Naive Bayes

Ben

30/09/2020

R Markdown

#Remove warnings

```
options(warn=-1)
```

#Removing env variables

```
rm(list=ls(all=TRUE))
```

#Reading libraries

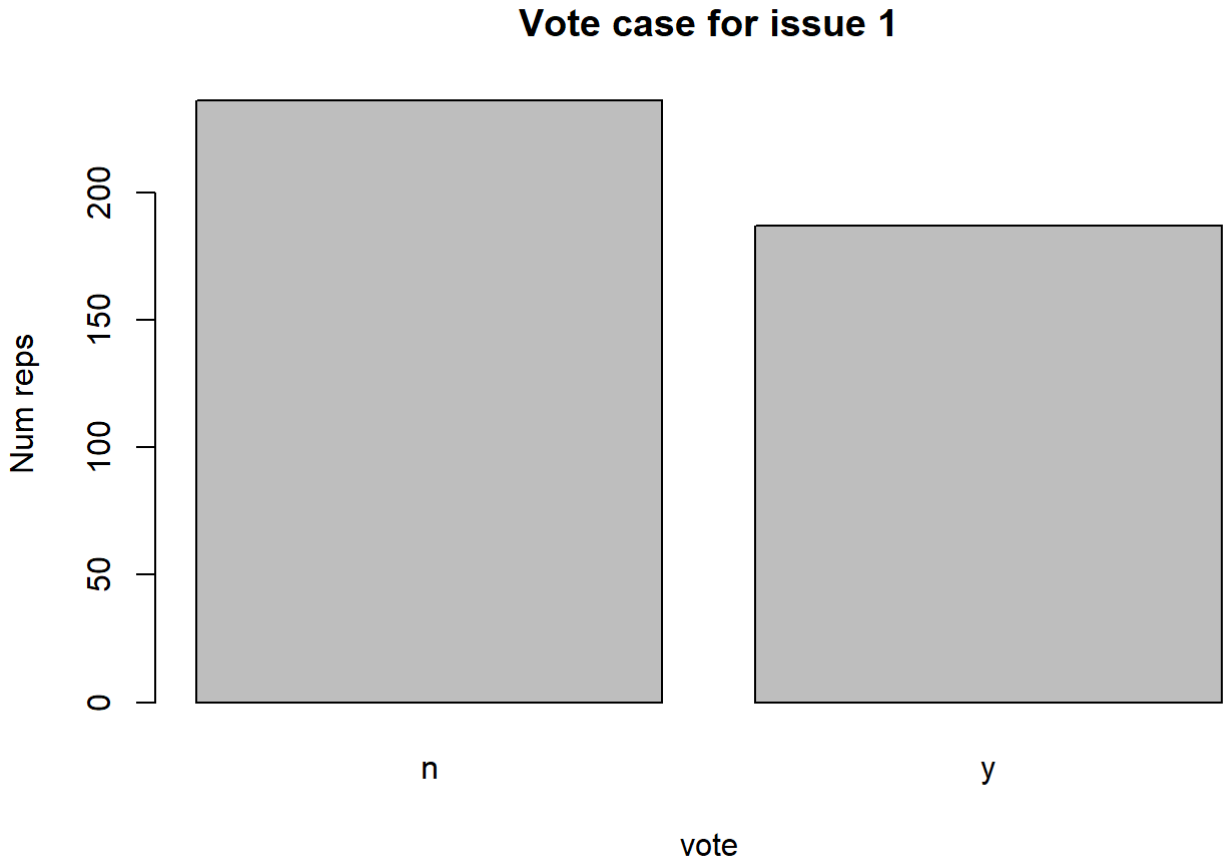
```
library(mlbench) #Data
library(e1071) #NB
```

#Getting a data from ml bench

```
data("HouseVotes84")
```

#Plot-EDA ##barplots for specific issue

```
plot(as.factor(HouseVotes84[,2]))
title(main='Vote case for issue 1',xlab="vote",ylab="Num reps")
```

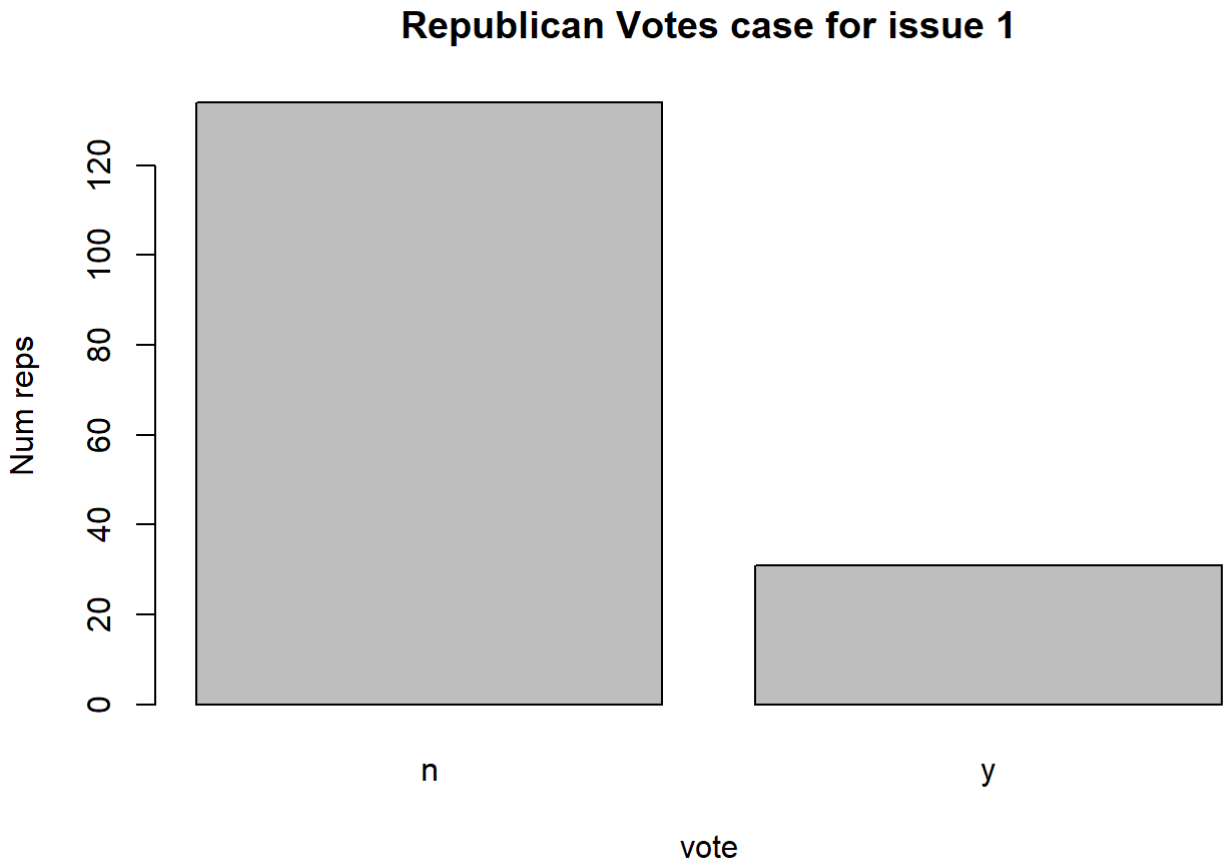


#byparty

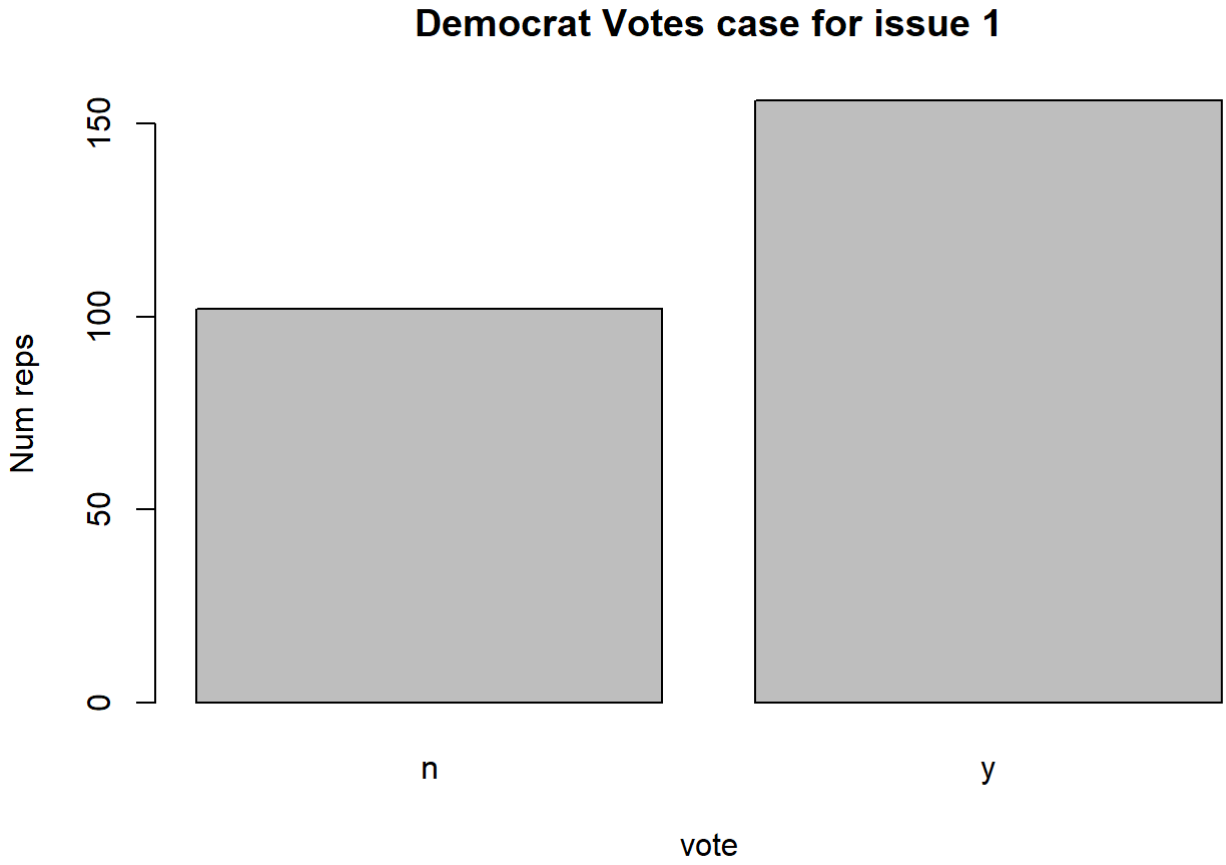
```
repub<-HouseVotes84$Class=="republican"
democrat<-HouseVotes84$Class=="democrat"
repub
```

```
##   [1]  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE  TRUE  TRUE
##  [13] FALSE FALSE  TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
##  [25] FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE  TRUE FALSE  TRUE
##  [37]  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [49] FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [61] FALSE  TRUE FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE
##  [73] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE  TRUE
##  [85]  TRUE FALSE  TRUE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
##  [97] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE
## [109] FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE  TRUE
## [121]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE
## [133]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE
## [145] FALSE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE  TRUE  TRUE
## [157]  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE  TRUE
## [169] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
## [181] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE
## [193] FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
## [205]  TRUE FALSE  TRUE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE
## [217] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE
## [229]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE  TRUE
## [241]  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE  TRUE  TRUE
## [253] FALSE  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
## [265] FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE
## [277]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE
## [289] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE
## [301]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE FALSE
## [313] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [325]  TRUE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE
## [337] FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE  TRUE FALSE  TRUE  TRUE  TRUE
## [349] FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE
## [361] FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
## [373] FALSE FALSE  TRUE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE
## [385] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE
## [397] FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE
## [409] FALSE  TRUE  TRUE FALSE  TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE
## [421]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE
## [433]  TRUE  TRUE  TRUE
```

```
plot(as.factor(HouseVotes84[repub,2]))
title(main='Republican Votes case for issue 1',xlab="vote",ylab="Num reps")
```




```
plot(as.factor(HouseVotes84[democrat,2]))
title(main='Democrat Votes case for issue 1',xlab="vote",ylab="Num reps")
```



Function to compute the conditional probability that a member of a party will cast a ‘yes’ vote for a particular issue. The probability is based on all members of the party who actually cast a vote on the issue

```
p_y_col_class <- function(col,cls){
  sum_y <- sum(HouseVotes84[,col]=="y" & HouseVotes84$Class==cls,na.rm =TRUE)
  sum_n <- sum(HouseVotes84[,col]=="n" & HouseVotes84$Class==cls,na.rm=TRUE)
  return (sum_y/(sum_y+sum_n))
}
```

#Building functions We have a lot of NA. We are going to impute the values. Functions needed for imputation. Function to return number of NAs by vote and class(democrat or republican)

```
na_by_class <- function(col,cls){return(sum(is.na(HouseVotes84[,col]) & HouseVotes84$Class==c
ls))}
na_by_class
```

```
## function(col,cls){return(sum(is.na(HouseVotes84[,col]) & HouseVotes84$Class==cls))}
```

Check the prob of yes vote by a democrat in issue 5

```
p_y_col_class(5,"democrat")
```

```
## [1] 0.05405405
```

Check the prob of yes vote by a republican in issue 5

```
p_y_col_class(5,"republican")
```

```
## [1] 0.9878788
```

Checking NA for democrat and Republican

```
na_by_class(5,"republican")
```

```
## [1] 3
```

```
na_by_class(5,"democrat")
```

```
## [1] 8
```

#Impute missing values If the republican congressman didn't vote, then we are allocating 'y' or 'n' based on if their party voted 'y' or 'n'

```
for (i in 2:ncol(HouseVotes84)){
  if(sum(is.na(HouseVotes84[,i])>0)){
    c1 <- which(is.na(HouseVotes84[,i])&HouseVotes84$Class=="democrat",arr.ind=TRUE)
    c2 <- which(is.na(HouseVotes84[,i])&HouseVotes84$Class=="republican",arr.ind=TRUE)
    HouseVotes84[c1,i] <-
      ifelse(runif(na_by_class(i,"democrat"))<p_y_col_class(i,"democrat"),"y","n")
    HouseVotes84[c2,i] <-
      ifelse(runif(na_by_class(i,"republican"))<p_y_col_class(i,"republican"),"y","n")
  }
}
```

Divide into test and training sets Create a new col "train" and assign 1 or 0 in 80/20 proportion via random uniform dist

```
HouseVotes84[,"train"] <- ifelse(runif(nrow(HouseVotes84))<0.80,1,0)
```

Get col number of train/test indicator column(needed later)

```
trainColNum <- grep("train",names(HouseVotes84))
```

Separate training and test sets and remove training column before modeling

```
trainHouseVotes84 <- HouseVotes84[HouseVotes84$train==1,-trainColNum]
testHouseVotes84 <- HouseVotes84[HouseVotes84$train==0,-trainColNum]
```

#Naive Bayes

```
nb_model <- naiveBayes(Class~.,data=trainHouseVotes84)
nb_model
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##   democrat republican
##  0.6160221  0.3839779
##
## Conditional probabilities:
##             V1
## Y                    n         y
##   democrat   0.3811659 0.6188341
##   republican 0.8201439 0.1798561
##
##             V2
## Y                    n         y
##   democrat   0.5201794 0.4798206
##   republican 0.5179856 0.4820144
##
##             V3
## Y                    n         y
##   democrat   0.1031390 0.8968610
##   republican 0.8633094 0.1366906
##
##             V4
## Y                     n          y
##   democrat   0.95515695 0.04484305
##   republican 0.00000000 1.00000000
##
##             V5
## Y                     n          y
##   democrat   0.78923767 0.21076233
##   republican 0.04316547 0.95683453
##
##             V6
## Y                     n          y
##   democrat   0.52017937 0.47982063
##   republican 0.09352518 0.90647482
##
##             V7
## Y                    n         y
##   democrat   0.2107623 0.7892377
##   republican 0.7769784 0.2230216
##
##             V8
## Y                    n         y
##   democrat   0.1704036 0.8295964
##   republican 0.8489209 0.1510791
##
##             V9
## Y                    n         y
##   democrat   0.2511211 0.7488789
##   republican 0.8992806 0.1007194
##
```

```
##             V10
## Y                    n         y
##   democrat   0.5291480 0.4708520
##   republican 0.4388489 0.5611511
##
##             V11
## Y                    n         y
##   democrat   0.4753363 0.5246637
##   republican 0.8561151 0.1438849
##
##             V12
## Y                    n         y
##   democrat   0.8565022 0.1434978
##   republican 0.1223022 0.8776978
##
##             V13
## Y                    n         y
##   democrat   0.7130045 0.2869955
##   republican 0.1223022 0.8776978
##
##             V14
## Y                     n          y
##   democrat   0.61434978 0.38565022
##   republican 0.02158273 0.97841727
##
##             V15
## Y                     n          y
##   democrat   0.34977578 0.65022422
##   republican 0.91366906 0.08633094
##
##             V16
## Y                     n          y
##   democrat   0.04932735 0.95067265
##   republican 0.34532374 0.65467626
```

Checking the summary

In this notation, the dependent variable(to be predicted) appears on the left hand side of the ~ and the independent is on the right side

```
summary(nb_model)
```

```
##           Length Class  Mode
## apriori    2     table  numeric
## tables    16     -none- list
## levels     2     -none- character
## isnumeric 16     -none- logical
## call       4     -none- call
```

```
str(nb_model)
```

```
## List of 5
##  $ apriori  : 'table' int [1:2(1d)] 223 139
##   ..- attr(*, "dimnames")=List of 1
##   .. ..$ Y: chr [1:2] "democrat" "republican"
##  $ tables   :List of 16
##   ..$ V1 : 'table' num [1:2, 1:2] 0.381 0.82 0.619 0.18
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ Y : chr [1:2] "democrat" "republican"
##   .. .. ..$ V1: chr [1:2] "n" "y"
##   ..$ V2 : 'table' num [1:2, 1:2] 0.52 0.518 0.48 0.482
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ Y : chr [1:2] "democrat" "republican"
##   .. .. ..$ V2: chr [1:2] "n" "y"
##   ..$ V3 : 'table' num [1:2, 1:2] 0.103 0.863 0.897 0.137
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ Y : chr [1:2] "democrat" "republican"
##   .. .. ..$ V3: chr [1:2] "n" "y"
##   ..$ V4 : 'table' num [1:2, 1:2] 0.9552 0 0.0448 1
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ Y : chr [1:2] "democrat" "republican"
##   .. .. ..$ V4: chr [1:2] "n" "y"
##   ..$ V5 : 'table' num [1:2, 1:2] 0.7892 0.0432 0.2108 0.9568
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ Y : chr [1:2] "democrat" "republican"
##   .. .. ..$ V5: chr [1:2] "n" "y"
##   ..$ V6 : 'table' num [1:2, 1:2] 0.5202 0.0935 0.4798 0.9065
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ Y : chr [1:2] "democrat" "republican"
##   .. .. ..$ V6: chr [1:2] "n" "y"
##   ..$ V7 : 'table' num [1:2, 1:2] 0.211 0.777 0.789 0.223
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ Y : chr [1:2] "democrat" "republican"
##   .. .. ..$ V7: chr [1:2] "n" "y"
##   ..$ V8 : 'table' num [1:2, 1:2] 0.17 0.849 0.83 0.151
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ Y : chr [1:2] "democrat" "republican"
##   .. .. ..$ V8: chr [1:2] "n" "y"
##   ..$ V9 : 'table' num [1:2, 1:2] 0.251 0.899 0.749 0.101
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ Y : chr [1:2] "democrat" "republican"
##   .. .. ..$ V9: chr [1:2] "n" "y"
##   ..$ V10: 'table' num [1:2, 1:2] 0.529 0.439 0.471 0.561
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ Y  : chr [1:2] "democrat" "republican"
##   .. .. ..$ V10: chr [1:2] "n" "y"
##   ..$ V11: 'table' num [1:2, 1:2] 0.475 0.856 0.525 0.144
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ Y  : chr [1:2] "democrat" "republican"
##   .. .. ..$ V11: chr [1:2] "n" "y"
##   ..$ V12: 'table' num [1:2, 1:2] 0.857 0.122 0.143 0.878
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ Y  : chr [1:2] "democrat" "republican"
##   .. .. ..$ V12: chr [1:2] "n" "y"
##   ..$ V13: 'table' num [1:2, 1:2] 0.713 0.122 0.287 0.878
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ Y  : chr [1:2] "democrat" "republican"
##   .. .. ..$ V13: chr [1:2] "n" "y"
```

```
##   ..$ V14: 'table' num [1:2, 1:2] 0.6143 0.0216 0.3857 0.9784
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ Y  : chr [1:2] "democrat" "republican"
##   .. .. ..$ V14: chr [1:2] "n" "y"
##   ..$ V15: 'table' num [1:2, 1:2] 0.3498 0.9137 0.6502 0.0863
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ Y  : chr [1:2] "democrat" "republican"
##   .. .. ..$ V15: chr [1:2] "n" "y"
##   ..$ V16: 'table' num [1:2, 1:2] 0.0493 0.3453 0.9507 0.6547
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ Y  : chr [1:2] "democrat" "republican"
##   .. .. ..$ V16: chr [1:2] "n" "y"
##  $ levels   : chr [1:2] "democrat" "republican"
##  $ isnumeric: Named logi [1:16] FALSE FALSE FALSE FALSE FALSE FALSE ...
##   ..- attr(*, "names")= chr [1:16] "V1" "V2" "V3" "V4" ...
##  $ call     : language naiveBayes.default(x = X, y = Y, laplace = laplace)
##  - attr(*, "class")= chr "naiveBayes"
```

#Lets test the model

```
nb_test_predict <- predict(nb_model,testHouseVotes84[,-1])
```

#Fraction of correct predictions

```
mean(nb_test_predict==testHouseVotes84$Class)
```

```
## [1] 0.890411
```

#Confusion matrix

```
table(pred=nb_test_predict,true=testHouseVotes84$Class)
```

```
##             true
## pred         democrat republican
##   democrat         38          2
##   republican        6         27
```

Function to create,run and record model results

```
nb_multiple_runs <- function(train_fraction,n){
  fraction_correct <- rep(NA,n)
  for (i in 1:n){
    HouseVotes84[,'train'] <- ifelse(runif(nrow(HouseVotes84))<train_fraction,1,0)
    trainColNum <- grep('train',names(HouseVotes84))
    trainHouseVotes84 <- HouseVotes84[HouseVotes84$train==1,-trainColNum]
    testHouseVotes84 <- HouseVotes84[HouseVotes84$train==0,-trainColNum]
    nb_model <- naiveBayes(Class~.,data = trainHouseVotes84)
    nb_test_predict <- predict(nb_model,testHouseVotes84[,-1])
    fraction_correct[i] <- mean(nb_test_predict==testHouseVotes84$Class)
  }
  return(fraction_correct)
}
```