# Dharmam Buch

## Capital One Code Challenge 1

1. Analyze the data set.

It is important to first analyze the data set before heading toward the analytics of the same. Given training data has 255 features and 5000 instances.

* 1. The given file is .txt format and to use R –Studio we will need .csv file, so first we convert the space-separated file into comma-separated csv file using linux command line.
  2. **Target is continuous** and doesn’t belong to a fixed class range.
  3. Some of the features, like f\_61,f\_121, **are categorical data.**

1. Once the overall data overview is done, we move toward modeling.
   1. As the target class are continuous, we cannot use traditional classifiers, hence, a **regression-prediction approach** is to be taken.
   2. Tool used **R-Studio** , Package Used **e1071 and caret package.**
2. Begin the analytics of the data.
   1. First we start with finding the features which are least important for our model. The approach here is to find the **variance of individual data.**

**Why Variance ?** Variance is in lay-men terms is “how far the data is distributed”, hence, a low variance brings the data points very near and hence, are difficult to do regression on.

* 1. Using the following code :

dataDirectory <- "C:/Users/Dharmam/Desktop/capital one/check.csv"

data <- read.csv(dataDirectory, sep="," , header = TRUE)

nzv <- nearZeroVar(data,saveMetrics = TRUE)

print(paste('Range :', range(nzv$percentUnique)))

head(nzv)

head(nzv[nzv$percentUnique < 0.5,])

Result :

freqRatio percentUnique zeroVar nzv

f\_61 1.033605 0.12 FALSE FALSE

f\_121 1.008434 0.14 FALSE FALSE

f\_215 1.057190 0.10 FALSE FALSE

f\_237 1.016364 0.08 FALSE FALSE

* 1. Now we can remove these four columns from our data in the code while making the model.

1. Now the data contains a lot of “” values in different features. So we can start with preliminary research and to check how different feature range impacts the number of predictions on the test data.
   1. Use code –

# Load the data from the csv file

dataDirectory <- "C:/Users/Dharmam/Desktop/capital one/check.csv"

data <- read.csv(dataDirectory, sep="," , header = TRUE,na.strings=c("") )

x<-data[,230:240]

#Filter Data with less variance.

drops <- c("f\_61","f\_121","f\_215","f\_237")

filteredData <-x[,!(names(x) %in% drops)]

#take target.

target<- data$target

#Create a linear regression model

model <- lm(target ~ .,filteredData)

#Read the test data.

testdata <- read.csv("C:/Users/Dharmam/Desktop/capital one/checktest.csv", sep="," ,

header = TRUE)

drops <- c("f\_61","f\_121","f\_215","f\_237")

TestData<-testdata[,!(names(testdata) %in% drops)]

#predicts the traget

result <- predict(model,TestData)

plot(result)

print(summary(result))

* 1. After changing the range from 0 – 255 we get the following result :

Capital One Regression :-

1. x<-data[,0:10]

summary(result)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

0.6642 1.0210 1.1260 1.1230 1.2230 1.5930 165

2. x<-data[,10:20]

summary(result)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

0.2382 0.9446 1.1040 1.1130 1.2840 2.1070 217

3. x<-data[,20:30]

summary(result)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

-1.9120 0.5609 1.1970 1.2180 1.8660 4.0230 183

4. x<-data[,40:50]

summary(result)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

-0.8716 0.6750 1.2330 1.2290 1.7590 4.2300 208

5. x<-data[,50:60]

summary(result)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

0.2290 0.9383 1.1430 1.1300 1.3140 1.8700 178

6. x<-data[,60:70]

drops <- c("f\_61")

X<-x[,!(names(x) %in% drops)]

Dropping f\_61 as for non numrical data.

summary(result)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

0.02044 0.90620 1.11500 1.11500 1.33800 2.31800 145

7. x<-data[,70:80]

summary(result)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

-0.6454 0.7349 1.1360 1.1300 1.5460 2.9460 200

8. x<-data[,80:90]

summary(result)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

0.1143 0.8907 1.0960 1.1140 1.3270 2.1080 178

9. x<-data[,90:100]

summary(result)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

-0.3562 0.8176 1.1570 1.1530 1.4580 2.6470 209

10. x<-data[,100:110]

summary(result)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

0.567 1.003 1.144 1.143 1.287 1.811 207

11. x<-data[,110:120]

summary(result)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

0.5604 1.0080 1.1370 1.1320 1.2620 1.7030 209

12. x<-data[,120:130]

x<-data[,120:130]

drops <- c("f\_61","f\_121")

X<-x[,!(names(x) %in% drops)]

#Dropping f\_121 for strings values gives error.

summary(result)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

0.05307 0.95220 1.15800 1.16100 1.36000 2.14200 161

13. x<-data[,130:140]

summary(result)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

0.6352 1.0680 1.1730 1.1830 1.3070 1.7210 204

14. x<-data[,140:150]

summary(result)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

-0.1769 0.8846 1.1100 1.1160 1.3430 2.2060 215

15. x<-data[,150:160]

summary(result)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

0.2179 0.8588 1.0630 1.0740 1.2950 2.0080 192

16. x<-data[,160:170]

summary(result)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

-5.5950 -0.4192 0.9647 0.9922 2.4440 7.2930 204

17. x<-data[,170:180]

summary(result)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

-7.9410 -0.9326 0.9942 1.0360 2.9680 11.2300 206

18. x<-data[,180:190]

summary(result)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

0.4688 1.0280 1.1650 1.1660 1.2990 1.8280 201

19. x<-data[,190:200]

summary(result)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

-3.1090 0.3026 1.2350 1.2420 2.1840 5.3930 198

20. x,-data[,200:210]

summary(result)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

-6.4770 -0.2007 1.0610 1.1200 2.5070 6.3170 179

21. x<-data[,210:220]

drops <- c("f\_61","f\_121","f\_215")

X<-x[,!(names(x) %in% drops)]

summary(result)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

-1.8100 0.3921 1.0160 1.0380 1.6650 3.9300 177

22. x<-data[,220:230]

summary(result)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

-1.9630 0.5801 1.2150 1.1690 1.7520 3.5910 191

23. target<- data$target

x<-data[,230:240]

drops <- c("f\_61","f\_121","f\_215","f\_237")

X<-x[,!(names(x) %in% drops)]

summary(result)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

0.1969 0.9143 1.0790 1.0800 1.2350 1.8960 178

24. x<-data[,240:250]

summary(result)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

0.5922 1.0140 1.1460 1.1430 1.2750 1.8470 195

25.x<-data[,250:255]

summary(result)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

0.6618 1.0180 1.1230 1.1230 1.2300 1.5280 108

1. Solution :-

Two approach :-

* 1. Dimensional Deduction + Principal component analysis.
  2. Normal Average of simplistic regression model for major features.

1. Dimensionality Reduction –
   1. dataDirectory <- "C:/Users/Dharmam/Desktop/capital one/check.csv"

gisetteRaw <- read.table(dataDirectory, sep="," , header = TRUE)

print(dim(gisetteRaw))

nzv <- nearZeroVar(gisetteRaw, saveMetrics = TRUE)

print(paste('Range:',range(nzv$percentUnique)))

print(paste('Column count before cutoff:',nrow(nzv)))

## [1] "Column count before cutoff: 255"

data <- gisetteRaw[c(rownames(nzv[nzv$percentUnique > 0.5,])) ]

gisette\_nzv <- data[apply(data, 1, Compose(is.finite, all)),]

print(dim(gisette\_nzv))

print(paste('Column count after cutoff:',ncol(gisette\_nzv)))

## [1] "Column count before cutoff: 251"

* 1. Removes columns with less variance.

1. Principal Component Analysis –
   1. Fine the AUC(area under curve) for the filtered data “gisett\_nzv” and then do the same with number of columns limited to N.

dfEvaluate <- cbind(as.data.frame(sapply(gisette\_nzv, as.numeric)),

cluster=gisette\_nzv$target)

EvaluateAUC(dfEvaluate)

pmatrix <- scale(gisette\_nzv)

princ <- prcomp(pmatrix)

* 1. EvaluateAUC() is from an online resource which uses XGBoost package to calculate AUC.

Code for the same is :

EvaluateAUC <- function(dfEvaluate) {

require(xgboost)

require(Metrics)

require(pROC)

CVs <- 5

cvDivider <- floor(nrow(dfEvaluate) / (CVs+1))

indexCount <- 1

outcomeName <- c('cluster')

predictors <- names(dfEvaluate)[!names(dfEvaluate) %in% outcomeName]

lsErr <- c()

lsAUC <- c()

for (cv in seq(1:CVs)) {

# print(paste('cv',cv))

dataTestIndex <- c((cv \* cvDivider):(cv \* cvDivider + cvDivider))

dataTest <- dfEvaluate[dataTestIndex,]

dataTrain <- dfEvaluate[-dataTestIndex,]

bst <- xgboost(data = as.matrix(dataTrain[,predictors]),

label = dataTrain[,outcomeName],

max.depth=6, eta = 1, verbose=0,

nround=5, nthread=4,

objective = "reg:linear")

predictions <- predict(bst, as.matrix(dataTest[,predictors]), outputmargin=TRUE)

err <- rmse(dataTest[,outcomeName], predictions)

auc <- auc(dataTest[,outcomeName],predictions)

lsErr <- c(lsErr, err)

lsAUC <- c(lsAUC, auc)

gc()

}

print(paste('Mean Error:',mean(lsErr)))

print(paste('Mean AUC:',mean(lsAUC)))

}

1. Once we are done we are done for the whole data we start doing the same with a selection of features.

nComp <- 1

dfComponents <- predict(princ, newdata=pmatrix)[,1:nComp]

dfEvaluate <- cbind(as.data.frame(dfComponents),

cluster=g\_labels$V1)

EvaluateAUC(dfEvaluate)

1. Continue increasing n till you reach new the original AUC.

However, in this case we were not able to converge on a specific value of K, this can also be seen by looking at the variance of individual features, which are very close to each other around ~50%.

1. Finally, we make model –

library(‘lattice’)

library(‘Metrices’)

library(‘e1071’)

library(‘caret’)

# Load the data from the csv file

dataDirectory <- "C:/Users/Dharmam/Desktop/capital one/check.csv"

gisetteRaw <- read.csv(dataDirectory, sep="," , header = TRUE,na.strings=c("") )

print(dim(gisetteRaw))

nzv <- nearZeroVar(gisetteRaw, saveMetrics = TRUE)

print(paste('Range:',range(nzv$percentUnique)))

print(paste('Column count before cutoff:',nrow(nzv)))

## [1] "Column count before cutoff: 255"

gisette\_nzv <- gisetteRaw[c(rownames(nzv[nzv$percentUnique > 0.5,])) ]

print(dim(gisette\_nzv))

print(paste('Column count after cutoff:',ncol(gisette\_nzv)))

## [1] "Column count before cutoff: 251"

#Create a linear regression model

target<- gisette\_nzv$target

model <- lm(target ~ .,gisette\_nzv)

head(model)

testdata <- read.csv("C:/Users/Dharmam/Desktop/capital one/checktest.csv", sep="," ,

header = TRUE)

na.omit(testdata)

drops <- c("f\_61","f\_121","f\_215","f\_237")

TestData<-testdata[,!(names(testdata) %in% drops)]

result <- predict(model,TestData)

plot(result)

print(summary(result))

1. Note that there are still 400 NA, hence, we can call individual feature range and take the average.
2. Solution :-

Take the features with maximum variance. nzv[nzv$percentUnique > 56,] and then take more models and consolidate them.

Please, see Solution excel for the results.