SMU
SINGAPORE MANAGEMENT UNIVERSITY

IS424 - Data Mining

Final Report

Instructors:

Professor Roy Lee Ka Wei

Professor Chiang Meng Fen

Prepared by: G1 Team 5

Project Title: Airbnb First Destination Booking Prediction

Group Members:

Austin Woon Quan	austinwoon.2017@sis.smu.edu.sg
Ong Jun Wen Brendan	brendan.ong.2017@sis.smu.edu.sg
Sean Kwek Si Wei	sean.kwek.2017@sis.smu.edu.sg
Tho Shu Ting Belinda	belinda.tho.2017@sis.smu.edu.sg
Yee Bing Cai Shanicus	shanicusyee.2017@sis.smu.edu.sg

Executive Summary

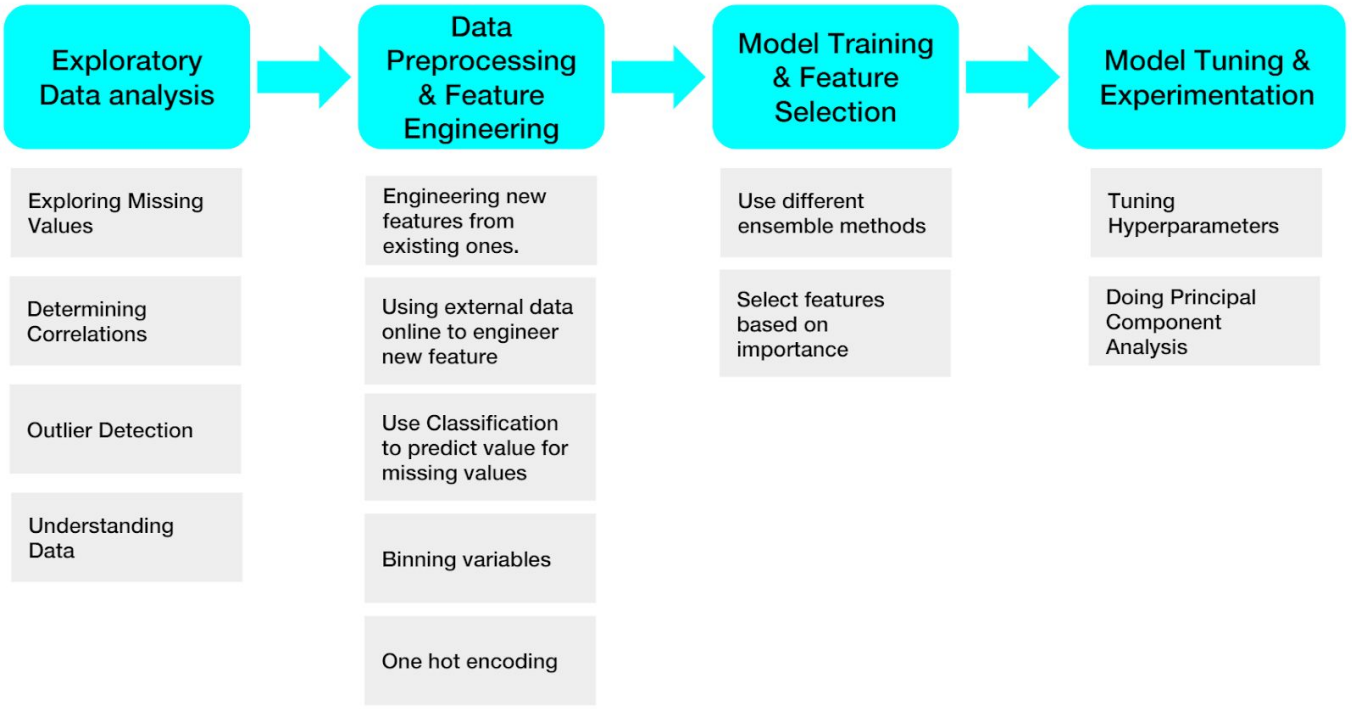
In today's context, Uber, the world's largest taxi company, owns no vehicles. Facebook, the world's most popular media owner, creates no content. Alibaba, the most valuable retailer, has no inventory. And Airbnb, the world's largest accommodation provider, owns no real estate, it is not easy to decide where to have your first trip with so many options on hand. However, with the help of our algorithm and Airbnb's platform, users will be guaranteed to have an unforgettable stay via Airbnb. Airbnb is now able to offer you personalised options to aid you in your virgin Airbnb booking experience.

The team found out that Airbnb has a plenty of untapped markets in the form of the older age groups as these less tech-savvy group of users have lower rates of access to online booking platforms. Also, Airbnb can potentially focus its marketing efforts on a wider scale internationally as its current bookings are dominated by US-based accommodations. It can explore marketing other destinations to its users as well as increase marketing efforts in to promote more users to become hosts in other destinations. Tailoring content to day of creation is extremely important as create day was the most important feature and this will significantly boost sales.

This report provides an analysis and evaluation on where new AirBnB users make their first booking and gain insights into which key factors are vital in determining the countries of first booking. The method used for this analysis is data mining task called classification. The classification method considers every factor in the dataset and predicts the country accordingly. Due to the large dataset and memory space needed to optimise the classification task, the team's limited memory size computers will be a concern during this study. Nonetheless, the team has successfully achieved an outstanding result of 0.88039 accuracy score. In other words, on average we are able to predict the first country booking of a new Airbnb user correctly 88.03% of the time. This will have great impact on their business model as they can customise the users interaction and experience on its applications individually based on the predicted destination of the user, thereby increasing customer conversion rate on their website.

Technical Summary

This project is a classification task where we are tasked with classifying where new users will travel to next on the Airbnb Platform. There are a total of 12 labels. The overall data size was about 2gb and had a total of 10,875,100 observations and 37 features. This project follows the data mining pipeline below:



We started with EDA to find out what data preprocessing we have to do and what correlations are in the dataset. Then, we did data preprocessing and explored new avenues for preprocessing such as :

1) Implemented machine learning techniques to predict missing values using other features
2) Engineering new features by combining multiple existing features
3) Using external data online to engineer new features
4) Filling in missing data with the Median value
5) Categorizing certain ranges to bins

After preprocessing, we merged user and sessions dataframes and the end result was a total of 959 features and around 200,000 observations.

For model implementation, we explored new models not taught class such as LightGBM, XGBoost, SVM and Random Forest. However, one major technical limitation for our project was that our computing resources were not strong enough to run certain models like XGBoost and Support Vector Machines (RBF). As such, we chose an ensemble technique that required less computing power, LightGBM, a lightweight gradient boosting framework.

After initial training, we ran feature selection to choose the n most important features in order to cut model run time and improve accuracy (NDCG Score). This model (cut by 450 features) cut our runtime by 400% and achieved similar accuracy than the model with all features selected. Our final score achieved was 0.88039 (top 10% of public leaderboards scores, first place was 0.88209) achieved with LightGBM technique and the decision tree model. [1]

For further experimentation, we explored Principal Component Analysis to combine multiple features in hopes of getting a faster run time without compromising accuracy score. We also dabbled with hyper parameter tuning to see if it helps increases our NDCG score. Lastly, we also used traditional state of the art ensemble methods such as Adaboost to see if they fare better compared to our new models.

Data

Dataset Name	No. Cols	No. Rows	Features
age_gender_bkts	5	420	age_bucket, country_destination, gender, population_in_thousands, year
countries	10	7	Country_destination, lat_destination, lng_destination, distance_km, destination_km2, destination_language, language_levenshtein_distance
sessions	6	10,600,000	User_id, action, action_type, action_detail, device_type, secs_elapsed
test_users	15	62,100	Id, date_account_created, timestamp_first_active, date_first_booking, gender, age, signup_method, signup_flow, language, affiliate_channel, affiliate_provider, first_affiliate_tracked, signup_app, first_device_type, first_browser
train_user_2	16	213,000	Id, date_account_created, timestamp_first_active, date_first_booking, gender, age, signup_method, signup_flow, language, affiliate_channel, affiliate_provider, first_affiliate_tracked, signup_app, first_device_type, first_browser, country_destination

[1] Guolin Ke, Q. M.-Y. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In Q. M.-Y. Guolin Ke, & I. G. Garnett (Ed.), *Advances in Neural Information Processing Systems 30 (NIPS 2017)* (pp. 3146-3154). California: Curran Associates, Inc.

1. Introduction

With the advent of social media and internet in our lives, more individuals aspire and want to experience the thrills and joy they get from travelling to various countries. However, standing in their way of pursuing their wants is the cost of travel (Transport, food, accomodation). Airbnb saw this as the perfect window of opportunity to disrupt the travel industry years ago and set up their company to offer a wallet-friendly alternatives to satisfy the growing need and demand for affordable accommodations.

With over 150 million users[2] and 2.9 million hosts, there is a need to help personalize individual user content to enhance their booking experience. Hence, Airbnb launched a kaggle airbnb competition challenging others to try and predict where new users will travel to next. This helps them curate content for users as well as better forecast demand for Airbnb.

By being able to forecast demand through accurate predictions, Airbnb will be able to better allocate their resources more efficiently hence increasing customer satisfaction by understanding what they want and providing them with the right information. Airbnb will also be able to reach out to first time users and expand their customer base.

In this project, the team will address a classification task on the first country a new airbnb will book their first travel to. Also in this study, the team aims to find important features that motivates and determine how new Airbnb users choose their first destination. The motivation behind this project we believe is the fact that we are millenials. Working on a project that reminds us of a holiday, adventure and fun attracted us to work on this problem and instead of other problems. In addition, we have always been in awe on predictive analytics, how it can help gain a competitive advantage and increase production and operational efficiency.

2. Problem Definition

This project is a classification task where we are tasked to classify 12 labels (['US', 'FR', 'CA', 'GB', 'ES', 'IT', 'PT', 'NL','DE', 'AU', 'NDF' (no destination found)]) on new users.

It takes a long time for users to look for their first travel destination. With over 34, 000+ cities across 190+ countries to choose from, users have a hard time deciding where to travel to. Airbnb does not have the resources to provide personalized content to users to aid in their decision making.

Having a prediction of where the user will travel during their first trip will allow Airbnb to reduce the time for users to make their booking. Airbnb manage demand better by forecasting where users will book their first trip.

With increasing competition within the travel industry, Airbnb wants to strategically improve its user experience. Currently, users are taking a long time to decide on their first travel destination. Having a prediction of where the user will travel for his/her first trip allows Airbnb to offer a more personalized content for each user. By effectively matching user preference, the users will find it easier book their first trip instead of sieving through 34,000+ cities across 190+ countries presented to them in the application.

3. Exploratory Data Analysis

For our project, we used the train, test and sessions dataset. The train dataset consist of 213451 rows and 16 columns while the test data set consist of 62096 rows and 15 columns. We combined both train and test sets and to do EDA on the users.

[2] Smith, C. (2019, March 31). *Airbnb Statistics and Facts (2019) | By the Numbers.* Retrieved from DMR: https://expandedramblings.com/index.php/airbnb-statistics/

Firstly, we analysed the number of null values in the dataset as shown in the tables below.

Null Values

affiliate_channel	0
affiliate_provider	0
age	116866
country_destination	62096
date_account_created	0
date_first_booking	186639
first_affiliate_tracked	6085
first_browser	44394
first_device_type	0
gender	129480
id	0
language	1
signup_app	0
signup_flow	0
signup_method	0
timestamp_first_active	0
dtype: int64	

Fig 1.1: Number of Null Values

affiliate_channel	0.000000
affiliate_provider	0.000000
age	0.424124
country_destination	0.225355
date_account_created	0.000000
date_first_booking	0.677340
first_affiliate_tracked	0.022083
first_browser	0.161112
first_device_type	0.000000
gender	0.469902
id	0.000000
language	0.000004
signup_app	0.000000
signup_flow	0.000000
signup_method	0.000000
timestamp_first_active	0.000000
dtype: float64	

Fig 1.2: Percentage of null values

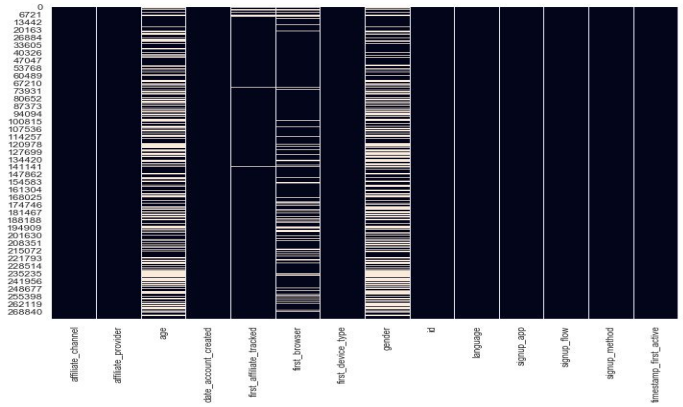


Fig 1.3: Visualization on missing data

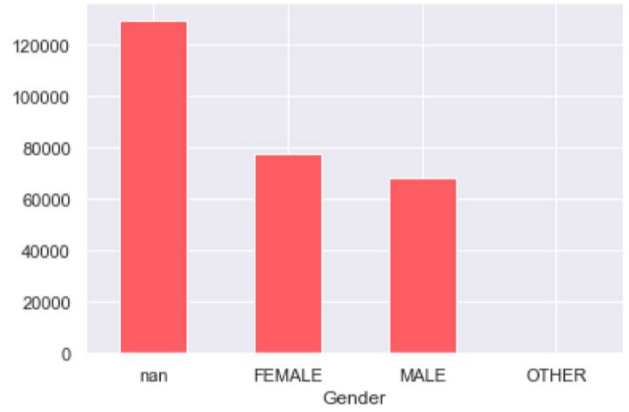


Fig 1.4: Gender Counts

As seen from the visualization in Fig 1.3, data for age and gender are especially sparse. We took note of this and knew that we had to do something to reduce the sparsity of the dataset.

After exploring for null values, the team explored for irrelevant values. Referring to Fig 1.4, there is extremely high number of Nan values as compared to male and female. Similarly with the feature 'Destination Country', there are 60% of users that have not booked any trips as seen in Fig 1.5. We further split the count of people and separated the users by genders shown in Fig 1.6. From Fig 1.5 and 1.6, we can infer that most users book trips to US. The reason for this finding could either be because most users in the data set prefer to travel to US or because there is a possibility that most of the users are from US, thus it is cheaper to book a trip domestically. We continued the analysis with the aforementioned possibility in mind.

Destination Country

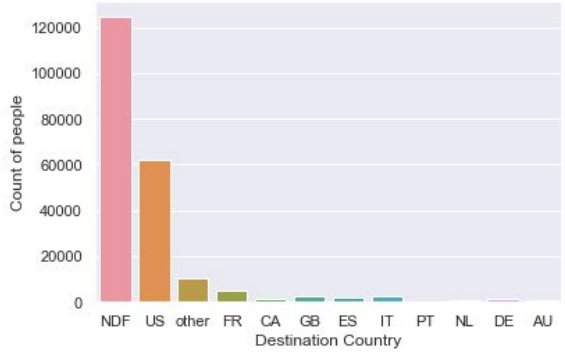


Fig 1.5: Destination country count

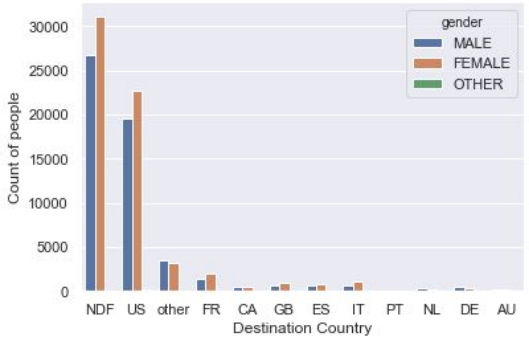


Fig 1.6: Destination country count gender split

Following which, we explored users who have booked a destination on Airbnb. Referring to Fig 1.7, more than 40% of user bookings originate from US, thereby correlating with our aforementioned data of high number of users booking US as their first destination. This could be possibly due to the cheaper costs associated with domestic travel. There is also approximately 60% of users who have not booked any destination before, this could be due to the users not being sure on where they prefer to book or the lack of suggestions by Airbnb to them to aid them in their decision making process.

In terms of gender profile, there is a relatively even split between both genders, indicating no significant difference in terms of booking preferences by gender.

User Booking Origin

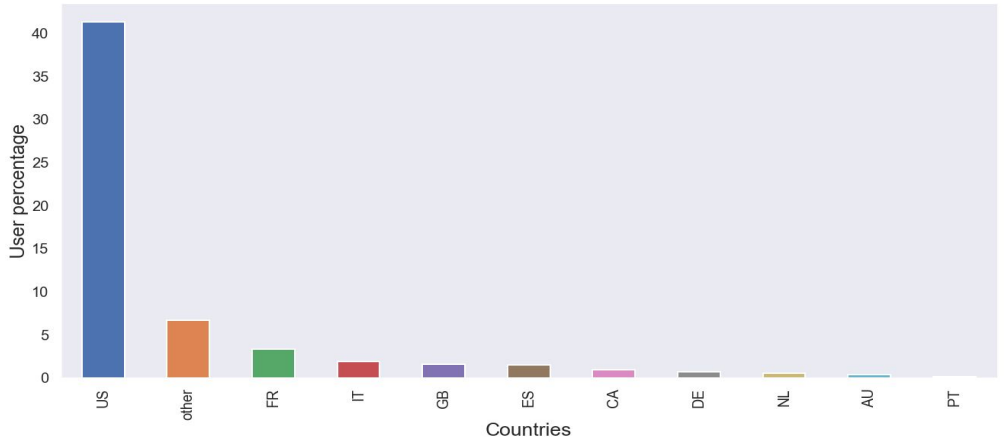


Fig 1.7: User Bookings Origin

Age

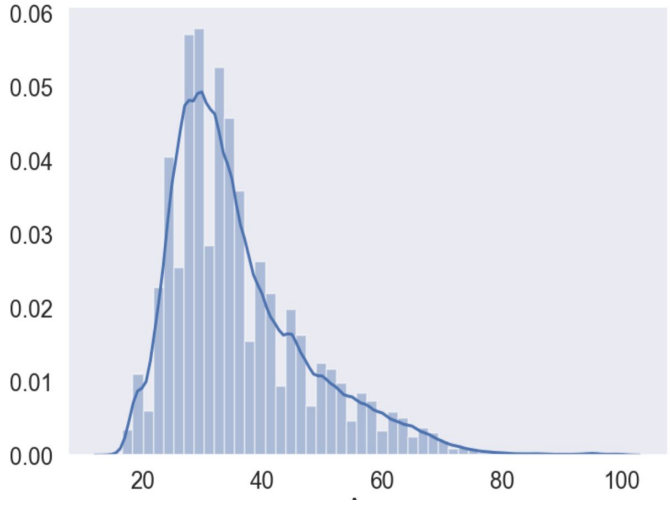


Fig 1.8: Age Distribution

Continuing with the exploratory on the demographics, the team processed age of users for the dataset. In Fig 1.8, we can visualize that most of the travellers are from the age group of 20 to 40 with the highest being approximately 30. The data set has almost no users below 20 years old. This is not surprising as most the travellers booking for a destination would only do so only if they are independent, matured or capable to do. Also, it is expected that most users fall in the 20-40 age group, these group of users tend to be more tech-savvy and hence are more likely to access Airbnb services which are only accessible online. However, we found that that there are outlier in age. We take appropriate actions in the processing stage.

In order to better understand how age and country correlates, the team binned age into 3 categories: 20-30 being ‘Young’, 30-50 being ‘Middle-aged’ and >50 being ‘Elderly’, shown in Fig 1.9. From figure Fig 1.5, we already understood that there is a high percentage of NDF values and most of the users are from US; What we learnt from Fig 1.9 is that there is no country that has one age group dominating the others, with all the countries having a good mix of all.

Age

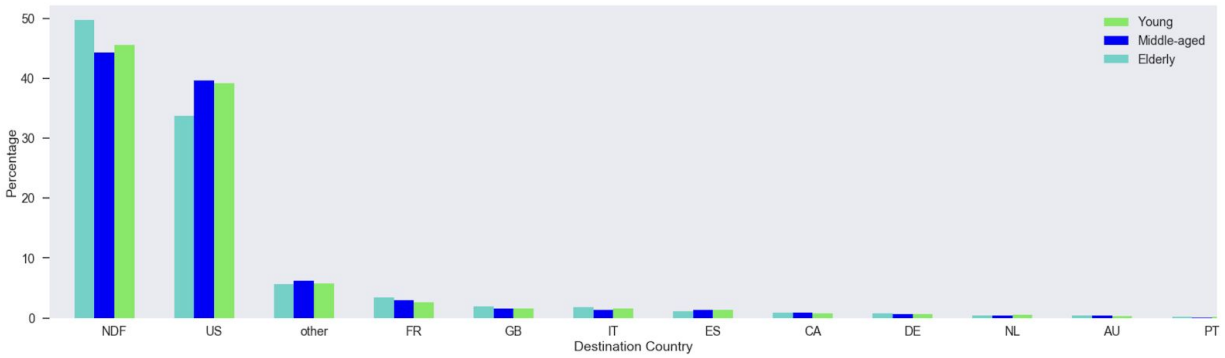


Fig 1.9: Age Distribution by Countries

Subsequently, we continue with an in-depth analysis on languages spoken by the users as it may be an important factor in determining where one will decide to travel to. From Fig 1.10, it is not surprising that english is the most common language as US-based users have dominated the dataset. Hence, in Fig 1.11, we removed english from the data to have a clearer picture of the distribution of other languages.

Language

Fig 1.10: Language Distribution

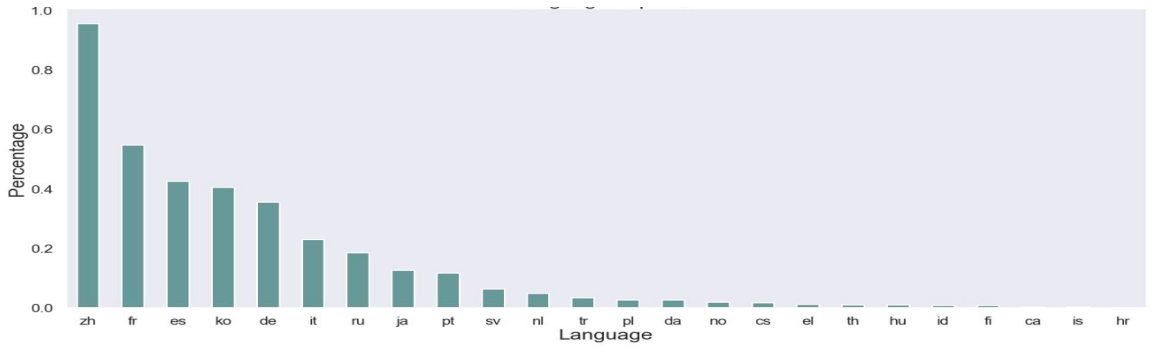


Fig 1.11: Language Distribution without English

From Fig 1.10 and 1.11, we learnt that the second most common language used is chinese. This confirms that the country destination of others would mostly refer to Asia. This also shows that there is a high correlation between language used and country destination.

Moving on, we explored when users made their first booking. Looking from Fig 1.12, there is a peak in mid 2014 followed by a sudden drop. This could be due to the MH370, widespread of Ebola and an increase of short-term rental tax in US. These events not only negatively affected confidence in air travel and travel, it also raised prices across rental rates in US, directly hiking Airbnb rental rates thereby leading to a decrease in demand in for Airbnb bookings. In addition, bookings peak during the middle of year (summer) and in the end of year (december holidays). Hence, we deduced that there might be some correlation between booking rates and holiday periods.

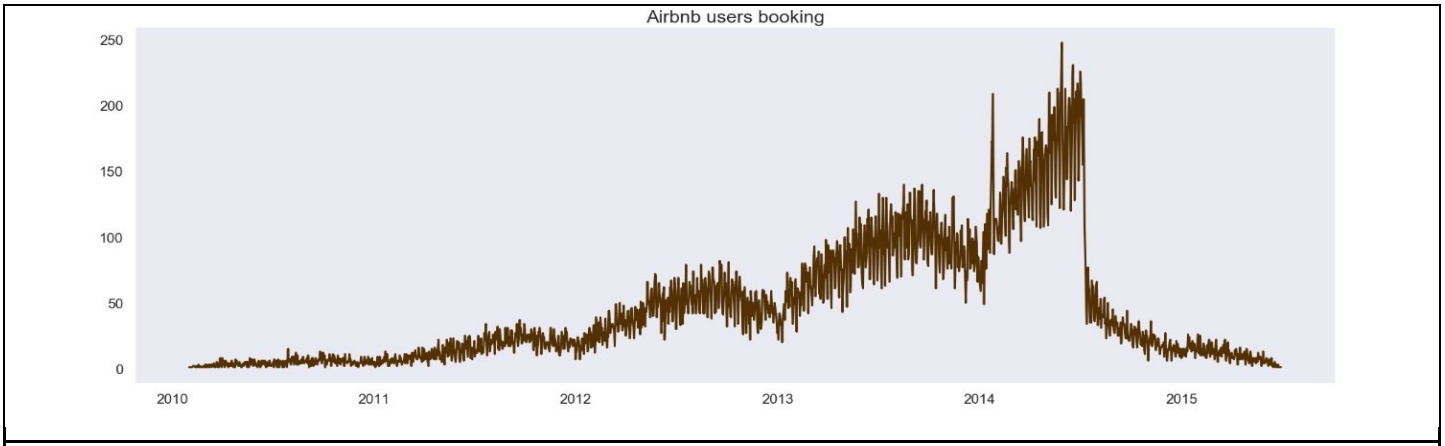


Fig 1.12: Time graph of users booking by year

After our EDA on the user dataset, we move on to sessions.csv. Firstly, we explored how users made their bookings. Sessions EDA figures are shown from Fig 1.13 to 1.17. Through this process, we learned that most of the users sign up with a basic method directly through chrome on Mac desktop.

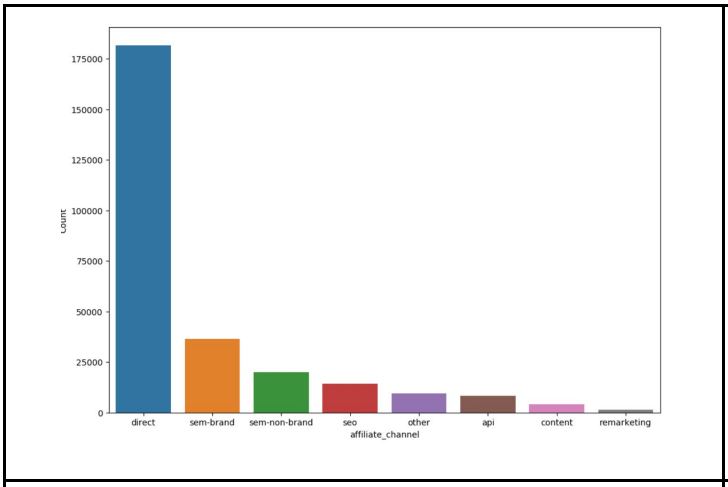


Fig 1.13 Counts of affiliate channel

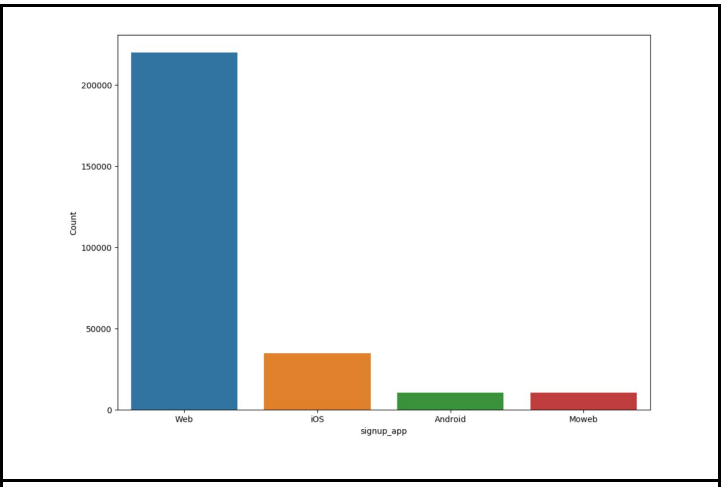


Fig 1.14 Counts of sign_up app

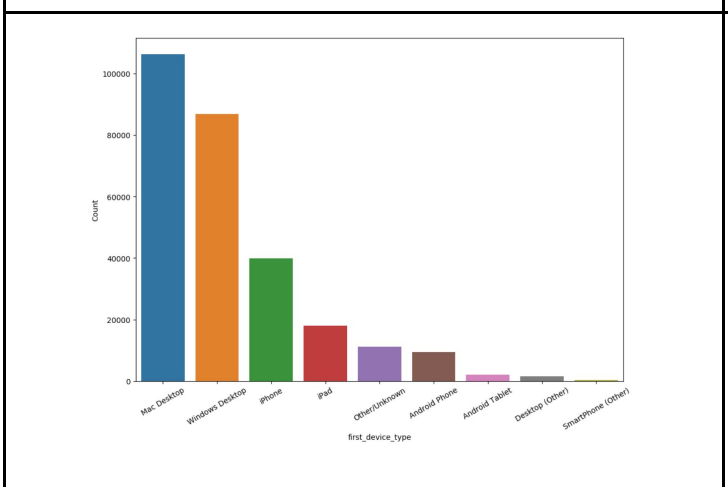


Fig 1.15 First device type

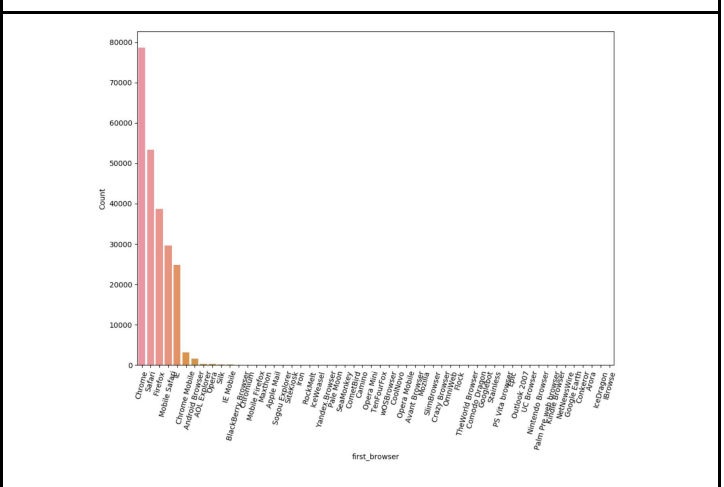


Fig 1.16 First browser used

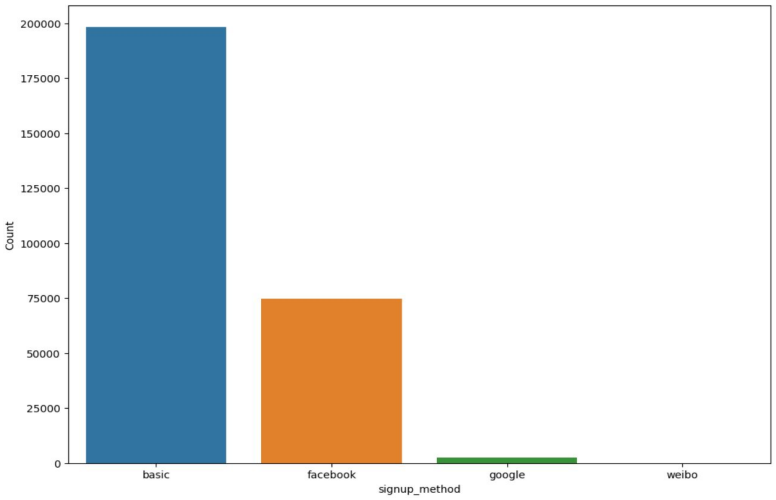


Fig 1.17 Signup Method count

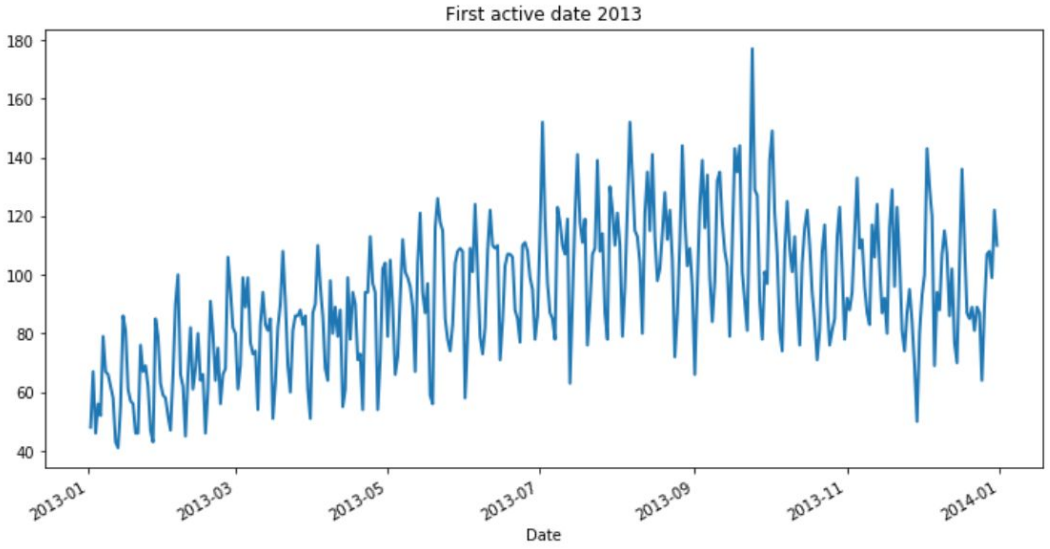


Fig 1.18 First Active date

Activities of users peaked on July, August and October. This may be due to the users planning for the trip in December. On average, a trip is being planned 2-4 months before the trip. The graph also shows that it has the least activity during the month of December, this may be due to December being the holiday season for almost every country, therefore users should not be active in December to plan for a December trip.

4. Preprocessing & Feature Engineering

4.1 Processing Users Dataframe

Processing Dates

Extracting year, month and day from Timestamp_first_active & Date Account Created

First, we needed to convert timestamp_first_active dates into integer format. Before processing, timestamp_first_active dates were being stored as unformatted strings.

Example of unformatted string: 20090319043255

Hence, to extract the dates, we used a lambda function to string slice the unformatted date string and convert it to an integer format. We then made us of the imported datetime library to convert these slices to a date and created new columns for active year, month and day.

```
timestamp = df_all.timestamp_first_active.astype(str).apply(lambda x: datetime(int(x[:4]),
                                                                                int(x[4:6]),
                                                                                int(x[6:8]),
                                                                                int(x[8:10]),
                                                                                int(x[10:12]),
                                                                                int(x[12:])))

df_all['active_year'] = np.array([date.year for date in timestamp])
df_all['active_month'] = np.array([date.month for date in timestamp])
df_all['active_day'] = np.array([date.day for date in timestamp])
```

Fig 2.1: Snippet of Timestamp code

The next date feature to process is date_account_created. Date account created was formatted differently.

Example of Date_account_created: 2010-06-28

Similar to timestamp first active, we extracted the date, year and month and converted them into separate columns.

Engineering time interval feature (days and seconds)

We derived a new feature from the timestamp and date account created columns known as time interval. To do this, we simply subtracted the timestamp_first_ative and date_account_created values and created two new columns, interval_days (time interval in days) and interval_seconds (time interval in seconds).

We find this feature to be useful as it tells us the conversion rate of the user from when he/she first entered the website and registered the account.

Engineering Holidays Feature

Bookings might happen more frequently if there are holidays. Hence, we decided to engineer a feature from the dates to see if the new user has registered on a holiday or not. First, we had to extract a list of holidays from the date range. We chose 2009-2014 as that was the date range given in the dataset.

To get the holidays dates, we used Python's Beautiful Soup library to extract holidays from https://www.timeanddate.com. Then, we stored these list of holidays and created a new column called holiday. Holiday is a binary feature where 1 means that timestamp_first_active is a holiday and 0 means that the date is not a holiday.

```
def holidays_check(year):

    response = requests.get("https://www.timeanddate.com/calendar/custom.html?year="+str(year)+"
    dom = BeautifulSoup(response.content, "html.parser")

    trs = dom.select("table.cht.lpad tr")

    df = pd.DataFrame(columns=["date", "holiday"])
    for tr in trs:
        datestr = tr.select_one("td:nth-of-type(1)").text
        date = datetime.strptime("{} {}".format(year, datestr), '%Y %b %d')
        holiday = tr.select_one("td:nth-of-type(2)").text
        df.loc[len(df)] = {"date" : date, "holiday": 1}
    return df

list_of_hols = []

for year in range(2009, 2014):
    df = holidays_check(year)
    list_of_hols.append(df)
    holidays = pd.concat(list_of_hols)

select_date = list(holidays["date"].astype("str"))
holiday = df_all.timestamp_first_active.apply(lambda x : str(x.date())).isin(select_date)

df_all["holiday"] = holiday #stored as True or False values
df_all["holiday"] = df_all['holiday'].apply(lambda x : 1 if x else 0)
```

Fig 2.2: Engineering Holiday Feature

Is weekend feature

Similarly, weekends might lead to a rise in travel bookings. Hence, we created a new binary feature known as weekend to derive if timestamp is a weekend or not.

```
weekday = df_all.filter(items=['id','timestamp_first_active'])
weekday = pd.to_datetime(weekday["timestamp_first_active"], format="%Y-%m-%d")
weekday = weekday.dt.dayofweek

df_all["weekend"] = weekday.apply(lambda x : 1 if x>=5 else 0)
```

Fig 2.3: Engineering Weekend Feature

Processing Gender

As seen in the EDA (Missing value portion), we have many missing gender values (0.46%). To fill up these values, our team has decided to use the other features to derive the classification for gender. Our goal was to classify gender of each observation into two labels, [“MALE”, “FEMALE”], thereby removing both Nan values and “OTHER” values for gender.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=0)

model_gender = lgb.LGBMClassifier(n_estimators=500, n_jobs=-1, reg_alpha=1).fit(X_train, y_train)
pred_gender = model_gender.predict(X_test)

pred_gender = model_gender.predict(gender_test)
pred_gender = pd.DataFrame(pred_gender)
```

```
#predicted gender
pred_gender = model_gender.predict(gender_test)
pred_gender = pd.DataFrame(pred_gender, columns=['gender'])
pred_gender = pd.concat([pred_gender, gender_test_id], axis=1)

#original gender
origin_gender = y
origin_gender = pd.DataFrame(origin_gender, columns=['gender'])
origin_gender = pd.concat([origin_gender, gender_train_id], axis=1)

#concat original gender and prediction gender
gender = pd.concat([origin_gender, pred_gender], axis=0)
```

Fig 2.4: Snippets of Code used to predict gender

We used the LightGBM method to help us predict the missing gender values.

Processing Age

```
def process_age(users):
    ''' Function takes in the user data frame and cleans ages.'''

    #Assume ages > year 1930 are the year born of people. Derive ages from that year
    users['age'] = users['age'].apply(lambda x: x-2014 if x > 1920 else x)

    #take away age outliers > 10 or < 13
    users['age'] = users['age'].apply(lambda x: np.nan if x > 100 or x < 13 else x)

    return users

df_all = process_age(df_all)
```

```
def get_age(age):
    if age < 0:
        return 'NAN'
    elif (np.logical_and(age<25, age>=15)):
        return 20
    elif (np.logical_and(age<35, age>=25)):
        return 30
    elif (np.logical_and(age<45, age>=35)):
        return 40
    elif (np.logical_and(age<55, age>=45)):
        return 50
    elif (np.logical_and(age<65, age>=55)):
        return 60
    elif (np.logical_and(age<75, age>=65)):
        return 70
    elif (np.logical_and(age<85, age>=75)):
        return 80


age = df_all.age
df_all['age'] = np.array([get_age(x) for x in age])
```

There are extreme outliers for age with values higher than 1900+ and some as low as 1. Firstly, for values higher than 1920, we assumed that these users mistakenly keyed in their birth-year. Hence, we derived their birthdays from values > 1920 by subtracting 2014 from the value. Secondly, we also removed extreme outliers of age > 100 or < 13 (the minimum age to register an account with AirBnb). Next, we binned users into ages of 20, 30, 40, 50, 60, 70 and 80 to help our model process data faster.

4.2 Processing Sessions

Data Aggregation of Session Duration for each user

```
def make_merged_sessions():

    sessions = df_sessions

    #session elapsed group by sum, mean
    ses = sessions.filter(items=('user_id', 'secs_elapsed'))

    print(" Group aggregation in process...")


    ses_groupby_sum = ses.groupby("user_id").agg(np.sum)
    ses_groupby_mean = ses.groupby("user_id").agg(np.mean)
    ses_groupby_median = ses.groupby("user_id").agg(np.median)
    ses_groupby_var = ses.groupby("user_id").agg(np.var)
```

We aggregated the seconds elapsed for each user by grouping by id and derived four aggregated values: sum, median, mean and variance.

Then, we counted the frequencies per action, action_type and action_detail per user and one hot encoded the actions.

Finally, we combined sessions data with user dataframe.

4.3 Processing merged sessions & users dataframe

```
%%time
from sklearn.preprocessing import Imputer
impute_list = df_all.columns.tolist()
imp = Imputer(missing_values='NaN', strategy='median', axis=0)

df_all[impute_list] = imp.fit_transform(df_all[impute_list])

gc.collect()
```

There were still missing values after merging sessions and users. To fill in these missing values, we used the imputer function from sklearn.preprocessing to fill in missing values using the median values. This helps us reduce sparsity of data.

4.4 Summary of Processing and Feature Engineering done

DataFrame	Newly Engineered Features	Engineering Method
Users	Time Interval	Derived from subtracting timestamp_first_active and date_account_created and extracting seconds and days value
Users	Weekend (Binary)	Derived from checking if timestamp_first_active is weekday or not and inferring weekend from that value.
Users	Holiday (Binary)	Use BeautifulSoup to send a request to an online calendar website and proceed to extract web list of holidays (new data from external sources). Then, crossed checked if timestamp falls in the list of holidays or not.
Sessions	Aggregated values of median, mean, sum and variance by user id	Grouped sessions by ID and derived aggregation values from secs_elasped
Sessions	Action, action_type and action_detail frequencies by user id	Grouped by ID and get each frequency

DataFrame	Processing Done	Processing Method
Users	Convert timestamp first active and date account created from an unformatted string to day, month and year columns	Combination of both datetime library and string slicing to extract day, month and year values.
Users	Removing "OTHER" and NaN gender values	Use classification technique to label ("MALE", "FEMALE") observation derived from other features.
Users	Processing and binning age	Removed outliers (age <13 or age >100) Derived age value from users whose age > 1920 Categorized age values to 8 different categories.
Merged DF	Filling NaN values with median value	Used imputer to derive median value for each feature and filled NaN features.

5. Methodology

5.1 Scoring Criterion (NDCG)

We will be utilising **Normalised Discounted Cumulative Gain** (NDCG)[3] at k where k = 5. We are allowed 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

Formula	Example
$DCG_k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{\log_2(i+1)},$ $nDCG_k = \frac{DCG_k}{IDCG_k},$	[FR] gives a $NDCG = \frac{2^1-1}{log_2(1+1)} = 1.0$ [US, FR] gives a $DCG = \frac{2^0-1}{log_2(1+1)} + \frac{2^1-1}{log_2(2+1)} = \frac{1}{1.58496} = 0.6309$ As seen above, when you only predict one country, France, and get the prediction correct, you will get an evaluation score of 1.0. However, if you predict two countries and only get one correct, the evaluation score dips to 0.6309. Hence, it is best if we can make one correct prediction to get the highest score possible.

5.2 Data mining task

Question to answer: Which ensemble method is the most computationally efficient without sacrificing accuracy?

To help with our classification, we used Decision Tree model to label countries new users will travel to.

[3] Rigaldies, B. (2018, February 26). *How to Implement a Normalized Discounted Cumulative Gain (NDCG) Ranking Quality Scorer in Quepid.* Retrieved from OpenSource Connection: https://opensourceconnections.com/blog/2018/02/26/ndcg-scorer-in-quepid/

5.2.1 Justification for selection of Ensemble Techniques/Experimental Setup on Benchmark Data

In our proposal, we considered using two methods to help with our classification task. Method 1 is **Random Forest** and method 2 is **XGBoost,** a gradient boosting framework. Random forest was taught to us in class and we wanted to test a tried and tested method of ensembling. We also chose to explore Gradient boosting techniques as it has had desirable results in many kaggle competitions[4].

During the project, we were introduced to a third ensemble method, **LightGBM**, which proved to be more computationally efficient and more accurate (according to NDCG) than the other two methods when tested on our baseline implementation.

Below are the **baseline (no data preprocessing done)** results for the three ensemble methods:

Ensemble Method	Processing Time (seconds)	NDCG Score (public)
Random Forest	53	0.81764
XGBoost (boost rounds = 100)	390	0.86419
LightGBM (boost rounds = 100)	57	0.86518

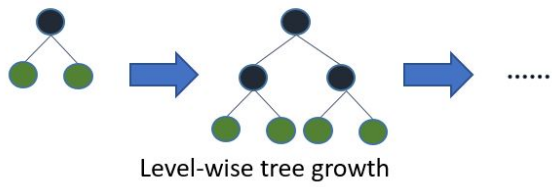
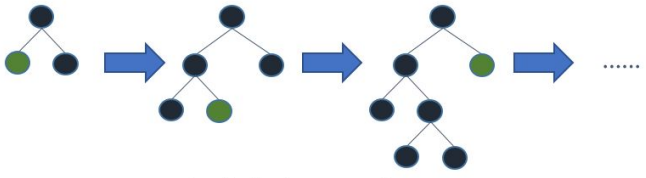
As seen in the table above, LightGBM has the most accurate NDCG score whilst being 600% faster than XGBoost[5] on the baseline implementation. Hence, our team decided to go ahead with the LightGBM ensemble method to do our project.

5.2.2 LightGBM vs XGBoost

Doing further research, we have found sources that supported the fact that LightGBM is alot faster and achieves similar/better accuracy to XGBoost (Khandelwal, 2017, NIPS 2017 & Moreno 2018). This is due to its histogram based algorithm, lower memory usage and better compatibility with large datasets as compared to XGBoost.

Reason for increased accuracy

LightGBM uses a leaf-wise tree growth instead of a level-wise tree growth.

XGBoost	LightGBM
 Level-wise tree growth	 Leaf-wise tree growth

[4]B. G. (2017, January 24). A Kaggle Master Explains Gradient Boosting. Retrieved from http://blog.kaggle.com/2017/01/23/a-kaggle-master-explains-gradient-boosting/

[5] Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In Tiangqi Chen, & C. Guestrin, *KDD '16 Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785-794). New York: ACM.

Leaf-wise growth tends to achieve lower-loss than level wise algorithms and hence can be more accurate. However, one drawback is that it may overfitting when data is small. To overcome this, we can hypertune parameters to prevent over fitting.

Reason for faster training times

A reason for faster training times could be due to the fact that LightGBM supports parallel learning and its histogram based algorithm.

5.3 In-depth view of LightGBM

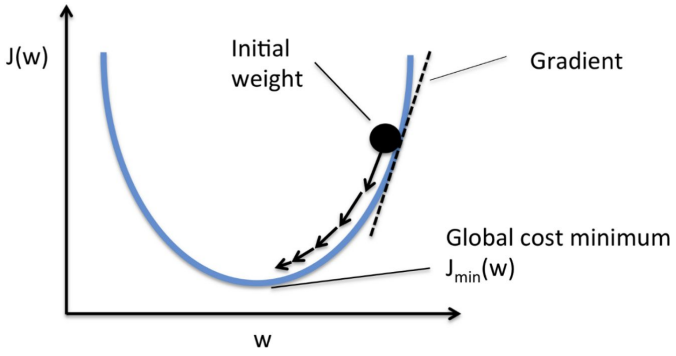
LightGBM is a gradient boosting framework that uses tree based learning algorithms. It has known to have a faster training speed and efficiency due to the fact that it grows trees leaf wise instead of depth wise. It will choose the leaf with the max delta loss to grow. In addition, LightGBM has a reduced communication cost of data parallel, making it more computationally efficient than other **gradient boosting** methods (CatBoost, XGBoost).

Gradient Boosting Explained(Compared to Adaboost)

Like Adaboost, LightGBM algorithm also iteratively creates weak learners to create a strong learner in the end. However, the algorithm differs from Adaboost in how they create the weak learners during the booting process.

Adaboost changes the sample distribution by modifying weights attached to each instance (increase weights for wrongly predicted samples, decrease weights for correctly predicted samples). In contrast, gradient boosting does not modify the **sample weights** but instead calculates the **error rate** (known as pseudo residuals) of the current ensemble and uses this value to improve on the next tree by adding additional learners (h) to the base learner (F0), this process stops when residuals have no more pattern that can be modelled. In other words, gradient boosting aims to minimize the loss of the model through the addition of weak learners using a **gradient descent**.

Gradient Descent: Adjusting weight of predictors in order to minimise error rate.



Gradient Boosting Algorithm

1. Initialize model with a constant value:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma).$$

2. For m = 1 to M:
    1. Compute so-called *pseudo-residuals*:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

    2. Fit a base learner (e.g. tree) $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$.
    3. Compute multiplier γ_m by solving the following one-dimensional optimization problem:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

    4. Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

3. Output $F_M(x)$.

Gradient Boosting methods builds an ensemble of trees one by one and the predictions of the individual trees are summed. After an ensemble is constructed, the algorithm calculates the pseudo-residual (gradient of the mean squared error) based on the discrepancy between the target function and the current ensemble's predicted value. Then, it adds on additional learners to complement the existing trees well by minimizing the training error (pseudo-residual) of the current ensemble (minimizing the residual).

Gradient Boosting Steps Visualized

Step 1: Gradient boosting learns a simple regression predictor (1 layer decision tree) based on the training data (red points) in image L1 below.

Step 2: Next, it tries to correct the errors by first plotting its residuals (green points) as shown in image L2. It learns a new simple regression model (1 layer decision tree) to predict its error residuals.

Step 3: Adding both predictors together, gives a better predictor as shown in Image L3.

As seen in Image L3, there are 2 transition points instead of 1 as compared to the predictor in Image L1.

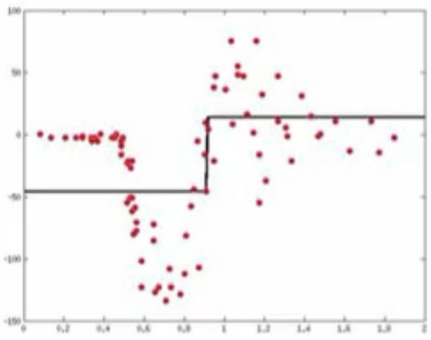


Image L1

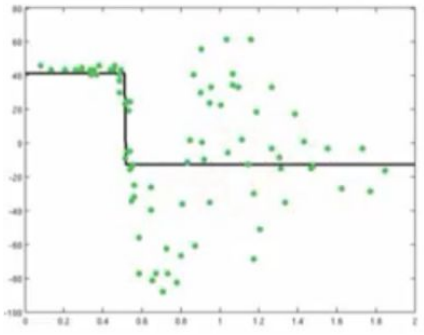


Image L2

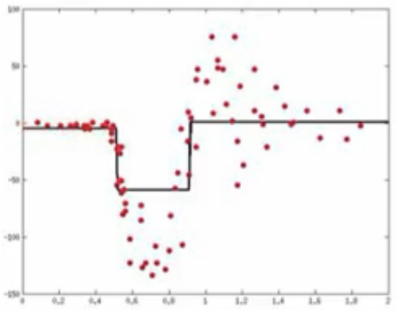


Image L3

Step 4: The residual of this prediction is once again plotted in another residual plot and a new model is fitted onto the new residual plot once again. Repeat steps 1 to 4 until the predictor has enough decision stumps to accurately predict the data points. Image L4 depicts an example of the flow of gradient boosting algorithm, showing how the predictor in becomes more complex as more decision stumps are integrated into it based on its learnings from residual predictors.

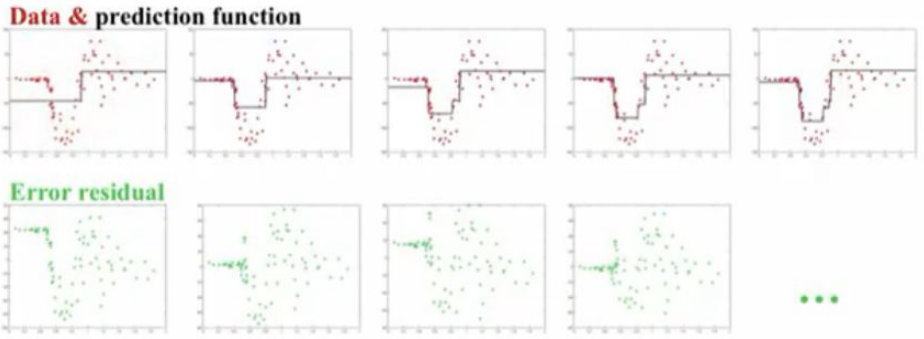


Image L4

5.4 Implementation of Model

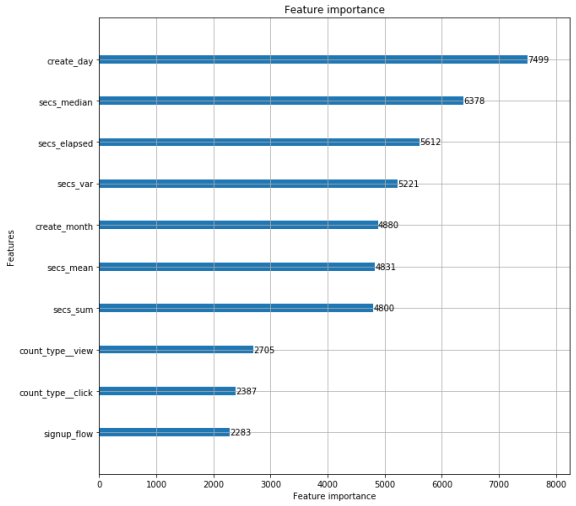
Before running LightGBM, we one hot encoded all categorical variables and formed a total of 959 features.

Step 1: Ensemble model with all features

First, we ran the model with all features selected and with high number of boosting rounds (400) with low learning rate so as to prevent overfitting.

Hyperparameters	Time Taken	NDCG Score
boosting_type= 'gbdt',nthread=3, n_jobs=-1, reg_alpha=1, reg_lambda=0, max_depth=-1, learning_rate=0.05, n_estimators=400, seed = 0	632s	0.88011

Step 2: Feature selection by importance



Example of 10 most important features

We then ran this code to filter our filters that do not meet the threshold of 1 according to sklearn's SelectFromModel threshold value.

```
fs_model1 = SelectFromModel(fs_model, prefit=True, threshold=1)
X_train = fs_model1.transform(df_train)
X_test = fs_model1.transform(df_test)
```

The code above reduced feature dimensionality from 900 to 451 features (almost 50% reduction in dimensionality).

Then the model was ran again, this time with n_estimators to be 100 and learning rate to be 0.1.

Hyperparameters	Time Taken	NDCG Score
boosting_type= 'gbdt',nthread=3, n_jobs=-1, reg_alpha=1, reg_lambda=0, max_depth=-1, learning_rate=0.01, n_estimators=100, seed = 0	151s	0.88039

After feature selection, the runtime of the model decreased by 419% and the score increased by 0.00029. With feature selection, we are able to come up with a less computationally and accurate model.

6. Further Experimentation of Model

6.1 HyperParameter Tuning with Feature Selection model

We conducted a few parameter tuning models to see how it will affect the NDCG Score.

Questions to answer: Will tuning parameters to have more boosting iterations improve accuracy score?

Constants	Tuned Parameters	Time Taken	NDCG Score
boosting_type= 'gbdt',nthread=3, n_jobs=-1, reg_alpha=1, reg_lambda=0, max_depth=-1, learning_rate=0.01, seed = 0	n_estimators= 300	532s	0.87902
	n_estimators = 200	362s	0.87221

With this, we found that the best number of boosting rounds is still the default set number of 100. Higher boosting rounds might cause overfitting of the model to the train dataset and hence resulted in the slightly lower NDCG score. The time taken for an increased number of estimators also resulted in much higher time taken to model the data.

6.2 Component Analysis + Further Selection

We did principal component analysis[6] which will help automatically combine features. This experiment was to see if combined features would result in higher NDCG scores.

Questions to answer: Will combining features to form new ones improve accuracy score and significantly reduce model processing time?

```
from sklearn.decomposition import PCA

pca_trafo_100 = PCA(n_components=100)
pca_data_100 = pca_trafo_100.fit_transform(train)
```

```
pca_trafo_200 = PCA(n_components=200)
pca_data_200 = pca_trafo_200.fit_transform(train)

pca_trafo_300 = PCA(n_components=300)
pca_data_300 = pca_trafo_300.fit_transform(train)
```

The code above ran PCA and selected the top 100,200 and 300 combined features to be used. We then ran the prediction model with the new PCA features and had the following results:

No. of Principal Components	Time Taken	NDCG Score
100	121s	0.86701
200	184s	0.86807
300	301s	0.86855

Since PCA results in drastically lower NDCG scores and almost the same time taken (151s vs 121s) as doing simple feature selection above, we decided to not use PCA to combine features.

6.3 Support Vector Machine + PCA experimentation

Questions to answer: Can PCA be combined with Support Vector Machine Model to produce a more accurate than LightGBM with PCA?

Before PCA, we tried to run SVC model on our dataset but it proved to be too computationally intensive for our computers. There needed to be more reduced dimensionality on the dataset in order for our computers to handle SVC predictions.

[6] Brems, M. (2018, April 18). *Towards Data Science.* Retrieved from A One-Stop Shop for Principal Component Analysis: https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c

Upon further research, we found out that PCA can be combined with support vector machines to make accurate predictions (Widi and Adiwijaya, 2018). We decided to use PCA to reduce dimensionality and run the SVC model. We reduced dimensions to 100 with PCA and ran a linear SVC model with maximum of 100 iterations to produce a submission file.

However, the results were not desirable and we only got a score of 0.66 when we submitted the predictions done with Support Vector Machines and PCA.

This finding might be due to the fact that PCA randomly merges different features and drastically reduced useful features to use for our training model. As we do not know what features have been combined and what features have been removed, it might result in a reduction in the accuracy score of the model.

6.4 Using AdaBoost

Questions to answer: Will the traditional boosting algorithm yield better results?

We also used went back to use Adaboost to see if a traditional boosting algorithm will yield better results. Upon submission, we received an accuracy score of 0.8521 (public). Hence, we deduced that gradient boosting methods are better for this project.

7. Conclusions

7.1 Final Score

We managed to achieve a top 10% placing on the leaderboard and our best score had a result of 0.88039 on the public leaderboard. The top placing had a score of 0.88209. He posted his solution online and we realised that his score of 0.88209 was achieved due to the fact that he had access to high computational power, allowing him to better optimize the boosting algorithm and run more experimentation. Although we were shy of the top score by 0.002, we believe that it was one of the best scores we could come up with given the time and availability of resources that we had.

Submission and Description	Private Score	Public Score	Use for Final Score
submission_f_select.csv 6 hours ago by Austin Woon add submission details	0.88416	0.88039	☐

7.2 Project Findings and Achievements

Key Finding #1: Untapped Potential - Reach out to more age groups

Our EDA shows that most of the users range from age 20 to 40. Airbnb only provides online platform for users to make their bookings. This is in line with the trends that users in this age group tend to be more tech-savvy and hence are more likely to access Airbnb's online booking platform to meet their needs. However, not all elderly have the capability to operate electronic devices which results in market share loss in the older age group segment, Airbnb can explore more ways to reach out to this age group of potential users.

Key Finding #2: Markets to target

US	62095
NDF	62087
other	62085
FR	60031
IT	35388
GB	12567
ES	10157
CA	3917
DE	741
NL	708
PT	380
AU	324

Figure 7.1

In figure 7.1, ,we can see the counts of each country predicted by our model. It is not surprising that US tops the rankings with 62095. Relating back to the business, Airbnb may want to increase marketing and sales efforts in countries like US, Great Britain and others(predominantly Asia) to maximise their sales and profits. On the other hand, Airbnb can also increase efforts in countries that have the least counts as these countries, like Austria and Portugal, are still developing which also means the most untap potential. The can lead to high growth in profit for Airbnb once these countries develop.

Key Finding #3: Tailoring content to day of creation

Looking at Figure A in section 5.4, creation day of new user account is the most important feature in determining the predicted country. Using this knowledge, Airbnb can create different content to present on its application depending on the day of user's account creation. This can improve the user experience on the application as the potential destinations would be presented nicely to the users rather than users needing to go through the application for their first booking.

Limitation #1: Data Quality for target is mainly USA and NDF

Most observations of users' have booking destinations of "USA" and "NDF". This might make it challenging to predict where users from other countries will their first booking to due to cultural differences, preferences or even variables such as average spending power of users from various countries.

Limitation #2: Large dataset prevents us from using other ensemble methods like XGBoost

We used LightGBM as it was computationally light and could give us a high NDCG score whilst using 100% of the dataset. However, our computers could not handle XGBoost (we experimented with XGBoost with 100 boosting rounds and all ran into memory error when using 100% of training data). However, given better computing resources, we could have ran other models with 100% of the training data and this could result in higher NDCG Scores (although it will probably have a higher run time). As such, the size of the data prevented us from using other models to compare with the current model implemented with LightGBM. Being able to utilise 100% of training data will give us a better representation of the user population as well.

Achievement #1: Knowledge on Pre-processing Data

Before this project, we did not know how to process data well. We were only taught to use Enterprise Miner to run models which involved no data pre-processing. However, we took up external courses (DataCamp) and read external sources (books) to learn how to manipulate data frames and clean up data using.

Below are a few data processing techniques we had picked up throughout the course of the project:

1) Using lambda functions to clean up data
2) Pulling data online with BeautifulSoup to get data to help process our users dataframe (Holiday List)
3) Using multiple features to derive a new feature (time interval feature)
4) Using Classification to predict gender with other features in observation
5) Using numpy to do more computationally efficient processing (numpy is implemented in C which is more computationally efficient)

Achievement #2: Knowledge on plotting Data with Python

We were only taught to do plotting on Enterprise Miner during the module. However, we picked up skills online to learn how to plot using python modules such as matplotlib and seaborn.

Achievement #3: Knowledge on using different Ensemble Methods

Throughout this project, we had to use different ensemble methods no available in Enterprise Miner. For example, for boosting algorithms, only Adaboost is available on miner. However, Python has other boosting methods that has proven to be more accurate or computationally efficient (Catboost, XGBoost, LightGBM). Adaboost has a tendency to be easily defeated by noisy data[7]. We explored the use of such Gradient Boosting ensemble methods to run our model and came up with more accurate predictions.

In addition to knowledge of these models, we also learned how to do hyperparameter tuning to potentially make our model more accurate.
Lastly, we also learnt how to do feature selection in order to cut run time of the model. By selecting the nth most important features, we managed to cut runtime of model significantly with a higher accuracy score. This taught us that we do not need to use all features just to make an accurate prediction and that reducing features could actually result in a more accurate and faster model.

[7] SauceCat. (2017, April 30). *Towards Data Science.* Retrieved from Boosting algorithm: AdaBoost: https://towardsdatascience.com/boosting-algorithm-adaboost-b6737a9ee60c

Achievement #4: Doing Principal Component Analysis

We also experimented with principal component analysis in contrast to feature selection to help combine and select n number of features. We did this in hopes to come up with a model that has less run time and could achieve similar/more accurate NDCG score. Even though this experiment failed, it was insightful to learn how to combine multiple features with PCA.

Achievement #5: Learnt how Data Mining could be used to solve real-world problems

Our understanding on data mining has been enhanced and we have learnt how to use data mining techniques to solve real world problems. We could take what we learnt from this project to help predict users preferences for external companies. An example use case is how one of our team members intends to use techniques learnt from this project to predict what users will purchase next in Shopee during his internship stint there.

7.3 Recommendations for Future Improvements

Recommendation #1: Explore more hyperparameters to tune

In this project, we only explored with number of iterations to tune. However, there are many other parameters to tune. As each model took very long to run, we had no time to run models with different tuned hyperparameters. Given more time, we would like to take more time to explore all the hyper parameters and learn how to tune them to fit our data.

In the future, given more advanced hardware, we could also use automated hyper parameter tuning modules such as GridSearch and RandomizedSearch from sklearn. At the time of this project, using such automated modules has proven to be too computationally intensive for our computers to run. However, it would give us much more insightful findings on how to tune our parameters.

Recommendation #2: Explore other models such

To do this project, we mainly looked at using decision tree to do the classification tasks. However, we could explore other models such as Support Vector Machines and compare NDCG Scores between different models.

Referencing online work on Kaggle, participants have found that gradient boosting methods results in the best scores for this kaggle competition. However, if given more time, we would like to explore other methods ourselves and come to this conclusion through our own experimentation.

Recommendation #3: Reducing number of data and increase number of k fold

This project requires a lot of memory to run the full-scale prediction algorithm, one other option we could explore on is to reduce the number of data set by half and increasing the number of k fold for cross validation.

Recommendation #4: Exploring similar datasets from the same industry (hotel/accommodation)

Due to school commitments and time constraint on the project, the team was only able to focus the analysis on the dataset presented in this competition. If given more time, it will be interesting to cross validate the findings from Airbnb's dataset to their competitors in the same industry. For example, we could compare to data from online booking giants Expedia and Booking.com. These companies begun their operations before Airbnb did, therefore we could possibly source more extensive datasets from them for analysis and comparison with Airbnb's dataset. Furthermore, we could potentially derive more insightful booking trends across the years as a supplement to our findings. Hence from the variety of comparisons and parallels we draw from other datasets, we can have more detailed insights on Airbnb dataset, thereby increasing the quality of our predictions.

Recommendation #5: Use Frameworks and Cloud Services to Run Algorithms

Moving forward, we could use external frameworks like Hadoop with Apache Spark and use a rented service space through cloud based services like Amazon Web Services/Google Cloud Platform. These would allow us to conduct more experimentation despite having limited hardware at hand. For example, using these external services might allow us to explore using XGBoost to its maximum potential instead of relying on just results from a computationally friendlier algorithm (LightGBM). It would also allow us to use automated hyperparameter tuning libraries such as SKLearn GridSearch or RandomizedSearch.

References

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q. and Liu, T. (2017).
LightGBM: A Highly Efficient Gradient Boosting Decision Tree. [online] NIPS Proceedings. Available at: https://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf [Accessed 19 Mar. 2019].

Kurita, K. (2018).
LightGBM and XGBoost Explained. [online] Machine Learning Explained. Available at: http://mlexplained.com/2018/01/05/lightgbm-and-xgboost-explained/ [Accessed 26 Mar. 2019].

Khandelwal, P. (2017).
Which algorithm takes the crown: Light GBM vs XGBOOST?. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/ [Accessed 22 Mar. 2019].

Moreno, H. (2018).
Gradient Boosting Decision trees: XGBoost vs LightGBM (and catboost). [online] Medium. Available at: https://medium.com/kaggle-nyc/gradient-boosting-decision-trees-xgboost-vs-lightgbm-and-catboost-72df6979e0bb [Accessed 20 Mar. 2019].

Guolin Ke, Q. M.-Y. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In Q. M.-Y. Guolin Ke, & I. G. Garnett (Ed.), *Advances in Neural Information Processing Systems 30 (NIPS 2017)* (pp. 3146-3154). California: Curran Associates, Inc.

Smith, C. (2019, March 31). *Airbnb Statistics and Facts (2019) | By the Numbers.* Retrieved from DMR: https://expandedramblings.com/index.php/airbnb-statistics/

Rigaldies, B. (2018, February 26). *How to Implement a Normalized Discounted Cumulative Gain (NDCG) Ranking Quality Scorer in Quepid.* Retrieved from OpenSource Connection: https://opensourceconnections.com/blog/2018/02/26/ndcg-scorer-in-quepid/

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In Tiangqi Chen, & C. Guestrin, KDD '16 Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794). New York: ACM.

Brems, M. (2018, April 18). Towards Data Science. Retrieved from A One-Stop Shop for Principal Component Analysis: https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c

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

B. G. (2017, January 24). A Kaggle Master Explains Gradient Boosting. Retrieved from http://blog.kaggle.com/2017/01/23/a-kaggle-master-explains-gradient-boosting/

Appendix

Appendix A: Key Source Code (Pre-processing & LightGBM Model)

Downloadable from:

https://drive.google.com/file/d/10JvQ2NZxCX105QqjOTESeHolYlyhm2jw/view?usp=sharing

Appendix B: Key Source Code (Benchmark Models)

Downloadable from:

https://drive.google.com/file/d/1rpDARFNKjJ4vSakgtc8N5DEUKjeAnLdB/view?usp=sharing

Appendix C: EDA Code

Downloadable from:

https://drive.google.com/file/d/1Kh-XhIfpvSV4ytFfah47ttpxTVJf9ygM/view?usp=sharing

Appendix D: Dataset downloadable from Kaggle

https://www.kaggle.com/c/airbnb-recruiting-new-user-bookings