Machine Learning Engineer Nanodegree

Capstone Project

Airbnb New User Booking Dataset

Pin-Yi Tseng October 13th, 2018

I. Definition

Project Overview

The distance between countries are becoming shorter due to faster transportation like airplanes and trains; on the other hand, thanks for broadband universal service, people tend to share their daily life, experiences, special events, etc. on Facebook or Instagram. Instead of joining in traveling agency, more and more people get their bag, pick up destinations and take off their adventures by themselves or with a small group of their friends. Companies like Airbnb, Booking.com, Agoda, trivago, and so on booming nowadays. People tend to continuously use their familiar and trust website when booking their hostels, especially when they become VIPs after many purchases. Therefore, the way to correctly predict the destination, providing related service or recommendation and attract people when they make their first consumption is the critical point for getting long-term customers.

Airbnb has become a global platform that connects travelers and hosts from over 34,000 cities. As such, it has collected a diverse set of dataset about users which can be utilized to predict patterns about its future users and provide them with customized suggestions to serve Airbnb's customers Airbnb had posted this on Kaggle as a Recruitment Challenge. Using user data could help organizations increase metrics such as sales, user experience, customer retention, and customer satisfaction. Apply Machine Learning methodology can help the organization to reveal the mutual effect between different events And furthermore, making the prediction using these data. The motivation for pursuing this project is to understand how to work on real-world datasets and challenges that companies like Airbnb consider to be important and valuable for their companies and learn to provide similar value for organizations that I Work with in the future.

The related work using machine learning method for recommendation system has been done in Book recommending using text categorization with extracted information. This paper focus on content-based recommender systems suggest documents, items, and services to users based on learning a profile of the user from rated examples containing information about the given items. Text categorization methods are very useful for this task but generally rely on unstructured text. We have developed a book-recommending system that utilizes semi-structured information about items gathered from the web using simple information extraction techniques. Initial experimental results demonstrate that this approach can produce fairly accurate recommendations.

Problem Statement

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.

Using the data from Airbnb New User Bookings dataset, the challenge is to predict the destination of choice for the users' first booking. This data includes demographics of users and their session data. The model will utilize these demographics and session data to make models that can predict the destinations.

In this project, I plan to use Machine Learning Techniques to predict in which country a new user will make their first booking on Airbnb. This project will involve data cleaning, data exploration using visualizations, and testing various algorithms for classification for the same.

Evaluation Metrics

Since this is a Kaggle Challenge, we already have an evaluation metric, that is the NDCG (Normalized Discounted Cumulative Gain)

Original method: For each new user, we are to 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.

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

$$nDCG = \frac{DCG_k}{IDCG_k}$$

where $gain_i$ is the gain of the result at position i and k=5..

For example, if for a particular user the destination is FR, then the predictions become:

[FR] gives a $NDCG = \frac{2^1 - 1}{log_2(1+1)} = 1.0$

[US, FR] gives a $DCG = \frac{2^{01} - 1}{log_2(1+1)} + \frac{2^1 - 1}{log_2(2+1)} = \frac{1}{1.58496} = 0.6309$

Modified Evaluation Matrics

Improved evaluation matrics based on http://dalelane.co.uk/blog/?p=3403 was implemented on **V. Conclusion: Improvement of NDCG Scoring Function** Since we only have 10 target countries in our dataset, if we want to put another gain for different prediction, some problems will appear: Some countries locate very far away from others, so when predicting, the other possible candidates will force the final score to be lowered since no matter what they predict, the predictions are very far from the target.

The reason why this metric is the optimal choice there is that based on the function:

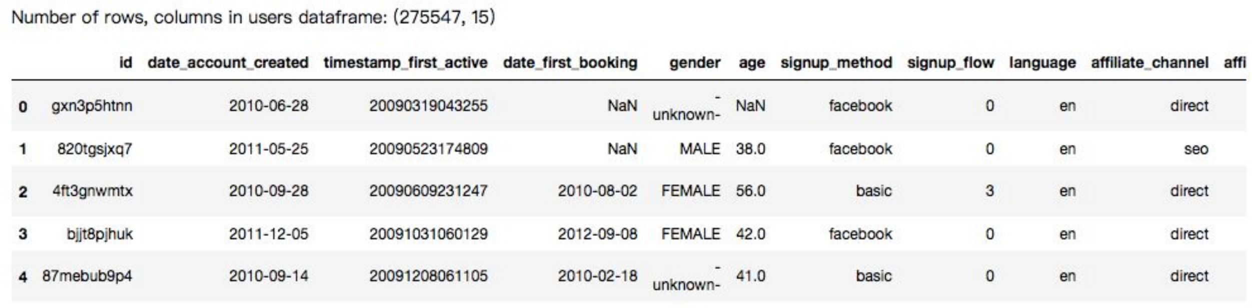
$$DCG_k = \sum_{i=1}^{k} \frac{2^{gain_i - 1}}{log_2(i+1)}$$

The ground truth country is marked with relevance = 1, while the rest have relevance = 0.. It looks like we only put static gain for the right answer; however, the gain will decrease if the right answer is not Your First Prediction. This design takes the advantage of putting different gain for different; on the other hand, by doing this, the outcome won't be impacted if there are only limited destination as our targets.

II. Analysis

Data Exploration

The train users data file has 213451 rows, where each rows describes 15 features about the user. The target variable is country_destination.



Number of rows, columns in users dataframe: (275547, 15)

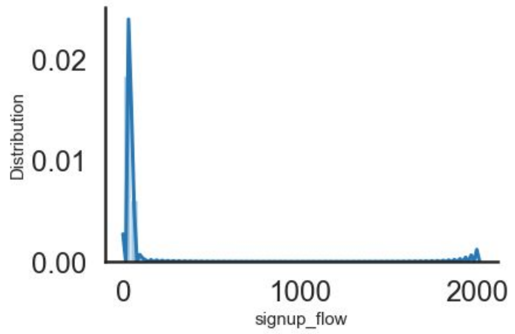
	id	date_account_created	timestamp_first_active	date_first_booking	gender	age	signup_method	signup_flow	language	affiliate_channel	affi
0	gxn3p5htnn	2010-06-28	20090319043255	NaN	-unknown-	NaN	facebook	0	en	direct	
1	820tgsjxq7	2011-05-25	20090523174809	NaN	MALE	38.0	facebook	0	en	seo	
2	4ft3gnwmtx	2010-09-28	20090609231247	2010-08-02	FEMALE	56.0	basic	3	en	direct	
3	bjjt8pjhuk	2011-12-05	20091031060129	2012-09-08	FEMALE	42.0	facebook	0	en	direct	
4	87mebub9p4	2010-09-14	20091208061105	2010-02-18	-unknown-	41.0	basic	0	en	direct	

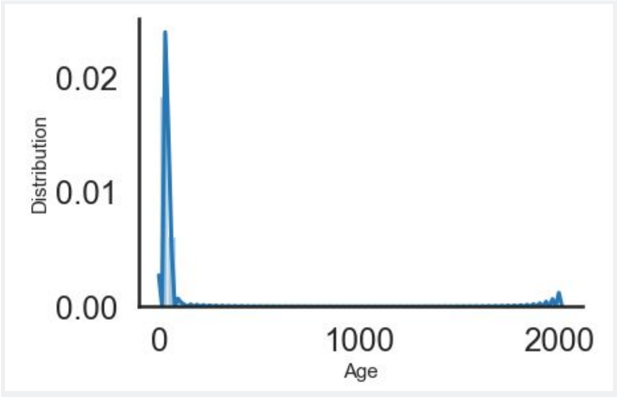
train_users/test_users

- 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
- country_destination: this is the **target** variable you are to predict
- sessions.csv - web sessions log for users

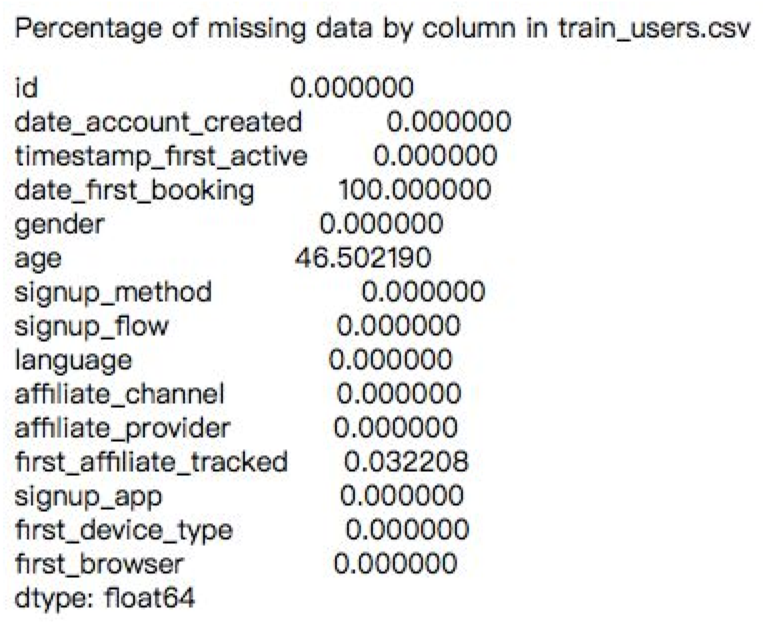
Because in raw data Age and signup_flow are the only numerical information I can show the distribution: both distributions are too narrowed and need to do some preprocessing for transformation. In addition, those categorical data will be well-handled to become useful ones in III. Methodology: Data Preprocessing





Exploratory Visualization

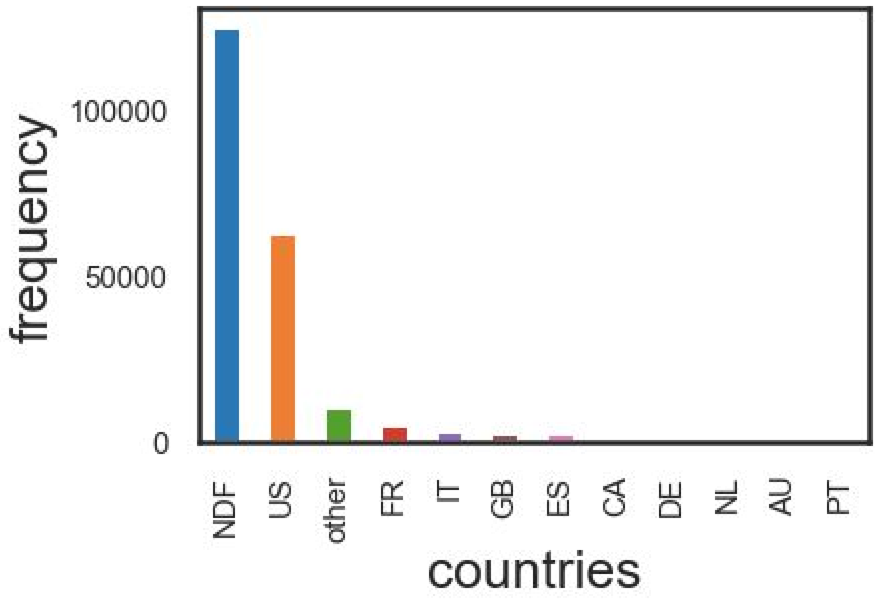
Before starting, it was important to find out the percentage of missing values. It was found that date_first_booking not available for the testing dataset. Also, I inferred that date_first_booking is only available for users who successfully booked a destination and since NDF is the most frequent, it is implicit that date_first_booking would be missing. So, I decided to drop this feature.



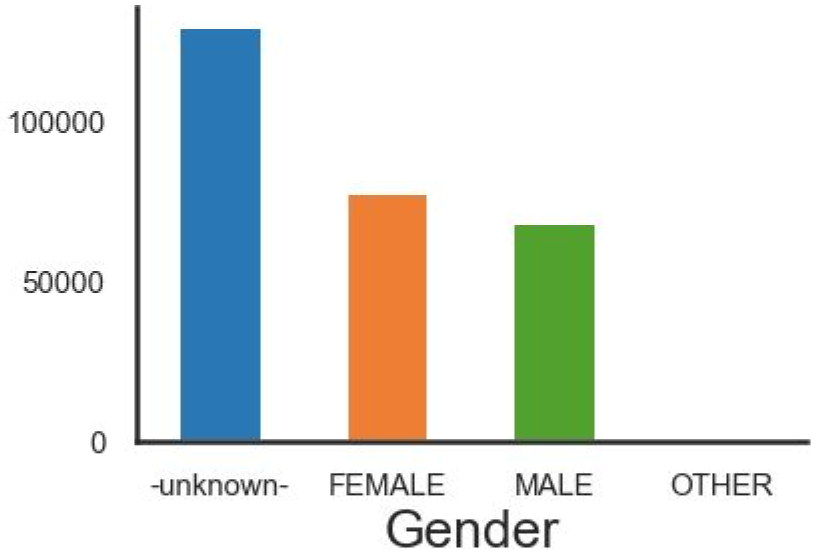
```
Percentage of missing data by column in train_users.csv

id                          0.000000
date_account_created        0.000000
timestamp_first_active      0.000000
date_first_booking        100.000000
gender                      0.000000
age                        46.502190
signup_method               0.000000
signup_flow                 0.000000
language                    0.000000
affiliate_channel           0.000000
affiliate_provider          0.000000
first_affiliate_tracked     0.032208
signup_app                  0.000000
first_device_type           0.000000
first_browser               0.000000
dtype: float64
```

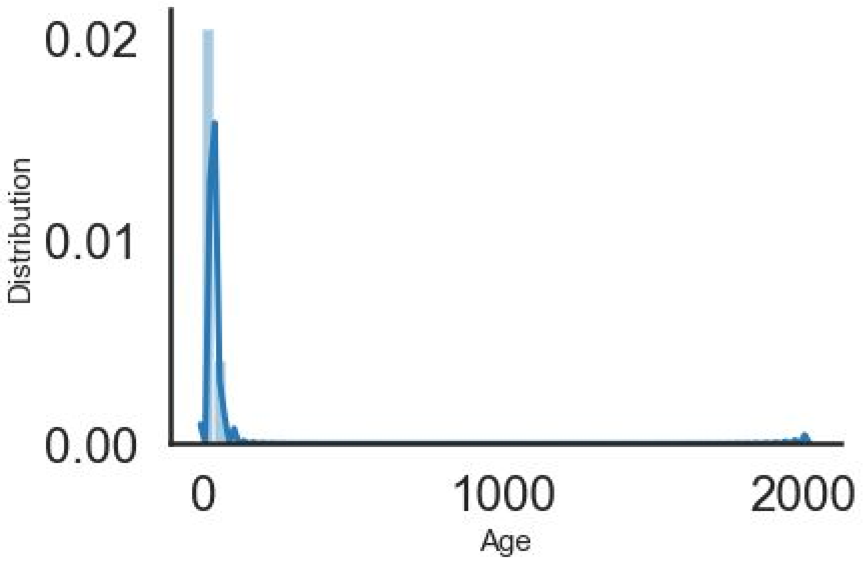
The plot below shows the frequency of the target variable. There are 12 possible outcomes of the destination country: 'US', 'FR', 'CA', 'GB', 'ES', 'IT', 'PT', 'NL','DE', 'AU', 'NDF' (no destination found), and 'other'. Please note that 'NDF' is different from 'other' because 'other' means there was a booking, but is to a country not included in the list, while 'NDF' means there wasn't a booking.



After solving the date_first_booking feature, we explored the gender data. It was found that most of the data was not filled by the user and was set as -unknown- in the dataset.



Then, it was found that the Age was also having erroneous values such as 2014, and negative values etc, I decided to set values < 15 and > 100 as NaN. The visualization is displayed below:



Algorithms and Techniques

Algorithms and Techniques

Given that this problem was a multi-class supervised classification problem, we decided to use Decision Trees. Decision trees are powerful in predicting a target by learning simle decision rules learnt using training data. They can handle numerical and categorical data, missing data, along with multiple target classes.

Although decision trees are good, I decided to use ensemble methods to improve the predictive power. These algorithms were, the random forest classifier and XGBoost.

Random Forest Classifier fits number of decision trees on subsamples of a dataset and averages the results.

A random forest is a collection of random decision trees. In which at each node you will randomly draw a subset of features and the decision tree will predict the classification. Then the same is done with several trees and bagged. This ensemble method will reduce overfitting and provide good classification. The bias-variance trade-off for this algorithm is good and therefore the possibility of overfitting is drastically reduced.

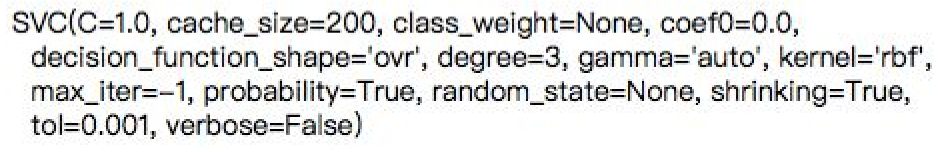
GridSearchCV There are many parameters which can be tuned to get a model with very good accuracy. The grid search technique helps us to generate a grid from parameters and use these multiple combinations of parameters and then select the best one which provides best cross-validation results. Although, it is important to note here that the grid search technique only performs hyper-parameter tuning, i.e. the parameters that are not directly learnt within the estimator. In scikit-learn, we have GridSearchCV which uses an estimator and a set of hyper-parameters to exhaustively generates candidates from set and generate a validation score and we can then obtain the most optimal parameters systematically. We use this technique to get the best hyperparameters for our Random Forest ensemble.

The **XGBoost** algorithm was choosen after research into Kaggle Competitions and it was found out that it proves extremely effective in such arenas. XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It produces an ensemble of weak decision tree learners via additive training (boosting). XGBoost is short for Extreme Gradient Boosting. This is based on Gradient boosted trees. Boosted trees are basically an ensemble of decision trees which are fit sequentially so that each new tree makes up for errors in the previously existing set of trees. The model is ”boosted” by focusing new additions on correcting the residual errors of the last version of the model. Then you take an approximate step in the gradient direction by training a model to predict the gradient given the data. XGBoost algorithm tuning is a tricky process. Heavy computation power is required for such level of tuning.

Benchmark - SVM

The given dataset is a typical supervised learning problem and so far SVM is the algorithm I used mot of time. So I will pick Support Vector Machine (SVM) as a benchmark. In addition, if we set `svc.probability = True`, then this sklearn module allows us to predict each destination with their probability. Then, we choose the top 5 possibility as our results, and I get score of `0.7647` in benchmark model.

Used Model:



```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=True, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

III. Methodology

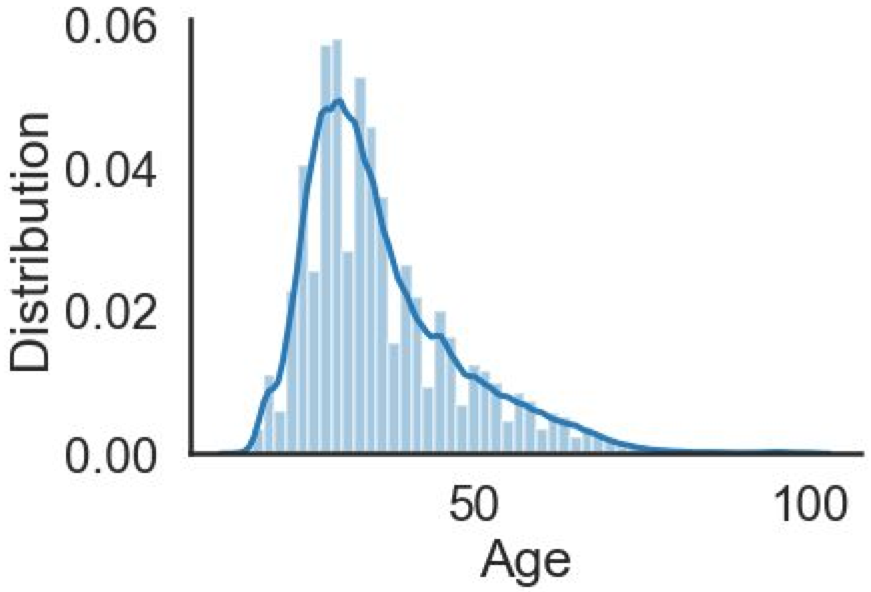
Data Preprocessing

Most features are categorical and very difficult to find a way of measuring them in numbers. Therefore, I use a lot of one-hot-encoding methodologies to do this. Take `age` for example, Intuitively, I believe the way people choose their destination country must have something to do with their age. However, despite we remove some outlier range of age, and made the distribution more normalized, we still have trouble to define this feature: almost every user have different ages. Here I defined the interval for ages and use one-hot-encoding (OHE) to transform these features.

Another example is `date time` related data. This may seem useless at first, but I think we weather plays an important role when you decide the destination, so I try to extract the `month` information from these and represented it with `season`. But I feel it's kind of a waste to throw out the rest of information: maybe they're useful in some way I don't realize at the first time. So I try to separate it into several parts: year, month, day, numbers of week in one year and weekday. By doing this. I think I can take the most advantages from this data instead of removing them.

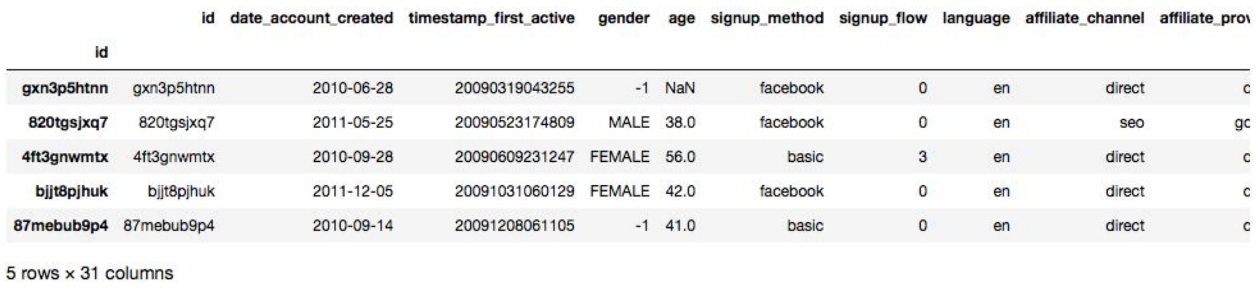
Age

As discussed above, we set the ages below 15 and above 100 to NaN. Afterwards, we One Hot Encoded the age in intervals of 5. After removing these values the distribution of age looks like this:



Creating Features And One-hot-encoding(OHE)

- One-hot encoding of the age according to intervals of 5: People in different age ranges must have different idea when choosing traveling place.



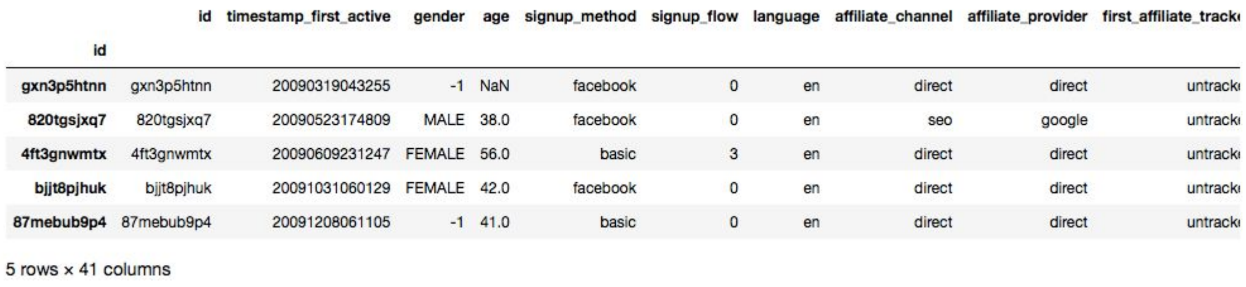
	id	date_account_created	timestamp_first_active	gender	age	signup_method	signup_flow	language	affiliate_channel	affiliate_prov
id										
gxn3p5htnn	gxn3p5htnn	2010-06-28	20090319043255	-1	NaN	facebook	0	en	direct	c
820tgsjxq7	820tgsjxq7	2011-05-25	20090523174809	MALE	38.0	facebook	0	en	seo	gc
4ft3gnwmtx	4ft3gnwmtx	2010-09-28	20090609231247	FEMALE	56.0	basic	3	en	direct	c
bjjt8pjhuk	bjjt8pjhuk	2011-12-05	20091031060129	FEMALE	42.0	facebook	0	en	direct	c
87mebub9p4	87mebub9p4	2010-09-14	20091208061105	-1	41.0	basic	0	en	direct	c

5 rows × 31 columns

Since column related to date can only provide limited use when analyzing, here I decide to separate date related column into several pieces, such as year, month, day, numbers of week in one year and weekday. By doing this, new features may give us more information about how people decide to choose their countries, maybe this is affected by months (trying to avoid extremely cold, hot or dangerous weather conditions); or maybe some people prefer doing something on certain weekday. For example, during Monday blue, people may have an urgent feel to some natural places.

In addition, the format for date_account_created and timestamp_first_active are different; which means by doing this, we can transform those 2 features into some others with the same format.

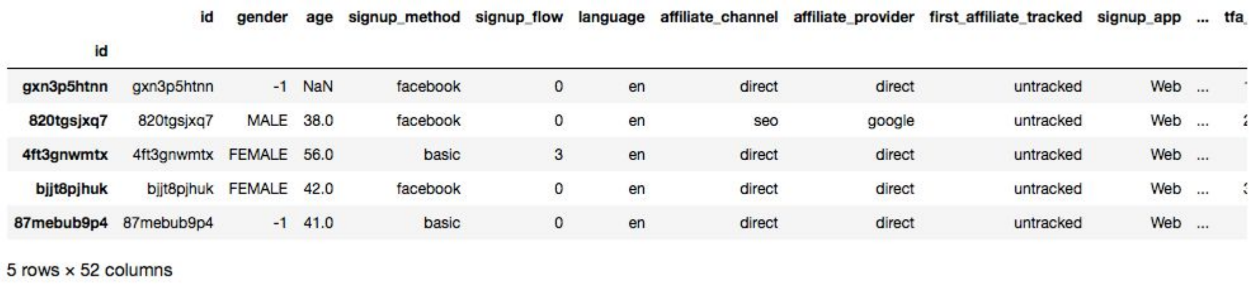
- date_account_created: We used this date column and split it into dac_y, dac_m, dac_d, dac_wn (week number), dac_w_{} (weekday, it was further split into each day).



	id	timestamp_first_active	gender	age	signup_method	signup_flow	language	affiliate_channel	affiliate_provider	first_affiliate_track
id										
gxn3p5htnn	gxn3p5htnn	20090319043255	-1	NaN	facebook	0	en	direct	direct	untrack
820tgsjxq7	820tgsjxq7	20090523174809	MALE	38.0	facebook	0	en	seo	google	untrack
4ft3gnwmtx	4ft3gnwmtx	20090609231247	FEMALE	56.0	basic	3	en	direct	direct	untrack
bjjt8pjhuk	bjjt8pjhuk	20091031060129	FEMALE	42.0	facebook	0	en	direct	direct	untrack
87mebub9p4	87mebub9p4	20091208061105	-1	41.0	basic	0	en	direct	direct	untrack

5 rows × 41 columns

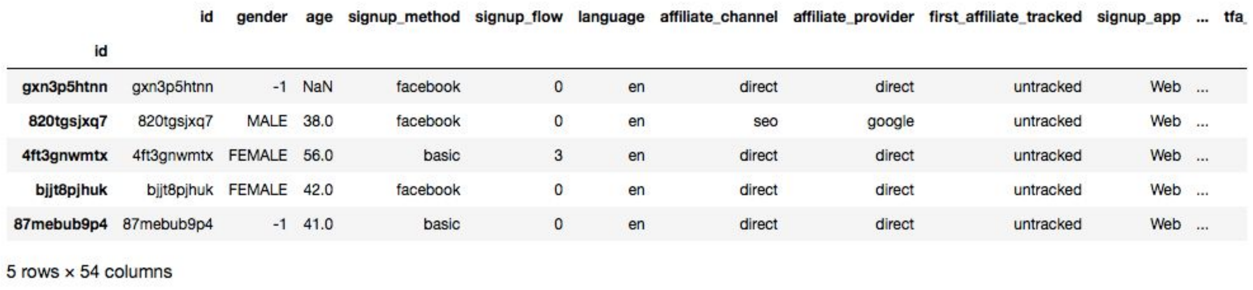
- timestamp_first_active: Similar treatment was given to the Timestamp First Active with new columns added as tfa_y, tfa_m, tfa_d, tfa_h (hour), tfa_wn (week number), tfa_w_{} (weekday, it was further split into each day).



	id	gender	age	signup_method	signup_flow	language	affiliate_channel	affiliate_provider	first_affiliate_tracked	signup_app	...	tfa_
id												
gxn3p5htnn	gxn3p5htnn	-1	NaN	facebook	0	en	direct	direct	untracked	Web	...	[illegible]
820tgsjxq7	820tgsjxq7	MALE	38.0	facebook	0	en	seo	google	untracked	Web	...	[illegible]
4ft3gnwmtx	4ft3gnwmtx	FEMALE	56.0	basic	3	en	direct	direct	untracked	Web	...	
bjjt8pjhuk	bjjt8pjhuk	FEMALE	42.0	facebook	0	en	direct	direct	untracked	Web	...	[illegible]
87mebub9p4	87mebub9p4	-1	41.0	basic	0	en	direct	direct	untracked	Web	...	

5 rows × 52 columns

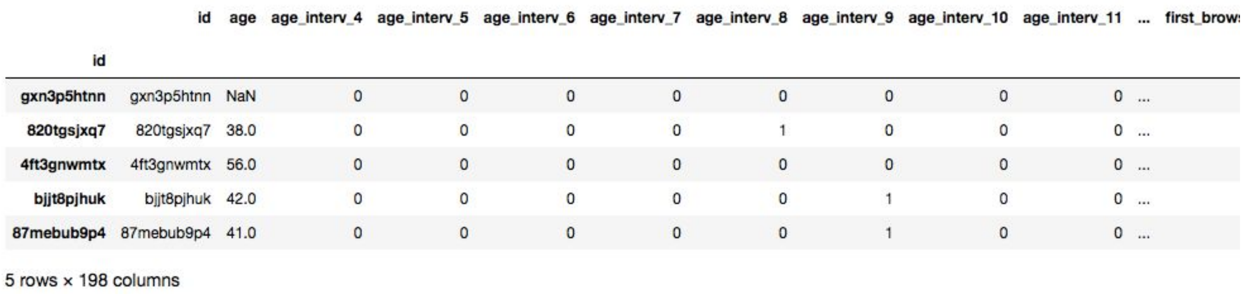
- Season (engineered feature): Using our studied domain knowledge, we know that season of booking can affect the destination choices. For example, people tend to visit cold places or beaches in summer, while the opposite is true in winter. We added two new features season_dac and season_tfa.



	id	gender	age	signup_method	signup_flow	language	affiliate_channel	affiliate_provider	first_affiliate_tracked	signup_app	...	tfa_
id												
gxn3p5htnn	gxn3p5htnn	-1	NaN	facebook	0	en	direct	direct	untracked	Web	...	
820tgsjxq7	820tgsjxq7	MALE	38.0	facebook	0	en	seo	google	untracked	Web	...	
4ft3gnwmtx	4ft3gnwmtx	FEMALE	56.0	basic	3	en	direct	direct	untracked	Web	...	
bjjt8pjhuk	bjjt8pjhuk	FEMALE	42.0	facebook	0	en	direct	direct	untracked	Web	...	
87mebub9p4	87mebub9p4	-1	41.0	basic	0	en	direct	direct	untracked	Web	...	

5 rows × 54 columns

- One-hot-encoding: Other categorical features had to be further one hot encoded. ['gender', 'signup_method', 'signup_flow', 'language', 'affiliate_channel', 'affiliate_provider', 'first_affiliate_tracked', 'signup_app', 'first_device_type', 'first_browser'] were encoded.



	id	age	age_interv_4	age_interv_5	age_interv_6	age_interv_7	age_interv_8	age_interv_9	age_interv_10	age_interv_11	...	first_brow
id												
gxn3p5htnn	gxn3p5htnn	NaN	0	0	0	0	0	0	0	0	...	
820tgsjxq7	820tgsjxq7	38.0	0	0	0	0	1	0	0	0	...	
4ft3gnwmtx	4ft3gnwmtx	56.0	0	0	0	0	0	0	0	0	...	
bjjt8pjhuk	bjjt8pjhuk	42.0	0	0	0	0	0	1	0	0	...	
87mebub9p4	87mebub9p4	41.0	0	0	0	0	0	1	0	0	...	

5 rows × 198 columns

Implementation

Evaluation Metrics: NDCG Scoring Function

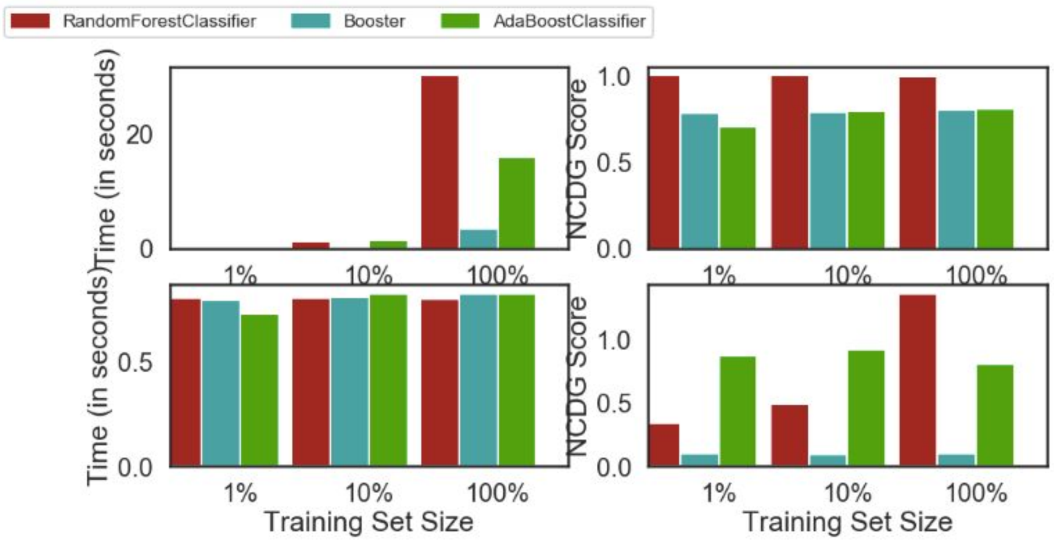
ref: https://www.kaggle.com/dietcoke/score-predictions-using-ndcg

Shuffle and Split Data

Now all *categorical variables* have been converted into numerical features, and all numerical features have been normalized. As always, we will now split the data (both features and their labels) into training and test sets. 80% of the data will be used for training and 20% for testing.

Random Forest, XGBoost and AdaBoost

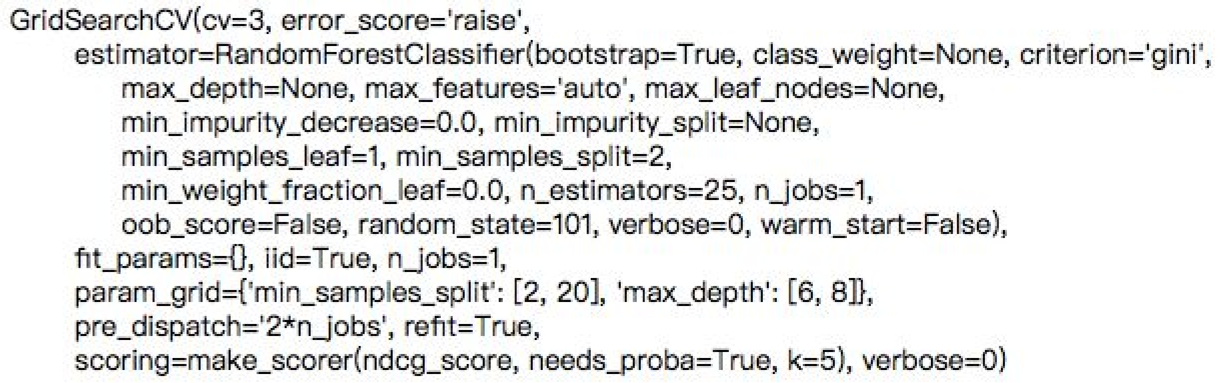
All models are trained with 1%, 10% and 100% of training data. The result:



IV. Results

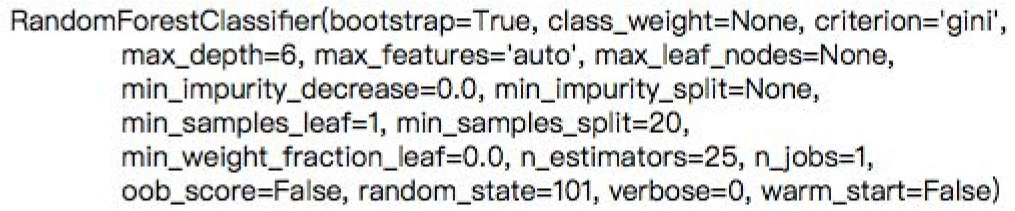
Based on the results, since random forest takes the longest time, but when it comes to NCDG score, it gets the best performance. So, I'll choose Random Forest as my final classifier. Besides, although XG booster gets decent score in the training phase, it gets a very low prediction outcome in testing phase; this indicated phenomena of overfitting. While for adaboost, it performances better than XGboost in both training and testing phases; however, NDGC scores in testing parts are slightly lower than those in training parts.

Using GridSearch for Random Forest:



```
GridSearchCV(cv=3, error_score='raise',
       estimator=RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=25, n_jobs=1,
            oob_score=False, random_state=101, verbose=0, warm_start=False),
       fit_params={}, iid=True, n_jobs=1,
       param_grid={'min_samples_split': [2, 20], 'max_depth': [6, 8]},
       pre_dispatch='2*n_jobs', refit=True,
       scoring=make_scorer(ndcg_score, needs_proba=True, k=5), verbose=0)
```

The best estimator: I used sklearns RandomForestClassifier along with Grid Search for cross validation. The parameters used for GridSearch were, min_samples_split over 2,20 and max_depth over 6,8. The best estimator over the training dataset was found to be having min_samples_split = 20 and max_depth was found to be 6. This model was used to predict and got scores: 0.81756



```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=6, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=20,
            min_weight_fraction_leaf=0.0, n_estimators=25, n_jobs=1,
            oob_score=False, random_state=101, verbose=0, warm_start=False)
```

Justification

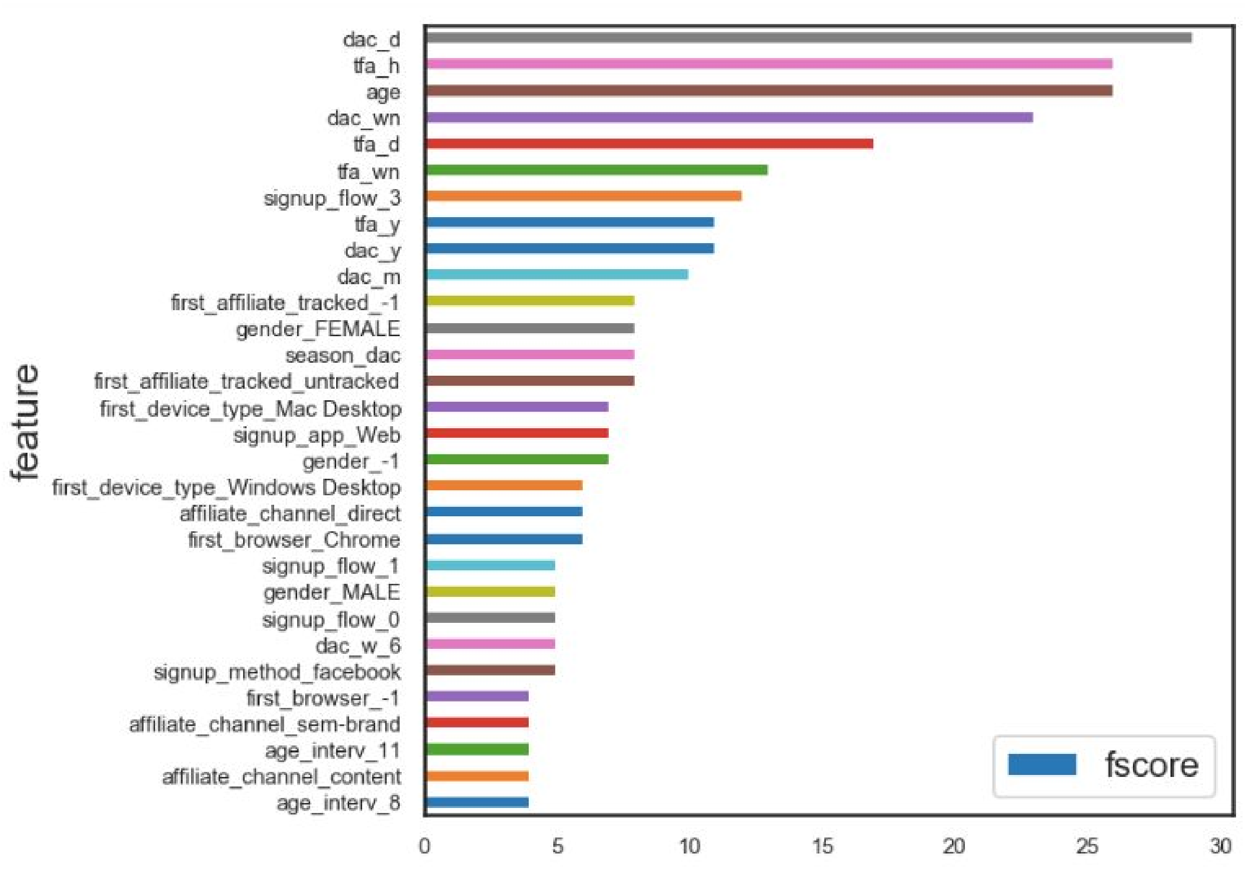
One of the important points to discuss about a trained machine learning algorithm is whether it will scale, will it be feasible in the scenario and performance should be consistent with varied data. We can then say whether the model is **robust** or not. The model, although training with a reduced feature space, gives a better result with the private subset of the training set as can be seen by the results on Kaggle. Also, the training and prediction times were very less due to the feature space reduction step and the NDCG score was also good enough. Depending on the scale of the deployment, the parameters and the number of features can be scaled as well. Hence, the model seems to be robust. Alongside this, the model seems to be trustable and aligns well with our expected solutions outcomes.

V. Conclusion

Free-Form Visualization

I tried to train XGBoost over the entire feature set of 198 columns but due to limited memory of 8 GB and low computational resources, training had to be stopped on my machine as it took more than an hour. Training was stopped around the 60 minute mark and it was decided to use the feature importances given by the model trained on a subset of the dataset.

The data was then trained over [10,20,30,40] top features and the maximum validation score was achieved for top 30 features. They were:



Reflection

what was the most interesting aspect of this work?

This is the freestyle project that I can choose any subject that I'm interested in. Besides, since this dataset is the real data from Airbnb, and I'm a person of traveler, so this topic tickles my fancy to find out what will affect people's choice when they make the decision of where to go.

What was the biggest challenge, and how did you overcome it?

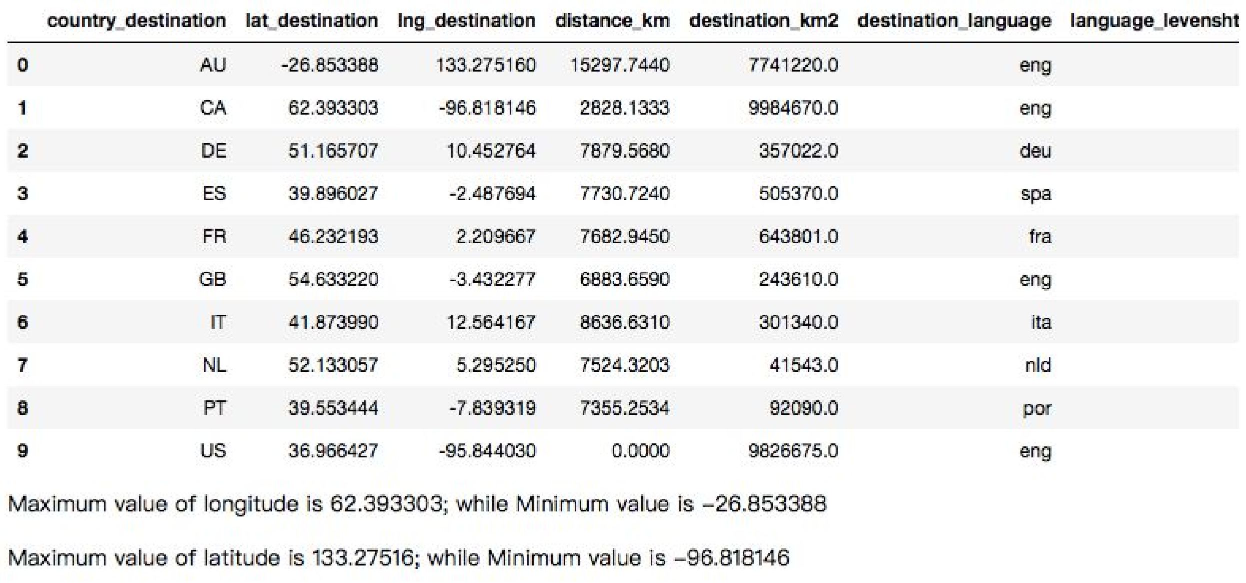
One of the biggest challenges is to define evaluation matrics. In the before, we preferred to use accuracy, precision, recall, and f-score to evaluate if the model is good enough. Because this dataset is from Kaggle, and the requirement is to use NDCG for evaluation. It took me plenty of time to understand and modified these matrics to see if this will improve or not. Thanks for the website(http://dalelane.co.uk/blog/?p=3403), this really helps me to understand those equations and finish the experiments on modified evaluation matrics (even though the outcome is not satisfied, but is still very useful).

What did you learn?

By doing this project, I realize that in the real world, not all data are well-prepared. The true value of a data scientist is to dig out mess of data and find the value, relationship of them.

Improvement for NDCG Scoring Function

Raw data from countries.csv

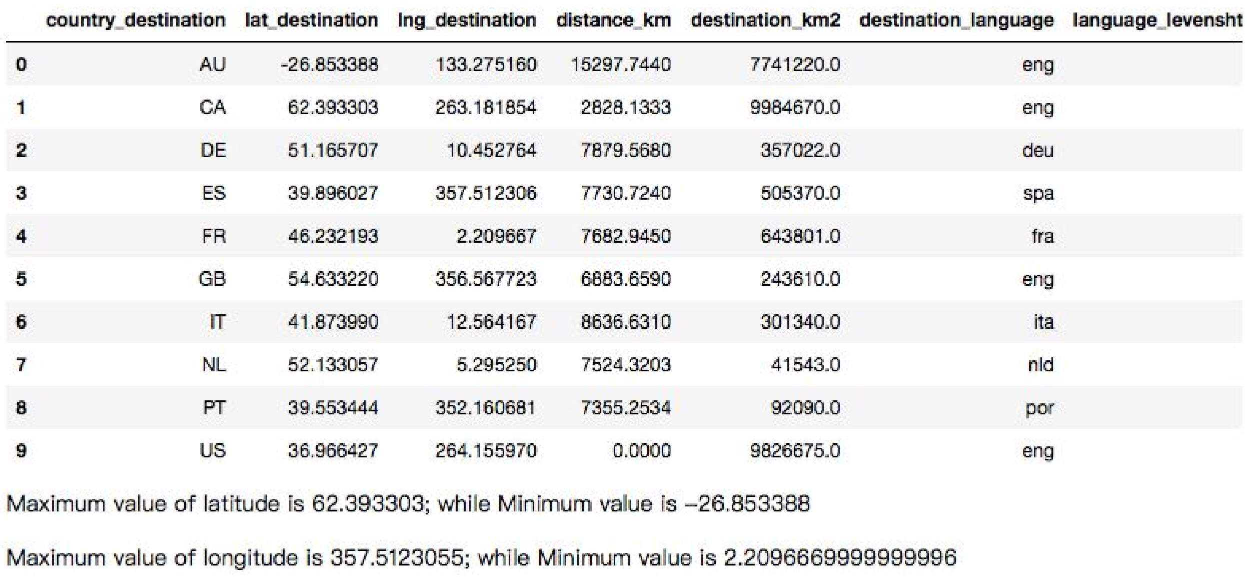


	country_destination	lat_destination	lng_destination	distance_km	destination_km2	destination_language	language_levensht
0	AU	-26.853388	133.275160	15297.7440	7741220.0	eng	
1	CA	62.393303	-96.818146	2828.1333	9984670.0	eng	
2	DE	51.165707	10.452764	7879.5680	357022.0	deu	
3	ES	39.896027	-2.487694	7730.7240	505370.0	spa	
4	FR	46.232193	2.209667	7682.9450	643801.0	fra	
5	GB	54.633220	-3.432277	6883.6590	243610.0	eng	
6	IT	41.873990	12.564167	8636.6310	301340.0	ita	
7	NL	52.133057	5.295250	7524.3203	41543.0	nld	
8	PT	39.553444	-7.839319	7355.2534	92090.0	por	
9	US	36.966427	-95.844030	0.0000	9826675.0	eng	

```
Maximum value of longitude is 62.393303; while Minimum value is -26.853388

Maximum value of latitude is 133.27516; while Minimum value is -96.818146
```

Since Latitude has the range from +90°N to -90°S, 180° totally; while Longitude has the wilder range of -180°W to 180°E, 360° totally. Here I re-define Longitude to be the value within 0° to 180° (Since -180° locates at the same point with 180°), so I replace negative value of longitude, x, to be (180+x):



	country_destination	lat_destination	lng_destination	distance_km	destination_km2	destination_language	language_levensht
0	AU	-26.853388	133.275160	15297.7440	7741220.0	eng	
1	CA	62.393303	263.181854	2828.1333	9984670.0	eng	
2	DE	51.165707	10.452764	7879.5680	357022.0	deu	
3	ES	39.896027	357.512306	7730.7240	505370.0	spa	
4	FR	46.232193	2.209667	7682.9450	643801.0	fra	
5	GB	54.633220	356.567723	6883.6590	243610.0	eng	
6	IT	41.873990	12.564167	8636.6310	301340.0	ita	
7	NL	52.133057	5.295250	7524.3203	41543.0	nld	
8	PT	39.553444	352.160681	7355.2534	92090.0	por	
9	US	36.966427	264.155970	0.0000	9826675.0	eng	

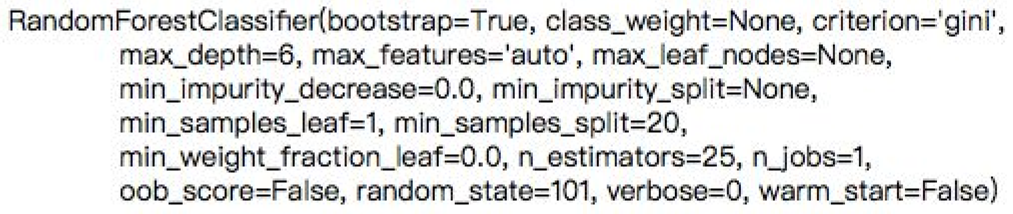
```
Maximum value of latitude is 62.393303; while Minimum value is -26.853388

Maximum value of longitude is 357.5123055; while Minimum value is 2.2096669999999996
```

Actual and Predict destination has Range for longitude between [0, 89.246691] latitude between [0, 175.3026385]

1. A perfect answer (gain = 3) difference in longitude < 20 and difference in latitude < 40 and
2. A good answer (gain = 2) difference in longitude < 40 and difference in latitude < 80 and
3. A relevant answer (gain = 1) difference in longitude < 60 and difference in latitude < 120 and
4. An incorrect answer (gain = 0) Others

Based on the results, use Random Forest to test this evaluation matrics with grid search.
The best estimator:



```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=6, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=20,
            min_weight_fraction_leaf=0.0, n_estimators=25, n_jobs=1,
            oob_score=False, random_state=101, verbose=0, warm_start=False)
```

As test result, after put distance information into NDCG evaluation model, the final scores dramatically dropping down, from 0.8 to 0.2. Because we evaluate 5 possible destinations and 5 of them will get their gains based on the different between Latitude and Longitude. The reason why this evaluation model is not useful here is that in this case, we only put 10 target counties as our prediction results. Some countries locate very far away from others, so when predicting, the other possible candidates will force the final score to be lowered since no matter what they predict, the predictions are very far from the target.

If we can access the dataset with much more destination counties to handle, at the same time, if I have GPU environment to process, then I believe this modified NDCG with our classifiers will have a better results.