CMPT 459 Spring 2021

Milestone 2

Zirui Huang 301307482
Henry Ye 301368702

1. Splitting Dataset

We split the dataset into training dataset and validation dataset with 8:2 ratio using numpy.split.

2. Build Models

We selected LightGBM and Random Forest as our models. Implementation of both models requires attributes in numerical format, even for categorical attributes. Therefore, we first convert categorical values into numbers (i.e 0, 1, …, n).

The reasons why we choose LightGBM compared with XGBoost and Adaboost etc are

- LightGBM model training time is less than the others and high efficiency: LGBM classifies continuous features into discrete bins using histogram.
- More accuracy than others: Based on the more complex tree algorithm, the leaf wise split with complex structure leads to higher accuracy.
- It could handle large dataset with a reduction in training time compared with others

We choose Random Forest based on following reasons

- It works well for both numerical and categorical attributes.
- No normalization is required as it uses a rule-based approach.
- It reduces overfitting in the decision tree to improve accuracy.
- It runs efficiently on a large dataset.

SVM has a long training process and the result is not poor because of the large data set and there is not proper kernel function. Besides, Naive Bayes is training fast but the performance is poor due to the assumption where the features are conditionally independent but there is a relationship between recovered, active, death. KNN takes a long time to predict with a large data set.

The parameter we choose for Random Forest model are n_estimators=100, max_depth=30

The parameters we choose for the LightGBM model are n_estimators = 50, boosting_type = gbdt, num_leaves = 200, with comparison of other parameters by Hyperparameter Tuning. The different parameter tables will be found in the plot folder.

3. Model Evaluation

The metrics we used to evaluate our models including *accuracy score, confusion matrix, precision and recall*.

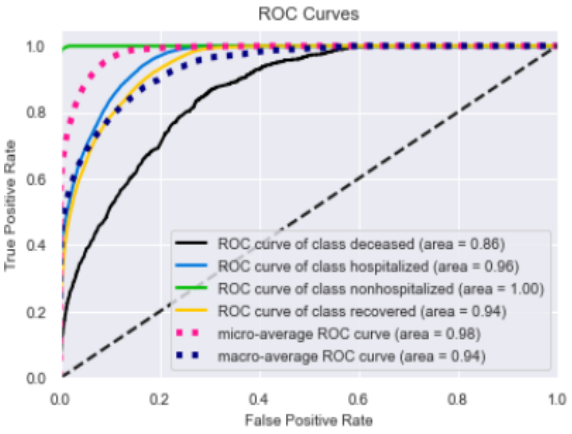
		Accuracy	Precision				Recall				F1-score				Confusion Matrix
Model	Data	Average	'd'	'h'	'n'	'r'	'd'	'h'	'n'	'r'	'd'	'h'	'n'	'r'	Have a problem with
RF	Train	0.887	0.96	0.81	1	0.81	0.23	0.89	1	0.73	0.37	0.85	1	0.77	classifying deceased
	Test	0.869	0.49	0.79	0.99	0.77	0.07	0.87	0.99	0.7	0.12	0.83	0.99	0.73	

Model	Data	Accuracy	Precision	Recall	hospitaliz	F1 nonhospitalized	F1 deceased	F1 recovered	Confusion Matrix
LGBM	Train	0.884	0.886	0.884	0.844	0.999	0.366	0.761	Have a problem with classifying
	Test	0.868	0.864	0.868	0.831	0.991	0.114	0.734	deceased and recovered

Both models have a good *accuracy score*. However, accuracy score alone is not a good measure of classifier performance when the classes are imbalanced. This means those labels with small numbers of occurence (such as *deceased* in this dataset) may not be correctly classified in the Random Forest model and *deceased and recovered* in the LGBM model.

A *confusion matrix* is a way to express how many of a classifier's predictions were correct, and when incorrect, where the classifier got confused. By using a confusion matrix, we can

intuitively tell exactly where mistakes were made and how well each of the four labels are classified. *Precision*, *recall*, and *F1* score give us even more insight into model performance. We know there is a tradeoff between precision and recall, in cases where we want to find an optimal blend of precision and recall we can combine the two metrics using the F1 score.

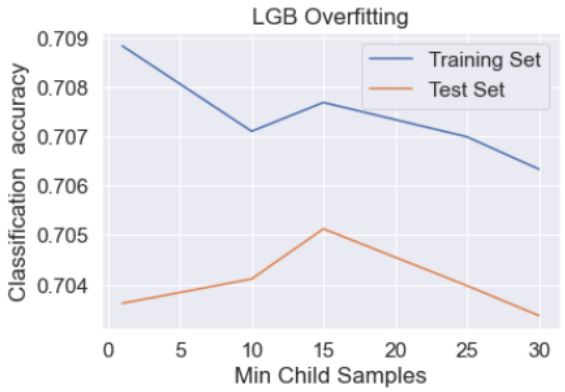


The ROC curve of LGBM demonstrates the tradeoff between sensitivity and specificity for all possible thresholds rather than just one chosen by the model because it does not only depend on class distribution. The better performance classifiers have, the closer to the top-left corner. If the curve closed to the dot black line, the accuracy will be less. From the plot of the ROC curve, we could easily find the three solid curves on the top left, which are green, blue and yellow. The area value is AUC that offers a measure of performance across all possible classification thresholds. They indicates the model can predict accurately with the label hospitalized(AUC = 0.95) and recovered(AUC = 0.93) and with the nearly perfect prediction non hospitalized (AUC = 1)

To conclude, using both of these evaluation metrics can give us a much more nuanced understanding of how our model performs.

4. Overfitting

We do observe overfitting in our models because evaluation metrics are higher in training dataset. To detect and reduce overfitting, we use cross validation in hyperparameter tuning. To find the best value of a particular parameter, we test the model with a range of values and pick the one that results in best performance in the cross validation. The parameters we tuned are *n_estimators* (200)*, max_features* (10)*, min_samples_leaf* (1) in the Random Forest model.



LGBM:

The parameters we fix are the number of estimators as 1 and num_leaves as 1000, which means there is only one tree in our model. With the number of estimators increasing, the training time will also increase. The only parameter we adjust is the min_child_samples as [1,10,15,25,30], we can determine where the classification accuracy decreases in training data but increase in test data in the range of 5 to 10, but the training accuracy score is greater than test data. The reason why it has overfitting is that the LGBM model is based on the decision tree, so it may not restrict the depth and min child sample when it does the split process.