NewBees - Group 12

Team Member Name, Kaggle Id's

- Zack Pan, Zack
- Xinke Sun, Xinke Sun
- Nicole Kacirek, Nicole Kacirek

Abstract

In the following paper, we discuss the data, featuring engineering, and machine learning techniques we used to answer the following two questions: 1) Is a user likely to make a purchase in the next *seven days*? and 2) Is a user likely to make a purchase in the next *fourteen days*? For this problem, we used a dataset provided by Leanplum containing information about users' interactions with a single mobile app during the course of three months.

Our first task was to create labels from the Events table that indicated whether a user made a purchase within the last week/two weeks of our training data. Once we had our labels, we began our featuring engineering. The final models used a little over 70 numerical features calculated over specific weeks leading up to the testing period. From there we experimented with different Machine Learning models, keeping the seven-day and fourteen-day problems separate. We assessed the performance (AUC score) of Random Forest, LightGBM, Logistic Regression, and XGBoost models using K-fold Cross Validation. In the end, we blended our two best models, XGBoost and RF with equal weights for 7-day prediction. Similarly, for fourteen-day prediction we blended equally XGBoost, LightGBM, and RF. Our final models gave us an AUC score of 0.98672 on the test values, placing us 11th in the class Kaggle competition.

Description of your dataset and EDA

The data from Leanplum was organized into four different tables: Sessions, Events, Attributes, and Messages. The tables included information about users who used a single app between 2018-10-01 and 2018-12-31. More specifically, 'Sessions' is a

time-series dataset that contains information related to user activities on the app. The second table is ‘Events’ which includes information regarding different types of activities users participated in on the app (including a purchase event). ‘Attributes’ tells attributes information about different sessions. Finally, ‘Messages’ stores information about messages the app sent to a given user.

In order to better understand the four datasets, we applied exploratory data analysis (EDA). By performing initial EDA we hoped to investigate data and discover some interesting patterns. More importantly, we expected to discover some anomalies and test some of our assumptions by summarizing the statistical results of the data. Furthermore, we hoped to visualize characteristics through graphical representations.

After loading the datasets, we first took a look at how many observations were in each dataset, and what the variables’ values looked like for each column. It turns out that ‘Attributes’ had 185,590,092 observations while ‘Messages’ had only 2,896. The ‘Sessions’ seemed to have the most features as it contained 21 columns. Second, we checked if any null values existed in the datasets so we could properly handle those before feature engineering. We found some null values throughout multiple tables, and replaced those with -1.

Next, we compared the attributes between users who made purchases and users who did not. After looking at user information on the seventh and fourteenth days, our aim was to replicate the same pattern when training the models. Another thing to take into consideration was that after EDA we figured out there were more than 30 columns across all four tables. However, many did not seem like they would be very helpful when making predictions about purchases. During our EDA we noted which features would be useful to drop include in our modeling, and which would likely just add noise to the data and increase training time.

Feature Engineering

The two problems (seven-day vs fourteen-day predictions) are very similar to each other, but also quite different. The fourteen-day problem requires two weeks to be held back for testing/validation compared to only one week for the seven day model. Therefore, in order to train with as much data as possible for both problems, we trained two separate models with two different feature timelines:

7 days:

- $Fea_{0-9} + Fea_{8-9} + Fea_{9} -> Purchase_{10}$ (train)
- $Fea_{0-10} + Fea_{9-10} + Fea_{10} -> Purchase_{11}$ (predict)

14 days:

- $Fea_{0-8} + Fea_{7-8} + Fea_{8} -> Purchase_{9-10}$ (train)
- $Fea_{0-10} + Fea_{9-10} + Fea_{10} -> Purchase_{11-12}$ (predict)

The subscripts here represent the week numbers ranging from 0-12 for each of the 13 weeks between October and December.

In the end, we are trying to predict the purchase activity for the 11th and 12th weeks of the data. In both cases we care about a user's *most recent history more than their older-past activity*. We incorporated this by first calculating our features on all of the weeks for a specific validation/prediction period, and then recalculating them on the most recent two weeks, and then one last time on the most recent week.

After trial and error of what features worked best, we ended up with a little over 70 features for both the seven-day and fourteen-day model. Nineteen of the features were shared by both models, while the rest were specific to the above timelines. Other than a few niche indicator features, the rest were numerical (no categories), and many were aggregates by user of particular columns from the dataset tables.

Some of our most important features for the fourteen-day model included the average number of purchases a user made (14buy_mean_tr), the number of unique attribute values a user had (att_val_uniq), and the number of events a user had between weeks 0-8 (user_count_08). As for the seven day model, the most important feature was surprisingly just an indicator for whether a user's attribute value contains a bracket '[]'. Att_val_uniq and user_count_09 also proved to be quite useful for the seven-day model.

See the appendix for a full description of the various feature importances across models.

Machine Learning

When creating our models, we followed five common steps of Machine Learning: 5-Fold Cross Validation, Baseline, Advanced Models test, Hyper-parameter tuning and Ensembling methods.

Random Forest

For the modeling part, we first took RandomForest as our baseline since it handles well between bias and variance. The AUC score on the leaderboard was not bad: 0.924 and the model spent more than 10 minutes for the two rounds of training (one week and two weeks). Looking at the RandomForest feature importances early on really helped to direct our further feature engineering. For example, we found twenty of our earlier features were not very important to model training, so we got rid of them and tested again. After dropping those, our AUC increased to 0.945. This result motivated us to continue giving up parts of less important features when training tree models, thus improving accuracy and training speed at the same time. After hyperparameter tuning, it could achieve above 0.99 in cv.

LightGBM

Compared to XGBoost, LightGBM is really fast but does not perform as well. We used it as an advanced model. However, it performed much worse than RF in terms of predicting seven-day purchases and worked really well on predicting fourteen-day. LightGBM was likely trapped in overfitting while RF generalized the data with bagging lots of trees and hence, outperformed LightGBM. Based on this, LightGBM was not used when modeling purchases in seven days. And after using Random Searing for hyper-parameter Turing, it reached 0.992 in cv during modeling purchases in fourteen days.

Logistic Regression

Regarding binary classification problem, Logistic Regression is always worth a try. We expected it would train fast on our dataset. In fact, it lowered the whole training time to

roughly half an hour per mode. Thankfully, the results were not bad: 0.982 in cv. This proved to be a promising model which we could include into ensemble method. Later given its long training time, we decided to turn to the following one.

XGBoost

This model was so amazing that we had to consider it in two modeling parts. We also first trained it on all features and selected 2/3 of them according to feature importance, to speed up the training. Eventually, it achieved 0.99 and 0.993 in seven-day and fourteen-day model respectively.

Experimental Results - Blending

Finally, by considering performances of our different Machine Learning algorithms, we decide to mix them together to create our final model, expecting that the performance will be further improved by the 'mixture-type' model. Specifically, we blended our two best models, XGBoost and RF with equal weights for seven-day prediction. And for fourteen-day prediction, XGBoost, LightGBM, and RF were also equally blended. Our final models gave us an AUC score of 0.98672 on the test values, placing us 11th in the class Kaggle competition.

Team Responsibilities

- Zack: Featuring engineering, modeling, Machine Learning section of paper
- Xinke: EDA, EDA and Experimental Results sections of Paper
- Nicole: Created labels for training, Abstract and Feature Engineering/Appendix sections of the paper

Team Repo

https://github.com/USF-ML2/final-project-group12

Appendix

I. Ten Most Important Features in RF-14-day model

Feature	Importance Score
14buy_mean_tr	0.154124
att_val_uniq	0.090406
att_ses_ct	0.071853
att_cnt	0.058829
user_pch_ct_08	0.056950
pch_mean_count_rec_mon	0.050188
att_val_cnt	0.046399
prev_ses_du_mean_08	0.035870
pch_mean_date_rec_mon	0.034669
pch_day_repeat_count_rec_mon	0.031092

II. Ten Most Important Features in XGB-14-day model

Feature	Importance Score
14buy_mean_tr	0.252187
user_count_08	0.147230
att_val_uniq	0.123907
du_since_start_08	0.115160
att_val_cnt	0.032070
pch_mean_date_rec_mon	0.032070
act_mean_date_78	0.032070
act_mean_date_8	0.021866
14buy_mean_tr>.8_tr	0.020408
att_cnt	0.017493

III. Ten Most Important Features in LightGBM-14-day model

Feature	Importance Score
14buy_mean_tr	402
att_val_uniq	353
du_since_start_08	341
user_count_08	285
att_cnt	158
user_eve_day_count_08	124
act_mean_date_8	109
14length<3_tr	101
pch_mean_date_rec_mon	89
next_time_max_08	86

IV. Ten Most Important Features in RF-7-day model

Feature	Importance Score
7bracket_tr	0.109999
att_val_uniq	0.091899
pch_mean_count_rec_mon	0.077231
pch_day_repeat_count_rec_mon	0.075702
pch_mean_date_rec_mon	0.067379
att_val_cnt	0.059913
user_pch_ct_09	0.053388
att_cnt	0.045426
att_ses_ct	0.039992
next_time_max_9	0.027080

V. Ten Most Important Features in XGB-7-day model

Feature	Importance Score
7bracket_tr	0.273669
att_val_uniq	0.130178
user_count_09	0.079882
avg_True_ses_89	0.051775
prev_ses_du_mean_09	0.041420
act_mean_date_9	0.036982
att_val_cnt	0.031065
pch_mean_date_rec_mon	0.029586
att_cnt	0.023669
user_pch_ct_9	0.022189