To stay or cancel? Predicting music subscription cancellations

Akshay Subramaniam, Man-Long Wong and Sravya Nimmagadda

{akshays,wongml,sravya}@stanford.edu

What are we predicting?

The goal of this project is to build an algorithm to predict if subscription users of a music streaming service will churn or stay after their current membership expires. Data for this is provided by the KKBox [1] music subscription service.
Prediction of whether a member will churn or not is based on intrinsic member data like age and city, their transaction logs and their user logs.

Data

The data is obtained from the *WSDM - KKBox's Churn Prediction Challenge* on Kaggle [2]. There are mainly three datasets: member information, transaction logs and daily user logs. The member information has the basic information for each user, such as age, gender, registration date, etc. User transaction logs contains the 2-year transaction records of each user and the user daily logs describe the listening behavior of each user.
One of the challenges with this dataset is that it is highly unbalanced. Every month, only $\sim 7\%$ of users churn. For such a dataset, even naively predicting that no user will churn has a 93% accuracy and better metrics than just accuracy should be used to judge the performance of an algorithm.

Features

During the feature engineering process, in total 66 features were selected. These include some raw features in the original datasets such as age, city, payment method in the last transaction, etc. We also derived some additional features, like if the user had any discount, and some historical statistics for the logs.

Models

- Logistic Regression:
  - Conditional distribution of $p(y|x)$ modeled as Bernoulli distribution

$$J_\theta = \sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log[1 - h_\theta(x^{(i)})]$$

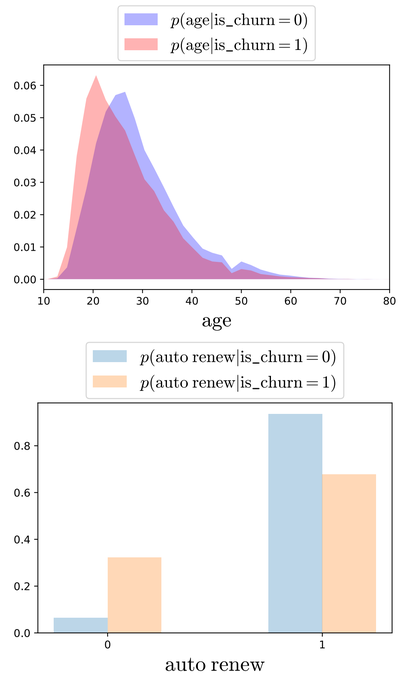
- Multi-layer Neural Network (Neural Net):
  - Three hidden layers with tanh function used as activation
  - Mini-batch gradient descent on cross-entropy (CE) loss function with regularization:

$$J_{MB} = \left(\frac{1}{B}\sum_{i=1}^{B} CE(y^{(i)}, \hat{y}^{(i)})\right) + \lambda \left(\sum_{k=1}^{4} \left\|W^{[k]}\right\|^2\right)$$

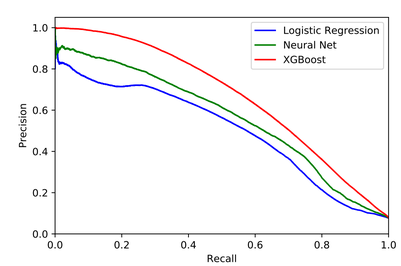
- Gradient Boosted Trees (XGBoost):
  - Ensemble of decision trees
  - Optimize one level of the trees at a time with objective function:

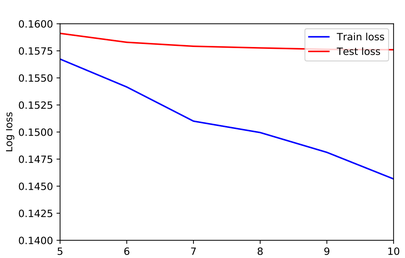
$$\sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \Omega(f_t)$$

Feature Engineering



Results





Model	Size		Log Loss		Test
	Train	Test	Train	Test	Recall
XGBoost (Full dataset)	1.57M	0.39M	0.15	0.16	0.67
XGBoost (Balanced dataset)	0.24M	0.39M	0.36	0.37	0.69
Neural Net (Full dataset)	1.57M	0.39M	0.17	0.17	0.63
Neural Net (Balanced dataset)	0.24M	0.39M	0.44	0.43	0.59
Logistic Regression (20% of full dataset)	0.31M	0.08M	0.18	0.19	0.58
Logistic Regression (Balanced dataset)	0.24M	0.39M	0.45	0.43	0.6

Discussion

We deployed three different binary classification models to predict whether users will cancel their subscription in the next month. The accuracies of the method was compared and XGBoost had best prediction performance based on log loss and recall metrics. The predictions from gradient boosted trees model gave us the cross-entropy loss of 0.117 on unseen data and we ranked 35th out of 469 teams on Kaggle.
A major part of the work for this project was feature engineering. We used a Naive-Bayes like approach to compute conditional probability densities and used that to inform our choice of features. We also added intuitive features like discount and cost per day.
We paid close attention to normalizing data and not using any future data in our training process. All dates were normalized by the last day of the previous month and all user logs were normalized by the total number of logs so as to not bias data for users with long histories.

Future Work

To improve our predictions, one possible way is to go through the logs and generate churn labels for past months. This way, we can increase our dataset size by an order of magnitude and potentially get much better predictions. Another option would be to use a kernelized SVM classifier. We didn't pursue this approach since our dataset is large and training is prohibitively expensive for > 200000 examples.

References

[1] *KKBOX*. https://www.kkbox.com/.
[2] *Kaggle*. https://www.kaggle.com.