Python Lab Essay

S. Gupta, L. Over, L. Sooter

March 22, 2020

1 Best Model

LightGBM produced the best results with the following train and validate AUC values:

- Train AUC=0.8371
- Validation AUC=0.8070

Table 1 shows the confusion matrix for the validation set. The values were calculated using the best threshold as determined by the J Statistic and F-Score.

Predicted True	0	1	All
0	10015	2464	12479
1	1252	2269	3521
All	11267	4733	16000

Table 1: Confusion matrix for the validation set with threshold 0.39

2 Method

In this python lab, four models were evaluated on surgical procedures data for procedures conducted between June 2017 and June 2018. The goal was to develop an algorithm to accurately predict a patient's level of risk for a length of stay (LOS) greater than five days post-surgery.

Four algorithms were created and evaluated using Area Under the Curve: Random Forests, LightGBM, XGBoost, and CatBoost. We chose these four algorithms because they all perform binary classification in different ways. A random forest creates multiple decision trees at one time, each from a different randomly selected set of variables, and then combines them into one, aggregated result at the end, either by 'the majority rules' or the average, depending on the variable type. The other three gradient boosting algorithms also create multiple trees, but only one at a time. Each new tree improves on the shortcomings of the previous trees. LightGBM is faster than XGBoost because it uses gradient one-sided sampling, which means it retains and stops training when gradients are low, i.e., more confident predictions. CatBoost handles non-numeric variables, thus categorical variables in the 'cat' dataset were left as strings and not converted to dummy variables. CatBoost can also prevent target leakage, which is when some information from the target is used to predict the target, and thus prevents overfitting.

3 Data Preprocessing

The training dataset contains 80,000 rows of data, and the test dataset contains 11,202 rows of data. A row represents one surgical procedure.

The training and test datasets were combined for preprocessing to make sure the same number of dummy variables were created for both sets. The following steps were taken to prepare the combined data for analysis.

1. New date variables were created including 1) the number of hours between the scheduled date and procedure date (actual date of the procedure), 2) the number of days since epoch for the procedure date, 3) the number of days since epoch for the scheduled date of the procedure, and 4) the number of days since epoch for the creation date (the date a record was created).

2. A description of numeric columns was evaluated for negative numbers. The ADI column has some negative values, but these were determined to be valid. LOS had seven rows with negative values. These are obvious errors and could have been deleted, however, they remain in the model as lengths of stay less than five.

3. Null values in categorical variables were populated with 'Unknown' or 'U.'

4. Null values in numeric variables were populated with -999.

5. A binary variable was created for LOS: abnormLOS was populated with a 1 if LOS was greater than five and a 0 if LOS was less than or equal to five.

6. Two copies of the data were created at this point, one for algorithms requiring numeric input and one for CatBoost, which allows categorical variables with string values.

7. With one copy of the dataset, dummy variables were created from the categorical variables.

8. The 'dummy' and 'cat' copies of the dataset were then each separated back into the original train and test sets.

9. Numeric variables in both 'dummy' and 'cat' datasets were scaled using z-score scaling.
10. The proportion of abnormal LOS to normal LOS was evaluated for the train set.
11. Both 'dummy' and 'cat' datasets were then split into train and validate sets, separately, using the same random_state seed.
12. The train datasets for 'dummy' and 'cat' were recreated using oversampling because only 22% of the data represented abnormal LOS. The same random_state seed was used to make sure the new samples were the same between the 'dummy' and 'cat' datasets.

4 Model Evaluation

The LightGBM model performed the best with a training AUC of 0.8371 and a validation AUC of 0.8070. Hyperband was implemented to find the best set of parameters. The resulting model was overfit with a train AUC of 0.86 and a validation AUC of 0.70. The following hyperparameters were modified to fix the overfitting:

- max_depth was reduced from 246 to 10
- reg_lambda was increased from 1.3 to 1.5
- reg_alpha was increased from 1.1 to 2
- min_num_samples was increased from 30 to 40
- learning_rate was decreased from 0.01 to 0.001

Table 2 below summarizes the final results for all of the models.

Model	Train AUC	Validation AUC	AUC Difference
Random Forest	0.8254	0.7990	0.0264
LightGBM	0.8371	0.8070	0.0301
XGBoost	0.7818	0.7714	0.0104
CatBoost	0.9400	0.8467	Overfit

Table 2: Train and validation AUC values and differences for each model

The CatBoost model is considerably overfit. Many parameters were modified to attempt to take advantage of CatBoost's ability to prevent overfitting, however, the AUC values did not change.

It is important to note that the AUC values had little, if any, improvement when the oversampling technique was applied. However, the accuracy of the models improved greatly. These values for the Random Forest and LightGBM models are summarized in Table 3 below.

	No Oversampling		Oversampling	
Model	AUC	Accuracy	AUC	Accuracy
Random Forest	0.7990	0.55	0.7939	0.70
LightGBM	0.7858	0.58	0.8070	0.77

Table 3: AUC and accuracy for Random Forest and LightGBM, with and without oversampling

5 Deployment

For well-fit models, i.e., those that are not overfit or underfit, the G-Mean, J Statistic, and F-Score were calculated to select the best threshold for classifying patients as either low or high risk for a long LOS.

The train and validate accuracy values and the decision threshold for each well-fit model are summarized in Table 4 below.

Model	Train Accuracy	Val Accuracy	Decision Threshold
Random Forest	0.74	0.70	0.45
LightGBM	0.76	0.77	0.39
XGBoost	0.69	0.63	0.75

Table 4: Train and validation accuracy and decision threshold for each well-fit model

For the best model, LightGBM, the validation set accuracy is a little higher than the train set accuracy. The validation set accuracy for the LightGBM model is higher than the accuracy for any other model. This further confirms that the LightGBM model is the best model.

For the LightGBM model, the threshold for classifying long LOS is 0.39. Table 5 below shows the low and high risk prediction bins for this model.

	Low Risk	High Risk
prediction	< 0.39	≥ 0.39

Table 5: Low risk and high risk bins for LightGBM model

Figure 1 below shows a portion of the SHAP summary plot for the LightGBM validation set.

The SHAP summary plot for the validation set shows that PHOSPHORUS_CLOSEST is the most influential variable to the model. High values of PHOSPHORUS_CLOSEST yield high SHAP values, which contribute to increasing the predicted probability. Low values of this variable results in lower SHAP values, and thus a lower predicted probability.

The variable SCHED_SURG_PROC_CD is listed in the summary plot as the second most influential variable, however, the plot says that low values both increase and decrease the predicted probability. This is difficult to interpret.

The third most influential variable is PREALBUMIN_CLOSEST. High values of this variable increase the predicted probability while low values decrease it.

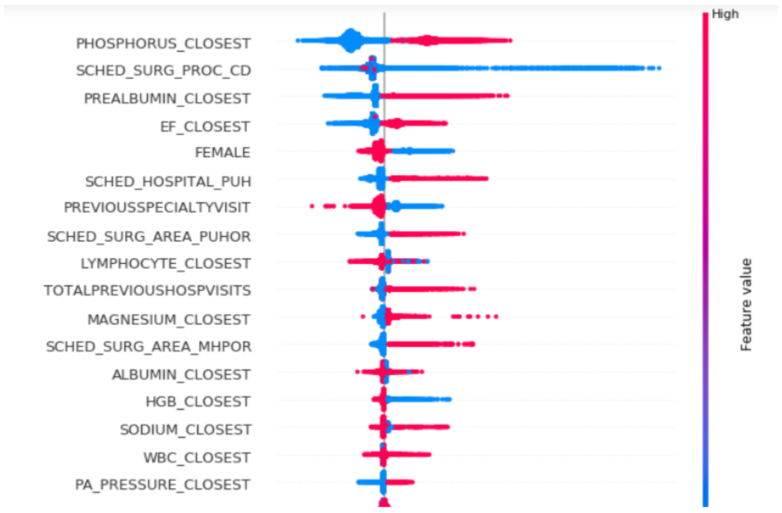

Figure 1: SHAP summary plot for LightGBM validation set

With an accuracy of 77%, this model is better than flipping a coin and should be deployed to assist physicians and hospitals in taking steps to prevent long lengths of stay. For example, physicians could postpone elective surgeries or attempt to mitigate the impact for patients who test high in phosphorus and prealbumin.

An interactive dashboard would provide hospitals with visualizations of the data, the model, and the SHAP values and could be created using Dash, an Open Source Python library for creating web applications.

6 References

Introducing Dash by Plotly on Medium
`https://medium.com/plotly/introducing-dash-5ecf7191b503`