Data Mining for Diabetes Readmission Prediction

Siyuan Meng
siyuanme@usc.edu

December 8, 2015

Reproducibility

In order to get the same results, need certain set of packages, as well as setting a pseudo-random seed equal the one I used.

- The following libraries were used for this project:

```
library(caret)
```

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

```
library(randomForest)
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(pander)
```

- Here is the seed I set to generate pseudo-random numbers for spliting training and test dataset. (see `Preprocessing` section)

```
set.seed(12345)
```

Getting data

```
setwd("~/Documents/2015Fall/EE660/EE660_Project")
data <- read.csv("~/Documents/2015Fall/EE660/EE660_Project/diabetic_data.csv",
                 stringsAsFactors=T,na.strings = '?')
dim(data)
```

```
## [1] 101766     50
```

Cleaning data

The original missing value information can be found in http://www.hindawi.com/journals/bmri/2014/781670/tab1/. For simplicity, only show features which have missing values. (cite)

Feature_name	Type	Discription	Propotional_missing
Race	Nominal	Values: Caucasian, Asian, African American, Hispanic, and other	2%
Weight	Numeric	Weight in pounds	97%
Payer code	Nominal	Integer identifier corresponding to 23 distinct values, for example, Blue Cross/Blue Shield, Medicare, and self-pay	40%
Medical specialty	Nominal	Integer identifier of a specialty of the admitting physician, corresponding to 84 distinct values	50%
Diagnosis 3	Nominal	Additional secondary diagnosis (coded as first three digits of ICD9), corresponding to 954 distinct values	1%

The way I deal with missing values is to delete `Weight`, `Payer code` feaures (columns), since both features have more than 50% missing values and they are not relevant to classification. However, `Medical speciality` was maintained, adding the value "missing" in order to account for missing values. (cite)

```
data <- data[,c(-6,-11)]
data$medical_specialty <- factor(data$medical_specialty,
                levels=c(levels(data$medical_specialty),'Missing'))
data[,10][is.na(data$medical_specialty)] <- 'Missing'
data <- na.omit(data)
dim(data)
```

```
## [1] 98053    48
```

In general, should split the original data into training and testing first and then deal with the missing values. However, both methods will yield same dimension of `data`. For simplicity, deal with NAs first here.

Preprocessing

(Preprocessing is more crucial when using model based algorithms, e.g. Linear Discrimant Analysis, Naive Bayes, Linear Regression... than using non-parametrical algorithms.)

- There are also other ways to impute data, like knnimpute... For convenience, try omitting NA rows first.
- The data has been preprocessed to ensure each encounter has a unique id. However, each patient may have more than one id.

```
length(unique(data$encounter_id))
```

```
## [1] 98053
```

```
length(unique(data$patient_nbr))
```

```
## [1] 68630
```

We thus used only one encounter per patient; in particular, we considered only the first encounter for each patient as the primary admission and determined whether or not they were readmitted within 30 days. (cite)

```
data <- data[order(data[,2]),]
unique_list <- array(TRUE,nrow(data))
for (l in 2:nrow(data)){
    if (data[l,'patient_nbr']==data[l-1,'patient_nbr']){
        unique_list[l]=FALSE
    }
}
data <- subset(data,unique_list)
```

- Kill the first two features(`encounter_id` and `patient_nbr`) which are ids for encounted and patients. Also, kill near zero variables, since they are not considered relevant to the outcome. Besides, extract label column, which is the last column of `data`.

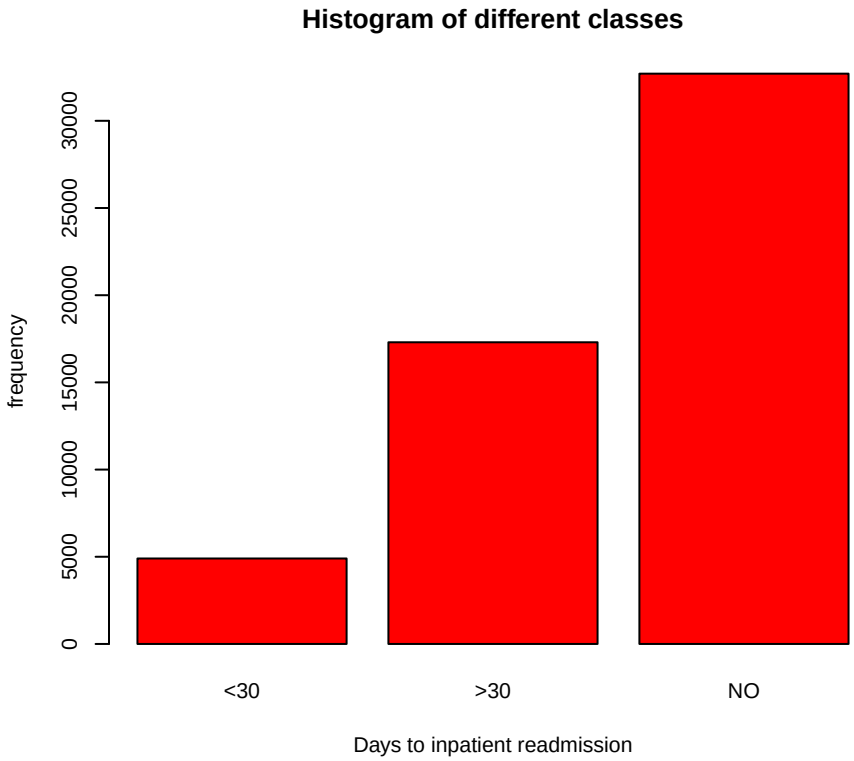
```
data <- data[,c(-1,-2)]
label <- data$readmitted
data <- data[,-46]
```

- Split this train data into training and test dataset according to the ratio 6:4 (set seed to make partioning reproducible) (Here I don't split validation since without looking data, there is possibility that test and training may not have the same distribution of different classes. Using cross validation is better in this case, though the computational complexity is high.)

```
inTrain <- createDataPartition(label,p=0.8,list=FALSE)
training <- data[inTrain,]
training_label <- label[inTrain]
test <- data[-inTrain,]
test_label <- label[-inTrain]
```

- Now, let's see if samples of different classes of `training` dataset are unbalanced.

```
par(cex=0.7,pin=c(4,3))
plot((training_label), xlab = 'Days to inpatient readmission', ylab = 'frequency',
     col = 'red', main = 'Histogram of different classes')
```



Classes are unbalanced distributed, but not very skewed.

- Kill unimportant features.(*nearZeroVar* diagnoses predictors that have one unique value (i.e. are zero variance predictors) or predictors that are have both of the following characteristics: they have very few unique values relative to the number of samples and the ratio of the frequency of the most common value to the frequency of the second most common value is large.) (cite)

```
NZV <- nearZeroVar(training,saveMetrics = T,freqCut=98/2)
training <- training[,-which(NZV$nzv==TRUE)]
```

- Kill same features for `test` dataset without looking inside.

```
test <- test[,-which(NZV$nzv==TRUE)]
```

- Use background information for all features to analyze which features should be converted to numerical features. (cite)
- Note: Need to combine training and test to a big dataset and then convert to numerical fearutes. If convert seperately, some feature may have some level which has only few points and are all assigned into `training` or `test` dataset. This may lead to different dimensions for `training` and `test` dataset after conversion.

```
source('~/Documents/2015Fall/EE660/EE660_Project/C2N.R')
data <- rbind(training,test)
for (i in c(1:6,8,15:17,19:29)){
    temp <- as.factor(data[,i])
    data <- cbind(data,C2N(temp))
}
data <- data[,-c(1:6,8,15:17,19:29)]
```

```
training <- data[1:length(training_label),]
test <- data[(length(training_label)+1):68630,]
```

- Again, kill unimportant features using *nearZeroVar*.

```
NZV_2 <- nearZeroVar(training,saveMetrics = T,freqCut=98/2)
training <- training[,-which(NZV_2$nzv==TRUE)]
test<- test[,-which(NZV_2$nzv==TRUE)]
```

- Normalize data (tried different methods, just write the best one I found here)

```
training <- apply(training,2,as.numeric)
test <- apply(test,2,as.numeric)
training_std <- apply(training,2,function(x) x/sum(x))
test_std <- apply(test,2,function(x) x/sum(x))
```

Save `training` and `test` into csv file for future use

```
write.csv(cbind(training_std,training_label),'training.csv',row.names = FALSE)
write.csv(cbind(test_std,test_label),'test.csv',row.names = FALSE)
```

Purpose for doing that is `R` is good for exploratory research, but not good at dealing with large dataset for clasification. Use `Python` to read csv as `SFrame` for classification will speed up! (cite)