# CS 273A
# Final Project Report:
# Diabetes 130-US Hospital

Fall 2018

Group name: DRT

Fangqi Liu
Te-Chih (Annie) Chen
Jonathan Harijanto

# 1. Introduction

Hospital readmission is an expensive process and reflects a poor quality in a healthcare system. In the United States alone, the cost of readmission diabetic patients could exceed 250 million dollars annually [1]. The goal of this project is to tackle the existing problem by analyzing a healthcare data and utilizing machine learning models to provide early identification of patients facing a high risk of readmission. Being able to determine which factors drive to higher readmission in a patient, and at the same time, predict which patients will get readmitted can help hospitals in the United States to save millions of dollars while improving the quality of care [2] [3]. In this paper, we are going to explain how we explore, cleanup and transform the features in the dataset, how we select which classifier models are suitable for this problem, and how we tune our classifier model to yield the highest prediction accuracy possible. Moreover, we are also going to briefly cover an interesting finding that was found during the experiment. At the end of this project, we will be able to understand: (1) what factors are the strongest predictors of hospital readmission in diabetic patients? (2) how accurate can machine learning models predict hospital readmission using a given dataset?

# 2. Feature (Dataset)

## 2.1 Data Exploration

The first and essential step in solving a problem using machine learning is to explore the dataset. This helps to get familiarized with the data so that we know which features have high potentials, and also which features need to be modified and cleaned.

The dataset contains 101,766 observations and 50 different features. The feature varies from patient characteristics, medical records, medications, etc. There are two datatypes in the features: int (or float) and string (or object). Our initial impression is that the dataset is rich in information; however, it is “dirty” -- unstructured content with some missing data. The dataset has three possible outputs: 0 (zero readmission), <30 (readmission before 30 days), and >30 (readmission after 30 days). This indicates that we are dealing with a three-class problem.

A Python data analysis library called panda is very useful in the feature steps. A built-in function "describe()" generates detailed statistics of the features with a numeric data type, including min, mean, median, max, etc. We noticed that some of the features don’t have symmetric distribution, and some of them even have outliers. Thus, the next best step is to visualize each feature in a histogram using a Python plotting library, matplotlib, so that the data distribution can be clearly seen. We discover that some features are positive skew (left picture) and some are negative skew. Furthermore, there is also a feature with a bimodal distribution (middle picture), and a couple features with many values (right picture).

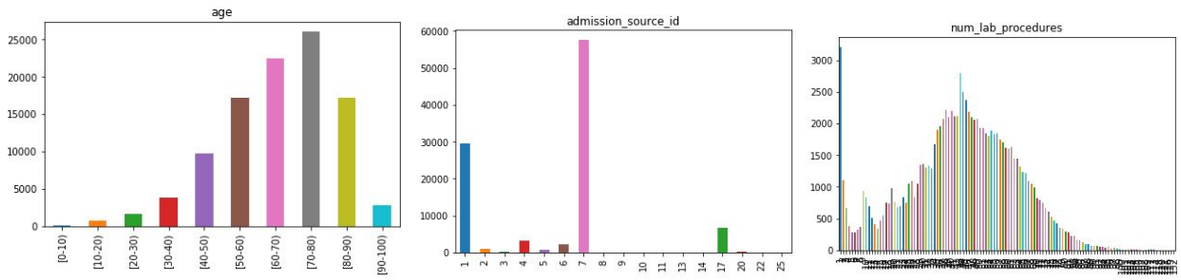



**Figure 1. Features data represented in histograms from the original dataset**

## 2.2 Data / Feature Cleanup

We open the dataset file (.csv) manually and notice there is a couple of features with “bad" data. In other words, those features contain either a lot of missing data points or a single value only. With the help of "count()" function, we retrieve eight features that have missing data. Three out of eight have more than 50% missing data. Thus, a "df.drop()” function from Panda is used to drop those three features (“weight”, “payer code”, and “medical_speciality”). Moreover, there are two features with only a single value "No". In this case, we also drop them (“examide” and “citoglipton”).

## 2.3 Feature Engineering

The goal of this step is to improve the quality of the dataset by either transforming some of the features data or creating new ones. We believe with a better-represented data, both the performance

of the classifiers and the prediction accuracy will improve. The engineering process consists of two phases: feature transformation and feature creation. At the end of each phase, we record the prediction accuracy by the classifiers and print out the feature importance.

### 2.3.1 Phase 1: Feature Transformation and Selection

The first step is to convert all the features with a string datatype into an int (numeric) datatype. The purpose of doing this is to make every feature acceptable as classifier's inputs. The encoding process is made possible with the help of a function "df.replace" from Panda. An example from this process is transforming “A1Cresult” (a feature) data from “>7” and “>8” (above normal level) into 1, “Norm” (normal level) into 0, and “None” into -1. The features that we converted into numeric datatypes are: "race”, “gender”, “change”, “diabetesMed”, “A1Cresult”, “max_glu_serum” and the 23 drugs name.There is one feature called “age” which originally comes in the form of numeric range from the dataset. In this case, we simplify the data by taking the midrange value ((Max-Min) / 2). For example, if the value of “age” of a patient is [20–30), we assume that his/ her age is ((30-20) / 2) = 25 years old. The process is done using a single function call “dataframe.map()” from panda. Furthermore, we also decide to merge some features because of their relevance. For instance, we think that “num_inpatient”, “num_outpatient”, and “num_emergency” are all the services that a hospital provides for a patient. Thus, adding those features together under a single feature called “num_services” will have the same meaning.

The last step is to collapse some of the features data, that have wide variations, into several categories. The features that we recategorized are: “admission_type_id”, “discharge_disposition_id”, “admission_source_id”, “diag_1”, “diag_2”, and “diag_3”. We use common sense to categorize the data in both "admission" and "discharge". For example, we remapped “Urgent (2)” and “Trauma Center (7)” into “Emergency (1)” because all of them have the same level of urgency. However, to categorize the three diagnosis features (“diag_x”), we base our mapping from a real diagnosis code (ICD-9-CM) that was found when doing research [4]. According to ICD-9-CM documentation, each group has its own range of number. For instance, a diagnosis code number between 290 and 319 means mental disorder, and it has the group number five (5).

After transforming the features, we apply Z-score normalization to all the numerical features (except the 23 drugs name). The reason is to bring any data into a comparable range (or scale). The implementation in Python is straightforward, we use a Python library called "numpy" that has built-in "mean()" and "std()" functions. Once we tested the features in the dataset into the classifiers, we generate the top 10 most important features according to each classifier to better understand which features have more weight than the rest. This was done using a function called “feature_importances_” from a scikit-learn library. The results are shown below:

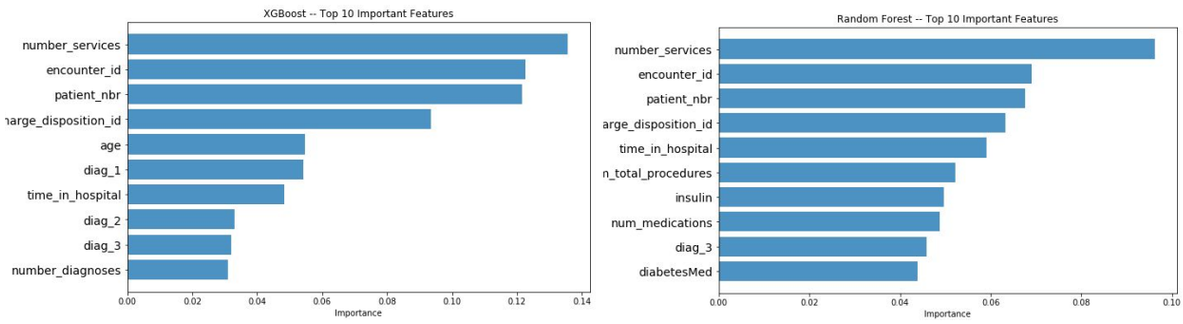



**Figure 2. Features Importance from Phase 1 (XGBoost and Random Forest)**

### 2.3.2 Phase 2: Feature Creation and Selection

The purpose of the second phase is to tune the dataset further to obtain more accurate prediction result. We start by investigating the “Feature Importance” graphs generated during the first phase. The result shows that the 23 drugs name, “race”, “gender”, “A1CResult”, “max_glu_serum” are not important compared to other features. Moreover, we also see that “encounter_id” and “patient_nbr” are the top 2 contributors, even though in real life, these two features are weak predictors of hospital readmission because they don't provide any information about a patient.

The next step is to create a new feature called "nummed", which counts the number of medications used (a “Steady”, “Up”, and “Down” data worths 1 and a “No” data worths 0). Some other new features that we created by analyzing which two features seemed highly correlated are: (1)

“Add_feature_1” = “num_medications” * “time_in_hospital”, (2) "Add_feature_2” = “change” * “num_medications”, (3) "Add_feature_3” = “age” * “number_diagnoses”. After that, we drop “race”, “gender”, “A1CResult”, “max_glu_serum”, “encounter_id”, “patient_nbr”, and the 23 drugs name. Furthermore, we also decided to unmerge “num_services” feature back into “num_inpatient”, “num_outpatient”, and “num_emergency” and “num_total_procedures” into “num_procedures” and “num_lab_procedures”. The intention behind this is to investigate which individual features among the merged are the actual standouts. The new “feature importance” results shown below:

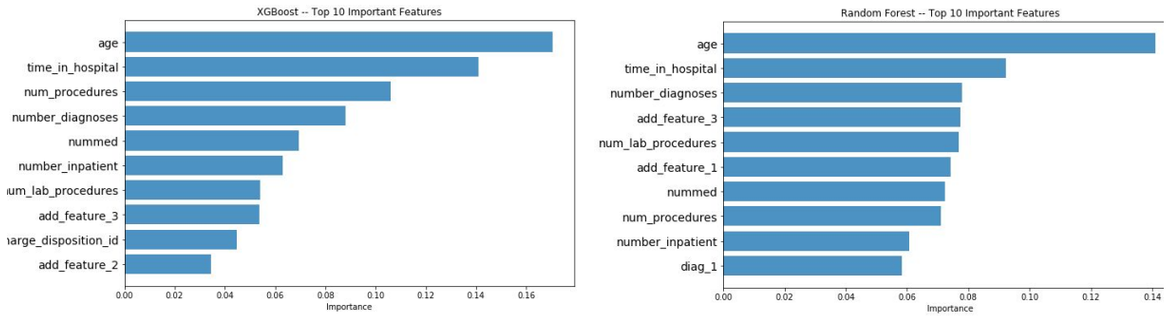



**Figure 3. Features Importance from Phase 2 (XGBoost and Random Forest)**

## 3. Model Selection

We start the experiment by trying the popular classifiers from the Scikit-Learn library that are also not covered in lecture such as XGBoost, etc. Then, we compare each classifier's performance (in its default parameters) and choose two with the best runtime and accuracy. The comparison test is done by invoking the classifier 10 times and then averaging the results (runtime + accuracy). Using phase 2 dataset as the input, we found that: (1) XGBoost has an average runtime of 11.37 seconds and an average accuracy of 61.57%, (2) Random Forest has an average runtime of 1.79 seconds and an average accuracy of 67.08%, (3) MLP (Neural Networks) has an average runtime of 26.85 seconds and an average accuracy of 51.82%, and (4) SVM has an average runtime of 683.69 seconds and an average accuracy of 56.52%. Based on this result, we decided to use XGBoost and Random Forest as our classifier models.

### 3.1 XGBoost

XGBoost is an open-source software library which provides a gradient boosting framework for C++, Java, Python, R, and Julia. It aims to provide a "Scalable, Portable and Distributed Gradient Boosting (GBM, GBRT, GBDT) Library" and it also supports the distributed processing frameworks like Apache Hadoop, Apache Spark, and Apache Flink.

We used the model xgb.XGBClassifier() provided by sklearn. After careful investigation, we finally decide to focus on tuning parameters including “colsample_bytree”, “gamma”, “learning_rate”, “maxdepth”, “n_estimators”, and “subsample” which could influence the model’s performance more. We chose appropriate range of values for each parameter in advance and then trained the data using different combinations of values of parameters. In fact, we tuned the values of two parameters using a nested for loop in each tuning step and the combination of values of parameters with the best performance will then be used in next tuning step.

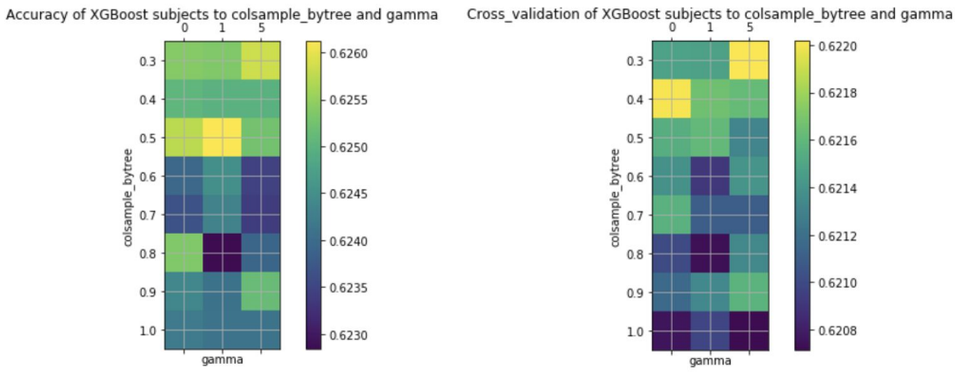
### 3.2 Random Forest

Random forests or random decision forests are an ensemble learning method for classification and regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. In this work, we explore the impact of parameters: 1) **criterion function ‘entropy’ and ‘gini’**; 2) **n_estimators**; 3) **max_depth;** 4) **min_samples_leaf** and 5) **min_samples_split** on the prediction accuracy of random forest classifier. To verify the effects of these parameters, we choose the appropriate range to tune these parameters and calculate the accuracy score for training data and validation data, and the cross_validation score to evaluate the prediction results.

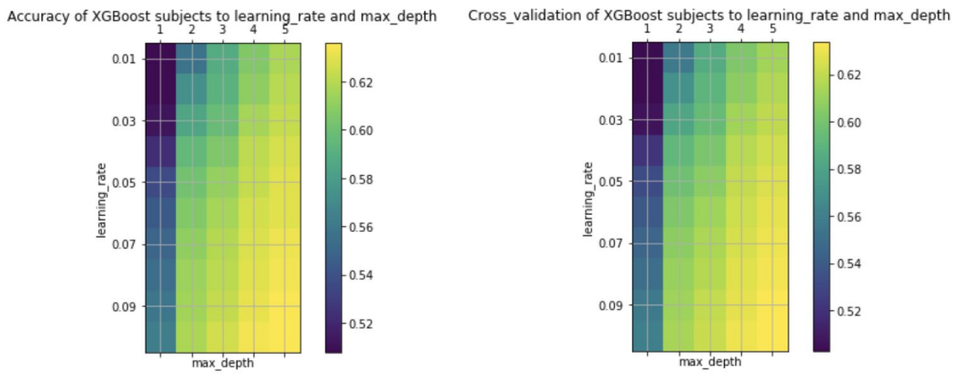
# 4. Experiments and Results

## 4.1 Phase 1 : XGBoost for original features

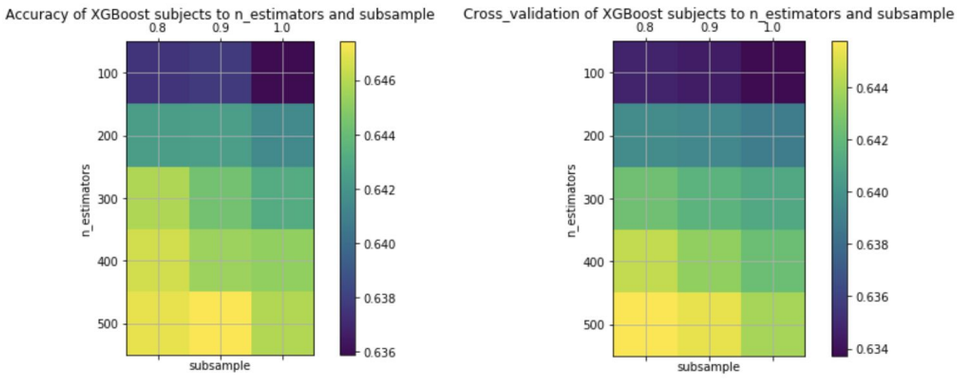
In each tuning step, we use accuracy function to find the most appropriate values of parameters for our data. In addition to accuracy, we also use cross-validation function to verify each other's results.




According the above plot, we selected colsample_bytree = 0.5 and gamma = 1 and used these two values in the next tuning process.




According the above plot, we selected learning_rate = 0.1 and max_depth = 5 and used these two values in the next tuning process.




According the above plot, we selected n_estimators = 500 and subsample = 0.9. Finally, we selected all values for our XGBoost classifier parameters and got the final results as below.

**Accuracy** score: 0.6474243888965613 (64.74%)

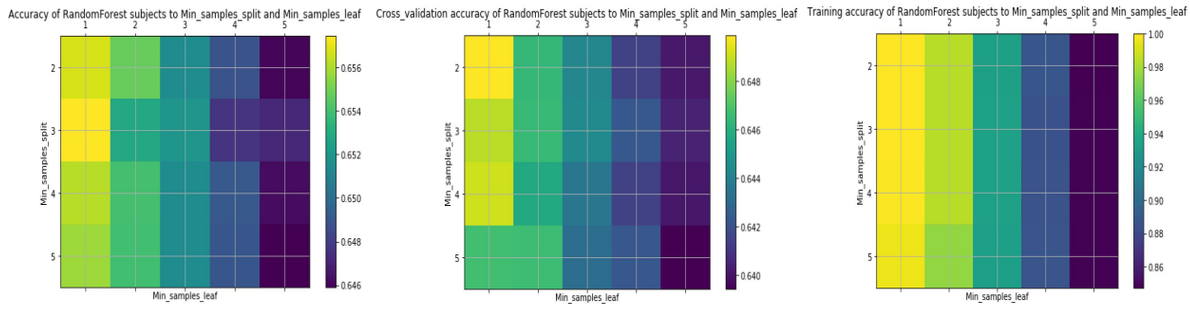
**Cross Validation** score: 0.6453670815081887 (64.54%)

## 4.2 Phase 1 : Random Forest for original features

First we explore the impact of criterion function on the accuracy score got by random forest classifier. Here we fixed the parameters: n_estimators= 240; min_sample_leaf=2; max_depth=80 ;min_sample_split=2. The accuracy score we got by setting parameter **'criterion=entropy' is 0.649**, and the accuracy score with **'criterion=gini' is 0.651.** By comparison the accuracy score, we can conclude that using criterion function 'gini' is a better choice for our prediction work.

Then we explored the parameters' min_samples_leaf' and ' min_samples_split' in randomforest function with fixed parameters: **n_estimators = 240, max_depth = 80, criterion = 'gini'**. Here we set the testing range for 'min_samples_leaf' and 'min_samples_split' are [1,5] and [2,5]. And we compare the training accuracy and the validation accuracy in figure 10. From the graph, we can

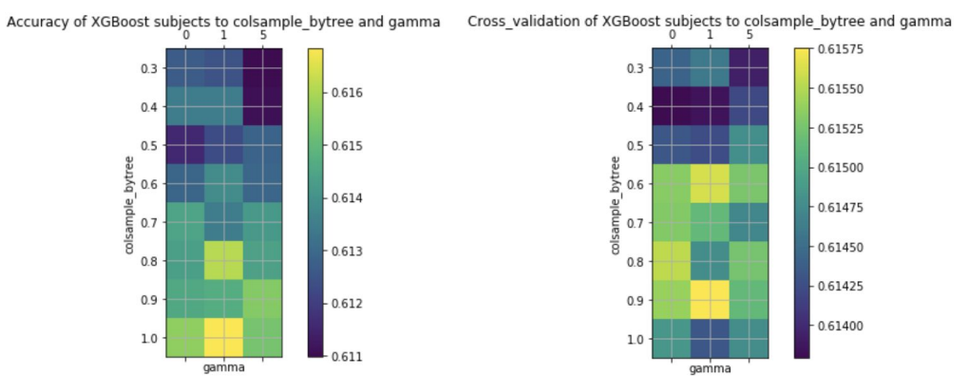
see that when the **min_samples_leaf = 1 and min_samples_split = 2,** both the validation accuracy and the training accuracy get to the peak value, which is **validation accuracy score = 0.6567**, **cross_valdition accuracy = 0.6498.** and **training accuracy score = 1** respectively.



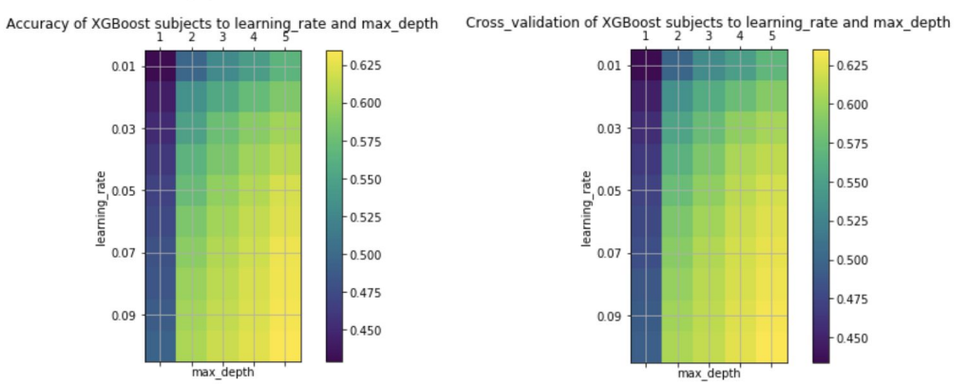

Next we explore the impact of parameters ‘ n_estimators’ and ‘ max_depth’ on the predication accuracy of random forest classifier, with fixed **min_samples_leaf = 1, min_samples_split = 2, and criterion = ’gini’.** From the result we can see that the **maximal accuracy score = 0.6592** with **N_estimator = 250, Max_depth = 60.**The cross_validation is slightly lower compared what we got from accuracy function, where the **maximum cross_validation outcome = 0.65** with **N_estimator = 300** and **Max_depth = 60.** From the figures above, we also can infer that the accuracy of data prediction increases as the increase the number of estimators. Here, we figure out when we set **min_samples_leaf = 1, min_samples_split = 2,** the **training accuracy = 1**, which means all training data are predicted correctly. Thus the best accuracy score for original feature set is **Accuracy score: 0.6592 (65.92%); Cross Validation score: 0.65 (65%).**

### 4.3 Phase 2 : XGBoost for improved features

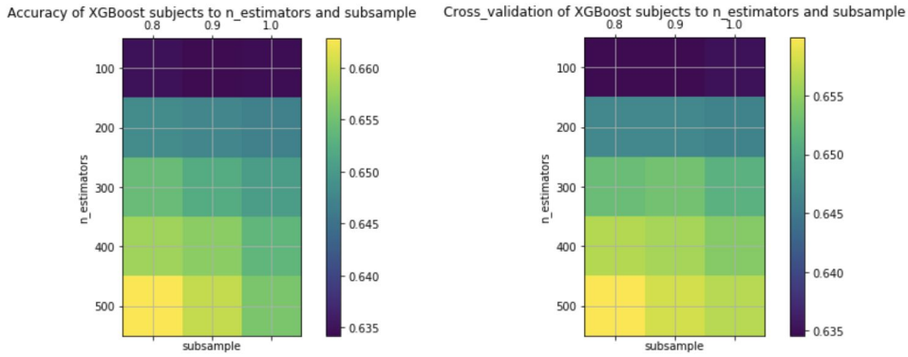
In each tuning step, we use accuracy function to find the most appropriate values of parameters for our data. In addition to accuracy, we also use cross-validation function to verify each other’s results.




According the above plot, we selected colsample_bytree = 1.0 and gamma = 1 and used these two values in the next tuning process.




According the above plot, we selected learning_rate = 0.1 and max_depth = 5 and used these two values in the next tuning process.




According the above plot, we selected n_estimators = 500 and subsample = 0.8. Finally, we selected all values for our XGBoost classifier parameters and got the final results as below.
**Accuracy** score: 0.6555127694418625 (65.55%)
**Cross Validation** score: 0.6544942139816379 (65.45%)

### 4.4 Phase 2 : Random Forest for improved features

First we explore the impact of criterion function on the accuracy score got by random forest classifier. Here we fixed the parameters: **n_estimators = 140; min_sample_leaf = 2; max_depth = 80; min_sample_split = 10**. The accuracy score we got by setting parameter **'criterion = entropy' is 0.699,** and the accuracy score with **'criterion = gini' is 0.702.** By comparison the accuracy score, we can conclude that using criterion function 'gini' is a better choice for our prediction work.

Then we explored the parameters 'min_samples_leaf' and 'min_samples_split' in randomforest function with fixed parameters: **n_estimators = 240, max_depth = 80, criterion = 'gini'**. Here we set the testing range for 'min_samples_leaf' and 'min_samples_split' are [1,5] and [2,5]. And we compare the training accuracy and the validation accuracy in figure below. From the graph, we can see that when the **min_samples_leaf = 1 and min_samples_split = 2,** both the validation accuracy and the training accuracy get to the peak value, which is **validation accuracy score = 0.72164**, and **training accuracy score = 1** respectively.

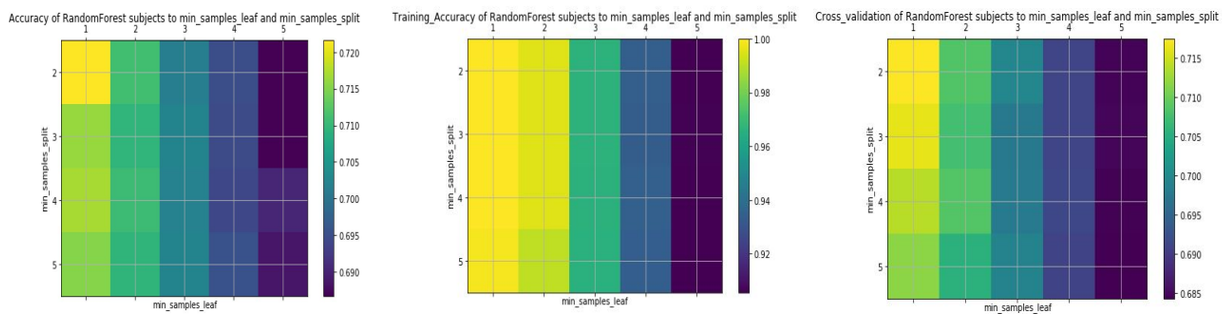
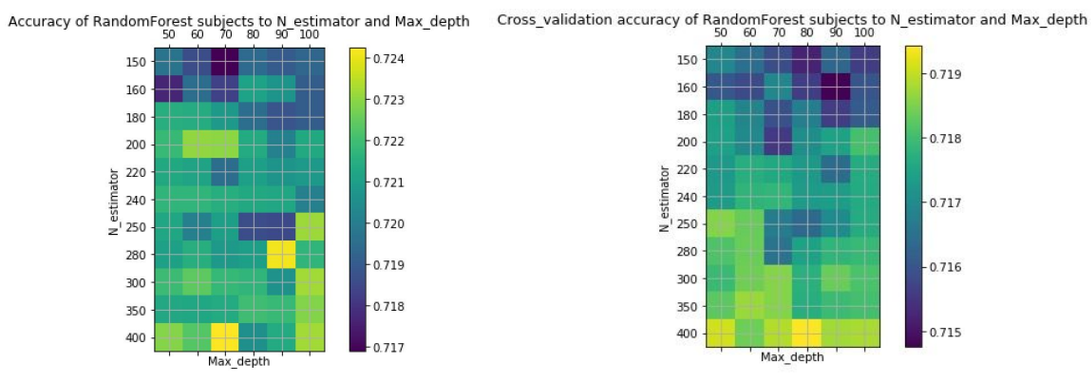



Figure below shows the cross validation result of the prediction of random forest. Comparing it with the accuracy score we got using accuracy function, we can see the the tendency of the cross_validation score is similar with that of validation accuracy, and the peak value is also at the condition when **min_samples_leaf= 1 and min_samples_split=2,** with **cross_valdition accuracy= 0.7175.** Next we explore the impact of parameters ' n_estimators' and ' max_depth' on the predication accuracy of random forest classifier, with fixed **min_samples_leaf =1, min_samples_split=2, and criterion='gini'.**



From figures above, we can see that the **maximal accuracy score = 0.7248** with **N_estimator =400, Max_depth=70.**The cross_validation is slightly lower compared what we got from accuracy function, where the **maximum cross_validation outcome=0.7198** with **N_estimator =400** and **Max_depth=80.** From the figures above, we also can infer that the accuracy of data prediction increases as the increase the number of estimators. Here, I figure out when we set **min_samples_leaf =1, min_samples_split=2,** the **training accuracy =1**, which means all training data are predicted correctly. Thus the best accuracy score for original feature set is
**Accuracy score: 0.7248 (72.48%) ; Cross Validation score: 0.7198 (71.98%)**

## 5. Interesting Finding

In this study, we are working on predicting the possibility for patients readmit after the current treatment, and there are three kind of outputs: '>30','<30' and 'No'. Instead of predicting the readmission state in three possible outputs, we tried to minimize the three-class prediction outputs into two through combining the '>30' and '<30' into one state that is 'readmission' and leave the other state 'No'. By doing this, using random forest classifier, we got a really good prediction accuracy score that **maximal accuracy score=0.93**. While for three prediction outputs for three-class results we got the **maximal accuracy score= 0.73** by using random forest. The better prediction for two-class outputs implies that when we decrease the classifications for prediction outputs by combining certain possible results, we can get a higher prediction accuracy. We think the reason for that is combining prediction results would widen the coverage range or possibility for certain prediction output, soften the prediction boundary, and thus increase the successful prediction rate. Whereas, the drawback for this method is that would sacrifice the prediction precision.

## 6. Conclusion

Through the tuning process, we successfully improved the performance of XGBoost classifier from about 61-62% to 64-65%. For XGBoost classifier, we got the best accuracy score of 65.55% and 64.74% for improved features and original features, respectively. As for Random Forest classifier, we got even better results! Through similar tuning process as XGBoost classifier, we successfully improved the performance of Random Forest classifier from about 64-65% to 72-73%. As a result, for Random Forest classifier, we finally got the best accuracy score of 65% for original features and **72.48%** for improved features in the Diabetes 130-US hospitals dataset.

## 7. Distribution of Work

***Fangqi Liu*** contributed to entirely tuning the parameters of Random Forest classifier on both original (Phase 1) and improved (Phase 2) features. She used both Train/ Test Split and K-Fold Cross Validation (K=10) methods to test the accuracy of the classifier. ***Te-Chih (Annie) Chen*** focused on tuning the parameters of XGBoost classifier on both original (Phase 1) and improved (Phase 2) features. Similar to what Fangqi did, Annie used both evaluation methods to test the accuracy of the classifier. ***Jonathan Harijanto*** worked on the features portion, both the original (Phase 1) and improved (Phase 2) version. He focused on exploring the dataset, cleaning up the features, engineering the features including transformation, creation, and selection. He also did experimentation and researched to discover which two classifiers have the best performance (runtime) and yield the best accuracy.

## 8. References

[1] M. Bhuvan, A. Kumar, A. Zafar, and V. Kishore, "Identifying Diabetic Patients with High Risk of Readmission," pp. 1–10, Feb. 2016. Available at: https://arxiv.org/pdf/1602.04257.pdf
[2] Kripalani, S., Theobald, C., Anctil, B. and Vasilevskis, E. (2014). Reducing Hospital Readmission Rates: Current Strategies and Future Directions. Annual Review of Medicine, [online] 65(1), pp.471-485. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4104507/.
[3] Medium. (2018). How to use machine learning to predict hospital readmissions? (Part 1 | Part 2). [online] Available at: https://medium.com/berkeleyischool/how-to-use-machine-learning-to-predict-hospital-readmissions-part-1-bd137cbdba07.
[4] Icd.codes. (2018). ICD-9-CM Chapters List. [online] Available at: https://icd.codes/icd9cm.