HbA1c measurements and hospital readmission rate

Michal Guzek
mguzek@uci.edu

Qingyu Song
qingys1@uci.edu

Yukang Wang
yukangw2@uci.edu

Abstract

HbA1c (Hemoglobin A1c) test is commonly used to diagnose diabetes. However, data for the years (1999–2008) shows [1] that HbA1c measurement was not performed frequently as part of inpatient care for both intensive care unit (ICU) and non-ICU settings. Measurements were performed only for 18.4% of the total number of admissions. This factor, along with the primary diagnosis being other than "diabetes", contributed to higher readmission rates for patients with actual diabetes, who were readmitted to hospitals within the next 30 days. This paper builds on [1] and describes a combination of Machine Learning models (along with our own ensembles of those models) which aim to improve prediction of readmission rate for patients with diabetes. This amounts to a binary classification problem.

In the following work, Michal Guzek was responsible for performance validation of random forests, ensemble methods and analyzing the origins of class overlap. Qingyu Song analyzed in depth different trade-offs between Neural Network's hyperparameters. Yukang Wang performed data exploration and all the necessary feature engineering and data cleaning.

1 Data exploration

To get a deeper understanding of the data, we performed some visualizations of the data. Figure 1 includes the figures of the five main numerical variables which we believe to be relevant. In these figures, we draw scatter plots to show the pairwise relationship between the five main numerical variables. Also, to find out how variables affected the readmission rate, we mark the patients who were readmitted within the next 30 days with orange and mark other patients with blue.

According to Figure 1, we can see that there does exist some relationship between the numerical data and the readmission rate. In the figure for "num_procedures" and "number_outpatient", we can clearly see that the readmitted people tend to cluster at a low level of "number_outpatient" for different number of procedures. Also, in the figure for "num_outpatient" and "number_inpatient", we can still see this trend. So the number of outpatients may play an important role in predicting the readmission rate.

2 Performance validation

Given the fact that we were faced with a binary classification problem, we had to choose between accuracy and AUC metrics in order to assess performance of our models. While accuracy of our different models was approximately at the same level, 0.88, for all the hyperparameter combinations (this trend is visible on Figure 2), it was not a good error metric for our data as the labels ("readmitted", "not readmitted") were heavily imbalanced towards the "not readmitted" side. Indeed, there were

Figure 1: Scatter Plot for Main Numerical Variables

$90,409$ patients in our data set who were not readmitted to the hospital within the next 30 days and $11,357$ who were. Accordingly, while it would be relatively easy for most of the models to predict "not readmitted" labels, a low sensitivity rate would still cause accuracy to take on high values. Thus, we deployed AUC as a metric for all our models. It is a more comprehensive validation metric in our case, as it evaluates a given classifier on all possible threshold values.

3 Data preprocessing and feature design

The feature engineering process of the raw data is summarized in Table 1. In this project, we also did some pre-processing to the dataset. Since there is much missing data in the variables "Weight", "Payer code" and "Medical specialty", we chose to disregard those variables. Naturally, we also chose to drop the variables "Encounter ID" and "Patient number" for the purpose of performing readmission prediction. For other numerical data, we chose to keep them in the same form, while for nominal data, we one-hot encoded them. For the variables "Diagnosis 1", "Diagnosis 2" and "Diagnosis 3", we did the same transformation as in [1] to put thousands of different diagnoses into several main groups so that the information of the relationship between diagnosis can be exploited. The diagnosis transformation is shown in Table 2.

Lastly, we had to decide on a feature combination to feed into our models. Because it would be computationally intractable to test models on every possible combination of all the features, we measured the AUC of different combinations of scalar features such as "time_in_hospital", "num_lab_procedures"... and included each of the remaining categorical features such as "A1Cresult", "race", "gender", "diagnosis" in each of the combinations as we believed each of the categorical features is indeed relevant in order to predict diabetes-related implications in the long run. We started off by comparing the AUCs of a logistic regression model on those combinations and later confirmed that Neural Network models also achieved one of the largest AUC values on this particular combination: ["num_procedures", "number_outpatient", "number_emergency", "number_inpatient", "number_diagnoses", "diag_*" + all the categorical features], just as the logistic regression.

Table 1: Processing of raw data

Feature name	Type	Process
Encounter ID	Numeric	Drop, not practical for prediction
Patient number	Numeric	Drop, not practical for prediction
Race	Nominal	One-Hot Encode
Gender	Nominal	One-Hot Encode
Age	Nominal	One-Hot Encode
Weight	Numeric	Drop, because of high missing rate (97%)
Admission type	Nominal	One-Hot Encode
Discharge disposition	Nominal	One-Hot Encode
Admission source	Nominal	One-Hot Encode
Time in hospital	Numeric	Keep
Payer code	Nominal	Drop, because of high missing rate (52%)
Medical specialty	Nominal	Drop, because of high missing rate (53%)
Number of lab procedures	Numeric	Keep
Number of procedures	Numeric	Keep
Number of medications	Numeric	Keep
Number of outpatient visits	Numeric	Keep
Number of emergency visits	Numeric	Keep
Number of inpatient visits	Numeric	Keep
Diagnosis 1	Nominal	Group, One-Hot Encode
Diagnosis 2	Nominal	Group, One-Hot Encode
Diagnosis 3	Nominal	Group, One-Hot Encode
Number of diagnoses	Numeric	Keep
Glucose serum test result	Nominal	One-Hot Encode
A1c test result	Nominal	One-Hot Encode
Change of medications	Nominal	One-Hot Encode
Diabetes medications	Nominal	One-Hot Encode
24 features for medications	Nominal	One-Hot Encode
Readmitted	Nominal	Let '<30'=1, 'No'='>30'=0

Table 2: Diagnosis transformations

Group name	icd9 codes
Circulatory	390-459,785
Respiratory	460-519,786
Digestive	520-579,787
Diabetes	250.xx
Injury	800-999
Musculoskeletal	710-739
Genitourinary	580-629,788
Neoplasms	140-239
Other	680-709, 782, 001-139, 290-319, E-V, 280-289, 320-359 , 630- 679, 360-389, 740-759, 789, 783,365.44

4 Model exploration

Having trained a logistic regression model to help us determine the initial feature combination, and later used Neural Network models to confirm that they achieved similar performance on that particular combination, here is how we proceed: At first, we measured the performance of Random Forest classifiers for different choices of hyperparameters. We discovered that the hyperparameter "n_estimators" (number of trees in the forest) accounted for most of the differences in obtained AUCs. Models that performed best had "n_estimators" of magnitude 100. Secondly, we established that the optimal number of the hidden layers in MLPClassifier is two (10 nodes each) and larger numbers of hidden layers did not contribute to AUC significantly. We also explored the combinations of the remaining hyperparameters (activation function for the hidden layer, optimization algorithm and learning rate schedule for weight updates) for the Neural Network and in each such iteration computed the accuracy and AUC, which were plotted in Figure 2.

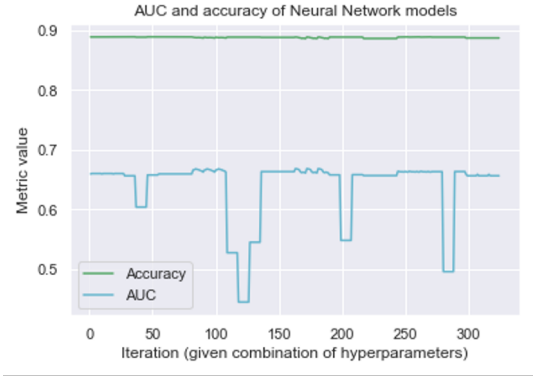

Figure 2: AUC and accuracy of NN models

We established that L-BFGS [2] optimization algorithm achieved best results in our MLPClassifier. While the computational cost is definitely higher than in case of SGD, it tends to better approximate the optimum and works better with sparse datasets, such as ours. Finally, we combined all of the aforementioned models, that is: Random Forest, Neural Network and logistic regression, into different ensembles. We deployed four such versions: two weighted ensembles (assigning more weights to either Random Forest and Neural Network model) and two unweighted ones, one of which combined Random Forest and NN models and the other included the logistic regressor on top of the two previous ones. Although the differences were relatively thin, the best result in terms of AUC was produced by the unweighted combination of all three models, as Table 3 indicates.

Table 3: Ensemble results

Ensemble	AUC
Unweighted Random Forest and NN	0.68136
Weighted Random Forest (75%) and NN (25%)	0.68131
Weighted Random Forest (25%) and NN (75%)	0.67891
Unweighted Random Forest, NN and logistic classifier	0.68170

For comparison, no single random forest and neural network model achieved better AUC than 0.67 and 0.66, respectively, so all the ensembles did improve the prediction performance to some extent.

5 Origins of class overlap

At the end, we decided to use 5-fold cross validation to compare the average performance (AUC) of our three models: logistic regression, Random Forest and Neural Network. The results for each of the three cross validations (the average AUC) coincided with the respective original model, which was evaluated on test data that accounted for 20% of the overall data. Therefore, we concluded that the data is obviously not perfectly separable, which is common in practice when the classes overlap due to the nature of the problem.

References

[1] Beata Strack, Jonathan P. DeShazo, Chris Gennings, Juan L. Olmo, Sebastian Ventura, Krzysztof J. Cios, John N. Clore, "Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records", BioMed Research International, vol. 2014, Article ID 781670, 11 pages, 2014. https://doi.org/10.1155/2014/781670

[2] Quoc V. Le, Jiquan Ngiam, Adam Coates, Abhik Lahiri, Bobby Prochnow, and Andrew Y. Ng. 2011. On optimization methods for deep learning. In Proceedings of the 28th International Conference on International Conference on Machine Learning (ICML'11). Omnipress, Madison, WI, USA, 265–272.

Appendix

```
# Sample code snippets

# Data cleaning and feature engineering
import numpy as np
import pandas as pd
data=pd.read_csv('diabetic_data.csv')
data.replace('?',np.nan,inplace=True)

Ylist=['readmitted']
#idList=['encounter_id','patient_nbr']
numList=['time_in_hospital','num_lab_procedures','num_procedures',\
         'num_medications','number_outpatient','number_emergency',\
         'number_inpatient','number_diagnoses']
dumList=['race','gender','age',\
         'max_glu_serum','A1Cresult',\
         'metformin','repaglinide','nateglinide','chlorpropamide',\
         'glimepiride','acetohexamide','glipizide','glyburide','tolbutamide',\
         'pioglitazone','rosiglitazone','acarbose','miglitol','troglitazone',\
         'tolazamide','examide','citoglipton','insulin','glyburide-metformin',\
         'glipizide-metformin','glimepiride-pioglitazone','metformin-rosiglitazone',\
         'metformin-pioglitazone','change','diabetesMed']
dumIDList=['admission_type_id','discharge_disposition_id','admission_source_id']
diagList=['diag_1','diag_2','diag_3']

data.loc[:,dumIDList]=data.loc[:,dumIDList].astype('str')
dumList=dumIDList+dumList

changelist=dict(\
[(str(v), 'Circulatory') for v in (list(range(390,460))+[785])]+\
[(str(v), 'Respiratory') for v in (list(range(460,520))+[786])]+\
[(str(v), 'Digestive') for v in (list(range(520,580))+[787])]+\
[(str(v), 'Injury') for v in (list(range(800,1000)))]+\
[(str(v), 'Musculoskeletal') for v in (list(range(710,740)))]+\
[(str(v), 'Genitourinary') for v in (list(range(580,630))+[788])]+\
[(str(v), 'Neoplasms') for v in (list(range(140,240)))]+\
[(str(v),'Other') for v in ([780,781,784]+list(range(790,800))+list(range(240,250))+\
                            list(range(251,280))+list(range(680,710))+[782]+\
                            list(range(1,140))+list(range(290,320))+list(range(280,290))+\
                            list(range(320,360))+list(range(630,680))+list(range(360,390))+\
                            list(range(740,760))+[789,783,365.44]\
                           )])
for i in diagList:
    data[i].replace('250.*','Diabetes' ,regex=True ,inplace=True )
    data[i].replace('[EV].*','Other' ,regex=True,inplace=True  )
    data[i].replace(changelist,inplace=True)


#dataID=data.loc[:,idList]
dataDum=pd.get_dummies(data.loc[:,dumList])
dataNum=data.loc[:,numList]
dataDiag=(pd.get_dummies(data['diag_1'])+pd.get_dummies(data['diag_2'])+pd.get_dummies(data['diag_3']))
```

```
dataDiag.columns='diag_'+dataDiag.columns

X=pd.concat([dataNum,dataDiag,dataDum],axis=1)
Y=data.loc[:,Ylist]

# Ensembles

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import roc_auc_score

X_best_comb = X.filter(regex=("|".join(['num_procedures','number_outpatient','number_emergency',
'number_inpatient','number_diagnoses','diag_*'] + [regex+'*' for regex in dumList])))

X_train, X_test, y_train, y_test = train_test_split(X_best_comb,
Y_transformed['readmitted'].values, test_size=0.20, random_state=6)

classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)

randomForest = RandomForestClassifier(criterion='entropy',n_estimators=90, max_depth=None,
min_samples_split=2, min_samples_leaf=15, max_features='sqrt', bootstrap=True)
randomForest.fit(X_train, y_train)

clf = MLPClassifier(solver='lbfgs', activation='tanh', max_iter=200, learning_rate='constant',
hidden_layer_sizes=(10,10), random_state=1)
clf.fit(X_train, y_train)

auc_unweighted = roc_auc_score(y_test,
0.5*(randomForest.predict_proba(X_test)[:,1] + clf.predict_proba(x_te)[:,1]))

auc_weighted_towards_rf = roc_auc_score(y_test,
0.5*(0.75*(randomForest.predict_proba(X_test)[:,1]) + 0.25*(clf.predict_proba(x_te)[:,1])))

auc_weighted_towards_nn = roc_auc_score(y_test,
0.5*(0.25*(randomForest.predict_proba(X_test)[:,1]) + 0.75*(clf.predict_proba(x_te)[:,1])))

auc_combined_all = roc_auc_score(y_test,
(1/3)*(randomForest.predict_proba(X_test)[:,1] +
clf.predict_proba(x_te)[:,1] +
classifier.predict_proba(X_test)[:,1]))

# 5-fold cross validations

from sklearn.model_selection import cross_val_score

clf = MLPClassifier(solver='lbfgs', activation='tanh', max_iter=200, learning_rate='constant',
hidden_layer_sizes=(10,10), random_state=1)

crossval_scores_NN = cross_val_score(clf, X_best_comb, Y_transformed['readmitted'].values,
scoring='roc_auc', cv=5)

print('Mean crossval_scores_NN:', crossval_scores_NN.mean())

randomForest = RandomForestClassifier(criterion='entropy',n_estimators=90, max_depth=None,
min_samples_split=2,min_samples_leaf=15, max_features='sqrt', bootstrap=True)

crossval_scores_RF = cross_val_score(randomForest, X_best_comb, Y_transformed['readmitted'].values,
scoring='roc_auc', cv=5)

print('Mean crossval_scores_RF:', crossval_scores_RF.mean())

classifier = LogisticRegression(random_state = 0)
```

```
crossval_scores_LR = cross_val_score(classifier, X_best_comb, Y_transformed['readmitted'].values,
scoring='roc_auc', cv=5)

print('Mean crossval_scores_RF:', crossval_scores_LR.mean())
```