CS 6375.501 MACHINE LEARNING

# PREDICTING AIRBNB NEW USER BOOKING

POJECT REPORT

Team:

Deeptha Parsi Diwakar (dpd140030)

Shivu Gururaj (sxg144730)

Sudharsanan Muralidharan (sxm149130)

## 1. Introduction

Online travel agencies have a huge amount of data, hence there is scope to utilize this data into making predictions so as to effectively enhance user experience and increase total bookings

This Airbnb Kaggle challenge revolves around predicting the destination of a new user. New users on Airbnb can book a place to stay in 34,000+ cities across 190+ countries. Hence, from their point of view, 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

## 2. Problem Definition and Algorithm

### 2.1 Task Definition

The task is 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”(No destination) label in the destination field. In the dataset, Airbnb provides a labeled training set of 213451 entries, and a test set of 62096 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 expected output of the entire task is to obtain a list of ranked destinations that a new user might make as the destination of his first booking.

**2.2 Algorithm Definition**

To accomplish this task,we have made use of 4 classification algorithms,namely

i) Naive Bayes

ii) Decision Trees

iii) Random Forests

iv) Gradient boosting

Below is a short description of each algorithm.

**Naive Bayes Algorithm:**

Naive Bayes is a simple probabilistic classifier based on applying Bayes' theorem (or Bayes's rule) with strong independence (naive) assumptions.

Explanation of Bayes's rule:

Bayes's rule: $P(H | E) = (P(E | H) \times P(H))/ P(E)$

The basic idea of Bayes's rule is that the outcome of a hypothesis or an event (H) can be predicted based on some evidences (E) that can be observed. From Bayes's rule, we have

(1) A priori probability of H or P(H): This is the probability of an event before the evidence is observed.

(2) A posterior probability of H or P(H | E): This is the probability of an event after the evidence is observed.

Example: To predict the chance or the probability of raining, we usually use some evidences such as the amount of dark cloud in the area.Let H be the event of raining and E be the evidence of dark cloud, then we have

P(raining | dark cloud) = (P(dark cloud | raining) x P(raining))/P(dark cloud)
Here, P(dark cloud | raining) is the probability that there is dark cloud when it rains. Of course, “dark cloud” could occur in many other events such as overcast day or forest fire, but we only consider “dark cloud” in the context of event “raining”. This probability can be obtained from historical data recorded by some meteorologists.

P(raining) is the priori probability of raining. This probability can be obtained from statistical record, for example, the number of rainy days throughout a year. P(dark cloud) is the probability of the evidence “dark cloud” occurring. Again, this can be obtained from the statistical records, but the evidence is not usually well recorded compared to the main event. Therefore, sometimes the full evidence, i.e., P(dark cloud), is hard to obtain.
Explanation of Naive Bayes:
As you can see from Example 1, we can predict an outcome of some events by observing some evidences. Generally, it is “better” to have more than one evidence to support the prediction of an event.
Typically, the more evidences we can gather, the better the classification accuracy can be obtained. However, the evidence must relate to the event (must make sense). For example, if you add an evidence of “earthquake” to Example 1, the above model might yield worse performance. This is since “raining” is not related to the evidence of “earthquake”, i.e., if there is an earthquake, it doesn't mean that it will rain.
Suppose we have more than one evidence for building our NB model, we could run into a problem of dependencies, i.e., some evidence may depend on one or more of

other evidences. For example, the evidence “dark cloud” directly depends on the evidence “high humidity”. However, including dependencies into the model will make it very complicated. This is because one evidence could depend on many other evidences. To make our life easier, we make an assumption that all evidences are independent of each other (this is why we call the model “naive”).

Bayes's rule for multiple evidences::
P(H | E1, E2, ..., En) = (P( E1, E2, ..., En | H) x P(H) )/ P(E1, E2, ..., En)
With the independence assumption, we can rewrite the Bayes's rule as follows:
P(H | E1, E2, ..., En) =( P( E1 | H) x P( E2 | H) x ... P( En | H) x P(H))/P(E1, E2, ..., En)

Decision Trees:
A decision tree is a simple representation for classifying examples. Assume that all of the features have finite discrete domains, and there is a single target feature called the classification. Each element of the domain of the classification is called a class. A decision tree or a classification tree is a tree in which each internal (non-leaf) node is labeled with an input feature. The arcs coming from a node labeled with a feature are labeled with each of the possible values of the feature. Each leaf of the tree is labeled with a class or a probability distribution over the classes.

A tree can be "learned" by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node has all the same value of the target variable, or when splitting no longer adds value to the predictions.

**Gradient Boosting Trees:**

The algorithm for Boosting Trees evolved from the application of boosting methods to regression trees. The general idea is to compute a sequence of (very) simple trees, where each successive tree is built for the prediction residuals of the preceding tree. As described in the General Classification and Regression Trees Introductory Overview, this method will build binary trees, i.e., partition the data into two samples at each split node. Now suppose that you were to limit the complexities of the trees to 3 nodes only: a root node and two child nodes, i.e., a single split. Thus, at each step of the boosting (boosting trees algorithm), a simple (best) partitioning of the data is determined, and the deviations of the observed values from the respective means (residuals for each partition) are computed. The next 3-node tree will then be fitted to those residuals, to find another partition that will further reduce the residual (error) variance for the data, given the preceding sequence of trees.

It can be shown that such "additive weighted expansions" of trees can eventually produce an excellent fit of the predicted values to the observed values, even if the specific nature of the relationships between the predictor variables and the dependent variable of interest is very complex (nonlinear in nature). Hence, the method of gradient boosting - fitting a weighted additive expansion of simple trees - represents a very general and powerful machine learning algorithm.

**Random Forests:**

Random Forests grows many classification trees. To classify a new object from an input vector, put the input vector down each of the trees in the forest. Each tree gives a classification, and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest).

Each tree is grown as follows:

If the number of cases in the training set is N, sample N cases at random - but with replacement, from the original data. This sample will be the training set for growing the tree.

If there are M input variables, a number m<<M is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing. Each tree is grown to the largest extent possible. There is no pruning.

How random forests work:

To understand and use the various options, further information about how they are computed is useful. Most of the options depend on two data objects generated by random forests.

When the training set for the current tree is drawn by sampling with replacement, about one-third of the cases are left out of the sample. This oob (out-of-bag) data is used to get a running unbiased estimate of the classification error as trees are added to the forest. It is also used to get estimates of variable importance.

After each tree is built, all of the data are run down the tree, and proximities are computed for each pair of cases. If two cases occupy the same terminal node, their proximity is increased by one. At the end of the run, the proximities are normalized by dividing by the number of trees. Proximities are used in replacing missing data, locating outliers, and producing illuminating low-dimensional views of the data

# 3. Experimental Evaluation

## 3.1 Methodology

Each of the classification algorithms were evaluated on the basis their accuracy.Accuracy is calculated by splitting the input data into training set and validation set.
Then, training set is used for learning,validation set for predicting. Now, by counting the correct classifications and dividing it by total number of inputs, we get the accuracy.

In our project, we made use SPARK Machine Learning Library (MLLib) to build the model and perform the classification tasks. Before this could be done, we had to analyze,preprocess the input data and convert it into LibSVM format.

Data set details:
The data set consists of the files train_users.csv - the training set of users and test_users.csv - the test set of users. The various attributes are:

- id: user id
- date_account_created: the date of account creation
- timestamp_first_active: timestamp of the first activity, note that it can be earlier than date_account_created or date_first_booking because a user can search before signing up
- date_first_booking: date of first booking
- gender
- age
- signup_method
- signup_flow: the page a user came to signup up from

- language: international language preference
- affiliate_channel: what kind of paid marketing
- affiliate_provider: where the marketing is e.g. google, craigslist, other
- first_affiliate_tracked: whats the first marketing the user interacted with before the signing up
- signup_app
- first_device_type
- first_browser

Class label

- country_destination: this is the target variable to predict

sessions.csv - web sessions log for users

- user_id: to be joined with the column 'id' in users table
- action
- action_type
- action_detail
- device_type
- secs_elapsed

**Data pre processing**

A major part of the our project was deciding the attributes which actually contribute towards the prediction and ways we can handle them in order to improve our accuracy. We did a lot of trial and error experiments with the attributes and finally decided to go with the below description.

Age : Age is a number. Our strategy is to represent it in intervals. So we convert age into vector form of a value in range between 1(inclusive) and 20 (inclusive). Then change the value to the resulting value based on above computation. We disregard the age values less than 5 and greater than 100. We consider unknown values as well.

Date_First_booking:   Instead of using the dates given, our strategy is to use this attribute in terms of season which can give better classification result and avoid over fitting. We decided to categorize into 4 seasons.

Country: We transformed the country representation from string to number . This will be our class label.

Date_Account_Creation - Date_First_Booking : Initially we evaluated considering Date_Account_Creation and Date_First_Booking as separate entries. T We subtracted the date of account creation and date of first booking so that we get a pattern in which a user usually performs the booking. In comparison , we got better results on merging these two entities.

Date_first_booking: if the values are null,then we disregard that row since it does not make sense to classify it without having first booking.

Language: since majority of the values were english and other languages had very less significance which would not contribute towards accuracy or prediction, we classified the language value just to be “en or non en”. This helped us to get better results.

First browser: We ran tests considering all the values for classification and considering only mozilla, safari ,chrome and others. Second test gave us better results because mozilla, safari ,chrome and others ,these 4 had maximum significance than the other parameters. Hence we decided to go with 2nd approach.

Signup app: Values other than web and mobile didn't contribute for getting better results. Hence we restricted the values only to web and mobile for signup app since other parameters had a negligible count.

Gender: If the values are unknown then we go ahead with classification with gender attribute taking the value as unknown because unknown gender means that the user does not wish to disclose gender.

We  are neglecting the user ID and timestamp column since they do not contribute anything towards the classification.

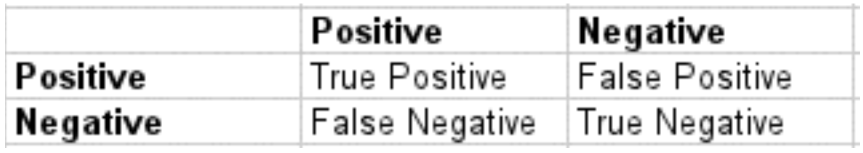
**Algorithm Evaluation**

We compute the following metrics to evaluate the performance of each algorithm:

1) Accuracy
2) F-Score
3) Precision
4) Recall

Accuracy: Simply said, accuracy is nothing but the proportion of correct results that a classifier achieved.

Once we have a model that can make robust predictions you need to decide whether it is a good enough model to solve your problem. Classification accuracy alone is typically not enough information to make this decision. A clean and unambiguous way to present the prediction results of a classifier is to use a use a confusion matrix (also called a contingency table) as shown in Fig 1. Several important results can be computed from this confusion matrix and these additional measures are also required to evaluate a classifier.



| | **Positive** | **Negative** |
|---|---|---|
| **Positive** | True Positive | False Positive |
| **Negative** | False Negative | True Negative |

Fig 1: Confusion Matrix

Precision: Precision is the number of True Positives divided by the number of True Positives and False Positives. Put another way, it is the number of positive predictions divided by the total number of positive class values predicted. It is also called the Positive Predictive Value (PPV).

Precision can be thought of as a measure of a classifiers exactness. A low precision can also indicate a large number of False Positives.

Recall: It is the number of True Positives divided by the number of True Positives and the number of False Negatives. Put another way it is the number of positive predictions divided by the number of positive class values in the test data. It is also called Sensitivity or the True Positive Rate.

Recall can be thought of as a measure of a classifiers completeness. A low recall indicates many False Negatives.

F Score: The F Score is the 2*((precision*recall)/(precision+recall)). It is also called the F1 Score or the F Measure. Put another way, the F score conveys the balance between the precision and the recall.

## 3.2 Results

| | Accuracy | Precision | Recall | F Score |
|---|---|---|---|---|
| **Naive Bayes** | 76.65% | 70.13% | 73.78% | 71.05% |
| **Decision Trees** | 70.66% | 68.32% | 68.84% | 68.84% |
| **Gradient Boosting** | 82.43% | 77.23% | 75.67% | 77.52% |
| **Random Forests** | 78.83% | 71.33% | 70.40% | 71.86% |

TABLE I

## 3.3 Discussion

As for the state-of-the-art methods of multi-classification, it's impossible to design one best model for all different datasets. A good model should be designed according to the actual properties and characters of the specific dataset. From the results we propose that Gradient boosting performs best compared to Random Forests, Decision trees and Naive Byes for our dataset.  Decision tree is seen to perform the least best, as expected since it is a weak classifier when compared to the ensemble techniques. This is also evident from table I .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.

## 4. Related Work

To accomplish the task, we referred onto many of the Spark MLLib example codes and other examples specific to our chosen classifiers. In order to improve our accuracy, we did some research on the different ways to select the required features from the input file. We also explored how different attributes can be combined to improve the accuracy.

## 5. Future Work

For the further work, we may consider processing the sessions.csv file and include the attributes present in it for training the sample. Changes in accuracy can be measured and we can figure out if sessions.csv has any contribution towards prediction.

## 6. Conclusion

After analyzing all the results obtained, we can conclude two things,one is that out of 4 different classifiers used, gradient boosting shows the best performance since it is an ensemble technique and a very powerful classifier. The second conclusion we can derive is that since in the train data, more than 80% of the destination was US,when we ran the algorithms on the test data,classifiers predicted the destination to be US. for most of instances .

## Bilbiography

1. Gradient boosting: http://xgboost.readthedocs.org/en/latest/model.html
2. Naive Bayes: http://suanpalm3.kmutnb.ac.th/teacher/FileDL/choochart82255418560.pdf
3. Random Forest: https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm
4. Spark MLLib guide: http://spark.apache.org/docs/latest/mllib-guide.html
5. Evaluation matrices : http://machinelearningmastery.com/classification-accuracy-is-not-enough-more-performance-measures-you-can-use/