Machine Learning Engineer Nanodegree

Capstone Project

Airbnb new user first destination prediction

Definition

Project Overview

Airbnb is a platform where users can book the place to live in the destination they want to travel. The places provided to travellers are not hotels or hostels, these are spare rooms people would like to offer traveller with exchange of money (usually cheaper than hotel and more comfortable than hostel). New users on Airbnb can book a place to stay in 34,000+ cities across 190+ countries. Some places are hot, so as room demands, while other places are not. Airbnb is interested in to know where users are likely to travel, so that proper recommendations and personalized contents can be used to increase the profits. As we all know, recommendation system suffers from cold start. When there is a new user, we don't

really know what he likes, which destination he/she would like to travel, and thus the personalized contents will be random, just from existing users database. By accurately predicting where a new user will book their first travel experience, Airbnb can share more personalized content with their community, decrease the average time to first booking, and better forecast demand.

The data used is from a finished Kaggle competition, which is real data from Airbnb. The data input contains a list of users along with their demographics, web session records, and some summary statistics. The whole dataset contains 5 csv files:train-users, test-users, sessions, countries, age-gender-bkts.

The data used is from a finished Kaggle competition, which is real data from Airbnb. The data input contains a list of users along with their demographics, web session records, and some summary statistics. The whole dataset contains 5 csv files:train-users, test-users, sessions, countries, age-gender-bkts.

1. train-users and test-users: The train-users files contains 213451 training examples with 16 properties:

- id
- date-account-created
- date-first-booking
- gender
- age
- signup-method
- signup-flow
- language
- affiliate-channel
- affiliate-provider
- first-affiliate-tracked
- signup-app
- first-device-type
- first-browser
- country-destination
- timestamp-firstactive

The test-users have 62096 users and 15 properties. The label "country-destination" is missing because this is the value we will predict.

The training and test sets are split by dates. In the test set, we are expected to predict country destination of all the new users with first activities after 4/1/2014.

2. sessions: The sessions file is the web sessions log records for users. The sessions file contains 5600850 examples and 6 properties:

- user-id
- action
- action-type
- action-detail
- device-type
- secs-elapsed

There are actually 74610 different users in the file.

3. countries: The countries file contains statistics of destination countries in this dataset and their geometric information. It has information for 10 countries and their 7 different properties, such as longitude and latitude.

4. age-gender-bkts: This file contains statistics of users' age group, gender, country of destination. It consists 420 examples and 5 properties.

Problem Statement

First we need to explore the data and find out useful feature to train our model. By first glance at the dataset, the major problem of the datasets is there are lots of missing values, and unreasonable values which should be excluded from model training. In addition, the features and the corresponding values in session dataset is not easy to decipher, and some values do not make sense. For example, some user have been online for more than 24 hours, but some user only took a few minutes. This data does not seem useful in model learning.

Another problem is which model to pick. The models comes into mind are random forest, gradient boosting, logistic regression, SVM, etc. The target label we want to predict is multi-class, and since SVM is very slow compared to random forest/gradient boosting, we can ignore this and compare the other models and then choose the best one.

Metrics:

By first glance at the target, we can see there is a very obvious skew towards "NDF", which means "not booked". We can thus make a baseline model that is always to predict "NDF" for every new user on Airbnb. Then we compare our training model against the baseline model using the training data.

We want to pick the prediction which has the highest probability among 12 possibilities. We need a metric scorer that calculate the rank based on relevance. The model picked is called

NDCG@k (Normalized discounted cumulative gain) where k = 5.

The formula is defined as below:

$$nDCGk = \frac{DCGk}{IDCGk}$$

$$DCGk = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{log_{2(i+1)}}$$

Where rel_i is the relevance of result at position i.

$IDCG_k$ is the maximum possible DCG for a given set of possible predictions. $nDCG$ is relative values in the interval between 0.0 and 1.0. For each prediction, we list 5 possible countries in descending probability order. We divide the training set into 5 folds, and use cross validation method scored by NDCG@k.

Analysis

Data Exploration

Here are a few questions that I am looking for the answers to:

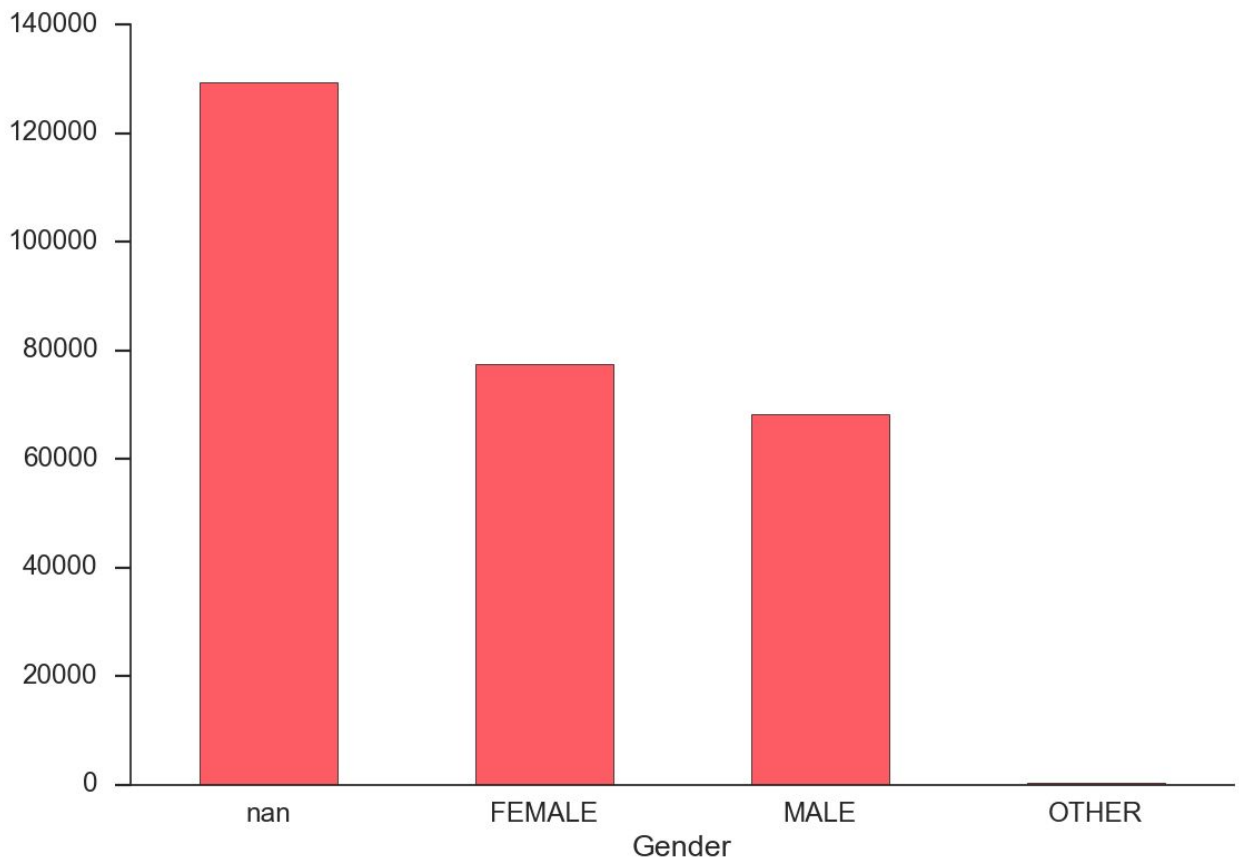
1. Does the data have missing values?
2. Does the data have strange and unrealistic behaviors that need to be removed?
3. Does the data show some peculiar behavior?

Training data

The training user data has plenty of missing values represented by “-unknown-”. They should be replaced by “NAN”. We have high missing percentage on “age”, “gender” and “data_first_booking”.

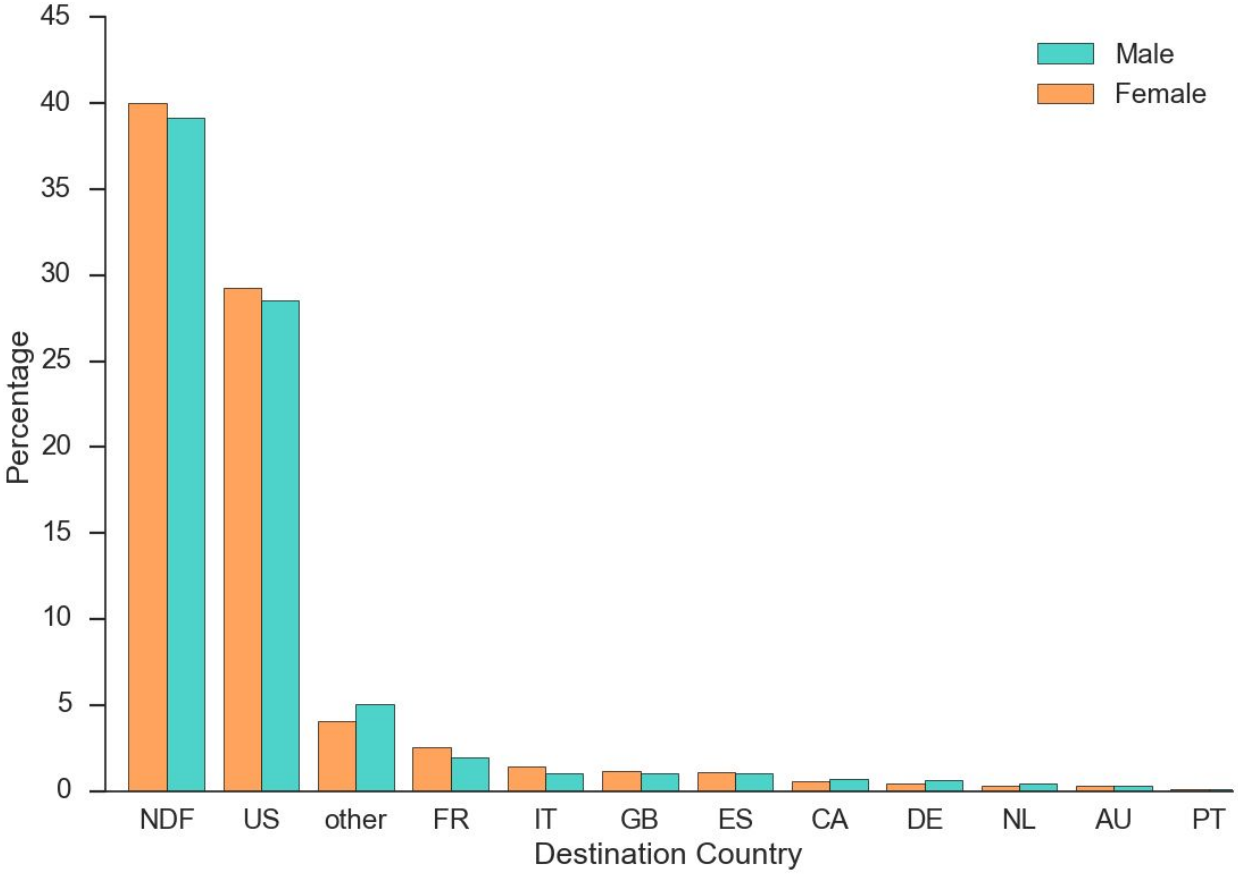
Meanwhile, “age” shows unreasonable standard deviation, min age and max age. No one can live 2014 years, and no 1-year-old baby can register an account on any website by its own. The age range between 16 and 100 is reasonable, so the data out of this range are omitted.

It is hard to explore data via words. I will show the visualizations of the data.



Picture 1. Gender distribution over the training dataset

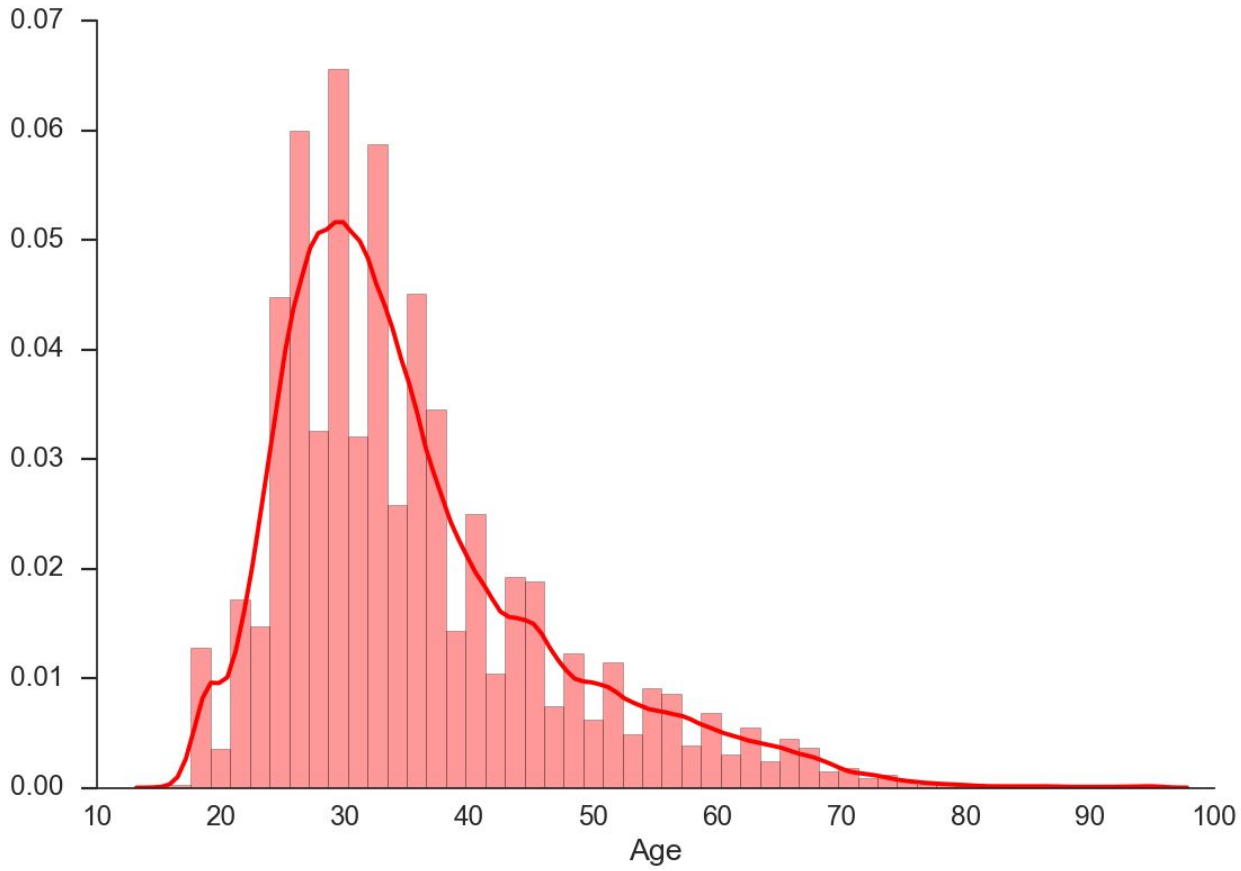
There is no big difference between number of female and male travellers. It is interesting that airbnb allows “unknown” gender which contributes to missing gender data. Now let's see if there is gender preference for destinations.



Picture 2. Gender distribution per destination country

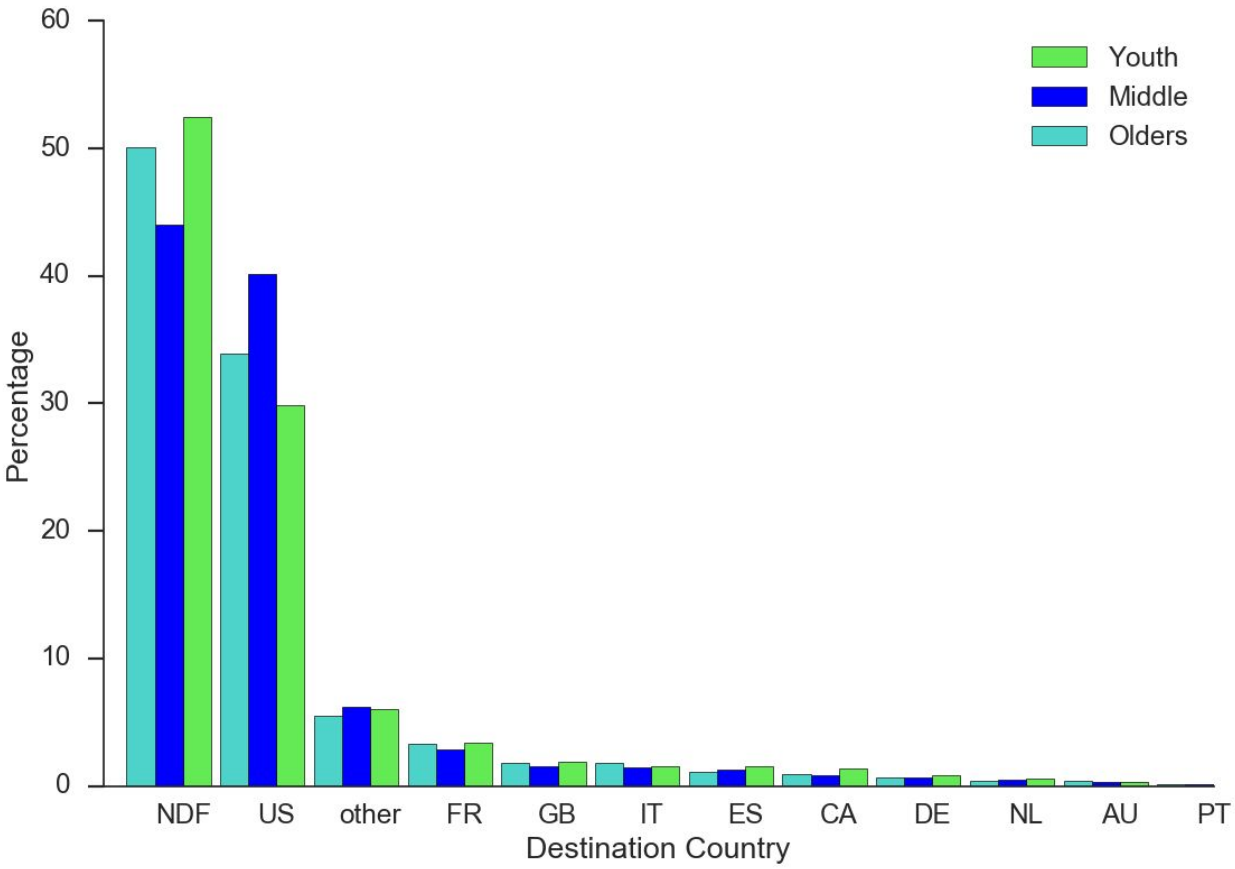
Picture 2 shows the gender distribution per travelling country. There is no obvious difference from male and female. However, there picture tells us the destination preference. United States has most travellers.

Let's see if there is age preference for the destinations. “NDF” are dropped here because it means “not booking” which is not useful to describe age distribution over destination countries.



Picture 3. Age distribution over booked training users

The distribution is right skewed, meaning that young and middle people between 20 to 45 like traveling more than older people. We can split the data and show if there is difference between young people, middle people and old people.

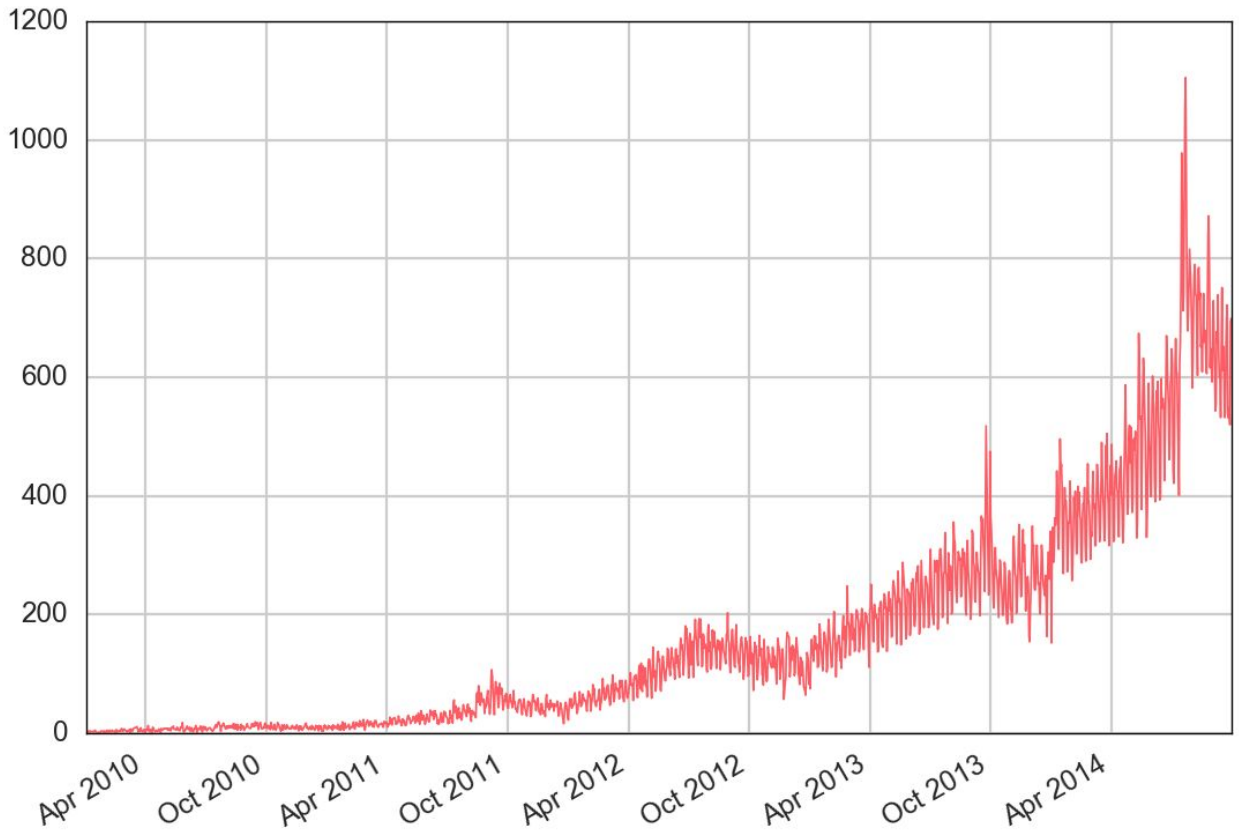


Picture 4. Destination country distribution per age group

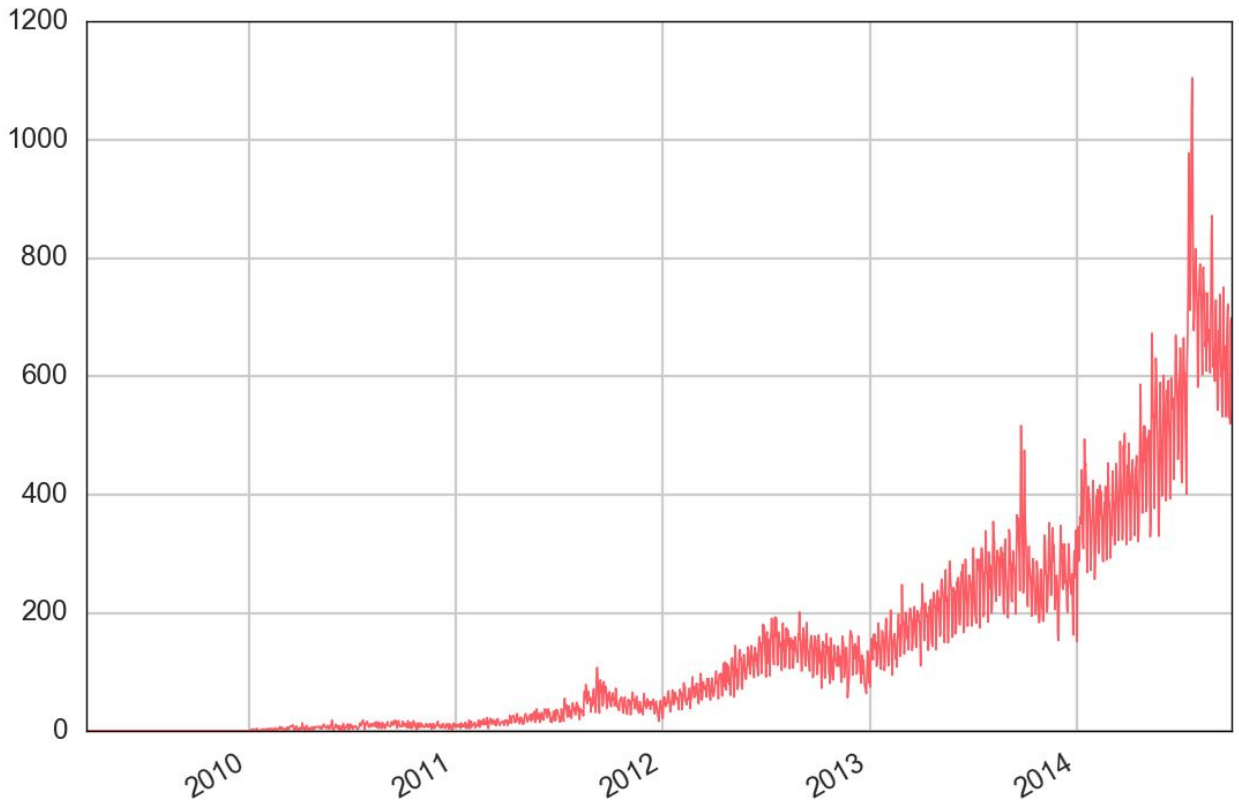
Looks like middle age people are more likely to travel to/within the United States. Compared to middle age people, young/old people are more likely to travel to other countries.

The fact that most people like to travel to/within the United States is interesting. There is a feature “language”. There is 96% of users who speak English. Therefore, there are more people to travel to/within US. However, this does not explain why there is so few people traveling to GB which is also a English speaking country. There must be another factor that is not taken into account.

Let’s see the distribution of the number of accounts created and distribution of the date account are first active



Picture 5. distribution of the number of accounts created



Picture 6. distribution of dates that accounts that are first active

The trend of date_account_created and date_first_active over time are almost the same. We can see the rapid growth from 2013.

Session data:

The session data contains the devices the users use to browse Airbnb website, actions the users take on the website, time elapsed when the users spent time browsing. These data does not seem obviously related to where the user want to travel, but these data represents users' habits, how much they earn, and can be used as "voices" from the users.

There are 10 unique action type, among which I think "booking_request" and "booking_response" means the user is booking a place to live. A user typically first spends time browsing and viewing the rooms, and makes a booking request. So a serial actions may indicate that whether the user will book a room or not.

The device a user uses may indicate if the user is rich or not. For example, Mac/Iphone users are likely richer than Windows/Android users. So they can afford more expensive destination countries, and more expensive room but also more comfortable.

Feature “time elapsed” shows how much time the user spends on the website. However, this feature may not be useful in this dataset. Here is the reason, the mean is 1.940581e+04 (13.5 hours), and the variance is 8.888424e+04 (24.7 hours). This is not reasonable. The long time elapsed may due to that the user did not close the website during this period but the server is still incrementing the elapsed time.

Algorithms and Techniques

The Airbnb dataset is a real world data, which means it must have lots of features and plenty of missing values. Thus it is not suitable to use SVM. As we know, we needs to define a kernel function for a SVM and then compare the similarities between each data point. However, at beginning we have no idea what kernel function to choose that fit the dataset. Moreover, “dimension curse” indicates that too many features will cause significantly long run time, especially for SVM which is computation intensive. In other words, SVM is not suitable for such many-feature training because it does not scale well.

One might also ask how about Multiclass Logistic Regression? Well, the “dimension curse” still holds. We have no idea what is a perfect function to represent the model, and neither do we know in advance the linearity/non-linearity relationship. Plus, Logistic Regression is sensitive to both noise happened in input data, and the learning rate. As we can see, there is lots of missing data, and there is data has large variance.

We should choose a model that can adapt the unknown and hidden nonlinearity itself, and is not very sensitive to noise in the input data. Decision tree, or tree-ensembles like random forest or gradient boosting descent tree. Single decision tree is like an opinion from a single person, which dictates the outcome of the prediction. We can include more opinions from more people, and give them right to vote. Trees that are more correct than others get more attention. Hence, gradient boosting descent tree is chosen.

Benchmark

Airbnb does not provide a baseline or benchmark for the prediction performance. However, we can build one. Most users travel to/within the United States, so a reasonable baseline model is to always predict “US” as new users first travel experience.

Methodology

Data Preprocessing

Definitely the dataset needs to be pre-processed. As mentioned before, there are missing data and anomalies in the training data.

1. Feature “age” has a unrealistic range from infant to 2014. This is not possible for small kids to signup Airbnb and book a trip, and no one can ever live 2014 years. So “age” will be pre-processed to only include age from 16 to 100, and will be divided into different buckets, e.g., 17 years ago will be in bucket [16, 20].
2. Target are destination countries which are strings. So they should be encoded as multi-class label.
3. Non-numeric feature will be one-hot encoded.
4. Missing data are encoded to be negative one.
5. Session data includes the information of “when users signed up Airbnb”, “when users are actively the first time after they signed up Airbnb” and “when users book a trip the first time after they ”. These values are transformed into years, months and dates. Also the time users spent on one session are transformed into the following categories:
    a. Session time elapsed sum, mean, max, min, 10% quantile, 25% quantile, 75% quantile, 90% quantile, median, standard deviation, variance, and skew.
    b. Day_pauses (how many times the user spent time on Airbnb for more than 24 hours)
    c. Short_sessions (how many times the user spent time on Airbnb for less than 1 hour)
    d. Long_pauses (how many times the user spent time on Airbnb for longer than 300000 seconds)

Implementation

Metrics is already defined which is NDCG@k. The parameter k is set to 5. Model is chosen as well which is gradient boosting descent tree (GBDT). We use one-hot-encoding to encode the non-numeric features in order to make xgboost work. The training label are encoded into 12 classes, ranging from 0 to 11. One challenge is how to treat the missing values. In order to distinguish them from other values which are all nonnegative, the missing values are hardcoded to be a negative one.

Refinement

One challenge to build a prediction model is to pick the best set of parameters so that the model can generalize well. One problem to overcome is the overfitting problem. Machine can cheat if there is merely one set of training data. There are two approaches that can be combined into parameters selection. First one is cross validation to overcome overfitting problem, which is to cut the training data into multiple folds, and for each fold the training process pick the data not in this fold evaluate the data on the current fold. The other approach is grid search which searches the best set of model parameters. The data set are cut into 5 pieces for cross validation. For parameter selection, there are a few interesting ones from GBDT that are focused on:

max_depth	5, 6, 7
learning_rate	0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4
subsample	0.7, 0.8, 0.9, 1
colsample_bytree	0.7, 0.8, 0.9, 1
n_estimators	25, 50, 100

Table 1. Parameter selection table

In python xgboost lib, the default value of max_depth is 6 which is a small number. This is different from random forest in that boosting usually fits shallow tree while random forest fits deep trees. Learning_rate is the boosting learning rate. The larger the rate, the faster the model change. However, if the rate is too large, it is likely that the model will skip the optimal point. Subsample is the fraction of data sampled without replacement, and comsample_bytree is the fraction of predictors to use in fitting the trees. These parameters

are similar to random forest to avoid overfitting. The parameter n_estimators is the number of trees to fit, and the default value is 100.

The parameters for the training are picked from the table, so in total there are 3 * 7 * 4 * 4 * 3 = 2 = 1008 different sets of parameters. I have a computer with 16GB memory, it is not feasible to train the model with such many times on the training data that has 213452 training point and hundreds of features. So instead of training all the data, I picked 20% of the entire input data. Below parameters are picked by the learning process:

n_estimators: 50

subsample: 1

learning_rate: 0.35

colsample_bytree: 0.7

max_depth: 5

Results

Model Evaluation and Validation

The final model is the GBDT with parameters described in the previous section. The final score is generated from submission score on Kaggle is 0.85266. The best score on the leaderboard is 0.88697. The baseline model which always predicts “US” gets a relatively low score 0.67908.

Validation average score	Kaggle private score	Kaggle public score	baseline model score
0.919974784698	0.85600	0.85266	0.67908

Concluding remarks

Working on this project illustrated how much work can go into pre-processing data before fitting a model, including joining data from multiple files, one hot encoding categorical variables, and handling missing or erroneous data

In addition, the process of feature selection showed how much flexibility exists in the creation and utilization of features. Due to erroneous data some data were dropped, and there are many features that could have been created but were not tested out.

Some of the feature selection ideas that were explored but didn't improve performance include:

- Reducing dimension by PCA to 30 dimensions
- Adding language_levenshtein_distance from the countries file
- Lower the sample rate from 100% to avoid overfitting.