

**ISSS610 Applied Machine Learning**

**Airbnb - Predicting User First Travel Destination**

***Group 9***

Alakananda Giridhar

Himani Mukka

Manikantha Devasish Patnana

Sai Aswin Jami

***08 December 2019***

Contents

[1 Introduction – Problem Statement 2](#_Toc26891224)

[2 Visual Representation of Modelling Methodology 2](#_Toc26891225)

[3 Data description, EDA and Data Cleaning 3](#_Toc26891226)

[4 Feature Engineering 3](#_Toc26891227)

[5 Data preparation for modelling 4](#_Toc26891228)

[6 Modelling and Evaluation 5](#_Toc26891229)

[7 Challenges 9](#_Toc26891230)

[8 Future Work – Measures for improvement 9](#_Toc26891231)

[9 References 10](#_Toc26891232)

[10 Appendix 11](#_Toc26891233)

# Introduction – Problem Statement

The dataset chosen for this project was the “Airbnb new user bookings” challenge on Kaggle, which requires the analyst to predict the first travel destination booking of the users, who are based in US. New users on Airbnb have the option to book a place to stay in 34,000+ cities across 190+ countries. The datasets provided include basic user details, sessions data and destination country-wise summary statistics. Using these details, we were required to predict the travel destination of each user from amongst 12 options - 'US', 'FR', 'CA', 'GB', 'ES', 'IT', 'PT', 'NL','DE', 'AU', 'NDF' (no destination found), and 'other'. Airbnb has used the NDCG method of scoring, with the requirement to predict top 5 destinations per user, such that Airbnb can share personalized content based on the top 5 predicted countries. By doing so, the average time to first booking will be decreased, demand can be forecasted better and the satisfaction and conversion rate of customers will improve. (Airbnb, 2015)

# Visual Representation of Modelling Methodology

Evaluation using NDCG

NDCG is a scoring method that takes top k predictions for each user and gives a higher score if the relative position of the accurate prediction (ground truth label) is higher up in the list of top k predictions.

Some of the reasons for us using NDCG as the evaluation metric are as follows:

* To have a common ground of comparison with other kagglers in the competition.
* The sessions data houses details of time elapsed, device used etc for a generic user action (like ‘view’ or ‘search’) and doesn’t contain any information about which country the user clicked on or searched for, thereby making it immensely hard to make an accurate prediction of destination. Hence, for the purpose of displaying highly suitable recommendations based on top 5 predictions, NDCG was used, as highly relevant destinations are more useful when they appear earlier in a search engine result list (have higher ranks) and highly relevant destinations are more useful than marginally relevant destinations, which are in turn more useful than non-relevant destinations. (Zou, 2017)

# Data description, EDA and Data Cleaning

The data provided included sessions data (revolving around generic actions, as mentioned in section 2), basic user profile and summary statistics (about population, distance from USA etc) of the destination countries.

In addition to that, we also procured external data (language\_data.xlsx (Compare Languages)), in order to explore the hypothesis regarding similarity of destination language to the user’s language.

Subsequently, data exploration was done to understand the data and its issues to decide on an appropriate course of action for cleaning or dropping the features, the details of which are included in the appendix.

# Feature Engineering

Upon exploring the data, which was quite limited in the information contained, we decided that our main focus was going to be on deriving the best features possible such that the algorithms would give meaningful and best possible results. For this purpose, some hypotheses were arrived at, based on which features were created:

* Age might be a major predictor because
  + Older people have more savings and time, meaning they can afford to travel to expensive places
  + Younger people are more adventurous. So, they might be open to trying offbeat countries, but can probably only afford cheaper ones
  + The more the proportion of people of their own age in the destination country, more the suitability of the activities/tourist attractions to the user
  + Older people might prefer going to countries where it is easy to get by, maybe due to similarity of languages. Younger ones might be more open to countries with alien languages
* Financial affluence of the person
  + No direct indicator present in the data. Therefore, it wasn’t possible to do any form of user-level feature cross using user’s financial status and the cost of living of the destination country.
  + Instead, we tried to derive implicit features, like number of unique devices the person owns, the number of apple products and number of tablets (the more they own, the higher the probability of them being affluent)
* Seasonality
* Latent information contained in browsing patterns
  + There was no information in sessions data regarding which country the user browsed for. So, action-wise time elapsed and frequencies were used to determine if they implicitly captured any hidden pattern connected to the first travel destination.

Based on the above hypothesis, many features were derived from each dataset, the details of which are given in Appendix.

# Data preparation for modelling

Once all the features were in place, the transformed datasets were combined as follows to get the final dataset:

Transformed train\_users.csv

Transformed countries.csv

Left Join

Transformed Languages.xlsx

Inner join

Transformed Sessions.csv

Inner Join

After obtaining the merged dataset, correlation analysis was done to remove features which were highly correlated (>0.8). The variables which were excluded that way are listed in the Appendix.

Subsequently, keeping in mind that different algorithms have different data requirements, 3 different datasets were created as follows:

* Dataset 1: The dataset obtained from the above steps is retained as is
* Dataset 2: The categorical variables in the dataset obtained above is one-hot encoded to create dummy variables
* Dataset 3: The ‘Dataset 2‘ is further processed to replace null/nan vaues with extreme values (-99)

# Modelling and Evaluation

1. First and foremost, a **baseline** score of NDCG was obtained by assigning the top 5 countries by proportion in the dataset (NDF, US, Others, FR, IT) to each user. The NDCG of this baseline model was 85.669%, while the winning model on Kaggle had a score of 88.697%. With this, we came to an understanding of how difficult it would be to raise our model performance by even 3% from the baseline.

A screenshot of a cell phone

Description automatically generated

Then, the following models were built, all of which are tree-based algorithms. This was done intentionally, as this dataset had a lot of missing values in key features, and tree models handle them well by treating them as a separate bin.

1. **Decision Trees & Random Forest**  
     
   Decision tree model was the first to be implemented, as it is the simplest of all and works well with categorical target variable, while simultaneously being easy to interpret and quick to train. This was then extended and built into Random forest model, which uses a bagging ensemble approach with majority voting, comprising many individual decision trees.

Parameter tuning was also done on the same to identify the best set of parameters that reduce overfitting and improve model performance

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Parameters** | **Accuracy on train data** | **Accuracy on test data** | **NDCG Score on Unseen data** |
| Decision tree | criterion='gini', max\_depth=7, | 64.10% | 63.80% | 87.314% |
| Random Forest : GridSearchCV | criterion: 'gini', max\_depth: 11, n\_estimators: 17 | 64.10% | 63.80% | 87.733% |
| Random Forest : RandomisedSearchCV | n\_estimators: 50, max\_depth: 20, bootstrap: False | 76.10% | 64.60% | 87.962% |
| Random Forest : RandomisedSearchCV | n\_estimators: 150, max\_depth: 19, bootstrap: True | 73.30% | 64.80% | 88.058% |
| Random Forest : RandomisedSearchCV | criterion='gini',  max\_depth = 20,n\_estimators = 140, min\_samples\_split = 5,min\_samples\_leaf = 4, bootstrap = False | 70.80% | 64.80% | 88.150% |

During the modelling process, it was observed that the model performance increased with change in parameters. However, the accuracy on the train and test split of the labelled dataset indicated overfitting, while the kaggle NDCG score displayed negligible improvement.

The final set of parameters obtained gave the best Random Forest model with an NDCG score of 88.15%. The fact that such a simple model had performed so well was indicative of the quality of the features that we had derived.

1. **AdaBoost**

Following the bagging ensemble (Random Forest), the first boosting ensemble algorithm that we tried was AdaBoost classifier, which was implemented on variedly sampled train data to assess the performance. This was because, AdaBoost is known to do a better job than bagging algorithms in cascading the classifiers, by giving higher weightage to weak learners with a higher predictive accuracy, in an iterative fashion. Also, AdaBoost supposedly has a high degree of precision and is less susceptible to overfitting.

While it was expected that AdaBoost model could not compete with the performance of other more efficient algorithms like CatBoost, it surprisingly couldn’t outperform even Random Forest, owing to the limitations of the algorithm. This could have been because of the severe data imbalance and Adaboost’s limitation when it comes to adjusting to significant outliers. (Qiang, 2019)

Sampling techniques were used to combat the same. With undersampling, training data was massively reduced, as under-sampling tries to match all the records with the least frequented data in the dataset, which in our case was ‘PT’ with 217 records. The same applied to other sampling techniques such as cluster centroids and AdaSYN. Smote/ Over sampling on the other hand sampled the data taking the majority class as reference, leaving us with the sample size of 96705 for ‘PT’ class as well. Balance Cascade, however, is bound to give reasonable sampling with the data. (G. Lemaitre, 2016-2017)

We then attributed the persistent poor performance of Adaboost to the way the algorithm is designed. Adaboost seems to have been specifically designed for balanced binary classification, wherein the probability of the weak classifier will have to be greater than 50%, or better than random guessing, to be considered. But, with our dataset having 12 target classes which are imbalanced, we do not achieve the cutoff threshold for weak learners, and in the end AdaBoost does not change the end result of the classification. (SauceCat, 2017)

1. **Superior Models - CatBoost, XGBoost & LightGBM**

CatBoost, XGBoost & LightGBM are gradient boosting decision tree (GBDT) based state-of-the-art algorithms for structured data. Since GBDTs train a basic model on the data and then use the first model’s error as a feature to build successive models, they reduce the model’s error because each successive model improves against the previous model’s weaknesses. Additionally, they can train on large datasets very quickly by parallelising over many CPUs distributed amongst a cluster of computers and can extract most of the information from a dataset in under a minute.

Each of these three algorithms has its own unique yet efficient ways of giving high performance, as follows:

* **Catboost** handles categorical features innately by generating random permutations of the dataset and for each sample, computing the average label value for the sample with the same category value placed before the given one in the permutation. It also processes the data using GPU acceleration, and does feature discretization into a fixed number of bins (128 and 32). This way, it overcomes the curse of dimensionality associated with one-hot encoding of categorical features that have many distinct values.
* **XGBoost:** Using Sparsity-aware Split finding, it learns a default direction for the data that allows the algorithm to skip samples that are missing an entry for the feature. This technique takes advantage of the sparsity in the dataset, reducing the computation to a linear search on only the non-missing entries.
* **LightGBM** uses Gradient-based One-Side Sampling, which inspects the most informative samples while skipping the less informative samples, thereby extracting the most information from the dataset as fast as possible. It also uses Exclusive Feature Bundling which takes advantage of sparse datasets by grouping similar features in a near lossless way. Also, it doesn't do exact searches for optimal splits like XGBoost does in its default setting, but rather does splits through histogram approximations, which result in a slight decrease in predictive performance but provides a much larger speed increase in training. It can also handle large datasets much faster, unlike XGBoost. (Moreno, 2018)

Of all the aforementioned algorithms, the LightGBM Model performed the best. This could be because it produces much more complex trees by following leaf wise split approach rather than a level-wise approach, which is the main factor in achieving higher accuracy. However, it can sometimes lead to overfitting, which was combatted by setting a max\_depth parameter. (Aranglol, 2019)

|  |  |  |
| --- | --- | --- |
| **Model** | **Parameters** | **NDCG Score on Unseen Data** |
| CatBoost | {iterations=10000, learning\_rate=0.05, depth=7, l2\_leaf\_reg=None, model\_size\_reg=None, loss\_function='MultiClass', random\_seed=1234, use\_best\_model=True, task\_type='GPU', early\_stopping\_rounds=100} | 88.200% |
| XGBoost | {"eta": 0.05, "max\_depth": 7, "objective": 'multi:softprob', "num\_class": 12, "subsample": 0.5, "colsample\_bytree":0.8, "colsample\_bylevel":0.8, "colsample\_bynode":0.8,"lambda":1, "alpha":1} | 88.187% |
| LightGBM | {'boosting':'gbdt','num\_leaves':150, 'objective':'multiclass','num\_class':12,'max\_depth':7,'learning\_rate':0.05,'max\_bin':200,'feature\_fraction':0.8,'feature\_fraction\_bynode':0.8,'metric':'multi\_logloss'} | 88.293% |
| LightGBM with dart | {'boosting':'dart', 'num\_leaves':150, 'objective':'multiclass','num\_class':12,'max\_depth':7,'learning\_rate':0.05,'max\_bin':200,'feature\_fraction':0.8,'feature\_fraction\_bynode':0.8,'metric':'multi\_logloss'} | 88.264% |
| LightGBM tuned model with dart | { 'boosting':'dart', 'objective':'multiclass','num\_class':12,'max\_depth':10,'learning\_rate':0.1,'max\_bin':200,'feature\_fraction':0.6,'feature\_fraction\_bynode':0.6,'metric':'multi\_logloss','min\_data\_in\_leaf': 500, 'num\_iterations': 200, 'num\_leaves':300} | 88.242% |
| LightGBM tuned model with goss | {'boosting':'goss', 'objective':'multiclass','num\_class':12,'max\_depth':10,'learning\_rate':0.1,'max\_bin':200,'feature\_fraction':0.6,'feature\_fraction\_bynode':0.6,'metric':'multi\_logloss', 'min\_data\_in\_leaf': 500, 'num\_iterations': 200, 'num\_leaves':300} | 87.949% |
| LightGBM tuned model with gbdt | {'objective':'multiclass','num\_class':12,'max\_depth':10,'learning\_rate':0.1,'max\_bin':200,'feature\_fraction':0.6,'feature\_fraction\_bynode':0.6,'metric':'multi\_logloss', 'min\_data\_in\_leaf': 100, 'num\_iterations': 200, 'num\_leaves':300} | 88.247% |

Following this, parameter tuning with the lightGBM model was attempted, but no parameter combination surpassed the original LightGBM model performance. The feature importance of the model too was in alignment with the hypotheses originally defined. (Feature importance chart in Appendix)

# Challenges

The following are some of the challenges that arose during the project:

* Unacceptable amount of Missing data in few prominent features
* Heavily imbalanced dataset
* Leveraging the data collectively from all datasets to create a final table of meaningful features was a herculean task
* As mentioned in Section 4 (Feature Engineering), there was no information regarding the content the user browsed for
* Additional features like cost of living of destination country couldn’t be combined with this data since user-level financial information or other pertinent details were not provided, thereby preventing meaningful feature crosses
* Custom generation of NDCG scorer and memory handling was a challenge too

# Future Work – Measures for improvement



While the work done so far put us on par with other Kagglers with a rank of 247 out of 1462 teams, the following could be done to improve the model performance a bit more:

* An Ensemble of the top models (CatBoost, XGBoost & LightGBM) or using Neural Networks or stacking
* Exploring Adaboost.M1 or Adaboost.M2
* Exploring fastAI API
* Building 12 one Vs All AdaBoost classifiers and cascading them

# References

Airbnb. (2015). *Airbnb New User Bookings*. Retrieved from Kaggle: https://www.kaggle.com/c/airbnb-recruiting-new-user-bookings

Aranglol. (2019, April 19). *Lightgbm vs xgboost vs catboost*. Retrieved from datascience.stackexchange: https://datascience.stackexchange.com/questions/49567/lightgbm-vs-xgboost-vs-catboost

*Compare Languages*. (n.d.). Retrieved from elinguistics: http://www.elinguistics.net/Compare\_Languages.aspx

G. Lemaitre, F. N. (2016-2017). *imbalanced-learn API*. Retrieved from imbalanced-learn: https://imbalanced-learn.readthedocs.io/en/stable/api.html

Gasquez, D. (2015). *NDCG Scorer*. Retrieved from Kaggle: https://www.kaggle.com/davidgasquez/ndcg-scorer

Moreno, H. (2018, October 15). *Gradient Boosting Decision trees: XGBoost vs LightGBM (and catboost)*. Retrieved from Medium: https://medium.com/kaggle-nyc/gradient-boosting-decision-trees-xgboost-vs-lightgbm-and-catboost-72df6979e0bb

Qiang, X. (2019, July 18). *Adaboost*. Retrieved from EasyAI: https://easyai.tech/en/ai-definition/adaboost/

SauceCat. (2017, April 30). *Boosting algorithm: AdaBoost*. Retrieved from TowardsDataScience: https://towardsdatascience.com/boosting-algorithm-adaboost-b6737a9ee60c

Zou, J. (2017, October 23). *Information Retrieval notes*. Retrieved from zhuanlan.zhihu: https://zhuanlan.zhihu.com/p/30180260

# Appendix

1. **Dataset description, Exploration and Actions taken**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sessions.csv:** |  |  |  |
| **Column Name** | **Description** | **Issues found** | **Actions taken for analysis** |
| User ID | User ID of the user | 0.32 % missing values | Dropped the records |
| Action | Action made on the web application | 0.75 % missing values | Concatenated action, action\_type and action\_detail. Imputed missing values with None |
| Action\_type | Action performed by the user to access the pages of the application | 20 % missing values |
| Action\_detail | Web elements on which the actions are performed | 20 % missing values |
| Device type | Type of the device that is used in browsing the listings | None | None |
| Secs\_Elapsed | Time spent on the each web element | 1.28 % missing data | Aggregated at user level as avg time spent per action type (one column per action), imputed the null values with 0. |

|  |  |  |  |
| --- | --- | --- | --- |
| **train\_users.csv:** |  |  |  |
| **Column Name** | **Description** | **Issues found** | **Actions taken for analysis** |
| ID | User ID of the user | None | None |
| Date\_account\_created | Date of account creation | None | None |
| timestamp\_first\_active | Timestamp of the first activity | None | None |
| date\_first\_booking | Date of the Booking | Variable from the future | Variable is excluded |
| Gender | Gender of the User | 44 % missing data | Considered as a separate category ‘Unknown’ |
| Age | Age of the User | 42 % missing data. Few inconsistencies (age >1000 etc). | Replaced missing values with -99. Inconsistencies were cleaned. |
| signup\_method | Sign up Channel | None | None |
| signup\_flow | The page from which a user came to signup | None | None |
| Language | Language of the User | None | None |
| affiliate\_channel | Type of paid marketing | None | None |
| affiliate\_provider | Marketing channel that site was listed on | None | None |
| first\_affiliate\_tracked | First marketing the user interacted with before signing up | 2 % missing data and 51 % ‘untracked’ | Dropped. Assumiing that the columns ‘affiliate channel’ and ‘affiliate provider’ implicitly capture this data. |
| signup\_app | OS Platform | None | None |
| first\_device\_type | First Device used | None | None |
| first\_browser | First Browser used | None | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Countries.csv:** |  |  |  |
| **Column Name** | **Description** | **Issues found** | **Actions taken** |
| Country\_Destination | Country name | None | None |
| Lat\_destination | Latitude | None | None |
| Long\_destination | Longitude | None | None |
| distance\_km | Distance from USA | None | None |
| destination\_km2 | Destination area | None | None |
| destination\_language | Predominant language spoken at the region | None | None |
| language\_levenshtein\_distance | Levenshtein distance from English | None | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Age\_gender\_bkts.csv** |  |  |  |
| **Column Name** | **Description** | **Issues found** | **Actions taken** |
| Age\_bucket | Age Buckets | None | None |
| Country\_destination | Destination country | None | None |
| Gender | Gender | None | None |
| Population | Number of people in that gender and age bucket in that country | None | None |
| year | Year | None | None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Language\_Data.xlsx:** |  |  |  |
| **Column Name** | **Description** | **Issues found** | **Actions taken** |
| Destination\_Language | Languages of the destination country | None | None |
| Languages | List of languages of the users | None | None |
| Language\_levenshtein\_distance | Similarity score of user language and each destination language | None | None |

1. **Feature Engineering**
2. train\_users.csv :

* DaysToFirstActivity : Time difference between date of account creation and date of first usage of Airbnb
* day\_account\_created : Day (1 to 31) of account creation
* weekday\_account\_created : Weekday (0 to 6) of account creation
* week\_account\_created : Week (1 to 52) of account creation
* month\_account\_created : Month (1 to 12) of account creation
* year\_account\_created : Year the account was created
* day\_first\_active : Day (1 to 31) of first activity
* weekday\_first\_active : Weekday (0 to 6) of first activity
* week\_first\_active : Week (1 to 52) of first activity
* month\_first\_active : Month (1 to 12) of first activity
* year\_first\_active : Year of first activity user
* Age : Age >1900 and < 2000 were assumed to be year of birth and were subtracted from 2015 to get age. Age < 15 and above 100 were replaced with nan.

1. Sessions.csv:

* Sequence : Variables named Action, Action Type and Action Detail were concatenated to form an action sequence.
* Groupby operation was done on sessions data using user\_id and action sequence, which then was pivoted to get aggregate values for average time spent and total frequency for each user action.
* Similarly, groupby operation was performed on sessions data using user id and device type to get the following features:
  + Number\_of\_UniqueDevices : Number of unique devices used by the users
  + NumberofAppleProducts : Number of apple products used by the user
  + NumberofTablets : Number of Tablets used by the user

1. Countries.csv:

Group the countries dataset based on country-age bucket and country- gender to get aggregated population in thousands. Then, it is pivoted to transform the table from long format to wide format, and then the percentages of population with respect to age/gender and country are calculated.

1. Languages.xlsx:

Language dataset is pivoted from long format to wide format, to get the similarity score of each of the destination countries’ languages and the user’s native language.

1. **Variables excluded due to very high correlation**

The below list of variables are excluded from the analysis as they are highly correlated.

AU\_gender, CA\_gender, DE\_gender, ES\_gender, FR\_gender, GB\_gender, IT\_gender, NL\_gender,PT\_gender,US\_gender,sequence\_recode\_freq\_similar\_listings,avg\_secs\_elapsed\_Booking\_confirmation,avg\_secs\_elapsed\_view\_locations\_Europe,PT\_age,CA\_age,GB\_age,AU\_age,US\_age,DE\_age,FR\_age,IT\_age,day\_account\_created,weekday\_account\_created,week\_account\_created,month\_account\_created,week\_first\_active,year\_account\_created,CA\_eng,GB\_eng,AU\_eng,US\_eng,PT\_por,IT\_ita.

1. **Feature Importance chart of LightGBM**

A screenshot of a social media post

Description automatically generated

1. **NDCG Score**

