553
## class 0.21515039 -0.20395533  1.0000000
```

```r
# Standard Deviation of entire dataset
sapply(X = junk2, FUN = sd)
```

```
##         a         b     class
## 1.2980758 1.3143855 0.2420917
```

```r
# Standard Deviation of class=0
sapply(X=junk2[junk2$class==0,], FUN=sd)
```

```
##        a        b    class
## 1.303156 1.322861 0.000000
```

```r
# Standard Deviation of class=1
sapply(X=junk2[junk2$class==1,], FUN=sd)
```

```
##         a         b     class
## 0.4900410 0.4938064 0.0000000
```

```r
# Make class into a factor
junk2$class <- as.factor(junk2$class)

# table of junk2 class
table(junk2$class)
```

```
##
##    0    1
## 3750  250
```

```r
# summary of junk2
summary(junk2)
```

```
##        a                  b             class
##  Min.   :-4.16505   Min.   :-3.90472   0:3750
##  1st Qu.:-1.01447   1st Qu.:-0.89754   1: 250
##  Median : 0.08754   Median :-0.08358
##  Mean   :-0.05126   Mean   : 0.05624
##  3rd Qu.: 0.89842   3rd Qu.: 1.00354
##  Max.   : 4.62647   Max.   : 4.31052
```

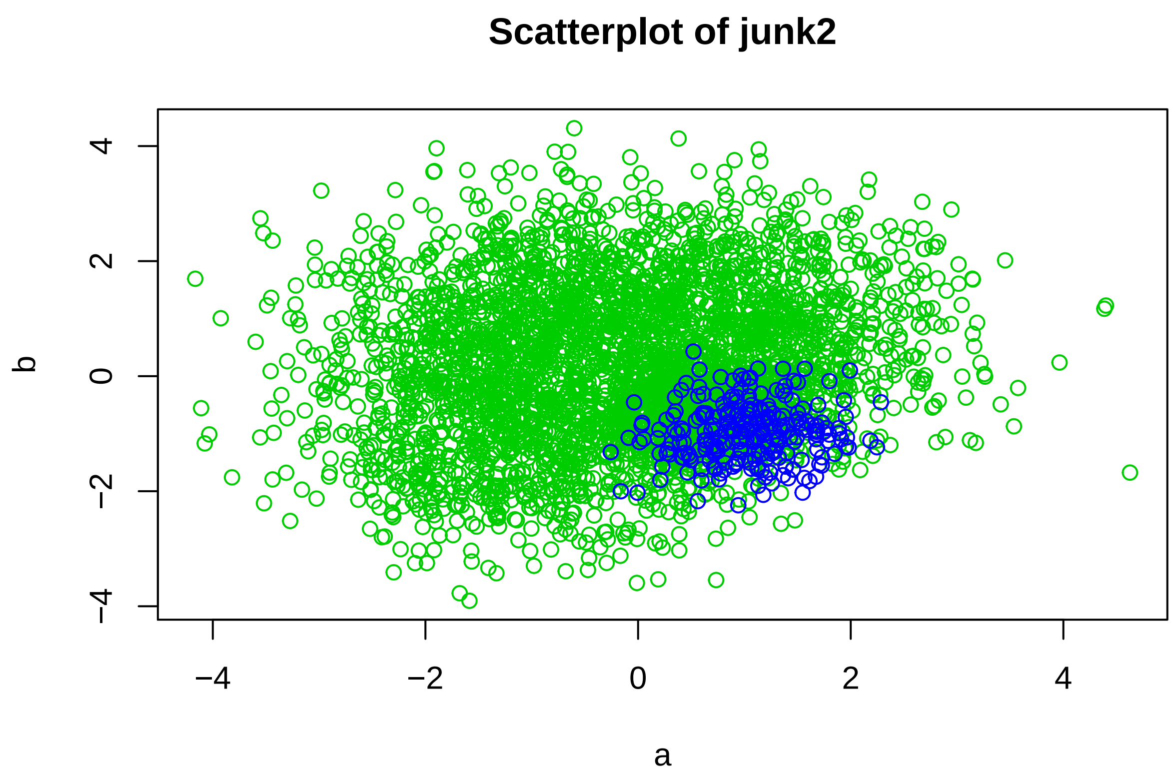
```r
# summary of junk2, larger class only
summary(junk2[junk2$class==0,])
```

```
##        a                  b              class
##  Min.   :-4.16505   Min.   :-3.904721   0:3750
##  1st Qu.:-1.08985   1st Qu.:-0.802595   1:   0
##  Median :-0.02757   Median : 0.009806
##  Mean   :-0.12336   Mean   : 0.125449
##  3rd Qu.: 0.81831   3rd Qu.: 1.090155
##  Max.   : 4.62647   Max.   : 4.310516
```

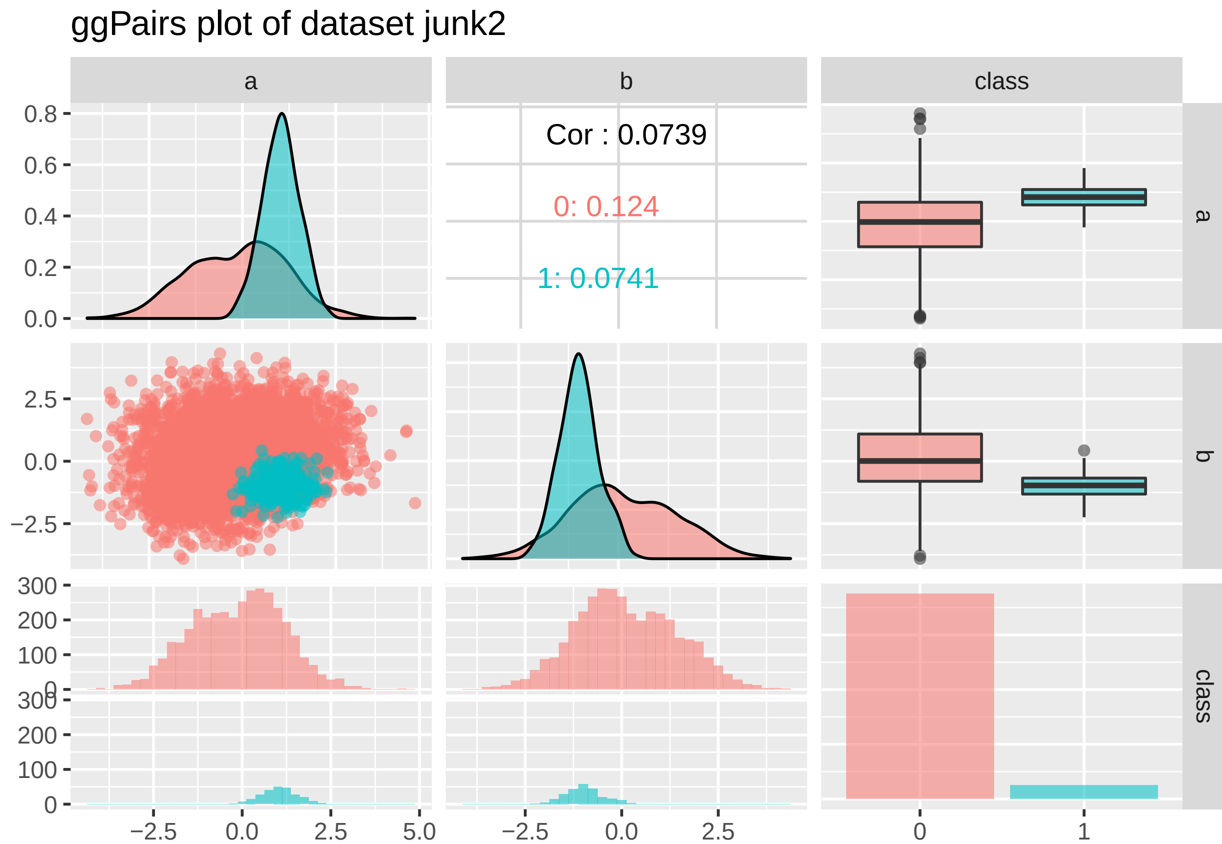
```
# summary of junk2, smaller class only
summary(junk2[junk2$class==1,])
```

```
##        a                 b             class
##  Min.   :-0.2573   Min.   :-2.2443   0:  0
##  1st Qu.: 0.7024   1st Qu.:-1.3154   1:250
##  Median : 1.0378   Median :-0.9689
##  Mean   : 1.0303   Mean   :-0.9819
##  3rd Qu.: 1.3630   3rd Qu.:-0.6745
##  Max.   : 2.2836   Max.   : 0.4284
```

```
# plot the data with classes colored green and blue
plot(b~a,data=junk2,col=as.numeric(class)+2, main="Scatterplot of junk2")
```




```
ggpairs(junk2, aes(col = class, alpha = 0.25),
        title = "ggPairs plot of dataset junk2",
        lower=list(combo=wrap("facethist",binwidth=0.25)))
```



This is a much larger dataset, with 4000 observations.

However, the two classes are imbalanced, as there are 3750 observations in one class and 250 observations in the other, which is a ratio of 15:1.

The larger class is centered close to (0,0) with a standard deviation of 1.3 in each direction.

The smaller class is centered around (a=1,b=-1) with a much smaller standard deviation (0.5)

As such, if classification is the goal, this may be possible because the values in the smaller class are clustered in a narrow range.

A radial basis function may catch most of the items in the smaller class, however it would likely misclassify those elements from the larger class which happen to fall within the area dominated by the smaller class.

The issue of class imbalance may lead to overfitting, but this could be addressed by undersampling the larger dataset.

It is important to gain more information about the goal because it is unclear, for example, whether the two datasets (junk1 and junk2) are supposed to be somehow related to each other, or whether each represents unrelated data.

Q3: K-nearest neighbors

Load the ICU data

```r
# Please find icu.csv
# Read the icu.csv
icu_raw <- read.csv("icu.csv")
dim(icu_raw)
```

```
## [1] 200  21
```

```r
summary(icu_raw)
```

```
##        ID              STA             AGE             SEX             RACE      
##  Min.   :  4.0   Min.   :0.0   Min.   :16.00   Min.   :0.00   Min.   :1.000  
##  1st Qu.:210.2   1st Qu.:0.0   1st Qu.:46.75   1st Qu.:0.00   1st Qu.:1.000  
##  Median :412.5   Median :0.0   Median :63.00   Median :0.00   Median :1.000  
##  Mean   :444.8   Mean   :0.2   Mean   :57.55   Mean   :0.38   Mean   :1.175  
##  3rd Qu.:671.8   3rd Qu.:0.0   3rd Qu.:72.00   3rd Qu.:1.00   3rd Qu.:1.000  
##  Max.   :929.0   Max.   :1.0   Max.   :92.00   Max.   :1.00   Max.   :3.000  
##       SER             CAN             CRN             INF             CPR       
##  Min.   :0.000   Min.   :0.0   Min.   :0.000   Min.   :0.00   Min.   :0.000  
##  1st Qu.:0.000   1st Qu.:0.0   1st Qu.:0.000   1st Qu.:0.00   1st Qu.:0.000  
##  Median :1.000   Median :0.0   Median :0.000   Median :0.00   Median :0.000  
##  Mean   :0.535   Mean   :0.1   Mean   :0.095   Mean   :0.42   Mean   :0.065  
##  3rd Qu.:1.000   3rd Qu.:0.0   3rd Qu.:0.000   3rd Qu.:1.00   3rd Qu.:0.000  
##  Max.   :1.000   Max.   :1.0   Max.   :1.000   Max.   :1.00   Max.   :1.000  
##       SYS             HRA              PRE             TYP       
##  Min.   : 36.0   Min.   : 39.00   Min.   :0.00   Min.   :0.000  
##  1st Qu.:110.0   1st Qu.: 80.00   1st Qu.:0.00   1st Qu.:0.000  
##  Median :130.0   Median : 96.00   Median :0.00   Median :1.000  
##  Mean   :132.3   Mean   : 98.92   Mean   :0.15   Mean   :0.735  
##  3rd Qu.:150.0   3rd Qu.:118.25   3rd Qu.:0.00   3rd Qu.:1.000  
##  Max.   :256.0   Max.   :192.00   Max.   :1.00   Max.   :1.000  
##       FRA             PO2             PH              PCO            BIC       
##  Min.   :0.000   Min.   :0.00   Min.   :0.000   Min.   :0.0   Min.   :0.000  
##  1st Qu.:0.000   1st Qu.:0.00   1st Qu.:0.000   1st Qu.:0.0   1st Qu.:0.000  
##  Median :0.000   Median :0.00   Median :0.000   Median :0.0   Median :0.000  
##  Mean   :0.075   Mean   :0.08   Mean   :0.065   Mean   :0.1   Mean   :0.075  
##  3rd Qu.:0.000   3rd Qu.:0.00   3rd Qu.:0.000   3rd Qu.:0.0   3rd Qu.:0.000  
##  Max.   :1.000   Max.   :1.00   Max.   :1.000   Max.   :1.0   Max.   :1.000  
##       CRE            LOC       
##  Min.   :0.00   Min.   :0.000  
##  1st Qu.:0.00   1st Qu.:0.000  
##  Median :0.00   Median :0.000  
##  Mean   :0.05   Mean   :0.125  
##  3rd Qu.:0.00   3rd Qu.:0.000  
##  Max.   :1.00   Max.   :2.000  
```

```r
# The formula to fit is "STA ~ TYP + COMA + AGE + INF"
# subset it with these 5 features in the formula, and STA is the labelcol.
```

```r
# The dataset MUST Be numeric, except the labelcol
# The labelcol must be the last column in the data.frame
# All the other columns must be before the labelcol

icu_raw %>%
  mutate(COMA = ifelse(LOC == 2, 1, 0)) %>%
  select(TYP, COMA, AGE, INF, STA) -> icu


summary(icu)
```

```
##       TYP              COMA           AGE             INF             STA
##  Min.   :0.000   Min.   :0.00   Min.   :16.00   Min.   :0.00   Min.   :0.0
##  1st Qu.:0.000   1st Qu.:0.00   1st Qu.:46.75   1st Qu.:0.00   1st Qu.:0.0
##  Median :1.000   Median :0.00   Median :63.00   Median :0.00   Median :0.0
##  Mean   :0.735   Mean   :0.05   Mean   :57.55   Mean   :0.42   Mean   :0.2
##  3rd Qu.:1.000   3rd Qu.:0.00   3rd Qu.:72.00   3rd Qu.:1.00   3rd Qu.:0.0
##  Max.   :1.000   Max.   :1.00   Max.   :92.00   Max.   :1.00   Max.   :1.0
```

```r
# TYP
table(icu$TYP)
```

```
##
##   0   1
##  53 147
```

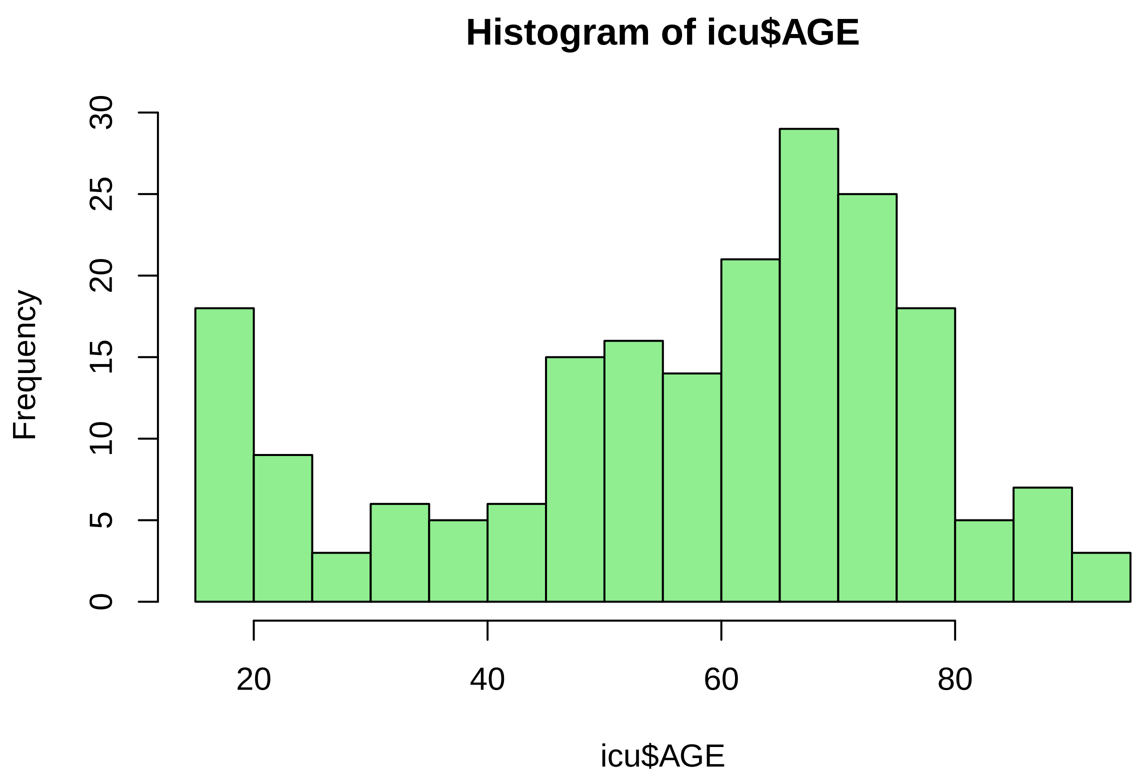
```r
# COMA
table(icu$COMA)
```

```
##
##   0   1
## 190  10
```

```r
# AGE
table(icu$AGE)
```

```
##
## 16 17 18 19 20 21 23 24 25 27 28 30 31 32 34 35 36 40 41 42 45 46 47 48 49 50
##  2  3  4  4  5  3  4  1  1  1  1  1  1  2  1  2  1  4  3  1  2  3  3  3  3  3
## 51 52 53 54 55 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77
##  2  1  3  2  8  2  2  3  7  4  2  5  6  4  5  5  6  7  6  6  5  4  2  8  4  6
## 78 79 80 82 83 84 85 87 88 89 91 92
##  4  1  3  2  1  1  1  2  4  1  2  1
```

```r
hist(icu$AGE,breaks = 16,col="lightgreen")
```




```r
# INF
table(icu$INF)
```

```
##
##   0   1
## 116  84
```

```r
# STA
table(icu$INF)
```

```
##
##   0   1
## 116  84
```

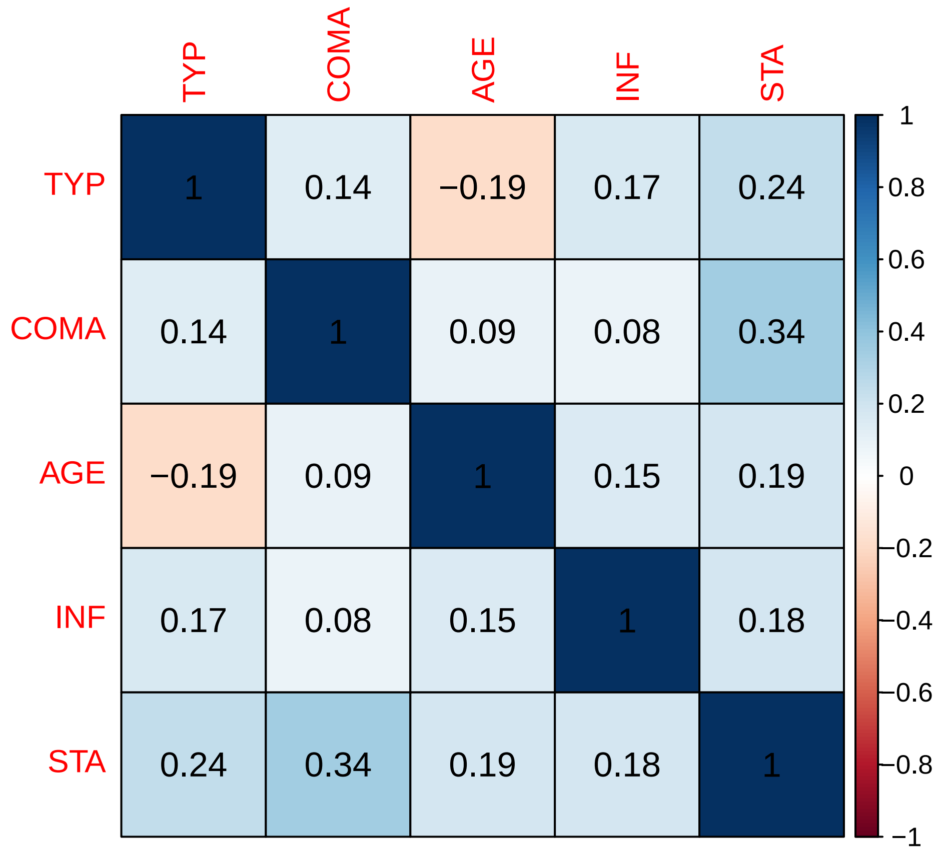
```r
# Correlation
cor(icu)
```

```
##             TYP       COMA         AGE        INF       STA
## TYP   1.0000000 0.13775344 -0.18695714 0.16664849 0.2435801
## COMA  0.1377534 1.00000000  0.09008299 0.08366755 0.3441236
## AGE  -0.1869571 0.09008299  1.00000000 0.15355452 0.1894579
## INF   0.1666485 0.08366755  0.15355452 1.00000000 0.1823492
## STA   0.2435801 0.34412360  0.18945786 0.18234920 1.0000000
```

```r
cormat <- as.matrix(cor(icu))
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```r
corrplot(corr = cormat, type = "full", outline = T,
         method = "color", sig.level = 0.05, insig = "blank",
         addCoef.col = "black",number.cex = 1.1,
         number.font = 1, number.digits = 2 )
```




```r
# summary stats
icu %>%
  mutate(TYP = factor(TYP), COMA = factor(COMA), INF = factor(INF),
         STA = factor(STA)) %>%
  summary()
```

```
##  TYP      COMA          AGE          INF      STA    
##  0: 53   0:190   Min.   :16.00   0:116   0:160  
##  1:147   1: 10   1st Qu.:46.75   1: 84   1: 40  
##                  Median :63.00                  
##                  Mean   :57.55                  
##                  3rd Qu.:72.00                  
##                  Max.   :92.00                  
```

Split the icu 70/30 train/test

```
# create training/test partitions

## Trying to find a seed which will reduce or eliminate class imbalance
## between the testing and training split.  After trial and error,
## this seed worked exactly to form an 80/20 split on the class variable.
set.seed(3)


N <- nrow(icu)   # N = 200
train_index <- sample(N, size = 0.7 * N)    # 140 random indices
train_icu <- icu[train_index, ]               # 140 cases
test_icu  <- icu[-train_index, ]              # 60 cases
save_test_icu <- test_icu

# check for class imbalance
STA_column <- which(colnames(icu)=="STA")      # column containing class labels
# Proportion of STA in Training set
table(STA_train = train_icu[,STA_column])/length(train_icu$STA)
```

```
## STA_train
##   0   1
## 0.8 0.2
```

```
# Proportion of STA in Testing set
table(STA_test = test_icu[,STA_column])/length(test_icu$STA)
```

```
## STA_test
##   0   1
## 0.8 0.2
```

Each of the training and testing sets contains the same proportion of each item in the STA class. This was found by trial-and-error adjustment of the initial seed.

KNN.R code

```
euclideanDist <- function(a, b){
  d = 0
  mincols = min(length(a),length(b)) # I had to change this in order to avoid
  for(i in c(1:mincols))             # subscript out-of-bounds errors
  {                                  # as extra columns are appended to the test set
    d = d + (a[[i]]-b[[i]])^2        # but those columns are not to be calculated
  }
  d = sqrt(d)
  return(d)
}
```

Euclidean Distance

```r
knn_predict2 <- function(test_data, train_data, k_value, labelcol){
  pred <- c()  #empty pred vector
  #LOOP-1
  for(i in c(1:nrow(test_data))){   #looping over each record of test data
    eu_dist =c()          #eu_dist & eu_char empty  vector
    eu_char = c()
    good = 0              #good & bad variable initialization with 0 value
    bad = 0

    #LOOP-2-looping over train data
    for(j in c(1:nrow(train_data))){

      #adding euclidean distance b/w test data point and train data to eu_dist vector
      eu_dist <- c(eu_dist, euclideanDist(test_data[i,-c(labelcol)], train_data[j,-c(labelcol)]))

      #adding class variable of training data in eu_char
      eu_char <- c(eu_char, as.character(train_data[j,][[labelcol]]))
    }

    eu <- data.frame(eu_char, eu_dist) #eu dataframe created with eu_char & eu_dist columns

    eu <- eu[order(eu$eu_dist),]       #sorting eu dataframe to gettop K neighbors
    eu <- eu[1:k_value,]               #eu dataframe with top K neighbors

    tbl.sm.df<-table(eu$eu_char)
    cl_label<-  names(tbl.sm.df)[[as.integer(which.max(tbl.sm.df))]]

    pred <- c(pred, cl_label)
    }
    return(pred) #return pred vector
}
```

KNN-Predict2 function

```r
accuracy <- function(test_data,labelcol,predcol){
  correct = 0
  for(i in c(1:nrow(test_data))){
    if(test_data[i,labelcol] == test_data[i,predcol]){
      correct = correct+1
    }
  }
  accu = (correct/nrow(test_data)) * 100
  return(accu)
}
```

Accuracy Metric

run the kNN.R for K=(3,5,7,15,25,50)

```r
# set of k values
kvalues <- c(3, 5, 7, 15, 25, 50)
numk <- length(kvalues)

# accuracy metric & contingency table
accuracy_results <- c()
confusion_matrix <- list()


### Reset test_icu, if re-running, to drop any extraneous columns on the right
test_icu <- test_icu[,1:5]

labelcol <- STA_column   ### column containing "STA" colname

# loop over the values for K
for (i in 1:numk) {
  #print(i)
    kval <- kvalues[i]
  #print(kval)
    whichk <- paste0("k=", kval)
  print(paste(i,whichk))
    # calc kNN predictions & add to test df
  #print("call knn_predict2")
    predictions <- knn_predict2(test_icu, train_icu, kval, labelcol)
  #print("return from knn_predict2")
    test_icu[whichk] <- predictions # append a column to test_icu

    # compute accuracy and contingency table
  #print("call accuracy")
    accuracy_results[whichk] <- accuracy(test_icu, labelcol, labelcol + i)
  print(paste("Accuracy[",whichk,"]",accuracy_results[whichk]))
  print(paste("confusion",whichk))
    confusion_matrix[[whichk]] <-
      addmargins(table(Pred = factor(test_icu[[labelcol + i]],
                                     levels = c("0", "1")),
                       Obs = test_icu[[labelcol]]))
  print(confusion_matrix[[whichk]])
  print("________________________________________")
}
```

```
## [1] "1 k=3"
## [1] "Accuracy[ k=3 ] 68.3333333333333"
## [1] "confusion k=3"
##      Obs
## Pred   0  1 Sum
##   0   40 11  51
##   1    8  1   9
##   Sum 48 12  60
## [1] "________________________________________"
## [1] "2 k=5"
## [1] "Accuracy[ k=5 ] 75"
```

```
## [1] "confusion k=5"
##      Obs
## Pred   0  1 Sum
##   0   43 10  53
##   1    5  2   7
##   Sum 48 12  60
## [1] "________________________________________"
## [1] "3 k=7"
## [1] "Accuracy[ k=7 ] 76.6666666666667"
## [1] "confusion k=7"
##      Obs
## Pred   0  1 Sum
##   0   44 10  54
##   1    4  2   6
##   Sum 48 12  60
## [1] "________________________________________"
## [1] "4 k=15"
## [1] "Accuracy[ k=15 ] 80"
## [1] "confusion k=15"
##      Obs
## Pred   0  1 Sum
##   0   48 12  60
##   1    0  0   0
##   Sum 48 12  60
## [1] "________________________________________"
## [1] "5 k=25"
## [1] "Accuracy[ k=25 ] 80"
## [1] "confusion k=25"
##      Obs
## Pred   0  1 Sum
##   0   48 12  60
##   1    0  0   0
##   Sum 48 12  60
## [1] "________________________________________"
## [1] "6 k=50"
## [1] "Accuracy[ k=50 ] 80"
## [1] "confusion k=50"
##      Obs
## Pred   0  1 Sum
##   0   48 12  60
##   1    0  0   0
##   Sum 48 12  60
## [1] "________________________________________"
```

submit the result confusionMatrix, Accuracy for each K

Accuracy results

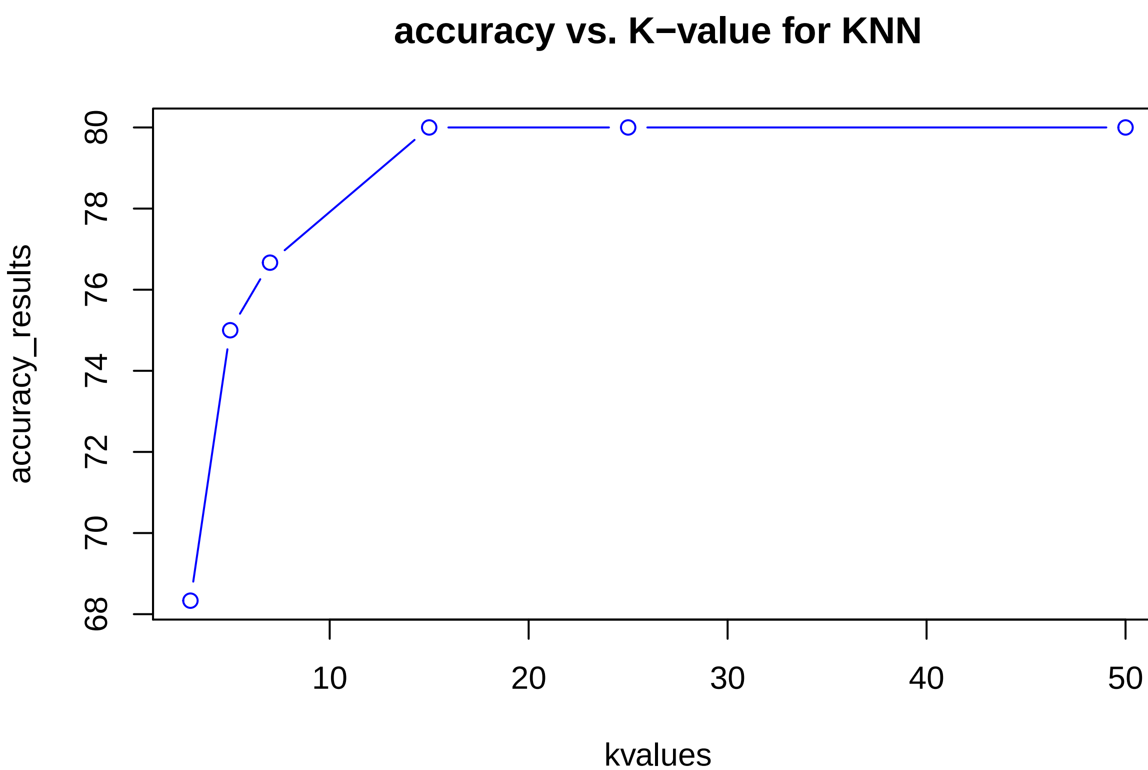
```r
accuracy_results %>%
  kable(caption = "kNN Accuracy for various K") %>%
  kable_styling(c("bordered","striped"),full_width = F)
```

Table 2: kNN Accuracy for various K

	x
k=3	68.33333
k=5	75.00000
k=7	76.66667
k=15	80.00000
k=25	80.00000
k=50	80.00000

Accuracy plot

```
plot(accuracy_results ~ kvalues,typ="b", col="blue",main = "accuracy vs. K-value for KNN")
```



Plot Accuracy vs K.

List of Confusion matrices, for each k

```
confusion_matrix
```

```
## $`k=3`
##      Obs
```

```
## Pred   0  1 Sum
##   0   40 11  51
##   1    8  1   9
##   Sum 48 12  60
##
## $`k=5`
##      Obs
## Pred   0  1 Sum
##   0   43 10  53
##   1    5  2   7
##   Sum 48 12  60
##
## $`k=7`
##      Obs
## Pred   0  1 Sum
##   0   44 10  54
##   1    4  2   6
##   Sum 48 12  60
##
## $`k=15`
##      Obs
## Pred   0  1 Sum
##   0   48 12  60
##   1    0  0   0
##   Sum 48 12  60
##
## $`k=25`
##      Obs
## Pred   0  1 Sum
##   0   48 12  60
##   1    0  0   0
##   Sum 48 12  60
##
## $`k=50`
##      Obs
## Pred   0  1 Sum
##   0   48 12  60
##   1    0  0   0
##   Sum 48 12  60
```

Commentary

write a short summary of your findings. While the accuracy increases as k is increased, the problem is that this model will ultimately classify every observation into the larger class, achieving 100 percent accuracy on those cases but achieving 0% accuracy on the items in the smaller class, all of which become misclassified into the larger class.

This is a problem which arises with imbalanced classes. It could be addressed, for example, by undersampling the large class, or oversampling (e.g., via repetition) the smaller class.

Grade

- Grade–>40

- Changing the code 10
- Running for different values of K 10
- Plot Accuracy 10
- Summary 10