Predicting Readmission for Diabetes Patient

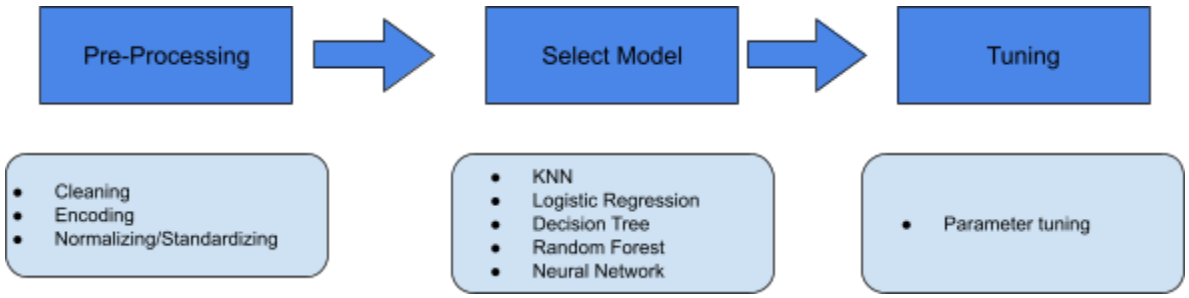
Abstract

We propose a way to predict the readmission for diabetes patients through a series of clinical parameters. Our object is to compare and evaluate the performance of different models on the UCI diabetes dataset*. We conduct feature engineering and hyper parameter tuning to improve the performance of the selected model, achieving auc of 0.67 on the testing dataset.

1 Introduction

According to the CDC, diabetes affects an estimated 29.1 million people in the United States and is the 7th leading cause of death. In the data set, 54.25% of the patients are readmitted shortly after they are discharged from the hospital. And according to HCUP, readmissions were associated with about $41.3 billion in hospital costs in 2011. We believe that there is some correlation between a patient's clinical parameters and the readmission. In this project, we are evaluating different ways of feature engineering and trying out several models to predict whether a patient will be readmitted to the hospital within 30 days after discharge.

1.1 Overall Process/Pipeline



2 Exploratory Data Analysis

Dataset contains records of 101,766 patients and each record entry contains 49 clinical parameters and the readmission result. Readmission result is a categorical data with 3 potential classes {No, <30, >30}, meaning that patient is not readmitted to the hospital, admitted to the hospital within 30 days after discharged, admitted to the hospital after 30 days after discharged respectively. 39 out of 49 features are categorical data. It is worth noting that some categorical data contains too many classes and some features are mixed with categorical data and numerical data. Discharge_disposition_id indicated the reason that the patient is discharged from the hospital, we dropped some entries that patients are either expired or transferred to some other facilities that they are no longer able to be readmitted to the hospital.

2.1 Feature Engineering

At the beginning, we replace all missing numbers filled with '?' with a nan representation. Then, we applied different approaches for different features as some of the categorical data have way too many classes.

- **Dropped**: (Features below are dropped, given that some of the features are just patient identification information, some are numerical mixed with categorical data, and some have way too many null values.)
    - encounter_id, patient_nbr, examide, citoglipton, payer_code

- **Assign each class with a number**: (Since data in these features below is categorical data, in each feature, we assign each unique classes to a digit, and finally we replace original data with these digits we've assigned in our dataset.)
    - race, gender, age, weight, admission_type_id, discharge_disposition_id, admission_source_id, time_in_hospital, payer_code, max_glu_serum, A1Cresult, metformin, repaglinide, nateglinide, chlorpropamide, glimepiride, acetohexamide, glipizide, glyburide, tolbutamide, pioglitazone, rosiglitazone, acarbose, miglitol, troglitazone, tolazamide

- **Combined some classes**: (Features below have too many unique classes. We don't want to keep all of these different classes because many of them only have a few samples. In that case, we decide to keep only 11 classes (the top 10 classes + an "Other" category that groups all the other classes).)
    - Medical_specialty, num_lab_procedures

- **Keep**: (Features below are kept, we don't do anything with them. This is because they are already digital data and can be used to train models directly.)
    - Num_procedures, *num_medications,* num_procedures, *num_medications*,number_outpatient, number_emergency, number_inpatient, number_diagnoses

Also, we have successfully converted **diag_1**, **diag_2**, **diag_3** into 10 categories according to the table: https://www.hindawi.com/journals/bmri/2014/781670/tab2/. We believe that this

is a very important feature which indicates the symptoms of the patient like disease for circulatory, respiratory, digestive system and etc.

2.2 Data Normalization

Numerical Variables	Min	Max	Mean	Median	SD
num_lab_procedures	1	132	43	44	19
num_procedures	0	6	1.3	1	1.7
num_medications	1	81	16	15	8
number_outpatient	0	42	0.37	0	1.2
number_emergency	0	76	0.19	0	0.93
number_inpatient	0	21	0.63	0	1.26
number_diagnoses	1	16	7.42	8	1.9

By observing the statistics of different numerical features, we found out that some features are skewed and some have outliers. We removed the outliers by filtering out all data that is 3 SD away from the mean (keep central 99.7%).

3.1 Model Exploration

We shuffle and split the dataset into:

a) 70% of the entire Dataset for training (Training data)
b) 30% of the entire Dataset for validation (Validation data)

Then, we train several models we've learnt in this machine learning course and use some techniques to try to optimize them. Finally, we select the best performing model based on the performance on the validation data.

Model Name	Training Error	Validation Error	AUC
K Nearest Neighbors	0.644	0.227	0.602
Logistic Regression	0.477	0.480	0.635
Decision Tree	0.802	0.337	0.622
Random Forest	0.763	0.302	0.670
Neural Network	0.473	0.471	0.632

Since KNN and LR are too simple and don't perform well, we'll not optimize these two models. Also, we won't optimize Decision Tree since it performs worse than Random Forest.

3.2 Hyperparameter Tuning

One technique we use to optimize our models is called **Hyperparameter Optimization** or **Tuning** which can help us choose a set of optimal parameters for specific models easily.

Random Forest:
We use RandomForestClassifier() to train RF. We start training with default parameters and RF has better performance than KNN and LR even though we haven't optimized it. Then we use hyperparameter tuning to try to find optimal parameters which can make RF perform best. Finally, we find that when max_depth = 14, min_sample_split = 6, and n_estimators = 200, our RF performs best. The highest AUC we get is 0.666. When we try to optimize neural network, we reconsider how to handle these features on dataset and do a lot to improve it. During this process, we are pleasantly surprised to find that RF also improved a bit, and optimized ACU ≈ 0.67.

Neural Network:
Like what we do for RF, at first we train NN with default parameters and then we use hyperparameter tuning to optimize it. However, although we replace default parameters with optimal parameters(activation = 'tanh', hidden_layer_sizes = (80,), solver = 'adam') we find, the performance of NN is still not very good (AUC≈0.54). We guess if it's because we didn't handle raw data well. So we come back to data exploration and check if we can improve pre-processing so that we could also improve model performance. We re-add some features to our dataset which we just dropped at first, such as diag_123. Besides, we implement data normalization which we think could also improve our performance. Fortunately, after we improved data exploration, our model performance also improved a lot. For Neural Network, we finally get our AUC = 0.632. But it still performs worse than what we expect. Even it doesn't perform much better than KNN and logistic regression. So we can conclude that neural network doesn't perform well on this dataset.

4 Conclusion

We applied different approaches in pre-processing data and our best model achieved AUC = 0.67.