Predict in-app Purchase Probability

• Anna Zeng (Kaggle ID: zztywl92123) • Hai Vu Le (Kaggle ID: haivule) • Aditi Sharma (Kaggle ID: aditisharma04)

Team Aha!

***Abstract*:**

In-app purchase is a piece of content or feature being purchased inside of the app. Whether a user will make an in-app purchase is a problem of high interest to mobile apps in general due to the direct impact on the bottom line. The ability to predict the probability of users purchasing within a certain time window in the future promises various benefits. An example is that marketers could better target their communication to provide purchase incentives to the right group of users at the right time to boost revenue for the app.

The problem we are solving is to predict the probability of in-app purchase for a 7-day and 14-day time window of 312,568 users *of a particular app based on their past behaviors on the app and their receptiveness of marketing communication from the app itself. We use Random Forest and XGBoost algorithms to build predictive models, using features that are aggregated to user-level. The performance of models is assessed using AUC as the performance metric. Our best model achieved a 0.9765 AUC score. This paper discusses the details about the characteristics of the data, the features we extract, and our model training process.*

---

1. Data Description:

We have 2.5 months' worth of data regarding attributes of users (in Attributes dataset), activities users have on the app (in Events and Sessions datasets), as well as how users receive and respond to messages sent by the app. The target prediction is on the user level (whether a user will make a purchase or not). However, the raw data is not on the readily usable granularity. For instance, in *Events* dataset, each row records a specific action a user had in a particular session. Because of this very nature of the raw datasets, there is very little data we could use straight out of the box. Therefore, feature engineering becomes highly important for this problem. Below is an overview of each dataset:

	Attributes	**Events**	**Sessions**	**Messages**
Grain	*User-Session*	*User-Session-Event*	*User-Session*	*Message*
Size	*29.44GB*	*13.78GB*	*1.87Gb*	*162.4 kB*
Entries	*184,880,449*	*111,946,597*	*6,239,836*	*2,897*

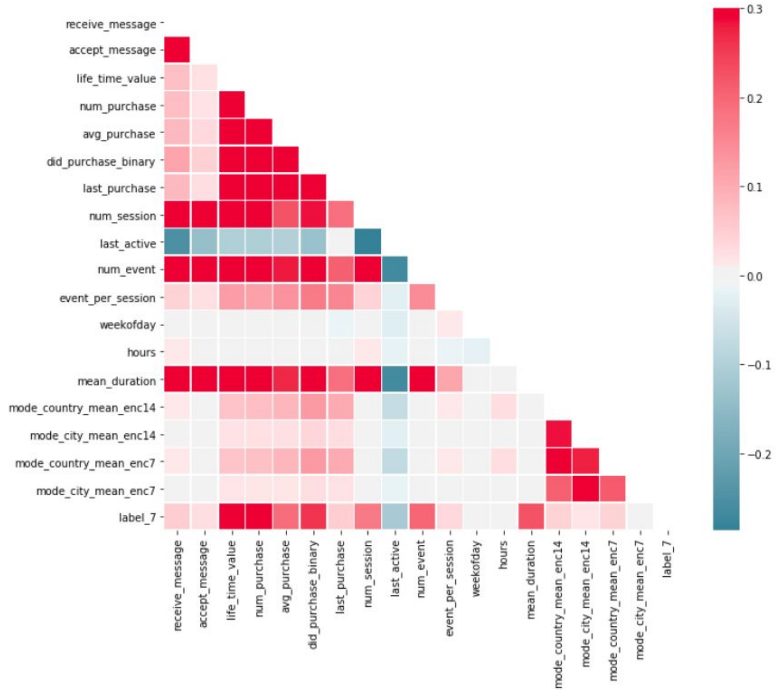
Table 1: Overview of datasets

We use *Attributes* dataset to demonstrate the way to interpret the grain of the data. In this dataset, the grain is *User-Session*, meaning that session ids are unique to each user, and each row in the dataset represents the state of the user at the time of a session, according to attributes defined by the app.

2. Exploratory Data Analysis:

To understand and verify relationships in the data, we conducted exploratory data analysis.

The data has several continuous and categorical variables, describing the customer activity on the app. To facilitate accurate prediction of the purchase made by user in the future, we need to study the existing customer history available to us. The plot on the right shows the correlation matrix of the extracted features with the label. The label taken in the correlation matrix is for 7 days prediction.



From the plot, we can see that features like‘*num_session’*,‘*num_event’,‘mean_duration’,* etc. are highly correlated to the rest of the features, whereas, *‘hours’, ‘weekofday’, ‘mean encodings’* have little to no correlation with rest of the features. Notice the 7 day’s label has high correlation with features like *‘life_time_value’, ‘num_purchase’, ‘mean_duration’*, which means that such features would be highly important while building our model for the prediction. A similar study was done for the purchase within 14 days prediction and the graph and insights can be found in the appendix section.

3. Feature Engineering:

Feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work. Feature engineering is fundamental to the application of machine learning, and is both difficult and expensive.

3.1 Compute target

By filtering out the event history for those whose *‘event’* == ‘8’, which means a purchase behavior, using the Events dataset, we computed the labels for training dataset. The labels we use are binary. As long as a user makes at least a purchase, the label is 1, otherwise 0.

To train models for 7-day prediction, we reserve the last week in our dataset to compute labels and use the rest of the data to derive features. For 14-day prediction models, we use the last 2 weeks to compute labels and the rest of the data for features. When it comes to final prediction (prediction into the future), we use all the data that spans all 2.5 months that we have.

In *events.csv*, we split the dataset by cut_offf:

$$cut\_off\_7 = last\_timestamp - timedelta(days=7) \quad \text{eq. (1)}$$

$$cut\_off\_14 = last\_timestamp - timedelta(days=14) \quad \text{eq. (2)}$$

Figure 1. Data splitting for label computation

Then, for *events.after.last.7.days*, we filter out the unique users who has *'event'* == '8', and label them for - 'having a purchase history in last 7 days'. For *events.after.last.14.days*, we do the same, and label these unique users for 'having a purchase history in last 14 days'. The result is:

Users' group	**Number of unique users**	**Number of purchased users**
Events	*621,001*	*---*
Events in last 7 days	*49,328*	*3,286*
Events in last 14 days	*89,291*	*6,126*

Table 2: Label counts

By joining the results on *'user_id_hash*', we get the *label.csv,* where each user have a column - '*label_7'* indicating having a purchase history in last 7 days, and a column - '*label_14'* indicating having a purchase history in last 14 days.

3.2 Features

As discussed above, the raw data does not have desirable granularity that we need for model training (that is, per session or per event instead of per user). Therefore, the overall approach we take to extract features is to aggregate raw data to the user level, using, the mean, mode, max value, etc.

The underlying hypotheses that motivate our feature engineering are that: (1) app usage behaviors, (2) past purchase behaviors, and (3) response to messages. For instance, it's reasonable to expect highly active users to make more purchases than less engaged ones.

The table below provides an overview of few features we extracted. See appendix for whole table.

Features	**Description**	**Type**
App usage behaviors		
num_session	*Number of sessions*	*numerical*
num_event	*Number of events*	*numerical*
event_per_session	*Number of events*	*numerical*
mean_duration	*Mean duration of sessions*	*numerical*

Features	Description	Type
Past purchase behaviors		
lifetime_value	*How much have purchased during training time per user*	*numerical*
num_purchase	*Total number of purchase made by the user*	*numerical*
Responses to messages		
accept_messagee	*Messages accepted by users*	*numerical*
receive_message	*Number of messages received by user*	*numerical*
Attributes of users		
app_age	*Age of the user on the app. Extracted by using the start_timestamp*	*numerical*
num_device	*Number of device per user*	*numerical*
attribute_[j]_mean	*Mean value of attribute j*	*numerical*
attribute_[j]_max	*Max value of attribute j*	*numerical*

Table 3: Features

4. Machine Learning Method:

Given the dataset is highly imbalanced (0.27% of users have ever made a in-app purchase), stratified sampling is used to ensure the same proportion of purchased users in training and validation set.

As a start, we wanted to choose a model that is not too complex yet promises good performance to gauge a better idea of what features point us to the right direction as well as delivering a good performance baseline. Therefore, we fitted Random Forest, a high performing ensemble algorithms on our 8 initial features, which are *'num_event', 'num_session', 'event_per_session', 'last_active', 'accept_message', 'receive_message', 'num_purchase', 'life_time_value'* and *'avg_purchase'*. The performance of this model is promising. For the validation, we got an AUC score of 0.9579 for 14-day prediction, 0.9680 for 7-day prediction and 0.9669 for the test set on Kaggle.

We then tuned some hyper-parameters of Random Forest, yet the performance did not improve much. So we employed a stronger algorithm, XGBoost on the same set of features. The performance was again, not substantially improved. We realized extracting other features should be next step.

The most important feature indicated by first model was *'last_active'*. Inspired by this finding, we extracted some similar and related features, which are the last date when users had each event, assuming that some events may have a strong correlation with purchase (for instance, checking a promotion). Also, we used mean encoding for each country and extracted other features such as *'app_age'*, the number of devices a user uses the app on, the time when a user is most activate. Finally, we extracted features from the *Attributes* dataset to incorporate more information about users.

The new features enhanced the performance of the model by 2%, from AUC *0.9669* to *0.98335*. An interesting finding is that as we added many more features which could potentially capture very similar

signals (for example, *'last_date_event_5'* and *'last_date_event_1'*), employing overfitting-control measures such as using subsampling, restricting the number of trees or the max-depth of each tree is important to ensure good performance of the model. Without these measures, new features may lead to a poor performance, causing a drop to a 0.85 AUC score.

5. Results and Conclusions:

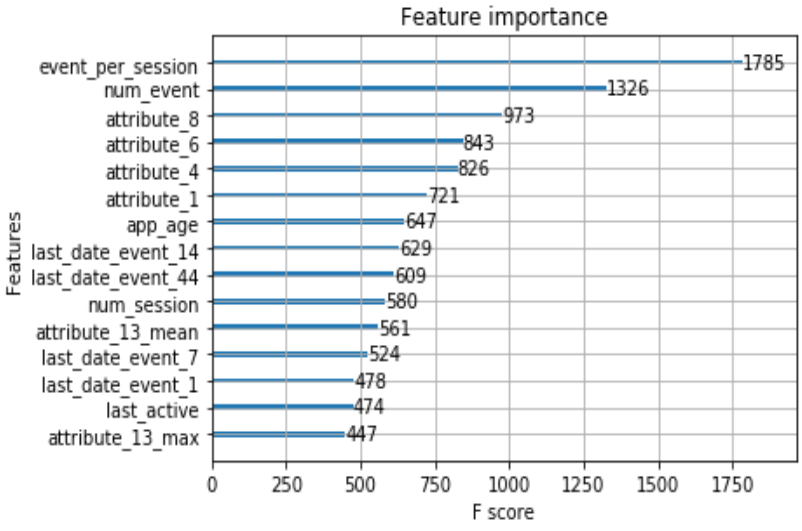
Below are the performance of the first and the latest models that we have:

Model	**AUC - validation 7 days**	**AUC - validation 14 days**	**AUC - test**	**Top 5 Important features**
Random Forest *(max_depth = 8)*	*0.9579*	*0.9680*	*0.9669*	*last_active, life_time_value, num_purchase, avg_purchase, num_session*
XGB *(n_estimators=100, max_depth=8, learning_rate=0.1, subsample=0.5)*	*0.9783*	*0.9715*	*0.9765 (+ 0.99%)*	*Event_per_session, num_event, num_session, last_date_event_14, life_time_value*
XGBoost *(n_estimators=100, max_depth=8, learning_rate=0.1, subsample=0.5)*	*0.9766*	*0.9726*	*0.9833 (+0.68%)*	*Event_per_session, num_event, attribute_8, attribute_6, attribute_4*

Table 4: Model evaluation

Combined with new features, we again fit 2 models to predict the 7 and 14 day in-app purchase probability. We were able to achieve an aggregate AUC of 0.9765 by using extreme gradient boosting model (XGB) with 100 trees and a max_depth of 8.

The graph to the right shows the most important features for our best model. From the graphs, we can see that the most important features are those about the activeness of users on the app or characteristics of themselves. Features related to past purchase behaviors, surprisingly, do not give much hint about the future purchase behaviors.



In conclusion, we can say that the new features we extracted gave us significantly better results.

Future use: The feature engineering, label computing techniques used in here can be generalized to solve similar prediction/classification problems of future activities of an entity.

6. Appendix:

6.1 Personal responsibility

Hai Le:

- Extracted the first batch of features and part of those in the second and the third batch
- Trained the first model and other later versions of Random Forest and XGBoost models

Aditi Sharma:

- Extracted features from sessions and messages
- Trained model on XGboost
- Exploratory data analysis

Anna Zeng:

- Computed target
- Conducted mean encoding on locations, extracted previous session duration time
- Helped to build training dataset and prediction dataset

6.2 Github link

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

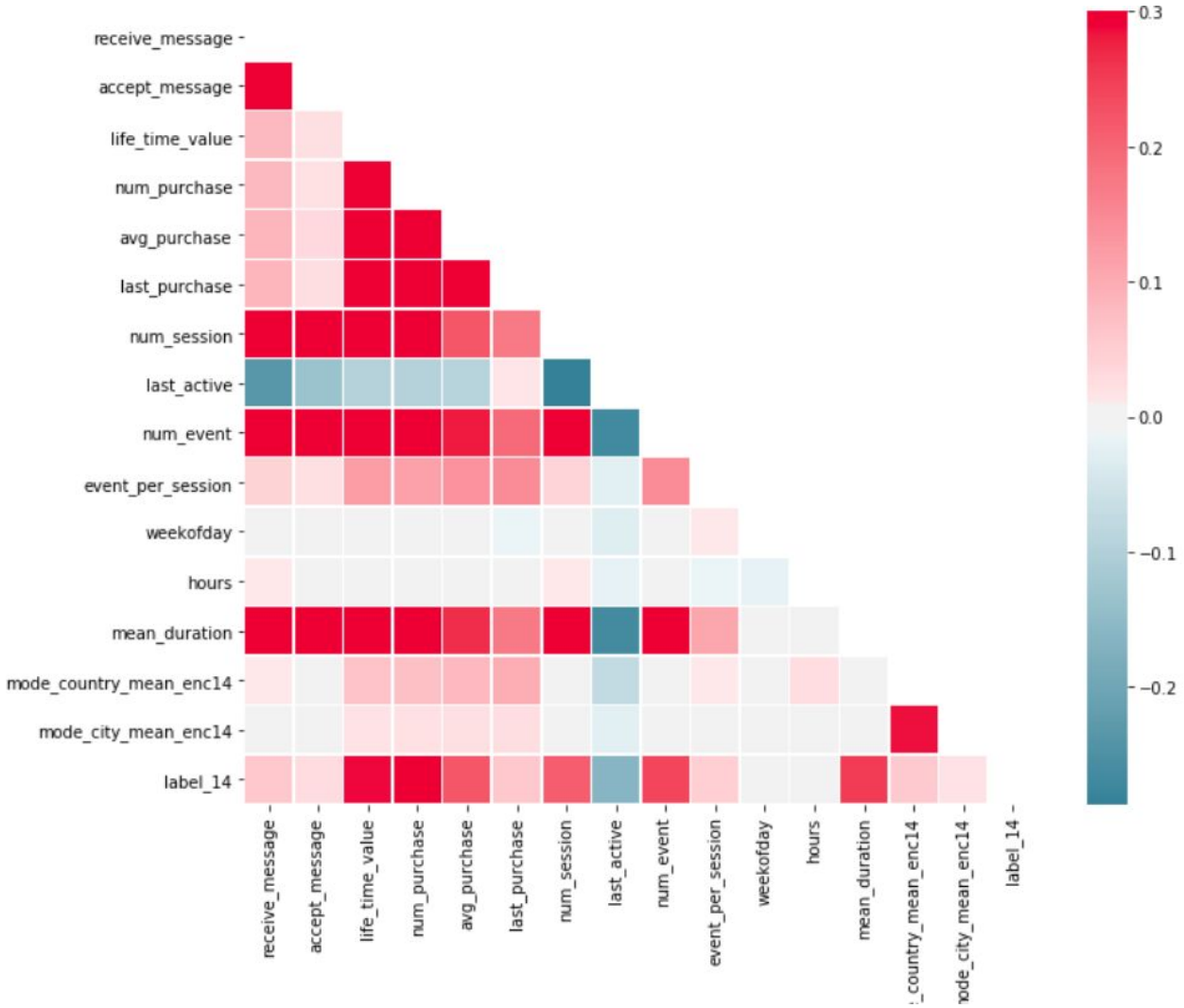
6.3 Detailed description of datasets

- *Message.csv*: Every line contains detail about messages the user received, defined by the app and message id.
    - Columns:[app_id,message_id,action_type,delivery_type,delivery_time_mode, goal_kind].
    - Entries: 2,897.
- *Attributes.csv* : every line represents the state of the user at the time of a session, defined by the app.
    - Columns:[app_id,session_id,attribute,attribute_value,user_id_hash].
    - Entries: *184,880,449.*
- *Events.csv*: every line represents a session that contains user's activities on the app.
    - Columns:[app_id,session_id,event,event_timestamp,event_value, user_id_hash].
    - Entries: *111,946,597.*
- *Sessions.csv*: every line line represents a session that contains the records of the user at the time.
    - Columns:[app_id,is_mau,is_wau,user_id_hash,latitude,longitude, session_id,start_timestamp,timezone,timezone_offset,previous_sessions_duration,user_c

reated_timestamp,country,is_user_first_session,is_session,is_developer,region,city,locale
,os_name,session_index, device_id].

- Entries: *6,239,836.*

6.4 Correlation matrix of features



6.5 Detailed description of features

Features	Description	Type
App usage behaviors		
num_session	*Number of sessions*	*numerical*
num_event	*Number of events*	*numerical*
event_per_session	*Number of events*	*numerical*
mean_duration	*Mean duration of sessions*	*numerical*
last_active	*Number of days passed since the user's last activity (regardless of event)*	*numerical*
last_date_event_[enc]	*Number of days passed since the last time user had an activity with event encoding [enc]*	*numerical*
hours	*Mode of time (military time) when users use the app (i.e. time in the day when users have activities on the app)*	*categorical*
weekofday	*Mode of day of week when users use the app (i.e. have activities on the app)*	*categorical*
Past purchase behaviors		
last_purchase	*Number of days ago when users made the last purchase*	*numerical*
lifetime_value	*How much have purchased during training time per user*	*numerical*
num_purchase	*Total number of purchase made by the user*	*numerical*
avg_purchase	*(Total amount of purchase) / (Total number of purchase)*	*numerical*
mode_country_mean_enc	*Mean target encoding by taking mode of the country per user*	*numerical*
Responses to messages		
accept_messagee	*Messages accepted by users*	*numerical*
receive_message	*Number of messages received by user*	*numerical*
Attributes of users		
app_age	*Age of the user on the app. Extracted by using the start_timestamp*	*numerical*
num_device	*Number of device per user*	*numerical*
attribute_[j]_mean	*Mean value of attribute j*	*numerical*
attribute_[j]_max	*Max value of attribute j*	*numerical*