Predicting 30-day ICU readmissions from the MIMIC-III database

Capstone Project

Machine Learning Engineer Nanodegree

Yaron Blinder
March 14, 2017

I. Definition

Project Overview

This project describes the development of a model for predicting whether a patient discharged from an intensive care unit (ICU) is likely to be readmitted within 30 days. To do this, we will use the MIMIC-III database which contains details records from ~60,000 ICU admissions for ~40,000 patients over a period of 10 years. Using these records, we will find patients in the database that were readmitted within 30 days of discharge, and use their records to train a generalized classifier to predict readmission. The project also includes parameter optimization and in depth performance analysis for the classifier.

Problem Statement

Implement a machine learning model to predict the probability of intensive care readmission within 30 days of discharge, based on the patient's release records upon discharge. To achieve this, we will use the MIMIC-III database, which is a collection of health data associated with >40,000 critical care patients collected over 10 years[1].

The solution outline will be as follows:

1. Preliminary research: Consult the relevant literature to assemble a list of relevant features to use.

2. Data wrangling: Use SQL to obtain the relevant subset of data from the MIMIC-III database.
3. Exploratory data analysis: Produce relevant statistics and visualizations to characterize the data
4. Data preprocessing: Prepare the dataset for application of a machine learning model – clean missing or invalid values, calculate the time between subsequent admissions to produce label dataframe.
5. ML model implementation and refinement: Define model metrics, implement relevant code and perform hyperparameter optimization. Once optimized values are found, estimate model performance with cross-validation.

After extracting our features of interest (see table 1) for all admission records or interest (i.e. admissions to the MICU[1]), and cleaning up the extracted dataset, we will create the label for each admission which will indicate whether that discharge resulted in readmission within 30 days. We will then feed the clean dataframe and calculated labels into a gradient-boosted decision tree classifier, perform hyperparameter optimization and analyze the model's predictive capability.

Metrics

Our problem can be described as a binary classification problem on an unbalanced dataset (only ~12% of the observations fall into the "positive" label category). For this reason, we cannot rely on a simplistic metric such as prediction accuracy, and must look at more robust metrics.

A standard approach in such cases is to look at the precision, sensitivity, specificity, F1 score and the Receiver Operating Characteristic curve, all defined in the following:

- **Precision** is defined as:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

- **Recall**, also known as **Sensitivity** or **true positive rate (TPR)**, is defined as:

[1] The MIMIC-III database contains records from 5 different intensive care units: MICU –is Medical Intensive Care Unit; CCU is Coronary Care Unit; CSRU is Cardiac Surgery Recovery Unit; SICU is Surgical Intensive Care Unit; TSICU is Trauma Surgical Intensive Care Unit.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

- **Specificity** is defined as:

$$Specificity = \frac{True\ Negative}{True\ Negative + False\ Positive}$$

- **F1 score**, the harmonic mean of precision and recall:

$$F1 = \frac{2 * (True\ Positive)}{(True\ Positive + False\ Negative) + (True\ Positive + False\ Positive)}$$

- **ROC curve**: A plot of the True Positive Rate (or sensitivity) as a function of the False Positive Rate (FPR, equal to 1-specificity), which implicitly sets the discrimination threshold. Specifically, the area under the ROC curve (commonly referred to as the AUC) is a good nominal metric for the classifier performance, where an AUC of 0.5 represent a classification that is no better than chance, and the closer the AUC get to 1, the better the classifier.

II. Analysis

Data Exploration

The MIMIC-III (**M**edical **I**nformation **M**art for **I**ntensive **C**are III) database contains deidentified health-related data from various intensive care units at the Beth Israel Deaconess Medical Center, covering over forty thousand patients between 2001 and 2012. The database consists of large inter-related data tables (totaling just under 38GB in 26 csv files), separated for clarity into categories such as admissions, patients, chart events, lab events etc[2]. These relational tables can be accessed and combined via SQL queries to produce many different subsets of interest. The database contains "static" data (such as

[2] The full description of the database is too long to include in this report, and can be found at https://mimic.physionet.org/

patient id, demographics, admission id, admission and discharge timestamps) and "dynamic" data (such as measurements of vital signs over time, lab results etc.).

A study of recent literature[2,3,4] was conducted to arrive at a list of 43 features considered useful or potentially predictive in a readmission model. These are:

Admission specific	Subject id
	Admission id
	Admit time
	Discharge time
	Death time (if died during stay)
	First care unit
	Last care unit
Demographics	Age
	Gender
	Marital status
	Insurance provider
Lab results	Urea (min, max, mean)
	Platelets (min, max, mean)
	Magnesium (max)
	Albumin (min)
	Calcium (min)
Chart events	Respiratory rate (min, max, mean)
	Glucose (min, max, mean)
	Heart rate (min, max, mean)
	Systolic blood pressure (min, max, mean)
	Diastolic blood pressure (min, max, mean)
	Body temperature (min, max, mean)
	Urine output (min, max, mean)
Aggregated metrics	SAPS-II
	SOFA

Table 1: Selected data features

The features detailed in table 1 are a combination of numerical and categorical features and will require some preprocessing before being fed into our classifier. Admission specific and demographic information required merely a straightforward query. Lab

results appear as "dynamic data", and so rudimentary aggregation (minimum, maximum and mean values calculation). Chart events were somewhat more complicated – due to multiple data collection methods (e.g. blood/urine glucose, various blood pressure measuring devices) as well as the MIMIC-III database containing data from two different critical care information systems (Philips CareVue and iMDSoft Metavision), each type of event had multiple (between 2 and 7) identifying item ids. In the case of body temperature, some measurements were in Celsius and some in Fahrenheit. Unification and aggregation of all these measurements were performed on the SQL query level as well.

Based on recommendations from the MIMIC team, the following value pre-filtering was implemented during extraction of the relevant data table:

Respiratory rate	0 < value < 70
Glucose	0 < value
Heart rate	0 < value < 300
Systolic blood pressure	0 < value < 400
Diastolic blood pressure	0 < value < 300
Body temperature[3]	70 < value (°F) < 120 10 < value (°C) < 50

Table 2: Value limits implemented at the SQL level.

Finally, the aggregated metrics (SAPS-II[4], SOFA[5]) are pre-engineered features, derived by the MIMIC team from other features in the database and were readily available via SQL.

Statistical overview of the raw data (after SQL preprocessing):

The extracted dataset contains 58,072 records from 34,000 unique patients. The categorical features' value distributions are visualized in figure 1, and summary statistics for the numerical features are shown in table 3.

[3] Temperatures in Fahernheit were first filtered as indicated, then converted to Celsius.

[4] SAPS-II - **Simplified Acute Physiology Score.** See Le Gall et al. "A New Simplified Acute Physiology Score (SAPS II) Based on a European/North American Multicenter Study. JAMA. 1993;270:2957-2963"

[5] SOFA score: **Sepsis-related Organ Failure Assessment score**, also known as **Sequential Organ Failure Assessment score.** See Vincent JK et al., "The SOFA (Sepsis-related Organ Failure Assessment) score to describe organ dysfunction/failure. Intensive Care Med 1996 Jul;22(7):707-10."

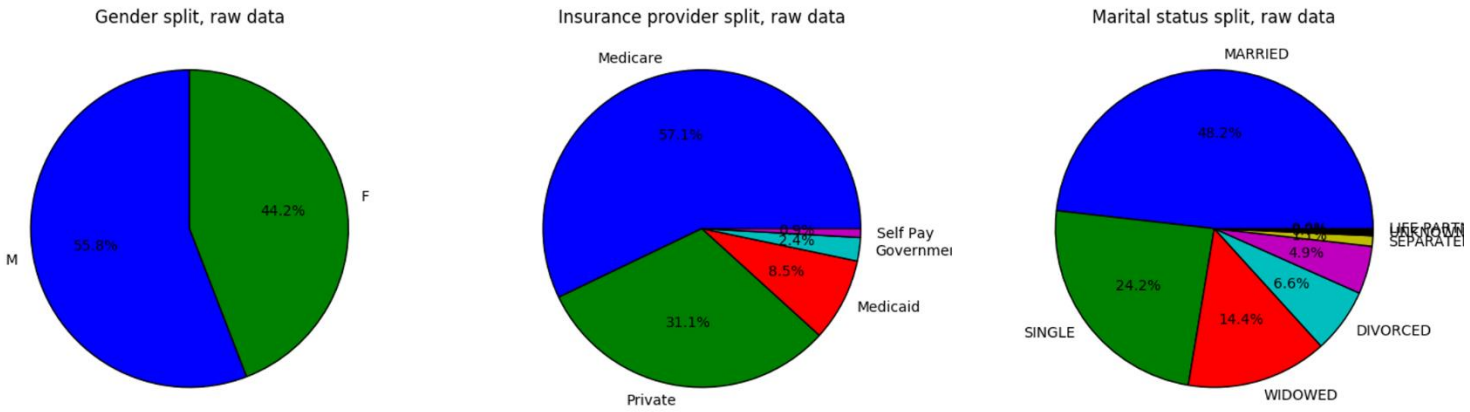

Figure 1: Value distributions for the categorical features, raw dataset.

Age (raw[6])	75 ± 54		
Age (under 300[7])	64 ± 16		
SAPS-II	37 ± 14		
SOFA	4.5 ± 3.2		
	min	**mean**	**max**
Urea (min, max, mean)	16 ± 13	27 ± 19	42 ± 31
Platelets (min, max, mean)	159 ± 94	239 ± 118	354 ± 197
Magnesium (max)	-	-	2.5 ± 1
Albumin (min)	2.8 ± 0.8	-	-
Calcium (min)	7.7 ± 0.9	-	-
Respiratory rate (min, max, mean)	9.8 ± 3.7	19.4 ± 3.5	33 ± 9
Glucose (min, max, mean)	87 ± 30	138 ± 87	271 ± 5968
Heart rate (min, max, mean)	63 ± 15	86 ± 13	118 ± 25
Systolic blood pressure (min, max, mean)	80 ± 21	120 ± 16	166 ± 29
Diastolic blood pressure (min, max, mean)	36 ± 12	60 ± 10	100 ± 28
Body temperature (min, max, mean)	35.6 ± 0.9	36.8 ± 0.5	37.9 ± 0.9
Urine output (min, max, mean)	26 ± 62	126 ± 537	669 ± 19,345

Table 3: Summary statistics (mean ± standard deviation) for the numerical features, raw dataset.

[6] The split between "raw" and "under 300" is due to an irregularity with the age feature, described in further detail under the "Exploratory Visualization" section.
[7] See footnote 6 above.

Outliers and abnormal values:

As can be seen in table 3, most numerical features show a reasonable distribution around the mean, with the few exceptions being *max Glucose, mean Urine output,* and *max Urine output*. This will be addressed in the preprocessing stage.

Exploratory Visualization

(A)

(B)

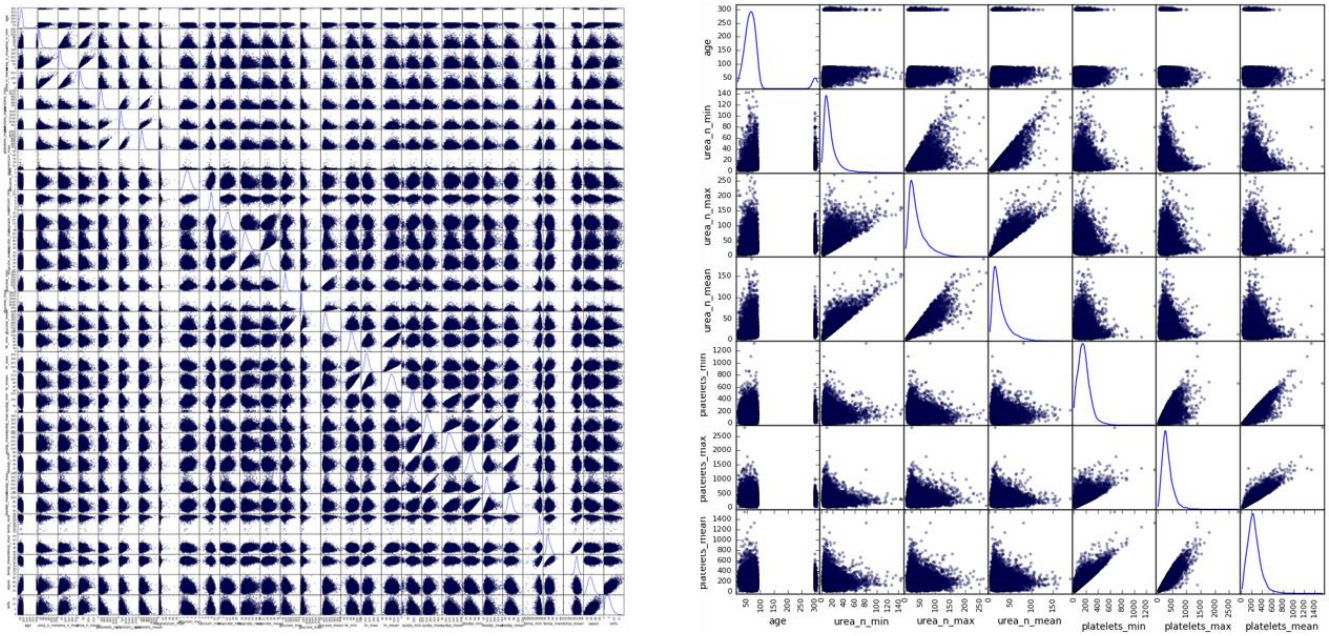

Figure 2 – Raw numerical data scatter matrix. A: Scatter matrix of all numerical features. B: zoomed-in view on the first 6 features. Each square shows the scatter plot corresponding to the two features defined by the row and column. The diagonal shows the kernel density estimation plots.

Figure 2 shows a scatter matrix of all numerical features in the data prior to preprocessing. While the large number of features represented (30) makes a clear representation difficult, a few important attributes can nevertheless be observed. For example - looking at the scatter matrix main diagonal, the kernel density estimation plots reveal many non-symmetrical/long-tailed distributions. Additionally, several of the features appear to have a strong positive correlation. A closer look reveals these strong correlations are between closely-related features such as the min, max, and mean values of a single parameter (e.g. urea_n_min, urea_n_max, and urea_n_mean) and are to be expected. However, these correlations are not absolute, and are not enough to warrant elimination of any of these features.

Another interesting attribute which can be noticed in the scatter matrix is the odd distribution of age, which seems to show two separate "normal" distributions – one centered around ~70 years and a second one centered around ~300 years. This is a result of the de-identification process performed by the MIMIC team, in which patients older than 89 years old were deemed of greater risk of identification, and were therefor automatically given an age of 300 upon their first admission. As old-age is likely to play a factor in an ICU readmission scenario, it is especially important to retain the observations for older patients and not discard them as outliers.

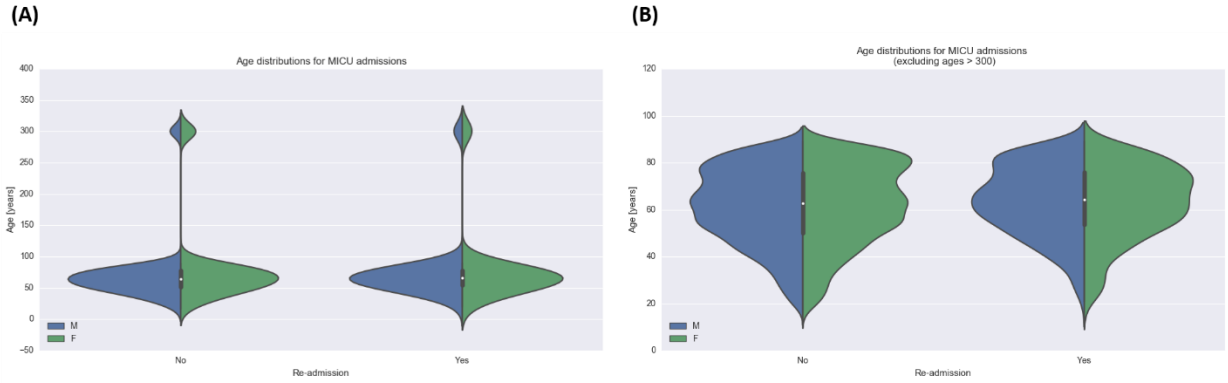

Figure 3: Age distribution, split by gender and readmission label. (A) shows the entire range, (B) excludes values >300.

Algorithms and Techniques

For our binary classification problem, we will use a gradient-boosted decision tree ensemble as our classifier. Decision tree ensembles are basically combinations of decision trees. "Gradient boosting" refers to a method by which the optimized tree parameters are learned. Generally speaking, the method begins with a tree of depth 0, and then iteratively tries adding splits using the various features inherent in the data, and selects the best split. During this training process, the tree is also regularized to control for complexity. Specifically, we will use the XGBoost implementation, which is considered highly efficient and is a top-performing classifier in many Kaggle competitions. This classifier has many hyperparameters, and we will manually set the minimal amount required for a preliminary solution, and perform an optimization procedure using a grid search with k-fold cross validation to determine the rest. Finally, we will use k-fold cross validation to analyze the optimized model performance. Folds will be stratified to retain the class imbalance.

Default values: The classifier hyperparameters will initially be set using the default setting, except for the following (which remained constant throughout the optimization process):

- ***objective***: Defines the type of classification. Set to binary:logistic
- ***nthread***: Defines the amount of parallel threads. Set to 4.
- ***scale_pos_weight***: Defines label weighting for unbalanced data. Set to the ratio of negative labels to positive labels (7.091)
- ***seed***: Defines a setting for the random number generator. Needed to ensure repeatable results. Set to 1.

Benchmark

As discussed above, this is a binary classification problem with an unbalanced dataset, where the minority class represents only ~12% of the dataset. A simple and useful benchmark therefore is one that always predicts the majority class, i.e. no readmission. Such a model is guaranteed both high precision (~88%), perfect recall, and high F1-score on the majority class, while obtaining zero on all metrics for the minority class.

III. Methodology

Data Preprocessing

The data preparation stage was comprised of the following step:

1. "Data wrangling" – obtaining the relevant data from the database into a dataframe format.
2. Ascertain which records can be flagged as "readmission" based on admission and discharge information. This step reveals duplicate records and so was performed before the following clean-up step.
3. Clean up missing/invalid values.
4. Define label threshold value and create label column.
5. Prepare the remaining data for classification:
    a. Scale the numerical features and remove outliers.
    b. Encode categorical features.

The first "data wrangling" step was to formulate the SQL query to produce a dataset with all the relevant features in a way that could be easily interpreted by pandas (SQL code provided in the supporting material). The data was sorted by patient id and by admission time to facilitate the extraction of readmission labels.

Once the relevant dataset had been properly extracted from the MIMIC-III database, the following features were calculated using the admissions and discharge time features, where a patient had more than one admission:

- **readmit_dt** – Time delta between admission and the previous discharge.
- **next_readmit_dt** – Time delta between discharge and the next admission.
- **readmit_last_careunit** – Care unit from which the patient was discharged.

The next step was to remove invalid values:

- Records where **readmit_dt** was negative were determined to be a result of duplicate record-keeping and were remove.
- Records where the patient died during their stay were considered unhelpful for determining future readmission for obvious reasons and were removed. After that, the entire deathtime column was removed as it only contained empty values.
- Remaining records with empty values were removed.

The next step was to define the threshold for predicted readmission (I chose 30 days based on my preliminary research), and define the label column:

- **future_readmit** – Contains 'Yes' where a discharge resulted in a readmission before the threshold, and 'No' otherwise.

The final "data wrangling" step was to further narrow the dataset down by isolating MICU admissions only.

Preprocessing the clean dataset:

The **future_readmit** column was used to define the label dataframe. At this point, several features could be removed as they would not be used for the predictive model. These were **subject_id, hadm_id, admittime, dischtime, first_careunit, last_careunit, readmit_dt, readmit_last_careunit, future_readmit,** and **next_readmit_dt**.

The remaining features were split into numerical and categorical features. All numerical features were scaled uniformly (mean removed and standard deviation normalized to 1) and outliers were then identified and removed using Tukey's test, i.e. measures that were outside of the range [Q1-1.5*(Q3-Q1), Q3+1.5*(Q3-Q1)] were considered outliers. As I pointed out in the exploratory analysis stage, certain features in the original dataset had abnormal distributions. *Urine output* remained too noisy even after outlier exclusion and

so was left out. *Max Glucose* seemed to normalize better, and so was left in. The categorical features were transformed into one-hot encoded vectors. The categorical features' value distributions are visualized in figure 4, and summary statistics for the numerical features are shown in table 4.

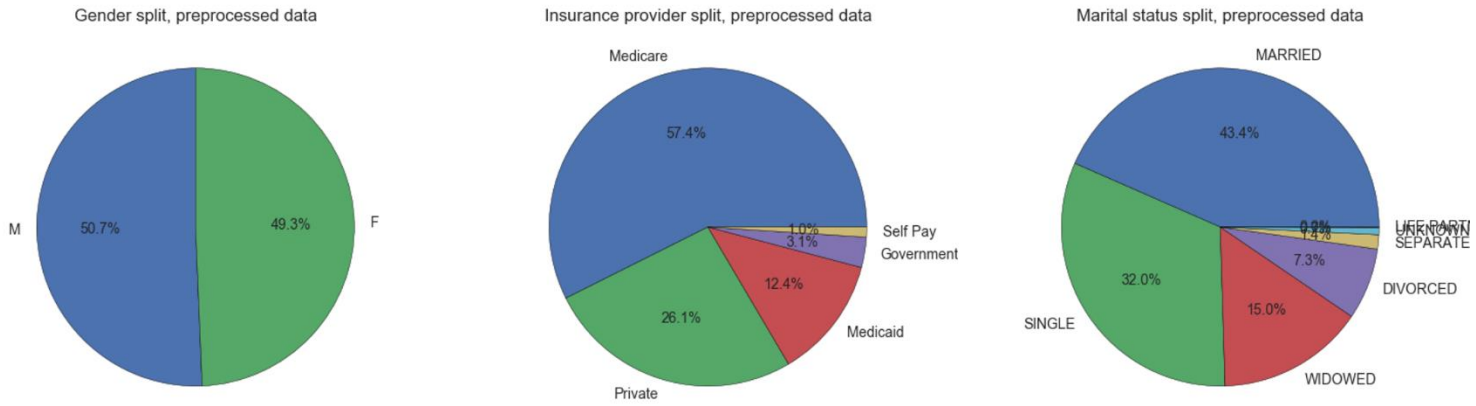

Figure 4: Value distributions for the categorical features, preprocessed dataset.

Features	Raw data			Preprocessed data		
Age (raw)	75 ± 54			76 ± 60		
Age (under 300*)	64 ± 16			62 ± 17		
SAPS-II	37 ± 14			36 ± 13		
SOFA	4.5 ± 3.2			4.5 ± 3		
	min	**mean**	**max**	**min**	**mean**	**max**
Urea (min, max, mean)	16 ± 13	27 ± 19	42 ± 31	16 ± 14	27 ± 20	41 ± 31
Platelets (min, max, mean)	159 ± 94	239 ± 118	354 ± 197	172.5 ± 103	242 ± 127	341 ± 188
Magnesium (max)	-	-	2.5 ± 1	-	-	2.4 ± 1
Albumin (min)	2.8 ± 0.8	-	-	2.9 ± 0.7	-	-
Calcium (min)	7.7 ± 0.9	-	-	7.5 ± 0.9	-	-
Respiratory rate (min, max, mean)	9.8 ± 3.7	19.4 ± 3.5	33 ± 9	11 ± 3.8	19.7 ± 3.7	31.8 ± 8.6
Glucose (min, max, mean)	87 ± 30	138 ± 87	271 ± 5968	91 ± 29	137 ± 40	221 ± 137
Heart rate (min, max, mean)	63 ± 15	86 ± 13	118 ± 25	67 ± 13.8	86.6 ± 14	115.2 ± 23.6
Systolic blood pressure (min, max, mean)	80 ± 21	120 ± 16	166 ± 29	86.4 ± 18.7	122.1 ± 16.1	162.5 ± 28.3
Diastolic blood pressure (min, max, mean)	36 ± 12	60 ± 10	100 ± 28	38.6 ± 12.7	62.6 ± 10.7	99.6 ± 26.7
Body temperature (min, max, mean)	35.6 ± 0.9	36.8 ± 0.5	37.9 ± 0.9	35.8 ± 0.7	36.8 ± 05	37.9 ± 0.9
Urine output (min, max, mean)	26 ± 62	126 ± 537	669 ± 19,345	-	-	-

Table 4: Summary statistics (mean ± standard deviation) for the numerical features, in the raw dataset and the preprocessed dataset.

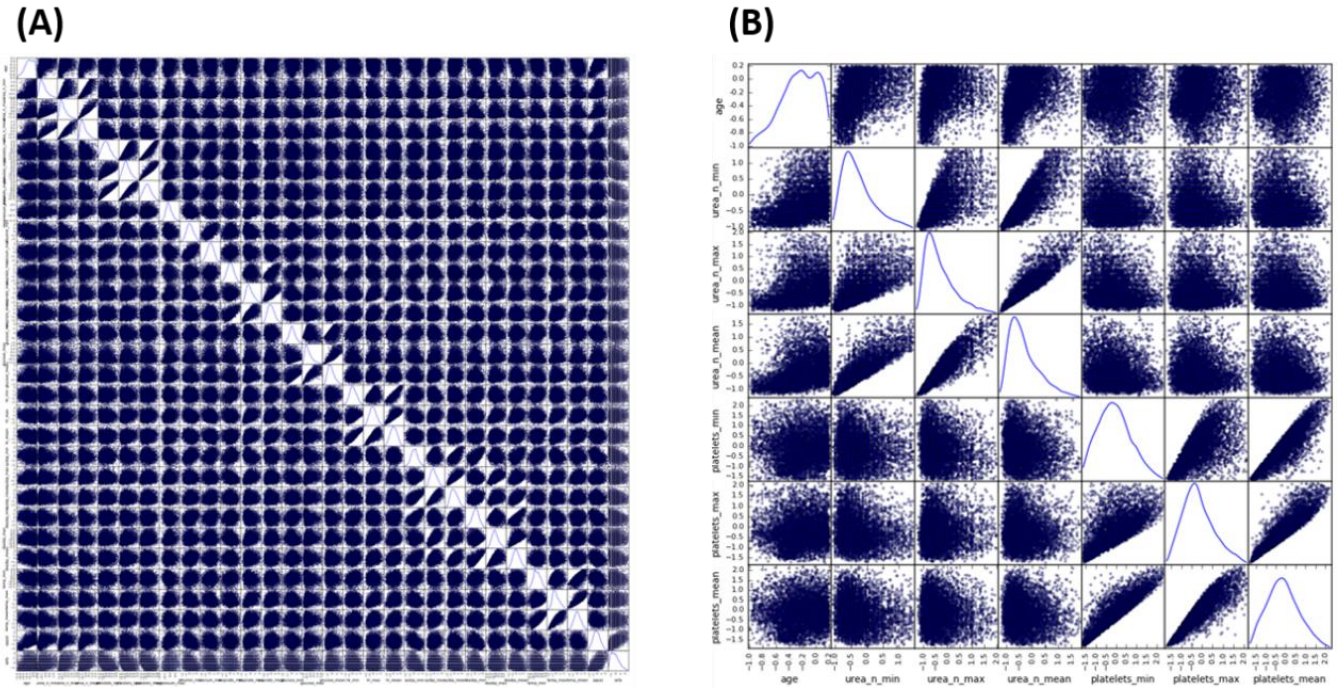

Figure 5 – Preprocessed numerical data scatter matrix. A: Scatter matrix of all numerical features. B: zoomed-in view on the first 6 features. Each square shows the scatter plot corresponding to the two features defined by the row and column. The diagonal shows the kernel density estimation plots.

Figure 5 shows the scatter matrix for the preprocessed numerical features. When compared with figure 2, several clear differences can be observed. First, the normalized scaling is clearly apparent as a more complete utilization of the feature value space (the non-diagonal subplots appear "fuller"). Second, the diagonal kernel density estimation plots are much more reminiscent of normal distributions, with higher symmetry and less pronounced "long tails". Roughly speaking, most features appear to be generally uncorrelated, apart from the few inherently correlated features discussed in the exploratory visualization section.

Implementation

Dataset extraction:

The relevant data was pooled together from the various data tables of the MIMIC-III database into a single SQL materialized view, which was then imported directly as a pandas dataframe.

Data preprocessing:

All data preprocessing and label calculations described in the preprocessing section were performed using pandas native functionality.

Classifier and metric definition:

I used scikit-learn tools and methodology throughout the classification process. The classifier used is the eXtreme Gradient Boosted decision tree ensemble classifier, which is not included in the scikit-learn library, but rather has an scikit-learn API allowing it to be used almost identically as other scikit-learn classifiers. Hyperparameter optimization and metrics were entirely implemented through scikit-learn. Hyperparameters were explored using grid search with 5-fold cross-validation (GridSearchCV). Due to the unbalanced nature of the data, stratified k-fold was used during cross-validation to preserve the class ratio throughout the different folds.

Refinement

Hyperparameter optimization for our classifier was performed by a multi-step grid search procedure with 5-fold cross validation at each step (using the scikit-learn implementation GridSearchCV) across the hyperparameter space, with ROC AUC as the scoring method. Briefly, a parameter range was set for each step and the classifier performance (averaged over a cross-validation split) was measured at each point in the range. The top performing combination of hyperparameters was then used to inform the next step, either by evaluating a narrower range around the same hyperparameters, or used as-is for a new set of hyperparameters. The order of hyperparameters to be optimized (and their definitions) is as follows:

1. *n_estimators* - The number of trees used.
2. *max_depth* - Maximum tree depth for base learners.
3. *min_child_weight* - Minimum sum of instance weight(hessian) needed in a child.
4. *gamma* - Minimum loss reduction required to make a further partition on a leaf node of the tree.
5. *colsample_bytree* - Subsample ratio of columns when constructing each tree.
6. *Subsample* - Subsample ratio of the training instance.
7. *Alpha* - L1 regularization term on weights.

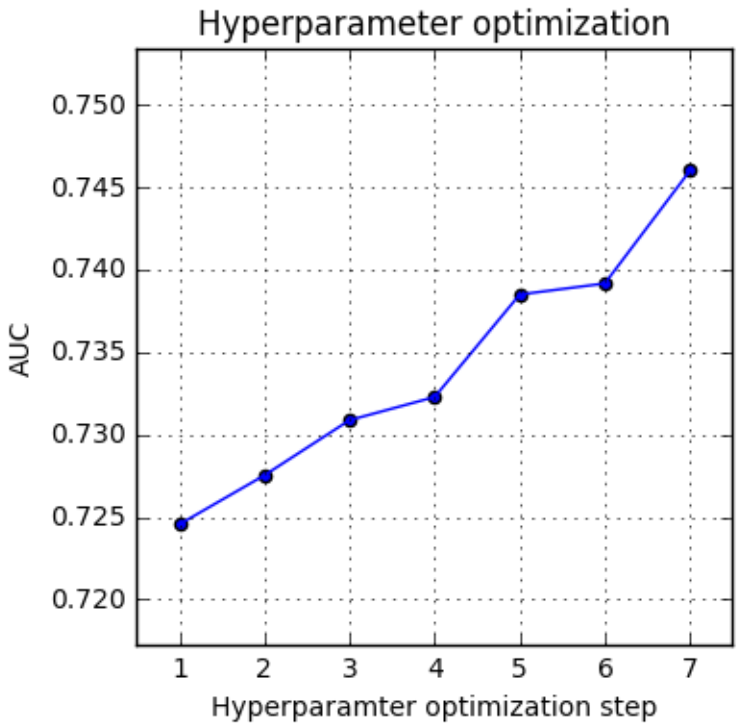

Figure 6: XGBoost classifier hyperparameter optimization. ROC AUC is depicted for each successive gridsearch step.

IV. Results

Model Evaluation and Validation

After hyperparameter optimization, the classifier was run over the entire cleaned dataset in a 5-fold stratified cross-validation process, and set up to provide prediction probabilities to produce a Receiver Operating Characteristic curve (see below). The area under the curve (AUC) was calculated for each fold, and a mean AUC was calculated at the end (figure 7). Some minor variation can be seen across the different folds, with a minimal AUC of ~0.72, a maximal AUC of 0.77 and a mean AUC of 0.75. The cross-validation process resulted in relatively consistent performance overall, and the model can be said to generalize quite well to unseen data.

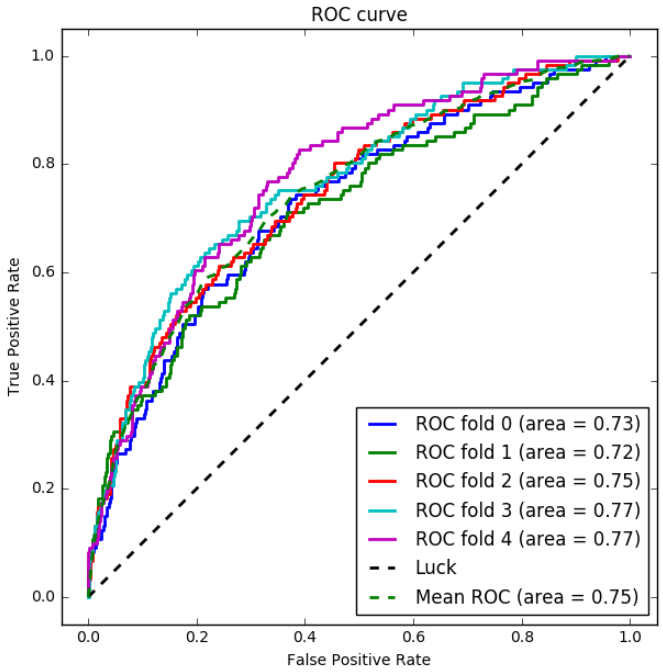

Figure 7: Receiver Operating Characteristic curve for the optimized model, using 5-fold cross validation.

Justification

To compare our model's performance to our predefined benchmark, we will look at the following classification report:

	Precision		Recall		F1-score		Support
	Benchmark	**Classifier**	**Benchmark**	**Classifier**	**Benchmark**	**Classifier**	
0 (no 30-day readmission)	0.88	0.96	1	0.74	0.93	0.83	4290
1 (30-day readmission)	0	0.3	0	0.78	0	0.43	605
Avg/Total	0.77	0.88	0.88	0.74	0.82	0.78	4895

Table 5: Performance comparison between benchmark and optimized XGBoost model.

As seen in table 5, readmission precision is low at only 30%, meaning that on average only one in three readmission predictions is correct. However, non-readmission precision shot up from 88% to 97%. Readmission recall went from 0 to 78%, meaning that 78% of all 30-day readmissions are "caught" by our model. Conversely, non-readmission recall was reduced from 100% to 74%, meaning 26% of non-readmissions were "missed" by our model. The F1 score summarizes the analysis above, with a decrease from 0.93 to 0.83 for

non-readmission, and an increase from 0 to 0.43 for readmission. With this analysis, it is safe to say that our classifier significantly outperforms the benchmark at predicting 30-day readmission.

V. Conclusion

Free-Form Visualization

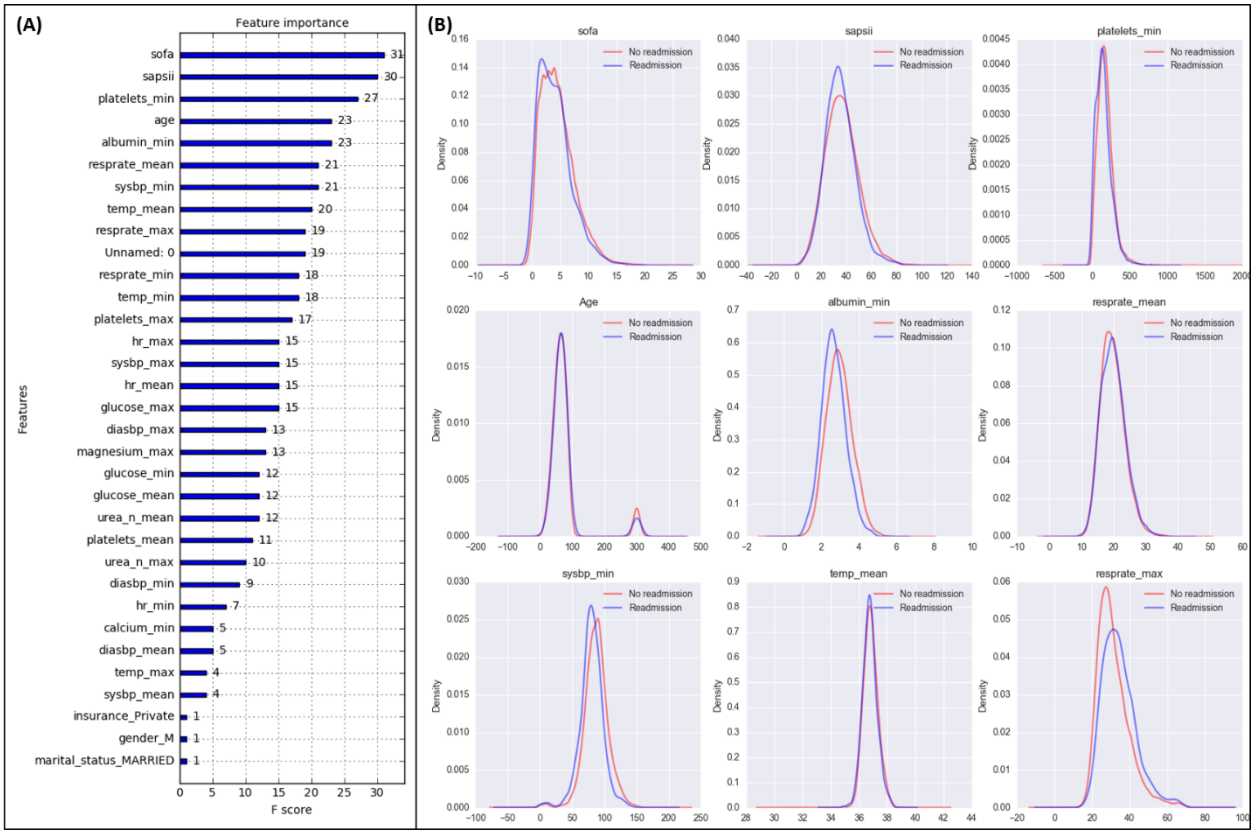

Figure 8: Important features. (A) shows the features sorted by "weight". (B) shows kde plots for the top 9 most important features, separated by label.

Figure 8 reveals some interesting insights about the features used in our model. Firstly, figure 8A shows the feature importance, calculated by "weight", i.e. the number of times each feature is used in a decision tree. The more times a feature is examined within a decision tree, the more important it is deemed. The top features in terms of importance are the sofa and sapsii scores. This is not surprising as these scores are in-fact engineered features, designed to capture the severity of a patient's situation. Figure 8B shows the top 9 most important features, and how their value distributions vary between negatively and positively-labeled observations. It is quite evident that the difference between the classes

are quite minute and in some cases, appear non-existent. Nevertheless, these differences appear to be important within the right context (i.e. the tuned decision tree).

Reflection

This project entailed exploring a large database with many different types of values (static numerical and categorical features, date-time objects, dynamic/time-series features, free-text notes), and wrangling the relevant features into a easily digestible pandas dataframe before applying fairly standard machine-learning methodology to obtain and optimize a predictive model. As I had very little SQL experience prior to this project, the first part presented the most significant challenge. Next, learning to build a predictive model for highly-unbalanced data proved highly educational, as I explored the various metrics used and delved deeply into where each one was most relevant.

The nature of the data and the required preprocessing steps resulted in a significant reduction in size of the relevant data, from nearly 60,000 observation in the raw data to slightly less than 5,000 in the final dataset. Nevertheless, the resulting model performed better than I expected. This is an exciting result, as real-world applications of such a model could have significant financial and public-health consequences.

Improvement

The MIMIC-III is an impressively wide database, containing information that goes far beyond what I ended up using for this project. One example is the "notes" table, which contains free textual notes from the various healthcare professionals (physicians, nurses) that oversaw the patient during each step of the admission. These notes contain information that could prove highly pertinent in an ICU readmissions context, such as a medical history specific conditions, hospital-acquired infections etc. Extraction of this information is likely to provide highly-predictive new features, but would involve a significant amount of work in processing the natural language of these notes, and therefore fell beyond the scope of this project.

VI. Bibliography

1. *MIMIC-III, a freely accessible critical care database. Johnson AEW, Pollard TJ, Shen L, Lehman L, Feng M, Ghassemi M, Moody B, Szolovits P, Celi LA, and Mark RG. Scientific Data (2016). DOI:* 10.1038/sdata.2016.35. Available from: http://www.nature.com/articles/sdata201635
2. RB Alfonso, Feature Extraction and Selection for Prediction of ICU Patient's Readmission Using Artificial Neural Networks, Master's thesis, 2013.
3. Nguyen et al., Predicting All-Cause Readmissions Using Electronic Health Record Data From the Entire Hospitalization: Model Development and Comparison, Journal of Hospital Medicine, 2016.
4. Kareliusson et al., Risk prediction of ICU readmission in a mixed surgical and medical population. Journal of Intensive Care, 2015.