CE4041 Machine Learning

Assignment Report

KKBOX's Churn Prediction Challenge

Group 20

Choo Wei Liang U1522460A

Andrew Koh U1522383H

Feng Wei U1521064L

Neo Boon Kiat U1521662G

Table of Contents

1.0 Project Description

1.1 | Background

KKBOX is a subscription-based music streaming service. There are many different types of subscription and renewal methods. When a user signs up for the service, the user can either renew their subscription manually or automatically renew their subscription. A user can cancel their membership at any time.

Our goal is to predict whether a user will churn within 30 days after the user's subscription expires. A churn is defined as no new subscription within 30 days after the current membership expires.

1.2 | Goal

To predict whether a user will churn within 30 days after the user's subscription expires.

1.3 | Data Provided

We are provided with 5 data sets,

1. TRAIN.CSV

   train.csv is the training set for us to use to train our model. It contains:

   1. msno, the id of the user.
   2. is_churn, what we are supposed to predict. If is_churn = 1, the user has churned. If is_churn = 0, the user has not churned.

2. TRANSACTIONS.CSV

   transactions.csv contains the information of transactions made regarding the service. It contains:

   1. msno, the id of the user.
   2. payment_method_id, the type of payment method.
   3. payment_plan_days, the length of the membership plan (days).
   4. plan_list_price, the list price of the plan (in New Taiwan Dollar).
   5. actual_amount_paid, the actual amount paid for the plan (in New Taiwan Dollar).
   6. is_auto_renew, whether the member has enabled auto renewal. If auto_renewal = 1, it is enabled. If auto_renewal = 0, it is disabled.
   7. transaction_date, the transaction date (format %y%m%d).

8. membership_expire_date, the expiry date of the membership (format %y%m%d).
9. is_cancel, whether the user has cancelled their membership. If is_cancel = 1, the user has cancelled. If is_cancel = 0, the user has not cancelled.

3. USER_LOGS.CSV

user_logs.csv contains the daily user logs of the listening behaviours of a member. It contains:

1. msno, the id of the user.
2. date, the date the user listened to the song (format %y%m%d).
3. num_25, the number of songs that are played less than 25% of the song length.
4. num_50, the number of songs that are played between 25% to 50% of the song length.
5. num_75, the number of songs played that are between 50% to 75% of the song length.
6. num_985, the number of songs played that are between 75% to 98.5% of the song length.
7. num_100, the number of songs that are played over 98.5% of the song length.
8. num_unq, the number of unique songs played.
9. total_secs, the total seconds of all the songs played.

4. MEMBERS.CSV

members.csv contains the information about each member subscribed to the service. It contains:

1. msno, the id of the user.
2. city, the city the user lives in.
3. bd, the age of the user.
4. gender, the gender of the user.
5. registered_via, the registration method used by the user.
6. registration_init_time, the time the member registered (format %y%m%d).
7. expiration_date, the expiration date of the subscription (format %y%m%d).

5. SAMPLE_SUBMISSION.CSV

sample_submission.csv is the set of data for us to submit after we have run our model. It contains:

1. msno, the id of the user.
2. is_churn, our prediction of whether the user has churned. If is_churn = 1, we predict that the user will churn. If is_churn = 0, we predict that the user will not churn.

1.4 | Definition of Terms

Word	Definition
Auto Renewal	A user automatically renews their subscription at the end of the month.
Churn	A user does not re-subscribe to KKBOX within 30 days after their current membership expires
Member	A person who is subscribed to KKBOX
Manual Renewal	A user manually renews their subscription.
Subscription	An active service that allows a user to use KKBOX.

1.5 | Roles

Member	Role
Choo Wei Liang	Project Report (main content) Coding (features)
Andrew Koh	Project Report (coding content) Coding (algorithm)
Feng Wei	Project Report (coding content) Coding (algorithm)
Neo Boon Kiat	Project Report (main content) Coding (features)

2.0 Problem Statement

The challenges that could be faced when trying to derive the solution are in no order of magnitude,

1. The definition of a “churn”. The definition of a “churn” and its application can be very vague. The duration of a subscription is 30 days, and within those 30 days, customers can re-subscribe at any time. To cater to this vague definition of a “churn”, our algorithm had to also be vague and not as precise as desired. This could lead to less accurate predictions.
2. The definition of a “cancellation”. According to the official project’s documentation, “A user may cancel service subscription due to change of service plans or other reasons.”. To put it plainly, a user can cancel his/her subscription and re-subscribe under a different plan. The user can therefore be not considered a churn but have had cancelled his subscription. This creates a more complex algorithm that has to take all these factors into account.
3. Coming up with relevant and useful features. Some of the data were said to be incorrectly labelled or not even labelled at all. This can make it a bit tricky when trying to add features that uses those data as the new feature may also contain data that is incorrect or incorrectly labelled.

3.0 Methodology

3.1 | Concept

Our concept revolves around trying to find correlations between members who churn or do not churn, and the given raw sets of data (*Figure 1*). Once we had come up with new features, we ran multiple algorithms to see which one performed the best for us.

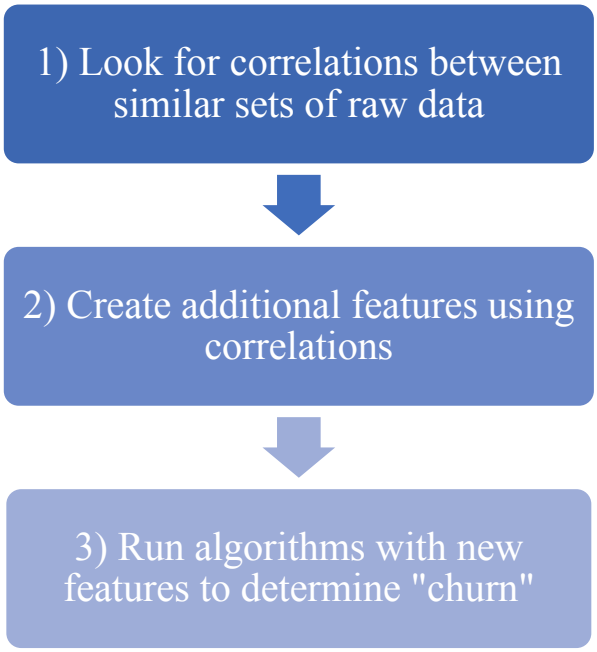

Figure 1

3.2 | Description

There are always correlations between what people do and the outcome of what they do. If a person searches about the price of a book on Google, the correlation is that they are probably going to be interested in buying a book in the future. It is basis Google's AdSense model. As such, we decided to employ this model in coming up with additional features to best help predict the outcome of what a member that has subscribed to KKBOX's service will do. That is, whether they will churn.

We first tried to look for correlations between any given two sets of data. For example, if the subscription type is set to autorenew (*is_auto_renew* = 1), the member is less likely to cancel his/her subscription service.

After which, we thought about new additional features that may not have any correlation between them but may be beneficial in determining a churn. These features were more complicated to derive and code.

3.3 | Added Features

We added a total of 6 features,

S/N	Name	Description
01	discount	How much discount was offered.
02	is_discount	Whether there was any discount used.
03	amount_per_day	The cost of the plan per day.
04	membership_duration	The duration of the membership.
05	autorenew_&_not_cancel	Whether members have auto-renewed and cancelled.
06	notAutorenew_&_cancel	Whether members have not auto-renewed and cancelled.

3.4 | Derivation of Added Features

S/N	Name	Description
01	discount	plan_list_price - actual_amount_paid
02	is_discount	if discount > 0 = TRUE, if discount <= 0 = FALSE
03	amount_per_day	actual_amount_paid/payment_plan_ days
04	membership_duration	transaction_date - membership_expire_date
05	autorenew_&_not_cancel	is_auto_renew && is_cancel
06	notAutorenew_&_cancel	~autorenew_&&_not_cancel

4.0 Proposed Solution

4.1 | Explanation

Feature engineering is fundamental in this challenge as the raw data provided did not provide a good representation of the outcome to predict.

4.2 | Experiments

Many experiments were conducted to achieve a highest score of 0.122, position 70/575 (top 12%). We used about 100k data points for validation and testing, and 800k data points for training.

For all our experiments, training accuracy and f1 score were generally quite high, at about 0.7-0.8 and 0.94 – 9.80 respectively.

Various models were tried and are delineated below,

Multilayer Perceptron (MLP)

```
clfa = MLPClassifier(solver = 'adam', alpha = 0.0001, hidden_layer_sizes= (15, 8, 4), verbose=True)
```

Figure 2

We built a neural network in the early stages of our experiments (*Figure 2*). The neural network had 3 hidden layers (15, 8, 4). The number of perceptrons in each layer was decided intuitively; there were about 16 input features leading up to a binary outcome. We realised that increasing the number of layers and number of perceptrons had limited improvements in score but resulted in long computation times.

However, we managed to reduce our initial score of 0.27464 to 0.22823 through sheer hyper-parameters tuning (*Figure 3*)

Figure 3

XGBoost (XGB)

We next tried a popular gradient boosting tree to train our data on. XGBoost had much more success than the MLP (*Figure 4*). Using the same dataset that we trained the MLP on, we achieved a score of 0.190. Yet this was not good enough.

Figure 4

After augmenting the dataset (described in 5.0 Problems Faced and Solutions), we managed to get the score down to 0.135 (*Figure 5*).

Figure 5

CATBoost (CAT)

Another state of the art gradient boosting tree was used to train our dataset on. We did not perform much hyper-parameters tuning as by now, we realised that the quality of the features took precedence over the hyper-parameters in improving our score.

On our unaugmented dataset, CATBoost did slightly better than MLP but worse than XGB (*Figure 6*).

Figure 6

On the augmented dataset, CATBoost did much better than MLP but still slightly worse than XGB (*Figure 7*).

Figure 7

LightGBM (LGB)

We also another popular boosting algorithm, LightGBM. As we had already asserted that our augmented dataset improved our score tremendously, we did not try the non-augmented dataset on LGB (*Figure 8*).

Figure 8

LGB's performance was average even though it was much faster. This is probably because LGB uses some sort of modified average encoding for categorical data. This might have caused overfitting and as a result would not perform as well in the test data.

Ensemble

We performed ensemble learning with all the results. We averaged the score of the models we wanted to include in the ensemble to get the final score.

We first tried MLP with XGB which did not do very well. It was at this point we decided to remove MLP from the ensemble and instead focus on gradient boosting trees instead (*Figure 9*).

Figure 9

We then tried XGB with CAT. This gave us our best score (*Figure 10*).

Figure 10

Lastly, we tried XGB, CAT and LGB (*Figure 11*).

Figure 11

This performed well and confirmed our beliefs that ensemble learning worked much better than their individual constituents.

In summary, Gradient Boosting trees performed the best, and neural networks failed to give a good prediction. This was probably because this challenge to predict churn rate is less intuitive and the data provided did not directly correlate to the churn rate. If we had higher level features or data, MLP might have worked better.

4.3 | Scores

The scores of all the algorithms and combinations that we used are listed below,

S/N	Algorithm	Score	Figure No.
01	Multilayer Perceptron (MLP)	0.22823	3
02	XGBoost (XGB)	0.19073	4
03	XGBoost, augmented dataset	0.13526	5
04	CATBoost (CAT), unaugmented dataset	0.21329	6
05	CATBoost, augmented dataset	0.14897	7
06	LightGBM (LGB)	0.17297	8
07	Ensemble, MLP with XGB	0.19553	9
08	**Ensemble, XGB with CAT**	**0.12210**	**10**
09	Ensemble, XGB with CAT, LGB	0.13117	11

5.0 Problems Faced and Solutions

5.1 | Difficulties

1. FEATURE ENGINEERING

The raw data provided did not give a good representation of the class to predict. We had to constantly think of higher level features which would create a better correlation to the churn objective.

2. DATA

KKBox provided us with a few versions of the training and test data. We used the most updated version as we assumed that the older versions would be obsolete. Also, many of the training labels were wrongly labelled (https://www.kaggle.com/c/kkbox-churn-prediction-challenge/discussion/45991).

5.2 Solutions

1. FEATURE ENGINEERING

We used a brainstorming chart to help us think creatively and outside of the box to create more higher-level features. We also read through forums and discussion boards for feature ideas, and also came up with a few more. Those mentioned above are the few ones we find them to be useful.

2. DATA

We decided to include all versions of the training data. We merged all the transaction, members, and userlogs data together. We refer to this data as the augmented data. Surprisingly, the older data helped to improve the score.

6.0 Kaggle Result

Our best result was using Ensemble with XGBoost and CATBoost. It achieved a score of 0.12210 (*Figure 12*) and a ranking of 70/575 (top 12%).

Figure 12

7.0 Conclusion

7.1 | Learning Points

We learned that different learning models can have different effects on the churn prediction. It is important to choose the necessary model to get the optimal score and churn prediction.

We learned that it is best to utilize all the training data provided. The more relevant features we can come up with, the more accurate our prediction can be.

We learned that feature engineering is fundamental and plays an important role in churn prediction. Finding the necessary correlations is key to derive the features that we need for a better, more accurate churn prediction.

7.2 | Areas for Improvement

We would like to have more high-level features implemented based on the raw data given. We still think our solution might be lacking in several high-level features to get a better Kaggle score and churn prediction.

We can also try to experiment on more types of learning models in order to find the optimal model for our project to improve our churn prediction. This could be looking into more complex, but efficient models or algorithms that many machine learning scientists use.