Growing Smarter: Understanding User Acquisition

Mac Strelioff

null

# Background

As part of my fellowship at Insight Data Science, I worked on a 2 week consulting project with the company Elastic Projects. The overall goal was to understand user conversion on their product, [Abstract](https://www.abstract.com/), and identify users on a free trial who would convert to paying customers. Abstract is a file sharing and versioning application, much like GitHub, with an emphasis on digital artwork. Users can be individuals or organizations, and a user account can assign roles of ‘contributor’ or ‘viewer’ to individual users associated with the account. Contributors have privileges to upload and edit files, while viewers cannot.

# Business Need

The survival of any company hinges on it’s ability to acquire new users. One way that Elastic Projects acquires users is through a 30-day free trial of Abstract. However, around 80% of free-trial users fail to convert to paying customers. This leaves much opportunity to increase their userbase.

## Project Goals

As a consultant for Elastic Projects, I helped with two major goals focused on issuing nudges to users in order to increase user acquisition;

1. Identify, as early as possible, which users were likely to convert.
2. Determine which features to focus on when reaching out to users that were currently unlikely to convert.

# Data sources and processing

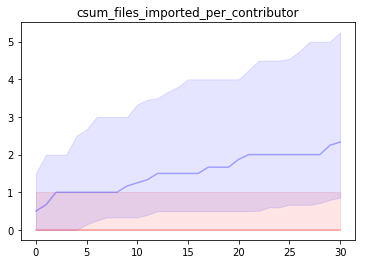
The original dataset consisted of user activity and product performance data for all accounts over a one year period. The dataset contained a large number of users, including some that were not germain to the project goals. Since the primary goal was to identify free-trial users to nudge, I excluded any users who were never on a free trial during the data collection period. In exploratory analysis, I found a large number of users who showed little to no activity. This prompted me to believe there are two types of trial users that do not convert – 1) users who make an account then never engage with the product, and 2) users who engage with the product, fail to find the product valuable, and then decide to stop engagement. Since the strategies to address these users might differ, I decided to exclude anyone without a minimal degree of engagement with the product. The criteria for a minimal level of engagement was chosen based on dependencies between product features, and decided during discussion with Elastic Projects. Finally, some product features were not avilable during the full duration of data collection. Since there was a large sample size, the easiest way to make the analyses pertenant for all features was to exclude data collected before all product features were available.

## Feature Engineering and Exploration

The original user activity data, at an account level, was in terms of counts of an action (e.g. file edits) on each date. To make the users easily comparible to each other across dates, I created a new variable for account age (days since the account was created). Also, rather than working with daily counts, I computed cumulative sums which represented a user’s total usage of a product feature up to a particular age of their account. Finally, since multiple users (contributors) could be associated with an account, and since this is likely to be confounded with other measures of user activity because the counts were aggregated across contributors, I divided the cumulative sums by the number of contributors. After these changes, the features that I used were the total usage of each product feature per contributor up to a particular day since the creation of the account.

The time of conversion could be many months after after a trial had ended, as it may take time for a user associated with an organization to gain approval to purchase a subscription, or it may take time for a large organization to negotiate a price with Elastic Projects. To focus on classifying users as *potential* paying users based on activity, I created a variable that indicated whether an individual ever ended up as a paying customer. I then conceptualized the problem as a classification problem where, based on aaccount activity up to a particular day, I estimated the probability that the account would ever be a paying customer.

I initially thought that the accounts that converted would differ in terms of the distribution of these features (cumulative activity per contributor), relative to those that did not convert. Hence, in exploration, I focused on probing this intuition by plotting the median and to quantiles of cross sections of these features across days since the account was created, split by users who ended up converting versus those who did not. Ideally, there would be features that the converters clearly used more than those who did not convert.



Distribution of cumulative files imported per contributor, split by those who ever paid (blue) and those that were on a free trial forever (red)

# Modeling

## Considerations

My intuition about differing distributions is natrually expressed in [linear discriminant analysis](https://en.wikipedia.org/wiki/Linear_discriminant_analysis) or [quadratic discriminant analysis](https://en.wikipedia.org/wiki/Quadratic_classifier). However, these algorithms hinge on an assumption that the features are Gaussian distributed, which they clearly were not – values were strictly non-negative, and distributions were skewed such that there were lots of values around 0 and some values far from the mean. Because of this violation, I also considered a [support vector classifier](https://en.wikipedia.org/wiki/Support-vector_machine) with radial basis functions, and tree based algorithms – [random forests](https://en.wikipedia.org/wiki/Random_forest), and [gradient boosting decision trees](https://en.wikipedia.org/wiki/Gradient_boosting). An advantage of the tree-based approaches is their robustness to any distribution of the features, and any functional relationship between the features and the probability of a user converting.

Another issue was class imbalance, since around 80% of users did not convert. I used [SMOTE](https://en.wikipedia.org/wiki/Oversampling_and_undersampling_in_data_analysis), an algorithm that generates synthetic data from the underrepresented class (paid users), to address the class imbalance when fitting the model in training sets. I also considered metrics beyond accuracy, such as [AUC](https://en.wikipedia.org/wiki/Receiver_operating_characteristic) and the [F1 score](https://en.wikipedia.org/wiki/F1_score), when selecting a final model.

## Initial Performance

To quickly hone in on a model, I assessed the algorithms mentioned above with default hyperparameter values using [scikit learn](https://scikit-learn.org/stable/index.html) in Python. I focused on classification accuracy in a test set, and the tree-based algorithms outperformed the others by approximately 10%. This outperformance is likely due to the non-standard distributions of the features.

## Hyperparameter Tuning

To assess performance across hyperparameters for the tree-based algorithms, I created [validation curves](https://scikit-learn.org/stable/auto_examples/model_selection/plot_validation_curve.html) with 5-fold cross validation and a set of hyperparameter values. The performance of both tree based algorithms was mostly stable in terms of [F1 scores](https://en.wikipedia.org/wiki/F1_score), except for poor performance when there were very few () estimators. To mitigate potential overfitting, I increased the minimum number of observations in a leaf to 10, but left all other hyperparameters at their default values.

## Fitting Procedure

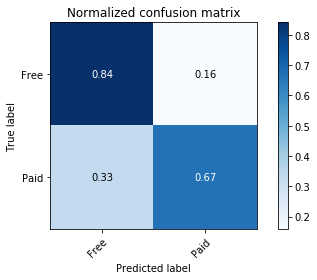
I fit the random forests and gradient boosting trees algorithms to cross-sections of the data at 7, 14, and 30 days since account creation. For each cross section, I split the data into 5 folds, and used [SMOTE](https://en.wikipedia.org/wiki/Oversampling_and_undersampling_in_data_analysis) when training the algorithms within each fold to account for the class imbalance. In each fold, I logged the feature importances and performance metrics from both algorithms.

# Actionable Results

To identify users to nudge, I applied the random forests algorithm using all data and focused on the confusion matrix.

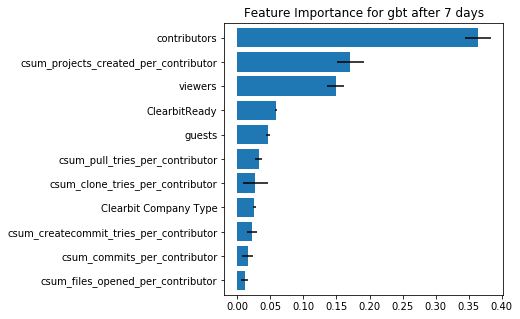
This could help identify who to nudge in the original dataset. The erros that this would make are in the bottom left cell – users who intend to convert to paid users, that we identify as users who would not convert. The effect of this would be targeting onboarding resources toward these users, which may further increase their engagement.

* top left: Users who are on a free trial, that are classified as users unlikely to convert. Can target onboarding efforts toward them.
* top right: Users on free trial, classified as users likely to convert – can target salies efforts toward them.
* bottom left: Users who end up paying that are classified as users unlikely to pay – onboarding resources would be targeted toward them, which has minimal detremental effect, and could further increase their product engagement and hence their likelihood of converting sooner or staying with the product for longer.
* bottom right: Users who end up paying, that the model correctly classifies ad potential paid users – Either sales efforts would be allocated to them, if not currently paid users, or these users would be left alone / not annoyed with onboarding resources or sales efforts.



confusion matrix based on all data, for identifying who to nudge.

Important features from the cross validation procedure described in the previous section point to features to focus on when considering how to nudge users.

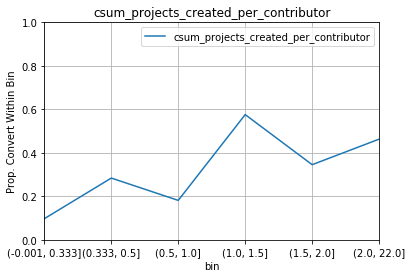


Mean (bar length) and standard deviation (black error bars) of feature importance evaluated across the 5 folds.

To understand the form of the relationship between these features and the proportion of users who converted, I binned users based on product feature usage and plotted the proportion of users across these bins.

* Focus on the important features. For users classified as unlikely to convert, identify which important features they are underutilizing, and target onboarding resources for those features.

For example, if their projects per user is below 1, we could work with the organization to train contributors in project creation.



Proportion of users that convert as a function of projects per contributor. There seems to be a bump in conversion rates from around 20% to around 40% above around 1 project per contributor.

# Summary

I began with an intuition that users who convert would have a different distribution of aggregate product feature usage, relative to those who did not convert, when feature usage was assessed at cross sections based on account age. To explore this, I looked at the quantiles of distributions across feature usage. Focusing on this with formal models, I found that tree-based algorithms could achieve a high accuracy in identifying the users that eventually converted, based on daily snapshots of aggregate feature usage.

To derive insights from this data, I focused on confusion matracies, which identified free users that acted like paid users (good targets for sales teams), and users who were currently unlikely to convert (good targets for onboarding or support teams). I focused on the features found to be important by the tree based algorithms in order to identify specific features to target when reaching out to users that were unlikely to convert. To discover how to nudge on these features, I looked at the proportion of users that converted across levels of engagement with the important product features.

Overall, my work provided valuable tools for identifying users to target with sales and onboarding resources, and found that engagement with core features of the product were more important than engagement with the more compelx features.