# Airbnb's New User Bookings Kaggle Competition

Lisa Oshita

April 5, 2018

## 1 About the Competition

Founded in 2008, Airbnb is an online accommodation marketplace featuring millions of houses, rooms and apartments for rent or lease in over 200 countries. As part of a recruiting competition, Airbnb partnered with Kaggle to host a data science competition starting in November 2015 and ending in February 2016. The task of this competition was to build a model to accurately predict where new Airbnb users will make their first bookings. There are a total of 12 possible destinations to predict: US, France, Canada, United Kingdom, Spain, Italy, Portugal, Netherlands, Germany, Australia, Other and no destination found. No destination found indicates that the user did not make a booking, while Other indicates that a booking was made to a country not already listed. For this competition, Airbnb provided six data sets for participants to use in model building. I used two of these data sets: train_users, which contains information about the users, including where they first booked, and sessions, which contains information about user's web session activity.

## 2 XGBoost, Random Forest, Stacked Models

Extreme Gradient Boosting (XGBoost) is a fairly new method of supervised learning that performs consistently better than single-algorithm models. It is a form of gradient boosting that introduces a different, more formal form of regularization to prevent overfitting—enabling it to outperform other models. Additionally, XGBoost algorithms are fast. At its core, the algorithm is parallelizable which allows this model to fully utilize the power of computers. As XGBoosts have been used to place highly in Kaggle competitions and were also used by many of the top 100 participants of this Airbnb competition, this was the model I first decided to explore and implement.

The second model I decided to implement was a random forest. Random forests are another form of ensemble modeling that have performed well in Kaggle competitions.

As the top three winners of this Airbnb competition used a form of stacked modeling, I also decided to explore a general form of this. Stacking, also called meta ensembling, is a technique used to combine information from individual

predictive models to create a new model. Stacked models usually outperform its base models—as it's able to correct and build upon the performance of those base models.

# 3 Exploratory Analysis

imbalanced classes ages gender number of missing values

# 4 Feature Engineering

From the train_users data I created a total of ? features. I pulled apart the month, year and day of the week of the date features (date account created, date first active, date of first booking) and created a season variable (winter, spring, summer, fall). I also calculated the difference in days between each date feature: days between the date an account was created and the date of first booking, and days between date first active and date of first booking. I also cleaned the age and gender features.

From the sessions data I created a total of blah features. The data contained five features describing userâĂŹs web session activity: action (193 levels), action_type (9 levels), action_detail (93 levels), device_type (13 levels), and secs_elapsed. I aggregated the data by each user and created features counting the number of unique levels for each of these features. From secs_elapsed, I also calculated summary statistics like the mean and median for each unique user. These features were then joined by user id to the training data.

The following is a list of all features included in the training data:

Features in the original data

- signup_method
- signup_flow
- language
- affiliate_channel
- affiliate_provider
- first_affiliate_tracked
- signup_app
- first_device_type
- first_browser

Features derived or cleaned

- Year, day of the week, month and season of date account created, date first active, date first booking

- days between date account created and date of first booking
- days between date first active and date of first booking
- age
- gender
- count features created from the sessions data
- mean, median, standard deviation, minimum, maximum of secs_elapsed

One-hot encoding was used to convert categorical features to a form that works better with machine learning algorithms. Essentially, a boolean column indicating a 1 or 0 was generated for each level of the categorical feature. Continuous features were left as is. After one-hot encoding and feature engineering, there were a total of 596 features to use in the models.

# 5 Model Building and Results

The processes used for building the three models (XGBoost, random forest, and stacked models) was not linear. It was an iterative process in that, when new findings were discovered or new features were added in, I went back a few steps to make changes. But overall, the same process was used for all three models.

The full data was partitioned into one training set, containing 70% of the full data (149,422 rows), and one test set containing 64,029 rows. All model building was performed on just the training set. For both the XGBoost and random forest models, five fold cross-validation was performed on the training set, including all 596 features. Both models achieved 87% classification accuracy in this process, as well as a Normalized Discounted cumulative gain score of 0.92. However, both models only made predictions for a few out of the 12 possible countries. Predictions made included just the US, Other, and no destination found. Examining feature importance for each model showed that not all features were contributing to the predictions. So, the top 200 most important features for each model were extracted. The models were then refit again to the same data, but only with those top 200 features, and five fold cross-validation was again performed. Accuracy and NDCG scores remained the same, but computational time was much faster. These top 200 features were the only features considered from this point forward.

To account for the highly imbalanced classes, various techniques were explored—regular oversampling with replacement combined with undersampling from the overrepresented classes, before settling on synthetic minority oversampling techniques (SMOTE). With SMOTE, underrepresented classes are upsampled by generating synthetic examples by selecting neighbors from the k-nearest neighbors of the minority classes. This was combined with an undersampling of the majority class. The resulting training set contained 115,137 observations. Table 5 shows the number of observations and proportion for each country after SMOTE and undersampling was performed.

| Country | N | Proportion |
|---|---|---|
| AU | 5670 | 0.05 |
| CA | 5000 | 0.04 |
| DE | 5944 | 0.05 |
| ES | 6300 | 0.05 |
| FR | 7034 | 0.06 |
| GB | 6508 | 0.06 |
| IT | 5955 | 0.05 |
| NDF | 35000 | 0.30 |
| NL | 5340 | 0.05 |
| other | 7066 | 0.06 |
| PT | 5320 | 0.05 |
| US | 20000 | 0.17 |

The same model fitting process (cross-validation with only the top 200 features) was again performed on this new training data. For both models, accuracy and NDCG scores remained the same. However, with this new training data, the models were now able to make predictions for all countries instead of just a few. Although the predictions for these minority countries were not significantly accurate. Tables blah and blah show the confusion matrices for the XGBoost and random forest fit to this new training data.

After individually exploring the random forest and XGBoost models, they were combined into one stacked model according to this guideline.

# 6 Discussion and Conclusion

# 7 References