KKBox's User Churn Prediction

Yujia Wang, Liyi Cao, Tong Ye, Weixuan Jiang
yujia8@bu.edu, caoliyi@bu.edu , tongyewh@bu.edu, jwx0728@bu.edu

1. Project Task

KKBOX is an Asia's leading music streaming service and it holds tens of millions of Asia-Pop musics and since it provides services to millions of people, they need a model on accurately predicting churn of their paid users.

So our input is the data provided by Kaggle, and the task is to build an algorithm that predicts whether users will be lost after the subscription expires. And to analyze the reason for the user leaving in order to be proactive in keeping users by using different methods.

2. Related Work

Currently, the company uses survival analysis techniques to determine the residual membership life time for each subscriber. Survival analysis is a statistical method that considers the results and considers the survival time. It can make full use of the incomplete information provided by the censored data to describe the distribution characteristics of survival time and analyze the main factors affecting the survival time.

3. Approach

3.1 Logistic Regression: Logistic Regression is a basic classification method taught in class. Basically, it uses a function to model a binary variable.

$$g(z) = \frac{1}{1+e^{-z}} \qquad h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

And when $h_\theta(x)$> 0.5, we predict it as 1;else we predict it as 0.

3.2 Neural Network: Neural networks are a set of artificial neurons like the human brain, which are designed to recognize patterns. They interpret sensory data through Neural network, labeling or clustering raw input.

3.3 LightGBM: LightGBM contains two key points: light is lightweight, GBM gradient hoist**.** It is a gradient boosting framework that uses a decision tree based on learning algorithms. It can be said to be a distributed, efficient machine learning algorithm.

3.4 Random Forest: Random forest refers to a classifier that uses multiple trees to train and predict samples. And the category of its output is determined by the mode of the category output by the individual tree.

3.5 Grid search:

Grid search is the classic optimization method to get the best hyperparameters. It will exhaustively try every combination of the hyperparameters in the subsets and calculate the output of each pair. Finally, we will choose the one that has the best performance and use it in our model. Grid search guarantees to find out the best result in the subsets, but it always takes too much time.

3.6 Bayesian Optimization:

Since the objective function is unknown, the Bayesian strategy is to treat it as a random function and place a prior over it.

The first step is to have a prior guessing of our model. Since we need to find the best optimization that satisfies the condition, if we know the function graph of the model, we can directly calculate the best parameters. But we don't know what the model looks like, and what distribution features the function has. So we can only guess first.

In general, our common assumption is that the function satisfies the Gaussian distribution, which is a normal distribution. Of course, there are other distributions besides GP. Different distribution assumptions are needed for different problems. If we choose Gaussian distribution here, now if I input the two sets of data into the GP model, then the model can be modified to make it closer to the real distribution of the objective function. Then we have a corrected GP. We choose a point what we interest from the modified GP according to the Acquisition Method, to make the GP get closer to the real distribution of the objective function faster and more accurately.

If the point found in the above step does not meet our needs, then this point will return to the GP model, and correct it again.

4. Dataset and Evaluation Metric

4.1.Dataset

Our data set comes from the WSDM - KKBox's Churn Prediction Challenge project on kaggle, which contains

7 files. By analyzing, each user has 23 features, but in fact only 13 of the features that help us work. Converting all the features into one hot code, we got the new 1082190*61 training document and the 907470*61 test document. Due to too much data and excessive memory consumption, we reduced the memory by converting the data type. In addition, we also encountered some outliers when processing the data set. We use data replenishment and data filtering methods for data cleaning. Finally, we scaled the data from 0 to 1, and the scaled data are only used for logistic regression and Neural Networks.

4.2. Evaluation Metrics:

Though calculating the accuracy is a direct way to evaluate the model, the users who churned is only a minor part of all the users. So probably the accuracy is always very high, it's definitely not a good way to evaluate our model.

Then we decide to use logloss of binary classification to judge if our model is good or not. So here is the problem: How low should the logloss be, to determine our model is good? That's to say, is there a threshold to determine our model actually did a good job in classification?

The percentage for the users churned in training data is about 14%. If we assume the distribution of testing data is the same as training data, it will be also 14%. So according to logloss:

$$logloss = -\sum_{i=1}^{n}\left(\frac{y_i}{n}\log(p_i) + \frac{1-y_i}{n}\log(1-p_i)\right)$$

If we know the distribution in advance, and assign all of the probability 0.14, just like a random guessing, the logloss we get from that will be:

$$logloss = -(0.14log(0.14) + 0.86log(0.86)) \approx 0.405$$

So that's to say, if we are sure our distribution is about 14% of the testing users are churned, and the logloss is lower than like 0.405, our prediction will be not bad. And the lower the logloss we got, the better model we will have.

Eventually we will upload our predicted probabilistic results on kaggle. The system will automatically score for us.

5. Results

5.1 Logistics Regression

We tuning the parameters of Inverse of regularization strength and choose the form of regularization.

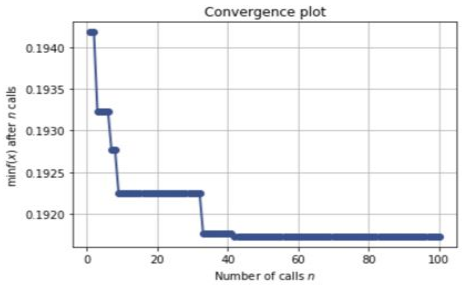
After using the grid search to get the best pairs in in the red. And use these in our testing dataset, the logloss of the result is 0.14948. It is worse than the LightGBM because the logistic regression need the independent observation. If the observations are related to one another, the model will tend to overweight the significance of those observations. Also it's likely to be overfitting.

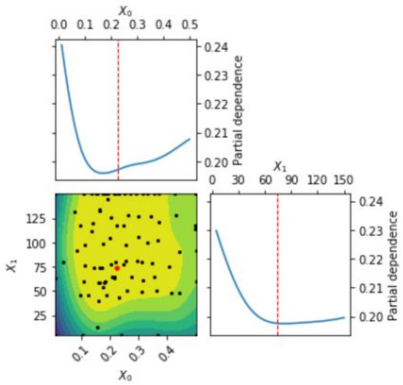
Penalty \ C	0.01	0.1	1	10	100
L1	0.274892	0.274657	0.274634	0.274028	0.274469
L2	0.274829	0.274670	0.273530	0.273675	0.274472

5.2 LightGBM

According to the Bayesian Optimization method we discussed in 3.5, we imported a python package named SKOPT for the optimization directly.

First we did some test of it, and chose 'learning rate' and 'max bin' for hyperparameters tuning. For this experience, we set 'learning rate' from 0.01 to 0.5, and the 'max_bin' from 5 to 150.

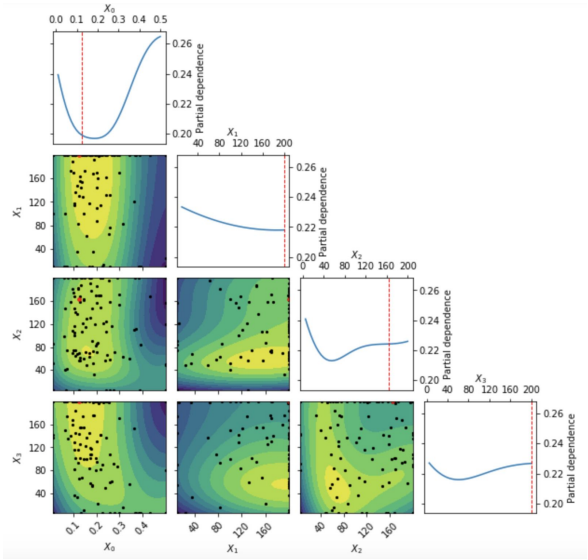




As you can see, the output(logloss) converges after about 40 calls of fitting. And the best combination of these two parameters is when learning rate is about 0.2 and 'max_bin' is about 70. The logloss is finally converged to about 0.1917.

Then we chose some scales of 'learning rate','num_leaves','max_bin' and 'n_estimators'. And we used the mean of K-fold cross validation to be the objective function to be minimized. In this experience, the 'learning rate' is set from 0.01 to 0.5, the

'num_leaves' is set from 5 to 200, then 'max_bin' 5 to 200, finally the 'n_estimators' also from 5 to 200. Typically, the 'num_leaves' means the maximum tree leaves for base learners, the 'max_bin' means number of bins used when discretizing continuous features, and the 'n_estimators' means the number of boosted trees to fit. So generally the model becomes better when these hyperparameters become larger.

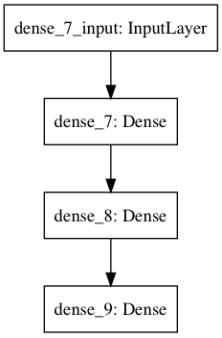


In this plot, you can find the combination' partial dependence easily. The yellow area means the Bayesian Optimization process is going to explore more points around here(we have discussed it in 3.5 too). And the best combination for these 4 hyperparameters is [0.12317374291398388, 200, 164, 200]. Actually, some parameters just reach their upper bounds, this is mainly because, to some extent, the more numbers of leaves and estimators a model has, the better accuracy the model will have. And the logloss is about 0.1812 for this model.

5.3 Neural Network

We also tried a Neural Network from Keras to predict this binary classification problem.

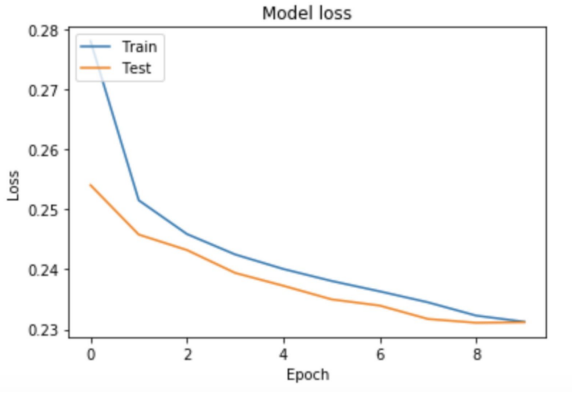
First we tried a simple Neural Network, before we do that, we scaled all the data from 0 to 1 first. And the whole construction of this model is as the right image shown:



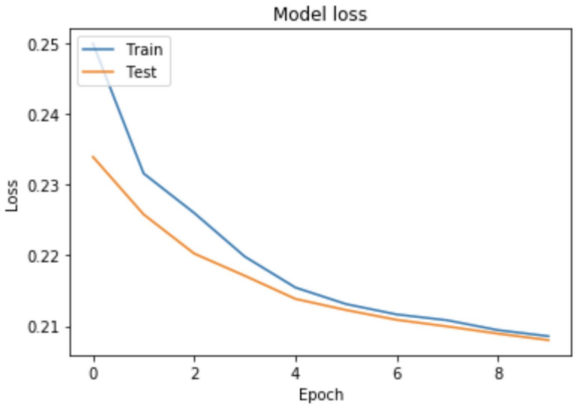
Note that dense_7 has 32 neurons with RELU activation function, dense_8 has 16 neurons also with RELU activation function, and dense_9 has only 1 neuron with Sigmoid activation function. It's a simple NN and we ran 10 epochs with 10% validations for this model.

The result is as the image shown:

The train and validation logloss converges in about 8 epochs. And the result is around 0.232 for both train and validation.



Then we add one more hidden layer to the model, and add some neurons for each layer, except the last one. So now the neurons for each layer becomes [512, 256, 128, 1]. We also ran 10 epochs with 10% validations for this model. The result is as it shown below:



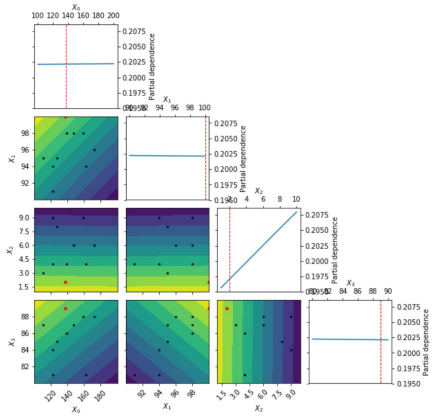
The train and validation logloss converges in about 7 epochs. And the result is around 0.202 for both train and validation dataset.

So the first guessing up to our mind is that generally the more neurons we have for the model, the better accuracy we will have. However, it's not like that. When we tried to add more hidden layers and sizes for each layer, the validation logloss becomes bad. After we read some paper, we found that ***'the optimal size of the hidden layer is usually between the size of the input and size of the output layers'.*** So generally, it has to follow two principles to make a simple but generally optimized neural network: number of hidden layers is one; the number of neurons in that layer is the mean of the neurons in the input and

output layers. It's a very useful conclusion, which is referenced from:This link

5.4.Random Forest

We use a Random Forest Classifier from sklearn library. And apply Bayesian Optimization method to select the best parameters of decision trees. We chose some scales of parameters: 'n_estimators', 'max_depth', 'min_samples_leaf','min_samples_split'. But we found that the scales of parameters is hard to limit. The scales we chose are shown below. The straight line means the scales we chose are not suitable. Finally, we chose 'n_estimators' = 138, 'max_depth'=100,'min_samples_split'=109, 'min_samples_leaf' = 1.



As the result shows, the logloss of RF classifier is around 0.21. And the kaggle score is listed below.

```
1 log_loss(y_test_cv,result[:,1])
```

```
0.20786568513026854
```

Kaggle Leaderboard

#	change	Team	Score	Entries
1	+1	Bryan Gregory	0.07974	84
2	+3	Swimming	0.08926	15
3	+3	JonahWang	0.09344	25
...	-			
64	-	LightGBM **(ours)**	0.12108	48
325		Logistic Regression **(ours)**	0.14948	5
341	-	Random Forest **(ours)**	0.15352	16
N/A		Neural Network**(ours)**	0.181	20

So our best result is from LGBM, the rank is 64/575.(Actually our name is not on the leaderboard because the competition is closed 1 year ago)

6. Timeline and Roles

Task	**Deadline**	**Lead**
Data collection and cleaning	11/04/18	Liyi Cao
Implement Logistic regression	11/14/18	Tong Ye
Research in Bayesian Optimization	11/20/18	Yujia Wang
Implement LightGBM	11/24/18	Yujia Wang
Implement Random forest	12/01/2018	Weixuan Jiang
Implement Neural network	12/07/2018	Liyi Cao&Yujia Wang
Prepare report and presentation	12/10/2018	ALL

References

1) Ma, X., Sha, J., Wang, D., Yu, Y., Yang, Q., & Niu, X. (2018). Study on a prediction of P2P network loan default based on the machine learning LightGBM and XGboost algorithms according to different high dimensional data cleaning. *Electronic Commerce Research and Applications*, *31*, 24-39.
2) Specht, D. F. (1991). A general regression neural network. *IEEE transactions on neural networks*, *2*(6), 568-576.
3) Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems* (pp. 2951-2959).
4) Svetnik, V., Liaw, A., Tong, C., Culberson, J. C., Sheridan, R. P., & Feuston, B. P. (2003). Random forest: a classification and regression tool for compound classification and QSAR modeling. *Journal of chemical information and computer sciences*, *43*(6), 1947-1958.