Crosswalk Project

Jagannathan Govindan

September 20, 2018

library(ggplot2)

## Warning: package 'ggplot2' was built under R version 3.5.1

library(readr)  
  
library(ggplot2)  
library(dplyr)

##   
## Attaching package: 'dplyr'

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

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

library(tidyr)  
library(plyr)

## -------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.  
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:  
## library(plyr); library(dplyr)

## -------------------------------------------------------------------------

##   
## Attaching package: 'plyr'

## The following objects are masked from 'package:dplyr':  
##   
## arrange, count, desc, failwith, id, mutate, rename, summarise,  
## summarize

library(foreign)  
library(nnet)  
library(reshape2)

##   
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':  
##   
## smiths

library(gridExtra)

##   
## Attaching package: 'gridExtra'

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

require("RWeka")

## Loading required package: RWeka

## Warning: package 'RWeka' was built under R version 3.5.1

##   
## Attaching package: 'RWeka'

## The following objects are masked from 'package:foreign':  
##   
## read.arff, write.arff

library(RWeka)  
require("rpart")

## Loading required package: rpart

## Warning: package 'rpart' was built under R version 3.5.1

require("arules")

## Loading required package: arules

## Warning: package 'arules' was built under R version 3.5.1

## Loading required package: Matrix

##   
## Attaching package: 'Matrix'

## The following object is masked from 'package:tidyr':  
##   
## expand

##   
## Attaching package: 'arules'

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

## The following objects are masked from 'package:base':  
##   
## abbreviate, write

library(arules)

## 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:

## Including Plots

You can also embed plots, for example:

# Reading old and new data for test and train models  
  
old\_data <- read.csv("C:\\Users\\jagan\\Desktop\\CSV datasets for R\\health-insurance-marketplace\\Crosswalk2015.csv", stringsAsFactors=FALSE)  
new\_data <- read.csv("C:\\Users\\jagan\\Desktop\\CSV datasets for R\\health-insurance-marketplace\\Crosswalk2016.csv", stringsAsFactors=FALSE)  
  
# retrieving column names from old\_data  
names(old\_data)

## [1] "State" "DentalPlan"   
## [3] "PlanID\_2014" "IssuerID\_2014"   
## [5] "MultistatePlan\_2014" "MetalLevel\_2014"   
## [7] "ChildAdultOnly\_2014" "FIPSCode"   
## [9] "ZipCode" "CrosswalkLevel"   
## [11] "ReasonForCrosswalk" "PlanID\_2015"   
## [13] "IssuerID\_2015" "MultistatePlan\_2015"   
## [15] "MetalLevel\_2015" "ChildAdultOnly\_2015"   
## [17] "AgeOffPlanID\_2015" "IssuerID\_AgeOff2015"   
## [19] "MultistatePlan\_AgeOff2015" "MetalLevel\_AgeOff2015"   
## [21] "ChildAdultOnly\_AgeOff2015"

head(old\_data)

## State DentalPlan PlanID\_2014 IssuerID\_2014 MultistatePlan\_2014  
## 1 AK Y 21989AK0010001 21989 N  
## 2 AK Y 21989AK0010001 21989 N  
## 3 AK Y 21989AK0010001 21989 N  
## 4 AK Y 21989AK0010001 21989 N  
## 5 AK Y 21989AK0010001 21989 N  
## 6 AK Y 21989AK0010001 21989 N  
## MetalLevel\_2014 ChildAdultOnly\_2014 FIPSCode ZipCode CrosswalkLevel  
## 1 Low 0 2013 0 1  
## 2 Low 0 2016 0 1  
## 3 Low 0 2020 0 1  
## 4 Low 0 2050 0 1  
## 5 Low 0 2060 0 1  
## 6 Low 0 2068 0 1  
## ReasonForCrosswalk PlanID\_2015 IssuerID\_2015 MultistatePlan\_2015  
## 1 6 21989AK0030001 21989 N  
## 2 6 21989AK0030001 21989 N  
## 3 6 21989AK0030001 21989 N  
## 4 6 21989AK0030001 21989 N  
## 5 6 21989AK0030001 21989 N  
## 6 6 21989AK0030001 21989 N  
## MetalLevel\_2015 ChildAdultOnly\_2015 AgeOffPlanID\_2015  
## 1 High 0 00000XX0000000  
## 2 High 0 00000XX0000000  
## 3 High 0 00000XX0000000  
## 4 High 0 00000XX0000000  
## 5 High 0 00000XX0000000  
## 6 High 0 00000XX0000000  
## IssuerID\_AgeOff2015 MultistatePlan\_AgeOff2015 MetalLevel\_AgeOff2015  
## 1 0 X X  
## 2 0 X X  
## 3 0 X X  
## 4 0 X X  
## 5 0 X X  
## 6 0 X X  
## ChildAdultOnly\_AgeOff2015  
## 1 X  
## 2 X  
## 3 X  
## 4 X  
## 5 X  
## 6 X

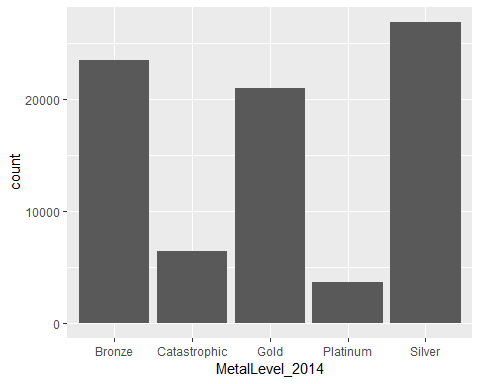
table(old\_data$DentalPlan)

##   
## N Y   
## 81629 50876

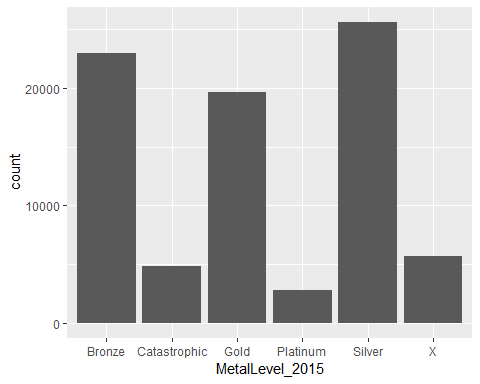
table(new\_data$Dentalplan)

## < table of extent 0 >

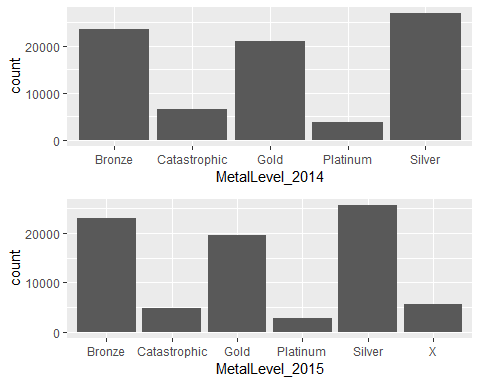
# Getting only the Dental plan part of the Test and Train data  
old\_data <- subset(old\_data,DentalPlan != 'Y')  
new\_data <- subset(new\_data,DentalPlan != 'Y')  
  
# Merging the ID and FIPS code column for key  
old\_data <- old\_data %>% unite("ID",PlanID\_2014,FIPSCode, sep = "-")  
new\_data <- new\_data %>% unite("ID",PlanID\_2015,FIPSCode, sep = "-")  
  
library(gridExtra)  
  
# Visualizing the train data  
p1 <-ggplot(aes(x=MetalLevel\_2014),data = old\_data) + geom\_bar()  
p2 <- ggplot(aes(x=MetalLevel\_2015),data = old\_data) + geom\_bar()  
  
p1



p2



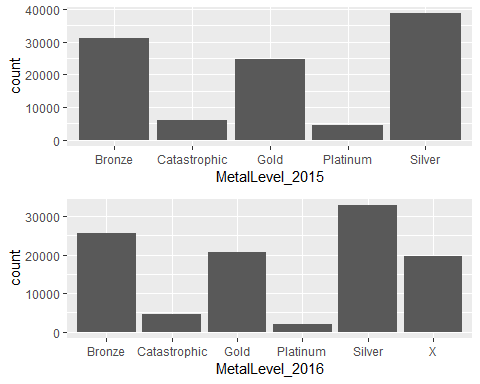
grid.arrange(p1,p2)



ggsave(filename = "MetalLevel2014\_2015.jpg")

## Saving 5 x 4 in image

# Visualizing the test data  
p3 <-ggplot(aes(x=MetalLevel\_2015),data = new\_data) + geom\_bar()  
p4 <-ggplot(aes(x=MetalLevel\_2016),data = new\_data) + geom\_bar()  
  
grid.arrange(p3,p4)



ggsave(filename = "Plan Metals 2015\_2016.jpg")

## Saving 5 x 4 in image

# Creating 0 and 1s based on Metal levels for prediction algorithms  
old\_data$diffmetal\_level <- ifelse(old\_data$MetalLevel\_2014 == old\_data$MetalLevel\_2015,1,0)  
table(old\_data$diffmetal\_level)

##   
## 0 1   
## 7385 74244

names(old\_data)

## [1] "State" "DentalPlan"   
## [3] "ID" "IssuerID\_2014"   
## [5] "MultistatePlan\_2014" "MetalLevel\_2014"   
## [7] "ChildAdultOnly\_2014" "ZipCode"   
## [9] "CrosswalkLevel" "ReasonForCrosswalk"   
## [11] "PlanID\_2015" "IssuerID\_2015"   
## [13] "MultistatePlan\_2015" "MetalLevel\_2015"   
## [15] "ChildAdultOnly\_2015" "AgeOffPlanID\_2015"   
## [17] "IssuerID\_AgeOff2015" "MultistatePlan\_AgeOff2015"  
## [19] "MetalLevel\_AgeOff2015" "ChildAdultOnly\_AgeOff2015"  
## [21] "diffmetal\_level"

new\_data$diffmetal\_level <- ifelse(new\_data$MetalLevel\_2015 == new\_data$MetalLevel\_2016,1,0)  
table(new\_data$diffmetal\_level)

##   
## 0 1   
## 23451 81901

names(new\_data)

## [1] "State" "DentalPlan"   
## [3] "ID" "IssuerID\_2015"   
## [5] "MultistatePlan\_2015" "MetalLevel\_2015"   
## [7] "ChildAdultOnly\_2015" "ZipCode"   
## [9] "CrosswalkLevel" "ReasonForCrosswalk"   
## [11] "PlanID\_2016" "IssuerID\_2016"   
## [13] "MultistatePlan\_2016" "MetalLevel\_2016"   
## [15] "ChildAdultOnly\_2016" "AgeOffPlanID\_2016"   
## [17] "IssuerID\_AgeOff2016" "MultistatePlan\_AgeOff2016"  
## [19] "MetalLevel\_AgeOff2016" "ChildAdultOnly\_AgeOff2016"  
## [21] "diffmetal\_level"

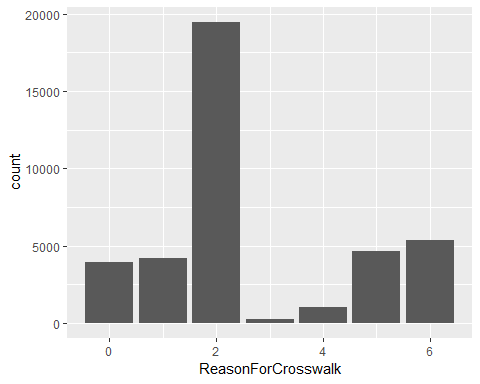
# Creating 0 and 1s based on Crosswalk for prediction algorithms  
old\_data$diff\_crosswalk <- ifelse(old\_data$CrosswalkLevel == old\_data$ReasonForCrosswalk ,1,0)  
table(old\_data$diff\_crosswalk)

##   
## 0 1   
## 34981 46648

new\_data$diff\_crosswalk <- ifelse(new\_data$CrosswalkLevel == new\_data$ReasonForCrosswalk,1,0)  
table(new\_data$diff\_crosswalk)

##   
## 0 1   
## 55248 50104

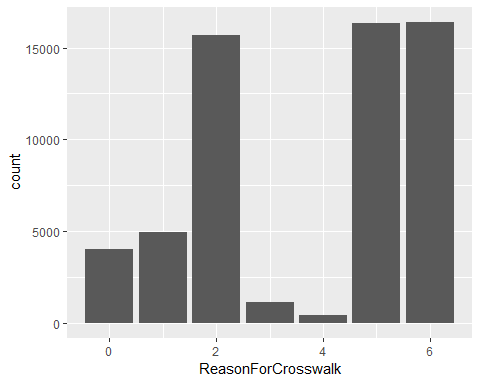
ggplot(aes(x=ReasonForCrosswalk),data = subset(old\_data,CrosswalkLevel != 0)) + geom\_bar()



ggsave(filename = "Crosswalk\_Details.jpg")

## Saving 5 x 4 in image

ggplot(aes(x=ReasonForCrosswalk),data = subset(new\_data,CrosswalkLevel != 0)) + geom\_bar()



ggsave(filename = "Crosswalk\_Details\_2015.jpg")

## Saving 5 x 4 in image

with(old\_data, table(MultistatePlan\_2014, ReasonForCrosswalk))

## ReasonForCrosswalk  
## MultistatePlan\_2014 0 1 2 3 4 5 6  
## N 41881 4194 18408 229 1047 4631 5192  
## Y 4800 6 1065 0 2 0 174

with(old\_data, table(MetalLevel\_2014, ReasonForCrosswalk))

## ReasonForCrosswalk  
## MetalLevel\_2014 0 1 2 3 4 5 6  
## Bronze 12900 1274 6477 6 284 1262 1288  
## Catastrophic 3262 115 2008 97 70 538 374  
## Gold 12447 1114 4275 3 286 1070 1857  
## Platinum 2141 248 596 21 0 649 36  
## Silver 15931 1449 6117 102 409 1112 1811

with(old\_data, table(CrosswalkLevel, ReasonForCrosswalk))

## ReasonForCrosswalk  
## CrosswalkLevel 0 1 2 3 4 5 6  
## 0 42757 0 0 0 0 0 0  
## 1 0 1643 17300 193 0 0 3224  
## 2 2988 2557 2173 36 380 263 2142  
## 3 936 0 0 0 594 0 0  
## 4 0 0 0 0 75 4368 0

old\_data$Factors\_RFC <- as.factor(old\_data$ReasonForCrosswalk)  
  
str(old\_data)

## 'data.frame': 81629 obs. of 23 variables:  
## $ State : chr "AK" "AK" "AK" "AK" ...  
## $ DentalPlan : chr "N" "N" "N" "N" ...  
## $ ID : chr "38344AK0570001-2013" "38344AK0570001-2016" "38344AK0570001-2020" "38344AK0570001-2050" ...  
## $ IssuerID\_2014 : int 38344 38344 38344 38344 38344 38344 38344 38344 38344 38344 ...  
## $ MultistatePlan\_2014 : chr "Y" "Y" "Y" "Y" ...  
## $ MetalLevel\_2014 : chr "Bronze" "Bronze" "Bronze" "Bronze" ...  
## $ ChildAdultOnly\_2014 : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ ZipCode : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ CrosswalkLevel : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ ReasonForCrosswalk : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ PlanID\_2015 : chr "38344AK0570001" "38344AK0570001" "38344AK0570001" "38344AK0570001" ...  
## $ IssuerID\_2015 : int 38344 38344 38344 38344 38344 38344 38344 38344 38344 38344 ...  
## $ MultistatePlan\_2015 : chr "Y" "Y" "Y" "Y" ...  
## $ MetalLevel\_2015 : chr "Bronze" "Bronze" "Bronze" "Bronze" ...  
## $ ChildAdultOnly\_2015 : chr "0" "0" "0" "0" ...  
## $ AgeOffPlanID\_2015 : chr "00000XX0000000" "00000XX0000000" "00000XX0000000" "00000XX0000000" ...  
## $ IssuerID\_AgeOff2015 : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ MultistatePlan\_AgeOff2015: chr "X" "X" "X" "X" ...  
## $ MetalLevel\_AgeOff2015 : chr "X" "X" "X" "X" ...  
## $ ChildAdultOnly\_AgeOff2015: chr "X" "X" "X" "X" ...  
## $ diffmetal\_level : num 1 1 1 1 1 1 1 1 1 1 ...  
## $ diff\_crosswalk : num 1 1 1 1 1 1 1 1 1 1 ...  
## $ Factors\_RFC : Factor w/ 7 levels "0","1","2","3",..: 1 1 1 1 1 1 1 1 1 1 ...

old\_data$RFC2 <- relevel(old\_data$Factors\_RFC, ref = '0')  
  
# same for test data  
  
with(new\_data, table(MultistatePlan\_2015, ReasonForCrosswalk))

## ReasonForCrosswalk  
## MultistatePlan\_2015 0 1 2 3 4 5 6  
## N 44948 4388 15299 1102 387 18551 13672  
## Y 2452 577 402 32 22 789 2731

with(new\_data, table(MetalLevel\_2015, ReasonForCrosswalk))

## ReasonForCrosswalk  
## MetalLevel\_2015 0 1 2 3 4 5 6  
## Bronze 14349 1189 4722 379 127 5424 4959  
## Catastrophic 3357 263 563 25 10 914 949  
## Gold 9745 1410 4505 291 77 4510 4204  
## Platinum 1190 445 1074 58 1 1441 421  
## Silver 18759 1658 4837 381 194 7051 5870

with(new\_data, table(CrosswalkLevel, ReasonForCrosswalk))

## ReasonForCrosswalk  
## CrosswalkLevel 0 1 2 3 4 5 6  
## 0 43355 0 0 0 0 3016 0  
## 1 0 4062 12961 183 0 1214 10670  
## 2 3465 903 1621 543 256 1046 5733  
## 3 580 0 1119 408 138 0 0  
## 4 0 0 0 0 15 13421 0  
## 5 0 0 0 0 0 643 0

new\_data$Factors\_RFC <- as.factor(new\_data$ReasonForCrosswalk)  
  
str(new\_data)

## 'data.frame': 105352 obs. of 23 variables:  
## $ State : chr "OR" "OR" "OR" "OR" ...  
## $ DentalPlan : chr "N" "N" "N" "N" ...  
## $ ID : chr "10091OR0360004-41021" "10091OR0360004-41037" "10091OR0360004-41009" "10091OR0360004-41023" ...  
## $ IssuerID\_2015 : int 10091 10091 10091 10091 10091 10091 10091 10091 10091 10091 ...  
## $ MultistatePlan\_2015 : chr "N" "N" "N" "N" ...  
## $ MetalLevel\_2015 : chr "Bronze" "Bronze" "Bronze" "Bronze" ...  
## $ ChildAdultOnly\_2015 : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ ZipCode : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ CrosswalkLevel : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ ReasonForCrosswalk : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ PlanID\_2016 : chr "10091OR0360004" "10091OR0360004" "10091OR0360004" "10091OR0360004" ...  
## $ IssuerID\_2016 : int 10091 10091 10091 10091 10091 10091 10091 10091 10091 10091 ...  
## $ MultistatePlan\_2016 : chr "N" "N" "N" "N" ...  
## $ MetalLevel\_2016 : chr "Bronze" "Bronze" "Bronze" "Bronze" ...  
## $ ChildAdultOnly\_2016 : chr "0" "0" "0" "0" ...  
## $ AgeOffPlanID\_2016 : chr "00000XX0000000" "00000XX0000000" "00000XX0000000" "00000XX0000000" ...  
## $ IssuerID\_AgeOff2016 : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ MultistatePlan\_AgeOff2016: chr "X" "X" "X" "X" ...  
## $ MetalLevel\_AgeOff2016 : chr "X" "X" "X" "X" ...  
## $ ChildAdultOnly\_AgeOff2016: chr "X" "X" "X" "X" ...  
## $ diffmetal\_level : num 1 1 1 1 1 1 1 1 1 1 ...  
## $ diff\_crosswalk : num 1 1 1 1 1 1 1 1 1 1 ...  
## $ Factors\_RFC : Factor w/ 7 levels "0","1","2","3",..: 1 1 1 1 1 1 1 1 1 1 ...

new\_data$RFC2 <- relevel(new\_data$Factors\_RFC, ref = '0')

# Variables for kNN

pred\_label <- old\_data$MetalLevel\_2014

knn\_old\_data1 <- as.numeric(old\_data$MultistatePlan\_2014)

knn\_old\_data2 <- as.numeric(old\_data$MetalLevel\_2014)

ktrain <- cbind(knn\_old\_data1,knn\_old\_data2,old\_data[,9])

knn\_new\_data1 <- as.numeric(new\_data$MultistatePlan\_2014)

knn\_new\_data2 <- as.numeric(new\_data$MetalLevel\_2014)

## as base reference for the multinomial logistic model  
  
test <- multinom(RFC2 ~ MultistatePlan\_2014 + MetalLevel\_2014 + CrosswalkLevel, data = old\_data)

## # weights: 56 (42 variable)  
## initial value 158842.699557   
## iter 10 value 70763.377845  
## iter 20 value 69435.521212  
## iter 30 value 64994.627092  
## iter 40 value 62912.850682  
## iter 50 value 60030.246549  
## iter 60 value 59918.697689  
## iter 70 value 59911.911586  
## iter 80 value 59910.710769  
## iter 90 value 59910.489109  
## iter 100 value 59910.420713  
## final value 59910.420713   
## stopped after 100 iterations

# SVM  
  
library(e1071)

## Warning: package 'e1071' was built under R version 3.5.1

svm\_test <- svm(RFC2 ~ MultistatePlan\_2014 + MetalLevel\_2014 + CrosswalkLevel, data = old\_data)  
svm\_test

##   
## Call:  
## svm(formula = RFC2 ~ MultistatePlan\_2014 + MetalLevel\_2014 +   
## CrosswalkLevel, data = old\_data)  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: radial   
## cost: 1   
## gamma: 0.1428571   
##   
## Number of Support Vectors: 20024

# Renaming the column names  
  
new\_cols <- colnames(new\_data)  
  
new\_cols <- c("State","DentalPlan","ID","IssuerID\_2015","MultistatePlan\_2014","MetalLevel\_2014","ChildAdultOnly\_2015","ZipCode","CrosswalkLevel","ReasonForCrosswalk","PlanID\_2016","IssuerID\_2016","MultistatePlan\_2016","MetalLevel\_2016","ChildAdultOnly\_2016","AgeOffPlanID\_2016","IssuerID\_AgeOff2016","MultistatePlan\_AgeOff2016","MetalLevel\_AgeOff2016","ChildAdultOnly\_AgeOff2016","diffmetal\_level","diff\_crosswalk","Factors\_RFC","RFC2")  
colnames(new\_data) <- new\_cols  
  
pred=predict(svm\_test, newdata=new\_data, type=c("class"))  
pred

## 1 2 3 4 5 6 7 8 9 10   
## 0 0 0 0 0 0 0 0 0 0   
## 11 12 13 14 15 16 17 18 19 20   
## 0 0 0 0 0 0 0 0 0 0   
## 21 22 23 24 25 26 27 28 29 30   
## 0 0 0 0 0 0 0 0 0 0   
## 31 32 33 34 35 36 37 38 39 40   
## 0 0 0 0 0 0 2 2 2 2   
## 41 42 43 44 45 46 47 48 49 50   
## 2 2 2 2 2 2 2 2 0 0   
## 51 52 53 54 55 56 57 58 59 60   
## 0 0 0 0 0 0 0 0 0 0   
## 61 62 63 64 65 66 67 68 69 70   
## 0 0 0 0 0 0 2 2 2 2   
## 71 72 73 74 75 76 77 78 79 80   
## 2 2 2 2 2 2 2 2 2 2   
## 81 82 83 84 85 86 87 88 89 90   
## 2 2 2 2 2 2 2 2 2 2   
## 91 92 93 94 95 96 97 98 99 100   
## 2 2 2 2 2 2 2 2 2 2   
## 101 102 103 104 105 106 107 108 109 110   
## 2 2 2 2 2 2 2 2 2 2   
## 111 112 113 114 115 116 117 118 119 120   
## 2 2 0 0 0 0 0 0 0 0   
## 121 122 123 124 125 126 127 128 129 130   
## 0 0 0 0 0 0 0 0 0 0   
## 131 132 133 134 135 136 137 138 139 140   
## 2 2 2 2 2 2 2 2 2 2   
## 141 142 143 144 145 146 147 148 149 150   
## 2 2 2 2 0 0 0 0 0 0   
## 151 152 153 154 155 156 157 158 159 160   
## 0 0 0 0 0 0 0 0 0 0   
## 161 162 163 164 165 166 167 168 169 170   
## 0 0 2 2 2 2 2 2 2 2   
## 171 172 173 174 175 176 177 178 179 180   
## 2 2 2 2 2 2 2 2 2 2   
## 181 182 183 184 185 186 187 188 189 190   
## 2 2 2 2 2 2 2 2 2 2   
## 191 192 193 194 195 196 197 198 199 200   
## 2 2 2 2 2 2 2 2 0 0   
## 201 202 203 204 205 206 207 208 209 210   
## 0 0 0 0 0 0 2 2 2 2   
## 211 212 213 214 215 216 217 218 219 220   
## 2 2 2 2 2 2 2 2 2 2   
## 221 222 223 224 225 226 227 228 229 230   
## 2 2 2 2 2 2 2 2 2 2   
## 231 232 233 234 235 236 237 238 239 240   
## 2 2 0 0 0 0 0 0 0 0   
## 241 242 243 244 245 246 247 248 249 250   
## 0 0 0 0 0 0 0 0 0 0   
## 251 252 253 254 255 256 257 258 259 260   
## 0 0 0 0 0 0 0 0 0 0   
## 261 262 263 264 265 266 267 268 269 270   
## 2 2 2 2 2 2 2 2 2 2   
## 271 272 273 274 275 276 277 278 279 280   
## 2 2 2 2 2 2 2 2 2 2   
## 281 282 283 284 285 286 287 288 289 290   
## 2 2 2 2 2 2 2 2 0 0   
## 291 292 293 294 295 296 297 298 299 300   
## 0 0 0 0 0 0 0 0 0 0   
## 301 302 303 304 305 306 307 308 309 310   
## 0 0 0 0 0 0 2 2 2 2   
## 311 312 313 314 315 316 317 318 319 320   
## 2 2 2 2 2 2 2 2 2 2   
## 321 322 323 324 325 326 327 328 329 330   
## 2 2 2 2 0 0 0 0 2 2   
## 331 332 333 334 335 336 337 338 339 340   
## 2 2 2 2 2 2 2 2 2 2   
## 341 342 343 344 345 346 347 348 349 350   
## 2 2 2 2 2 2 2 2 2 2   
## 351 352 353 354 355 356 357 358 359 360   
## 2 2 2 2 0 0 0 0 0 0   
## 361 362 363 364 365 366 367 368 369 370   
## 0 0 0 0 0 0 0 0 2 2   
## 371 372 373 374 375 376 377 378 379 380   
## 2 2 2 2 2 2 2 2 2 2   
## 381 382 383 384 385 386 387 388 389 390   
## 2 2 2 2 2 2 2 2 2 2   
## 391 392 393 394 395 396 531 532 533 534   
## 2 2 2 2 2 2 0 0 0 0   
## 535 536 537 538 539 540 541 542 543 544   
## 0 0 0 0 0 0 0 0 0 0   
## 545 546 547 548 549 550 551 552 553 554   
## 0 0 0 0 0 0 0 0 0 0   
## 555 556 557 558 559 560 561 562 563 564   
## 0 0 0 0 0 0 0 0 0 0   
## 565 566 567 568 569 570 571 572 573 574   
## 0 0 0 0 0 0 0 0 0 0   
## 575 576 577 578 579 580 581 582 583 584   
## 0 0 0 0 0 0 0 0 0 0   
## 585 586 587 588 589 590 591 592 593 594   
## 0 0 0 0 0 0 0 0 0 0   
## 595 596 597 598 599 600 601 602 603 604   
## 0 0 0 0 0 0 0 0 0 0   
## 605 606 607 608 609 610 611 612 613 614   
## 0 0 0 0 0 0 0 0 0 0   
## 615 616 617 618 619 620 621 622 623 624   
## 0 0 0 0 0 0 0 0 0 0   
## 625 626 627 628 629 630 631 632 633 634   
## 0 0 0 0 0 0 0 0 0 0   
## 635 636 637 638 639 640 641 642 643 644   
## 0 0 0 0 0 0 0 0 0 0   
## 645 646 647 648 649 650 651 652 653 654   
## 0 0 0 0 0 0 0 0 0 0   
## 655 656 657 658 659 660 661 662 663 664   
## 0 0 0 0 0 0 0 0 0 0   
## 665 666 667 668 669 670 671 672 673 674   
## 0 0 0 0 0 0 0 0 0 0   
## 675 676 677 678 679 680 681 682 683 684   
## 0 0 0 0 0 0 0 0 0 0   
## 685 686 687 688 689 690 691 692 693 694   
## 0 0 0 0 0 0 0 0 0 0   
## 695 696 697 698 699 700 701 702 703 704   
## 0 0 0 0 0 0 0 0 0 0   
## 705 706 707 708 709 710 711 712 713 714   
## 0 0 0 0 0 0 0 0 0 0   
## 715 716 717 718 719 720 721 722 723 724   
## 0 0 0 0 0 0 2 2 2 0   
## 725 726 727 728 729 730 731 732 733 734   
## 0 0 0 0 0 0 0 0 0 0   
## 735 736 737 738 739 740 741 742 743 744   
## 0 0 0 0 0 0 0 0 0 0   
## 745 746 747 748 749 750 751 752 753 754   
## 0 0 0 0 0 0 0 0 0 0   
## 755 756 757 758 759 760 761 762 763 764   
## 0 0 0 0 0 0 0 2 2 2   
## 765 766 767 768 769 770 771 772 773 774   
## 2 2 2 2 2 2 2 2 2 2   
## 775 776 777 778 779 780 781 782 783 784   
## 2 2 2 2 2 2 0 0 0 0   
## 785 786 787 788 789 790 791 792 793 794   
## 0 0 0 0 0 0 0 0 0 0   
## 795 796 797 798 799 800 801 802 803 804   
## 0 0 0 0 0 0 0 0 0 0   
## 805 806 807 808 809 810 811 812 813 814   
## 0 0 0 0 0 0 0 0 0 0   
## 815 816 817 818 819 820 821 822 823 824   
## 0 0 0 0 0 0 0 0 0 0   
## 825 826 827 828 829 830 831 832 833 834   
## 0 0 0 0 0 0 0 0 0 0   
## 835 836 837 838 839 840 841 842 843 844   
## 0 0 0 0 0 0 0 0 0 0   
## 845 846 847 848 849 850 851 852 853 854   
## 0 0 0 0 0 0 0 0 0 0   
## 855 856 857 858 859 860 861 862 863 864   
## 0 0 0 0 0 0 0 0 0 0   
## 865 866 867 868 869 870 871 872 873 874   
## 0 0 0 0 0 0 0 0 0 0   
## 875 876 877 878 879 880 881 882 883 884   
## 0 0 0 0 0 0 0 0 0 0   
## 885 886 887 888 889 890 891 892 893 894   
## 0 0 0 0 0 0 0 0 0 0   
## 895 896 897 898 899 900 901 902 903 904   
## 0 0 0 0 0 0 0 0 0 0   
## 905 906 907 908 909 910 911 912 913 914   
## 0 0 0 0 0 0 0 2 2 2   
## 915 916 917 918 919 920 921 922 923 924   
## 2 2 2 2 2 2 2 2 2 2   
## 925 926 927 928 929 930 931 932 933 934   
## 2 2 2 2 2 2 2 2 0 0   
## 935 936 937 938 939 940 941 942 943 944   
## 0 0 0 0 0 0 0 0 0 0   
## 945 946 947 948 949 950 951 952 953 954   
## 0 0 2 2 2 2 2 2 2 2   
## 955 956 957 958 959 960 1198 1199 1200 1201   
## 2 2 2 2 2 2 0 0 0 0

# Running Naive Bayes Algorithm   
  
nb=naiveBayes(RFC2 ~ MultistatePlan\_2014 + MetalLevel\_2014 + CrosswalkLevel, data = old\_data,laplace = 1, na.action = na.pass)  
nb

##   
## Naive Bayes Classifier for Discrete Predictors  
##   
## Call:  
## naiveBayes.default(x = X, y = Y, laplace = laplace)  
##   
## A-priori probabilities:  
## Y  
## 0 1 2 3 4 5   
## 0.571867841 0.051452302 0.238554925 0.002805376 0.012850825 0.056732289   
## 6   
## 0.065736442   
##   
## Conditional probabilities:  
## MultistatePlan\_2014  
## Y N Y  
## 0 0.8971958613 0.1028469827  
## 1 0.9988095238 0.0016666667  
## 2 0.9453602424 0.0547424639  
## 3 1.0043668122 0.0043668122  
## 4 0.9990467112 0.0028598665  
## 5 1.0002159361 0.0002159361  
## 6 0.9677599702 0.0326127469  
##   
## MetalLevel\_2014  
## Y Bronze Catastrophic Gold Platinum Silver  
## 0 0.2763651164 0.0698999593 0.2666609541 0.0458859065 0.3412951736  
## 1 0.3035714286 0.0276190476 0.2654761905 0.0592857143 0.3452380952  
## 2 0.3326657423 0.1031684897 0.2195860936 0.0306578339 0.3141786063  
## 3 0.0305676856 0.4279475983 0.0174672489 0.0960698690 0.4497816594  
## 4 0.2716873213 0.0676835081 0.2735938990 0.0009532888 0.3908484271  
## 5 0.2727272727 0.1163895487 0.2312675448 0.1403584539 0.2403368603  
## 6 0.2402161759 0.0698844577 0.3462541931 0.0068952665 0.3376816996  
##   
## CrosswalkLevel  
## Y [,1] [,2]  
## 0 0.1881708 0.6333205  
## 1 1.6088095 0.4880750  
## 2 1.1115904 0.3148699  
## 3 1.1572052 0.3647915  
## 4 2.7092469 0.5912210  
## 5 3.8864176 0.4629364  
## 6 1.3991800 0.4897755

pred2 = predict(nb, newdata=new\_data, type=c("class"))

## Warning in data.matrix(newdata): NAs introduced by coercion

## Warning in data.matrix(newdata): NAs introduced by coercion  
  
## Warning in data.matrix(newdata): NAs introduced by coercion  
  
## Warning in data.matrix(newdata): NAs introduced by coercion  
  
## Warning in data.matrix(newdata): NAs introduced by coercion  
  
## Warning in data.matrix(newdata): NAs introduced by coercion  
  
## Warning in data.matrix(newdata): NAs introduced by coercion  
  
## Warning in data.matrix(newdata): NAs introduced by coercion

## Warning in data.matrix(newdata): NAs introduced by coercion  
  
## Warning in data.matrix(newdata): NAs introduced by coercion  
  
## Warning in data.matrix(newdata): NAs introduced by coercion

pred2

## [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [35] 0 0 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2  
## [69] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [103] 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2  
## [137] 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2  
## [171] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1  
## [205] 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1  
## [239] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2  
## [273] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [307] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2  
## [341] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2  
## [375] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0  
## [409] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [443] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [477] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [511] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [545] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [579] 0 0 0 0 0 0 0 0 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [613] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [647] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [681] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [715] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [749] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2  
## [783] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2  
## [817] 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [851] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [885] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [919] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [953] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [987] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

# Running KNN Algorithm   
  
knn\_train <- knn(train=ktrain, test=ktest, cl=pred\_labels, k=3)

knn\_train

library(class)

predKNN <- knn(train=ktrain, test=ktest, cl=pred\_label, k=3)

predKNN

## 1 2 3 4 5 6 7 8 9 10   
## 0 0 0 0 0 0 0 0 0 0   
## 11 12 13 14 15 16 17 18 19 20   
## 0 0 0 0 0 0 0 0 0 0   
## 21 22 23 24 25 26 27 28 29 30   
## 0 0 0 0 0 0 0 0 0 0   
## 31 32 33 34 35 36 37 38 39 40   
## 0 0 0 0 0 0 2 2 2 2   
## 41 42 43 44 45 46 47 48 49 50   
## 2 2 2 2 2 2 2 2 0 0   
## 51 52 53 54 55 56 57 58 59 60   
## 0 0 0 0 0 0 0 0 0 0   
## 61 62 63 64 65 66 67 68 69 70   
## 0 0 0 0 0 0 2 2 2 2   
## 71 72 73 74 75 76 77 78 79 80   
## 2 2 2 2 2 2 2 2 2 2   
## 81 82 83 84 85 86 87 88 89 90   
## 2 2 2 2 2 2 2 2 2 2   
## 91 92 93 94 95 96 97 98 99 100   
## 2 2 2 2 2 2 2 2 2 2   
## 101 102 103 104 105 106 107 108 109 110