New User Booking Prediction for Airbnb Historical Data

Yingzhi Wu
A53102471
M.S. Computer Science
University of California, San Diego
yiw376@eng.ucsd.edu

Zhimin Zhou
A53089795
M.S. Computer Science
University of California, San Diego
zhz249@eng.ucsd.edu

Jingyuan Li
A53101535
M.S. Computer Science
University of California, San Diego
jil663@eng.ucsd.edu

ABSTRACT

Performing data mining based on customers' historical behavior statistics can reveal quite much hidden information that companies weren't paid attention to in the past. Correctly modeling the customers helps to guide the marketing strategies. In this paper we explain our current best solution to predict Airbnb's new users' first booking destination. First, we show our findings about the characteristics of the dataset which is given by the company. We then talk about our feature selection strategy based on these observations. For model design, we land on three types of classifiers. Then we discuss model optimization by introducing the one-vs-the-rest strategy, and parameter tuning. Our prediction results are evaluated according to the metric called NDCG (Normalized discounted cumulative gain), which is suggested by the hosting company. By looking at the value of NDCG and comparing with our competitors (We are at the **1st** place when this report is written), we are firmly convinced that our solution is very effective.

Author Keywords

Supervised machine learning algorithm, Classification, Gradient boosting

INTRODUCTION

Background

For online travel agencies (OTAs), the enormous user data that they have been inherently maintaining is a mine of gold. How to effectively enhance user experience and increase total bookings by utilizing huge dataset remains a question to answer. Airbnb opens this challenge to its potential employees. From their point of view, by accurately predicting the destination a new user might travel to for the first time, "Airbnb can share more personalized content with their community, decrease the average time to first booking, and better forecast demand".

December 1, 2015, La Jolla, California, USA.

Problem Definition

As participants, we are required to predict the new users' first booking destination at the granularity of country. There are 11 destinations to choose from, including Australia, Canada, Germany, Spain, France, Great Britain, Italy, Netherlands, Portugal, the U.S., and all the rest are labeled as "others". Users who haven't made a booking are categorized with the "NDF" label in the destination field.

In the dataset, Airbnb provides a labeled training set of 171239 entries, and a test set of 43673 entries to predict with. There are 2 other statistical data sheets regarding some geographical information about each of the listed countries and there population age and gender distribution. A session set documented the time elapsed for each client side and server side actions, with no references provided to explain how the naming matches with operations.

The answer to every entry that each participating team are expected to submit, is a list of ranked destinations that a new user might make as the destination of his first booking. One can list up to 5 destinations for predicting a new user's decision. The prediction is evaluated by NDCG (Normalized discounted cumulative gain).The points reward will decrease as the match of prediction comes in lower ranks.

Our Solution

First, we study the dataset by collecting statistics and visualization. According to the calculated variances and some reasoning based on travelers' behaviors, we pick out the features for our model. For model selection, we first make our mind on staying with supervised learning algorithms, then narrow the category down to utilize classification methods. By taking a close look to different classifiers, we choose 3 models to perform a basic test. From there, we are convinced that the Gradient Boosting algorithm seems to work best for this problem. As for optimization and model tuning, we utilize a "one-vs-the-rest" strategy to help conduct better classification, and our parameters are tuned based on this improved model. We evaluate our results against the NDCG metric, which is also proposed by the company. Our submission results beat all the rest teams up to the time this report is being finalized.

Related Work

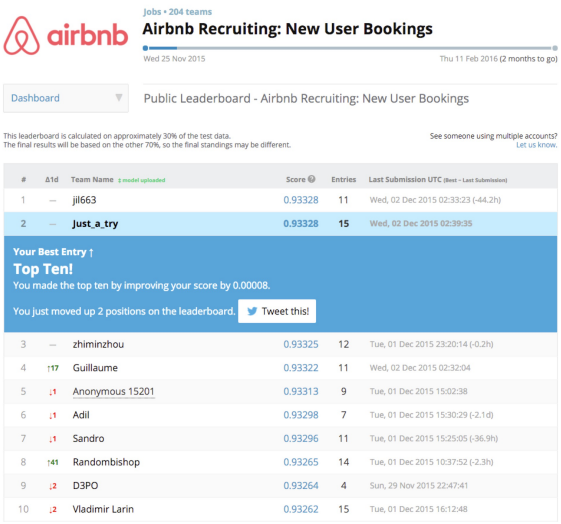

Figure 1. Kaggle Competition Dashboard Screen Shot ***Our team members take up the top 3 positons***

There are some literatures on tourist destination choice which are related to our topic. One of them is using Spatial Interaction Models to predict the likelihood of people or even goods moving between two locations in space. Generally, this model predicts the interchange of people between all points of a discrete set of locations or zones in a square matrix of movements or flows from each zone to zone. This model strongly relies on three types of model: spatial data, typically the amount of population, environment, etc. within a zone; the distance or travel time between two locations; spatial interaction data, whether in the form of prior matrices or thorough the estimation of structural parameters, like records of the movements of travelers between locations in space. The strength of this model is its simplicity. However, its simplicity may also be the major weakness of this model approach, since it closely depends on the spatial data. If the actual process is context dependent or cant provide spatial data, like our experiment, this model may result in misleading predictions.[1, 3]

Another approach is a rule-based model of tourist destination choice. Different from the algebraic spatial interaction model, qualitative rule-based model assumes that a set of logical rules drive the choice behavior of interest. Traditionally, rule-based model depends on expert knowledge which lacks any test of whether expert knowledge constitutes a valid representation of observed choice behavior. Besides, there are only 3 valid research spatial data. From what we learn, this approach likes the Decision Trees Model but using empirical data from recent researches. However, the dataset used by this model mainly presents the rules of European, while the users of our test set are all American. Therefore, we doubt that this model using scarce empirical data can work well with our dataset.

The evaluation metrics of our problem is a popular evaluation method for multiclass classification models. As described in [2], a perfect classifier can lead to perfect DCG score. To illustrate this argument, the author casted a Webpage Ranking problem into a classification problem and adopted Gradient Boosting model for classification. Their experimental results demonstrated that this model outperformed the typical regression model. For our problem, this model could also be very useful due to the similarity of the evaluation metrics.

DATASET ANALYSIS

We are using the dataset from the Airbnb Recruiting: New User Bookings completion of Kaggle for this experiment. This dataset reveals basic information of users who have or havent book their first destination in Airbnb. Besides, this dataset includes other information like demographics of specified countries, users web session records and some summary statistics. All information is captured in four files:[4]

- train_users.csv: the training set of users. Id, date_account_created, timestamp_first_active, date_first_booking, gender, age, signup_method, signup_flow, language, affiliate_channel(what kind of paid marketing), affiliate_provider(where the marketing is, e.g. Google, Craigslist), first_affiliate_tracked(whats the first marketing the user interacted with before the signing up), signup_app, first_device_type, first_browser, country_destination(the target variable we want to predict)

- sessions.csv: web sessions log for users. user_id, action, action_type, action_detail, device_type, secs_elapsed(the number of seconds between actions were recorded)

- countries.csv: summary statistics of destination countries in this dataset and their locations. Country_destination, lat_destination(the latitude of geographical center of the country), lng_destination, distance_km(how far is the destination country from the US), destination_km2, destination_language, language_levenshtein_distance(the language levenshtein distance between the destination country and English)

- age_gender_bkts.csv: summary statistics of users age group, gender, country of destination. Age_bucker(age range, e.g. 35-39, 40-44), country_destination, gender, population_in_thousands, year.

From the train_users.csv, we calculate the number of users who booked their first destination within different countries. From Figure 2, more than half of the users (99,152, while the total number of users in train_users.csv is 171,239) havent booked their first destination yet (NDF = no destination found). In addition, since all the users in this dataset are from the USA, almost one third of the whole users choose the USA as their first destination in Airbnb. While the remaining one sixth of the whole users choose from 10 different countries (FR, CA, GB, ES, IT, PT, NL, DE, AU and other).

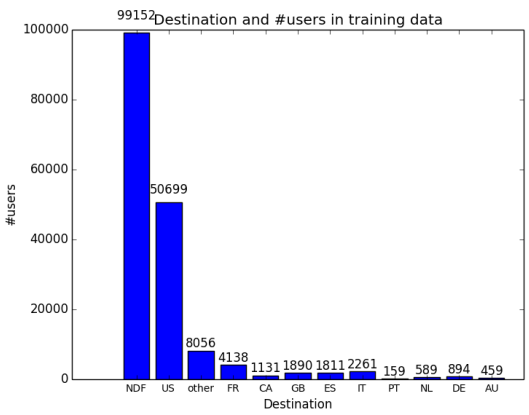

Figure 2. Number of users of different destination countries in training data

Data Pruning

Since our target is to predict which country a new users first booking destination will be, and those whose first destination is NDF cant provide any effective assumption for us to predict a specified destination, we decide to discard those user records which destination is NDF.

Data	Users	Destination
Original	171,239	12
Dataset A	72,087	11
Dataset B(with destination is NDF, discarded)	99,152	1

Table 1. Dataset Information

Feature Parsing

After discarding those NDF data, we start to study the basic information of users from train_users.csv. We notice that most of the features in train_users.csv are not numeric but using words to present different value. In order to make the predict algorithm work effectively, we decide to use Encode Label for reference. The main idea of Encode Label is encoding some feature with value between 1 and n_classes which is the number of types of this feature. In instance, first_device_types values include Android Phone, Android Tablet, iPhone, iPad, Desktop and etc. so we encode these device type into the integer number from 1 to 9, using 1 to represent Android Phone, 2 to represent Android Tablet and so on.

Meanwhile, we also find that the raw data use different formats to describe the time information, like 2014-04-01 in date_account_created and date_first_booking, while 20140401000102 in timestamp_first_active. Although Encode Label can translate these time value into distinct numbers, it will introduce a huge range of value while each value only contributes to a limited number of samples. Therefore, we decide to split the original time value into 3 different features: year, month and day. This process could cause overfitting because of increasing the number of features, but we will use feature selection to address this problem.

Besides, considering the problem that some feature has a huge range of values but most values correlate to a small number of samples, we combine the value of age feature into groups. In each age group, the range presents one decade. Because age is sensitive information, we find out that almost a quarter of the users dont provide the age information. However, via summing up the number of users in different age ranges, we notice that the destination distribution of unknown is the same as the one of the average age range (30-39). So we change the unknown age to the average age range.(As Figure 3)

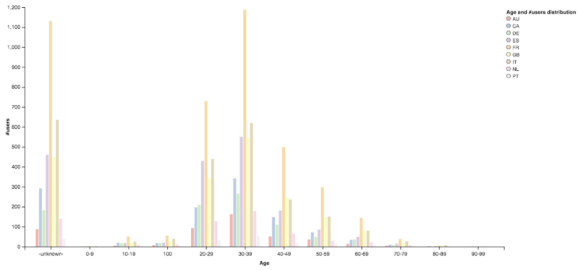

Figure 3. Number of users of different age group in training data

Feature Selection

At the glance of the original dataset, we find out that some features have the same value in most samples. We decide to perform feature selection for three benefits:[5]

1. Reduce overfitting. Less redundant data means less opportunity to make decisions based on noise.

2. Improve Accuracy. Less misleading data means improving modeling accuracy.

3. Reduce training time. Less data means less time for the algorithm training data.

First, we remove features with low variance.[6] After parsing features from the raw data, we have 19 features (year, month, day, created_year, created_month, created_day, book_year, book_month, book_day, affiliate_channel, gender, age, language, affiliate_provider, first_affiliate_tracked, first_device_type, first_browser, signup_flow, signup_app, signup_method). And then we use VarianceThreshold from sklearn,feature_selection library to remove features with low variance. As we set the variance threshold to 1.4, this algorithm helps us remove all features whose variance is less than 1.4. After that the number of features is decreased to 8 (month, book_month, age, language, affiliate_channel, affiliate_provider, first_device_type, first_browser). And comparison of the nDCG (the evaluation metric which will be discussed in the later chapter and it values on the interval 0.0 to 1.0. Closer to 1.0 means better prediction) and consuming time to predict the validation set between using all 19 features and using only 8 features after removing features with low variance is shown in table 2. From the table, we find that albeit the nDCG of prediction of validation set with 19

features is better than the one with 8 features, the nDCG of prediction of test set with 8 features outperforms the one with 19 features because of overfitting and the consuming time is reduced by half. Therefore, we decide to use the 8 features for now.

#features	nDCG of validation set	nDCG of test set	Time to predict validation label
19	0.924904200671	0.92134	0:00:22.911990
8	0.924822141755	0.93311	0:00:11.391343
7	0.924874824008	0.93320	0:00:09.113650

Table 2. Results of using Gradient Boosting Classification with different features

Second, we find out that among the remaining 8 features, there are two features are closely correlated, which are affiliate_channel, affiliate_provider. From the raw data, we notice that whenever affiliate_provider is google, affiliate_channel is seo; whenever affiliate_provider is craigslist, affiliate_channel is other; and whenever affiliate_provider is direct, affiliate_channel is direct. Hence, we decide to discard affiliate_channel and then we have only 7 features with better nDCG results and less time to predict.

To better view how the chosen features affect the prediction, we use bar diagrams to observe distribution of the number of users within different features. From these diagrams, we notice that these features have a remarkable influence on predict the destination. Like first_created_month and first_booking_month, the number of users who choose European countries stays higher in the middle of the year than in the winter, while the number of users who choose Australia stays higher in the winter than in the summer.(as Figure 4 and 5) By calculating the ratio of users in specified age group in all users in this country, we also find different trends among different age groups. Users whose age is from 20 to 29 prefer Spain and Germany, while those whose age is from 50 to 59 prefer Portugal and Australia.(as Figure 6 and 7)

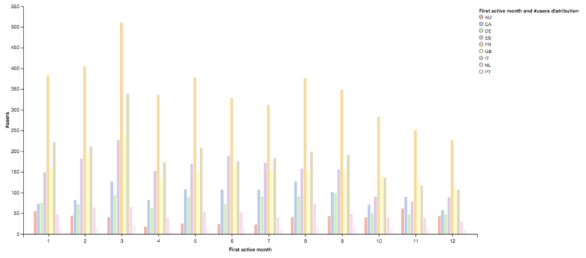

Figure 4. Number of users of different first created month in training data

As for information from other files, like sessions.csv, countries.csv and age_gender_bkts.csv, we decide not to use any features generated from these files based on the following reasons.

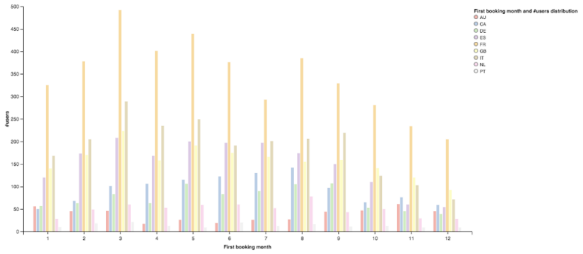

Figure 5. Number of users of different first book month in training data

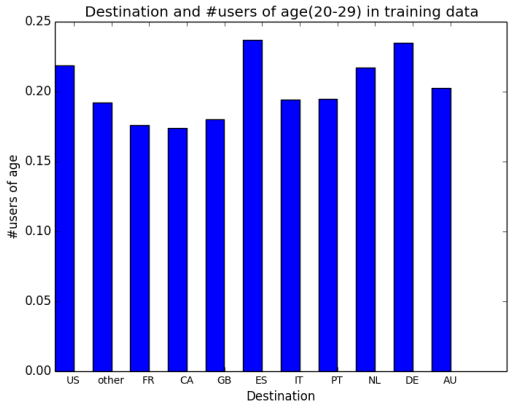

Figure 6. ratio of users whose age is 20-29 of different destinations

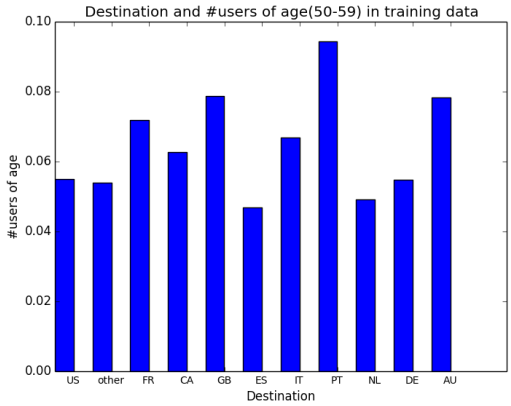

Figure 7. ratio of users whose age is 50-59 of different destinations

1. In the sessions dataset, the data only dates back to 2014/1/1, while the users dataset dates back to 2010, which means there are more two thirds of users cant be matched with any records in sessions dataset. We dont want incomplete samples affect the accuracy of our prediction.

2. In the countries dataset, since we are predicting users first booking destination, while this dataset is about the information of 10 different destinations, we dont have means to pick any feature because of not knowing the destination yet. So we dont use features from this dataset.

3. In the age gender buckets dataset, we already choose age group based on the feature parsing and feature selection procedures before. In order to get rid of replication, we decide not to use any features from this dataset, either.

METHODOLOGY

Given that our dataset comes with categorical labels, which are the destinations of users' fist booking. Each data entry is a pair of an input vector and a desired output value. This matches with the supervised learning problem setting. Since the expected prediction output is discrete categorical values, we further identify such problem to be best solved by classification methods.

Classification

We then focus on finding a classifier which is the optimal fit for solving this problem.

Gaussian Naïve Bayes

We start from the simplest classification model, Naïve Bayes. Although the assumptions of Gaussian Naïve Bayes model is apparently over-simplified, it works very well on our feature set. As discussed in the lecture, Naïve Bayes methods are a set of supervised learning algorithms based on applying Bayes theorem with the naïve assumption of independence between every pair of features. The classification result replies on the prior probability of classes and the posterior probability of features given labels. Different naïve Bayes classifiers differ mainly by the way to compute the posterior distribution. Gaussian Naïve Bayes assumes the likelihood of features to be Gaussian.

There are two main factors that essentially influence the performance of Naïve Bayes classifier according to [7], an even distribution of nodes in each feature vector and an even dependence distribution of feature vectors. Our feature selection criteria just guaranteed that our feature set satisfy both of the requirements. First, from *Figure 3 Figure 5*, we can see that the data point distribution in each feature class is relatively even and Gaussian shaped. Second, out of 19 features, we only chose 7 of them with the highest variance and lowest correlation. This ensures the model to give a second best performance of all of our three models.

Nearest Neighbor Classification

Nearest Neighbor[8, 9] Classification assumes that the data is in a feature space. More exactly, the data are in a matric space. Based on different applications, the data can be scalars or even multidimensional matrix. Since the data points are in the feature space, they have a notion of distance between each other. Therefore, Nearest Neighbor Classification is a type of instance-based learning or non-generalizing learning: it doesnt try to construct a general internal model but simply stores instances of the training data. And then the classification is computed form a simple majority vote of the nearest neighbors of each point: a query point is assigned the data class which has the most representatives within the nearest neighbors of the point.

Among several implementations of Nearest Neighbor Classification, we decide to pick the most commonly used technique K Neighbors Classification. Basically, what this algorithm do is that it tries to find the k nearest neighbor and do a majority voting. A very common thing to do is weighted based on its distance, for example, the weight equals the inverse of the distance. In this case, neighboring points have a higher vote than the farther ones. Therefore, the optimal choice of the value k is highly data-dependent, while in general, a larger k suppresses the effects of noise, but makes the classification boundaries less distinct.

To pick an optimal k, we make a experiment to try different k range from 1 to 100 to calculate a better validation predictions. As a result learned from Figure, we find out when k=25, the nDCG of validation set is the best and those values larger than 25 stay the same performance but cost more time to classify a new data set. Therefore, we decide to use k=25 to train this model.

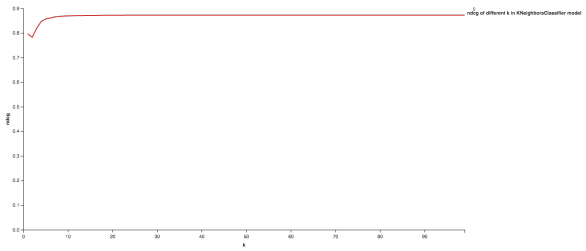

Figure 8. nDCG of different K in Nearest Neighbor Classification. The best performance is when K=25

Gradient Boosting

"When designing a model in domain-specific areas, one strategy is to build a model from theory and adjust its parameters based on the observed data"[13]. However, researchers often aren't able to acquire such information in the first place. So we sometimes fall back to data-driven models. Instead of building a single strong predictive model, the ensemble approach maybe a better choice.

Gradient Boosting is such a machine learning algorithm that produces a prediction model which ensembles weak prediction models such as decision trees. It builds up the model in a stage-wise way simulates other boosting methods. It generalizes following the gradient descent fashion, that is, by optimize an arbitrary differentiable loss function.

The algorithm optimizes the loss function over the function space by iteratively picking a weak hypothesis function that points in the negative gradient direction, and then combine those weak learners together iteratively into a single strong learner. It introduces a weak learner in each stage to compensate the shortcomings of existing weak learners. Such shortcomings are measured by calculating gradients on loss functions.

Since we have discrete non-linear data, we pick the K-class multinomial deviance loss function for cost measurement.

$$L(y, p(x)) = -\sum_{k=1}^{K} I(y = G_k) log p_k(x)$$

$$= -\sum_{k=1}^{K} I(y = G_k) f_k(x) + log \sum_{l=1}^{K} e^{f_l(x)}$$

,

where,

$$p_k(x) = \frac{e^{f_k(x)}}{\sum_{l=1}^{k} e^{f_l(x)}}$$

We calculate the probabilities for each class from generalizing logistic model to K classes.

To build up our model, we first choose K score functions: f_1, f_2,..., f_K. Each of these functions assigns a score for the matching class. The scores are used to calculate probabilities $p_k(x)$ as mentioned above.

Predicted Label = Class that has the highest probability.

Next, iteratively calculate loss Function for each data point.

- Step 1: turn the label yi into a (true) probability distribution $Y_c(xi)$.
- Step 2: calculate the predicted probability distribution $p_c(xi)$ based on the current model f_1, f_2,..., f_K.
- Step 3: calculate the difference between the true probability distribution and the predicted probability distribution.

Here, our goal is to minimize the total loss, so as to match the true probability distribution as closely as possible. After iterations, we achieve this goal by adjusting our models f_1, f_2,..., f_K by taking gradient descent.

Given any differentiable loss function L, start with initial models f_1, f_2,..., f_K, iterate until converge:
calculate negative gradients for class 1:

$$g(x_i) = \frac{L(y_i, f_1(x_i))}{f_1(x_i)}$$

calculate negative gradients for class 2:

$$g(x_i) = \frac{L(y_i, f_2(x_i))}{f_2(x_i)}$$

...

calculate negative gradients for class K:

$$g(x_i) = \frac{L(y_i, f_K(x_i))}{f_K(x_i)}$$

fit a regression tree h_1 to negative gradients $g_1(x_i)$ fit a regression tree h_2 to negative gradients $g_2(x_i)$

...

fit a regression tree h_K to negative gradients $g_K(x_i)$

$$f_1 := f_1 + \rho_1 h_1$$

$$f_2 := f_2 + \rho_2 h_2$$

...

$$f_K := f_K + \rho_K h_K$$

where ρ is the learning rate to control. Learning rate is used for scale the contribution of each new base model.

It turns out that, for this specific travel prediction task, the Gradient Boosting algorithm performs the best.

We will visit the discussion of performance optimization regarding different parameter settings in detail, in the next section.

OPTIMIZATION

OVR Strategy

Typically, there are two multiclass classification strategies, one-vs-one (OvO) and one-vs-the-rest (OvR). One-vs-one (OvO) strategy constructs one classifier per pair of classes. At prediction time, the class which receives the most votes will be selected. If there is tie, then it selects the class with the highest aggregate classification confidence by summing over the pair-wise classification confidence levels computed by the underlying binary classifiers. Since it requires one classifier with each pair of classes, it works well with situations where comparisons between individual classes are important. For our problem, it will complicate the problem since it is not necessary to do classification between one class with each of the rest of classes individually.[10]

Instead, one-vs-the-rest (OvR) consists in fitting one classifier per class. For each classifier, the class is fitted against all the other classes. The strategy fits perfectly with the goal of our problem since we only care about the probability estimates of each class for feature *X* returned by its own sub-classifier. With the probability estimate list, we can easily generate a list with the top 5 most possible destination countries for each test user.

One-vs-the-rest (OvR) strategy applies over the layer of detailed classification models. It accepts an estimator object parameter which implements fit and prediction functions. In our project, we passed the three sub-classifiers discussed in this section to the strategy function respectively. The advantage of wrapping the sub-classifiers is that each class has one classifier instead of only one classifier for all the classes. The performance on the validation set is much better than directly fitting one classifier for all the classes.

Parameter Tuning

The Gradient Boosting model with default parameter settings gives the best testing performance on Kaggle. To further improve the performance, we tuned the parameters from three aspects.

1. Number of classes: This parameter controls the numbers of boosting stages. Generally speaking, with more boosting stages, the validation error will be smaller. We set other parameters to be default, the evaluation score on validation set with number of classes from 1 to 20 is shown in Table 3. The data shows a strong linear relationship. But for testing data, large value of number of classes will overfit the model and give bad evaluation score. After sufficient amount of trials, we found that the best value is 11.

2. Max depth: This parameter controls the depth of the learning tree. The size of the learning tree cant be two large or too small. We tuned this parameter between 1 to 4 with n_estimator = 11 and other parameters to be default. The evaluation score with the value of max depth from 1 to 4 is listed in Table 4. The data indicates that with max depth = 3, the evaluation score is the highest. The test result on Kaggle also justifies the rule.

3. Learning rate: The learning rate shrinks the contribution of each tree. As we change the value of learning rate and calculate the evaluation score of number of classes from 1 50, we found that smaller learning rate favors smaller number of classes. Since our best performance number of classes is 11, after great amount of trials, we settle the learning rate to be a smaller value as 0.05. And this combination also gives the best performance on Kaggle as expected.

number of classes	nDCG of validation set
1	0.924898
2	0.924874
3	0.924814
4	0.924839
5	0.924860
6	0.924873
7	0.924860
8	0.924894
9	0.924891
10	0.924852
11	0.924914
12	0.924939
13	0.924914
14	0.924916
15	0.924913
16	0.924946
17	0.924984
18	0.924989
19	0.925004
20	0.925007

Table 3. The evaluation score on validation set with number of classes from 1 to 20, other parameters are set default

max depth	nDCG of validation set
1	0.924874
2	0.924824
3	0.924895
4	0.924712

Table 4. The evaluation score on validation set with max depth from 1 to 4, number of classes = 11, other parameters are set default

EVALUATION

We evaluate different models on the rating prediction problem. First, let's explain the metric used for this task's evaluation.

Metric

As Airbnb proposed, the evaluation metric for this task is to calculate NDCG (Normalized discounted cumulative gain[11, 12]) @k where k = 5. Discounted cumulative gain (DCG) is a measure of ranking quality. In information retrieval, it is commonly used to measure effectiveness of web search engine algorithms or related applications. Using a graded relevance scale of documents in the result set, DCG measures the effectiveness or gain of a prediction based on its position in the result list. The gain is accumulated from the beginning of the result list to the end with the gain of each prediction result. DCG is calculated as:

$$DCG_k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

Where rel_i is the relevance of the result at position i. Since for each new user in test set, we can make a maximum of 5 predictions on the country of the first booking. The ground truth country is marked with relevance = 1, while the rest have relevance = 0. Therefore, in our case, to gain a better accuracy, we decide to make 5 predictions for those who have booked their first destination based on whether the value of date_first_booking is not empty, while for those whose date_first_booking value is empty, we predict their destination is 'NDF'. If we predict the accurate destination in our result list, then the DCG for this prediction equals 1 / log2(i + 1), where i is the position of the accurate destination in our prediction list. All the wrong guesses dont need to be accumulated since their relevance is 0 making the numerator of the fraction above equals 0. In the worst case, we cant give a right assumption in all 5 chances, the DCG will be 0; while in the best case, we predict the right result in the first position, then the DCG will be 1. Therefore, the DCG calculations are relative values on the interval 0.0 to 1.0.

To normalize the discounted cumulative gain, we need NDCG, whichG is calculated as:

$$nDCG_k = \frac{DCG_k}{IDCG}$$

IDCG is the ideal DCG for a given set of predictions. As we mentioned before, the maximum value of DCG is 1 when we predict the destination accurately and at the first try. So the final result of NDCG is the result of the sum of DCG of all test set prediction divided by the number of users in the test set.

Performance on Different Models

We use the above three model: Nearest Neighbors Classification, Gaussian Nave Bayes, Gradient Boosting Classification, with one-vs-the-rest(OvR) multiclass/multilabel strategy to predict the test dataset. As we mentioned in Feature Selection, we decide to use the following features: month, book_month, age, language, affiliate_provider, first_device_type, first_browser. While we compare the performance of different models, we randomly split our dataset from train_users.csv into 9:1 (train:validation). While before we submit our results to Kaggle, we use all data in train_users.csv for better performance to train data with different models and get the predictions of the test data separately. In this case the ratio of train with test is 171239:43673, almost 4:1.

model	nDCG of validation set	nDCG of test set	Time to predict validation label
Nearest Neighbors Classification	0.915984	0.92582	0:00:25.85
Gaussian Nave Bayes	0.92482	0.92680	0:00:01.16
Gradient Boosting Classification	0.92502	0.93328	0:00:08.85

Table 5. Results of classification with different models

In this experiment, we use K=25 in Nearest Neighbors Classification model, and use learning rate=0.05 and number of classes=11 in Gradient Boosting Classification. From the result, based on the nDCG in validation set, all three model pass 0.915 but the result of Gradient Boosting Classification outperforms other models subtly with the secondary speed of predicting the result. Since we use all data to train the models, the nDCG in test set is bigger than the one in validation set among these three models. Similarly, the performance of Gradient Boosting Classification in test set is still better than the remaining two models. From this result, we cant conclude that Nearest Neighbors Classification and Gaussian Nave Bayes fail, since based on the ranking of Kaggle and the subtle difference among Gradient Boosting Classification and themselves. However, with the help of more advanced algorithm, Gradient Boosting Classification is the best model for us to predict in which country a new user will make his or her first booking in Airbnb. By using this model, due to December 1st, 2015, our team is still No. 1 in this competition on Kaggle, and all the top three are our three team members(as Figure 1).

CONCLUSION

We propose to use a Gradient Boosting model (with deviance loss functions) compounding with the "one-vs-the-rest" strategy for such travel destination prediction task. In our case, the recommend system made with our model could best perform on the test set with a high matching nDCG value being 0.93328. The optimal parameter tuple that we figured out has the learning rate being 0.05, with 11 estimators to build decision trees up to 3 levels. Always making full ranks (up to 5) of prediction and ordering by the predicted certainty optimize our solution. The features that are most helpful to this prediction task including the following attributes from each record: month, booking month, user age, user language, the affiliate provider, user's first used device type and first used browser for logging into Airbnb's system.

ACKNOWLEDGEMENTS

We thank the University of California, San Diego, the Computer Science and Engineering department, and our CSE 255 class.

REFERENCES

1. Spatial Interaction Models. **http://tfresource.org/Spatial_Interaction_Models#Related_Content**
2. McRank: Learning to Rank Using Multiple Classification and Gradient Boosting. **http://machinelearning.wustl.edu/mlpapers/paper_files/NIPS2007_845.pdf**
3. Modelling tourist destination choice using a decision table induction algorithm. **http://epn.sagepub.com/content/35/9/1669.full.pdf+html**
4. Airbnb Recruiting: New User Bookings. **https://www.kaggle.com/c/airbnb-recruiting-new-user-bookings/data**
5. Feature Selection in Python with Scikit-Learn. **http://machinelearningmastery.com/feature-selection-in-python-with-scikit-learn/**
6. Feature Selection with Scikit-Learn. **http://scikit-learn.org/stable/modules/feature_selection.html**
7. The Optimality of Naive Bayes. **http://www.cs.unb.ca/profs/hzhang/publications/FLAIRS04ZhangH.pdf**
8. Nearest Neighbors Classification. **http://scikit-learn.org/stable/modules/neighbors.html**
9. A Detailed Introduction to K-Nearest Neighbor (KNN) Algorithm. **https://saravananthirumuruganathan.wordpress.com/2010/05/17/a-detailed-introduction-to-k-nearest-\neighbor-knn-algorithm/**
10. One Vs Rest Strategy **http://scikit-learn.org/stable/modules/multiclass.html**

11. Discounted cumulative gain in wikipedia.
https://en.wikipedia.org/wiki/Discounted_cumulative_gain#Normalized_DCG

12. Airbnb Recruiting: New User Bookings Evaluation.
https://www.kaggle.com/c/airbnb-recruiting-new-user-bookings/details/evaluation

13. Alexey Natekin1, and Alois Knoll2 Gradient boosting machines, a tutorial *Front Neurorobot. 2013; 7: 21*