Data Mining for Diabetes Readmission Rate Prediction

Siyuan Meng
siyuanme@usc.edu

December 8, 2015

Project Homepage

The link of project repository is https://github.com/siyuan1992/EE660_Project. For more information, please see `README.md`.

Abstract

The management of hyperglycemia in the hospitalized inpatient becomes increasing recognized, due to the morbidity and mortality outcome [1,2]. For most non-ICU (Inensive Care Unit) patients, anecdotal evidence says that inpatient managenent is arbitrary and often leads to either no treatment or wide fluctuations in glucose when traditional management strategies are employed [3,4]. Thus, protocols are recommended. Effective prediction on readmissions enables hospitals to identify and target patients at high risk [5]. Therefore, the goals are discovering major factors contributing to hospitals readmissions as well as finding the effective method to predict the type of readmissions. Top three most important features for prediction are `Number of inpatient visits`, `Admission type` and `Admission source`. The best method for classification is boosting-tree classifier. However, with data pre-mining and QUEST (Quick, Unbiased and Effecient Statistical Tree), there will be 5%~7% further accuray improvement.

Problem Statement and Goals

The data is from the Center for Clinical and Translational Research, Viginia Commonwealth University, which presents 10 years (1999-2008) of clinical care at 130 US hospitals and integrated delivery networks [6]. The goals here are identify the major factors that contribute to hospital readmissions as well as find the best model to predict the readmission type given inpatient information.

The difficulties of this problem are mainly in the following aspects:

- The dataset is relatively large (~100,000 samples, 49 features)
- Several features have large propotional missing data due to the fact that prior to the HITECH legislation of the American Reinvestment and Recovery Act in 2009 hospitals and clinics were not required to capture it in a structured format [6].
- Significant amounts of preprocessing required:
  - The preliminary dataet contained multiple inpaitent visits for some patients and the observations could not be considered statistically independent [paper].
  - Even though the preliminary dataset was extracted from the database with processing, lots of redundant features still exist.
  - Almost 80% features are nominal. Thus, after feature encoding, data will be expanded into a large dimension, which makes feature selection and classification extremely time consuming.
  - Class labels are hierarchical. Three types of readmission are `NO`, `>30`, `<30`, i.e., whether a patient will have readmission and if the patient have, a hospitalization occurs within 30 days or not.

Literature Review

The research artical [6] talked about the background information about th dataset and the information of each feature, which as a reference, helps me to deal with redundant features.

The other paper [5] analyze the influence of different features and use some tricks to improve the performance. The trick they used are listed here:

- Data pre-mining
  - Re-categorize readmission group from 3 groups to 2 groups (Readmission <30 days and Non-within 30 days.)
  - Subsample to make readmission group's ratio be 1:1
  - Review each feature's relationship with readmission
  - Review numeric variables correlation
- Use QUEST model to improve accuracy.

Several gruops use the similar trick, like recategorizing readmission groups to two (Readmission or not). However, data pre-mining is of upmost importance in improving the model accuracy. With data pre-mining and ensemble models, the highest accuracy could reach 80% [7].

Prior and Related Work

This project is exclusively for EE660.

Project Formulation and Setup

I tried several modals and algorithms for this project, e.g. Perceptron, SVM, logistic classifier, etc. I want to list two of them which have best performances.

- The first is gradient tree boosting classifier. The algorithm (pseudocode) is as follows [8]:

```
Input: training set $(\mathbf{x}_i, y_i)_{i=1}^{N}$, loss function $L(y, f(\mathbf{x}))$ and number of iteration M.
Algorithm:
I. Initialize model with a constant value:

    $$f_0(x) = \arg\min_{\gamma} \sum_{i=1}^{N} L(y_i, \gamma)$$

.
II. for m=1 to M:
1. Compute pesudo-residuals:

    $$r_{im} = -[\frac{\partial L(y_i, f(\mathbf{x}_i))}{\partial f(\mathbf{x}_i)}]_{f(\mathbf{x})=f_{m-1}(\mathbf{x})} \qquad for\ i = 1, ..., N$$

.
2. Train with training set $(\mathbf{x}_i, r_{im})_{i=1}^{N}$.
```

3. Compute γ_m by solving:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(\mathbf{x}_i) + \gamma h_m(\mathbf{x}_i))$$

.

4. Update the model:

$$f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \gamma_m h_m(\mathbf{x})$$

III. Output $F_M(\mathbf{x})$.

Parameters need to tune here are iteration number M, maximum depth of a tree and step size to solve γ. In general, boosting will not cause overfitting since it is a maximum margin classifier, which is fundamentally equivalent to SVM. However, boosting is a non-parametric, non-linear and non-binary method, which will lead to better result.

- The second method is neuralnet classifier.

There is no single formal definition of what an artificial neural network is. Typically, a class of statistical models could be called "neural" if it possesses the following characteristics [9]:

```
+ Contains sets of adaptive weights.
+ Capability of approximating non-linear functions of their inputs.
```

Training a neural network essentially means selecting one model form set of allowed models [wiki]. The general framework is:

```
+ Model representation
+ Feedforward and cost function
+ Backpropagation to compute the gradient for the cost function
+ Gradient checking (Optional)
+ Get learning parameters, output result
```

Parameters need to tune are number of iteration, number of layers, model parameters... In general, neural network may cause overfitting and the speed is pretty low. However, neural network is good for non-linear classification, and that's the reason why it has better performance here.

Since both two methods are computational complicated, the trick I used here is to use `graphlab_create` in python, which reads csv files in SFrame and classification is much faster than other packages and other programming languages. Thus, tuning parameters is not time-consuming. Also, due to the size of the dataset and the computation complexity, using validation set randomly generated from training set instead of using cross validation, though cross validation may lead to better accuracy.

Methodology

Here is my whole precedure for this project:

1. Download data online.
2. Set problem and goals.

3. Review revelant papers and online materials to have a better understand of data.
4. Preprocessing. This includes:
   - Deal with missing values.
   - Split the original data into training and test dataset with ratio 8:2. (No need to generate validation set here, since the classifier will directly use part of training dataset as validation.)
   - Subsample data. Considered only the first encounter for each patient as the primary admission and determined whether or not they were readmitted within 30 days. (Could do before splitting)
   - Remove irrelevant features and near-zero features (variables) for both training and test dataset.
   - Plot the histgram of class labels of training dataset. This is a imbalanced problem, so subsample training set to let different classes have same ratios. (Here, could try both over-sampling and under-sampling or give weights for different classes as a parameter to classifiers.)
   - Convert catogegorical features to numerical for both training and test dataset.
   - Feature dimension reduction for both training and test dataset.
   - Feature normalization.
5. Training and testing. This includes:
   - Set hypothesis set first. (For different model, the hypothesis sets are different.)
   - A learning algorithm is used on training dataset.
   - Couldn't choose the best hypothesis based only on training set, since this may cause overfitting. Instead, using validation set to choose the best hypothesis.
   - For different model, there will be different "best" hypothesis. Choose the best one based on the horizontal comparison.
   - Testing on test dataset.
6. Result interpretation.

Implementation

- 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 in R:

```r
library(caret)
library(randomForest)
library(pander)
library(psych)
library(FactoMineR)
library(ggplot2)
```

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

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

- Getting data

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

- Feature space The preliminary dataset has 101766 samples and 49 features. The original feature could be found in http://www.hindawi.com/journals/bmri/2014/781670/tab1/. Here, just talk about some important features.
  - `Race`, `Gender` and `Age` nominal features for a patient.
  - `Admission type`, `Discharge disposition` and `Admission source` are nominal features, which describes the hospitalization condition. In the preliminary dataset, these three features are given as integer identifier. The mappings are as follows.

admission_type_id	description
1	Emergency
2	Urgent
3	Elective
4	Newborn
5	Not Available
6	NULL
7	Trauma Center
8	Not Mapped

discharge_disposition_id	description
1	Discharged to home
2	Discharged/transferred to another short term hospital
3	Discharged/transferred to SNF
4	Discharged/transferred to ICF
5	Discharged/transferred to another type of inpatient care institution
6	Discharged/transferred to home with home health service
7	Left AMA
8	Discharged/transferred to home under care of Home IV provider
9	Admitted as an inpatient to this hospital
10	Neonate discharged to another hospital for neonatal aftercare
11	Expired
12	Still patient or expected to return for outpatient services
13	Hospice / home
14	Hospice / medical facility
15	Discharged/transferred within this institution to Medicare approved swing bed
16	Discharged/transferred/referred another institution for outpatient services
17	Discharged/transferred/referred to this institution for outpatient services
18	NULL
19	Expired at home. Medicaid only, hospice.
20	Expired in a medical facility. Medicaid only, hospice.
21	Expired, place unknown. Medicaid only, hospice.

discharge_disposition_id	description
22	Discharged/transferred to another rehab fac including rehab units of a hospital .
23	Discharged/transferred to a long term care hospital.
24	Discharged/transferred to a nursing facility certified under Medicaid but not certified under Medicare.
25	Not Mapped
26	Unknown/Invalid
30	Discharged/transferred to another Type of Health Care Institution not Defined Elsewhere
27	Discharged/transferred to a federal health care facility.
28	Discharged/transferred/referred to a psychiatric hospital of psychiatric distinct part unit of a hospital
29	Discharged/transferred to a Critical Access Hospital (CAH).

admission_source_id	description
1	Physician Referral
2	Clinic Referral
3	HMO Referral
4	Transfer from a hospital
5	Transfer from a Skilled Nursing Facility (SNF)
6	Transfer from another health care facility
7	Emergency Room
8	Court/Law Enforcement
9	Not Available
10	Transfer from critial access hospital
11	Normal Delivery
12	Premature Delivery
13	Sick Baby
14	Extramural Birth
15	Not Available
17	NULL
18	Transfer From Another Home Health Agency
19	Readmission to Same Home Health Agency
20	Not Mapped
21	Unknown/Invalid
22	Transfer from hospital inpt/same fac reslt in a sep claim
23	Born inside this hospital
24	Born outside this hospital
25	Transfer from Ambulatory Surgery Center

admission_source_id	description
26	Transfer from Hospice

  - Features with `Number of` are numerical features, which are records for encounters.
  - `Diagnosis 1,2,3` are nominal features encoded in a hierarchical way.
  - `Medical specialty` is an integer identifier of a specialty of the admitting physician, corresponding to 84 distinct values, for example, cardiology, internal medicine, family/general practice, and surgeon.

- 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.

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. [6]

```
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 and Feature Extraction (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. [6]

```
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. Besides, extract label column, which is the last column of `data`.

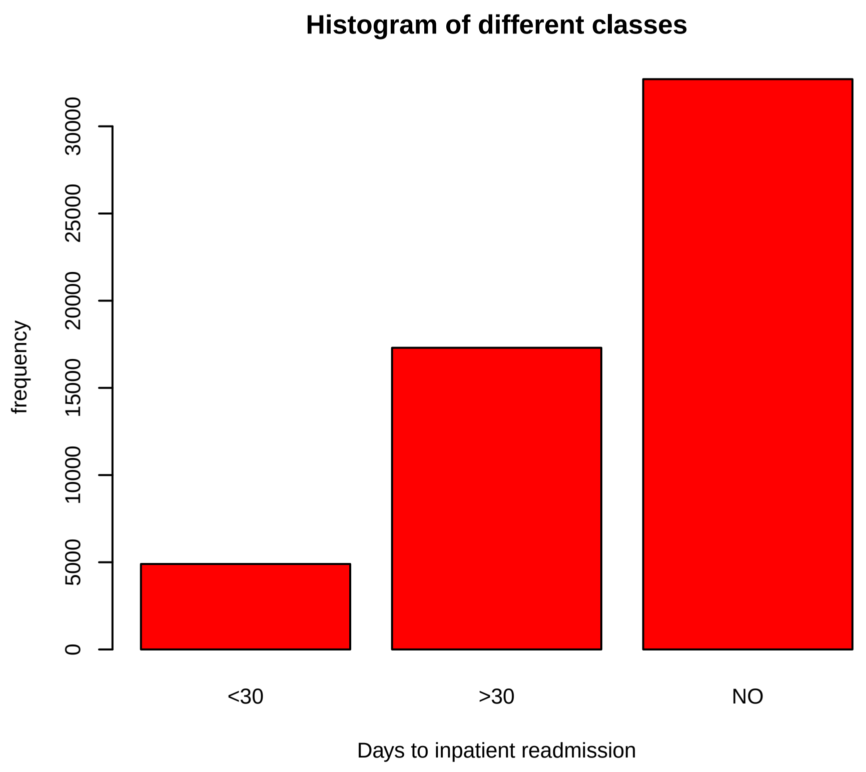
```
data <- data[,c(-1,-2)]
label <- data$readmitted
label_gen <- label
levels(label_gen) <- c('Readmission','Readmission','NO')
```

- Split this train data into training and test dataset according to the ratio 8:2 (set seed to make partioning reproducible).

```
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 in three levels, but not very skewed. Sampling data to deal with imbalanced classes, or to give weights in classifiers.
- 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.) [6]

```
NZV <- nearZeroVar(training,saveMetrics = T,freqCut=95/5)
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.(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.)
- First, drop large factors `diag2` and `diag3`.

```
    training <- training[,c(-16,-17)]
    test <- test[,c(-16,-17)]
```

- Second, drop last nine features.

```
    training <- training[,-c(17:25)]
    test <- test[,-c(17:25)]
```

– Third, convert categorical features to numerical features.

```
source('~/Documents/2015Fall/EE660/EE660_Project/C2N.R')
data <- rbind(training,test)
for (i in c(1:6,8,15)){
    temp <- as.factor(data[,i])
    data <- cbind(data,C2N(temp))
}
data <- data[,-c(1:6,8,15)]
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=95/5)
training <- training[,-which(NZV_2$nzv==TRUE)]
test<- test[,-which(NZV_2$nzv==TRUE)]
```

– Normalize data (There are lots of differnt methods for normalization,

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

- Training and testing are finished in python using `graphlab_create`. Take `boosting tree classifier` as an example. The way I choose parameters are in a heuristics way. For example, the higher iteration number, the higher the accuracy for training dataset. However, accuracy will not increase any more at some iteration for validation set. Thus, choose that iteration number as the best parameter. Same way to choose other parameters. The best parameters I used in `boosting tree classifier` are:
  – Number of iteration:30
  – Maximum depth of tree:6
  – Stepsize:0.3 Here, `boosting tree classifier` has the best performance.

Final results

The final accuracy using `boosting tree classifier` is around 70% accuracy, which is the best among all models I used. Since randomly assign class label will yield around 30% accuracy, 70% accuracy is a huge improvement. There are lots of researches using this dataset for classification. The best performance is around 70%~80% accuracy. However, the most importance factor for improving the result is data pre-mining instead of choosing more advanced models.

Summary and Conclusions

- Data pre-mining is of upmost importance in improving the model accuracy, especially for this imbalanced-class problem.
- Import factors for predicting readmission types are `admission source`, `admission type`, `discharge disposition` and `number of inpatient visits`.
- `Boosting tree classifier` is the most powerful classifier in this problem. In general, `boosting tree` a powerful tool in classification and regression.
- Hospitals are advised to concern not only inpatient treatment but also continuing care after discharge since `<30` is the least frequent class.

Reference

1. G. E. Umpierrez, S. D. Isaacs, N. Bazargan, X. You, L. M. Thaler, and A. E. Kitabchi, "Hyperglycemia: an independent marker of in-hospital mortality in patients with undiagnosed diabetes," Journal of Clinical Endocrinology and Metabolism, vol. 87, no. 3, pp. 978–982, 2002.
2. C. S. Levetan, M. Passaro, K. Jablonski, M. Kass, and R. E. Ratner, "Unrecognized diabetes among hospitalized patients," Diabetes Care, vol. 21, no. 2, pp. 246–249, 1998.
3. A. G. Pittas, R. D. Siegel, and J. Lau, "Insulin therapy for critically ill hospitalized patients: a meta-analysis of randomized controlled trials," Archives of Internal Medicine, vol. 164, no. 18, pp. 2005–2011, 2004.
4. A. C. Tricco, N. M. Ivers, J. M. Grimshaw et al., "Effectiveness of quality improvement strategies on the management of diabetes: a systematic review and meta-analysis," The Lancet, vol. 379, no. 9833, pp. 2252–2261, 2012.
5. Data Mining for Diabetes Readmission Prediction Team Evolution Yi Chun Chien, Xiayu Zeng, Hong Zhang, Yixi Zhang
6. Strack, B., DeShazo, J. P., Gennings, C., et al.: Impact of HbA1c measurement on hospital readmission rates: Analysis of 70,000 clinical database patient records. Biomed. Res. Int., 2014, 2014, ID 781670.
7. https://cepd.okstate.edu/files/11-Improving-the-Performance-of-two-stage-modeling.pdf.
8. https://en.wikipedia.org/wiki/Gradient_boosting.
9. https://en.wikipedia.org/wiki/Artificial_neural_network.