Homework 2

Vivek Gopalan

2022-03-14

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
setwd("C:/ISE 5984")
library(data.table)

## Warning: package 'data.table' was built under R version 4.0.5

library(caret)

## Warning: package 'caret' was built under R version 4.0.5

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.0.5

## Loading required package: lattice

library(pROC)

## Warning: package 'pROC' was built under R version 4.0.5

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var

library(mlbench)

## Warning: package 'mlbench' was built under R version 4.0.5

library(tm)

## Warning: package 'tm' was built under R version 4.0.5
```

```
## Loading required package: NLP

##
## Attaching package: 'NLP'

## The following object is masked from 'package:ggplot2':
##
##     annotate

library(class)
library(ISLR)

## Warning: package 'ISLR' was built under R version 4.0.5

library(MASS)
library(rpart)
library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 4.0.5

library(Rcpp)

## Warning: package 'Rcpp' was built under R version 4.0.5

library(InformationValue)

## Warning: package 'InformationValue' was built under R version 4.0.5

##
## Attaching package: 'InformationValue'

## The following objects are masked from 'package:caret':
##
##     confusionMatrix, precision, sensitivity, specificity

library(ISLR)
library("partykit")

## Warning: package 'partykit' was built under R version 4.0.5

## Loading required package: grid

## Loading required package: libcoin

## Warning: package 'libcoin' was built under R version 4.0.5

## Loading required package: mvtnorm

## Warning: package 'mvtnorm' was built under R version 4.0.5

library(dplyr)

## Warning: package 'dplyr' was built under R version 4.0.5
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:MASS':
##
##     select

## The following objects are masked from 'package:data.table':
##
##     between, first, last

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(ROSE)

## Warning: package 'ROSE' was built under R version 4.0.5

## Loaded ROSE 0.0-4

library(car)

## Warning: package 'car' was built under R version 4.0.5

## Loading required package: carData

## Warning: package 'carData' was built under R version 4.0.5

##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
##
##     recode

library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.0.5

## -- Attaching packages --------------------------------------- tidyverse
1.3.1 --

## v tibble  3.1.6     v purrr   0.3.4
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## Warning: package 'tibble' was built under R version 4.0.5

## Warning: package 'tidyr' was built under R version 4.0.5
```

```
## Warning: package 'readr' was built under R version 4.0.5

## Warning: package 'purrr' was built under R version 4.0.5

## Warning: package 'stringr' was built under R version 4.0.5

## Warning: package 'forcats' was built under R version 4.0.5

## -- Conflicts ---------------------------------------
tidyverse_conflicts() --
## x NLP::annotate()    masks ggplot2::annotate()
## x dplyr::between()   masks data.table::between()
## x dplyr::filter()    masks stats::filter()
## x dplyr::first()     masks data.table::first()
## x dplyr::lag()       masks stats::lag()
## x dplyr::last()      masks data.table::last()
## x purrr::lift()      masks caret::lift()
## x car::recode()      masks dplyr::recode()
## x dplyr::select()    masks MASS::select()
## x purrr::some()      masks car::some()
## x purrr::transpose() masks data.table::transpose()

library(tinytex)

## Warning: package 'tinytex' was built under R version 4.0.5

library(lme4)

## Warning: package 'lme4' was built under R version 4.0.5

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

library(ggformula)

## Warning: package 'ggformula' was built under R version 4.0.5

## Loading required package: ggstance

## Warning: package 'ggstance' was built under R version 4.0.5

##
## Attaching package: 'ggstance'

## The following objects are masked from 'package:ggplot2':
##
##     geom_errorbarh, GeomErrorbarh
```

```
## Loading required package: scales

## Warning: package 'scales' was built under R version 4.0.5

##
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard

## The following object is masked from 'package:readr':
##
##     col_factor

## Loading required package: ggridges

## Warning: package 'ggridges' was built under R version 4.0.5

##
## New to ggformula?  Try the tutorials:
##  learnr::run_tutorial("introduction", package = "ggformula")
##  learnr::run_tutorial("refining", package = "ggformula")

library(mosaic)

## Warning: package 'mosaic' was built under R version 4.0.5

## Registered S3 method overwritten by 'mosaic':
##   method                           from
##   fortify.SpatialPolygonsDataFrame ggplot2

##
## The 'mosaic' package masks several functions from core packages in order
to add
## additional features.  The original behavior of these functions should not
be affected by this.

##
## Attaching package: 'mosaic'

## The following object is masked from 'package:scales':
##
##     rescale

## The following object is masked from 'package:lme4':
##
##     factorize

## The following object is masked from 'package:Matrix':
##
##     mean
```

```
## The following object is masked from 'package:purrr':
##
##     cross

## The following objects are masked from 'package:car':
##
##     deltaMethod, logit

## The following objects are masked from 'package:dplyr':
##
##     count, do, tally

## The following object is masked from 'package:tm':
##
##     inspect

## The following objects are masked from 'package:pROC':
##
##     cov, var

## The following object is masked from 'package:caret':
##
##     dotPlot

## The following object is masked from 'package:ggplot2':
##
##     stat

## The following objects are masked from 'package:stats':
##
##     binom.test, cor, cor.test, cov, fivenum, IQR, median, prop.test,
##     quantile, sd, t.test, var

## The following objects are masked from 'package:base':
##
##     max, mean, min, prod, range, sample, sum

library(Stat2Data)
library(emmeans)

## Warning: package 'emmeans' was built under R version 4.0.5

library(psych)

## Warning: package 'psych' was built under R version 4.0.5

##
## Attaching package: 'psych'

## The following objects are masked from 'package:mosaic':
##
##     logit, rescale
```

```
## The following objects are masked from 'package:scales':
##
##     alpha, rescale

## The following object is masked from 'package:car':
##
##     logit

## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha

library(ICC)
library(naivebayes)

## Warning: package 'naivebayes' was built under R version 4.0.5

## naivebayes 0.9.7 loaded

##
## Attaching package: 'naivebayes'

## The following object is masked from 'package:data.table':
##
##     tables

library("FNN")

## Warning: package 'FNN' was built under R version 4.0.5

##
## Attaching package: 'FNN'

## The following objects are masked from 'package:class':
##
##     knn, knn.cv

library(pROC)
```

1.: Apply the Naïve Bayes, LDA, and Multiple Logistic Regression to the Zip code data using only digits 2, 3, and 5. Compare the testing error.

```
train <- read.table(file.path(getwd(), "zip.train.gz"))
test <- read.table(file.path(getwd(), "zip.test.gz"))
head(train)

##   V1 V2 V3 V4     V5     V6     V7     V8     V9    V10    V11    V12
V13
## 1  6 -1 -1 -1 -1.000 -1.000 -1.000 -1.000 -0.631  0.862 -0.167 -1.000 -
1.000
```

```
## 2  5 -1 -1 -1 -0.813 -0.671 -0.809 -0.887 -0.671 -0.853 -1.000 -1.000 -
0.774
## 3  4 -1 -1 -1 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -0.996  0.147
1.000
## 4  7 -1 -1 -1 -1.000 -1.000 -0.273  0.684  0.960  0.450 -0.067 -0.679 -
1.000
## 5  3 -1 -1 -1 -1.000 -1.000 -0.928 -0.204  0.751  0.466  0.234 -0.809 -
1.000
## 6  6 -1 -1 -1 -1.000 -1.000 -0.397  0.983 -0.535 -1.000 -1.000 -1.000 -
1.000
##      V14    V15    V16 V17 V18 V19 V20    V21    V22    V23    V24    V25
## 1 -1.000 -1.000 -1.000  -1  -1  -1  -1 -1.000 -1.000 -1.000 -0.992  0.297
## 2 -0.180  0.052 -0.241  -1  -1  -1  -1  0.392  1.000  0.857  0.727  1.000
## 3 -0.189 -1.000 -1.000  -1  -1  -1  -1 -1.000 -1.000 -1.000 -1.000 -1.000
## 4 -1.000 -1.000 -1.000  -1  -1  -1  -1 -1.000 -0.114  0.974  0.917  0.734
## 5 -1.000 -1.000 -1.000  -1  -1  -1  -1 -1.000 -0.370  0.739  1.000  1.000
## 6 -1.000 -1.000 -1.000  -1  -1  -1  -1 -1.000 -1.000  0.692  0.536 -0.767
##      V26    V27    V28    V29    V30    V31    V32 V33 V34 V35 V36    V37
## 1  1.000  0.307 -1.000 -1.000 -1.000 -1.000 -1.000  -1  -1  -1  -1 -1.000
## 2  0.805  0.613  0.613  0.860  1.000  1.000  0.396  -1  -1  -1  -1 -0.548
## 3 -1.000 -0.882  1.000  0.390 -0.811 -1.000 -1.000  -1  -1  -1  -1 -1.000
## 4  0.994  1.000  0.973  0.391 -0.421 -0.976 -1.000  -1  -1  -1  -1 -0.323
## 5  1.000  1.000  0.644 -0.890 -1.000 -1.000 -1.000  -1  -1  -1  -1 -1.000
## 6 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000  -1  -1  -1  -1 -1.000
##      V38    V39    V40    V41    V42    V43    V44    V45 V46    V47
V48
## 1 -1.000 -1.000 -0.410  1.000  0.986 -0.565 -1.000 -1.000  -1 -1.000 -
1.000
## 2  1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000   1  1.000
0.875
## 3 -1.000 -1.000 -1.000 -1.000 -1.000 -0.715  1.000  0.029  -1 -1.000 -
1.000
## 4  0.991  0.622 -0.738 -1.000 -0.639  0.023  0.871  1.000   1 -0.432 -
1.000
## 5  0.616  1.000  0.688 -0.455 -0.731  0.659  1.000 -0.287  -1 -1.000 -
1.000
## 6 -0.921  0.928 -0.118 -1.000 -1.000 -1.000 -1.000 -1.000  -1 -1.000 -
1.000
##      V49 V50 V51 V52    V53    V54    V55    V56 V57    V58    V59    V60
## 1 -1.000  -1  -1  -1 -1.000 -1.000 -0.683  0.825   1  0.562 -1.000 -1.000
## 2 -0.957  -1  -1  -1 -0.786  0.961  1.000  1.000   1  0.727  0.403  0.403
## 3 -1.000  -1  -1  -1 -1.000 -0.888 -0.912 -1.000  -1 -1.000 -0.549  1.000
## 4 -1.000  -1  -1  -1  0.409  1.000  0.000 -1.000  -1 -1.000 -1.000 -0.842
## 5 -1.000  -1  -1  -1 -1.000 -0.376 -0.186 -0.874  -1 -1.000 -0.014  1.000
## 6 -1.000  -1  -1  -1 -1.000 -0.394  1.000 -0.596  -1 -1.000 -1.000 -1.000
##      V61    V62    V63   V64 V65 V66 V67    V68    V69    V70    V71
V72
## 1 -1.000 -1.000 -1.000 -1.00  -1  -1  -1 -1.000 -1.000 -0.938  0.540
1.000
## 2  0.171 -0.314 -0.314 -0.94  -1  -1  -1 -1.000 -0.298  1.000  1.000
```

```
1.000
## 3  0.361 -1.000 -1.000 -1.00  -1  -1  -1 -1.000 -0.938  0.694  0.057 -
1.000
## 4  0.714  1.000 -0.534 -1.00  -1  -1  -1 -0.879  0.965  1.000 -0.713 -
1.000
## 5 -0.253 -1.000 -1.000 -1.00  -1  -1  -1 -1.000 -1.000 -1.000 -1.000 -
1.000
## 6 -1.000 -1.000 -1.000 -1.00  -1  -1  -1 -1.000 -1.000  0.060  0.900 -
0.951
##      V73    V74    V75    V76    V77    V78    V79 V80 V81 V82 V83    V84
## 1  0.778 -0.715 -1.000 -1.000 -1.000 -1.000 -1.000  -1  -1  -1  -1 -1.000
## 2  0.440  0.056 -0.755 -1.000 -1.000 -1.000 -1.000  -1  -1  -1  -1 -1.000
## 3 -1.000 -1.000 -0.382  1.000  0.511 -1.000 -1.000  -1  -1  -1  -1 -1.000
## 4 -1.000 -1.000 -1.000 -0.606  0.977  0.695 -0.906  -1  -1  -1  -1 -0.528
## 5 -1.000 -0.978  0.501  1.000 -0.540 -1.000 -1.000  -1  -1  -1  -1 -1.000
## 6 -1.000 -1.000 -1.000 -0.647  0.455 -0.333 -1.000  -1  -1  -1  -1 -1.000
##      V85    V86    V87    V88    V89    V90    V91    V92    V93    V94
V95 V96
## 1 -1.000  0.100  1.000  0.922 -0.439 -1.000 -1.000 -1.000 -1.000 -1.000  -
1  -1
## 2  0.366  1.000  1.000  1.000  1.000  1.000  0.889 -0.081 -0.920 -1.000  -
1  -1
## 3 -0.311  1.000 -0.043 -1.000 -1.000 -1.000 -0.648  1.000  0.644 -1.000  -
1  -1
## 4  1.000  0.931 -0.888 -1.000 -1.000 -1.000 -0.949  0.559  0.984 -0.363  -
1  -1
## 5 -1.000 -1.000 -0.998 -0.341  0.296  0.371  1.000  0.417 -0.989 -1.000  -
1  -1
## 6 -1.000  0.259  0.676 -1.000 -1.000 -1.000 -0.984  0.677  0.981  0.551  -
1  -1
##   V97 V98 V99  V100   V101   V102   V103   V104   V105  V106   V107   V108
## 1  -1  -1  -1 -1.00 -0.257  0.950  1.000 -0.162 -1.000 -1.00 -1.000 -0.987
## 2  -1  -1  -1 -1.00 -0.396  0.886  0.974  0.851  0.851  0.95  1.000  1.000
## 3  -1  -1  -1 -1.00  0.489  1.000 -0.493 -1.000 -1.000 -1.00 -0.564  1.000
## 4  -1  -1  -1 -0.97 -0.266 -0.555 -1.000 -1.000 -1.000 -1.00 -0.186  1.000
## 5  -1  -1  -1 -1.00 -1.000 -1.000 -0.008  1.000  1.000  1.00  1.000  0.761
## 6  -1  -1  -1 -1.00 -0.994  0.699  0.305 -1.000 -1.000 -1.00 -0.499  1.000
##     V109   V110 V111 V112 V113 V114 V115   V116   V117   V118   V119
V120
## 1 -0.714 -0.832   -1   -1   -1   -1   -1 -0.797  0.909  1.000  0.300 -
0.961
## 2  0.539 -0.754   -1   -1   -1   -1   -1 -1.000 -1.000 -0.886 -0.505 -
1.000
## 3  0.693 -1.000   -1   -1   -1   -1   -1 -0.966  0.988  1.000 -0.893 -
1.000
## 4  0.488 -1.000   -1   -1   -1   -1   -1 -1.000 -1.000 -1.000 -1.000 -
1.000
## 5 -0.731 -1.000   -1   -1   -1   -1   -1 -1.000 -1.000 -1.000  0.242
1.000
## 6 -0.092  0.751   -1   -1   -1   -1   -1 -1.000 -0.923  0.966 -0.107 -
```

```
1.000
##   V121   V122   V123  V124   V125   V126   V127 V128 V129 V130 V131   V132
## 1   -1 -1.000 -0.550 0.485  0.996  0.867  0.092   -1   -1   -1   -1  0.278
## 2   -1 -0.649  0.405 1.000  1.000  0.653 -0.838   -1   -1   -1   -1 -1.000
## 3   -1 -1.000 -0.397 1.000  0.903 -0.977 -1.000   -1   -1   -1   -1 -0.559
## 4   -1 -1.000  0.697 0.992 -0.458 -1.000 -1.000   -1   -1   -1   -1 -1.000
## 5    1  0.319  0.259 1.000  0.742 -0.757 -1.000   -1   -1   -1   -1 -1.000
## 6   -1 -1.000 -0.300 0.854 -0.382  0.617 -1.000   -1   -1   -1   -1 -1.000
##     V133   V134   V135   V136   V137   V138   V139   V140   V141   V142
V143
## 1  1.000  0.877 -0.824 -1.000 -0.905  0.145  0.977  1.000  1.000  1.000
0.990
## 2 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -0.550  0.993  1.000
0.618
## 3  1.000  1.000 -0.297 -1.000 -1.000 -1.000 -0.611  1.000  0.873 -0.698 -
0.552
## 4 -1.000 -1.000 -1.000 -1.000 -1.000 -0.341  1.000  0.608 -1.000 -1.000 -
1.000
## 5 -1.000 -1.000 -0.975 -0.467 -0.989 -1.000 -1.000 -0.171  0.998  0.669 -
0.945
## 6 -0.409  1.000 -0.529 -1.000 -1.000 -1.000  0.048  0.614 -0.268  0.544 -
1.000
##     V144 V145  V146   V147   V148   V149   V150   V151   V152   V153
V154
## 1 -0.745   -1 -1.00 -0.950  0.847  1.000  0.327 -1.000 -1.000  0.355
1.000
## 2 -0.869   -1 -0.96 -0.512  0.134 -0.343 -0.796 -1.000 -1.000 -1.000 -
1.000
## 3 -1.000   -1 -1.00 -1.000 -0.126  1.000  1.000  0.766 -0.764 -1.000 -
1.000
## 4 -1.000   -1 -1.00 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000
0.471
## 5 -1.000   -1 -1.00 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -
1.000
## 6 -1.000   -1 -1.00 -1.000 -1.000  0.050  0.971 -0.839 -1.000 -1.000 -
1.000
##     V155   V156   V157   V158   V159   V160 V161   V162  V163   V164
V165
## 1  0.655 -0.109 -0.185  1.000  0.988 -0.723   -1 -1.000 -0.63  1.000
1.000
## 2 -1.000 -1.000 -0.432  0.994  1.000  0.223   -1  0.426  1.00  1.000
1.000
## 3 -0.577  1.000  0.933  0.484 -0.197 -1.000   -1 -1.000 -1.00 -0.818 -
0.355
## 4  0.998 -0.416 -1.000 -1.000 -1.000 -1.000   -1 -1.000 -1.00 -1.000 -
1.000
## 5 -1.000 -1.000  0.228  1.000  0.038 -1.000   -1 -1.000 -1.00 -1.000 -
1.000
## 6  0.172  0.526 -0.003  0.307 -1.000 -1.000   -1 -1.000 -1.00 -1.000
0.398
```

```
##     V166   V167   V168   V169   V170   V171   V172   V173   V174   V175
V176
## 1  0.068 -0.925  0.113  0.960  0.308 -0.884 -1.000 -0.075  1.000  0.641 -
0.995
## 2  0.214 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000  0.292  1.000
0.967
## 3  0.334  1.000  0.868 -0.289 -0.677 -0.596  1.000  1.000  1.000 -0.581 -
1.000
## 4 -1.000 -1.000 -1.000 -0.644  0.963  0.590 -0.999 -1.000 -1.000 -1.000 -
1.000
## 5 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -0.826  0.918  0.933 -
0.794
## 6  0.459 -1.000 -1.000 -1.000 -1.000  0.372  0.555  0.520 -0.045 -1.000 -
1.000
##    V177   V178   V179   V180   V181   V182   V183 V184   V185   V186
V187
## 1 -1.00 -1.000 -0.677  1.000  1.000  0.753  0.341    1  0.707 -0.942 -
1.000
## 2 -0.88  0.449  1.000  0.896 -0.094 -0.750 -1.000   -1 -1.000 -1.000 -
1.000
## 3 -1.00 -1.000 -1.000 -1.000 -1.000 -0.954  0.118    1  1.000  1.000
1.000
## 4 -1.00 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000   -1  0.061  1.000 -
0.079
## 5 -1.00 -1.000 -1.000 -0.666  0.337  0.224 -0.908   -1 -1.000 -1.000 -
1.000
## 6 -1.00 -1.000 -1.000 -1.000  0.671  0.176 -1.000   -1 -1.000 -1.000
0.236
##     V188   V189   V190   V191   V192   V193   V194   V195   V196   V197
V198
## 1 -1.000  0.545  1.000  0.027 -1.000 -1.000 -1.000 -0.903  0.792  1.000
1.000
## 2 -1.000 -1.000 -0.627  1.000  1.000  0.198 -0.105  1.000  1.000  1.000
0.639
## 3  1.000  0.973 -0.092 -0.995 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -
1.000
## 4 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -
1.000
## 5 -1.000 -1.000  0.418  1.000 -0.258 -1.000 -1.000 -0.246  1.000  1.000
0.355
## 6  0.934  0.971 -0.712 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000  0.763
0.084
##     V199   V200   V201   V202   V203   V204   V205   V206   V207   V208
V209
## 1  1.000  1.000  0.536  0.184  0.812  0.837  0.978  0.864 -0.630 -1.000 -
1.000
## 2 -0.168 -0.314 -0.446 -1.000 -1.000 -0.999 -0.337  0.147  0.996  1.000
0.667
## 3 -0.993 -0.464  0.046  0.290  0.457  1.000  0.721 -1.000 -1.000 -1.000 -
1.000
```

```
## 4 -1.000 -1.000  0.773  0.958 -0.714 -1.000 -1.000 -1.000 -1.000 -1.000 -
1.000
## 5 -0.958 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -0.077  1.000  0.344 -
1.000
## 6 -1.000 -1.000 -1.000 -1.000  0.073  1.000  0.265 -1.000 -1.000 -1.000 -
1.000
##     V210   V211   V212   V213   V214   V215   V216   V217   V218   V219
V220
## 1 -1.000 -1.000 -0.452  0.828  1.000  1.000  1.000  1.000  1.000  1.000
1.000
## 2 -0.808  0.065  0.993  1.000  1.000  1.000  1.000  0.996  0.970  0.970
0.970
## 3 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -0.426
1.000
## 4 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -0.545  0.989  0.432 -1.000 -
1.000
## 5 -1.000  0.075  1.000  1.000  0.649  0.256 -0.200 -0.351 -0.733 -0.733 -
0.733
## 6 -1.000 -1.000 -1.000  0.563  0.210 -1.000 -1.000 -0.930 -0.127  0.890
0.935
##     V221   V222 V223   V224   V225 V226   V227   V228   V229   V230   V231
## 1  1.000  0.135   -1 -1.000 -1.000   -1 -1.000 -1.000 -0.483  0.813  1.000
## 2  0.998  1.000    1  1.000  0.109   -1 -1.000 -0.830 -0.242  0.350  0.800
## 3  0.555 -1.000   -1 -1.000 -1.000   -1 -1.000 -1.000 -1.000 -1.000 -1.000
## 4 -1.000 -1.000   -1 -1.000 -1.000   -1 -1.000 -1.000 -1.000 -1.000 -1.000
## 5 -0.433  0.649    1  0.093 -1.000   -1 -0.959 -0.062  0.821  1.000  1.000
## 6 -0.845 -1.000   -1 -1.000 -1.000   -1 -1.000 -1.000  0.093  0.793 -0.205
##     V232   V233   V234   V235   V236   V237   V238   V239   V240  V241
V242
## 1  1.000  1.000  1.000  1.000  1.000  0.219 -0.943 -1.000 -1.000 -1.00   -
1
## 2  1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000  0.616 -0.93   -
1
## 3 -1.000 -1.000 -1.000  0.024  1.000  0.388 -1.000 -1.000 -1.000 -1.00   -
1
## 4 -0.348  1.000  0.798 -0.935 -1.000 -1.000 -1.000 -1.000 -1.000 -1.00   -
1
## 5  1.000  1.000  1.000  1.000  1.000  1.000  1.000  0.583 -0.843 -1.00   -
1
## 6  0.214  0.746  0.918  0.692  0.954 -0.882 -1.000 -1.000 -1.000 -1.00   -
1
##   V243 V244   V245   V246   V247   V248   V249   V250   V251   V252   V253
## 1   -1   -1 -1.000 -0.974 -0.429  0.304  0.823  1.000  0.482 -0.474 -0.991
## 2   -1   -1 -1.000 -1.000 -0.858 -0.671 -0.671 -0.033  0.761  0.762  0.126
## 3   -1   -1 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -0.109  1.000 -0.179
## 4   -1   -1 -1.000 -1.000 -1.000 -0.318  1.000  0.536 -0.987 -1.000 -1.000
## 5   -1   -1 -0.877 -0.326  0.174  0.466  0.639  1.000  1.000  0.791  0.439
## 6   -1   -1 -0.898  0.323  1.000  0.803  0.015 -0.862 -0.871 -0.437 -1.000
##     V254   V255   V256 V257
## 1 -1.000 -1.000 -1.000   -1
```

```
## 2 -0.095 -0.671 -0.828   -1
## 3 -1.000 -1.000 -1.000   -1
## 4 -1.000 -1.000 -1.000   -1
## 5 -0.199 -0.883 -1.000   -1
## 6 -1.000 -1.000 -1.000   -1
```

```
head(test)
```

```
##   V1 V2 V3 V4     V5   V6     V7     V8     V9    V10    V11    V12    V13
V14
## 1  9 -1 -1 -1 -1.000 -1.0 -0.948 -0.561  0.148  0.384  0.904  0.290 -0.782
-1
## 2  6 -1 -1 -1 -1.000 -1.0 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000
-1
## 3  3 -1 -1 -1 -0.593  0.7  1.000  1.000  1.000  1.000  0.853  0.075 -0.925
-1
## 4  6 -1 -1 -1 -1.000 -1.0 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000
-1
## 5  6 -1 -1 -1 -1.000 -1.0 -1.000 -1.000 -0.858 -0.106  0.802 -0.210 -1.000
-1
## 6  0 -1 -1 -1 -1.000 -1.0 -1.000  0.195  1.000  0.054 -1.000 -1.000 -1.000
-1
##   V15 V16 V17 V18 V19    V20    V21    V22    V23    V24    V25    V26
V27
## 1  -1  -1  -1  -1  -1 -1.000 -1.000 -0.748  0.588  1.000  1.000  0.991
0.915
## 2  -1  -1  -1  -1  -1 -1.000 -1.000 -1.000 -0.783 -0.973 -1.000 -1.000 -
1.000
## 3  -1  -1  -1  -1  -1 -0.553  0.998  1.000  1.000  1.000  1.000  1.000
1.000
## 4  -1  -1  -1  -1  -1 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -
1.000
## 5  -1  -1  -1  -1  -1 -1.000 -1.000 -1.000 -1.000 -0.854  0.597  1.000
0.798
## 6  -1  -1  -1  -1  -1 -1.000 -1.000 -1.000 -0.801  0.790  1.000  0.856 -
0.282
##      V28    V29    V30    V31 V32 V33 V34 V35    V36    V37    V38    V39
## 1  1.000  0.931 -0.476 -1.000  -1  -1  -1  -1 -1.000 -0.787  0.794  1.000
## 2 -1.000 -1.000 -1.000 -1.000  -1  -1  -1  -1 -1.000 -1.000 -0.364  0.789
## 3  1.000  0.961 -0.076 -0.999  -1  -1  -1  -1  0.228  1.000  0.849 -0.150
## 4 -1.000 -1.000 -1.000 -1.000  -1  -1  -1  -1 -1.000 -1.000 -1.000 -0.417
## 5 -0.388 -1.000 -1.000 -1.000  -1  -1  -1  -1 -1.000 -1.000 -1.000 -0.481
## 6 -0.831 -1.000 -1.000 -1.000  -1  -1  -1  -1 -1.000 -1.000 -0.937  0.498
##      V40    V41    V42    V43    V44    V45    V46    V47 V48 V49 V50 V51
## 1  0.727 -0.178 -0.693 -0.786 -0.624  0.834  0.756 -0.822  -1  -1  -1  -1
## 2 -0.371 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000  -1  -1  -1  -1
## 3 -0.705 -1.000 -0.850 -0.333 -0.072  0.929  1.000 -0.451  -1  -1  -1  -1
## 4 -0.330 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000  -1  -1  -1  -1
## 5  0.600  1.000  0.653 -0.780 -1.000 -1.000 -1.000 -1.000  -1  -1  -1  -1
## 6  1.000  0.975 -0.232  0.932  0.926 -0.069 -0.965 -1.000  -1  -1  -1  -1
```

```
##      V52    V53    V54    V55    V56    V57    V58    V59 V60    V61
V62
## 1 -0.922  0.810  1.000  0.010 -0.928 -1.000 -1.000 -1.000  -1 -0.390
1.000
## 2 -1.000 -0.467  0.963  0.609 -0.986 -1.000 -1.000 -1.000  -1 -1.000 -
1.000
## 3 -0.586  0.777 -0.524 -1.000 -1.000 -1.000 -1.000 -1.000  -1  0.344
1.000
## 4 -1.000 -0.883  0.447  0.999  0.775 -1.000 -1.000 -1.000  -1 -1.000 -
1.000
## 5 -1.000 -1.000 -0.386  0.913  1.000  0.658 -0.825 -1.000  -1 -1.000 -
1.000
## 6 -1.000 -1.000  0.070  1.000  1.000  0.446 -0.953 -0.385   1  1.000
0.276
##      V63    V64 V65 V66 V67    V68    V69    V70    V71    V72    V73 V74
## 1  0.271 -1.000  -1  -1  -1  0.012  1.000  0.248 -1.000 -1.000 -1.000  -1
## 2 -1.000 -1.000  -1  -1  -1 -0.875  0.605  0.960 -0.351 -1.000 -1.000  -1
## 3  0.544 -0.999  -1  -1  -1 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000  -1
## 4 -1.000 -1.000  -1  -1  -1 -0.589  0.803  1.000  0.602  0.257 -1.000  -1
## 5 -1.000 -1.000  -1  -1  -1 -1.000 -0.373  0.939  0.999  0.129 -0.835  -1
## 6 -0.895 -1.000  -1  -1  -1 -1.000 -0.423  0.960  1.000  0.762 -0.799  -1
##      V75    V76    V77 V78    V79    V80 V81 V82    V83    V84    V85
V86
## 1 -1.000 -0.402  0.326   1  0.801 -0.998  -1  -1 -0.981  0.645  1.000 -
0.687
## 2 -1.000 -1.000 -1.000  -1 -1.000 -1.000  -1  -1 -1.000  0.050  1.000
0.096
## 3 -1.000 -0.803  0.930   1  0.650 -0.999  -1  -1 -1.000 -1.000 -1.000 -
1.000
## 4 -1.000 -1.000 -1.000  -1 -1.000 -1.000  -1  -1 -0.588  0.862  0.981
0.150
## 5 -1.000 -1.000 -1.000  -1 -1.000 -1.000  -1  -1 -1.000 -0.340  0.986
1.000
## 6 -0.987 -0.238  0.941   1 -0.288 -1.000  -1  -1 -1.000 -0.989  0.321
1.000
##      V87    V88 V89 V90    V91    V92    V93    V94    V95    V96    V97
V98
## 1 -1.000 -1.000  -1  -1 -0.792  0.976  1.000  1.000  0.413 -0.976 -1.000 -
1.000
## 2 -1.000 -1.000  -1  -1 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -
1.000
## 3 -1.000 -1.000  -1  -1 -0.579  0.821  1.000  1.000 -0.131 -1.000 -1.000 -
1.000
## 4 -0.980 -1.000  -1  -1 -1.000 -1.000 -0.969 -0.437 -0.295 -0.295 -0.939 -
0.651
## 5  0.007 -1.000  -1  -1 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -
1.000
## 6  0.924 -0.512  -1  -1 -1.000 -1.000 -0.015  1.000  0.906 -0.758 -1.000 -
1.000
##      V99   V100   V101   V102   V103 V104   V105   V106   V107   V108
```

```
V109
## 1 -0.993  0.834  0.897 -0.951 -1.000   -1 -1.000 -0.831  0.140  1.000
1.000
## 2 -0.582  0.970  0.532 -0.922 -1.000   -1 -1.000 -1.000 -1.000 -1.000 -
0.602
## 3 -1.000 -1.000 -1.000 -1.000 -1.000   -1 -0.621  0.156  0.934  1.000
1.000
## 4  0.863  0.965 -0.219 -1.000 -1.000   -1 -1.000 -1.000 -0.946 -0.592
0.488
## 5 -0.741  0.763  1.000  0.119 -0.986   -1 -1.000 -1.000 -1.000 -1.000 -
1.000
## 6 -1.000 -0.530  0.992  1.000 -0.055   -1 -1.000 -1.000 -1.000 -1.000 -
0.651
##     V110   V111   V112   V113   V114   V115   V116   V117   V118   V119
V120
## 1  0.302 -0.889 -1.000 -1.000 -1.000 -1.000  0.356  0.794 -0.836 -1.000 -
0.445
## 2  0.307  0.718  0.718 -0.373 -0.998  0.723  1.000 -0.431 -1.000 -1.000 -
1.000
## 3  0.575 -0.933 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -0.952 -0.176
0.602
## 4  1.000  1.000  1.000  0.251  0.338  1.000  0.054 -1.000 -1.000 -1.000 -
1.000
## 5 -1.000 -1.000 -1.000 -1.000 -1.000  0.271  1.000  0.879 -0.871 -1.000 -
1.000
## 6  0.810  1.000  0.206 -1.000 -1.000 -1.000  0.424  1.000  0.868 -0.717 -
1.000
##     V121   V122   V123   V124   V125   V126   V127   V128   V129   V130
V131
## 1  0.074  0.833  1.000  1.000  0.696 -0.881 -1.000 -1.000 -1.000 -1.000 -
1.000
## 2 -1.000 -1.000 -0.817 -0.136  0.808  1.000  1.000  1.000  0.697 -0.670
0.965
## 3  1.000  1.000  1.000  0.952 -0.093 -1.000 -1.000 -1.000 -1.000 -1.000 -
1.000
## 4 -1.000 -0.698  0.628  1.000  1.000  0.673  0.059  0.890  0.995  0.995
0.878
## 5 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -0.620
0.933
## 6 -1.000 -1.000 -1.000 -1.000 -0.989  0.721  1.000 -0.125 -1.000 -1.000 -
0.944
##     V132   V133  V134 V135 V136 V137   V138   V139   V140   V141   V142
V143
## 1 -0.368  0.955  1.00    1    1    1  0.905  1.000  1.000 -0.262 -1.000 -
1.000
## 2  0.659 -1.000 -1.00   -1   -1   -1 -0.512  0.738  1.000  0.839 -0.336 -
0.977
## 3 -1.000 -1.000 -0.41    1    1    1  1.000  1.000  0.792 -0.715 -1.000 -
1.000
## 4 -0.814 -1.000 -1.00   -1   -1   -1  0.388  1.000  0.711 -0.501 -0.926 -
```

```
0.672
## 5  0.983 -0.349 -1.00   -1   -1   -1 -1.000 -1.000 -0.930 -0.356  0.552
0.647
## 6  0.867  1.000  0.00   -1   -1   -1 -1.000 -1.000 -1.000 -1.000  0.579
1.000
##     V144   V145   V146   V147   V148   V149   V150   V151   V152   V153
V154
## 1 -1.000 -1.000 -1.000 -1.000 -1.000 -0.507  0.451  0.692  0.692 -0.007 -
0.237
## 2  0.433  0.878  0.161  1.000 -0.102 -1.000 -1.000 -1.000 -1.000 -0.643
0.870
## 3 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -0.345  0.333 -0.050 -
0.333
## 4  0.963  0.473  0.989  0.775 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000
0.677
## 5  0.618 -0.633  0.148  1.000  0.549 -1.000 -1.000 -1.000 -1.000 -1.000 -
0.853
## 6  0.344 -1.000 -1.000 -0.111  0.996  0.978 -0.604 -1.000 -1.000 -1.000 -
1.000
##     V155   V156   V157   V158   V159   V160   V161   V162   V163   V164
V165
## 1  1.000  0.882 -0.795 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -
1.000
## 2  0.970  0.264 -0.971 -1.000 -0.399  1.000  0.117  0.835  0.968 -0.701 -
1.000
## 3 -0.172  0.606  0.950 -0.101 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -
1.000
## 4  1.000 -0.654 -1.000 -1.000  0.122  1.000 -0.357  0.970  0.922 -0.506 -
0.994
## 5 -0.172  0.666  1.000  1.000  1.000  1.000  0.821  0.861  1.000  0.086 -
1.000
## 6 -1.000 -1.000 -1.000  0.111  1.000  0.013 -1.000 -1.000  0.406  1.000
0.988
##     V166 V167 V168   V169   V170   V171   V172   V173   V174   V175   V176
## 1 -1.000   -1   -1 -1.000  0.155  1.000  0.436 -1.000 -1.000 -1.000 -1.000
## 2 -1.000   -1   -1  0.198  1.000  0.052 -1.000 -1.000 -0.291  0.876  0.790
## 3 -1.000   -1   -1 -1.000 -1.000 -1.000 -0.838  0.548  1.000 -0.266 -1.000
## 4 -1.000   -1   -1 -1.000  0.440  0.262 -0.996 -0.985 -0.323  0.986  0.668
## 5 -1.000   -1   -1 -0.280  0.878  1.000  1.000  0.806 -0.097 -0.191  1.000
## 6 -0.861   -1   -1 -1.000 -1.000 -1.000 -1.000 -1.000  0.253  1.000  0.460
##     V177   V178   V179   V180   V181   V182   V183   V184   V185   V186
V187
## 1 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -0.991  0.703
1.000
## 2 -0.819  0.392  0.962 -0.461 -1.000 -1.000 -1.000 -0.948  0.820  1.000 -
0.168
## 3 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -
1.000
## 4 -0.954  0.053  1.000  1.000  0.499  0.038  0.179 -0.215 -0.215 -0.215 -
0.215
```

```
## 5  0.695  0.909  1.000 -0.666 -1.000 -1.000 -1.000 -0.316  0.999  1.000
0.712
## 6 -1.000 -1.000  0.148  1.000  0.811 -0.888 -1.000 -1.000 -1.000 -1.000 -
1.000
##     V188   V189   V190   V191   V192   V193   V194   V195   V196   V197
V198
## 1 -0.025 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -
1.000
## 2 -0.475  0.280  0.968  0.880 -0.613 -1.000 -0.551  0.854  0.980  0.498
0.324
## 3 -1.000 -0.285  1.000  0.729 -1.000 -1.000 -1.000 -1.000 -0.611 -0.944 -
1.000
## 4  0.184  0.637  1.000  0.443 -0.741 -1.000 -0.979  0.189  0.942  1.000
1.000
## 5 -0.530 -0.995 -0.859  0.569  1.000 -0.267  0.955  1.000 -0.552 -1.000 -
1.000
## 6 -1.000 -0.872  0.456  1.000  0.171 -1.000 -1.000 -0.465  0.913  1.000
0.281
##     V199   V200   V201   V202   V203   V204   V205   V206   V207   V208
V209
## 1 -1.000 -1.000 -0.833  0.959  1.000 -0.629 -1.000 -1.000 -1.000 -1.000 -
1.00
## 2  0.324  0.328  0.998  1.000  0.970  0.995  0.976  0.250 -0.642 -1.000 -
1.00
## 3 -1.000 -1.000 -1.000 -1.000 -1.000 -0.556  0.943  1.000  0.779 -0.943 -
1.00
## 4  1.000  1.000  1.000  1.000  1.000  1.000  0.896  0.177 -0.911 -1.000 -
1.00
## 5 -1.000  0.646  1.000  0.317 -0.926 -1.000 -0.849  0.598  1.000  0.169 -
0.97
## 6 -0.754 -1.000 -1.000 -1.000 -0.740 -0.436  0.657  1.000  1.000  0.008 -
1.00
##     V210   V211   V212   V213   V214   V215   V216   V217   V218   V219
V220
## 1 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -0.600  0.998  0.841 -
0.932
## 2 -1.000 -0.640  0.661  0.971  1.000  1.000  1.000  0.950  0.774  0.774
0.302
## 3 -1.000 -0.943  0.779  0.555 -0.333 -0.333 -0.333 -0.166  0.389  1.000
1.000
## 4 -1.000 -1.000 -0.723 -0.451 -0.081 -0.611 -0.021 -0.414 -0.021 -0.182 -
0.648
## 5  0.631  1.000  0.754  0.046 -0.244 -0.661  0.984  1.000  0.142 -0.584
0.075
## 6 -1.000 -1.000 -0.006  0.976  1.000  0.868  0.744  0.744  0.744  0.850
1.000
##     V221 V222   V223   V224 V225   V226   V227   V228   V229   V230   V231
## 1 -1.000   -1 -1.000 -1.000   -1 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000
## 2 -0.522   -1 -1.000 -1.000   -1 -1.000 -1.000 -1.000 -0.663 -0.606 -0.606
## 3  1.000    1  0.497 -1.000   -1 -1.000 -1.000  0.507  1.000  1.000  1.000
```

```
## 4 -0.780   -1 -1.000 -1.000   -1 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000
## 5  0.833    1  0.123 -0.963   -1 -0.537  0.896  1.000  1.000  1.000  1.000
## 6  1.000    1  0.782 -0.736   -1 -1.000 -1.000 -1.000 -0.310  0.686  1.000
##     V232   V233 V234   V235 V236  V237   V238   V239 V240 V241 V242   V243
## 1 -1.000 -0.424    1  0.732   -1 -1.00 -1.000 -1.000   -1   -1   -1 -1.000
## 2 -0.606 -0.688   -1 -1.000   -1 -1.00 -1.000 -1.000   -1   -1   -1 -1.000
## 3  1.000  1.000    1  1.000    1  0.83  0.053 -0.946   -1   -1   -1 -1.000
## 4 -1.000 -1.000   -1 -1.000   -1 -1.00 -1.000 -1.000   -1   -1   -1 -1.000
## 5  1.000  1.000    1  1.000    1  0.83 -0.387 -0.976   -1   -1   -1 -0.697
## 6  1.000  1.000    1  1.000    1  1.00  0.602 -0.906   -1   -1   -1 -1.000
##     V244   V245   V246   V247   V248   V249   V250   V251   V252   V253
V254
## 1 -1.000 -1.000 -1.000 -1.000 -1.000 -0.908  0.430  0.622 -0.973 -1.000 -
1.000
## 2 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -
1.000
## 3 -0.941  0.059  0.615  1.000  1.000  0.717  0.333  0.162 -0.393 -1.000 -
1.000
## 4 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000 -
1.000
## 5 -0.108  0.312  0.901  0.901  0.901  0.901  0.901  0.290 -0.369 -0.867 -
1.000
## 6 -1.000 -1.000 -0.903  0.009  0.224  1.000  0.988  0.187  0.139 -0.641 -
0.812
##   V255 V256 V257
## 1   -1   -1   -1
## 2   -1   -1   -1
## 3   -1   -1   -1
## 4   -1   -1   -1
## 5   -1   -1   -1
## 6   -1   -1   -1
```

```
## Filtering data set to 2's, 3's, and 5's
train <- train[train[,1] %in% c(2, 3,5),]
test <- test[test[,1] %in% c(2, 3,5),]
# Reducing models to key variables
pixel <- c("V1", "V3", "V5", "V7", "V15")
train <- train[,pixel]
test <- test[,pixel]



train$V1 <- as.numeric(train$V1)
train$V3 <- as.numeric(train$V3)
train$V5 <- as.numeric(train$V5)
train$V7 <- as.numeric(train$V7)
train$V15<- as.numeric(train$V15)

cor(train[c("V1", "V3", "V5", "V7", "V15")])
```

```
##              V1          V3           V5          V7          V15
## V1   1.00000000 -0.08705182 -0.137484360 -0.27046609  0.431486903
## V3  -0.08705182  1.00000000  0.379312157  0.09936139 -0.017752337
## V5  -0.13748436  0.37931216  1.000000000  0.49781470  0.005154581
## V7  -0.27046609  0.09936139  0.497814705  1.00000000 -0.105463820
## V15  0.43148690 -0.01775234  0.005154581 -0.10546382  1.000000000
```

```
# Building Naive Bayes Model
TrainY = train[,1]
TrainY = as.matrix(TrainY)
TrainY = factor(TrainY)
TrainX = train[2:5]
head(TrainX)
```

```
##    V3     V5     V7    V15
## 2  -1 -0.813 -0.809  0.052
## 5  -1 -1.000 -0.928 -1.000
## 7  -1 -0.830  1.000 -1.000
## 27 -1 -1.000 -0.104 -1.000
## 31 -1 -1.000 -1.000 -1.000
## 36 -1 -1.000  0.492 -1.000
```

```
bayes.model.train <- naive_bayes(TrainY~., data=train[,pixel])
bayes.model.train
```

```
##
## ================================== Naive Bayes
==================================
##
##  Call:
## naive_bayes.formula(formula = TrainY ~ ., data = train[, pixel])
##
## ---------------------------------------------------------------------------
-------
##
## Laplace smoothing: 0
##
## ---------------------------------------------------------------------------
-------
##
##  A priori probabilities:
##
##         2         3         5
## 0.3758355 0.3383033 0.2858612
##
## ---------------------------------------------------------------------------
-------
##
##  Tables:
##
## ---------------------------------------------------------------------------
```

```
-------
##  ::: V1 (Gaussian)
## ------------------------------------------------------------------------
-------
##
## V1     2 3 5
##   mean 2 3 5
##   sd   0 0 0
##
## ------------------------------------------------------------------------
-------
##  ::: V3 (Gaussian)
## ------------------------------------------------------------------------
-------
##
## V3               2           3           5
##   mean -0.96039124 -0.98044225 -0.99509892
##   sd    0.21586006  0.13240860  0.05479478
##
## ------------------------------------------------------------------------
-------
##  ::: V5 (Gaussian)
## ------------------------------------------------------------------------
-------
##
## V5              2          3          5
##   mean -0.7993461 -0.7305897 -0.9338058
##   sd    0.4652299  0.5079160  0.2659924
##
## ------------------------------------------------------------------------
-------
##  ::: V7 (Gaussian)
## ------------------------------------------------------------------------
-------
##
## V7              2          3          5
##   mean -0.3736717 -0.1031353 -0.8024029
##   sd    0.7374306  0.7489323  0.4490188
##
## ------------------------------------------------------------------------
-------
##  ::: V15 (Gaussian)
## ------------------------------------------------------------------------
-------
##
## V15              2           3           5
##   mean -0.99729275 -0.97465805 -0.59708993
##   sd    0.04194251  0.14583086  0.63029931
##
```

```
## ------------------------------------------------------------------------
-------

summary(bayes.model.train)

##
## ================================== Naive Bayes
==================================
##
## - Call: naive_bayes.formula(formula = TrainY ~ ., data = train[, pixel])
## - Laplace: 0
## - Classes: 3
## - Samples: 1945
## - Features: 5
## - Conditional distributions:
##     - Gaussian: 5
## - Prior probabilities:
##     - 2: 0.3758
##     - 3: 0.3383
##     - 5: 0.2859
##
## ------------------------------------------------------------------------
-------

predBayes <- predict(bayes.model.train, test)
summary(predBayes)

##   2   3   5
## 203 162 159

TestY = test[,1]
TestY = as.matrix(TestY)
TestY = factor(TestY)
TestX = test[2:5]
head(TestX)

##        V3     V5     V7 V15
## 3  -1.000 -0.593  1.000  -1
## 12  0.572  0.063 -0.847  -1
## 13 -1.000  0.413  1.000  -1
## 16 -1.000  0.264 -0.210  -1
## 21 -1.000 -1.000 -1.000  -1
## 22 -1.000 -0.831  0.140  -1

bayes.model.test <- naive_bayes(TestY~., data=test[,pixel])
bayes.model.test

##
## ================================== Naive Bayes
==================================
##
##  Call:
```

```
## naive_bayes.formula(formula = TestY ~ ., data = test[, pixel])
##
## ---------------------------------------------------------------------------
-------
##
## Laplace smoothing: 0
##
## ---------------------------------------------------------------------------
-------
##
##  A priori probabilities:
##
##         2         3         5
## 0.3778626 0.3167939 0.3053435
##
## ---------------------------------------------------------------------------
-------
##
##  Tables:
##
## ---------------------------------------------------------------------------
-------
##  ::: V1 (Gaussian)
## ---------------------------------------------------------------------------
-------
##
## V1     2 3 5
##   mean 2 3 5
##   sd   0 0 0
##
## ---------------------------------------------------------------------------
-------
##  ::: V3 (Gaussian)
## ---------------------------------------------------------------------------
-------
##
## V3              2          3          5
##   mean -0.9503838 -0.9611627 -0.9868750
##   sd    0.2558233  0.2393437  0.1008493
##
## ---------------------------------------------------------------------------
-------
##  ::: V5 (Gaussian)
## ---------------------------------------------------------------------------
-------
##
## V5              2          3          5
##   mean -0.8052374 -0.6965482 -0.9583687
##   sd    0.4523138  0.5393461  0.1704971
##
```

```
## ------------------------------------------------------------------------
-------
##  ::: V7 (Gaussian)
## ------------------------------------------------------------------------
-------
##
## V7               2           3           5
##   mean -0.36917172 -0.03910241 -0.76425000
##   sd    0.73638758  0.78476763  0.49284619
##
## ------------------------------------------------------------------------
-------
##  ::: V15 (Gaussian)
## ------------------------------------------------------------------------
-------
##
## V15             2          3          5
##   mean -0.9898788 -0.9705301 -0.5616562
##   sd    0.1411382  0.1840315  0.6846652
##
## ------------------------------------------------------------------------
-------
```

```
summary(bayes.model.test)
```

```
##
## ================================= Naive Bayes
==================================
##
## - Call: naive_bayes.formula(formula = TestY ~ ., data = test[, pixel])
## - Laplace: 0
## - Classes: 3
## - Samples: 524
## - Features: 5
## - Conditional distributions:
##     - Gaussian: 5
## - Prior probabilities:
##     - 2: 0.3779
##     - 3: 0.3168
##     - 5: 0.3053
##
## ------------------------------------------------------------------------
-------
```

```
# Conducting LDA analysis on train and test data
mylDA <- lda(formula =  V1 ~.,data=train)
mylDA
```

```
## Call:
## lda(V1 ~ ., data = train)
##
```

```
## Prior probabilities of groups:
##         2         3         5
## 0.3758355 0.3383033 0.2858612
##
## Group means:
##           V3         V5         V7        V15
## 2 -0.9603912 -0.7993461 -0.3736717 -0.9972927
## 3 -0.9804422 -0.7305897 -0.1031353 -0.9746581
## 5 -0.9950989 -0.9338058 -0.8024029 -0.5970899
##
## Coefficients of linear discriminants:
##             LD1        LD2
## V3  -0.28206247 -3.0687501
## V5  -0.09066664  0.1476385
## V7  -0.92898907  1.0518466
## V15  2.29820633  1.4903142
##
## Proportion of trace:
##    LD1    LD2
## 0.9395 0.0605

summary(mylDA)

##         Length Class  Mode
## prior    3     -none- numeric
## counts   3     -none- numeric
## means   12     -none- numeric
## scaling  8     -none- numeric
## lev      3     -none- character
## svd      2     -none- numeric
## N        1     -none- numeric
## call     3     -none- call
## terms    3     terms  call
## xlevels  0     -none- list

# Predicting LDA training model on test data
predLDA <- predict(mylDA, test)
predLDA

## $class
##   [1] 3 2 3 3 2 3 2 2 3 5 2 2 5 2 2 3 3 2 3 3 3 2 2 2 3 3 3 2 3 3 2 5 2 2
2 2 2
##  [38] 2 2 3 3 2 3 2 3 2 3 2 3 3 2 2 2 3 2 3 3 5 2 2 2 2 2 2 3 2 2 2 2 3 2
2 3 5
##  [75] 5 2 3 5 2 2 3 2 3 2 2 2 3 3 3 2 2 2 2 2 5 2 3 3 5 2 2 2 2 2 2 2 3 3
3 3 2
## [112] 3 2 3 3 3 2 3 2 3 2 3 3 2 2 2 5 2 2 3 2 2 2 3 2 2 3 3 3 2 3 3 2 2 5
2 2 2
## [149] 2 3 3 2 2 2 2 2 5 3 2 2 3 2 2 2 3 2 3 3 3 2 3 3 2 3 2 3 3 2 3 3 2 2
2 2 2
## [186] 3 5 2 2 2 2 3 2 2 5 2 2 2 2 2 2 2 3 2 2 2 2 2 2 3 2 3 2 2 3 3 2 3
```

```
2 2 2
## [223] 3 2 2 2 2 3 2 3 2 2 2 5 3 5 2 2 2 2 2 2 2 2 2 5 3 5 2 3 2 2 3 2 5 5
2 3 2
## [260] 2 5 3 2 2 2 3 3 2 2 3 2 2 2 3 2 2 5 3 2 3 2 3 3 2 2 2 2 3 3 2 5 5 2
2 5 2
## [297] 5 2 3 2 2 3 2 2 3 3 2 2 3 3 3 2 3 5 2 2 3 2 2 2 2 2 5 2 2 3 3 2 2 2
2 2 2
## [334] 3 2 2 2 2 2 2 5 5 3 5 3 2 5 5 2 3 5 3 3 5 3 2 5 2 2 2 2 3 3 5 2 5 2 2
3 5 5
## [371] 3 3 3 2 2 2 2 3 3 2 2 2 2 3 2 2 3 2 3 3 3 2 2 2 3 3 3 3 2 2 2 5 2 5
5 5 3
## [408] 2 3 3 2 2 3 5 2 3 2 2 2 3 3 5 2 2 2 5 5 2 2 3 2 2 2 3 3 2 3 2 3 2 2
2 2 2
## [445] 2 2 2 3 2 3 3 2 3 3 2 2 2 2 2 2 2 2 3 5 3 3 3 2 2 3 2 2 2 2 3 2 2 2
3 2 5
## [482] 5 3 2 2 5 5 2 2 5 3 3 2 2 2 2 5 2 2 2 5 5 2 3 2 2 2 5 2 2 2 2 2 2 5
5 2 2
## [519] 5 2 3 5 3 2
## Levels: 2 3 5
##
## $posterior
##                2           3          5
## 3    0.328187385 0.646369164 0.02544345
## 12   0.834447057 0.095674682 0.06987826
## 13   0.312854945 0.664927873 0.02221718
## 16   0.441825664 0.451488982 0.10668535
## 21   0.459908972 0.266728075 0.27336295
## 22   0.426280433 0.493934885 0.07978468
## 26   0.459908972 0.266728075 0.27336295
## 28   0.410250773 0.273733503 0.31601572
## 30   0.350289191 0.617442489 0.03226832
## 33   0.007521920 0.013293078 0.97918500
## 37   0.459908972 0.266728075 0.27336295
## 49   0.877380340 0.057629973 0.06498969
## 52   0.004589553 0.006049298 0.98936115
## 54   0.468661578 0.341079398 0.19025902
## 57   0.459908972 0.266728075 0.27336295
## 59   0.333286844 0.638306906 0.02840625
## 64   0.314718939 0.662690439 0.02259062
## 68   0.459908972 0.266728075 0.27336295
## 69   0.403598787 0.536029764 0.06037145
## 72   0.387638552 0.560490716 0.05187073
## 74   0.417735772 0.511472343 0.07079189
## 79   0.459908972 0.266728075 0.27336295
## 81   0.459908972 0.266728075 0.27336295
## 84   0.459908972 0.266728075 0.27336295
## 89   0.327896150 0.646725072 0.02537878
## 95   0.374279524 0.583435128 0.04228535
## 104  0.392128890 0.555919439 0.05195167
## 107  0.466744396 0.366388368 0.16686724
```

```
## 111  0.409860373 0.525646810 0.06449282
## 112  0.312976080 0.664782621 0.02224130
## 117  0.459908972 0.266728075 0.27336295
## 122  0.118739778 0.075094364 0.80616586
## 123  0.459908972 0.266728075 0.27336295
## 127  0.459908972 0.266728075 0.27336295
## 132  0.459908972 0.266728075 0.27336295
## 144  0.459908972 0.266728075 0.27336295
## 145  0.459908972 0.266728075 0.27336295
## 146  0.459908972 0.266728075 0.27336295
## 148  0.459908972 0.266728075 0.27336295
## 152  0.316115257 0.661011075 0.02287367
## 153  0.313530032 0.663838773 0.02263120
## 161  0.460471195 0.302415249 0.23711356
## 166  0.419678178 0.507763011 0.07255881
## 169  0.450342806 0.436793858 0.11286334
## 170  0.444348304 0.453371313 0.10228038
## 196  0.459908972 0.266728075 0.27336295
## 198  0.379599535 0.574987005 0.04541346
## 201  0.463419217 0.281047804 0.25553298
## 205  0.334438408 0.638696533 0.02686506
## 212  0.395830577 0.549912350 0.05425707
## 215  0.461937780 0.274530597 0.26353162
## 225  0.463171030 0.279894781 0.25693419
## 226  0.459908972 0.266728075 0.27336295
## 230  0.408505104 0.528356102 0.06313879
## 231  0.469951460 0.464363434 0.06568511
## 232  0.373125952 0.582404766 0.04446928
## 235  0.380895888 0.573491749 0.04561236
## 240  0.093344075 0.061051866 0.84560406
## 241  0.459908972 0.266728075 0.27336295
## 242  0.459908972 0.266728075 0.27336295
## 243  0.459908972 0.266728075 0.27336295
## 247  0.459908972 0.266728075 0.27336295
## 248  0.459908972 0.266728075 0.27336295
## 269  0.563463602 0.397353523 0.03918288
## 275  0.430342294 0.486180755 0.08347695
## 276  0.454851241 0.422639166 0.12250959
## 280  0.468781448 0.331546612 0.19967194
## 281  0.459908972 0.266728075 0.27336295
## 282  0.468761400 0.336508309 0.19473029
## 287  0.342358647 0.628232978 0.02940838
## 294  0.461657429 0.273384794 0.26495778
## 307  0.459908972 0.266728075 0.27336295
## 310  0.451996037 0.452028906 0.09597506
## 312  0.009110165 0.018116055 0.97277378
## 324  0.006307719 0.004930842 0.98876144
## 333  0.459908972 0.266728075 0.27336295
## 343  0.340941238 0.630121723 0.02893704
## 352  0.020159718 0.014511652 0.96532863
```

```
## 353  0.465682986 0.293224563 0.24109245
## 378  0.474631596 0.268022790 0.25734561
## 381  0.334438408 0.638696533 0.02686506
## 383  0.845214169 0.079963098 0.07482273
## 403  0.442915092 0.457054525 0.10003038
## 407  0.459908972 0.266728075 0.27336295
## 408  0.468384904 0.318490789 0.21312431
## 412  0.468738307 0.327979268 0.20328243
## 417  0.415175028 0.516274021 0.06855095
## 428  0.420137167 0.505480717 0.07438212
## 432  0.353011726 0.620369839 0.02661843
## 438  0.456509936 0.416913212 0.12657685
## 446  0.462437014 0.276634638 0.26092835
## 448  0.466369657 0.369381137 0.16424921
## 449  0.468359505 0.344249176 0.18739132
## 455  0.468629637 0.342073823 0.18929654
## 462  0.104637389 0.066539365 0.82882325
## 464  0.465835940 0.294196108 0.23996795
## 471  0.324925757 0.650347344 0.02472690
## 475  0.442762325 0.457441725 0.09979595
## 482  0.245003811 0.149478066 0.60551812
## 485  0.459908972 0.266728075 0.27336295
## 490  0.459908972 0.266728075 0.27336295
## 491  0.459908972 0.266728075 0.27336295
## 498  0.464611156 0.287024061 0.24836478
## 499  0.466821755 0.301211456 0.23196679
## 500  0.465494266 0.292059655 0.24244608
## 501  0.459908972 0.266728075 0.27336295
## 508  0.344231213 0.625221181 0.03054761
## 510  0.330074065 0.644060196 0.02586574
## 519  0.369029144 0.591147418 0.03982344
## 521  0.332470048 0.641119525 0.02641043
## 529  0.462392418 0.276443182 0.26116440
## 549  0.332226552 0.641366621 0.02640683
## 557  0.459908972 0.266728075 0.27336295
## 558  0.425631424 0.495991847 0.07837673
## 559  0.318115439 0.657658601 0.02422596
## 562  0.357848719 0.607077678 0.03507360
## 567  0.459908972 0.266728075 0.27336295
## 577  0.373448821 0.581944419 0.04460676
## 580  0.465851429 0.373171857 0.16097671
## 582  0.368493368 0.590023140 0.04148349
## 585  0.459908972 0.266728075 0.27336295
## 588  0.340799363 0.629649126 0.02955151
## 595  0.410532817 0.526740003 0.06272718
## 599  0.459908972 0.266728075 0.27336295
## 600  0.451420811 0.433434875 0.11514431
## 601  0.450804789 0.435421910 0.11377330
## 602  0.003031537 0.003090977 0.99387749
## 605  0.459908972 0.266728075 0.27336295
```

```
## 608  0.459908972 0.266728075 0.27336295
## 611  0.374061361 0.582986310 0.04295233
## 612  0.443477955 0.258329144 0.29819290
## 614  0.449268286 0.439926079 0.11080563
## 624  0.459908972 0.266728075 0.27336295
## 625  0.438932665 0.459243386 0.10182395
## 626  0.460325631 0.268245431 0.27142894
## 627  0.459908972 0.266728075 0.27336295
## 628  0.420393750 0.505419464 0.07418679
## 630  0.432469839 0.481351348 0.08617881
## 631  0.434685546 0.474214125 0.09110033
## 632  0.466989957 0.302579482 0.23043056
## 635  0.420185053 0.560151573 0.01966337
## 636  0.441061937 0.460753598 0.09818447
## 637  0.459908972 0.266728075 0.27336295
## 640  0.463387966 0.387229593 0.14938244
## 641  0.001774896 0.001313338 0.99691177
## 645  0.459908972 0.266728075 0.27336295
## 647  0.459908972 0.266728075 0.27336295
## 648  0.462568995 0.389125751 0.14830525
## 651  0.459908972 0.266728075 0.27336295
## 654  0.381234102 0.572975773 0.04579012
## 657  0.398307342 0.545626102 0.05606656
## 659  0.461323180 0.272049710 0.26662711
## 663  0.459908972 0.266728075 0.27336295
## 664  0.740171370 0.183083707 0.07674492
## 667  0.459908972 0.266728075 0.27336295
## 668  0.468468044 0.320069496 0.21146246
## 672  0.013443653 0.009827945 0.97672840
## 673  0.383482221 0.569526499 0.04699128
## 677  0.458316997 0.409703996 0.13197901
## 678  0.459908972 0.266728075 0.27336295
## 683  0.332147317 0.641516177 0.02633651
## 693  0.459908972 0.266728075 0.27336295
## 701  0.459908972 0.266728075 0.27336295
## 705  0.529101655 0.437299402 0.03359894
## 715  0.441527341 0.460535773 0.09793689
## 717  0.467300694 0.361400894 0.17129841
## 719  0.395906812 0.548927126 0.05516606
## 721  0.316464612 0.660590456 0.02294493
## 731  0.377613734 0.578460360 0.04392591
## 733  0.462874307 0.278551148 0.25857455
## 736  0.417918553 0.508546343 0.07353510
## 751  0.360423247 0.504988927 0.13458783
## 752  0.459908972 0.266728075 0.27336295
## 755  0.413647801 0.516127896 0.07022430
## 770  0.459908972 0.266728075 0.27336295
## 775  0.422261079 0.502734136 0.07500479
## 776  0.333592372 0.639738776 0.02666885
## 779  0.465031945 0.289345644 0.24562241
```

```
## 789  0.443072638 0.449152622 0.10777474
## 791  0.332873842 0.640403522 0.02672264
## 792  0.468532809 0.321451796 0.21001539
## 793  0.468785114 0.333133508 0.19808138
## 794  0.459908972 0.266728075 0.27336295
## 796  0.459908972 0.266728075 0.27336295
## 797  0.459908972 0.266728075 0.27336295
## 798  0.315766011 0.661431383 0.02280261
## 799  0.230816400 0.141303392 0.62788021
## 801  0.459908972 0.266728075 0.27336295
## 802  0.459908972 0.266728075 0.27336295
## 803  0.459908972 0.266728075 0.27336295
## 805  0.459908972 0.266728075 0.27336295
## 806  0.361612699 0.601785077 0.03660222
## 807  0.459908972 0.266728075 0.27336295
## 808  0.459908972 0.266728075 0.27336295
## 809  0.004723031 0.006289151 0.98898782
## 816  0.459908972 0.266728075 0.27336295
## 817  0.468493869 0.345456561 0.18604957
## 824  0.461515125 0.272812391 0.26567248
## 826  0.459908972 0.266728075 0.27336295
## 828  0.459908972 0.266728075 0.27336295
## 829  0.459908972 0.266728075 0.27336295
## 830  0.459908972 0.266728075 0.27336295
## 831  0.459908972 0.266728075 0.27336295
## 832  0.362589747 0.599220798 0.03818946
## 841  0.459908972 0.266728075 0.27336295
## 849  0.459908972 0.266728075 0.27336295
## 854  0.459908972 0.266728075 0.27336295
## 855  0.459908972 0.266728075 0.27336295
## 858  0.459908972 0.266728075 0.27336295
## 861  0.459908972 0.266728075 0.27336295
## 865  0.437552497 0.468248636 0.09419887
## 872  0.459147308 0.407012669 0.13384002
## 873  0.420027577 0.506665998 0.07330643
## 881  0.459908972 0.266728075 0.27336295
## 882  0.464997744 0.289152010 0.24585025
## 885  0.321183586 0.654891079 0.02392534
## 889  0.308512904 0.670057835 0.02142926
## 892  0.459908972 0.266728075 0.27336295
## 896  0.408536006 0.526053480 0.06541051
## 907  0.439724150 0.256392262 0.30388359
## 908  0.459908972 0.266728075 0.27336295
## 910  0.459908972 0.266728075 0.27336295
## 915  0.337850649 0.634065536 0.02808381
## 919  0.468667529 0.340880541 0.19045193
## 921  0.459908972 0.266728075 0.27336295
## 925  0.459908972 0.266728075 0.27336295
## 929  0.452350580 0.430710967 0.11693845
## 930  0.440579936 0.462343481 0.09707658
```

```
## 936  0.459908972 0.266728075 0.27336295
## 944  0.442378840 0.458409295 0.09921186
## 953  0.459908972 0.266728075 0.27336295
## 959  0.459908972 0.266728075 0.27336295
## 961  0.459908972 0.266728075 0.27336295
## 962  0.007902080 0.005542687 0.98655523
## 964  0.354835955 0.611273017 0.03389103
## 966  0.009261067 0.006458825 0.98428011
## 968  0.460377005 0.268435277 0.27118772
## 980  0.463044809 0.279318731 0.25763646
## 992  0.459908972 0.266728075 0.27336295
## 993  0.447538534 0.444809599 0.10765187
## 994  0.459908972 0.266728075 0.27336295
## 998  0.459908972 0.266728075 0.27336295
## 999  0.459908972 0.266728075 0.27336295
## 1008 0.466594278 0.299454394 0.23395133
## 1011 0.459908972 0.266728075 0.27336295
## 1012 0.002081656 0.001694411 0.99622393
## 1022 0.339995950 0.631377335 0.02862671
## 1024 0.187016640 0.115779929 0.69720343
## 1025 0.459908972 0.266728075 0.27336295
## 1031 0.389817807 0.558662111 0.05152008
## 1033 0.459908972 0.266728075 0.27336295
## 1034 0.459908972 0.266728075 0.27336295
## 1036 0.350473653 0.616935979 0.03259037
## 1038 0.459908972 0.266728075 0.27336295
## 1041 0.047025728 0.068893955 0.88408032
## 1046 0.167707745 0.122363667 0.70992859
## 1048 0.465301469 0.295348146 0.23935039
## 1049 0.317163647 0.659748288 0.02308807
## 1051 0.464004284 0.290846890 0.24514883
## 1056 0.459908972 0.266728075 0.27336295
## 1058 0.028430388 0.019030353 0.95253926
## 1059 0.411363368 0.523249370 0.06538726
## 1061 0.460981207 0.270716468 0.26830232
## 1064 0.459908972 0.266728075 0.27336295
## 1066 0.459908972 0.266728075 0.27336295
## 1070 0.312718684 0.665091236 0.02219008
## 1072 0.389017518 0.560061859 0.05092062
## 1075 0.461844960 0.274148515 0.26400652
## 1077 0.459908972 0.266728075 0.27336295
## 1080 0.305779139 0.673376544 0.02084432
## 1089 0.449063965 0.440512792 0.11042324
## 1092 0.455085528 0.421850160 0.12306431
## 1094 0.464535959 0.381548722 0.15391532
## 1098 0.390041632 0.559256461 0.05070191
## 1099 0.459908972 0.266728075 0.27336295
## 1105 0.459908972 0.266728075 0.27336295
## 1109 0.336926284 0.287657691 0.37541602
## 1111 0.374195530 0.583563528 0.04224094
```

```
## 1113 0.459908972 0.266728075 0.27336295
## 1115 0.444719975 0.452400620 0.10287940
## 1116 0.468395830 0.318688064 0.21291611
## 1127 0.374680887 0.582715525 0.04260359
## 1128 0.335391313 0.637404624 0.02720406
## 1134 0.459908972 0.266728075 0.27336295
## 1136 0.459908972 0.266728075 0.27336295
## 1140 0.468772445 0.330356997 0.20087056
## 1144 0.462831287 0.278359339 0.25880937
## 1146 0.425446458 0.496367511 0.07818603
## 1175 0.340367040 0.630742822 0.02889014
## 1180 0.459908972 0.266728075 0.27336295
## 1182 0.217966882 0.167917315 0.61411580
## 1186 0.333615527 0.199466026 0.46691845
## 1187 0.459908972 0.266728075 0.27336295
## 1191 0.459908972 0.266728075 0.27336295
## 1197 0.214086749 0.131605616 0.65430764
## 1203 0.465167177 0.290120480 0.24471234
## 1205 0.082916628 0.053243385 0.86383999
## 1211 0.459908972 0.266728075 0.27336295
## 1216 0.324665759 0.650663737 0.02467050
## 1221 0.459908972 0.266728075 0.27336295
## 1225 0.461177566 0.271478094 0.26734434
## 1238 0.432450239 0.481624860 0.08592490
## 1239 0.459908972 0.266728075 0.27336295
## 1243 0.464789237 0.287990849 0.24721991
## 1244 0.326926359 0.647058815 0.02601483
## 1252 0.403408699 0.537223804 0.05936750
## 1254 0.460170565 0.267676126 0.27215331
## 1261 0.459908972 0.266728075 0.27336295
## 1262 0.351549623 0.615800992 0.03264939
## 1279 0.362776189 0.600134518 0.03708929
## 1280 0.332854327 0.640646999 0.02649867
## 1288 0.459908972 0.266728075 0.27336295
## 1298 0.312740549 0.664514295 0.02274516
## 1302 0.220414158 0.135280867 0.64430497
## 1304 0.459908972 0.266728075 0.27336295
## 1306 0.459908972 0.266728075 0.27336295
## 1307 0.361961912 0.601290361 0.03674773
## 1313 0.459961606 0.266917606 0.27312079
## 1314 0.462257682 0.275869027 0.26187329
## 1317 0.461579572 0.396684603 0.14173583
## 1318 0.466326702 0.297504581 0.23616872
## 1322 0.459908972 0.266728075 0.27336295
## 1323 0.181197953 0.114780754 0.70402129
## 1327 0.459908972 0.266728075 0.27336295
## 1329 0.794233995 0.189527766 0.01623824
## 1333 0.427192530 0.492793602 0.08001387
## 1340 0.444867996 0.452012180 0.10311982
## 1347 0.459908972 0.266728075 0.27336295
```

```
## 1348 0.464997744 0.289152010 0.24585025
## 1349 0.459908972 0.266728075 0.27336295
## 1357 0.422401191 0.247374112 0.33022470
## 1358 0.459908972 0.266728075 0.27336295
## 1368 0.459908972 0.266728075 0.27336295
## 1369 0.364360753 0.597537128 0.03810212
## 1373 0.459908972 0.266728075 0.27336295
## 1374 0.459908972 0.266728075 0.27336295
## 1377 0.459908972 0.266728075 0.27336295
## 1378 0.463621704 0.282009585 0.25436871
## 1382 0.467196309 0.362398301 0.17040539
## 1385 0.002141527 0.001574046 0.99628443
## 1390 0.300014886 0.180736342 0.51924877
## 1391 0.397579354 0.544862365 0.05755828
## 1392 0.033127858 0.025880088 0.94099205
## 1393 0.369321008 0.589244772 0.04143422
## 1395 0.459908972 0.266728075 0.27336295
## 1400 0.005747609 0.004077923 0.99017447
## 1401 0.146370495 0.283059769 0.57056974
## 1404 0.459908972 0.266728075 0.27336295
## 1406 0.316707706 0.660297669 0.02299463
## 1408 0.337946740 0.201858813 0.46019445
## 1414 0.418852804 0.509143459 0.07200374
## 1423 0.377386555 0.578801739 0.04381171
## 1425 0.318431249 0.191037951 0.49053080
## 1426 0.400233119 0.542608678 0.05715820
## 1430 0.459908972 0.266728075 0.27336295
## 1431 0.020193344 0.013689080 0.96611758
## 1432 0.459908972 0.266728075 0.27336295
## 1434 0.459908972 0.266728075 0.27336295
## 1435 0.467401620 0.360403546 0.17219483
## 1436 0.467837614 0.355617398 0.17654499
## 1437 0.355304721 0.610623368 0.03407191
## 1438 0.407577829 0.529991506 0.06243066
## 1443 0.346718342 0.112485155 0.54079650
## 1445 0.459908972 0.266728075 0.27336295
## 1446 0.082181704 0.052790755 0.86502754
## 1449 0.459908972 0.266728075 0.27336295
## 1450 0.393638483 0.232856846 0.37350467
## 1452 0.329932733 0.643804304 0.02626296
## 1458 0.001774896 0.001313338 0.99691177
## 1459 0.362310270 0.234864478 0.40282525
## 1461 0.378835065 0.575082204 0.04608273
## 1467 0.338459348 0.633411638 0.02812901
## 1468 0.356007512 0.609647588 0.03434490
## 1473 0.459197113 0.406814331 0.13398856
## 1483 0.459908972 0.266728075 0.27336295
## 1491 0.459908972 0.266728075 0.27336295
## 1493 0.443032084 0.260481641 0.29648628
## 1498 0.361845523 0.601455309 0.03669917
```

```
## 1502 0.389762532 0.559334817 0.05090265
## 1507 0.459908972 0.266728075 0.27336295
## 1510 0.459908972 0.266728075 0.27336295
## 1518 0.456324520 0.264907272 0.27876821
## 1523 0.458693808 0.408797169 0.13250902
## 1527 0.346943004 0.622009657 0.03104734
## 1538 0.459908972 0.266728075 0.27336295
## 1543 0.459908972 0.266728075 0.27336295
## 1550 0.441342907 0.524508991 0.03414810
## 1555 0.459908972 0.266728075 0.27336295
## 1560 0.389490230 0.560132236 0.05037753
## 1561 0.389394761 0.560091668 0.05051357
## 1565 0.433401796 0.479531082 0.08706712
## 1566 0.464355236 0.285671873 0.24997289
## 1570 0.465992495 0.372174335 0.16183317
## 1575 0.464842733 0.290119082 0.24503819
## 1580 0.391845505 0.560877006 0.04727749
## 1584 0.312305903 0.665578738 0.02211536
## 1588 0.345191206 0.624436345 0.03037245
## 1589 0.393003812 0.554510096 0.05248609
## 1594 0.715545360 0.233703437 0.05075120
## 1599 0.459908972 0.266728075 0.27336295
## 1601 0.459908972 0.266728075 0.27336295
## 1602 0.051713936 0.026616304 0.92166976
## 1604 0.459908972 0.266728075 0.27336295
## 1608 0.048322364 0.031701746 0.91997589
## 1609 0.006120933 0.004540239 0.98933883
## 1610 0.316803870 0.190131100 0.49306503
## 1614 0.386052659 0.565539653 0.04840769
## 1616 0.467361665 0.360802477 0.17183586
## 1621 0.326257089 0.648725572 0.02501734
## 1622 0.341552538 0.628054097 0.03039337
## 1623 0.459908972 0.266728075 0.27336295
## 1627 0.459908972 0.266728075 0.27336295
## 1630 0.400871434 0.541534434 0.05759413
## 1643 0.028796055 0.010106089 0.96109786
## 1652 0.459908972 0.266728075 0.27336295
## 1657 0.396197106 0.547618169 0.05618472
## 1662 0.386954785 0.228555969 0.38448925
## 1664 0.459908972 0.266728075 0.27336295
## 1665 0.459908972 0.266728075 0.27336295
## 1667 0.355421883 0.610460847 0.03411727
## 1669 0.371148123 0.586029934 0.04282194
## 1670 0.014819545 0.010160636 0.97501982
## 1676 0.466043497 0.295557455 0.23839905
## 1677 0.459908972 0.266728075 0.27336295
## 1682 0.459908972 0.266728075 0.27336295
## 1687 0.005382328 0.006435892 0.98818178
## 1689 0.008091253 0.005670540 0.98623821
## 1691 0.462613824 0.277400819 0.25998536
```

```
## 1695 0.461323180 0.272049710 0.26662711
## 1696 0.308172140 0.670527043 0.02130082
## 1698 0.459908972 0.266728075 0.27336295
## 1700 0.459908972 0.266728075 0.27336295
## 1701 0.447468075 0.445004663 0.10752726
## 1703 0.392338710 0.553210537 0.05445075
## 1704 0.446183381 0.448512064 0.10530455
## 1706 0.459908972 0.266728075 0.27336295
## 1707 0.331118053 0.642780040 0.02610191
## 1709 0.459908972 0.266728075 0.27336295
## 1710 0.304007262 0.675481408 0.02051133
## 1711 0.459908972 0.266728075 0.27336295
## 1712 0.429456762 0.252277820 0.31826542
## 1714 0.468671182 0.325207928 0.20612089
## 1717 0.459908972 0.266728075 0.27336295
## 1719 0.459908972 0.266728075 0.27336295
## 1720 0.460911273 0.399666692 0.13942203
## 1722 0.459908972 0.266728075 0.27336295
## 1723 0.459908972 0.266728075 0.27336295
## 1725 0.377500157 0.578631070 0.04386877
## 1727 0.459908972 0.266728075 0.27336295
## 1728 0.327635618 0.647043342 0.02532104
## 1732 0.337387865 0.634695742 0.02791639
## 1736 0.461371403 0.272240324 0.26638827
## 1746 0.344837322 0.624912318 0.03025036
## 1750 0.413479692 0.519400875 0.06711943
## 1751 0.459908972 0.266728075 0.27336295
## 1752 0.459908972 0.266728075 0.27336295
## 1766 0.462481453 0.276826130 0.26069242
## 1769 0.459908972 0.266728075 0.27336295
## 1771 0.459908972 0.266728075 0.27336295
## 1775 0.462788110 0.278167565 0.25904433
## 1776 0.459908972 0.266728075 0.27336295
## 1777 0.459908972 0.266728075 0.27336295
## 1786 0.305448522 0.673769615 0.02078186
## 1787 0.008118640 0.005689040 0.98619232
## 1790 0.404772047 0.534879830 0.06034812
## 1792 0.383163100 0.569316524 0.04752038
## 1795 0.312733824 0.665073086 0.02219309
## 1799 0.459991496 0.403532907 0.13647560
## 1800 0.468493869 0.345456561 0.18604957
## 1801 0.396146895 0.549276060 0.05457705
## 1810 0.464364003 0.382545560 0.15309044
## 1817 0.459908972 0.266728075 0.27336295
## 1825 0.459908972 0.266728075 0.27336295
## 1831 0.695366063 0.157748217 0.14688572
## 1833 0.351799316 0.607951653 0.04024903
## 1843 0.463776433 0.287397856 0.24882571
## 1847 0.459908972 0.266728075 0.27336295
## 1848 0.459908972 0.266728075 0.27336295
```

```
## 1850 0.434173384 0.477815392 0.08801122
## 1863 0.468069047 0.352627294 0.17930366
## 1870 0.002505459 0.001831249 0.99566329
## 1871 0.007363429 0.003783003 0.98885357
## 1874 0.361285739 0.601364538 0.03734972
## 1877 0.464704677 0.380551777 0.15474355
## 1880 0.468435971 0.346651046 0.18491298
## 1881 0.010959557 0.007596968 0.98144348
## 1890 0.285400286 0.229603192 0.48499652
## 1897 0.459908972 0.266728075 0.27336295
## 1900 0.459908972 0.266728075 0.27336295
## 1917 0.090961616 0.174443799 0.73459459
## 1920 0.392801876 0.553604695 0.05359343
## 1921 0.359537918 0.603337466 0.03712462
## 1925 0.465285070 0.376962019 0.15775291
## 1928 0.461924769 0.395093322 0.14298191
## 1929 0.459908972 0.266728075 0.27336295
## 1933 0.459908972 0.266728075 0.27336295
## 1934 0.016196307 0.011627225 0.97217647
## 1938 0.459908972 0.266728075 0.27336295
## 1939 0.459908972 0.266728075 0.27336295
## 1942 0.459908972 0.266728075 0.27336295
## 1943 0.007966576 0.014702068 0.97733136
## 1944 0.087077074 0.149982120 0.76294081
## 1947 0.459908972 0.266728075 0.27336295
## 1951 0.340439810 0.630498151 0.02906204
## 1952 0.465765188 0.373770352 0.16046446
## 1959 0.459908972 0.266728075 0.27336295
## 1964 0.459908972 0.266728075 0.27336295
## 1965 0.016244891 0.001226508 0.98252860
## 1968 0.460428222 0.268625163 0.27094661
## 1969 0.458310240 0.395759158 0.14593060
## 1973 0.459908972 0.266728075 0.27336295
## 1974 0.459908972 0.266728075 0.27336295
## 1978 0.467238877 0.304731679 0.22802944
## 1979 0.459908972 0.266728075 0.27336295
## 1983 0.025594462 0.031438034 0.94296750
## 1984 0.039296291 0.025985154 0.93471855
## 1987 0.459908972 0.266728075 0.27336295
## 1988 0.454517737 0.263986990 0.28149527
## 1989 0.007769635 0.005453108 0.98677726
## 1993 0.461511796 0.347747978 0.19074023
## 1995 0.333500100 0.639852374 0.02664753
## 1999 0.036437410 0.035354592 0.92820800
## 2000 0.406777711 0.531333220 0.06188907
## 2003 0.464569962 0.381349341 0.15408070
##
## $x
##               LD1          LD2
## 3    -1.605321343  1.394584455
```

```
## 12   -0.392358040 -5.275400590
## 13   -1.696531983  1.543108825
## 16   -0.558945875  0.248376261
## 21    0.289558127 -0.769197689
## 22   -0.784812079  0.454858382
## 26    0.289558127 -0.769197689
## 28    0.410058518 -0.347965362
## 30   -1.441852904  1.193844022
## 33    3.528717757  3.748178641
## 37    0.289558127 -0.769197689
## 49   -0.249181188 -6.630510434
## 52    3.994063307  3.234696211
## 54   -0.064386710 -0.368444123
## 57    0.289558127 -0.769197689
## 59   -1.528980486  1.361990306
## 64   -1.685379987  1.524949285
## 68    0.289558127 -0.769197689
## 69   -0.994110749  0.697504170
## 72   -1.105139925  0.858482189
## 74   -0.875394172  0.549817984
## 79    0.289558127 -0.769197689
## 81    0.289558127 -0.769197689
## 84    0.289558127 -0.769197689
## 89   -1.607044009  1.397389587
## 95   -1.251816080  0.976111147
## 104  -1.103925484  0.808572255
## 107  -0.182368323 -0.234859601
## 111  -0.945139448  0.632970246
## 112  -1.695806650  1.541927716
## 117   0.289558127 -0.769197689
## 122   1.923582830  0.290415709
## 123   0.289558127 -0.769197689
## 127   0.289558127 -0.769197689
## 132   0.289558127 -0.769197689
## 144   0.289558127 -0.769197689
## 145   0.289558127 -0.769197689
## 146   0.289558127 -0.769197689
## 148   0.289558127 -0.769197689
## 152  -1.677038656  1.511366539
## 153  -1.683806392  1.539900649
## 161   0.140483081 -0.526743474
## 166  -0.856814390  0.528781052
## 169  -0.512159444  0.138545952
## 170  -0.591123515  0.227952916
## 196   0.289558127 -0.769197689
## 198  -1.200865249  0.928632904
## 201   0.219883946 -0.690309192
## 205  -1.568420021  1.334495570
## 212  -1.072339855  0.772809470
## 215   0.251469575 -0.726071977
```

```
## 225   0.225457881 -0.696620272
## 226   0.289558127 -0.769197689
## 230  -0.960861167  0.646587874
## 231  -0.919967164 -0.022818454
## 232  -1.215722955  0.997993747
## 235  -1.197753380  0.914808765
## 240   2.107539836  0.470913883
## 241   0.289558127 -0.769197689
## 242   0.289558127 -0.769197689
## 243   0.289558127 -0.769197689
## 247   0.289558127 -0.769197689
## 248   0.289558127 -0.769197689
## 269  -1.260836860 -1.062489293
## 275  -0.749980647  0.407818690
## 276  -0.445272231  0.062812995
## 280  -0.019795235 -0.418932761
## 281   0.289558127 -0.769197689
## 282  -0.043019962 -0.392636596
## 287  -1.506177753  1.264021846
## 294   0.257043509 -0.732383057
## 307   0.289558127 -0.769197689
## 310  -0.638259997  0.152431075
## 312   3.344777920  3.956444273
## 324   3.988829443  1.892390290
## 333   0.289558127 -0.769197689
## 343  -1.517325622  1.276644006
## 352   3.209656255  1.281549031
## 353   0.161357635 -0.624042854
## 378   0.237455619 -0.864286314
## 381  -1.568420021  1.334495570
## 383  -0.275870271 -5.708566055
## 403  -0.608774308  0.247938002
## 407   0.289558127 -0.769197689
## 408   0.041518044 -0.488354639
## 412  -0.003073431 -0.437866001
## 417  -0.899547888  0.577165997
## 428  -0.838269315  0.523931723
## 432  -1.577858421  1.116409696
## 438  -0.418331547  0.032309443
## 446   0.241250695 -0.714501664
## 448  -0.196303159 -0.219081902
## 449  -0.078391266 -0.350248453
## 455  -0.069031656 -0.363184890
## 462   2.027002115  0.357479848
## 464   0.156712689 -0.618783621
## 471  -1.624633338  1.426031464
## 475  -0.610632286  0.250041695
## 482   1.245611962 -0.149226981
## 485   0.289558127 -0.769197689
## 490   0.289558127 -0.769197689
```

```
## 491   0.289558127 -0.769197689
## 498   0.191085285 -0.657701946
## 499   0.123269083 -0.580917142
## 500   0.166931569 -0.630353934
## 501   0.289558127 -0.769197689
## 508  -1.479680464  1.251293167
## 510  -1.594169347  1.376424915
## 519  -1.294368244  1.024200814
## 521  -1.580025351  1.353393303
## 529   0.242179684 -0.715553511
## 549  -1.580072748  1.356235813
## 557   0.289558127 -0.769197689
## 558  -0.798288079  0.462514714
## 559  -1.637345623  1.502499205
## 562  -1.383732528  1.125473368
## 567   0.289558127 -0.769197689
## 577  -1.213524633  0.994874961
## 580  -0.213953951 -0.199096816
## 582  -1.265196670  1.037996931
## 585   0.289558127 -0.769197689
## 588  -1.502502007  1.283385237
## 595  -0.965467975  0.623674528
## 599   0.289558127 -0.769197689
## 600  -0.495959318  0.120923083
## 601  -0.505656520  0.131183026
## 602   4.386174672  2.777324202
## 605   0.289558127 -0.769197689
## 608   0.289558127 -0.769197689
## 611  -1.240613627  0.981374745
## 612   0.372293555 -0.715546378
## 614  -0.527023269  0.155375498
## 624   0.289558127 -0.769197689
## 625  -0.596037094  0.289877949
## 626   0.282126214 -0.760782916
## 627   0.289558127 -0.769197689
## 628  -0.840191509  0.521070212
## 630  -0.725443318  0.382150455
## 631  -0.682701257  0.352647985
## 632   0.116766159 -0.573554216
## 635  -1.792100121  0.298092448
## 636  -0.623665959  0.271726183
## 637   0.289558127 -0.769197689
## 640  -0.278803229 -0.123917035
## 641   4.885970794  2.211430715
## 645   0.289558127 -0.769197689
## 647   0.289558127 -0.769197689
## 648  -0.285265635 -0.110167651
## 651   0.289558127 -0.769197689
## 654  -1.194966413  0.911653225
## 657  -1.048392547  0.749023783
```

```
## 659   0.263546433 -0.739745983
## 663   0.289558127 -0.769197689
## 664  -0.559622479 -3.389542243
## 667   0.289558127 -0.769197689
## 668   0.034086132 -0.479939866
## 672   3.489108439  1.464419210
## 673  -1.176386632  0.890616292
## 677  -0.383666005 -0.004781466
## 678   0.289558127 -0.769197689
## 683  -1.581929350  1.356493712
## 693   0.289558127 -0.769197689
## 701   0.289558127 -0.769197689
## 705  -1.387838217 -0.720602835
## 715  -0.625496111  0.266871241
## 717  -0.159143596 -0.261155767
## 719  -1.060274005  0.773901812
## 721  -1.674953323  1.507970853
## 731  -1.224694063  0.945312317
## 733   0.231960804 -0.703983198
## 736  -0.847116880  0.548478345
## 751  -0.388895552  1.144868041
## 752   0.289558127 -0.769197689
## 755  -0.881879969  0.594945859
## 770   0.289558127 -0.769197689
## 775  -0.831731685  0.500381193
## 776  -1.573406686  1.342615690
## 779   0.179937416 -0.645079787
## 789  -0.550647813  0.232098644
## 791  -1.571851286  1.351605135
## 792   0.027583208 -0.472576939
## 793  -0.027227147 -0.410517988
## 794   0.289558127 -0.769197689
## 796   0.289558127 -0.769197689
## 797   0.289558127 -0.769197689
## 798  -1.679123988  1.514762225
## 799   1.309961739 -0.107498183
## 801   0.289558127 -0.769197689
## 802   0.289558127 -0.769197689
## 803   0.289558127 -0.769197689
## 805   0.289558127 -0.769197689
## 806  -1.353823545  1.091518999
## 807   0.289558127 -0.769197689
## 808   0.289558127 -0.769197689
## 809   3.969987567  3.248551492
## 816   0.289558127 -0.769197689
## 817  -0.084824470 -0.345303497
## 824   0.259830476 -0.735538597
## 826   0.289558127 -0.769197689
## 828   0.289558127 -0.769197689
## 829   0.289558127 -0.769197689
```

```
## 830   0.289558127 -0.769197689
## 831   0.289558127 -0.769197689
## 832  -1.323734296  1.089072382
## 841   0.289558127 -0.769197689
## 849   0.289558127 -0.769197689
## 854   0.289558127 -0.769197689
## 855   0.289558127 -0.769197689
## 858   0.289558127 -0.769197689
## 861   0.289558127 -0.769197689
## 865  -0.656435758  0.316791444
## 872  -0.371882094 -0.020282889
## 873  -0.849136928  0.525036205
## 881   0.289558127 -0.769197689
## 882   0.180866405 -0.646131633
## 885  -1.646846664  1.462202906
## 889  -1.720539072  1.586349163
## 892   0.289558127 -0.769197689
## 896  -0.934872864  0.648514659
## 907   0.390679206 -0.703623864
## 908   0.289558127 -0.769197689
## 910   0.289558127 -0.769197689
## 915  -1.537858176  1.305469651
## 919  -0.063457721 -0.369495970
## 921   0.289558127 -0.769197689
## 925   0.289558127 -0.769197689
## 929  -0.483360783  0.105938707
## 930  -0.632545786  0.278811586
## 936   0.289558127 -0.769197689
## 944  -0.615277231  0.255300928
## 953   0.289558127 -0.769197689
## 959   0.289558127 -0.769197689
## 961   0.289558127 -0.769197689
## 962   3.877058213  1.557182780
## 964  -1.407704911  1.152526103
## 966   3.769042516  1.487138013
## 968   0.281197225 -0.759731069
## 980   0.228244848 -0.699775811
## 992   0.289558127 -0.769197689
## 993  -0.550247996  0.181671664
## 994   0.289558127 -0.769197689
## 998   0.289558127 -0.769197689
## 999   0.289558127 -0.769197689
## 1008  0.131629984 -0.590383762
## 1011  0.289558127 -0.769197689
## 1012  4.737332542  2.379726176
## 1022 -1.524757534  1.285058779
## 1024  1.521396722  0.029610723
## 1025  0.289558127 -0.769197689
## 1031 -1.110011156  0.833230386
## 1033  0.289558127 -0.769197689
```

```
## 1034  0.289558127 -0.769197689
## 1036 -1.434857506  1.193975155
## 1038  0.289558127 -0.769197689
## 1041  2.278943462  2.626683130
## 1046  1.541463389  0.461152797
## 1048  0.153670541 -0.607377452
## 1049 -1.670782657  1.501179480
## 1051  0.177117119 -0.627582437
## 1056  0.289558127 -0.769197689
## 1058  2.994546981  0.984902127
## 1059 -0.934849472  0.617136169
## 1061  0.270049356 -0.747108910
## 1064  0.289558127 -0.769197689
## 1066  0.289558127 -0.769197689
## 1070 -1.697347983  1.544437572
## 1072 -1.118483671  0.840574312
## 1075  0.253327553 -0.728175670
## 1077  0.289558127 -0.769197689
## 1080 -1.739054637  1.612351299
## 1089 -0.529810236  0.158531038
## 1092 -0.441556274  0.058605609
## 1094 -0.252971492 -0.154919257
## 1098 -1.121576276  0.828557341
## 1099  0.289558127 -0.769197689
## 1105  0.289558127 -0.769197689
## 1109  0.562500061  0.362308415
## 1111 -1.252563736  0.976867716
## 1113  0.289558127 -0.769197689
## 1115 -0.586478570  0.222693683
## 1116  0.040589055 -0.487302792
## 1127 -1.246470876  0.972937853
## 1128 -1.559788495  1.326431828
## 1134  0.289558127 -0.769197689
## 1136  0.289558127 -0.769197689
## 1140 -0.014221300 -0.425243841
## 1144  0.232889793 -0.705035045
## 1146 -0.800146057  0.464618408
## 1175 -1.518371728  1.282956333
## 1180  0.289558127 -0.769197689
## 1182  1.240259629  0.446146744
## 1186  0.864109710 -0.396619138
## 1187  0.289558127 -0.769197689
## 1191  0.289558127 -0.769197689
## 1197  1.388100754 -0.056827500
## 1203  0.176221460 -0.640872400
## 1205  2.210858622  0.476704984
## 1211  0.289558127 -0.769197689
## 1216 -1.626174671  1.428541319
## 1221  0.289558127 -0.769197689
## 1225  0.266333400 -0.742901523
```

```
## 1238 -0.727684909  0.382574370
## 1239  0.289558127 -0.769197689
## 1243  0.186440340 -0.652442713
## 1244 -1.589466757  1.415317089
## 1252 -1.006381631  0.698128359
## 1254  0.284913181 -0.763938456
## 1261  0.289558127 -0.769197689
## 1262 -1.433716605  1.181977809
## 1279 -1.344533654  1.081000532
## 1280 -1.577758685  1.349702340
## 1288  0.289558127 -0.769197689
## 1298 -1.680079120  1.550884150
## 1302  1.358224072 -0.076201585
## 1304  0.289558127 -0.769197689
## 1306  0.289558127 -0.769197689
## 1307 -1.351036578  1.088363459
## 1313  0.288629138 -0.768145842
## 1314  0.244966651 -0.718709051
## 1317 -0.323574662 -0.074978913
## 1318  0.140919875 -0.600902228
## 1322  0.289558127 -0.769197689
## 1323  1.540371632  0.098160421
## 1327  0.289558127 -0.769197689
## 1329 -1.657807001 -4.044487300
## 1333 -0.782495264  0.444633322
## 1340 -0.584620592  0.220589989
## 1347  0.289558127 -0.769197689
## 1348  0.180866405 -0.646131633
## 1349  0.289558127 -0.769197689
## 1357  0.473414634 -0.649972553
## 1358  0.289558127 -0.769197689
## 1368  0.289558127 -0.769197689
## 1369 -1.325507419  1.068454375
## 1373  0.289558127 -0.769197689
## 1374  0.289558127 -0.769197689
## 1377  0.289558127 -0.769197689
## 1378  0.215239001 -0.685049959
## 1382 -0.163788541 -0.255896534
## 1385  4.759569446  2.129463434
## 1390  1.006598503 -0.304219658
## 1391 -1.029315856  0.759900764
## 1392  2.818337121  1.290819340
## 1393 -1.266095080  1.028398555
## 1395  0.289558127 -0.769197689
## 1400  4.093089609  1.697272315
## 1401  1.087285863  2.727841643
## 1404  0.289558127 -0.769197689
## 1406 -1.673502657  1.505608636
## 1408  0.845724060 -0.408541652
## 1414 -0.862639378  0.537805380
```

```
## 1423 -1.226552042  0.947416010
## 1425  0.928459488 -0.354890341
## 1426 -1.034251303  0.729683758
## 1430  0.289558127 -0.769197689
## 1431  3.233560440  1.139894804
## 1432  0.289558127 -0.769197689
## 1434  0.289558127 -0.769197689
## 1435 -0.154498650 -0.266415000
## 1436 -0.132202913 -0.291659319
## 1437 -1.403988955  1.148318717
## 1438 -0.969222068  0.656054494
## 1443  1.187795789 -1.798354717
## 1445  0.289558127 -0.769197689
## 1446  2.217753241  0.481175926
## 1449  0.289558127 -0.769197689
## 1450  0.602065656 -0.558275096
## 1452 -1.583442549  1.382002168
## 1458  4.885970794  2.211430715
## 1459  0.673368886 -0.287056933
## 1461 -1.190346235  0.939618276
## 1467 -1.536834392  1.298732785
## 1468 -1.398415020  1.142007637
## 1473 -0.370953105 -0.021334735
## 1483  0.289558127 -0.769197689
## 1491  0.289558127 -0.769197689
## 1493  0.365253338 -0.695297736
## 1498 -1.351965567  1.089415305
## 1502 -1.118726469  0.832212798
## 1507  0.289558127 -0.769197689
## 1510  0.289558127 -0.769197689
## 1518  0.307943778 -0.757275175
## 1523 -0.380242995 -0.010816269
## 1527 -1.468634577  1.223627759
## 1538  0.289558127 -0.769197689
## 1543  0.289558127 -0.769197689
## 1550 -1.400599058  0.204198013
## 1555  0.289558127 -0.769197689
## 1560 -1.126221222  0.833816574
## 1561 -1.124275329  0.835256830
## 1565 -0.717466029  0.371004057
## 1566  0.197588209 -0.665064873
## 1570 -0.209309006 -0.204356049
## 1575  0.177329030 -0.638527937
## 1580 -1.171966656  0.798317384
## 1584 -1.699592326  1.548553074
## 1588 -1.483882015  1.238777527
## 1589 -1.096493571  0.800157482
## 1594 -0.933275228 -2.868336634
## 1599  0.289558127 -0.769197689
## 1601  0.289558127 -0.769197689
```

```
## 1602  2.669117809  0.144270031
## 1604  0.289558127 -0.769197689
## 1608  2.615342936  0.739000283
## 1609  4.031239036  1.779579908
## 1610  0.935354107 -0.350419398
## 1614 -1.155019883  0.866423820
## 1616 -0.156356629 -0.264311307
## 1621 -1.616745340  1.413186911
## 1622 -1.482827038  1.281304910
## 1623  0.289558127 -0.769197689
## 1627  0.289558127 -0.769197689
## 1630 -1.028677369  0.723372678
## 1643  3.255336798 -0.537103467
## 1652  0.289558127 -0.769197689
## 1657 -1.046962520  0.772723203
## 1662  0.634289077 -0.545650559
## 1664  0.289558127 -0.769197689
## 1665  0.289558127 -0.769197689
## 1667 -1.403059966  1.147266870
## 1669 -1.242652630  1.013730951
## 1670  3.447293629  1.278494024
## 1676  0.150209766 -0.611420695
## 1677  0.289558127 -0.769197689
## 1682  0.289558127 -0.769197689
## 1687  3.923767041  2.948933470
## 1689  3.860970769  1.546750581
## 1691  0.237534739 -0.710294278
## 1695  0.263546433 -0.739745983
## 1696 -1.724638642  1.588876772
## 1698  0.289558127 -0.769197689
## 1700  0.289558127 -0.769197689
## 1701 -0.551176985  0.182723511
## 1703 -1.069878858  0.812125559
## 1704 -0.567898788  0.201656750
## 1706  0.289558127 -0.769197689
## 1707 -1.588004015  1.366385494
## 1709  0.289558127 -0.769197689
## 1710 -1.749753301  1.629772647
## 1711  0.289558127 -0.769197689
## 1712  0.435527747 -0.661535724
## 1714  0.009932416 -0.452591853
## 1717  0.289558127 -0.769197689
## 1719  0.289558127 -0.769197689
## 1720 -0.337509498 -0.059201214
## 1722  0.289558127 -0.769197689
## 1723  0.289558127 -0.769197689
## 1725 -1.225623053  0.946364164
## 1727  0.289558127 -0.769197689
## 1728 -1.608585342  1.399899443
## 1732 -1.541981444  1.309418514
```

```
## 1736  0.262617444 -0.738694137
## 1746 -1.486668982  1.241933067
## 1750 -0.915340702  0.595047390
## 1751  0.289558127 -0.769197689
## 1752  0.289558127 -0.769197689
## 1766  0.240321706 -0.713449818
## 1769  0.289558127 -0.769197689
## 1771  0.289558127 -0.769197689
## 1775  0.233818782 -0.706086891
## 1776  0.289558127 -0.769197689
## 1777  0.289558127 -0.769197689
## 1786 -1.741049304  1.615599347
## 1787  3.858672563  1.545260267
## 1790 -0.994304773  0.684454353
## 1792 -1.168331060  0.895934398
## 1795 -1.697257316  1.544289933
## 1799 -0.355522957 -0.038625574
## 1800 -0.084824470 -0.345303497
## 1801 -1.068058953  0.769986585
## 1810 -0.257616438 -0.149660024
## 1817  0.289558127 -0.769197689
## 1825  0.289558127 -0.769197689
## 1831 -0.036591712 -3.343268423
## 1833 -1.285252297  1.223357578
## 1843  0.192324040 -0.649118663
## 1847  0.289558127 -0.769197689
## 1848  0.289558127 -0.769197689
## 1850 -0.709105128  0.361537438
## 1863 -0.118268077 -0.307437019
## 1870  4.653851954  2.060908980
## 1871  4.054069519  0.849504184
## 1874 -1.339430940  1.099422962
## 1877 -0.248326547 -0.160178490
## 1880 -0.090398404 -0.338992417
## 1881  3.654132199  1.412622303
## 1890  0.876047175  0.370580721
## 1897  0.289558127 -0.769197689
## 1900  0.289558127 -0.769197689
## 1917  1.589932671  2.964058038
## 1920 -1.081374258  0.805034543
## 1921 -1.343565216  1.118138173
## 1925 -0.231604744 -0.179111730
## 1928 -0.316142749 -0.083393686
## 1929  0.289558127 -0.769197689
## 1933  0.289558127 -0.769197689
## 1934  3.365639656  1.354576805
## 1938  0.289558127 -0.769197689
## 1939  0.289558127 -0.769197689
## 1942  0.289558127 -0.769197689
## 1943  3.470384094  3.829340588
```

```
## 1944  1.689630587  2.739649433
## 1947  0.289558127 -0.769197689
## 1951 -1.514206566  1.283548134
## 1952 -0.216740918 -0.195941276
## 1959  0.289558127 -0.769197689
## 1964  0.289558127 -0.769197689
## 1965  4.289670969 -3.946933941
## 1968  0.280268236 -0.758679223
## 1969 -0.300659086 -0.050995738
## 1973  0.289558127 -0.769197689
## 1974  0.289558127 -0.769197689
## 1978  0.106547279 -0.561983903
## 1979  0.289558127 -0.769197689
## 1983  2.811982771  2.447993964
## 1984  2.764726348  0.835870706
## 1987  0.289558127 -0.769197689
## 1988  0.317136603 -0.751313919
## 1989  3.888549245  1.564634351
## 1993 -0.066289787 -0.280347807
## 1995 -1.573950686  1.343501521
## 1999  2.655275355  1.762998073
## 2000 -0.975680346  0.664266498
## 2003 -0.252042503 -0.155971104
```

```
mylDA2 <- lda(formula =  V1 ~.,data=test)
mylDA2
```

```
## Call:
## lda(V1 ~ ., data = test)
##
## Prior probabilities of groups:
##         2         3         5
## 0.3778626 0.3167939 0.3053435
##
## Group means:
##           V3         V5          V7        V15
## 2 -0.9503838 -0.8052374 -0.36917172 -0.9898788
## 3 -0.9611627 -0.6965482 -0.03910241 -0.9705301
## 5 -0.9868750 -0.9583687 -0.76425000 -0.5616562
##
## Coefficients of linear discriminants:
##            LD1        LD2
## V3   1.1541120  1.6637425
## V5   0.1541224 -0.3413991
## V7   0.9146936 -0.9803365
## V15 -2.0420320 -1.4628063
##
## Proportion of trace:
##    LD1    LD2
## 0.9377 0.0623
```

```
summary(mylDA2)
```

```
##         Length Class  Mode
## prior    3     -none- numeric
## counts   3     -none- numeric
## means   12     -none- numeric
## scaling  8     -none- numeric
## lev      3     -none- character
## svd      2     -none- numeric
## N        1     -none- numeric
## call     3     -none- call
## terms    3     terms  call
## xlevels  0     -none- list
```

2.:Apply the kNN method to the entire data set of the Home Equity Loan data. Find the best value of k using the cross validation error.

```
data = read.csv("hmeq.csv",stringsAsFactors = FALSE)
# Convert the variable 'BAD' from integer to factor
data$BAD <- as.factor(data$BAD)
# Convert integer variables to numeric variables
data$LOAN <- as.numeric(data$LOAN)
data$NINQ <- as.numeric(data$NINQ)
data$CLNO <- as.numeric(data$CLNO)
data$DEROG <- as.numeric(data$DEROG)
data$DELINQ <- as.numeric(data$DELINQ)
str(data)
```

```
## 'data.frame':    5960 obs. of  13 variables:
##  $ BAD    : Factor w/ 2 levels "0","1": 2 2 2 2 1 2 2 2 2 2 ...
##  $ LOAN   : num  1100 1300 1500 1500 1700 1700 1800 1800 2000 2000 ...
##  $ MORTDUE: num  25860 70053 13500 NA 97800 ...
##  $ VALUE  : num  39025 68400 16700 NA 112000 ...
##  $ REASON : chr  "HomeImp" "HomeImp" "HomeImp" "" ...
##  $ JOB    : chr  "Other" "Other" "Other" "" ...
##  $ YOJ    : num  10.5 7 4 NA 3 9 5 11 3 16 ...
##  $ DEROG  : num  0 0 0 NA 0 0 3 0 0 0 ...
##  $ DELINQ : num  0 2 0 NA 0 0 2 0 2 0 ...
##  $ CLAGE  : num  94.4 121.8 149.5 NA 93.3 ...
##  $ NINQ   : num  1 0 1 NA 0 1 1 0 1 0 ...
##  $ CLNO   : num  9 14 10 NA 14 8 17 8 12 13 ...
##  $ DEBTINC: num  NA NA NA NA NA ...
```

```
summary(data)
```

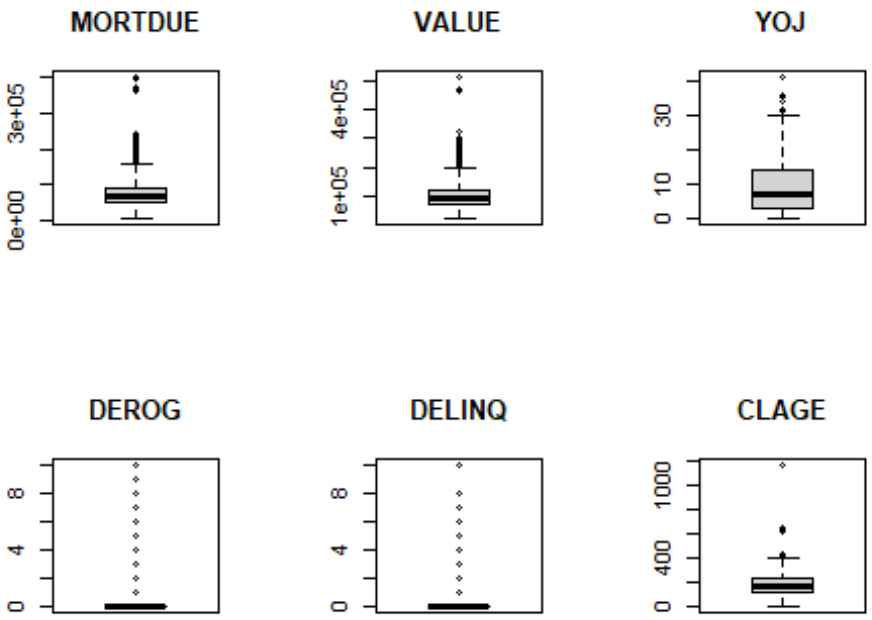
```
##  BAD           LOAN          MORTDUE           VALUE           REASON
##  0:4771   Min.   : 1100   Min.   :  2063   Min.   :  8000   Length:5960
##  1:1189   1st Qu.:11100   1st Qu.: 46276   1st Qu.: 66076   Class
:character
##           Median :16300   Median : 65019   Median : 89236   Mode
```

```
:character
##           Mean   :18608   Mean   : 73761   Mean   :101776
##           3rd Qu.:23300   3rd Qu.: 91488   3rd Qu.:119824
##           Max.   :89900   Max.   :399550   Max.   :855909
##                           NA's   :518      NA's   :112
##      JOB                 YOJ             DEROG             DELINQ
##  Length:5960        Min.   : 0.000   Min.   : 0.0000   Min.   : 0.0000
##  Class :character   1st Qu.: 3.000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Mode  :character   Median : 7.000   Median : 0.0000   Median : 0.0000
##                     Mean   : 8.922   Mean   : 0.2546   Mean   : 0.4494
##                     3rd Qu.:13.000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##                     Max.   :41.000   Max.   :10.0000   Max.   :15.0000
##                     NA's   :515      NA's   :708       NA's   :580
##      CLAGE             NINQ             CLNO         DEBTINC
##  Min.   :   0.0   Min.   : 0.000   Min.   : 0.0   Min.   :  0.5245
##  1st Qu.: 115.1   1st Qu.: 0.000   1st Qu.:15.0   1st Qu.: 29.1400
##  Median : 173.5   Median : 1.000   Median :20.0   Median : 34.8183
##  Mean   : 179.8   Mean   : 1.186   Mean   :21.3   Mean   : 33.7799
##  3rd Qu.: 231.6   3rd Qu.: 2.000   3rd Qu.:26.0   3rd Qu.: 39.0031
##  Max.   :1168.2   Max.   :17.000   Max.   :71.0   Max.   :203.3121
##  NA's   :308      NA's   :510      NA's   :222    NA's   :1267
```

```
data<-na.omit(data)
dim(data)
```

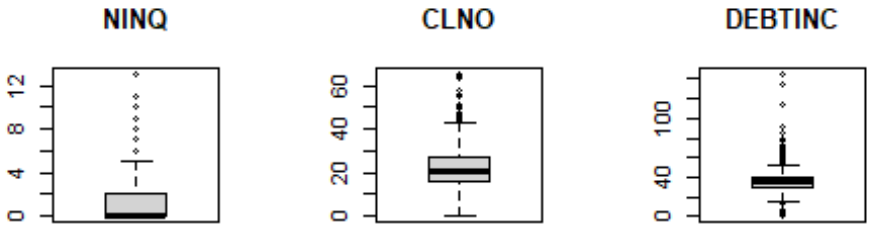
```
## [1] 3515   13
```

```
par(mfrow = c(2,3))

boxplot(data$MORTDUE, main = "MORTDUE")
boxplot(data$VALUE, main = "VALUE")
boxplot(data$YOJ, main = "YOJ")
boxplot(data$DEROG, main = "DEROG")
boxplot(data$DELINQ, main = "DELINQ")
boxplot(data$CLAGE, main = "CLAGE")
```

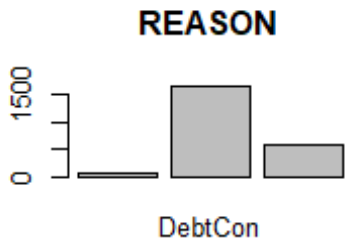


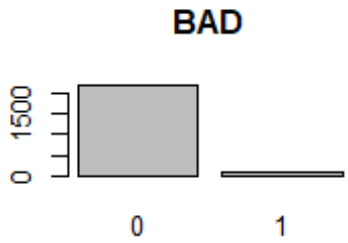

```
boxplot(data$NINQ, main = "NINQ")
boxplot(data$CLNO, main = "CLNO")
boxplot(data$DEBTINC, main = "DEBTINC")

#remove the outliers
outliers_remover <- function(a){
  df <- a
  aa<-c()
  count<-1
  for(i in 1:ncol(df)){
    if(is.numeric(df[,i])){
      Q3 <- quantile(df[,i], 0.75, na.rm = TRUE)
      Q1 <- quantile(df[,i], 0.25, na.rm = TRUE)
      IQR <- Q3 - Q1
      upper <- Q3 + 1.5 * IQR
      lower <- Q1 - 1.5 * IQR
      for(j in 1:nrow(df)){
        if(is.na(df[j,i]) == TRUE){
          next
        }
        else if(df[j,i] > upper | df[j,i] < lower){
          aa[count]<-j
          count<-count+1
        }
      }
    }
```

```
  }
  df<-df[-aa,]
}

data <- outliers_remover(data)
dim(data)

## [1] 2302   13

par(mfrow = c(2,2))
```

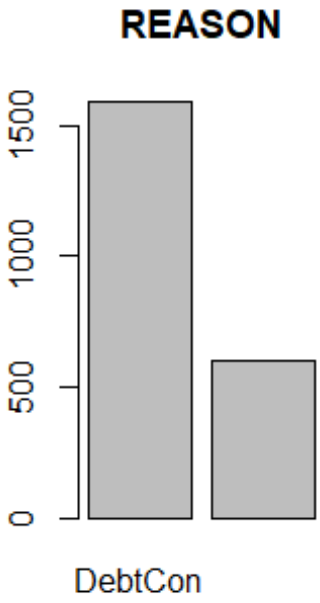


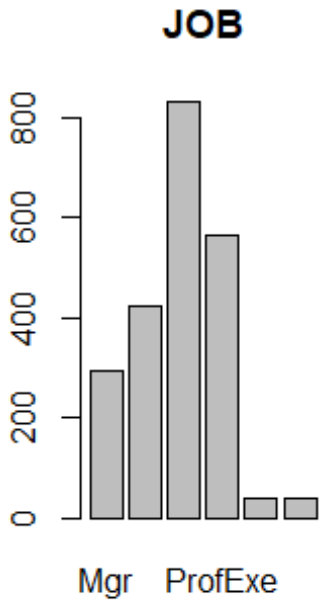

```
barplot(table(data$REASON), main = "REASON")
barplot(table(data$JOB), main = "JOB")
barplot(table(data$BAD), main = "BAD")



#correct the errors
data <- data[!(data$REASON ==""),]
data <- data[!(data$JOB ==""),]
par(mfrow = c(1,2))
```






```r
barplot(table(data$REASON), main = "REASON")
barplot(table(data$JOB), main = "JOB")
```





```
dim(data)
```

```
## [1] 2184   13
```

```
cor(data[c("LOAN","MORTDUE", "VALUE", "YOJ", "DEROG", "DELINQ", "CLAGE",
"NINQ", "CLNO", "DEBTINC")])
```

```
## Warning in stats::cor(x, y, ...): the standard deviation is zero
```

```
##                LOAN     MORTDUE      VALUE          YOJ DEROG DELINQ
## LOAN    1.000000000  0.17852824 0.34809670  0.077232083    NA     NA
## MORTDUE 0.178528245  1.00000000 0.84001285 -0.070994141    NA     NA
## VALUE   0.348096704  0.84001285 1.00000000  0.024763530    NA     NA
## YOJ     0.077232083 -0.07099414 0.02476353  1.000000000    NA     NA
## DEROG            NA          NA         NA           NA     1     NA
## DELINQ           NA          NA         NA           NA    NA      1
## CLAGE   0.067864250  0.07690745 0.20400150  0.223298794    NA     NA
## NINQ    0.005072811  0.06095620 0.03552415 -0.023927232    NA     NA
## CLNO    0.097396871  0.30790517 0.29373600  0.004381988    NA     NA
## DEBTINC 0.156149422  0.23864832 0.18322318 -0.035695438    NA     NA
##               CLAGE         NINQ        CLNO     DEBTINC
## LOAN     0.06786425  0.005072811 0.097396871  0.15614942
## MORTDUE  0.07690745  0.060956203 0.307905172  0.23864832
## VALUE    0.20400150  0.035524149 0.293735998  0.18322318
## YOJ      0.22329879 -0.023927232 0.004381988 -0.03569544
## DEROG            NA           NA          NA          NA
## DELINQ           NA           NA          NA          NA
## CLAGE    1.00000000 -0.034803256 0.197475140  0.04023405
## NINQ    -0.03480326  1.000000000 0.147204757  0.24175019
## CLNO     0.19747514  0.147204757 1.000000000  0.16301041
## DEBTINC  0.04023405  0.241750193 0.163010411  1.00000000
```

```
data$DEROG <- NULL
data$DELINQ <- NULL
#  multicollinearity is present between variables MORTDUE and VALUE
data$MORTDUE <- NULL # delete MORTDUE
dim(data)
```

```
## [1] 2184   10
```

```
# The training data is created
input_ones <- data[which(data$BAD == 1), ] #all 1's
input_zeros <- data[which(data$BAD == 0), ] # all 0's
set.seed(100) # for repeatability of sample
input_ones_training_rows <- sample(1:nrow(input_ones), 0.7 *
nrow(input_ones)) #1's for training
input_zeros_training_rows <- sample(1:nrow(input_zeros), 0.7 *
nrow(input_zeros)) #0's for training
#pick as many as 0's and 1's
training_ones <- input_ones[input_ones_training_rows, ]
training_zeros <- input_zeros[input_zeros_training_rows, ]
```

```r
# We row bind the  0's and 1's for training data
trainingData <- rbind(training_ones, training_zeros)

# The test data is created
test_ones <- input_ones[-input_ones_training_rows, ]
test_zeros <- input_zeros[-input_zeros_training_rows, ]
# We row bind the  0's and 1's for test data
testData <- rbind(test_ones, test_zeros)

table(trainingData$BAD)

##
##    0    1
## 1456   72

prop.table(table(trainingData$BAD))

##
##          0          1
## 0.95287958 0.04712042

dim(trainingData)

## [1] 1528   10

set.seed(111)
trControl <- trainControl(method  = "cv",
                          number  = 5)
s = function(seeds_list,k){
  seeds_list = lapply(seeds_list,"[",1:k)
  seeds_list[[length(seeds_list)+1]] = 999
  seeds_list
}

k = 1
model0 <- train(BAD~.,data=data,method="knn",
                trControl = trainControl(method = 'LOOCV'),tuneGrid =
expand.grid(k = 1:k))
model0

## k-Nearest Neighbors
##
## 2184 samples
##    9 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 2183, 2183, 2183, 2183, 2183, 2183, ...
## Resampling results:
##
```

```
##   Accuracy   Kappa
##   0.9285714  0.1826347
##
## Tuning parameter 'k' was held constant at a value of 1
```

```
k = 5

model <- train(BAD~.,data=data,method="knn",
               trControl = trainControl(method = 'LOOCV'),tuneGrid =
expand.grid(k = 1:k))
model
```

```
## k-Nearest Neighbors
##
## 2184 samples
##    9 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 2183, 2183, 2183, 2183, 2183, 2183, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   1  0.9285714  0.18263473
##   2  0.9299451  0.15425112
##   3  0.9491758  0.17075774
##   4  0.9523810  0.12218650
##   5  0.9510073  0.06101128
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 4.
```

```
k = 10

model1 <- train(BAD~.,data=data,method="knn",
               trControl = trainControl(method = 'LOOCV'),tuneGrid =
expand.grid(k = 1:k))
model1
```

```
## k-Nearest Neighbors
##
## 2184 samples
##    9 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 2183, 2183, 2183, 2183, 2183, 2183, ...
## Resampling results across tuning parameters:
##
```

```
##   k   Accuracy   Kappa
##    1  0.9285714  0.18263473
##    2  0.9299451  0.16262705
##    3  0.9491758  0.17075774
##    4  0.9514652  0.13182527
##    5  0.9510073  0.06101128
##    6  0.9523810  0.06506849
##    7  0.9519231  0.06369427
##    8  0.9523810  0.04960836
##    9  0.9528388  0.03480589
##   10  0.9528388  0.03480589
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 10.
```

```
k = 20

model2 <- train(BAD~.,data=data,method="knn",
                trControl = trainControl(method = 'LOOCV'),tuneGrid =
expand.grid(k = 1:k))
model2
```

```
## k-Nearest Neighbors
##
## 2184 samples
##    9 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 2183, 2183, 2183, 2183, 2183, 2183, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.9285714  0.18263473
##    2  0.9331502  0.18065206
##    3  0.9491758  0.17075774
##    4  0.9482601  0.10755923
##    5  0.9510073  0.06101128
##    6  0.9505495  0.05970149
##    7  0.9519231  0.06369427
##    8  0.9523810  0.04960836
##    9  0.9528388  0.03480589
##   10  0.9523810  0.03362832
##   11  0.9532967  0.03600360
##   12  0.9528388  0.01815706
##   13  0.9532967  0.03600360
##   14  0.9528388  0.01815706
##   15  0.9523810  0.00000000
##   16  0.9528388  0.01815706
```

```
##   17  0.9523810  0.00000000
##   18  0.9523810  0.00000000
##   19  0.9523810  0.00000000
##   20  0.9523810  0.00000000
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 13.
```

```
# The optimal value for k is 13, which gives the highest accuracy in
predicting the defaulters
```

3. : Split randomly the Home Equity Loan data into 3 parts (50% training, 25% validating, and 25% testing). Repeat this 3 times and compare the training, validating, and testing error of (i) the logistic regression model, (ii) LDA, and (iii) the tree model. Compare the results.

```
data1 = read.csv("hmeq.csv")
# The variable 'BAD' is converted from integer to factor
data1$BAD <- as.factor(data1$BAD)
# Integer variables are converted to numeric variables
data1$LOAN <- as.numeric(data1$LOAN)
data1$NINQ <- as.numeric(data1$NINQ)
data1$CLNO <- as.numeric(data1$CLNO)
data1$DEROG <- as.numeric(data1$DEROG)
data1$DELINQ <- as.numeric(data1$DELINQ)
str(data1)
```

```
## 'data.frame':    5960 obs. of  13 variables:
##  $ BAD    : Factor w/ 2 levels "0","1": 2 2 2 2 1 2 2 2 2 2 ...
##  $ LOAN   : num  1100 1300 1500 1500 1700 1700 1800 1800 2000 2000 ...
##  $ MORTDUE: num  25860 70053 13500 NA 97800 ...
##  $ VALUE  : num  39025 68400 16700 NA 112000 ...
##  $ REASON : chr  "HomeImp" "HomeImp" "HomeImp" "" ...
##  $ JOB    : chr  "Other" "Other" "Other" "" ...
##  $ YOJ    : num  10.5 7 4 NA 3 9 5 11 3 16 ...
##  $ DEROG  : num  0 0 0 NA 0 0 3 0 0 0 ...
##  $ DELINQ : num  0 2 0 NA 0 0 2 0 2 0 ...
##  $ CLAGE  : num  94.4 121.8 149.5 NA 93.3 ...
##  $ NINQ   : num  1 0 1 NA 0 1 1 0 1 0 ...
##  $ CLNO   : num  9 14 10 NA 14 8 17 8 12 13 ...
##  $ DEBTINC: num  NA NA NA NA NA ...
```

```
summary(data1)
```

```
##  BAD           LOAN          MORTDUE           VALUE           REASON
##  0:4771   Min.   : 1100   Min.   :  2063   Min.   :  8000   Length:5960
##  1:1189   1st Qu.:11100   1st Qu.: 46276   1st Qu.: 66076   Class
```

```
:character
##           Median :16300   Median : 65019   Median : 89236   Mode
:character
##           Mean   :18608   Mean   : 73761   Mean   :101776
##           3rd Qu.:23300   3rd Qu.: 91488   3rd Qu.:119824
##           Max.   :89900   Max.   :399550   Max.   :855909
##                           NA's   :518      NA's   :112
##      JOB                 YOJ             DEROG             DELINQ
##  Length:5960        Min.   : 0.000   Min.   : 0.0000   Min.   : 0.0000
##  Class :character   1st Qu.: 3.000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Mode  :character   Median : 7.000   Median : 0.0000   Median : 0.0000
##                     Mean   : 8.922   Mean   : 0.2546   Mean   : 0.4494
##                     3rd Qu.:13.000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##                     Max.   :41.000   Max.   :10.0000   Max.   :15.0000
##                     NA's   :515      NA's   :708       NA's   :580
##      CLAGE             NINQ             CLNO         DEBTINC
##  Min.   :   0.0   Min.   : 0.000   Min.   : 0.0   Min.   :  0.5245
##  1st Qu.: 115.1   1st Qu.: 0.000   1st Qu.:15.0   1st Qu.: 29.1400
##  Median : 173.5   Median : 1.000   Median :20.0   Median : 34.8183
##  Mean   : 179.8   Mean   : 1.186   Mean   :21.3   Mean   : 33.7799
##  3rd Qu.: 231.6   3rd Qu.: 2.000   3rd Qu.:26.0   3rd Qu.: 39.0031
##  Max.   :1168.2   Max.   :17.000   Max.   :71.0   Max.   :203.3121
##  NA's   :308      NA's   :510      NA's   :222    NA's   :1267
```

```
data1<-na.omit(data1)
dim(data1)
```

```
## [1] 3515   13
```

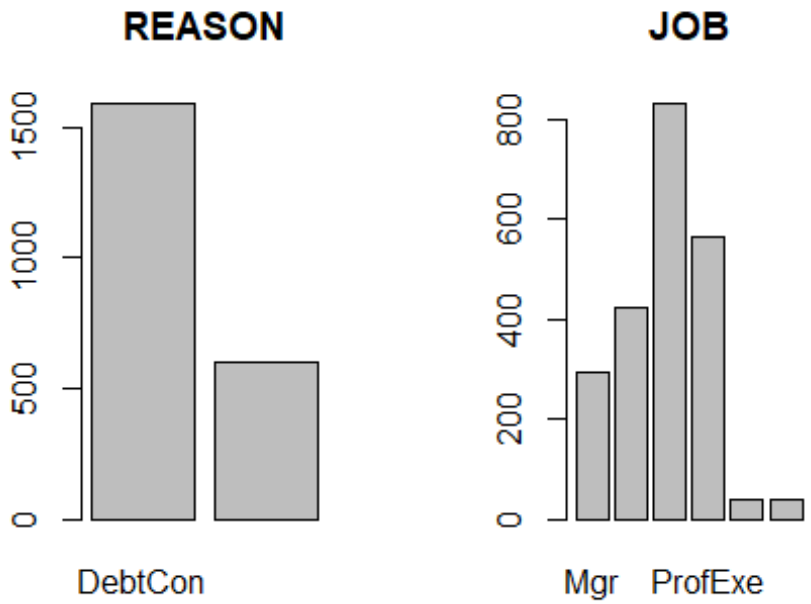
```
#The outliers are removed
outliers_remover <- function(a){
  df <- a
  aa<-c()
  count<-1
  for(i in 1:ncol(df)){
    if(is.numeric(df[,i])){
      Q3 <- quantile(df[,i], 0.75, na.rm = TRUE)
      Q1 <- quantile(df[,i], 0.25, na.rm = TRUE)
      IQR <- Q3 - Q1
      upper <- Q3 + 1.5 * IQR
      lower <- Q1 - 1.5 * IQR
      for(j in 1:nrow(df)){
        if(is.na(df[j,i]) == TRUE){
          next
        }
        else if(df[j,i] > upper | df[j,i] < lower){
          aa[count]<-j
          count<-count+1
        }
      }
```

```
    }
  }
  df<-df[-aa,]
}

data1 <- outliers_remover(data1)
dim(data1)
```

```
## [1] 2302   13
```

```
par(mfrow = c(2,2))
barplot(table(data1$REASON), main = "REASON")
barplot(table(data1$JOB), main = "JOB")
barplot(table(data1$BAD), main = "BAD")



#The errors are corrected

data1 <- data1[!(data1$REASON ==""),]
data1 <- data1[!(data1$JOB ==""),]
par(mfrow = c(1,2))
```


```
barplot(table(data1$REASON), main = "REASON")
barplot(table(data1$JOB), main = "JOB")
```




```
dim(data1)
```

```
## [1] 2184   13
```

```
#Correlation Matrix is created
cor(data1[c("LOAN", "MORTDUE", "VALUE", "YOJ", "DEROG", "DELINQ", "CLAGE",
"NINQ", "CLNO", "DEBTINC")])
```

```
## Warning in stats::cor(x, y, ...): the standard deviation is zero
```

```
##                LOAN     MORTDUE      VALUE          YOJ DEROG DELINQ
## LOAN    1.000000000  0.17852824 0.34809670  0.077232083    NA     NA
## MORTDUE 0.178528245  1.00000000 0.84001285 -0.070994141    NA     NA
## VALUE   0.348096704  0.84001285 1.00000000  0.024763530    NA     NA
## YOJ     0.077232083 -0.07099414 0.02476353  1.000000000    NA     NA
## DEROG            NA          NA         NA           NA     1     NA
## DELINQ           NA          NA         NA           NA    NA      1
## CLAGE   0.067864250  0.07690745 0.20400150  0.223298794    NA     NA
## NINQ    0.005072811  0.06095620 0.03552415 -0.023927232    NA     NA
## CLNO    0.097396871  0.30790517 0.29373600  0.004381988    NA     NA
## DEBTINC 0.156149422  0.23864832 0.18322318 -0.035695438    NA     NA
##               CLAGE         NINQ        CLNO     DEBTINC
## LOAN     0.06786425  0.005072811 0.097396871  0.15614942
## MORTDUE  0.07690745  0.060956203 0.307905172  0.23864832
## VALUE    0.20400150  0.035524149 0.293735998  0.18322318
## YOJ      0.22329879 -0.023927232 0.004381988 -0.03569544
## DEROG            NA           NA          NA          NA
```

```
## DELINQ           NA           NA          NA          NA
## CLAGE    1.00000000 -0.034803256 0.197475140  0.04023405
## NINQ    -0.03480326  1.000000000 0.147204757  0.24175019
## CLNO     0.19747514  0.147204757 1.000000000  0.16301041
## DEBTINC  0.04023405  0.241750193 0.163010411  1.00000000
```

```
data1$DEROG <- NULL
data1$DELINQ <- NULL
# there is multicollinearity between MORTDUE and VALUE
data1$MORTDUE <- NULL
dim(data1)
```

```
## [1] 2184   10
```

```
# The training set and validation set data is created
input_ones <- data1[which(data1$BAD == 1), ] #all 1's
input_zeros <- data1[which(data1$BAD == 0), ] # all 0's
set.seed(100) #This is for the repeatability of sample
input_ones_training_rows <- sample(1:nrow(input_ones), 0.5 *
nrow(input_ones)) #1's for training
input_zeros_training_rows <- sample(1:nrow(input_zeros), 0.5 *
nrow(input_zeros)) #0's for training
input_ones_validation_rows <- sample(1:nrow(input_ones), 0.25 *
nrow(input_ones)) #1's for validation
input_zeros_validation_rows <- sample(1:nrow(input_zeros), 0.25 *
nrow(input_zeros)) #0's for validation
#pick as many as 0's and 1's
training_ones <- input_ones[input_ones_training_rows, ]
training_zeros <- input_zeros[input_zeros_training_rows, ]
validation_ones <- input_ones[input_ones_validation_rows, ]
validation_zeros <- input_zeros[input_zeros_validation_rows, ]
# We row bind the 0's and 1's
trainingData <- rbind(training_ones, training_zeros)
validationData <- rbind(validation_ones, validation_zeros)

#We create the test set data
test_ones <- input_ones[-input_ones_training_rows, ]
test_zeros <- input_zeros[-input_zeros_training_rows, ]
#1's and 0's and row binded
testData <- rbind(test_ones, test_zeros)

table(trainingData$BAD)
```

```
##
##    0    1
## 1040   52
```

```
prop.table(table(trainingData$BAD))
```

```
##
##          0          1
## 0.95238095 0.04761905

treeMod <- rpart(BAD ~., data = trainingData)
pred_treeMod <- predict(treeMod, newdata = testData)


accuracy.meas(testData$BAD, pred_treeMod[,2])

##
## Call:
## accuracy.meas(response = testData$BAD, predicted = pred_treeMod[,
##     2])
##
## Examples are labelled as positive when predicted is greater than 0.5
##
## precision: 0.281
## recall: 0.173
## F: 0.107

#We re-sample the data set

data_balanced_over <- ovun.sample(BAD ~., data = trainingData, method =
"over", N = 2912)$data
table(data_balanced_over$BAD)

##
##    0    1
## 1040 1872

data_balanced_both <- ovun.sample(BAD ~., data = trainingData, method =
"both", p = 0.5, N = 1528)$data
table(data_balanced_both$BAD)

##
##   0   1
## 770 758

#Decision Tree Models are built
tree.over <- rpart(BAD ~., data = data_balanced_over)
tree.both <- rpart(BAD ~., data = data_balanced_both)


# We make predictions on test data
pred_tree.over <- predict(tree.over, newdata = testData)
pred_tree.both <- predict(tree.both, newdata = testData)


#ROC Curves are created to show error rates for Decision Tree Algorithm
par(mfrow = c(2,2))
```

```r
roc.curve(testData$BAD, pred_tree.over[,2], col = "BLACK", main = "ROC curve
of oversampling")
```

```
## Area under the curve (AUC): 0.686
```

```r
roc.curve(testData$BAD, pred_tree.both[,2], col = "RED", main = "ROC curve of
balanced sampling")
```

```
## Area under the curve (AUC): 0.686
```

```r
# We conduct a logistic regression on the training data
logisticModel <- glm(BAD ~., data = trainingData, family = binomial(link =
"logit"))
pred_logit <- predict(logisticModel, testData)
summary(logisticModel)
```

```
##
## Call:
## glm(formula = BAD ~ ., family = binomial(link = "logit"), data =
trainingData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1989  -0.2982  -0.1812  -0.1182   3.3995
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -3.715e+00  1.293e+00  -2.872  0.00407 **
## LOAN          -5.057e-05  2.467e-05  -2.050  0.04040 *
## VALUE         -3.340e-06  5.577e-06  -0.599  0.54928
## REASONHomeImp -3.206e-01  3.795e-01  -0.845  0.39819
## JOBOffice     -1.465e+00  7.252e-01  -2.021  0.04331 *
## JOBOther      -2.109e-01  4.754e-01  -0.444  0.65734
## JOBProfExe    -5.771e-01  5.759e-01  -1.002  0.31632
## JOBSales       1.804e+00  7.913e-01   2.280  0.02262 *
## JOBSelf        1.403e+00  9.687e-01   1.448  0.14748
## YOJ           -2.701e-02  2.812e-02  -0.961  0.33678
## CLAGE         -1.155e-02  2.780e-03  -4.154 3.27e-05 ***
## NINQ           1.347e-01  1.391e-01   0.968  0.33304
## CLNO          -2.303e-02  2.046e-02  -1.125  0.26050
## DEBTINC        1.261e-01  3.218e-02   3.918 8.92e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 418.11  on 1091  degrees of freedom
## Residual deviance: 334.96  on 1078  degrees of freedom
## AIC: 362.96
```

```
##
## Number of Fisher Scoring iterations: 7

summary(pred_logit)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -7.210  -4.650  -3.801  -3.754  -2.722   1.403

#Doing a logistic Regression on the validation set to fine tune the model
mylogit1 <- glm(BAD ~., data=validationData , family =binomial(link =
"logit"))
pred_logit1 <- predict(mylogit1, testData)
summary(mylogit1)

##
## Call:
## glm(formula = BAD ~ ., family = binomial(link = "logit"), data =
validationData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3731  -0.2875  -0.1570  -0.0769   3.5393
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -2.351e-01  1.714e+00  -0.137  0.89091
## LOAN          -1.196e-04  4.182e-05  -2.859  0.00425 **
## VALUE         -1.738e-05  9.448e-06  -1.839  0.06589 .
## REASONHomeImp -6.243e-01  5.359e-01  -1.165  0.24398
## JOBOffice      6.312e-01  9.198e-01   0.686  0.49253
## JOBOther       6.130e-01  8.263e-01   0.742  0.45820
## JOBProfExe     6.756e-01  9.557e-01   0.707  0.47961
## JOBSales       3.582e+00  1.503e+00   2.382  0.01720 *
## JOBSelf        3.295e+00  1.506e+00   2.188  0.02865 *
## YOJ           -1.850e-02  4.046e-02  -0.457  0.64754
## CLAGE         -1.375e-02  4.233e-03  -3.248  0.00116 **
## NINQ           3.947e-01  1.740e-01   2.269  0.02326 *
## CLNO          -4.557e-02  3.112e-02  -1.465  0.14304
## DEBTINC        7.374e-02  4.038e-02   1.826  0.06784 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 209.06  on 545  degrees of freedom
## Residual deviance: 157.79  on 532  degrees of freedom
## AIC: 185.79
##
## Number of Fisher Scoring iterations: 7

summary(pred_logit1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -10.197  -5.348  -4.016  -4.124  -2.840   2.466

# We look at the VIF to check for multicollinearity in the regression model
vif(mylogit1)

##             GVIF Df GVIF^(1/(2*Df))
## LOAN    1.141095  1        1.068221
## VALUE   1.300624  1        1.140449
## REASON  1.207600  1        1.098909
## JOB     1.766971  5        1.058578
## YOJ     1.123303  1        1.059860
## CLAGE   1.383429  1        1.176193
## NINQ    1.206555  1        1.098433
## CLNO    1.224231  1        1.106450
## DEBTINC 1.230260  1        1.109171

#Doing a logistic Regression on the test set after fine tuning the regression
model
mylogit2 <- glm(BAD ~ NINQ +VALUE +DEBTINC +CLAGE, data=testData, family =
binomial(link = "logit"))
pred_logit2 <- predict(mylogit2, testData)
summary(pred_logit2)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -6.8241 -4.1406 -3.3808 -3.4603 -2.6921 -0.9078

vif(mylogit2)

##     NINQ    VALUE  DEBTINC    CLAGE
## 1.054234 1.072927 1.102163 1.025191

summary(mylogit2)

##
## Call:
## glm(formula = BAD ~ NINQ + VALUE + DEBTINC + CLAGE, family = binomial(link
= "logit"),
##     data = testData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.7443  -0.3432  -0.2486  -0.1564   3.1113
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.442e+00  1.047e+00  -3.286 0.001016 **
## NINQ         6.422e-02  1.222e-01   0.526 0.599079
## VALUE       -2.071e-05  5.669e-06  -3.653 0.000259 ***
## DEBTINC      9.674e-02  2.872e-02   3.368 0.000757 ***
## CLAGE       -7.705e-03  2.322e-03  -3.318 0.000905 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 418.11  on 1091  degrees of freedom
## Residual deviance: 375.18  on 1087  degrees of freedom
## AIC: 385.18
##
## Number of Fisher Scoring iterations: 7
```

```
# Returns the cutoff that gives minimum misclassification error.
cutoff  <- optimalCutoff(testData$BAD, pred_logit)[1]
cutoff
```

```
## [1] 0.1628139
```

```
# Calculating Error Rate for Logistic Regression
misClassError(testData$BAD, pred_logit2, threshold = cutoff)
```

```
## [1] 0.0476
```

```
misClassError(testData$BAD, pred_logit2, threshold = 0.5)
```

```
## [1] 0.0476
```

```
confusionMatrix(testData$BAD, pred_logit, cutoff)
```

```
##      0  1
## 0 1040 50
## 1    0  2
```

```
specificity(testData$BAD, pred_logit2, cutoff)
```

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

```
sensitivity(testData$BAD, pred_logit2, cutoff)
```

```
## [1] 0
```

```
accuracy.meas(testData$BAD, pred_logit2, cutoff)
```

```
##
## Call:
## accuracy.meas(response = testData$BAD, predicted = pred_logit2,
##     threshold = cutoff)
##
## Examples are labelled as positive when predicted is greater than 0.1628139
##
## precision: NaN
## recall: 0.000
## F: NaN
```

```
plotROC(testData$BAD, pred_logit)
# We conduct an LDA analysis on training and validation datasets
Equity.lda <- lda(BAD ~., data = trainingData)
Equity.lda
```

```
## Call:
## lda(BAD ~ ., data = trainingData)
##
## Prior probabilities of groups:
##          0          1
## 0.95238095 0.04761905
##
## Group means:
##       LOAN    VALUE REASONHomeImp  JOBOffice  JOBOther JOBProfExe
JOBSales
## 0 17427.12 99054.77     0.2903846 0.21346154 0.3759615  0.2548077
0.01346154
## 1 14590.38 87962.25     0.2500000 0.05769231 0.5576923  0.1346154
0.07692308
##      JOBSelf      YOJ    CLAGE     NINQ     CLNO  DEBTINC
## 0 0.01442308 9.212500 177.3966 0.837500 21.05192 33.53054
## 1 0.03846154 6.365385 123.6583 1.115385 18.94231 37.09308
##
## Coefficients of linear discriminants:
##                            LD1
## LOAN           -5.266976e-05
## VALUE          -2.189875e-06
## REASONHomeImp -4.149341e-01
## JOBOffice      -8.040156e-01
## JOBOther       -2.561738e-02
## JOBProfExe     -3.256789e-01
## JOBSales        2.836376e+00
## JOBSelf         1.716459e+00
## YOJ            -1.200019e-02
## CLAGE          -7.648944e-03
## NINQ            9.624250e-02
## CLNO           -1.672174e-02
## DEBTINC         7.581878e-02
```

```
fit.hat <- predict(Equity.lda,testData)
summary(fit.hat)
```

```
##           Length Class  Mode
## class     1092   factor numeric
## posterior 2184   -none- numeric
## x         1092   -none- numeric
```

```
Equity.lda2 <- lda(BAD ~., data = validationData)
Equity.lda2
```

```
## Call:
## lda(BAD ~ ., data = validationData)
##
## Prior probabilities of groups:
##         0         1
## 0.95238095 0.04761905
##
## Group means:
##       LOAN    VALUE REASONHomeImp JOBOffice  JOBOther JOBProfExe
JOBSales
## 0 17832.50 100894.4     0.2923077 0.2000000 0.3692308  0.2596154
0.009615385
## 1 12780.77  80641.5     0.2692308 0.1923077 0.5000000  0.1538462
0.038461538
##      JOBSelf      YOJ    CLAGE      NINQ     CLNO  DEBTINC
## 0 0.02115385 9.303846 177.4786 0.8865385 20.88077 33.91158
## 1 0.03846154 6.538462 121.3760 1.3846154 16.96154 35.74296
##
## Coefficients of linear discriminants:
##                           LD1
## LOAN          -7.369884e-05
## VALUE         -3.871734e-06
## REASONHomeImp -5.007599e-01
## JOBOffice      1.416515e-01
## JOBOther       3.654024e-01
## JOBProfExe     3.388812e-01
## JOBSales       2.784490e+00
## JOBSelf        1.718807e+00
## YOJ           -6.706146e-03
## CLAGE         -7.750737e-03
## NINQ           2.295962e-01
## CLNO          -4.034104e-02
## DEBTINC        5.219889e-02
```

```
fit.hat2 <- predict(Equity.lda2,testData)
# There are vast differences in the means for the variable JOB
# We remove the variable JOB from the LDA analysis on the test to see if it
improves accuracy of model
Equity.lda3 <- lda(BAD ~ LOAN+VALUE+REASON+YOJ+CLAGE+NINQ+CLNO+DEBTINC, data
= testData)
Equity.lda3
```

```
## Call:
## lda(BAD ~ LOAN + VALUE + REASON + YOJ + CLAGE + NINQ + CLNO +
##     DEBTINC, data = testData)
##
## Prior probabilities of groups:
##         0         1
## 0.95238095 0.04761905
##
```

```
## Group means:
##       LOAN    VALUE REASONHomeImp      YOJ    CLAGE      NINQ     CLNO
DEBTINC
## 0 17446.06 98186.70     0.2548077 8.928846 175.2786 0.8788462 20.36442
33.73609
## 1 13753.85 79327.69     0.3076923 6.269231 134.3807 1.0961538 18.28846
36.10330
##
## Coefficients of linear discriminants:
##                           LD1
## LOAN          -5.174706e-05
## VALUE         -1.038321e-05
## REASONHomeImp  1.315388e-01
## YOJ           -2.734217e-02
## CLAGE         -6.293539e-03
## NINQ           6.419849e-02
## CLNO          -2.055394e-02
## DEBTINC        9.110237e-02

fit.hat3 <- predict(Equity.lda3,testData)
# It turns out that re-sampling the data improves the efficiency of the LDA
model
Equity.lda4 <- lda(BAD ~., data =data_balanced_both )
Equity.lda4

## Call:
## lda(BAD ~ ., data = data_balanced_both)
##
## Prior probabilities of groups:
##         0         1
## 0.5039267 0.4960733
##
## Group means:
##       LOAN     VALUE REASONHomeImp  JOBOffice  JOBOther JOBProfExe
JOBSales
## 0 17284.42 100964.58     0.2987013 0.21818182 0.3805195  0.2532468
0.01038961
## 1 15058.05  88857.97     0.2546174 0.05277045 0.5501319  0.1477573
0.08179420
##      JOBSelf      YOJ   CLAGE      NINQ     CLNO  DEBTINC
## 0 0.01948052 8.990909 174.463 0.8402597 21.07792 33.59819
## 1 0.04089710 6.362797 124.857 1.1015831 19.34433 37.06649
##
## Coefficients of linear discriminants:
##                           LD1
## LOAN          -2.900461e-05
## VALUE         -7.297490e-07
## REASONHomeImp -2.406279e-01
## JOBOffice     -1.308094e+00
## JOBOther       1.186391e-01
```

```
## JOBProfExe    -2.287448e-01
## JOBSales       1.475184e+00
## JOBSelf        8.979253e-01
## YOJ           -3.475800e-02
## CLAGE         -8.837612e-03
## NINQ           2.644739e-03
## CLNO          -4.082716e-03
## DEBTINC        8.761021e-02
```

 # Out of the three models, the decision tree had the lowest error rate, followed by the logistic regression model.