MEF UNIVERSITY

Hotel Recommendation for Online Travel Agencies

Capstone Project

Cem Kılıçlı

İSTANBUL, 2017

MEF UNIVERSITY

Hotel Recommendation for Online Travel Agencies

Capstone Project

Cem Kılıçlı

Advisor: Asst. Prof. Dr. Hande Küçükaydın

İSTANBUL, 2017

EXECUTIVE SUMMARY

Hotel Recommendation for Online Travel Agencies

Cem Kılıçlı

Advisor: Asst. Prof. Dr. Hande Küçükaydın

AUGUST, 2017, 28 pages

Since the early 2000s, online travel agencies (OTAs) have become a central online market source, used by millions of users in all over the world. Recommendation systems became one of the essential tools for them to increase their profit. In this manner, Expedia.com decided to create a challenge in an online data science community to create a basis for a recommendation system by a collaborative effort.

The challenge is to create a recommendation to their user by employing the data they have in hand. Because of the limited data that is available on user's personal choices, this is a complex problem to solve. The data set is provided by Expedia and competitors are asked to overcome this complex problem.

We approach this problem by analyzing and visualizing the data that is provided. In this phase, we understand that the distribution of data is highly unbalanced and have a lot of missing points that one can expect from a real-life problem.

By employing the knowledge that we gained in explanatory data analysis we have employed several different algorithms to solve the problem. Because of the unbalanced nature of the data set we have selected the algorithms that can handle this kind of situation.

Random forest and decision tree classifier algorithms perform well enough to carry out to tuning phase. We have tuned these algorithms by several different parameters that we think can improve the outcome.

Finally, we concluded on employing random forest algorithm by the model we build have a significant performance over the other algorithms. In a real-life problem, as such, we prove our model will perform and deliver reliable outcomes.

Key Words: Recommendation systems, collaborative filtering.

ÖZET

Online Seyahat Acentaları İçin Öneri Sistemleri

Cem Kılıçlı

Tez Danışmanı: Asst. Prof. Dr. Hande Küçükaydın

AĞUSTOS, 2017, 28 sayfa

2000'li yılların başından itibaren internette yaşanan gelişmeler, seyahat sektöründe online seyahat acentalarının yükselişini sağlamıştır. Öneri sistemleri karlılığın ve satışın artırımda önemli bir ekipman olarak karşımıza çıkmış ve birçok e-ticaret sitesi tarafından kullanılmaya başlanmıştır. Kişiselleştirilmiş önerilerin gücü Expedia gibi online seyahat acentaların ilgisini çekmiş ve ellerindeki limitli veriyi kullanarak öneri sistemleri geliştirmeye başlamıştır.

Online seyahat acentalerının önemli sorunlarından bir bu sistemleri efektif bir şekilde kullanmak için ellerinde yeterli kullanıcı tercihi verisi olmaması. Bu nedenle Expedia.com, bir online veri bilimi topluluğunda kendi modelini daha iyi bir noktaya getirebilmek için bir yarışma düzenlemiş ve şirketinde biriktirdiği bir kısım veriyi yarışmacın katılımcılarına açmıştır.

Bu yarışmada kişisel tercihler hakkında limitli veri bulunan bir veri kümesi paylaşılmış ve yarışmacıların veri kümesini kullanarak kullanıcılara en iyi öneriyi yapacak sistemi geliştirmesi istenmiştir.

İlk olarak veri kümesi incelenmiş ve analiz edilmiştir. Analiz sonucunda paylaşılan verinin oldukça dengesiz bir dağlıma sahip olduğu görülmüştür. Ayrıca veri setinde beklendiği üzere eksik veriler olduğu tespit edilmiştir.

Analiz aşamasında edinilen bilgi kullanılarak dengesiz veri dağılımında da efektif bir şekilde çalışabileceği kanıtlanmış olan rasgele orman ve kara ağacı algoritmaları seçilerek hedeflenen çıktılar elde edilmeye çalışılmıştır. Seçilen algoritmalar optimize edilmek için değişik parametrelerle denenmiştir.

Sonuç olarak rastgele orman algoritması ile kurulmuş olan modelin var olan bu kompleks problemi iyi bir performans ile çözebileceği bulunmuştur.

Anahtar Kelimeler: Öneri sistemleri, işbirliğine dayalı filtreleme.

TABLE OF CONTENTS

Academic Honesty Pledge vi

EXECUTIVE SUMMARY vii

ÖZET viii

TABLE OF CONTENTS ix

1. INTRODUCTION 1

    1.1 Recommendation systems 1

2. Project Definition 4

    2.1 Problem Statement 4

    2.2 Project Objectives 4

        2.2.1 Understanding the data 4

        2.2.2 Data Cleaning and Enrichment 4

        2.2.3 Model Selection 4

        2.2.4 Model Tuning 4

        2.2.5 Results and evaluation 5

    2.3 Project Scope 5

3. About The Data 6

    3.1 The Train Data 7

    3.2 The Test Data 8

    3.3 The Destinations Data 8

4. Methodology 9

    4.1 Sampling 9

    4.2 Data Explorations 9

        4.2.1 The Train Data 9

    4.3 Data Cleaning and Enrichment 12

    4.4 Model Selection 14

        4.4.1 Selected Algorithms 14

        4.4.2 Discarded Algorithms 15

    4.5 Model Tuning 16

5. Results 17

    5.1 Conclusion and Next Steps 19

6. Social and Ethical Aspects 20

7. Value Delivered (Contribution) 21

APPENDIX A 23

APPENDIX B 25

Project hub in Github - https://github.com/cemkilicli/bda_capstone_project 25

REFERENCES 26

Glossary 29

I. List of Tables 29
II. List of Figures 29

1. INTRODUCTION

Since the early 2000s, online travel agencies (OTAs) have become a central online market source, used by millions of users in all over the world. These agencies do not have any physical shops or stores. They only operate online, try to build a bridge between online users and property owners. OTAs are simply an online marketplace wherein property owners (such as hotels) list their available rooms and services, and users that are looking for accommodations can easily manage reservations.

Latest developments in machine learning and cloud computing have provided a useful tool called "recommendation systems" that is applicable to all online e-commerce and media sites. Making recommendations to the users became one of the most common practice in ecommerce to maximize the revenue. Online travel agencies are the market actors that also have been benefiting from this tool. Most of these agencies set their strategy to recommend best matching hotels to users according to preferences and needs.

The fact is that OTAs are not a place for a user to build an online profile and specify her/his preferences by means of likes or interests. These sites have access mostly to the basics of user profiles. The information that are acquired by the OTAs are often confined with the responses to questions like the followings: Where do they want to go? When do they want to go? How many people are they traveling with? This creates the OTAs inclinations to further consolidate their online recommendation systems, on the one hand, while at the same time having limited knowledge regarding the profiles and preferences of users, on the other, creates a hard problem to solve. Being aware of the problematic, one of the biggest online travel agency Expedia started a competition in an online data science community called Kaggle. The challenge is to build an algorithm to recommend hotel cluster that user might be willing to book based on individual preferences of users.

This capstone project is empirically grounded in a research to create an optimal hotel recommendation systems for Expedia.com customers that are searching for a hotel to book, while scientifically concerns with building an algorithm to recommend hotel clusters regarding the personalized preferences of users. As this project confines itself with the case of Expedia, its material is also limited with the data that is provided by Expedia in Kaggle competition." [13,14]

1.1 **Recommendation systems**

Even though the Recommendation systems (RSs) are well-known in online marketing world for several years, only in the recent years it has made a breakthrough by the ability to make predictions in an accurate manner.

In the most simplistic approach, all relevant information regarding the user choices are gathered by monitoring user behavior or by asking users to building profile with their preferences. The techniques about collecting information is explained in detail within the papers [7,8,9]. RSs filter information regarding user preferences on set of items (eg. movies, songs, books, gadgets, travel destinations). Finally, the system generates a list of recommendations for user.

Recommendation systems typically produce a list of recommended information generally in two main ways: collaborative filtering and content based filtering. Recent studies show there is also a suggestion about hybrid techniques, which makes it a third option to be used by RCs. [8]

Expedia data set that we select to use in this project only cover features related to users search preferences. The data does not have any information related with users likes or preferences. We do not know any solid information on user's choices on hotel type (if he/she like an all-inclusive hotel or a boutique hotel). Since the data set that is available consist of only with real-time search information of users, in this project, we mainly focus on collaborative filtering method because of the limitations in the data that is already discussed.

Collaborative filtering (CF) methods are based on collecting and analyzing used data to generate a model to recommend filtered information to users. These data are mainly collected from user's behaviors, activities or preferences. These systems predict what users will like comparing the similarity with the other users. CF allows users to rate set of elements, then store and analyze these data to recommend best possible options to the user [10, 11, 12]. In our project this rating is gathered on bases of whether the search is end up being a booking.

A key advantage of collaborative filtering is that model does not need to understand what item offered to the user. Since whole approach is based on preference and behavior, this approach is capable of accurately recommending complex items.

This approach assumes that users will keep their preferences from the past to the future. This means that they will like similar kinds of items as they liked in the past.

According to the latest research on collaborative filtering, using side information can improve collaborative filter – based models [3]. Although the fact that these enhanced collaborative filtering techniques show promising results, it is not within the scope of this project because of lack of data.

Content based filtering methods use descriptions of items and profile of the user preferences for making recommendations. User profile is build up to match with items that are described with keywords. The main operating structure of content based filtering is to make recommendation based on users past choices. Various candidate items are compared with items previously rated by the user and the best-matching items are recommended. "If the user purchased some fiction films in the past, the RS will probably recommend a recent fiction film that he has not yet purchased on this website" [2]. This data system analyzes the profile and matches the items that are matching with the similar items that are selected by the user in the past.

One of the most interesting approaches in the literature of recommendation systems is the hybrid recommendation systems.

Papers [5,6] discuss the strengths and weaknesses of knowledge based and collaborative filtering systems in recommendation systems. They propose to introduce a hybrid recommendation system which combines two approaches. They discuss that knowledge based recommendations system can be used as bootstrap of the collaborative filtering engine if the size of the data is small. Furthermore, the collaborative filter can be positioned as post filter of knowledge based recommendation system.

The data does not have enough profile data about user's preferences. Considering this limitation, content based or hybrid approach is not considered to be fit in this project because of the data structure that is available in data set.

2. Project Definition

2.1 Problem Statement

Currently, Expedia uses search parameters to adjust their hotel recommendations, despite the fact that it does not have enough customers to specify data and personalize them for each user. In Kaggle competition, Expedia asked Kagglers to contextualize customer data and predict the likelihood that a user will stay at 100 different hotel groups.

2.2 Project Objectives

2.2.1 Understanding the data

The first phase is the understanding of the data with explanatory data analysis. At this phase, we must find out anomalies, special conditions and correlations between variables. In this part, we have used several python libraries to understand the distribution of data. The data will be cleaned up using pandas [18] and numpy [19] library of python. Visualizations from the explanatory data analysis is done by seaborn [20] and matlibplot [21].

2.2.2 Data Cleaning and Enrichment

The data cleaning is relevant to clean up noise and anomalies in the data. In addition, the row data is not suitable to fed into any kind of machine learning algorithm. Considering that the data set features 37 million observations, we have applied several feature creation and enrichment methods to be able to create more precise predictions.

One important part in this phase is to keep the balance between bias and variance. In this manner, the correlation between variables is explored using skit-learn [17] library feature selection tools.

Finally, with the information from data exploration the imbalanced data is balanced by dropping click data that is creating anomaly in the data set.

2.2.3 Model Selection

In the model selection phase, firstly we have tried the conventional basic machine learning models to create a model to solve the recommendation problem. For this purpose, accuracy scores and mean average precision at 5 are derived to analyze efficiency of the model. Selected model performed below the expected accuracy metrics. Considering this we have reevaluated the model and tune it for more precise results.

2.2.4 Model Tuning

In model tuning phase, we tried to tune the model in order to get better results for our recommendation problem.

2.2.5 **Results and evaluation**

The evaluation of the results created by the model are conducted by mean average precision technique. As a result, the algorithm will recommend 5 clusters according to their probability. The recommendations will be scored due to the average precision at 5 (MAP@5). Number five points out that we will predict and recommend 5 clusters for each user in the data set.

To explain mean average precision, we need to explain average precision first. Average precision can be explained by following example. We are searching for hotel cluster of a hotels and we provide our recommendation system a sample hotel search, we do get back a bunch of ranked hotel clusters (from most likely to least likely). Considering we might be predicting some of the cluster wrong, we compute the precision at every correctly hotel cluster, and then take an average. If our returned result is:

User_Id →	Recommended_Hotel_Cluster_Ids
15,	1, 4, 100, 73, 8
18,	35, 80, 26, 39, 31
36,	16, 44, 25, 34, 25,
etc.	

where the results are hotel clusters that we recommend to user (for the user with Id of 15 we will recommend hotel cluster 1, 4, 100, 73, 8), then the precision at every correct point is: how many correct hotel clusters have been encountered up to this point (including current) divided by the total hotel clusters seen up to this point.

Mean average precision is just an extension, where the mean is taken across all AP scores for many queries, and again the above interpretation should hold true which is between 0 and 1. Mathematical representation and definition of computation can be found in [Appendix A - I].

2.3 **Project Scope**

The scope of this project is to create a recommendation to the users, according to which the given test data that is most relevant. By this way any OTA can maximize their earnings. This project is limited with the creation of the list of recommendations to users that is listed in Expedia kaggle.com data set.

This project does not cover any web interface visualization of recommendation system or implementation to any web site or such.

3. About The Data

Expedia provides logs of customer behavior. These include what customers searched for, how they interacted with search results (click/book), whether the search result was a travel package or done by a mobile device. The data in this competition is a random selection from Expedia and is not representative of the overall statistics.

The data that Expedia shared is listed below as files name.

- train.csv - the training set, which covers user search behavior for training purposes
- test.cv - the test set, which covers user search behavior for testing purposes
- destinations.csv - hotel search latent attributes, which represent a set of destination related information. The data is represented in a form of numerical values which might consist of reviews, hotel scores, etc.

The train and destination data set have;

Column name	Description
date_time	Timestamp
site_name	ID of the Expedia point of sale (i.e. Expedia.com, Expedia.co.uk, Expedia.co.jp, ...)
posa_continent	ID of continent associated with site_name
user_location_country	The ID of the country the customer is located
user_location_region	The ID of the region the customer is located
user_location_city	The ID of the city the customer is located
orig_destination_distance	Physical distance between a hotel and a customer at the time of search. A null means the distance could not be calculated
user_id	ID of user
is_mobile	1 when a user connected from a mobile device, 0 otherwise
is_package	1 if the click/booking was generated as a part of a package (i.e. combined with a flight), 0 otherwise
channel	ID of a marketing channel
srch_ci	Checkin date
srch_co	Checkout date
srch_adults_cnt	The number of adults specified in the hotel room
srch_children_cnt	The number of (extra occupancy) children specified in the hotel room
srch_rm_cnt	The number of hotel rooms specified in the search
srch_destination_id	ID of the destination where the hotel search was performed
srch_destination_type_id	Type of destination
hotel_continent	Hotel continent
hotel_country	Hotel country
hotel_market	Hotel market
is_booking	1 if a booking, 0 if a click
cnt	Numer of similar events in the context of the same user session
hotel_cluster	ID of a hotel cluster

Table 3-2 Train/Test data set features

Column name	Description
srch_destination_id	ID of the destination where the hotel search was performed
d1-d149	latent description of search regions

Table 3-1 Destinations data set features

3.1 **The Train Data**

The train data consists of user's search related information. To have a deeper understanding in this data we just need to refer to Expedia.com web site. First thing that you will meet in the site is a search bar that user can select a destination, check-in, check-out dates and the room details. Backbone of the train data is constructed by these features. The origins of data that is included in to train data set is represented Figure 3-1.

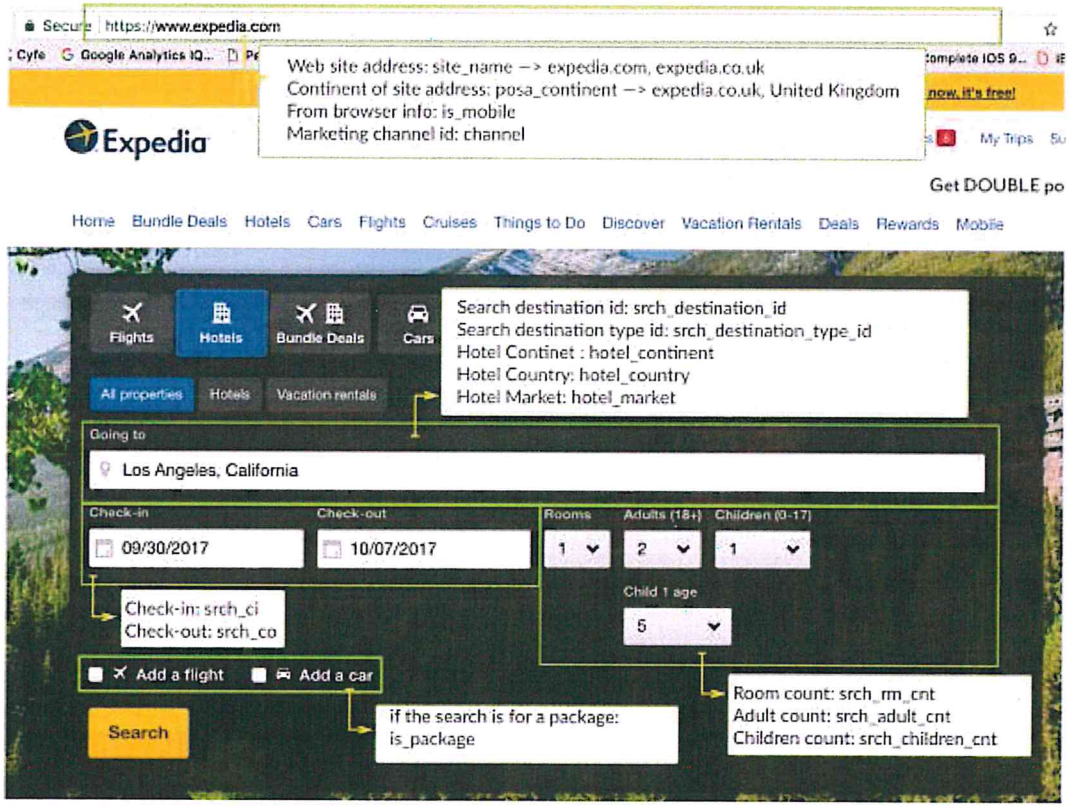

Figure 3-1 Expedia site and origins of features in data set (train)

Some of the user related features like user country (user_ location_ country), user city (user_ location_ city) or user region (user_ location_ region) are gathered by web analytics techniques. Having this kind of data in hand is easy for Expedia. Other features such as (origin_destination_distance) can be drived from these features.

The data that Expedia shared is a good basis to be able predict the hotel clusters that is the main objective of this project.

3.2 **The Test Data**

The test data have the same feature set as train data set. The only missing feature is hotel clusters associated with the given id. Since this data set is taken from a competition, Expedia keeps resulting labels out for evaluation purposes. In the competition, participants/competitors are asked to predict hotel clusters in the test set and create a submission file for evaluation. Since this kind of evaluation is not possible for this project, the test data set is not used in any calculation or model building purposes. For testing purposes we use a sample of train data which is does not have the same users whit in the training set itself.

3.3 **The Destinations Data**

The destinations data set contains an id that corresponds to srch_destination_id, along with 149 columns of latent information about that destination. The competition does not tell us exactly what each latent feature is, but it is safe to assume that it is some combination of destination characteristics, such as name, description. These latent features are converted to numbers, so they could be anonymized.

4. Methodology

4.1 Sampling

The observation count in the data set is around 37 million. Since this project is focused on creating a solution using regular python structure; it is hard to maneuver around the data with a single treated code structure. Because of this reason the data is sampled in various ways to find out also the best sampling approach to this kind of problems.

The structure of sampling methods is listed below;

- Method 1: Random sampling with 1 million observations.
- Method 2: Balanced sampling with 4 million observation equally structured from booking events and click events. (2 million book event, 2 million click events)
- Method 3: Random sampling of 2 million only booking events.

Along with the sampling method listed above we have also tried another one which is random sampling the raw training data by 20% to be able to create training data set and 5% to be able to create test data set. We have used pandas sampling function with different random state variables to ensure that we have a two-different data set.

- Method 4: Random sampling of train data by fraction of 20% for training data, random sampling of train data by fraction of 5% for test data.

We use the sampling method 4 because of the overall consistency that it provides.

4.2 Data Explorations

Data exploration is the first main phase of this project. The aim in the phase is to understand the distribution and the structure of the data. The outcome and resulting observations from this phase is represented in this section.

4.2.1 The Train Data

We have found that training data set has 37.670.293 observations and 24 features in it. There exist tree features as object data type which are time of event, check in and check out date and time. The remaining variables are integers except origin destination distance. List of all variables with their relevant data types is represented in Table 4-1.

Variable	Data Type	Variable	Data Type	Variable	Data Type
date_time	object	is_mobile	int64	srch_destination_id	int64
site_name	int64	is_package	int64	srch_destination_type_id	int64
posa_continent	int64	channel	int64	is_booking	int64
user_location_country	int64	srch_ci	object	cnt	int64
user_location_region	int64	srch_co	object	hotel_continent	int64
user_location_city	int64	srch_adults_cnt	int64	hotel_country	int64
orig_destination_distance	float64	srch_children_cnt	int64	hotel_market	int64
user_id	int64	srch_rm_cnt	int64	hotel_cluster	int64

Table 4-1 Training data, variables and data types

Top five rows of the train data are represented below in 4.2.

	date_time	site_name	posa_continent	user_location_country	user_location_region	user_location_city	...	srch_destination_type_id	is_booking	cnt	hotel_continent	hotel_country	hotel_market	hotel_cluster
0	16.07.2014 9:55:09	2	3	66	189	10067	...	1	0	1	2	50	675	70
1	23.11.2014 17:17:17	30	4	195	991	47725	...	1	0	1	3	151	1236	36
2	8.01.2014 14:09:14	2	3	66	462	41898	...	6	0	1	2	50	680	95
3	20.12.2014 5:23:05	24	2	3	64	9448	...	1	0	1	6	105	29	11
4	25.12.2014 15:32:15	24	2	3	51	9527	...	1	0	1	6	105	35	29

Table 4-2 Top 5 rows of training data

One of the important thing to understand is the distribution of hotel clusters in the data. The data is observed to have 100 hotel clusters. It is seen that some hotel clusters have more observations that others. Since we do not know the basis used in this clustering that is said to be done by Expedia it is hard to make any solid extractions.

The only extraction we might come up from this distribution is that the observation frequency of hotel clusters is not uniform or balanced. This indication creates highly complex problem for project to overcome.

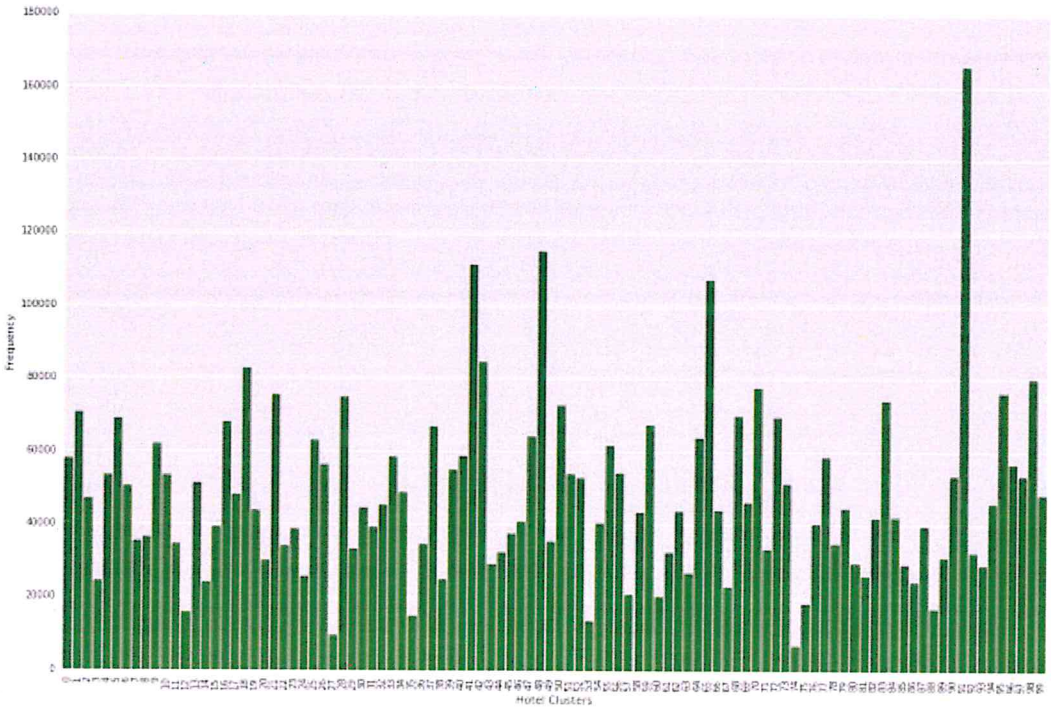

Figure 4-1 Frequency of hotel clusters

We would like to understand how the hotel clusters distributed over continents. It seems that hotel continent 2 has the biggest number of hotel clusters. It is very strong that hotel continent 2 is North America. Also by checking user locations we see that the there is a peak at user location 50 which might be United States.

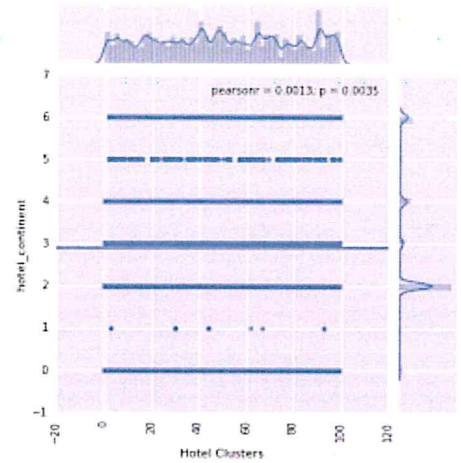

Figure 4-2 Hotel Cluster vs. Hotel Continent

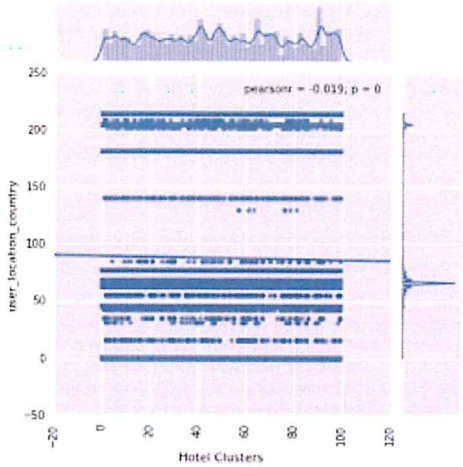

Figure 4-3 Hotel Cluster vs. User Location Country

Further in to explanatory data analysis we try to understand the distribution of booking and click events. As shown by Figure 4-4 much more click event exists in the data set than booking event.

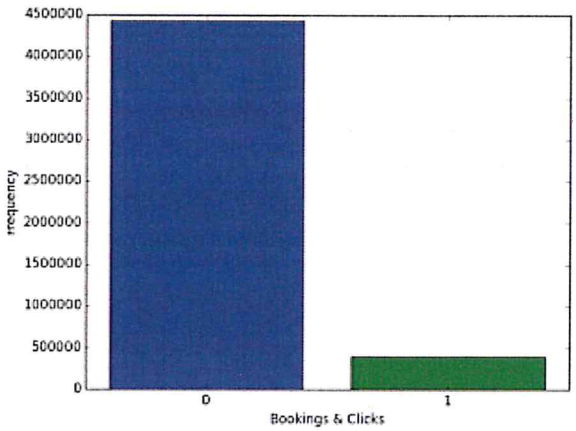

Figure 4-4 Distribution of Booking and Click events

Lastly, for the model building process, we would like to understand the correlation between features and hotel cluster. As by Figure 4-5, there is no significant correlation between any of the features. Keeping this in mind we decide to keep all the features with in the data set for model building phase.

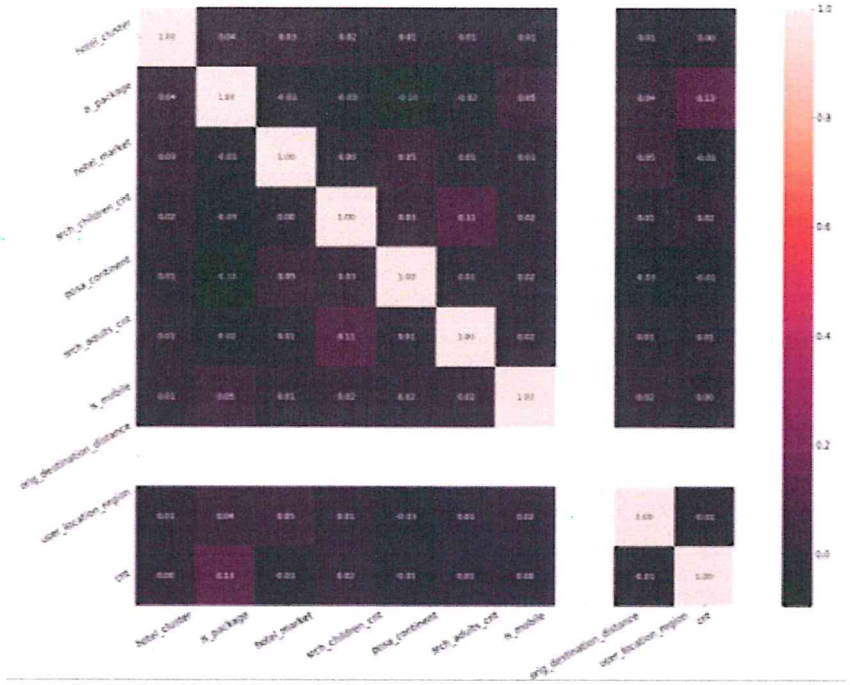

Figure 4-5 Training data - Feature correlation matrix

4.3 **Data Cleaning and Enrichment**

By the knowledge, we gain from explanatory data analysis new features are generated and included in further phases of the model building process.

Because of the huge sample we have, with enough observations that is available, all rows for check in and check out dates that are missing any value is removed from the data. It is considered that the effect of these rows to the overall consistency of the model is negligible.

The other part of the NaN values is mostly available in origin to destination variable. These rows also are also not included into the model and it is also considered as negligible.

We have split the event date variable and create the following variables:
Event seasons; that we use event month to find the event period. The features can get either True: 1, False:0. The features in this part are particularly important because users tend to select a different type of hotel according to the season that they are in.

- event_is_winter: showing if the search event is done in winter time.
- event_is_fall: showing if the search event is done in fall time.
- event_is_spring: showing if the search event is done in spring time.
- event_is_summer: showing if the search event is done in summer time.

Event day classifications: if the search event done in weekdays or weekend. The features can get either True: 1, False:0.

- event_weekday: showing if the search event is done in weekday.
- weekend_event: showing if the search event is done in weekend.

Event date values: We have split and breakdown the check-in and check-out related date information.

- event_month: showing search event month
- event_day: showing search event day
- event_year: showing search event year

Event timing classification: The time of the day influences booking habits. Users tend to have more booking in evenings. This may influence the hotel cluster that they are willing to stay. The features can get either True: 1, False:0.

- event_is_late_night: showing if the search event is done in between 24:00 and 6:00.
- event_is_early_morning: showing if the search event is done in between 6:00 and 8:00.
- event_is_morning: showing if the search event is done in between 8:00 and 10:00.
- event_is_mid_day: showing if the search event is done in between 10:00 and 14:00.
- event_is_afternoon: showing if the search event is done in between 14:00 and 16:00.
- event_is_evening: showing if the search event is done in between 16:00 and 20:00.
- event_is_night: showing if the search event is done in between 20:00 and 24:00.

Check-in, check-out seasons: that we use check-in, check-out month to find the event period. The features can get either True: 1, False:0. The features in this part are particularly important because users tend to select different type of hotel according to the season that they are traveling. The features can get either True: 1, False:0.

- srch_ci_winter: showing if the check-in period is winter.
- srch_ci_summer: showing if the check-in period is summer.
- srch_ci_fall: showing if the check-in period is fall.
- srch_ci_spring: showing if the check-in period is spring.
- srch_co_winter: showing if the check-out period is winter.
- srch_co_summer: showing if the check-out period is summer.
- srch_co_fall: showing if the check-out period is fall.
- srch_co_spring: showing if the check-out period is spring.

Check-in, check-out date values: We have spited and breakdown the check-in and check-out related date information.

- srch_ci_month: showing check-in month
- srch_ci_day: showing check-in day

- srch_ci_year: showing check-in year
- srch_ci_month: showing check-out month
- srch_ci_day: showing check-out day
- srch_ci_year: : showing check-out year

Also, accommodation related features are created based on the search criteria.

night_of_stay: created by subtracting check-in date and check-out date
adult_per_room: creates by dividing adult count by room count
children_per_room: created by dividing children count by room count
person_per_room: created by dividing the total of children and adult count to room count

Finaly we apply principle component analysis to destinations data to reduce the dimensionality of data to 3 features. To be able to include this into model data we have merged the data frame with the train data set.

4.4 **Model Selection**

We have tried to solve this problem with different approaches. One of the best performing approach is to build a model using random forest algorithm.

We have tried building models using algorithms listed below.

- Gaussian Naïve Bayes
- Decision Tree
- K - Nearest Neighbors
- Linear Discriminant Analysis
- Logistics Regression Classifier
- Multi-Layer Perceptron Classifier
- Random Forest
- Adaboos
- Bagging
- Voting Classifier
- Support Vector Machines

4.4.1 **Selected Algorithms**

One of the important limitation in this project is to have an unbalanced data. We have tried to balance the 100 cluster hotel data by sampling but the result is a loss of huge amount of observations. Keeping this in mind we mostly used the algorithms which have ability to handle this situation. Following algorithms have class weight support that one can assign a weight to hotel clusters, according to cluster distribution to balance it with weigth. [24,25,26]

These algorithms are: Random forest, decision tree, K- nearest neighbors, voting classifier.

There is a slight difference in K nearest neighbor's (KNN) algorithm to calculate class weights. We have used KNN attribute that is called "weights = distance". It is described in skit-learn documentation as given below:

> "distance: Weight points by the inverse of their distance. In this case, closer neighbors of a query point will have a greater influence than neighbors which are further away." [24]

This difference in the KNN's calculation algorithm also reflected into the results in favor of poor performance. Although the fact that KNN performs less accurate compared to others, it still performs better than unbalanced models.

Voting classifier does not have any class weight support, but since we have fed random forest, decision tree and KNN to it. It has delivered relatively good result compared to the unbalanced models. It performs little better than KNN algorithm. The performance does not seem enough, so we decide to remove it from further tuning effort.

We use MAP@5 metric to compare that how well the models perform. We set the basis of the performance metric by benchmarking with the results that is listed in the Kaggle site. Decision tree and random forest perform similar which is above 0.3 (which is so we act in favor of further tuning these algorithms.

4.4.2 **Discarded Algorithms**

Below listed algorithms are discarded for various reasons that is explained in this chapter.

- Gaussian Naïve Bayes (GNB)
- Linear Discriminant Analysis (LDA)
- Logistics Regression Classifier (LRC)
- Multi-Layer Perceptron Classifier (MLPC)
- Adaboos
- Bagging
- Support Vector Machines

We use the same benchmarking that we used as we are selecting best performing models. GNB, LDA, LRC and MLPC performed poor with our model since the accuracy score and MAP@5 is below 0,2 we choose to not to further tune these algorithms.

We have also tried adaboost, bagging algorithms with our model. Because of the large sample that we are using, algorithms do not finish the training instead they depleted the memory and force script to kill. Because of this reason, we remove these algorithms from model tuning phase candidates.

SVM is discarded because of the amount of computing time. As described in the skit-learn web site;

> "The fit time complexity is more than quadratic with the number of samples which makes it hard to scale to dataset with more than a couple of 10000 samples." [23]

The outputs of the discarded algorithms show in Figure 4-6;

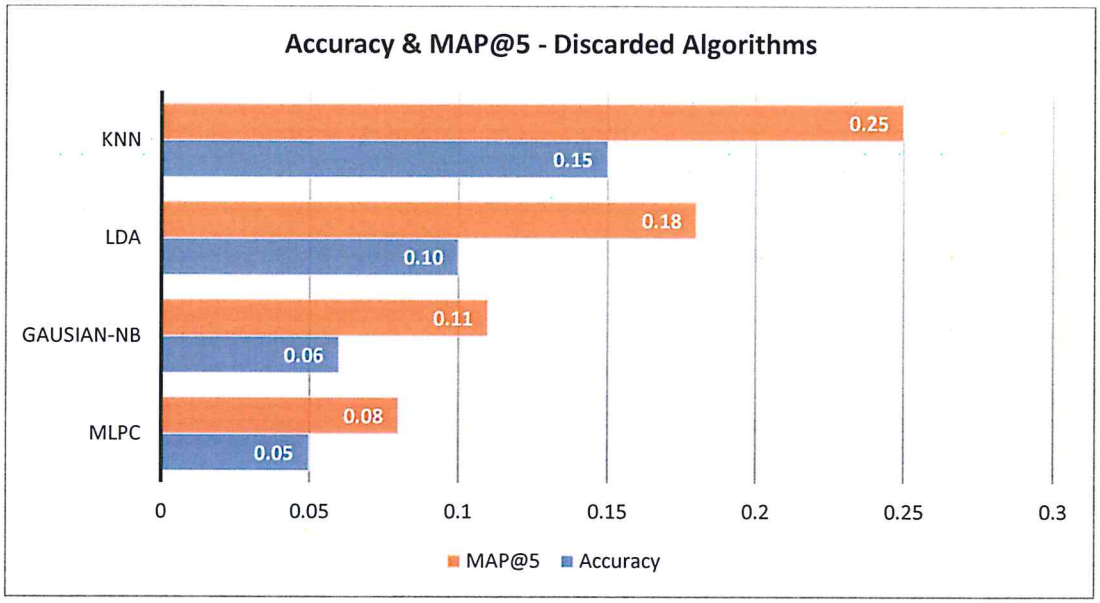

Figure 4-6 MAP@5 & Accuracy output of discarded models

4.5 **Model Tuning**

In this phase, we have worked further in our models that are build using decision tree and random forest algorithms. For both, we have tried several different scenarios where we tune the parameters that we are passing to the function. Below you may find the different variations. All scenarios keep n_jobs parameter set to -1 which is setting the number of jobs to in parallel to number of processor cores for a better computing time. Also, class weight is set to "balanced" because the data set is found out to be unbalanced.

	Model Parmeters						
Model No	n_estimators	min_samples_leaf	max_features	oob_score	min_samples_leaf	MAP@5 Score	Accuracy Score
1.Random Rorest	150	2	None	FALSE	1	0,4215	0,3549
2.Random Rorest	150	10	None	TRUE	1	0,3329	0,2185
3.Random Rorest	150	50	sqrt	TRUE	1	0,3092	0,1850
4.Random Rorest	150	150	sqrt	FALSE	1	0,2752	0,1612

	Model Parmeters						
Model No	max_depth	min_samples_leaf	max_features	splitter	min_samples_leaf	MAP@5 Score	Accuracy Score
1.Decision Tree	None	2	None	Best	1	0,3781	0,3713
2.Decision Tree	None	10	None	Best	1	0,3292	0,2134
3.Decision Tree	None	50	sqrt	Random	1	0,2827	0,1626
4.Decision Tree	None	2	sqrt	Random	1	0,2217	0,1426

Table 4-3 Model tuning matrix for Random Forest and Decision Tree

As clearly seen on the table 4-3, random forest brings the best result with the given parameters. The parameter definitions can be found in [Appendix A - II]

5. Results

The result we gain from two candidate models is promising. Where random forest model out performs decision tree classifier. The scores- calculated using MAP@5 criterion- for all considered methods are shown in Figure 5-1.

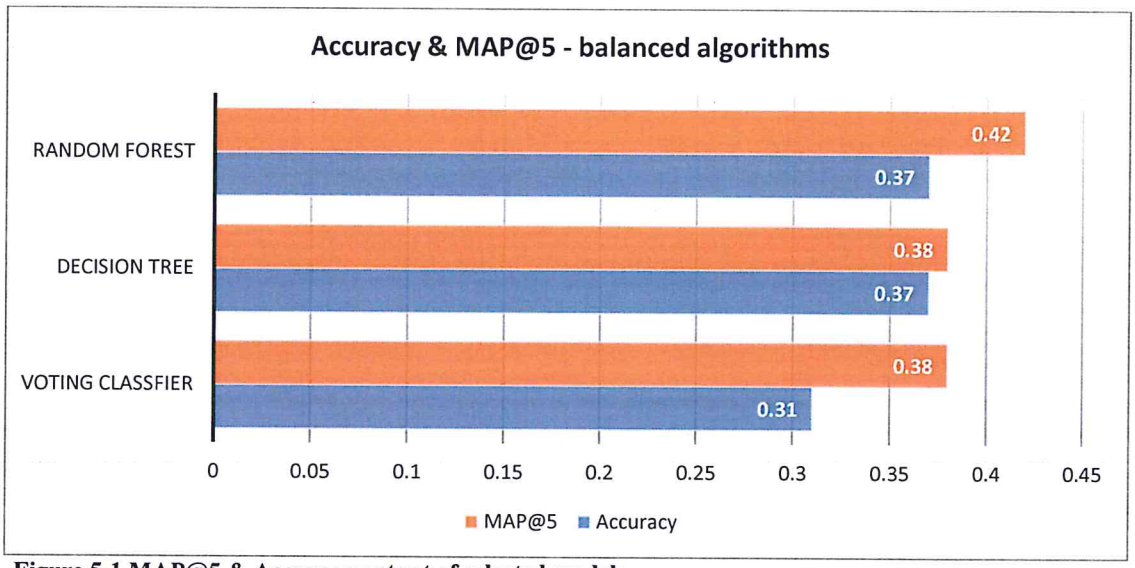

Figure 5-1 MAP@5 & Accuracy output of selected models

One of the algorithms that we use is the Voting classifier. The voting classifier is used to combining conceptually similar algorithms for classification via majority voting. Since we use prediction probabilities to create best matching cluster label set for each user, we select soft voting. In soft voting, we predict class labels by averaging the class probabilities. [27]

We have built a model over voting classifier using random forest, decision tree and KNN expecting to be a better learner than all tree. The voting classifier used with "soft voting" option to be able to handle the probabilistic approach that we are following.

In contrary to our expectation the voting classifier model did not perform as well as expected. We are expecting to have a better result than random forest and decision tree classifier, but it delivered a result approximately same with decision tree classifier.

As expected tree based algorithms perform well under the give circumstances. Because of the nature of the tree based algorithms ability to maximize the information gain the results that are generated by them is relatively good. As an information gain criterion we set it to "gini". We found relevant research that shows as information gain criterion "gini" performs well under similar conditions.[28]

Decision tree classifier performed well in tuning phase to become a candidate to be considered as a usable model for this problem. Although the fact that it is still out

performed by random forest model. The model delivered MAP@5 score around 0.37 which is better a good score considering the complexity of this problem.

The result and the confusion matrix for decision tree classifier is shown in Figure 5-3. It clearly demonstrates, decision tree model's ability to predict true labels.

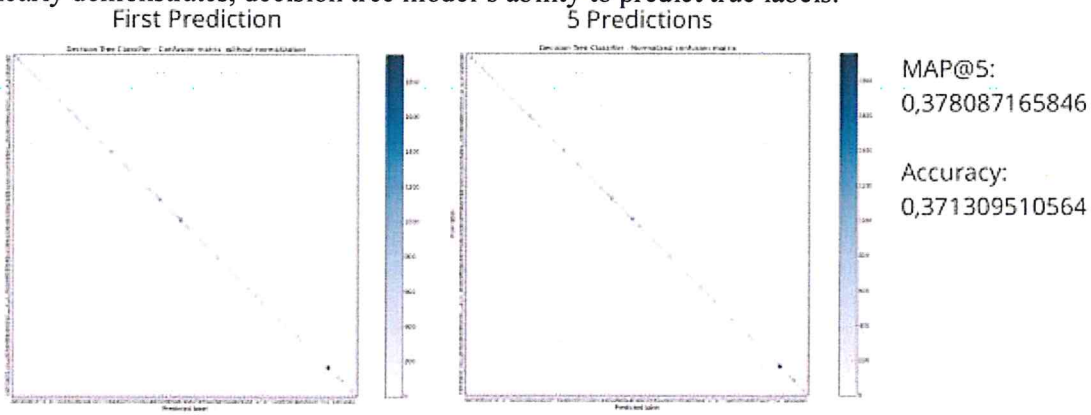

Figure 5-2 Output of Decision Tree Algorithm

The random forest model has critical advantages in this project. One important future of random forest is that it could easily handle unbalanced data. They also do not require binary encoding of features since they can learn that those features are categorical just by training. Moreover, their weak learner trees are equally likely to consider all variables. This is crucial in the problem because there is a few features which are very dominant over others. Therefore, other models always pick those dominant features first to make decisions. Conversely, for each grown weak tree, random forests randomly determine the set of features to be used. So, every feature has a chance to provide information.

Using all the advantages that is described above random forest became the best performing algorithm for the created model. The metrics and confusion matrix for random forest is as given below.

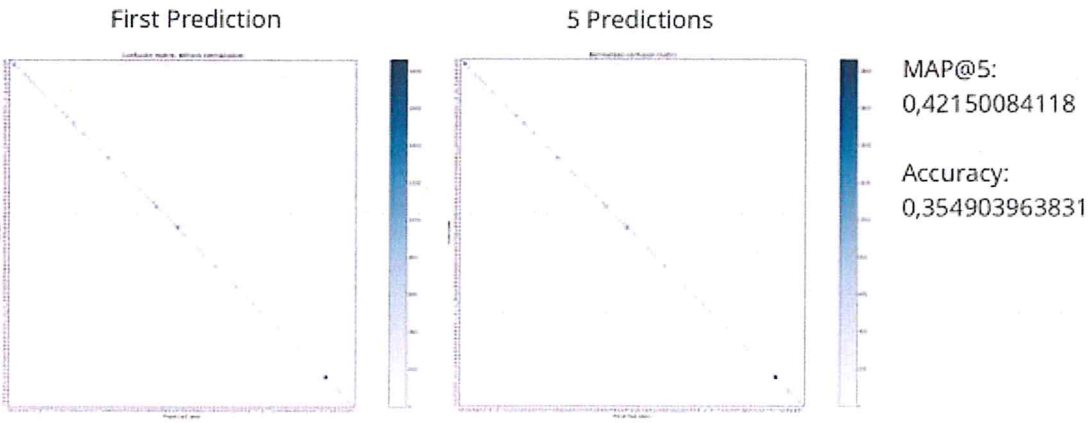

Figure 5-3 Output of Random Forest Algorithm

Compared to decision tree classifier it performed better. Since we are working on a problem with a 100 cluster, event smallest bit of performance is important and make a huge difference in the real-world practice. For this problem, we clearly choose to use random forest model.

5.1 **Conclusion and Next Steps**

The training dataset was analyzed by various machine learning algorithms that helped us come up with classification models for the hotel recommendations system. The dataset has multiple classes without any significant perceived pattern that relates them to the features. Moreover, most of the data is anonymized and missing which restricts the feature creation significantly.

The most challenging part of the implementation is to create meaningful features out of the data set. The exploration of the data, took a long time because of the size of the data set.

This initially made it difficult to achieve reasonable accuracy. After we applied Clustering and Ensemble methods, we could achieve noticeable increase in accuracy and mean average precision. The highest observed was by the random forest, as it handles the unbalance data efficiently.

Considering the complexity of the problem at hand, there is a lot of room for improvement in the future work. Firstly, the data set that is shared can be used better if the coding structure modified in a distributed system like pySpark. A Hadoop cluster can process this kind of huge data easily. With a distributed system (which is not in scope of this project) more complex models can be build. In general, any computing which employs in-memory computing can deliver faster results with reliable outcomes.

Our proposal for the next step of this project is to employ a computing system with in memory support.

6. Social and Ethical Aspects

The recommendation system usage by OTA can be discussed several ways. The first point in the usage of these systems is to optimize profit flow by increasing the conversion rate of the users. In this manner, most of the OTA's adjust their recommendation system in the favor of highly converting hotels or hotel clusters.

This fact is open to discussion in an ethical manner. Is it ethical to knowingly divert users to highly converting hotels, even though their choice might match better with some other hotel? In this situation, it is OTA's decision to better satisfy user's needs, and create a plain game field for hotels, or optimize their income.

Lately new generating of OTA which is concentrated on the P2P business model, prefer to make their decisions more in the favor of better satisfying their user's needs. This approach might be the starting point for creating more transparent interaction between OTA's and their users.

Another important fact is that recommendation systems are bounded by their algorithms performance. It is not likely for an algorithm to recommend a single traveler a hotel that is highly preferred by families. This fact has a potential to limit individual's potential by not letting the user experience different experiences.

This kind of recommendations systems keeps OTA users always in their comfort zone. Which also open for discussion on a social dispute. Is it socially meaning full to limit the expectation of user according to past preferences? Or with the users that have similar preferences?

On the other hand, since travel is a leisure activity, we might consider that users might want to stay in their comfort zone.

As a conclusion, the discussion about social and ethical aspects of recommendation systems is a complex field. Profit versus user preferences or comfort zone versus new experiences, these facts are always bounded with the algorithms capabilities. We believe with the latest developments in the machine learning technologies this kind of multi-tenant algorithms will emerge. [12]

7. Value Delivered (Contribution)

Once this competition is published in Kaggle.com by Expedia most of the competitors find a data leak in the data set. Which users that employ this approach create outstanding MAP@5 scores with just using one feature. Since this is a competition people who employ this method is grow exponentially. The resulting leaders of the competition win by employing this approach. Since this is a real-life problem where the chance of finding out a data leakage is low it is not possibly applicable.

Our contribution in this problem is to create a model without employing the data leakage approach. Although the fact there exist similar work to overcome this problem with a relevant machine learning model, it is still not enough to create a basis solve this problem in a more accurate way.

The feature creation process to enrich the data is have an important role in this process. Created features can easily be used in real life problems to extend the one's ability to understand and explore further details about their users.

We building our models based on involved ranking of clusters by their predicted class probabilities which seems fair. Also, used their ability to handle unbalanced data by assigning weights to certain clusters.

Well performing models in recommendation systems as ours are mostly developed, having the idea of creating a business value out of it. They are not open-sourced, instead they are sold by companies who developed them. Our approach is to provide a well performing model to the open-source community, where other people can benefit and create something special by employing our model as a basis.

In this manner, the model that we have coded can be found in Github as an open-source library which is open to any ones use for free. [Appendix B]

APPENDIX A

I. Mean Average Precision

Average precision is defined as below;

$$AP@5 = \frac{\sum_{k=1}^{5} P(k)}{\min(m, 5)}$$

where P(k) means the precision at cut-off k (amount of suggestions that we will make for one user) in the item list, and m is the number of relevant items in the list. P(k) equals 0 if k -th item is not relevant. If the denominator is 0, the result is set 0.

The mean average precision for N users at position 5 is the average of the average precision of each user, i.e.,

$$MAP@5 = \sum_{i=1}^{N} AP@\frac{5_i}{N}$$

II. Random Forest Parameters

n_estimators : integer, optional (default=10)
The number of trees in the forest.

min_samples_leaf: int, float, optional (default=1)
The minimum number of samples required to be at a leaf node:

- If int, then consider min_samples_leaf as the minimum number.
- If float, then min_samples_leaf is a percentage and ceil (min_samples_leaf * n_samples) are the minimum number of samples for each node.

oob_score: bool (default=False)
Whether to use out-of-bag samples to estimate the generalization accuracy.

class_weight: dict, list of dicts, "balanced", "balanced_subsample" or None, optional (default=None) Weights associated with classes in the form {class_label: weight}. If not given, all classes are supposed to have weight one. For multi-output problems, a list of dicts can be provided in the same order as the columns of y.

Note that for multioutput (including multilabel) weights should be defined for each class of every column in its own dict. For example, for four-class multilabel classification weights should be [{0: 1, 1: 1}, {0: 1, 1: 5}, {0: 1, 1: 1}, {0: 1, 1: 1}] instead of [{1:1}, {2:5}, {3:1}, {4:1}].

The "balanced" mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as n_samples / (n_classes * np.bincount(y))

The "balanced_subsample" mode is the same as "balanced" except that weights are computed based on the bootstrap sample for every tree grown.
For multi-output, the weights of each column of y will be multiplied.
Note that these weights will be multiplied with sample_weight (passed through the fit method) if sample_weight is specified.

max_features: int, float, string or None, optional (default= "auto")
The number of features to consider when looking for the best split:

- If int, then consider max_features features at each split.
- If float, then max_features is a percentage and int (max_features * n_features) features are considered at each split.
- If "auto", then max_features=sqrt(n_features).
- If "sqrt", then max_features=sqrt(n_features) (same as "auto").
- If "log2", then max_features=log2(n_features).
- If None, then max_features=n_features.

Note: the search for a split does not stop until at least one valid partition of the node samples is found, even if it requires to effectively inspect more than max_features features. [24]

III. Decision Tree

splitter: string, optional (default= "best")
The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

max_depth: int or None, optional (default=None)
The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

max_features, class_weight, min_samples_leaf and min_samples_leaf is given in the [Appendix A - II] and same as random forest classifier parameters. [25]

APPENDIX B

We have maintained a GitHub repository for all implementations related to this project. All the changes have been pushed to the following repository:

Project hub in Github - https://github.com/cemkilicli/bda_capstone_project

We have used Python as our development language, and the interpreter is Python 2.7. The following additional package is used for helping with the data-processing, and graph plotting tasks:

- Pandas – for data processing [18]
- Numpy – for data processing [19]
- Matplotlib – for plotting graphs [20]
- Seaborn – for plotting graphs [21]

The following library is used as an aid to perform machine learning algorithms;

- Skit-Learn – for machine learning algorithms [17]

REFERENCES

[1] Wikipedia - Recommender system https://en.wikipedia.org/wiki/Recommender_system

[2] J. Bobadilla, F. Ortega, A. Hernando, A. Gutiérrez, Recommender systems survey Madrid, 2013

[3] A. Almahairi, K. Kastner, K. Cho, A. Courville, Learning distributed representations from reviews for collaborative filtering, in: 9th ACM Conference on Recommender Systems (RecSys '15), 2015, pp. 147–154.

[4] D. Jannach, M. Zanker, A. Felfernig, G. Friedrich, Recommender Systems: An Introduction, Cambridge University Press, New York, 2011.

[5] R. Burke, Integrating Knowledge-based and Collaborative- filtering Recommender Systems, AAAI Technical Report WS- 99-01, pp. 69–72.

[6] R. Burke, Knowledge-based recommender systems, in: A. Kent (Ed.), Encyclopedia of Library and Information Systems, 69 (32), Marcel Dekker Publisher, 2000.

[7] S.K. Lee, Y.H. Cho, S.H. Kim, Collaborative filtering with ordinal scale-based implicit ratings for mobile music recommendations, Information Sciences 180 (11) (2010) 2142–2155.

[8] K. Choi, D. Yoo, G. Kim, Y. Suh, A hybrid online-product recommendation system: combining implicit rating-based collaborative filtering and sequential pattern analysis. Electronic Commerce Research and Applications, in press, doi: 10.1016/j.elerap.2012.02.004.

[9] E.R. Núñez-Valdéz, J.M. Cueva-Lovelle, O. Sanjuán-Martinez, V. Garcia-Diaz, P. Ordoñez, C.E. Montenegro-Mar´ın, Implicit feedback techniques on recommender systems applied to electronic books, Computers in Human Behavior 28 (4) (2012) 1186–1193.

[10] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions, IEEE Transactions on Knowledge and Data Engineering 17 (6) (2005) 734–749.

[11] J.L. Herlocker, J.A. Konstan, J.T. Riedl, L.G. Terveen, Evaluating collaborative filtering recommender systems, ACM Transactions on Information Systems 22 (1) (2004) 5–53.

[12] X. Su, T.M. Khoshgoftaar, A survey of collaborative filtering techniques, Advance in Artificial Intelligence 2009 (2009) 1–19.

[13] Expedia kaggle.com page https://www.kaggle.com/c/expedia-hotel-recommendations

[14] Expedia kaggle.com data set https://www.kaggle.com/c/expedia-hotel-recommendations/data

[15] Google Cloud DataLab Service, https://cloud.google.com/datalab/

[16] JetBrains IDE – PyCharm, https://www.jetbrains.com/pycharm/

[17] Skit-Learn, API Documentation, http://scikit-learn.org/stable/

[18] Pandas, API Documentation, http://pandas.pydata.org/

[19] Numpy, API Documentation, http://www.numpy.org/

[20] Seaborn, API Documentation, https://seaborn.pydata.org/

[21] Matlibplot, API Documentation, https://matplotlib.org/

[22] ML Metrics, API Documentation, https://github.com/benhamner/Metrics/tree/master/Python/ml_metrics

[23] Skit-Learn, Support Vector Machines API Documentation, http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC

[24] Skit-learn, Random Forest API Documentation, http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[25] Skit-learn, Decision Tree API Documentation, http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

[26] Skit-learn, K-Nearest Neighbors API Documentation, http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[27] Skit-learn, Voting Classifier API Documentation, http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html

[28] L.E. Raileanu, Formalization and comparison of split criteria for decision trees, Ph.D. thesis, University of Neuchâtel, Switzerland (May 2002).

Glossary

I. List of Tables

Table 3-1 Destinations data set features 6
Table 3-1 Train/Test data set features 6
Table 4-1 Training data, variables and data types 10
Table 4-2 Top 5 rows of training data 10
Table 4-3 Model tuning matrix for Random Forest and Decision Tree 16

II. List of Figures

Figure 3-1 Expedia site and origins of features in data set (train) 7
Figure 4-1 Frequency of hotel clusters 10
Figure 4-2 Hotel Cluster vs. Hotel Continent
Figure 4-3 Hotel Cluster vs. User Location Country 11
Figure 4-4 Distribution of Booking and Click events 11
Figure 4-5 Training data - Feature correlation matrix 12
Figure 4-6 MAP@5 & Accuracy output of discarded models 16
Figure 5-1 MAP@5 & Accuracy output of selected models 17
Figure 5-2 Output of Decision Tree Algorithm 18
Figure 5-3 Output of Random Forest Algorithm 18