

To Book or Not to Book?

Customer classification using Airbnb Data

DS5110: Big Data Analysis (Autumn 2022) – Final Project Report

Alex Bass (ujb3bu), Kaia Lindberg (pkx2ec), Elina Ribakova (uvg5bn)

Abstract

Airbnb has over 150 million users, but how can we classify users on whether they will make a booking? Predicting which type of customer is likely to book a stay after entering the Airbnb website can help target advertising, improve user experience, and hence Airbnb performance. We use Airbnb booking data on Kaggle with user information, including demographics, web session records, and summary statistics. We apply logistic regression (lasso and ridge), Naive Bayes, Random Forest, and Gradient Boosted Tree for our binary (booked/not booked) classification. To tune parameters, we use cross-validation. We find that Gradient Boosted Tree gives us the best results with an AUC of 0.739, but overall, the models produced similar results. Our sample is balanced so we did not need to do an adjustment for that. In addition, we also looked at F1 score, Precision, Recall, the true positive rate, the true negative rate, and the false positive rate. We conclude that it is possible to predict the booking with several binary classification modes. For future work, we would like to see if session log data substantially improves models based solely on user demographics. It would also be interesting to collect more data on session logs, including past search activity, for feature engineering.

Introduction

In this paper, we investigate a binary classification problem to predict the likelihood that a user will make a booking on Airbnb. Since one of the 12 possible outcomes is 'NDF' (no destination found), meaning that the user did not make a booking, we could explore a two-class problem of whether they made a booking rather than predicting which country. This binary classification problem could also be valuable for a company like Airbnb because they could know which users are most likely to book and/or the borderline users that may be more easily influenced into making a booking. This could help improve user experience by reducing search time, but also improve Airbnb's business results by decreasing the likelihood of new users booking a trip and/or decreasing the time between a user creating an account and booking their first place. All our coding files are saved [here](#).

Data Description

Airbnb booking data on [Kaggle](#) with information on users, including data on demographics, their web session records, and summary statistics. The sum of the file size between all 6 data files is 67.85 MB. There are a few different tables, with the main training and test sets containing features such as age,

gender, timestamps, digital data, and booking data. There are additional tables, including information about the user's web sessions, such as the actions taken, the device used, and the time elapsed. We appended the tables to create a coherent dataset. All data is for the users that are from the U.S. The original data was split into training and test datasets, but the test dataset did not have a response variable. The sessions' data goes back to 2014, and the users' data goes back to 2010.

Methods

Data import and preprocessing

We first imported the two datasets, `train_users.csv`, and `sessions.csv`, then combined the two datasets and did feature engineering and exploratory data analysis.

In the `train_users` dataset, there were 213,451 unique rows and unique IDs for the users. The dataset has 15 variables that we considered using as features and a response variable for the country of booking. The dataset had information about individual users (IDs), including demographics, age, gender, account history, sign-up method, and others.

We got information about web session logs from the sessions dataset for AirBnB users. The sessions dataset had 10,567,737 but 10,315,201 distinct rows, which is not surprising given that a user could do the same actions multiple times. However, it is important to note that the `train_users` data had more unique IDs than the sessions dataset (213,451 versus 135,483), so we did not have session data for all users. Below is an example of sessions' data for a user "d1mm9tcy42:"

User Id	Action	Action Type	Action Detail	Device Type	Time Elapsed
d1mm9tcy42	search_results	click	view_search_results	Windows Desktop	67,753
d1mm9tcy42	personalize	data	wishlist_content_update	Windows Desktop	831
d1mm9tcy42	ask_question	submit	contact_host	Windows Desktop	882
d1mm9tcy42	similar_listings	data	similar_listings	Windows Desktop	255

Combining the datasets

We combine the `train_users` and sessions dataset on the left join based on the distinct ID count. We ended up with the combined dataset for 213,451 unique AirBnB users with demographics and data documenting their interaction with the Airbnb site.

Preprocessing data

We then proceeded with the pre-processing of the data; we grouped infrequent categories, aggregated session logs to the user level, imputed missing values, and also did scaling before proceeding with the analysis. We had the most significant number of missing values in the age category, where about 40% of the data was missing, and we imputed the data. We also imputed data where a smaller percentage of values were missing. We also noted that the Age category had some illogical values; for example, 2014, so we cut off observations with ages less than 16 and over 100. The first affiliate tracked some missing values, but it was limited. Other features had nearly no missing values. For gender, we also noticed a large number of "unknown," which we decided to keep, as it might be a helpful indicator of whether a

user cares to fill out her profile. After the original pre-processing (cleaning of the data and imputing missing values), we proceeded with one-hot encoding and feature engineering as described below.

gender	Count	Percent
-unknown-	95688	44.83
FEMALE	63041	29.53
MALE	54440	25.5
OTHER	282	0.13


Feature Engineering

In addition to the preprocessing and cleaning the data, we created new user-level features, including the total elapsed time the user spent on Airbnb, the number of actions the user performed, number of unique actions. We also decided to add a new column, whether age was missing or not, thinking that an incomplete profile might mean the user might be less likely to make a booking.

Outcome variable

We had information on the country destination, but we were more interested in whether the user would make a booking based on our data. There were about 12 different destinations, with the most common “destination” of NDF—no destination found—meaning that the user did not book a place. The second most common destination was the U.S. To classify our users into those that would book or not, we transformed country_destination into a binary response variable with “yes” and “no” distributed as follows:

country_destination	Count	Percent
NL	762	0.36
PT	217	0.1
AU	539	0.25
CA	1428	0.67
GB	2324	1.09
other	10094	4.73
DE	1061	0.5
ES	2249	1.05
US	62376	29.22
FR	5023	2.35
NDF	124543	58.35
IT	2835	1.33



booked	Count	Percent
No	124543	58.35
Yes	88908	41.65

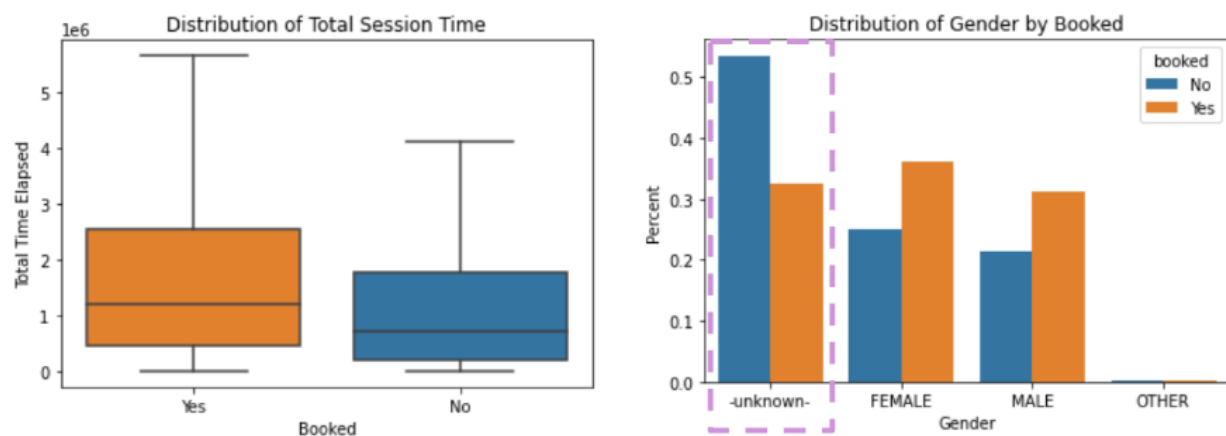
Data splitting and sampling

While the dataset itself had a split into a train and validation data for user demographics features, the users test part did not have the label. Furthermore, we wanted to join the sessions table documenting

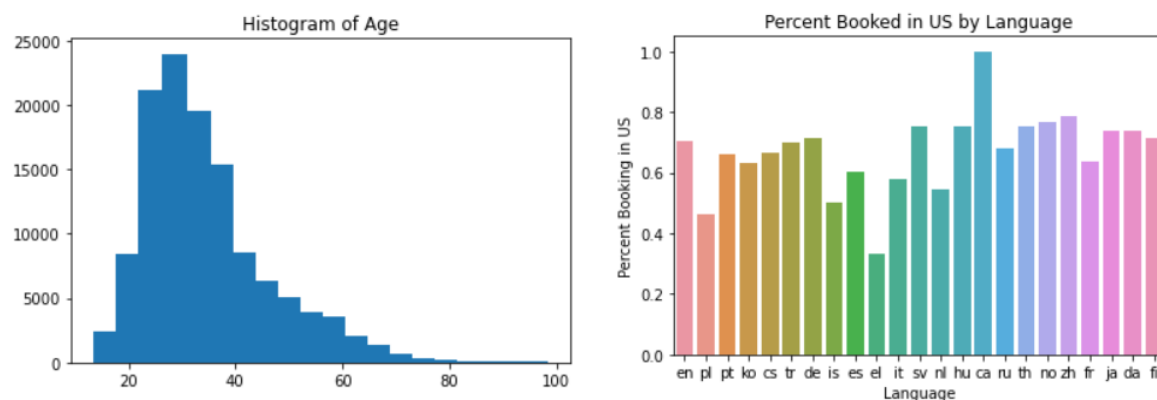
web session logs to our feature set. As a result, we randomly split the complete dataset into test and validation sets (70% train and 30% test).

Exploratory data analysis

We are interested in which features are most likely to be associated with a booking with Airbnb and also if there are interesting particularities in the data, we plotted the features by themselves and also separately for the users that booked and did not book a stay. For example, users that booked, tended to have higher elapsed time than those that didn't. In addition, users with more complete profiles, for example on gender, appeared to be more likely to book.



In addition, we plotted the variables that we thought might be interesting for our analysis, including the new age variable, after the cleaning and imputing, as well as languages, to see if it made a difference in bookings.

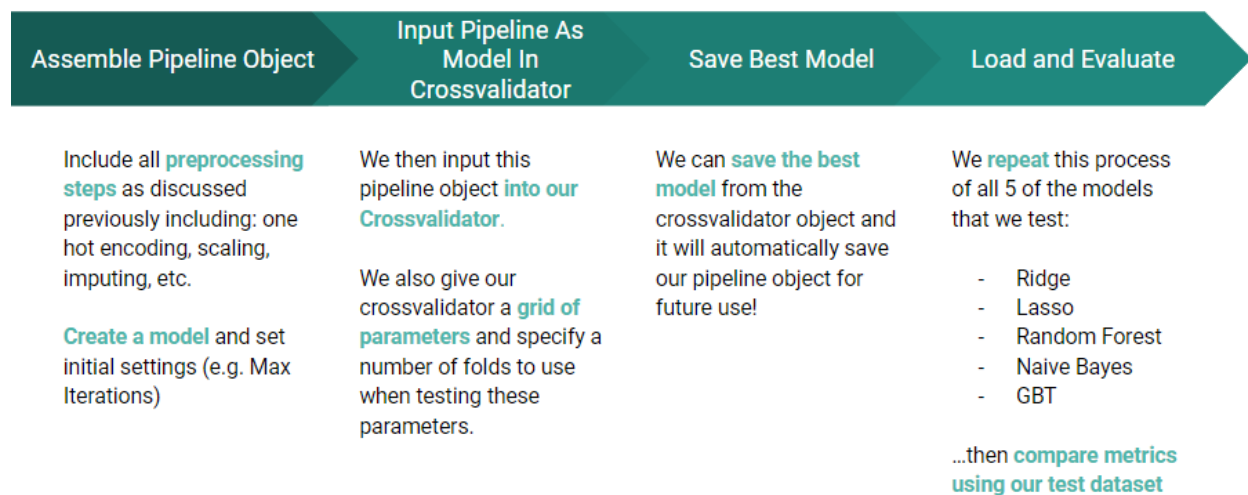


Model construction

In order to predict whether a certain type of user is more likely to book or not book an Airbnb, we selected Ridge Regression, Lasso Regression, Naive Bayes, Random Forest, and Gradient Boosting Trees. We used the same features for each of the models. To facilitate the analysis, we created a pipeline that included all the preprocessing, feature selection, model creation, cross-validation and then model output based on the test data. We used the same set of variables for all the models.

Cross-validation was an important tool for us to decide on the best parameters for our models. We first created a pipeline object with steps and then built a parameter grid. We then input the pipeline to the crossvalidator. Finally, we saved the final best models and the metrics to be evaluated and compared.

Pipeline Object + Crossvalidator



Variable selection

Following exploratory data analysis, we selected the variables that we thought would help us the most to classify the users and complemented them with the features that we engineered:

Age: since age had the highest percentage of missing values and also had improbable values, we imputed age and also cleaned out unlikely (ex. over 100) ages creating **age_new_imputed**. In addition, since not having a complete profile might be an important indicator of whether a user is likely to book, we added an additional binary feature, where age was missing or not **age_missing**.

Gender: gender itself might be an important indicator, however, not having a complete profile (gender “unknown”) just like with age, might be an important indicator. So we use gender_vec with female, male, unknown, and other in our models.

Language: there were over 20 languages spoken by Airbnb users, however English accounted for over 96%, we used one-hot encoding to transform the variable for our models. There were no missing records. We reduced the number to only 8 most frequently used languages.

Browser Used: was typically “basic” (70%), followed by facebook (28%), and google. We used one-hot encoding to see if it might matter for the likelihood of booking. **App used to sign up:** here again we used one-hot encoding to figure out the sign up application used, whether it was web (over 85%), iOS, Android, etc. **Laptop .vs Tablet vs. Phone:** here again, we used one-hot encoding to include data on the device used, including Mac Desktop (over 40%), Windows Desktop (over 34%) and others like an Iphone or Android phone, tablet, etc.

Time Spent Browsing: We observed significant difference in time a user spent on Airbnb, with users spending more time appearing to be more likely to make a booking (see EDA discussion above). We imputed a small number of missing values.

Total Online Actions: There appeared to be higher likelihood of a booking based on the higher number of online actions that we discovered during EDA, as a result, we including this feature imputing missing variables.

All features were scaled before being passed to the models. For all the models we used the pipeline as described below.

Logistic regression

Using all the variables described above, we first ran independently ridge and logistic regression by controlling the elasticNetParameter (Lasso =1 and Ridge = 0) within the `pyspark.ml.classification.LogisticRegression` package. The run time was very brief (train time of 5.3). For both models, we used maximum iterations at 10.

Random Forest (RF)

Similarly, we used all the variables to do random forest model using `pyspark.ml.classification.RandomForestClassifier` package. The model took somewhat longer to run, but was still very manageable (train time: 116.4). We used `CrossValidator` to estimate the best model, we used a grid for the number of trees (5, 20, 50) and max depth (3, 5, 5). We used three folds for cross-validation.

Naive Bayes (NB)

We then did naive bayes model using `NaiveBayes` module from `pyspark.ml.classification` package. We used `CrossValidator` and a parameter grid for tuning the model (with the grid of 0, 0.5, 1, 5). Similarly to the other models, we used three folds for cross-validation. The run time was 36.5.

Gradient Boosting Trees (GBT)

Finally, we did the binary gradient boosted tree classifier using the `GBTClassifier` module from the `pyspark.ml.classification` package. We did a grid search for max depth (3, 5, 6) and weight fraction per node parameters (0, 0.01, and 0.1). We used cross validation with three folds. The run time was 120.

Model evaluation

Using the same set of variables, we evaluated our 5 models using metrics on test accuracy, AUC, F1 score, Precision, Recall, the true positive rate (TPR or sensitivity = $TP/(TP+FN)$), the true negative rate (also called specificity = $TN/(TN+FP)$, and the false positive rate ($FPR = FP/(FP+TN)$). After training the models on training data and doing parameter tuning with cross validation, we evaluated our models on text data not seen by the models. We used the best saved models from cross validation from RF, NB, and GBT.

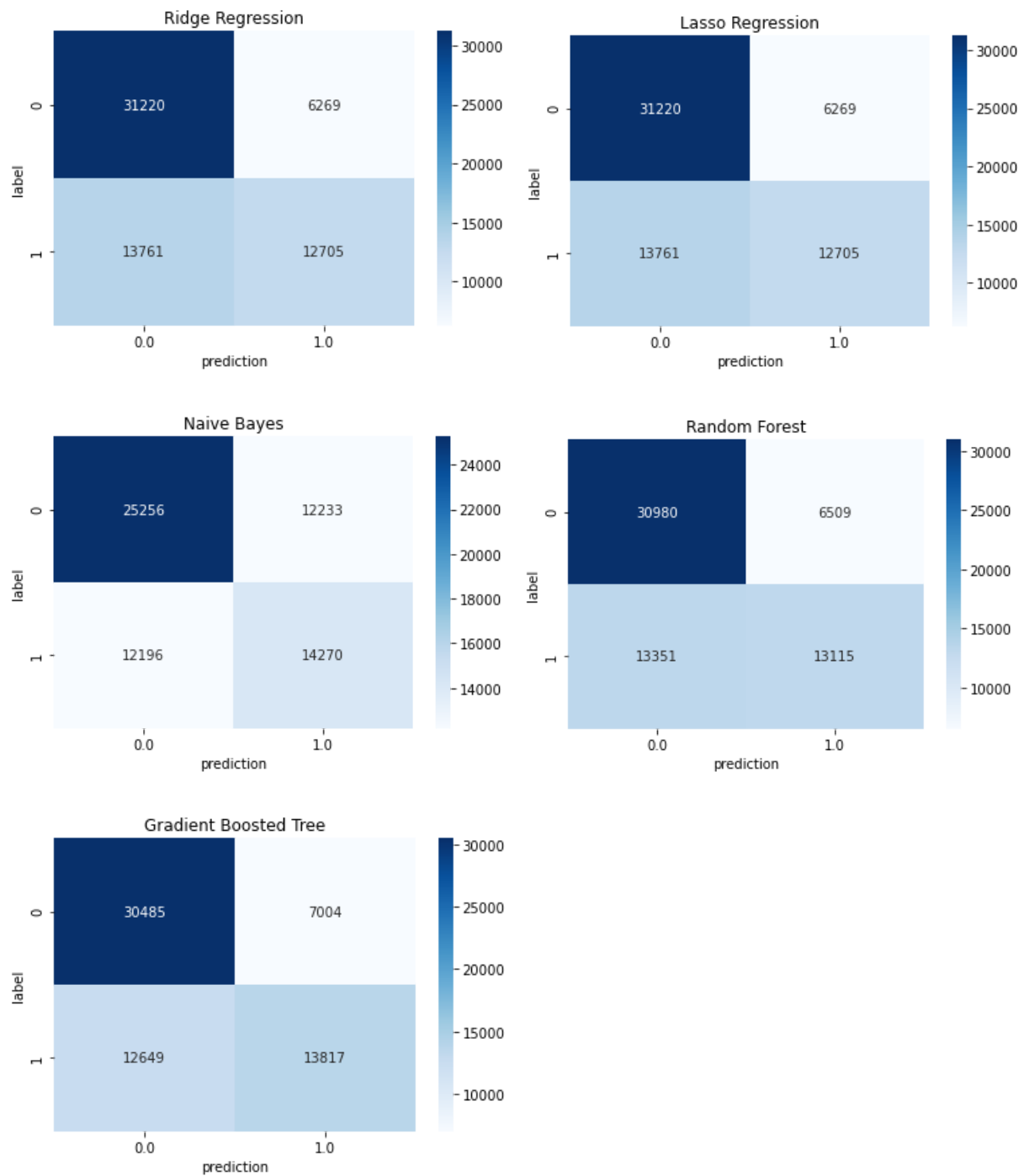
Results

Best on our evaluation criteria, the best results were obtained from Gradient Boosted Tree, but overall the models produced similar results; with Naive Bayes lagging behind the other models. Test data results are similar to training data results, possibly due to our use of cross-validation. We demonstrate some success in being able to predict the likelihood of booking on Airbnb using input data on user demographics and their interaction with the booking platform. Our sample is balanced between booked and not-booked outcomes, and therefore accuracy is used as the main metrics, however, we complemented it with other metrics as described above and presented in the table below. We also demonstrate the confusion matrixes for each of the models.

Comparison of Model Performance

Models	Naive Bayes	Ridge Logistic Regression	Lasso Logistic Regression	Random Forest	Gradient Boosted Tree
Training Accuracy	0.619	0.684	0.684	0.688	0.691
Test Accuracy	0.618	0.687	0.687	0.689	0.693
Training AUC	0.675	0.721	0.721	0.733	0.742
Test AUC	0.675	0.721	0.721	0.733	0.739
Training F1	0.619	0.673	0.673	0.678	0.684
Test F1	0.618	0.675	0.675	0.679	0.685
Training Precision	0.619	0.682	0.682	0.685	0.688
Test Precision	0.618	0.684	0.684	0.686	0.689
Training Recall	0.619	0.684	0.684	0.688	0.691
Test Recall	0.618	0.687	0.687	0.689	0.693
Training TPR	0.619	0.684	0.684	0.688	0.691
Test TPR	0.618	0.687	0.687	0.689	0.693
Training FPR	0.402	0.373	0.373	0.366	0.357
Test FPR	0.405	0.374	0.374	0.368	0.357

Confusion Matrices



Conclusions

Objective

Our objective was to classify users by whether they will likely book a stay on Airbnb using information such as demographics, web session records, and other summary statistics. By helping identify users that are most likely to book, we could help improve user experience, target advertising better, and increase AirBnB performance.

Results

The best results were obtained from Gradient Boosted Tree, but overall the models produced similar results, with Naive Bayes lagging behind the other models. We, therefore, show that it is possible to classify the users with success on whether they are likely to book or not. We also apply cross-validation to help us determine the best hyperparameters for each of the models.

While originally we tried including different variables (not demonstrated in this paper), we decided to evaluate all the models on the same set of variables and we saw that the results improved overall. We demonstrated that using demographics and sessions data, one can improve prediction of booking by users of Airbnb. Since our sample is balanced, we used accuracy and AUC as an indicator of model performance, but also evaluated F1 score, Precision, Recall, the true positive rate, the true negative rate, and the false positive rate. The results generally supported our findings using accuracy and AUC.

We found it particularly helpful to use the pipeline approach. We created a pipeline that included all the preprocessing, feature selection, model creation, cross-validation and then model output based on the test data. Cross-validation was an important tool for us to decide on the best parameters for our models. We first created a pipeline object with steps and then built a parameter grid. We then input the pipeline to the crossvalidator. Finally, we saved the final best models and the metrics to be evaluated and compared.

Future research

For future research, we would like to test whether a model with and without the session log features to pinpoint how much improvement those add on top of the user demographic features. Furthermore, we would like to do more feature engineering, using additional data, should it be with session log data and more detailed session log data (e.g. past search terms/locations). There are a few additional tables that we can join in to append and/or create richer features for our analysis and model training. For example, one table could be valuable in aggregating information on user activities during past web sessions. These other tables should provide some interesting opportunities to engineer additional features that we can join back to the primary datasets for our analysis.