Machine Learning Final Project

Import Data Set Into R

```
library(plyr)
library(mosaic)
```

```
## Loading required package: dplyr
```

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

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

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

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

```
## Loading required package: lattice
```

```
## Loading required package: ggformula
```

```
## Loading required package: ggplot2
```

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

```
## Loading required package: mosaicData
```

```
## Loading required package: Matrix
```

```
##
## 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.
##
## Note: If you use the Matrix package, be sure to load it BEFORE loading mosaic.
```

```
##
## Attaching package: 'mosaic'
```

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

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

```
## The following object is masked from 'package:plyr':
##
```

```
##     count
## 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(readr)
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.4
##
## Attaching package: 'caret'
## The following object is masked from 'package:mosaic':
##
##     dotPlot
```

```
library(party)
```

```
## Warning: package 'party' was built under R version 3.4.4
## Loading required package: grid
## Loading required package: mvtnorm
## Loading required package: modeltools
## Loading required package: stats4
##
## Attaching package: 'modeltools'
## The following object is masked from 'package:plyr':
##
##     empty
## Loading required package: strucchange
## Loading required package: zoo
##
## Attaching package: 'zoo'
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
## Loading required package: sandwich
```

```
diabetic_data <- read_csv("diabetic_data.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   encounter_id = col_integer(),
##   patient_nbr = col_integer(),
##   admission_type_id = col_integer(),
##   discharge_disposition_id = col_integer(),
```

```
##   admission_source_id = col_integer(),
##   time_in_hospital = col_integer(),
##   num_lab_procedures = col_integer(),
##   num_procedures = col_integer(),
##   num_medications = col_integer(),
##   number_outpatient = col_integer(),
##   number_emergency = col_integer(),
##   number_inpatient = col_integer(),
##   number_diagnoses = col_integer()
## )
```

```
## See spec(...) for full column specifications.
```

```
no_dup <- data.frame(diabetic_data)
#no_dup[1:10,3:6]
```

Data Clean Up

We dont have admission date data, and some patients have multiple admissions in this dataset. We removed duplicate patient encounters, keeping only the first. This ensures that each patient encounter is independent. We are then left with 70,000 encounters instead of 100,000.

```
no_dup <- no_dup[order(no_dup$encounter_id),]
no_dup <- no_dup[!duplicated(no_dup$patient_nbr), ]
no_dup<-no_dup[!(no_dup$gender=="Unknown/Invalid"),]
#write.csv(no_dup, "no_duplicate.csv")
```

Missing Values

```
no_dup$weight <- factor(no_dup$weight)
table(no_dup$weight)
```

```
##
##         ?    [0-25) [100-125) [125-150) [150-175) [175-200)   [25-50)
##     68662        46       566       131        33         9        89
##   [50-75)  [75-100)      >200
##       781      1195         3
```

According to the table, 96% of the weight data is missing, so we will remove that attribute from our model.

```
no_dup$weight <- NULL
#no_dup[1:10,3:6]
```

payer_code and medical_specialty are missing over 50% of their data, so we remove those from our model as well. Encounter Id and patient number are meaningless for purposes of this project, so we can get rid of those as well. examide, glimepriride-pioglitazone and citoglipton have the same output "No", for every instance, so there's no purpose in keeping those either. After removing problem categories, remove all other rows with null values.

```
no_dup$payer_code <- NULL
no_dup$medical_specialty <- NULL
no_dup$examide <-NULL
no_dup$citoglipton<- NULL
no_dup$encounter_id <- NULL
no_dup$patient_nbr<- NULL
no_dup$`glimepiride-pioglitazone` <- NULL
```

```
no_dup$admission_type_id <- NULL
no_dup$discharge_disposition_id <- NULL
no_dup$admission_source_id<- NULL
no_dup$num_lab_procedures <- NULL
no_dup$num_medications <- NULL
no_dup$number_outpatient <- NULL
no_dup$number_emergency <- NULL
no_dup$chlorpropamide <- NULL
no_dup$acetohexamide <- NULL
no_dup$tolbutamide <- NULL
no_dup$miglitol <- NULL
no_dup$troglitazone <- NULL
no_dup$glyburide.metformin <-NULL
no_dup$glipizide.metformin<-NULL
no_dup$glimepiride.pioglitazone <- NULL
no_dup$metformin.rosiglitazone<-NULL
no_dup$metformin.pioglitazone <-NULL
no_dup$number_inpatient <- NULL
no_dup$number_diagnoses <- NULL
no_dup$nateglinide <-NULL
```

```
no_dup <- no_dup[complete.cases(no_dup), ]
```

Factors

We want to relabel character values as factor attributes, and also relabel readmittance to be a 2 level factor variable, with No corresponding to no readmittance in 30 days, and Yes corresponding to a readmittance within 30 days.

This next code loops through the variables that need to become factors instead of integers or chars.

```
d_header <- colnames(no_dup)
d_header <- d_header[c(1:6, 10:22)]
no_dup$diag_1 <- as.numeric(no_dup$diag_1)
```

```
## Warning: NAs introduced by coercion
```

Need to collapse diagnoses into smaller factor levels.

```
tables <- lapply(no_dup[d_header], table)
```

```
v1 <- as.numeric(no_dup$diag_1)
v2 <- vector()
  for ( x in 1:length(v1)){
    if(!is.na(v1[x])) {
      if((v1[x] >= 390 & v1[x] <= 459) | v1[x] == 785){
        v2[x] <- "Circulatory"
      }
      else if((v1[x] >= 450 & v1[x] <= 519) | v1[x] == 786){
        v2[x] <- "Respiratory"
      }
      else if((v1[x] >= 520 & v1[x] <= 579) | v1[x] == 787){
        v2[x] <- "Digestive"
      }
      else if((v1[x] >= 250 & v1[x] < 251)){
        v2[x] <- "Diabetes"
```

```
      }
      else if((v1[x] >= 800 & v1[x] <= 999)){
        v2[x] <- "Injury"
      }
      else if((v1[x] >= 710 & v1[x] <= 739)){
        v2[x] <- "Genitourinary"
      }
      else if((v1[x] >= 140 & v1[x] <= 239)){
        v2[x] <- "Neoplasms"
      }
      else
        v2[x] <- "Other"
    }
    else v2[x] <- "Missing"

  }
no_dup$diag_1 <- as.factor(v2)
```

```
v1 <- as.numeric(no_dup$diag_2)
```

```
## Warning: NAs introduced by coercion
```

```
v2 <- vector()
  for ( x in 1:length(v1)){
    if(!is.na(v1[x])) {
      if((v1[x] >= 390 & v1[x] <= 459) | v1[x] == 785){
        v2[x] <- "Circulatory"
      }
      else if((v1[x] >= 450 & v1[x] <= 519) | v1[x] == 786){
        v2[x] <- "Respiratory"
      }
      else if((v1[x] >= 520 & v1[x] <= 579) | v1[x] == 787){
        v2[x] <- "Digestive"
      }
      else if((v1[x] >= 250 & v1[x] < 251)){
        v2[x] <- "Diabetes"
      }
      else if((v1[x] >= 800 & v1[x] <= 999)){
        v2[x] <- "Injury"
      }
      else if((v1[x] >= 710 & v1[x] <= 739)){
        v2[x] <- "Genitourinary"
      }
      else if((v1[x] >= 140 & v1[x] <= 239)){
        v2[x] <- "Neoplasms"
      }
      else
        v2[x] <- "Other"
    }
    else v2[x] <- "Missing"

  }
no_dup$diag_2 <- as.factor(v2)
```

```
#diagnose <- function(v1){
v1 <- as.numeric(no_dup$diag_3)
```

```
## Warning: NAs introduced by coercion
```

```
v2 <- vector()
  for ( x in 1:length(v1)){
    if(!is.na(v1[x])) {
      if((v1[x] >= 390 & v1[x] <= 459) | v1[x] == 785){
        v2[x] <- "Circulatory"
      }
      else if((v1[x] >= 450 & v1[x] <= 519) | v1[x] == 786){
        v2[x] <- "Respiratory"
      }
      else if((v1[x] >= 520 & v1[x] <= 579) | v1[x] == 787){
        v2[x] <- "Digestive"
      }
      else if((v1[x] >= 250 & v1[x] < 251)){
        v2[x] <- "Diabetes"
      }
      else if((v1[x] >= 800 & v1[x] <= 999)){
        v2[x] <- "Injury"
      }
      else if((v1[x] >= 710 & v1[x] <= 739)){
        v2[x] <- "Genitourinary"
      }
      else if((v1[x] >= 140 & v1[x] <= 239)){
        v2[x] <- "Neoplasms"
      }
      else
        v2[x] <- "Other"
    }
    else v2[x] <- "Missing"

}
no_dup$diag_3 <- as.factor(v2)
#}
```

Change other variables into factors

```
no_dup[d_header] <- lapply(no_dup[d_header], factor)
```

Collapsing Levels

Here we change readmittance variable.

```
no_dup$readmitted <- factor(no_dup$readmitted)
levels(no_dup$readmitted) <- list( No=c(">30", "NO"), Yes=c("<30"))
#no_dup[1:20,40]
#no_dup$readmitted <- as.numeric(no_dup$readmitted)
# no= 1, yes = 2
```

Set Aside Test and Train Data

```
# Random sampling
samplesize = 0.60 * nrow(no_dup)
set.seed(80)
index = sample( seq_len ( nrow ( no_dup ) ), size = samplesize )

# Create training and test set
datatrain = no_dup[ index, ]
datatest = no_dup[ -index, ]
```

First Model

Regression

Look at the distribution of attributes, and percent of readmission within attribute groups

```
count(no_dup, diag_1, readmitted)
```

```
## # A tibble: 18 x 3
##           diag_1 readmitted     n
##           <fctr>     <fctr> <int>
##  1   Circulatory         No 19816
##  2   Circulatory        Yes  2077
##  3      Diabetes         No  5280
##  4      Diabetes        Yes   525
##  5     Digestive         No  6048
##  6     Digestive        Yes   522
##  7 Genitourinary         No  3739
##  8 Genitourinary        Yes   341
##  9        Injury         No  4271
## 10        Injury        Yes   506
## 11       Missing         No   813
## 12       Missing        Yes   126
## 13     Neoplasms         No  2511
## 14     Neoplasms        Yes   231
## 15         Other         No 13659
## 16         Other        Yes  1274
## 17   Respiratory         No  9085
## 18   Respiratory        Yes   691
```

```
count(no_dup, diag_1)
```

```
## # A tibble: 9 x 2
##           diag_1     n
##           <fctr> <int>
## 1   Circulatory 21893
## 2      Diabetes  5805
## 3     Digestive  6570
## 4 Genitourinary  4080
## 5        Injury  4777
## 6       Missing   939
## 7     Neoplasms  2742
## 8         Other 14933
```

```
## 9   Respiratory  9776
```

```
readmit <- table(datatest$readmitted)
prop.table(readmit,)*100
```

```
##
##        No       Yes
## 91.050828  8.949172
```

```
diag1 <- table(no_dup$diag_1, no_dup$readmitted)
prop.table(diag1,1)*100
```

```
##
##                        No       Yes
##   Circulatory   90.512949  9.487051
##   Diabetes      90.956072  9.043928
##   Digestive     92.054795  7.945205
##   Genitourinary 91.642157  8.357843
##   Injury        89.407578 10.592422
##   Missing       86.581470 13.418530
##   Neoplasms     91.575492  8.424508
##   Other         91.468560  8.531440
##   Respiratory   92.931669  7.068331
```

Call linear regression on Diagnosis only.

```
contrasts(no_dup$readmitted)
```

```
##     Yes
## No    0
## Yes   1
```

```
no_dup.fit = glm(readmitted ~ diag_1, data=datatrain, family=binomial)
#summary(no_dup.fit)
```

Test predictions with the simple model

```
trainprob1 = predict(no_dup.fit, datatest, type="response")
testprob1 = predict(no_dup.fit, datatest, type="response")
predtrain1 = rep("No", dim(datatrain)[1])
predtest1 = rep("No", dim(datatest)[1])
predtrain1[trainprob1 > .5] = "Yes"
predtest1[testprob1 > .5] = "Yes"

table(predtrain1, datatrain$readmitted)
```

```
##
## predtrain1    No   Yes
##         No 39176  3733
```

```
table(predtest1, datatest$readmitted)
```

```
##
## predtest1    No   Yes
##        No 26046  2560
```

```
"Training Accuracy"
```

```
## [1] "Training Accuracy"
```

```r
mean(predtrain1 == datatrain$readmitted)
```

```
## [1] 0.9130019
```

```r
"Testing Accuracy"
```

```
## [1] "Testing Accuracy"
```

```r
mean(predtest1 == datatest$readmitted)
```

```
## [1] 0.9105083
```

So we're getting 91% accuracy, BUT that's not from any predictive power of our model, its just predicting that no patients will be readmitted. Let's try to add more variables to our model.

Call linear regression on Diagnosis and A1C result only.

```r
no_dup2.fit = glm(readmitted ~ diag_1 + A1Cresult, data=datatrain, family=binomial)
#summary(no_dup2.fit)
```

Test predictions with the simple model

```r
trainprob2 = predict(no_dup2.fit, datatest, type="response")
testprob2 = predict(no_dup2.fit, datatest, type="response")
predtrain2 = rep("No", dim(datatrain)[1])
predtest2 = rep("No", dim(datatest)[1])
predtrain2[trainprob2 > .5] = "Yes"
predtest2[testprob2 > .5] = "Yes"

table(predtrain2, datatrain$readmitted)
```

```
##
## predtrain2    No   Yes
##         No 39176  3733
```

```r
table(predtest2, datatest$readmitted)
```

```
##
## predtest2    No   Yes
##        No 26046  2560
```

```r
"Training Accuracy"
```

```
## [1] "Training Accuracy"
```

```r
mean(predtrain2 == datatrain$readmitted)
```

```
## [1] 0.9130019
```

```r
"Testing Accuracy"
```

```
## [1] "Testing Accuracy"
```

```r
mean(predtest2 == datatest$readmitted)
```

```
## [1] 0.9105083
```

Still not predicting anything useful. Use more terms.

Use all remaining attributes

```r
no_dup3.fit = glm(readmitted ~., data=datatrain, family=binomial)
#summary(no_dup3.fit)
```

Test predictions with the simple model

```
trainprob3 = predict(no_dup3.fit, datatest, type="response")
testprob3 = predict(no_dup3.fit, datatest, type="response")
predtrain3 = rep("No", dim(datatrain)[1])
predtest3 = rep("No", dim(datatest)[1])
predtrain3[trainprob3 > .5] = "Yes"
predtest3[testprob3 > .5] = "Yes"

table(predtrain3, datatrain$readmitted)
```

```
##
## predtrain3    No   Yes
##         No 39176  3733
```

```
table(predtest3, datatest$readmitted)
```

```
##
## predtest3    No   Yes
##        No 26046  2560
```

```
"Training Accuracy"
```

```
## [1] "Training Accuracy"
```

```
mean(predtrain3 == datatrain$readmitted)
```

```
## [1] 0.9130019
```

```
"Testing Accuracy"
```

```
## [1] "Testing Accuracy"
```

```
mean(predtest3 == datatest$readmitted)
```

```
## [1] 0.9105083
```

I think the data is too unbalanced, and the best accuracy is still by predicting that no patients will be readmitted.

Check for balanced data

After balancing the data we can run the models.

```
balanced <- downSample(datatrain[1:22], datatrain$readmitted, list = FALSE, yname = "readmitted")
balanceup <- upSample(datatrain[1:22], datatrain$readmitted, list = FALSE, yname = "readmitted")
```

```
table(balanced$readmitted)
```

```
##
##   No  Yes
## 3733 3733
```

```
table(balanceup$readmitted)
```

```
##
##    No   Yes
## 39176 39176
```

Use all predictors

```
balanced.fit = glm(readmitted ~., data=balanced, family=binomial)
balanceup.fit =  glm(readmitted ~., data=balanceup, family=binomial)
#summary(balanced.fit)
#summary(balanceup.fit)
```

Test predictions with the simple model

```
trainprobdown = predict(balanced.fit, datatest, type="response")
testprobdown = predict(balanced.fit, datatest, type="response")
predtraindown = rep("No", dim(datatrain)[1])
predtestdown = rep("No", dim(datatest)[1])
predtraindown[trainprobdown > .5] = "Yes"
predtestdown[testprobdown > .5] = "Yes"

table(predtraindown, datatrain$readmitted)
```

```
##
## predtraindown    No   Yes
##           No  20638  2012
##           Yes 18538  1721
```

```
table(predtestdown, datatest$readmitted)
```

```
##
## predtestdown    No   Yes
##          No  14229  1096
##          Yes 11817  1464
```

```
"Training Accuracy"
```

```
## [1] "Training Accuracy"
```

```
mean(predtraindown == datatrain$readmitted)
```

```
## [1] 0.5210795
```

```
"Testing Accuracy"
```

```
## [1] "Testing Accuracy"
```

```
mean(predtestdown == datatest$readmitted)
```

```
## [1] 0.5485912
```

```
trainprobup = predict(balanceup.fit, datatest, type="response")
testprobup = predict(balanceup.fit, datatest, type="response")
predtrainup = rep("No", dim(datatrain)[1])
predtestup = rep("No", dim(datatest)[1])
predtrainup[trainprobup > .5] = "Yes"
predtestup[testprobup > .5] = "Yes"

table(predtrainup, datatrain$readmitted)
```

```
##
## predtrainup    No   Yes
##         No  21169  2034
##         Yes 18007  1699
```

```r
table(predtestup, datatest$readmitted)
```

```
##
## predtestup    No   Yes
##        No  14516  1117
##        Yes 11530  1443
```

```r
"Training Accuracy"
```

```
## [1] "Training Accuracy"
```

```r
mean(predtrainup == datatrain$readmitted)
```

```
## [1] 0.5329418
```

```r
"Testing Accuracy"
```

```
## [1] "Testing Accuracy"
```

```r
mean(predtestup == datatest$readmitted)
```

```
## [1] 0.55789
```