Airbnb's New User Bookings Kaggle Competition

Lisa Oshita

April 7, 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. Participants were given access to six data sets containing information about the users and the 12 possible destinations to predict: Australia, Canada, France, Germany, Italy, Netherlands, Portugal, Spain, United Kingdom, US, other and no destination found (NDF). No destination found indicates that a user did not make a booking. Other indicates that a booking was made to a country not listed. This paper describes the process used to develop several models to predict booking destinations, with the goal of exploring and implementing machine learning concepts.

2 XGBoost, Random Forest, Stacked Models

This paper explores three different models: Extreme gradient boosting (XGBoost), random forests, and stacked models.

XGBoost algorithms are a fairly new method of supervised ensemble learning that performs consistently better than single-algorithm models. It is a form of gradient boosting that introduces a different, more formal method of regularization to prevent overfitting—enabling it to outperform other models. Additionally, XGBoost algorithms are parallelizable, allowing it to fully utilize the power of computers, which effectively decreases computation time. 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 first model implemented.

Random forest models are another form of ensemble modeling frequently used in Kaggle competitions.

Stacking, also called meta ensembling, is a technique used to combine information from individual predictive models to create a new model. Because stacked models are able to correct and build upon the performance of those base models, they usually achieve better results. This technique is also frequently used in Kaggle competitions and was used by many of the top participants of this competition.

3 Exploratory Analysis

Of the six available data sets, two were utilized: train_users and sessions. The sessions data, with 1,048,575 rows and 12,994 unique users, contained information regarding user's web session activity. To minimize computation time, 10% of rows from each unique user was randomly sampled. This new sampled data, containing 104,424 rows, was used for all following analyses. The train_users data, with 213,415 rows and 16 original columns, contained user information starting in 2010. Exploratory analysis of this data revealed several factors.

Destination	**Percentage of Bookings (%)**
NDF	58.35
US	29.22
other	4.73
FR	2.35
IT	1.33
GB	1.09
ES	1.05
CA	0.67
DE	0.50
NL	0.36
AU	0.25
PT	0.10

Table 1: Percentage of Bookings Made to Each Destination

Table 1 shows how imbalanced the target destinations. Together, NDF and US account for almost 90% of the entire data, indicating that a majority of users either have yet to make a booking, or book to locations in the US. The remaining 10, under-represented destinations make up less than 5% of the data each. The extremity of this imbalance will present challenges for the machine learning algorithms explored in this paper.

Figure 1 and 2 depict the trends in the date features of the train_users data. Figure 1 shows the increase number of accounts created each year. Figure 2 depicts the trend in the number of bookings made from 2011 to 2015. Interestingly, there seems to be a seaonality in the data. The number of bookings made seems to increase around September of each year, and then decrease.

Figures 3 and 4 depict information about the users. Note that these figures reflect the age and gender features of the train_users data after missing and erroneous values were recoded and cleaned. Figure 3 shows that a majority of users fall between 25 and 50 years old. Median age is 36 years old. Figure 4

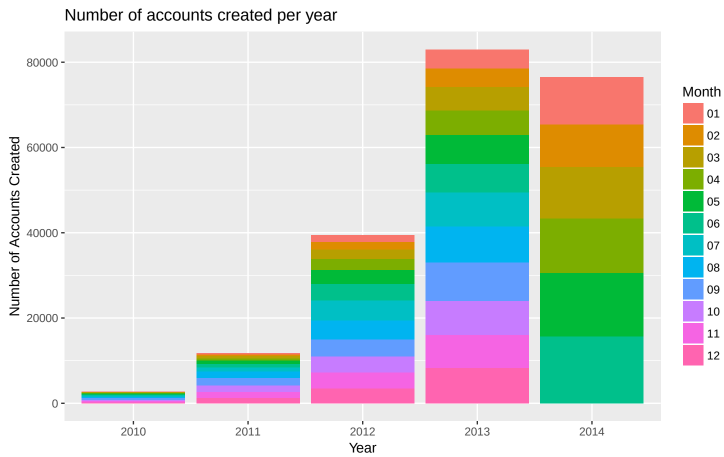

Figure 1: Number of accounts created each year (data only up to June 2014)

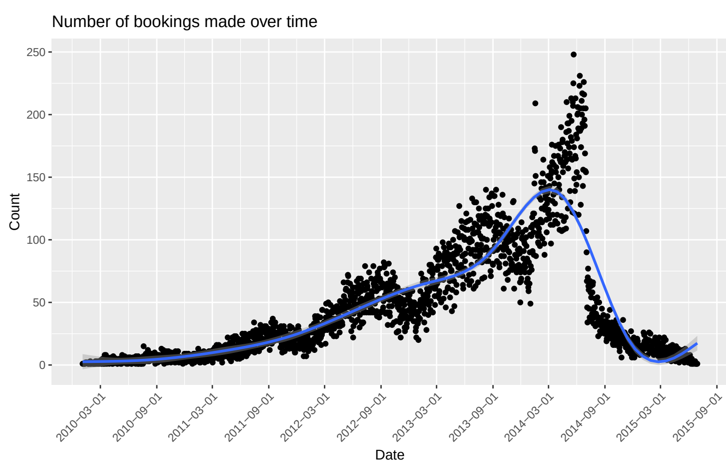

Figure 2: Number of bookings made over time

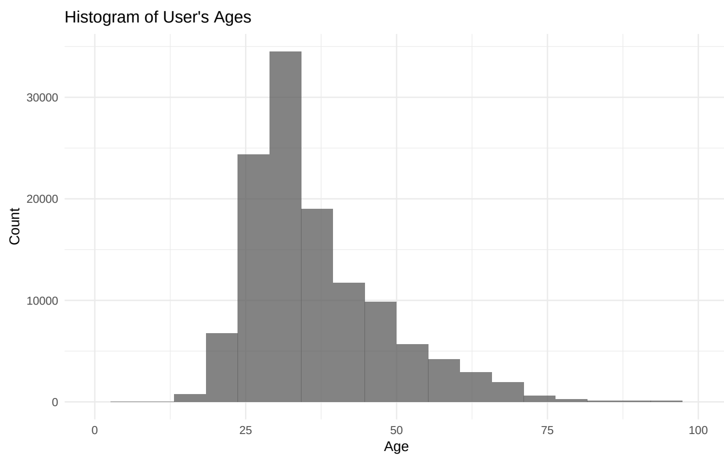

Figure 3: Distribution of user ages

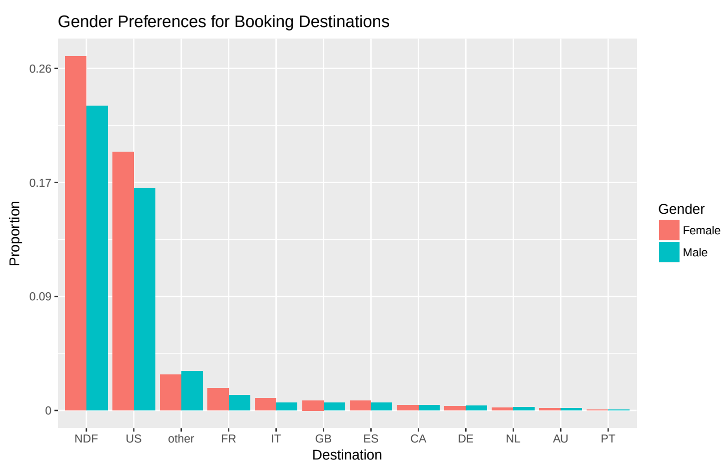

Figure 4: Gender preferences for booking destinations

shows that there are more female users than male. There does not seem to be any obvious preference in booking destination.

4 Feature Engineering

From the train_users data, 17 features were created. Date features were pulled apart into month, year, day of the week, and season features. The differences in days between date features (days between the date an account was created and the date of first booking, and days between the date a user was first active and the date of first booking) were also created. Missing and erroneous values in the gender feature were cleaned and recoded. The age feature contained several issues. 87,990 users (41% of users) did not enter an age, while 2,345 users (1% of users) entered an age of greater than 100. These values were recoded, while missing values were imputed using an XGBoost to predict those ages based on ages available in the data.

From the sessions data, 320 features were created. The data was aggregated by each user and features counting the number of unique levels of each categorical feature were created. From secs_elapsed, mean, median, standard deviation, minimum and maximums were calculated for each unique user. These features were then joined by user ID to the train_users data.

Original Features	Sign up method (facebook, basic, google)
	Sign up flow (17 levels)
	Language preference (25 levels)
	Affiliate channel (8 levels: direct, remarketing, api...)
	Affiliate provider (18 levels: craigslist, bing, email-marketing...)
	First affiliate tracked (8 levels: untracked, linked, local ops...)
	Sign up app (Web, Moweb, iOS, Android)
	First device type (9 levels: Mac Desktop, iPad, iPhone...)
	First browser (52 levels: Chrome, Firefox, Safari...)
Derived Features	Year, month, day of the week, season features 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 (314 features: number of times a user viewed recent reservations, number of times a user viewed similar listings...)
	Mean, median, standard deviation, minimum, maximum of seconds elapsed for each user's web activity

Table 2: Original and Derived Features

One-hot encoding was used to convert categorical features to a form compatible with machine learning algorithms. Essentially, a boolean column was generated for each level of a categorical feature. Continuous features were left as is. After one-hot encoding, there were a total of 596 features. Table 2 displays all features, both original and derived, that were included in the training data for model building.

5 Model Building and Results

Although the methods used for model building were circular and iterative processes, rather than linear. the following describes the general process used to develop the XGBoost, random forest, and stacked model.

The full data was partitioned into one training set, containing 149,422 rows (70% of the full data), 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, as well as a Normalized Discounted Cumulative Gain (NDCG) score (the metric used in the actual competition) of 0.92. However, both models only made predictions for NDF, US, and other. Examining feature importance for each model showed that not all features were necessary. So, the models were refit to just the 200 most important features and cross-validation was performed again. While accuracy and NDCG scores remained the same, computation time decreased tremendously. All following models were built only on just those 200 features.

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

To account for the highly imbalanced classes, several over-sampling techniques were explored. Data for under-represented destinations were over sampled with replacement and data from over-represented destinations were under sampled. Building and testing models on this new sampled training data resulted in worse predictive accuracy than previous models. The final method settled on was Synthetic Minority Oversampling Techniques (SMOTE), in which underrepresented classes are upsampled by with generated synthetic examples selected from the k-nearest neighbors of these under-represented destinations. This over-sampling technique was combined with undersampling of the over-represented destinations. The resulting training set contained 115,137 observations. Table **??** shows the number of observations and proportion of data each destination accounts for, after SMOTE and undersampling was performed.

The same model fitting processes (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, both models were able to make predictions for all countries (rather than just NDF, US and other), although predictions for these under-represented countries were not significantly accurate. Figures blah and blah show the confusion matrices of predictions for the XGBoost and random forest fit to this new training data. Figures blah and blah display the feature importance for each model.

Both models were then combined in a stacked model. The training data was partitioned into five folds, each fold making up around 20% of the full training data. Each model was then built on the training folds, and tested on the held out fold. For example, the XGBoost was first built on folds one, two, three and four, and used to make predictions on fold five. In the second iteration, the XGBoost was build on folds two, three, four and five, and used to make predictions on fold one. This process was repeated for both models until each fold had been used as a test fold. Predictions from these models were stored as two columns in the training data. Then, each model was fit to the full training data (ignoring the folds and predictions created in the previous step), and used to predict on the held out test set. Predictions from these models were stored in two columns in that test set. A final XGBoost was used as the model to combine, or stack the information from the previous processes. An XGBoost was fit to the predictions stored in the training set and tested on the predictions stored in the test set. Resulting accuracy and NDCG score for this stacked model was the same as its base models, as well as all previous models.

6 Discussion and Conclusion

Although various strategies for improving model performance were explored, no effective strategy was found—each model achieved around 87% accuracy and NDCG score of about 0.92. One explanation for this is how imbalanced the target classes are, as described above. Essentially, there is not enough information available about the under-represented destinations to enable models to make predictions for those underrepresented countries. Instead, each model classified them mainly as bookings to the US. Since NDF and US constitute such a large portion of the data, the accuracy of each model remained fairly high at 87%. In order to create a model that can make accurate predictions for all 12 possible destinations, a strategy to resolve this issue of highly imbalanced classes would need to be implemented.

To explain why the stacked model did not result in better performance, the base models need to be examined. Stacked models generally perform well when its base models differ in performance. For example, suppose base model 1 could predict five countries well, but not the other 7. Suppose the opposite was true for base model 2: it could predict those 7 countries well, but not the other 5. In this case, a stacked model would be useful because of its ability to build upon and correct the base models' performances. But since both the random forest and XGBoost could only make accurate predictions for NDF and the US, the stacked model's performance was not a significant improvement.

There are several possibilties for continued work on this project. One idea is to stack more than just two models, as the top three participants in this competition combined numerous algorithms in multiple layers. Another possible strategy, used by the second place participant of this competition, is to build additional models to predict and impute missing values for certain features. Missing values were a significant issue with several features, like age and gender. Building models to address this may be helpful with model performance. Another strategy is to find a way to incorporate the other two available data sets: countries, which contains summary statistics about the destinations, and age_gender_bkts, which contains statistics describing the users' age groups, gender, and destinations. The second place winner built models to predict features within the countries data, and use those features in the final destination predictions. Parameter tuning for each model, to optimize the way each model builds on the data, may also be helpful with model performance.

7 References