Wee Kim Wee School of Communication and Information

18S1-SCI- K6312: Information Mining and Analysis

Group Project Report

Prediction for Airbnb New User Bookings

Wang Sixin (G1801764B)

Xia Xiaolong (G1801412A)

Zhao Hengrui (G1801739D)

Gao Wenhan (G1801851D)

Date: 8 November 2018

Nanyang Technological University

1 Introduction

In the 21st century, the development of internet provides a window for people to know more about this colorful world, which drives them to desire to go to different cities they like. Compared with the past, they're willing to spend money on travelling to enjoy their life with the improvement of people's living standards. This leads to rising demand for travelling house booking. In order to solve this problem, Airbnb put up with an innovative idea --- providing a platform for travelers to book empty rooms in hosts' houses. This attracts a lot of users because it can not only reduce the cost of travelers, but also make full use of empty space of hosts.

Now, there are more than 150 billion travelers and 640,000 hosts in Airbnb. Their serve region covers 190 countries and 65000 cities (Smith, 2018). With more and more users, Airbnb hosted a recruitment competition, which requires to predict which country or city that the users will like to choose for their first booking by machine learning methods.

Our project aims at using different algorithms (Logistic Regression, Tree, SVM, XGBoost) to develop models to identify users' behavior patterns. By comparing those results, we will choose the best one for predicting new users' first booking destination of Airbnb.

This project can help to predict the most desirable travel destination for guests, followed by a lot of benefits. First of all, it can assist Airbnb to customize personal travel plan for traveler. What's more, by providing the most desirable destination, it will motivate them to book within less time, which can increase booking rate. Beyond that, this kind of service can establish a good reputation of Airbnb so that they can attract more users. As for travelers, this can reduce their time for searching and deciding their destinations. As for hosts, this can help them to make strategies of location choice.

2 Solution

In this project, the process consists of five major steps, including data exploration, feature engineering&selection, model training, model optimization and lastly, discussion & limitation. The high-level workflow is shown in the figure 2-1.

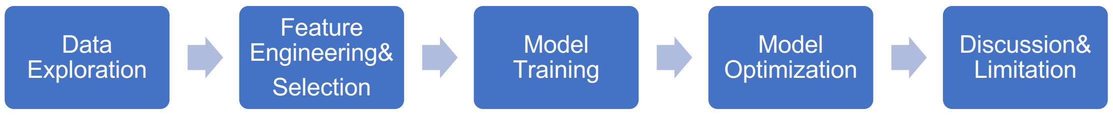


Figure 2-1 Project High-level Workflow

2.1 Data Exploration

The dataset we use in our project is from the Kaggle Airbnb Recruiting: Airbnb New User Bookings. This dataset concludes necessary information of users concerning their first booking destination countries in Airbnb. Moreover, these files comprise other related information such as user session and some summary statistics such as destination countries. All data is distributed in five ".csv" dataset:

(1) train_users_v2.csv - the training set: 213451 rows with 16 features
tests_user.csv - the test set: 62096 rows with 16 features

Parameters	**Description**
id	*user id*
date_account_created	*the date of account creation*
timestamp_first_active	*timestamp of the first activity*
date_first_booking	*date of first booking*
gender	*selected gender of the user*
age	*user's age*
signup_method	*user's method for sighing up*
signup_flow	*the web page where the user originated to sign up*
language	*user's language setting*
affiliate_channel	*paid market type*
affiliate_provider	*paid market name*
first_affiliate_tracked	*the first previews market of user before signing up*
signup_app	*the application user used to sign up*
first_device_type	*the device user used to login for the first time*
first_browser	*the browser user used to login for the first time*
country_destination	*the target variable to predict*

Table 2-1 Description of the train/test users dataset

(2) sessions.csv - web sessions log for users: 1048575 rows with 6 features

Parameters	**Description**
user_id	*user id*
action	*user's action*
action_type	*user's type of action*
action_detail	*user's action detail*
device_type	*user's device type*
secs_elapsed	*user's elapsed time*

Table 2-2 Description of the session dataset

(3) countries.csv - This file includes statistics of destination countries and their geometric information. There are ten countries in the dataset with seven different features, such as latitude, longitude and destination language.

(4) age_gender_bkts.csv – It is statistics of users' age bucket with gender, destination country, population is provided. There are 420 examples with five features.

We got some interesting information from these files, such as the chart of Registration Days vs Accounts creation in Figure 2-2. (“days = 0” represents the time of the first user creation in the data set). We found that the Accounts Created of the training dataset was concentrated on “days<1600”, and that of the test dataset was “days>1600”. Besides this, the general trend of user growth rate is increasing year by year.

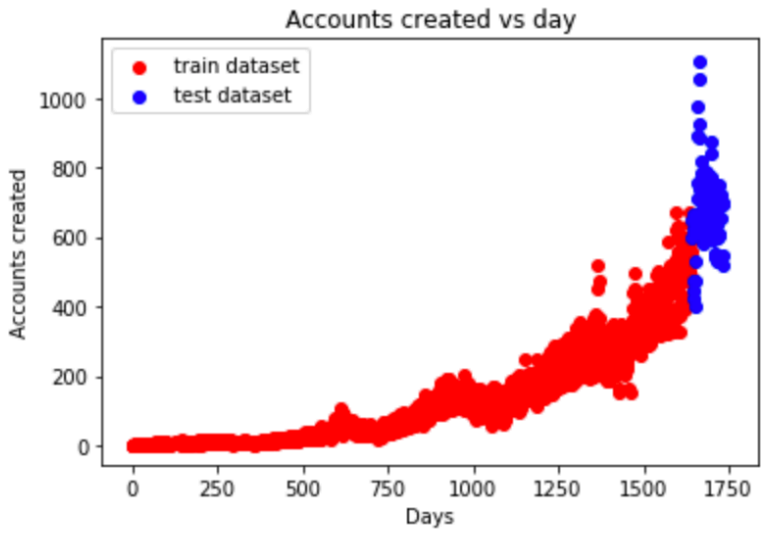


Figure 2-2 Registration days vs accounts created

User age distribution in Figure 2-3. One interesting finding was that users ageing from 25 to 45 were vast majority of the whole dataset.

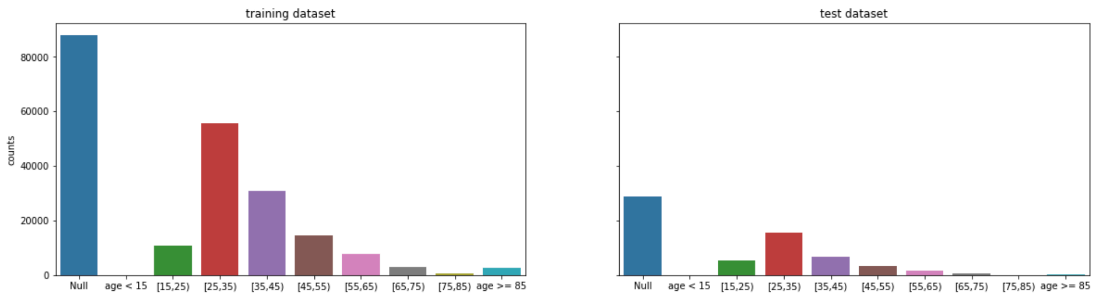


Figure 2-3 User age distribution

User gender distribution in Figure 2-4. It can be seen that there were more female users in the dataset than male users, and there were still a large number of users whose gender was unknown and very few “Other”.

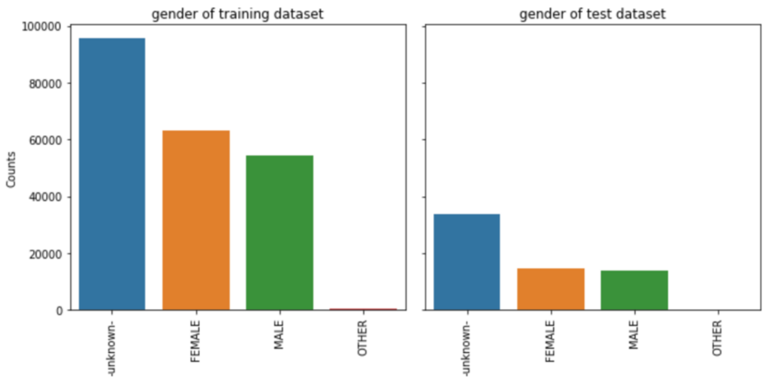


Figure 2-4 User gender distribution

Among the devices that users log in for the first time, the proportion of computer desktops is much higher than that of mobiles. As for computer desktops, Mac is more than Windows, and there are more iPhones in the mobiles than other Android devices.

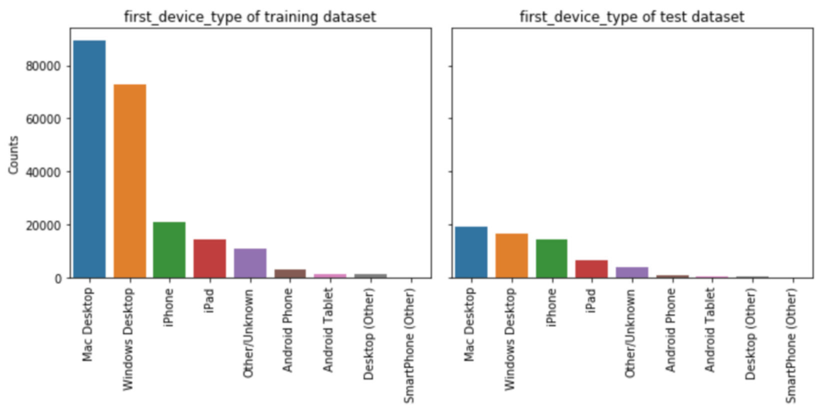


Figure 2-5 User device type distribution

As shown in Figure 2-6, Web is the most common signup method for users, reaching 175,000 times, far more than other signup apps such as iOS, Moweb or Android.

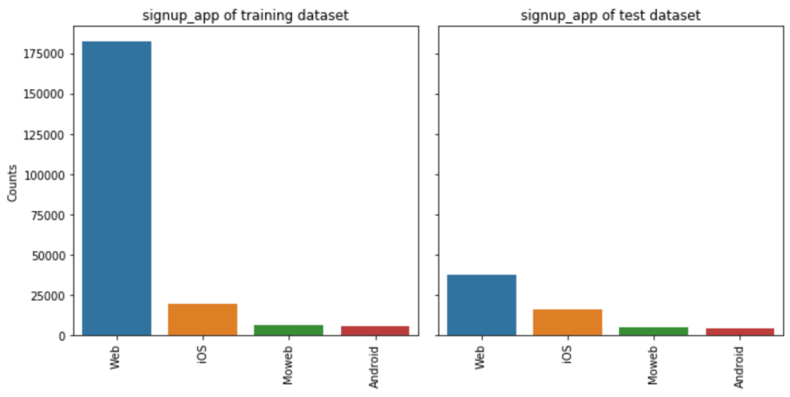


Figure 2-6 User signup app distribution

The statistics of the number of times each country is used as the first booking destination in Figure 2-7. Among all users, "NDF" had the largest share, reaching 120000 counts. In addition, the most popular destination was "US". Also, the first booking location for Other reached about 10,000 times. “NDF” means the user ended up not booking any accommodations and this seemed like almost half of the outcomes.

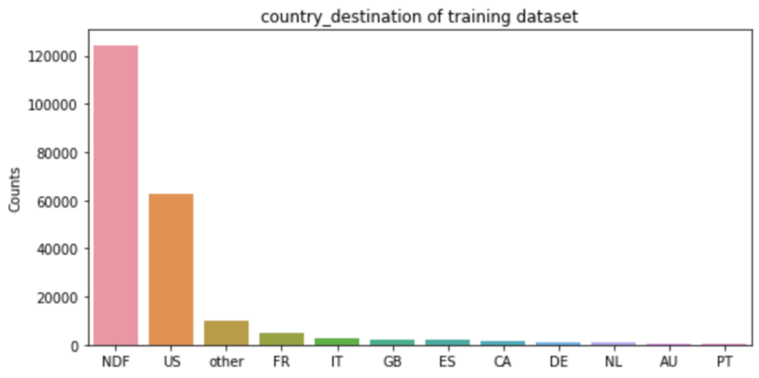


Figure 2-7 Booking counts of country destinations in training dataset

As shown in Figure 2-8, fewer bookings were made in weekends.

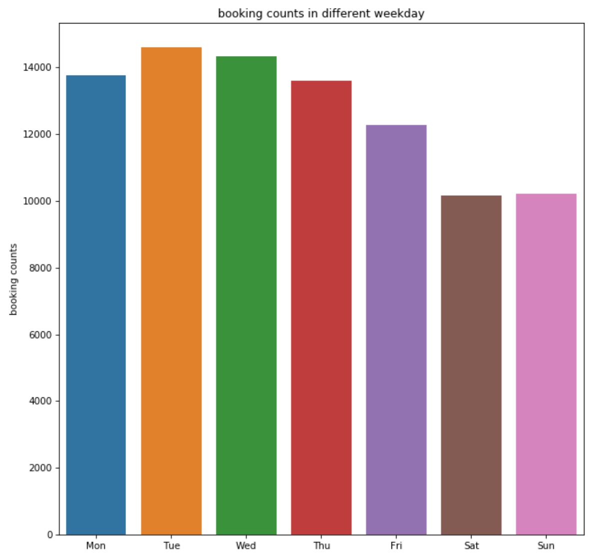


Figure 2-8 Booking counts of different weekdays in training dataset

In Figure 2-9, there was clear dip in between January to April, this emphasized seasonality of consumer interests in housing. Summer was the peak period of travelling. It also indicated planning need for future trips once the season change.

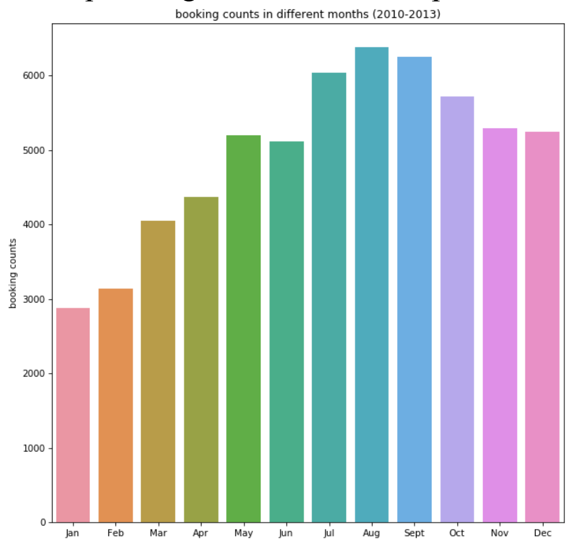
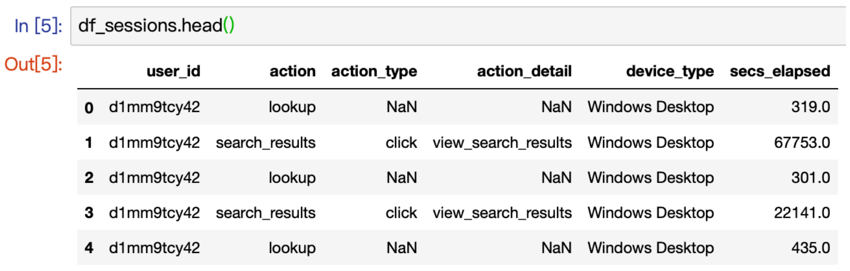


Figure 2-9 Booking counts of different months in training dataset (2010-2013)

2.2 Feature Engineering

Feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work which is fundamental to the application of machine learning. It is both difficult and expensive (Zabokrtsky, 2015). In this project, we first observed the dataset and processed it according to the characteristics of different features in each file, and finally integrated all the features.

(1) Process "sessions.csv".   View the dataset



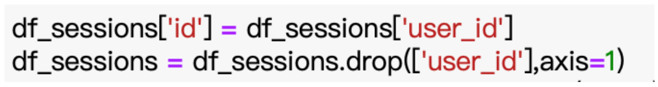
```
In [5]: df_sessions.head()
```

Out[5]:

	user_id	action	action_type	action_detail	device_type	secs_elapsed
0	d1mm9tcy42	lookup	NaN	NaN	Windows Desktop	319.0
1	d1mm9tcy42	search_results	click	view_search_results	Windows Desktop	67753.0
2	d1mm9tcy42	lookup	NaN	NaN	Windows Desktop	301.0
3	d1mm9tcy42	search_results	click	view_search_results	Windows Desktop	22141.0
4	d1mm9tcy42	lookup	NaN	NaN	Windows Desktop	435.0

Figure 2-10 dataset of the "sessions.csv"

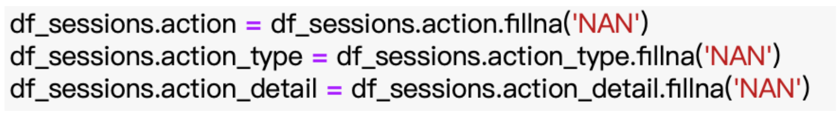
(2) Change "user_id" to "id"



```
df_sessions['id'] = df_sessions['user_id']
df_sessions = df_sessions.drop(['user_id'],axis=1)
```

Figure 2-11 Change "user_id" to "id"

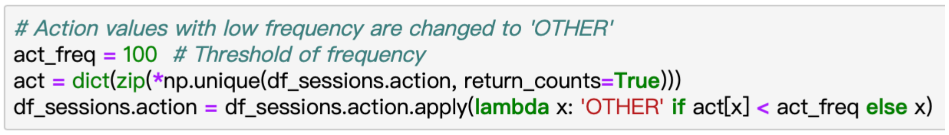
(3) Fill in missing values for "action",   "action_type",   "action_detail" with "-1".



```
df_sessions.action = df_sessions.action.fillna('NAN')
df_sessions.action_type = df_sessions.action_type.fillna('NAN')
df_sessions.action_detail = df_sessions.action_detail.fillna('NAN')
```

Figure 2-12 Fill in missing values

(4) Action values with low frequency are changed to 'OTHER'



```
# Action values with low frequency are changed to 'OTHER'
act_freq = 100  # Threshold of frequency
act = dict(zip(*np.unique(df_sessions.action, return_counts=True)))
df_sessions.action = df_sessions.action.apply(lambda x: 'OTHER' if act[x] < act_freq else x)
```

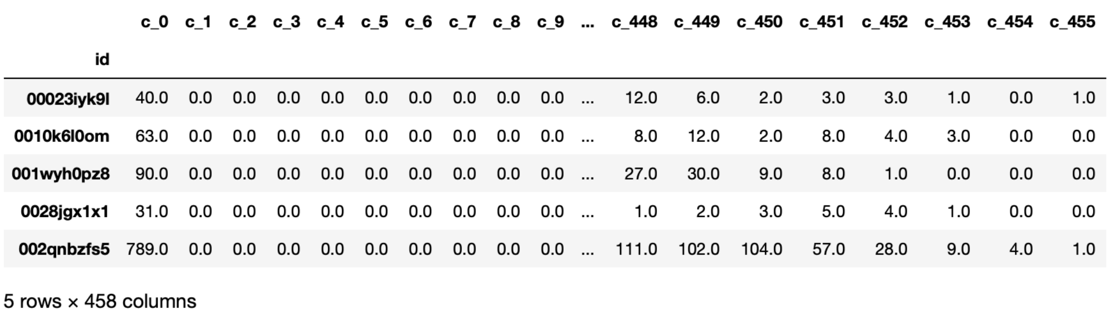
Figure 2-13 Change Action values with low frequency to 'OTHER'

(5) Process parameters of "sessions.csv"

One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction (Rakshith, 2017). In our process, we use one hot encoding to convert categorical data to integer data.

- Mapping non-numerical features to numerical data.
- "action": Count the number of times each user's total action occurs, the number of each action type, the average and the standard deviation.
- "action_detail": Count the number of times each user's total action_detail appears, the number of each action_detail type, the average and the standard deviation.
- "action_type": Count the number of times each user's total action_type appears, the number of each action_type type, the average, the standard deviation, and the total duration of the stay.
- "device_type": Count the number of times each user's total device_type appears, the number of each device_type type, the average value, and the standard deviation.

- "secs_elapsed": Fill the missing values with 0, and count the sum, average, standard deviation, median of each user's secs_elapsed time and the number of times secs_elapsed appears at various time.
- After the feature engineering process, the feature number becomes 458.



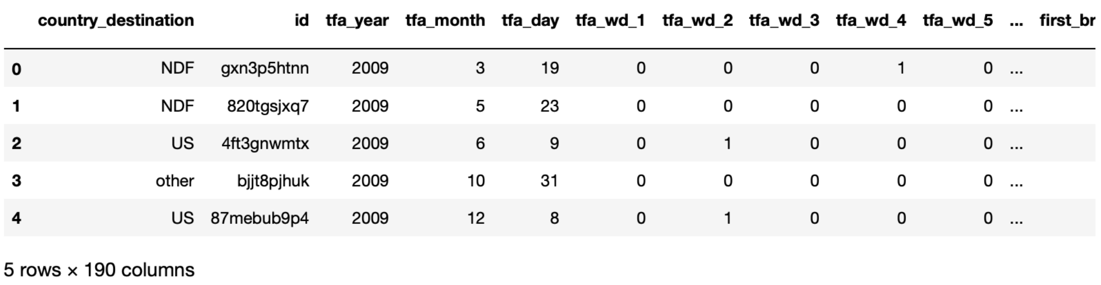
	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	...	c_448	c_449	c_450	c_451	c_452	c_453	c_454	c_455
id																			
00023iyk9l	40.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	12.0	6.0	2.0	3.0	3.0	1.0	0.0	1.0
0010k6l0om	63.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	8.0	12.0	2.0	8.0	4.0	3.0	0.0	0.0
001wyh0pz8	90.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	27.0	30.0	9.0	8.0	1.0	0.0	0.0	0.0
0028jgx1x1	31.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	2.0	3.0	5.0	4.0	1.0	0.0	0.0
002qnbzfs5	789.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	111.0	102.0	104.0	57.0	28.0	9.0	4.0	1.0

5 rows × 458 columns

Figure 2-14 Features of "session.csv" after feature engineering

(6) Process parameters in the "train_users_v2.csv", "tests_user.csv"

- "date_first_booking": This feature is completely empty in the test data, a large number of missing in the training data, delete the feature.
- "country_destination": Temporarily remove the value of the item in training data. (The model will be used to predict the country_destination, and then compare the result with the labels that have stored the country_destination to determine the performance of the model)
- Combine the train and test tables for processing.
- "timestamp_first_active": Convert it to datetime type and extract year, month, day, day of the week, season.
- "date_account_created": Convert to datetime type, extract the difference between the day, month, day, day of the week, season, calculation and timestamp_first_active, then divide it into "one day" and "other".
- Delete original features of timestamp_first_active and date_account_created.
- "age": The age distributed in the (1900, 2000) interval may be caused by the user's incorrect filling of the date of birth, which needs to be preprocessed; the age classification can be divided into NA, 15-25, 25-35, 35-45, 45-55, 55-65, 65-75, 75-85，abnormal（age>85 or age<15).
- After the feature engineering process, the feature number becomes 190.



	country_destination	id	tfa_year	tfa_month	tfa_day	tfa_wd_1	tfa_wd_2	tfa_wd_3	tfa_wd_4	tfa_wd_5	...	first_br
0	NDF	gxn3p5htnn	2009	3	19	0	0	0	1	0	...	
1	NDF	820tgsjxq7	2009	5	23	0	0	0	0	0	...	
2	US	4ft3gnwmtx	2009	6	9	0	1	0	0	0	...	
3	other	bjjt8pjhuk	2009	10	31	0	0	0	0	0	...	
4	US	87mebub9p4	2009	12	8	0	1	0	0	0	...	

5 rows × 190 columns

Figure 2-15 Features of train/test users after feature engineering

(7) Integrate all features

- Combine the features extracted by session and train and test.
- Delete “id”.
- Missing value handling for features without session data with “-1”.
- The number of null values per line as a new feature.

```
#Integrate the features extracted from the session
df_all = pd.merge(df, df_agg_sess, how='left')
df_all = df_all.drop(columns=['id']) #delete id
df_all = df_all.fillna(-2) | #Perform missing value processing on features without session data, fill it with -2

#Add a column to indicate how many null values ••are in each row, which is also a feature
df_all['all_null'] = np.array([sum(r<0) for r in df_all.values])
```

Figure 2-16 Integration of all features

- The total number of features is 648.

	tfa_year	tfa_month	tfa_day	tfa_wd_1	tfa_wd_2	tfa_wd_3	tfa_wd_4	tfa_wd_5	tfa_wd_6	tfa_wd_7	...	c_448	c_449	c_450	c_451	c_452	c_453	c_454	c_
0	2009	3	19	0	0	0	1	0	0	0	...	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0	
1	2009	5	23	0	0	0	0	0	1	0	...	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0	
2	2009	6	9	0	1	0	0	0	0	0	...	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0	
3	2009	10	31	0	0	0	0	0	1	0	...	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0	
4	2009	12	8	0	1	0	0	0	0	0	...	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0	

Figure 2-17 Total number of features

2.3 Feature Selection

Based on their importance obtained through XGBoost, we chose the top 250 features for subsequent training.

(1) Read feature file.

- There are 648 features in the xtrain.

```
xtrain = pd.read_csv("Airbnb_xtrain_v2.csv", index_col=0)
ytrain = pd.read_csv("Airbnb_ytrain_v2.csv", header=None)
xtrain.head()
```

	tfa_year	tfa_month	tfa_day	tfa_wd_1	tfa_wd_2	tfa_wd_3	tfa_wd_4	tfa_wd_5	tfa_wd_6	tfa_wd_7	...
0	2009	3	19	0	0	0	1	0	0	0	...
1	2009	5	23	0	0	0	0	0	1	0	...
2	2009	6	9	0	1	0	0	0	0	0	...
3	2009	10	31	0	0	0	0	0	1	0	...
4	2009	12	8	0	1	0	0	0	0	0	...

5 rows × 648 columns

Feature 2-16 Read the feature

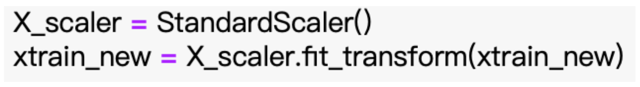
(2) Encode labels for the target variable.

```
ytrain_le = LabelEncoder().fit_transform(ytrain.values.ravel())
```

Figure 2-17 Label Encoding

- Before labels encoding: [AU, CA, DE, ES, FR, GB, IT, NDF, NL, PT, US, other]
- After labels encoding: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]

(3) Standardize the dataset



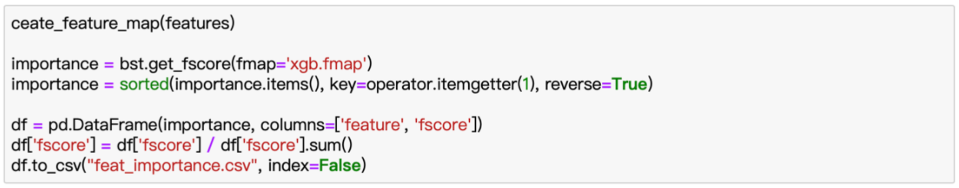
```
X_scaler = StandardScaler()
xtrain_new = X_scaler.fit_transform(xtrain_new)
```

Figure 2-18 Standardization of the dataset

(4) Feature selection using XGBoost

XGBoost is an improved algorithm based on the gradient boosting and can construct boosted trees efficiently and operate in parallel. Core of the algorithm is to optimize the value of objective function (Zheng, 2017). Advantage of using XGBoost is that it can mechanically deliver estimations of feature importance from a trained predictive model.

- Create feature_map and generate "feat_importance.csv"



```
ceate_feature_map(features)

importance = bst.get_fscore(fmap='xgb.fmap')
importance = sorted(importance.items(), key=operator.itemgetter(1), reverse=True)

df = pd.DataFrame(importance, columns=['feature', 'fscore'])
df['fscore'] = df['fscore'] / df['fscore'].sum()
df.to_csv("feat_importance.csv", index=False)
```

Figure 2-19 Create feature map

- Rank features with XGBoost, in order to source out most important features for future model enhancing and learning efficiency.Figure 2-20 is a snapshot of the top 50 important features. Features number from f187 to f648 are from session.csv. It's obvious that more features coming from session data rank in top importance. So, we can conclude that session data is highly important to the prediction.

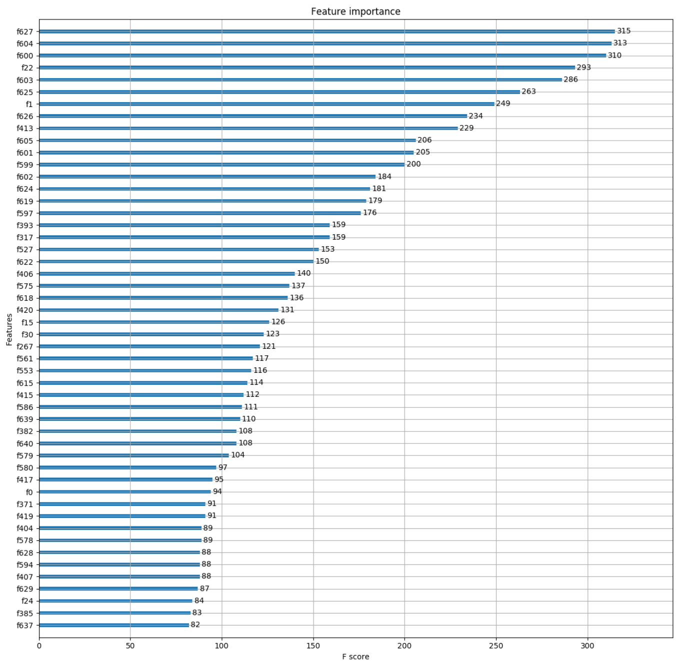


Figure 2-20 Top 50 features of the most importance

2.4 Task and Evaluation Method

2.4.1 Task

The task is to use machine learning algorithms to predict which country a new user will make the first booking on Airbnb.

2.4.2 Evaluation for Training and Testing

As Kaggle system adapts NDCG metric for evaluation of final prediction, we also used it during our model training process. Performances of models were measured by accuracy. To validate our models, we used 5-fold cross validation (80/20 split). In this way, each user record can be used for both training and validating the model.

$$DCG_n = \sum_{i=1}^{n} \frac{rel_i}{\log_2^{i+1}},$$

$$NDCG_n = \frac{DCG_n}{IDCG_n},$$

where rel_i is the relevance of the result at position ii;
$IDCG_k$ is the maximum possible (ideal) DCG for a given set of queries.
All NDCG calculations are relative values on the interval 0.0 to 1.0 (k=5)

2.5 Model Training

We first start with 50% of the processed dataset. Here are our experiments with four different types of models for our predictive task.

2.5.1 Logistic Regression

To start off, We built a logistic regression model based on 50% of the processed to set up a benchmark value. When the number of iterations increased, the overall accuracy of logistic regression model increased. But after iterated for over 20 rounds, the accuracy of logistic regression model stopped getting higher, as shown in Figure 2-21.

We also want to try another sub-dataset for the logistic regression model, to see if there is any improvement. For this attempt we decided to only adapt users' records which include session data. This new training dataset contained 76466 records (35.8% of original training set), all coming from January 2014 to June 2014. We tried to see whether better quality of session data improve model performance or not. It turned out this approach worked better than the previous one, even with less data records. Results were shown in Figure 2-22

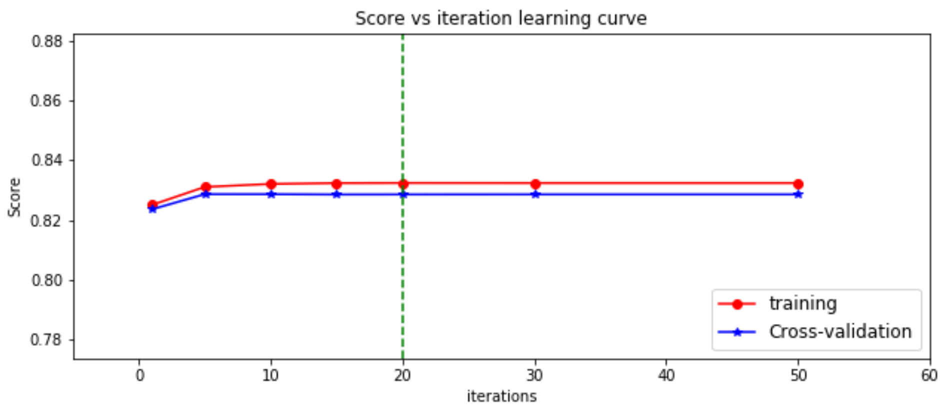


Figure 2-21 Learning curve of logistic regression
(with 50% of whole processed dataset)

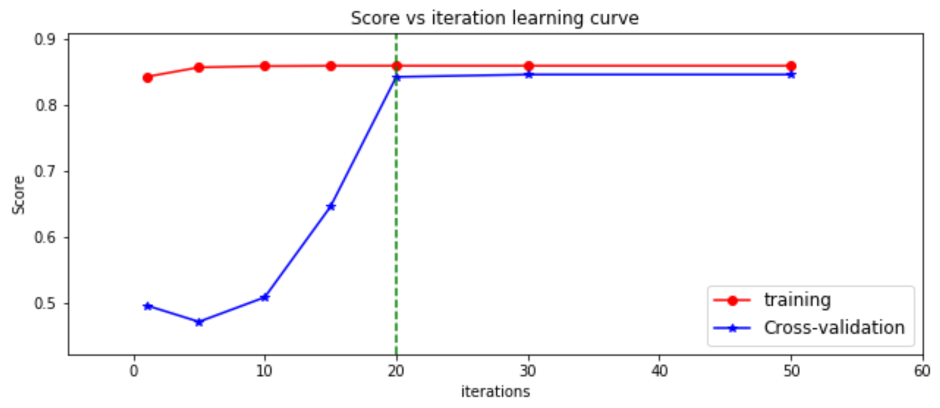


Figure 2-22 Learning curve of logistic regression
(with users' data which includes session record)

We adapted the new training dataset which includes session records to more algorithms to compare.

2.5.2 Tree

We used several different tree methods in experiments, including Decision Tree，Random Forest，AdaBoost，Bagging，Extra Tree，Gradient Boost. Performance of each tree methods is shown in Figure 2-23. Among them, Bagging was overfitting while Gradient Boosting performed good fits in both training and validation. This is an ensemble technique in which weak predictors are combined in building a better model. These weak predictors learn from misclassifications from the previous steps and better in the next steps by boosting the importance of incorrectly predicted data points.

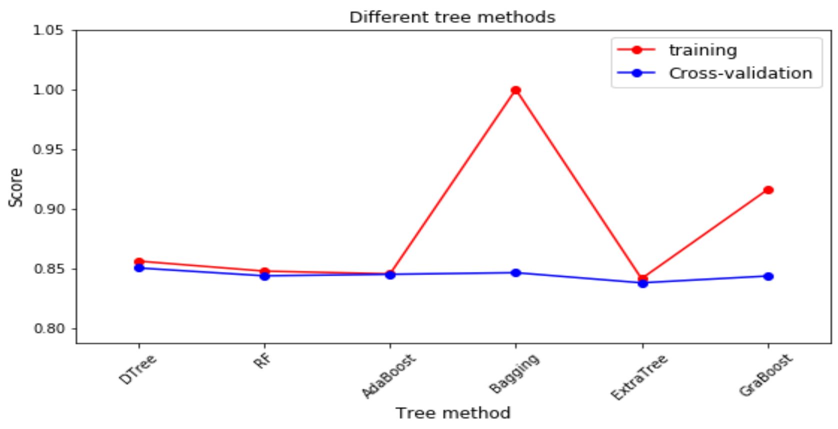


Figure 2-23 Performance of different tree methods

2.5.3 Support Vector Machine

SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall(Cortes, 1995). We adapted 4 types of SVM classifier in try-out: SVM-rbf, SVM-poly, SVM-linear and Linear SVC. Their performances are shown in Figure 2-24.

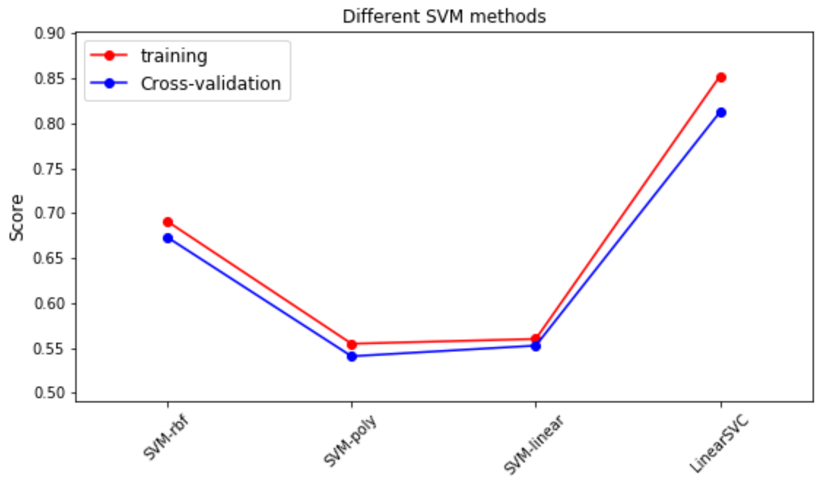


Figure 2-24 Performance of different SVM Classifier

2.5.4 XGBoost

As we earlier stated, XGBoost helps to boost trees algorithm and create several trees, intending to improve the accuracy by cross validation(Ulfsson, 2017). We set parameter “number of round” to 60 and other parameters to default. It yielded a validation NDCG score of 85.46%.

2.5.5 Comparison of results

Through comparison of all models (Figure 2-25), XGBoost yielded highest NDCG score in cross-validation, following by tree methods. So we would drill down and go for further test with different combination of parameters, in order to choose out the best accuracy.

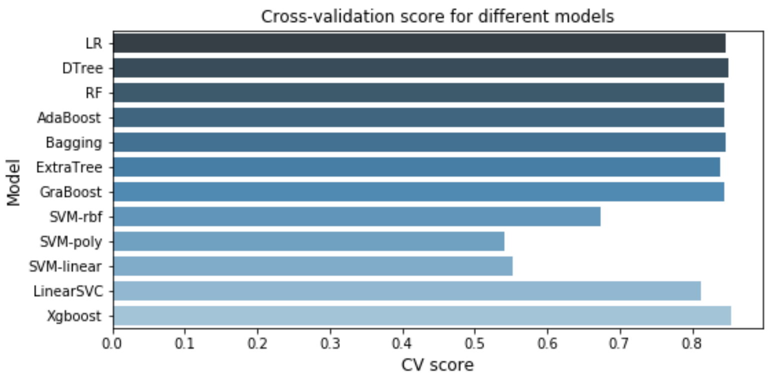


Figure 2-25 Cross-validation score of all models

2.6 Optimization

As the XGBoost algorithm performed better than other algorithms, we had a further study to specify the best performance parameters.
We first imported prediction of previously built XGBoost model (the one mentioned in 2.5.4) onto Kaggle.com. The outcome testing NDCG score was 82.01%, which variated quite a lot comparing to its validate NDCG score. Then we looked back into feature engineering and model training, trying to figure out possible reason. We found that there was no date overlapping between training dataset (users' records which includes session data, January 2014 to June 2014) and testing dataset(July 2014 to September 2014). So we decided to slightly change feature engineering: dropped "month_account_created" and "year_account_created". They benefitted training and validation performance, but comparative tests showed that dataset without these two features reported better testing NDCG score from Kaggle. Though there was slight drop-down in validation NDCG, we decided to use the revised user dataset since our objective was better prediction performance.
We first compared the NDCG-score of models in different num_round values to determine num_round parameter. Similar with the iteration number of logistic regressions, num_round is the number of rounds for boosting. Higher value will result in better training performance but will easily lead to overfitting.

Model	**Fixed Parameters**	**Comparison parameter**	**Validation NDCG**	**Testing NDCG**
1	max_depth=6, eta=0.1,	num_round =30	0.8546	0.8606
2	subsample=0.8,	num_round =60	0.8553	0.8617
	colsample_bytree=0.6,			
3	seed=2018, eval_metric= "merror", objective= "multi:softprob", nclass=12	num_round =90	0.8555	0.8571

Table 2-3 comparison of results of different num_round

The above table shows that model perform best when num_round equals to 60. The validation NDCG increased as num_round increased, but the testing NDCG increased first then decreased, which means the model is overfitting when num_round equals to 90.
After determined num_round, we compared models with different max_depth. Max_depth is the maximum depth of a tree. Increasing depth value will make the model more complex and more likely to overfit.

Model	Fixed Parameters	Comparison parameter	Validation NDCG	Testing NDCG
4	eta=0.1, num_round =60,	max_depth=4	0.8549	0.8481
5	subsample=0.8,	max_depth=5	0.8553	0.8645
6	colsample_bytree=0.6,	max_depth=6	0.8552	0.8567
	seed=2018, eval_metric=			
7	“merror”, objective=	max_depth=7	0.8550	0.8562
	"multi:softprob", nclass=12			

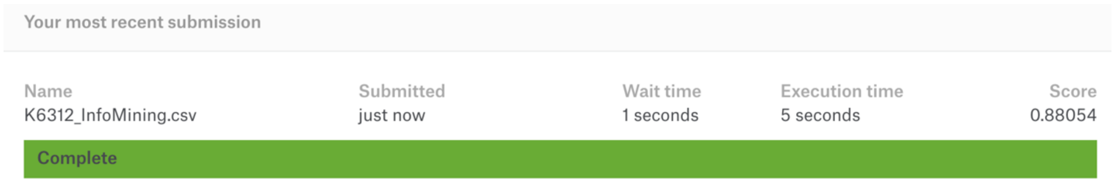
Table 2-4 Comparison of results of different max_depth

The results of three models show that model 5 performed best since it has highest testing NDCG. When max_depth is greater than 5, the model has a tendency to overfit.
In order to improve predicted accuracy, the top 400, top 300 and top 200 important features were used to train models again with the same settings of model 5, which is expected to achieve better results by removing less important features.

Model	Feature number	Validation NDCG	Testing NDCG
8	Full features	0.8553	0.8645
9	Top 400 features	0.8550	0.8803
10	Top 300 features	0.8549	0.8805
11	Top 200 features	0.8553	0.8801

Table 2-5 Comparison of results of different number of features

When the validation NDCG first falls and then rises, the testing NDCG is exactly the opposite, first rises and then falls. Removing unimportant features will increase testing accuracy, but removing too many features will affect the accuracy of the model. Besides, the time and memory consumption of model using a part of features is pretty low. In practical applications, determining the number of selected features is also an important issue.
The best performed-model – Model 10, got testing NDCG score as 0.88054 in Kaggle system.


Your most recent submission

Name	Submitted	Wait time	Execution time	Score
K6312_InfoMining.csv	just now	1 seconds	5 seconds	0.88054

Complete

Figure 2-26 Testing NDCG score reported by Kaggle

3 Discussion

Based on the Exploratory Data Analysis:

The largest share of Airbnb users lies in middle-aged (30-59), followed by the youth (under 30) and lastly the senior (above 60). This is a good reference for distributing marketing resource, middle-aged users should be priority.

Users on mobile devices are much more likely to fall into NDF outcome (not booking any accommodation) than web users, which means Web users more likely to book. This could indicate need for improving user experience on mobile devices.

58% of the first visits are via Apple products (including iPhone, iPad, Mac in top 5 devices). On the other hand, bookings are more likely made on desktop devices. Generally speaking, Airbnb needs to increase attractiveness of android platforms, in order to capture this market share. They could receive a small amount of credits to offset their first booking when downloading apps or booking through Android platforms to capture this market segment.

People who haven't filled out their information such as Age and Gender are the least likely to book Airbnb. This is likely due to the fact that they are only in the exploration stage. Maximum marketing effort must be aimed at these people and the most enticing yet affordable offers must be made to them to increase their rate of conversion.

Discussion on algorithms and prediction:

We can see from results presented above that the XGBoost algorithm performs best compared with the other three algorithms, which also ranks first in the cross-validation scores of all models that are built by integrating different parameters. To improve our output, we tried following three steps to specify the most suitable parameters.

We found that when the NDCG-score is highest when num_round equals to 60 compared with 30 and 90. Through adjusting max_depth of the tree, we chose model 5 with max_depth equaling to 6 to get a better performance and avoid overfitting. At last, we compared the performance with top 250 related features and full features. It shows that models using full features performs slightly better.

Based on the predicted results, the platform can personalize the recommendation information to different users to improve the booking rate.

4 Limitations and further improvement

Reflect on our project, there are some limitations need to be pointed out.

Aside with advantages of adapting user data and session data of 2014 for model training (benefits stated aforehand), there are also some disadvantages. The main problem is that we cannot use the majority of users if we want to efficiently make use of the session dataset since no session data in 2010-2013. Another problem is only considered a problem because of the limitations brought by the session data coverage and that is the fact that we have no seasonal overlap with the training and test set of users. These lead to lack of information for accurate prediction. The accuracy of prediction can be improved by providing more session data for a larger number of users spanning over a longer.

In terms of data quality, most of users' booking destinations in training dataset lie in "NDF" and "US". There might be challenge in predicting destinations in other countries.

Due to limitation on time and computation resources, we only use part of records for model training. But the accuracy will increase if we can use all the dataset. Same reasons, some more complex models are not adapted such as stacking. Though the prediction might perform better with stacking model to stack multiple classifiers, this is more complex and need more time to adjust to many different parameters. At last, considering the characters of dataset of the project-large volume and diverse features, it may be more suitable to train data with deep learning algorithms, which was not tried due to hardware conditions and time constraints. All these could be improved by deploying Cloud computing technique.

Exploring on this project enhanced our understanding of information mining and opened our eyes on how often the predicting algorithms are used in internet companies to customize personal marketing plans and how these technologies can benefit users in a more effective way. There is a bright future for machine learning.

Reference

[1] Zabokrtsky, Z. (2015). Feature engineering in machine learning. *Institute of Formal and Applied Linguistics, Charles University in Prague*.

[2] Zheng, H., Yuan, J., & Chen, L. (2017). Short-term load forecasting using EMD-LSTM neural networks with a Xgboost algorithm for feature importance evaluation. *Energies*, *10*(8), 1168.

[3] Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794). ACM.

[4] Smith, C. (2018). 100 Amazing Airbnb Statistics. Retrieved from https://expandedramblings.com/index.php/airbnb-statistics/

[5] Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3), 273-297.

[6] Ulfsson, H. (2017). Predicting Airbnb user's desired travel destinations.