airbnb – new user bookings

CAPSTONE PROJECT

machine learning nanodegree

UDACITY - MACHINE LEARNING ENGINEER NANODEGREE

2nd August 2016

1. DEFINITION

1.1 Project Overview

This project is a Kaggle competition that was run by Airbnb from Nov 2015 to Feb 2016.

Instead of waking to overlooked "Do not disturb" signs, Airbnb travelers find themselves rising with the birds in a whimsical treehouse, having their morning coffee on the deck of a houseboat, or cooking a shared regional breakfast with their hosts.

New users on Airbnb can book a place to stay in 34,000+ cities across 190+ countries.

By accurately predicting where a new user will book their first travel experience, Airbnb can share more personalized content with their community, decrease the average time to first booking, and better forecast demand.

The aim of this project, is to predict in which country a new user will make his or her first booking.

1.2 Problem Statement

The problem:

In this project, we are given a list of users along with their demographics, web session records, and some summary statistics. We are asked to predict which country a new user's first booking destination will be. All the users in this dataset are from the USA.

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

The training and test sets are split by dates. In the test set, you will predict all the new users with first activities after **7/1/2014.** In the sessions dataset, the data only dates back to 1/1/2014, while the users dataset dates back to 2010.

Intended Solution:

For greater clarity, the intended format of the solution is represented visually with an image below.

id	country
5uwns89zht	NDF
5uwns89zht	US
5uwns89zht	other
5uwns89zht	FR
5uwns89zht	IT
jtl0dijy2j	NDF
jtl0dijy2j	US
jtl0dijy2j	other
jtl0dijy2j	IT
jtl0dijy2j	FR
xx0ulgorjt	NDF
xx0ulgorjt	US
xx0ulgorjt	other
xx0ulgorjt	IT
xx0ulgorjt	FR

As can be seen in the image above, there needs to be two columns in the intended solution. The id is an alphanumeric string which denotes the user id of each relevant user.

The values in the "country" column is what needs to predicted. The options for the values are 'US', 'FR', 'CA', 'GB', 'ES', 'IT', 'PT', 'NL','DE', 'AU', 'NDF' (no destination found), and 'other'.

The destination country predictions (max. of 5 countries) must be ordered such that the most probable destination country goes first.

1.3 Metrics

The evaluation metric for this competition is NDCG (Normalized discounted cumulative gain) @k where k=5. NDCG is calculated as:

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

$$nDCG_k = \frac{DCG_k}{IDCG_k},$$

where:

rel_i is the relevance of the result at position i.
$IDCG_k$ is the maximum possible (ideal) *DCG* for a given set of queries.

All NDCG calculations are relative values on the interval 0.0 to 1.0.

For each new user, you are to make a maximum of 5 predictions on the country of the first booking. The ground truth country is marked with relevance = 1, while the rest have relevance = 0.

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

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

[US, FR] gives a $DCG = \frac{2^0-1}{log_2(1+1)} + \frac{2^1-1}{log_2(2+1)} = \frac{1}{1.58496} = 0.6309$

2. ANALYSIS

2.1 Data Exploration

Prima facie, the important files here appear to be the Training dataset, the Testing dataset and the Sessions dataset.

The training dataset which dates back to 2010 seems to have the activity details of 213451 users. 16 different attributes (features) of these users are being tracked.

Similarly, the testing dataset has activity details of 62096 users. Here, 15 user attributes are being tracked. The missing column here obviously being the country_destination which points to the first country that was booked by a new user.

The sessions dataset, which is a web log of the users' activity containing information such as action and device_type contains 6 columns across 10567737 rows. This is presumably because many rows are being attributed to the same user to track details about multiple actions.

The transpose of the Training data table reveals several things.

	0	1	2	3	4	5	6
id	gxn3p5htnn	820tgsjxq7	4ft3gnwmtx	bjjt8pjhuk	87mebub9p4	osr2jwljor	lsw9q7uk0j
date_account_created	2010-06-28	2011-05-25	2010-09-28	2011-12-05	2010-09-14	2010-01-01	2010-01-02
timestamp_first_active	20090319043255	20090523174809	20090609231247	20091031060129	20091208061105	20100101215619	20100102012558
date_first_booking	NaN	NaN	2010-08-02	2012-09-08	2010-02-18	2010-01-02	2010-01-05
gender	-unknown-	MALE	FEMALE	FEMALE	-unknown-	-unknown-	FEMALE
age	NaN	38	56	42	41	NaN	46
signup_method	facebook	facebook	basic	facebook	basic	basic	basic
signup_flow	0	0	3	0	0	0	0
language	en	en	en	en	en	en	en
affiliate_channel	direct	seo	direct	direct	direct	other	other
affiliate_provider	direct	google	direct	direct	direct	other	craigslist
first_affiliate_tracked	untracked	untracked	untracked	untracked	untracked	omg	untracked
signup_app	Web	Web	Web	Web	Web	Web	Web
first_device_type	Mac Desktop	Mac Desktop	Windows Desktop	Mac Desktop	Mac Desktop	Mac Desktop	Mac Desktop
first_browser	Chrome	Chrome	IE	Firefox	Chrome	Chrome	Safari
country_destination	NDF	NDF	US	other	US	US	US

Firstly, two columns – namely, **date_first_booking** and **age** – have missing values indicated by NaN. On further processing, these values would have to filled or avoided altogether.

Also, the **timestamp_first_active** column looks to have the full timestamp in a number format. For example, in the first column, the timestamp appears as 20090319043255. If it is indeed a timestamp, it should appear as 2009-03-19 04:32:55. This will need to be addressed while preprocessing

Diving deeper into the data, we'll be looking at what is undoubtedly the most important column in the dataset – country_destination.

NDF	58.347349
US	29.222632
other	4.728954
FR	2.353233
IT	1.328174
GB	1.088774
ES	1.053638
CA	0.669006
DE	0.497070
NL	0.356991
AU	0.252517
PT	0.101663

The breakdown of the destination country reveals that out of the 12 options for the destination country, nearly 90 % (87.56 % to be precise) of the users fall into the top 2 options, i.e. they are either yet to make any bookings or they have yet to venture outside their home country (all of the data comes from users in the US) in terms of AirBnB bookings.

Considering the concentration of data in only two categories, if we aren't careful while implementing the model and end up generalizing it too much, the model might be unable to pickup the subtle variations in the data and while pick the above mentioned 2 options every time a prediction is required.

In theory, this would mean we will need to pick that leans towards higher variance like Decision Trees.

The **age** column reveals quite a bit of problems.

A good chunk of the data, 87990 to be precise remains unknown. Unsurprisingly, 30 is the most populous age among AirBnB users, followed closely by 31, 29 and 28.

Also, some users report ages over 100 and under 15 which is highly unlikely. In part, this could simply be due to errors during user input. The other reason to consider would be users intentionally entering wrong ages for privacy reasons.

There is also evidence of some users confusing the age input field with year of birth as multiple instances of ages in the 1900's can be seen. All these will need to be dealt with during data pre processing.

The column **date_first_booking** column is also problematic. It would likely need to removed (more in the Methodology section)

The testing dataset mirrors these problems.

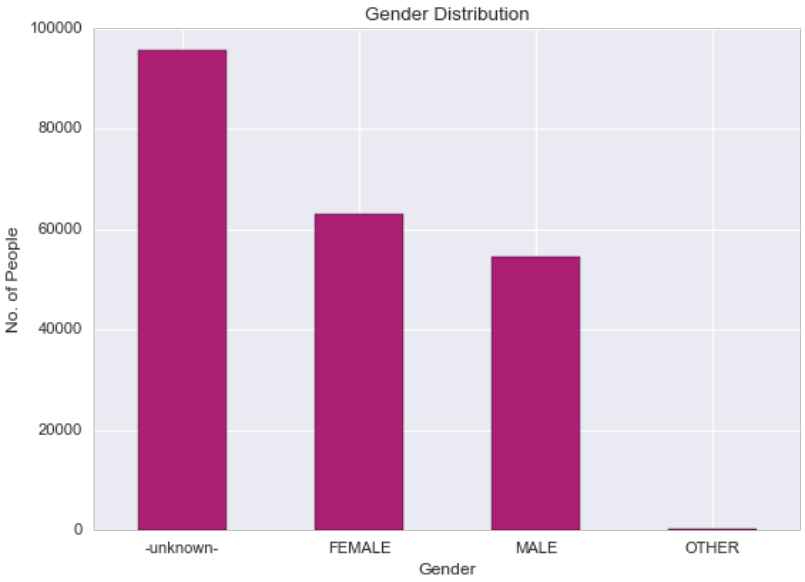
A quick peek into the Sessions dataset reveals what we suspected earlier of many rows being attributed to tracking multiple actions across a single user.

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

2.2 Exploratory Visualization

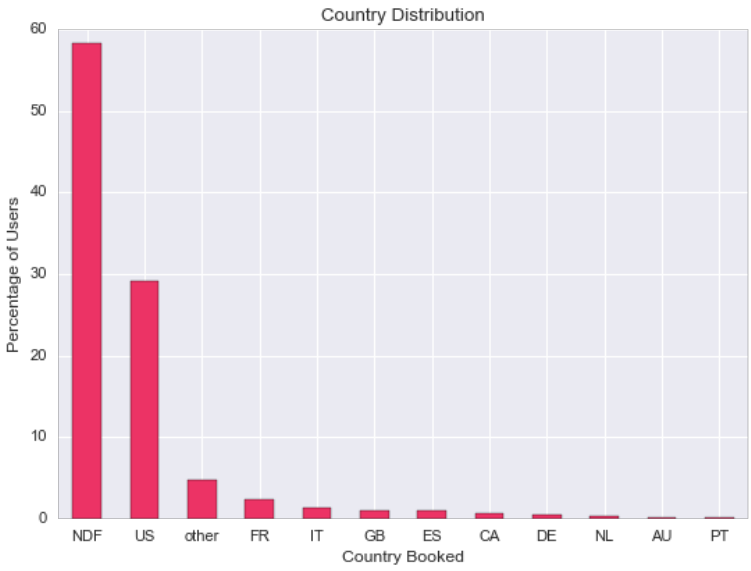
In this section, we will have some plots that summarizes part of the literature above visually.

a. Gender Distribution



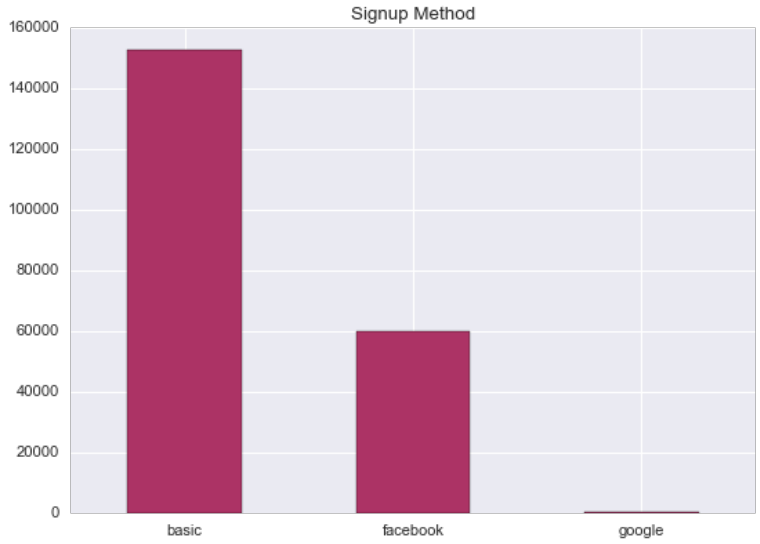
The clear standout point here, is the amount of missing (95688) or “unknown” gender data. Female users (63041) appear to lead Male users (54440) here.

b. Country Distribution



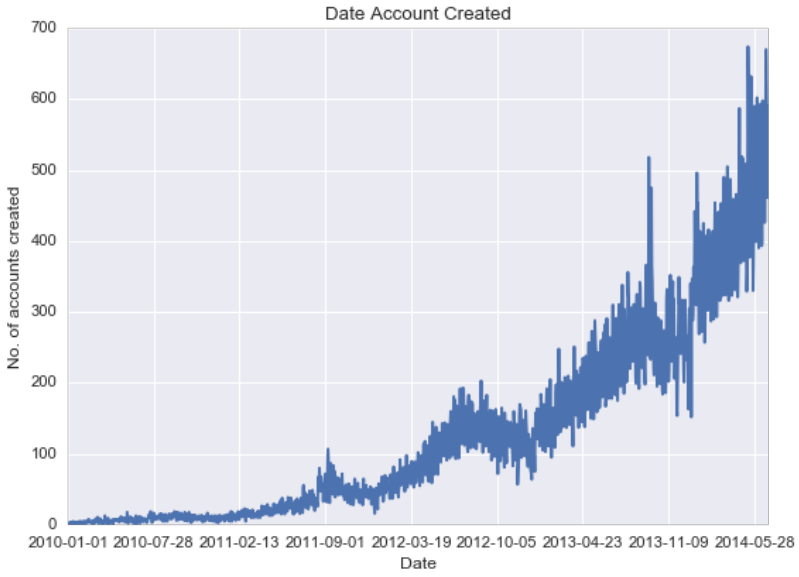
As explained above, ‘NDF’ and ‘US’ dominate proceedings here, amounting to nearly 90 % of the country_booked category.

c. Signup Method



This is the graph which depicts the method users use to sign up to AirBnB. Although the relevance of this graph is questionable, it is interesting to note that a large number of users are unwilling to or have not considered trusting AirBnB with their Facebook and Google login data.

d. Signup Method



This graph clearly shows AirBnB's rapid growth across 4 years. Majority of the accounts created and hence the training data has come in the last two years and particularly from 2014.

2.3 Algorithms and Techniques

The following algorithms and techniques will be used to solve the problems:

a. Firstly, the dataset will need to be cleaned. The format of the date and timestamps should be rewritten into a standard format. The age column needs to be restricted to believable/probable ages and all the outliers to be removed.

b. Then, all the categorical data will be transformed using One Hot Encoding.

c. Subsequently, data/features such as the day, week, quarter and year will be extracted from the date_account_created and timestamp_first_active.

d. After the training and testing datasets are dealt with, we will move our attention towards the sessions dataset. The secs_elapsed column plays a vital role in extracting information about the device usage patterns of each user (the primary and secondary devices, in hindsight)

e. Both the sessions dataset and the train-test data will then be combined, from which the training and testing dataset would be independently outlined for training the model.

f. The chosen classifier would be the XGBoost classifier. This classifier creates multiple, shallow trees iteratively where each tree learns from the previous trees predictions. Boosted decision trees work extremely well for probabilistic classification problems and is very fast compared to something like Gradient Boosting Machine (GBM)

2.4 Benchmark

In the competition, the submissions are evaluated based on the Normalized Discounted Cumulative Gain (NDCG), the formula for which was given above.

The all-NDF benchmark, i.e, the NDCG score you get if the fields to be predicted were filled with NDF, is 0.68411. Needless to say, any reasonable submission should surpass that score.

As we move up in the ranks in the Kaggle competition, the NDCG score, predictably, improves dramatically up to a point. To crack the top 1000 (out of 1462 participants), the NDCG score needs to be 0.86707. The improvement from there on slows to a crawl

The winner of the competition scored 0.88697

The score is obtained on submission of the CSV file to the competitions webpage at: https://www.kaggle.com/c/airbnb-recruiting-new-user-bookings/submissions/attach

3. METHODOLOGY

3.1 Data Preprocessing

This section can be subdivided into two sections: Data Cleaning and Data Transformation

a. Data Cleaning

All the observations of the problems in certain columns will be taken into account and fixed in this section.

Firstly, to make cleaning and processing the data easier, it is best to combine the testing and training dataset.

Next, the formats for the date and timestamps need to be fixed. For this, pandas has a function pd.to_datetime. This is applied to the '**date_account_created**' and '**timestamp_first_active'** columns which transform the date and timestamps to the "year-month-day hour:minute:second" format.

In case the **date_account_created** field is empty, the value in the '**timestamp_first_active**' column is used.

The date_first_booking column will be dropped.

The reason is that this field is only populated for users who have made a booking. In the training dataset, all the users that have a first booking country have a value in the **date_first_booking** column and for those that have not made a booking (**country_destination** = NDF) the value is missing. Also, in the testing dataset, the **date_first_booking** column is empty for all the records. This suggests that the date of the first booking will not be useful in predicting the destination country.

As identified in the Analysis section, there are several incorrect values in the age column. To combat this, we write a function which takes a dataframe, a column, a maximum age value (95) and a minimum age value (15). Everything outside these values will be replaced with NaN. After this, we change all the NaN values to -1.

The other column with missing values is the **first_affiliate_tracked** column. As before, all the missing values are filled with -1.

The output of all these processes will make the combined dataset look as follows:

	0	1	2	3	4	5	6	7	8	9
affiliate_channel	direct	seo	direct	direct	direct	other	other	direct	other	other
affiliate_provider	direct	google	direct	direct	direct	other	craigslist	direct	craigslist	craigslist
age	-1	38	56	42	41	-1	46	47	50	46
country_destination	NDF	NDF	US	other	US	US	US	US	US	US
date_account_created	2010-06-28 00:00:00	2011-05-25 00:00:00	2010-09-28 00:00:00	2011-12-05 00:00:00	2010-09-14 00:00:00	2010-01-01 00:00:00	2010-01-02 00:00:00	2010-01-03 00:00:00	2010-01-04 00:00:00	2010-01-04 00:00:00
first_affiliate_tracked	untracked	untracked	untracked	untracked	untracked	omg	untracked	omg	untracked	omg
first_browser	Chrome	Chrome	IE	Firefox	Chrome	Chrome	Safari	Safari	Safari	Firefox
first_device_type	Mac Desktop	Mac Desktop	Windows Desktop	Mac Desktop	Mac Desktop	Mac Desktop	Mac Desktop	Mac Desktop	Mac Desktop	Mac Desktop
gender	-unknown-	MALE	FEMALE	FEMALE	-unknown-	-unknown-	FEMALE	FEMALE	FEMALE	-unknown-
id	gxn3p5htnn	820tgsjxq7	4ft3gnwmtx	bjjt8pjhuk	87mebub9p4	osr2jwljor	lsw9q7uk0j	0d01nltbrs	a1vcnhxeij	6uh8zyj2gn
language	en	en	en	en	en	en	en	en	en	en
signup_app	Web	Web	Web	Web	Web	Web	Web	Web	Web	Web
signup_flow	0	0	3	0	0	0	0	0	0	0
signup_method	facebook	facebook	basic	facebook	basic	basic	basic	basic	basic	basic
timestamp_first_active	2009-03-19 04:32:55	2009-05-23 17:48:09	2009-06-09 23:12:47	2009-10-31 06:01:29	2009-12-08 06:11:05	2010-01-01 21:56:19	2010-01-02 01:25:58	2010-01-03 19:19:05	2010-01-04 00:42:11	2010-01-04 02:37:58

b. Data Transformation

Data transformation is undertaken with the intention to enhance the ability of the classification algorithm to extract information from the data.

The method we explore to data transformation for the AirBnB data is One Hot Encoding. What this transformation does is take one column with *x* categories and convert it into x columns where each column represents one category in the original column.

For One Hot Encoding, we write a function which takes a dataframe and the columns to convert as the argument.

The columns chosen for OHE are: gender, signup_method, signup_flow, language, affiliate_channel, affiliate_provider, first_affiliate_tracked, signup_app, first_device_type, first_browser.

The train-test dataset looks like the below image after OHE

	0	1	2	3	4	5	6	7	8	9
age	-1	38	56	42	41	-1	46	47	50	46
country_destination	NDF	NDF	US	other	US	US	US	US	US	US
date_account_created	2010-06-28 00:00:00	2011-05-25 00:00:00	2010-09-28 00:00:00	2011-12-05 00:00:00	2010-09-14 00:00:00	2010-01-01 00:00:00	2010-01-02 00:00:00	2010-01-03 00:00:00	2010-01-04 00:00:00	2010-01-04 00:00:00
id	gxn3p5htnn	820tgsjxq7	4ft3gnwmtx	bjjt8pjhuk	87mebub9p4	osr2jwljor	lsw9q7uk0j	0d01nltbrs	a1vcnhxeij	6uh8zyj2gn
timestamp_first_active	2009-03-19 04:32:55	2009-05-23 17:48:09	2009-06-09 23:12:47	2009-10-31 06:01:29	2009-12-08 06:11:05	2010-01-01 21:56:19	2010-01-02 01:25:58	2010-01-03 19:19:05	2010-01-04 00:42:11	2010-01-04 02:37:58
gende_unknown	1	0	0	0	1	1	0	0	0	1
gende_male	0	1	0	0	0	0	0	0	0	0
gende_female	0	0	1	1	0	0	1	1	1	0
gende_other	0	0	0	0	0	0	0	0	0	0
signu_facebook	1	1	0	1	0	0	0	0	0	0
signu_basic	0	0	1	0	1	1	1	1	1	1
signu_google	0	0	0	0	0	0	0	0	0	0
signu_weibo	0	0	0	0	0	0	0	0	0	0
signu_0	1	1	0	1	1	1	1	1	1	1
signu_3	0	0	1	0	0	0	0	0	0	0

After OHE, the original columns are dropped.

Now, we can move on to the Sessions dataset.

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

As we can see, a single user occupies multiple rows with different actions.

We will need to merge this dataset into the combined dataset referred to earlier. For this we will need to condense a single users' information to one row.

Firstly, based on the secs_elapsed column, the users primary and secondary device need to be determined.
The code to aggregate the seconds elapsed and to determine the primary and secondary device is given below:

```
#DETERMINING THE USER'S PRIMARY DEVICE

#filtering the required columns and rows
sessions_device = sessions.loc[:, ['user_id', 'device_type', 'secs_elapsed']]

#aggregating
aggregated_primary = sessions_device.groupby(['user_id', 'device_type'], as_index=False, sort=True).aggregate(np.sum)

#max value of secs_elapsed

idx = aggregated_primary.groupby(['user_id'], sort=False)['secs_elapsed'].transform(max) == aggregated_primary['secs_el

#dataframe for primary device
df_primary = pd.DataFrame(aggregated_primary.loc[idx, ['user_id', 'device_type', 'secs_elapsed']])

#renaming the columns
df_primary.rename(columns = {'device_type':'primary_device', 'secs_elapsed':'secs_on_primary_device'}, inplace=True)

#OHE
df_primary = convert_cols_binary(df_primary, 'primary_device')

#drop the original column
df_primary.drop('primary_device', axis=1, inplace=True)
```

```
# DETERMINING THE SECONDARY DEVICE

remaining = aggregated_primary.drop(aggregated_primary.index[idx])
idx = remaining.groupby(['user_id'], sort=False)['secs_elapsed'].transform(max) == remaining['secs_elapsed']
df_secondary = pd.DataFrame(remaining.loc[idx, ['user_id', 'device_type', 'secs_elapsed']])
df_secondary.rename(columns = {'device_type':'secondary_device', 'secs_elapsed':'secondary_secs'}, inplace=True)
df_secondary = convert_cols_binary(df_secondary, 'secondary_device')
df_secondary.drop('secondary_device', axis=1, inplace=True)
```

The primary and secondary datasets are combined and the combined dataset is then merged with the train-test dataset.

Combining the primary and secondary dataset will require an “outer” join since not very user will have a secondary device.

Combining the Sessions dataset to the Train-Test dataset requires an “inner” join. The reason for this is, we want only the users that have sessions data in our final training dataset.

3.2 Implementation

This section too can be divided into two parts: 'Feature Engineering' and 'Training and Prediction'.

a. Feature Engineering

Like data transformation, the main purpose of feature engineering is to enhance the data in such a way that it increases the likelihood that the classification algorithm will be able to make meaningful predictions.

The two fields that can be used to create some new features are the two date fields – **date_account_created** and **timestamp_first_active**. We want to extract all the information we can out of these two date fields that could potentially differentiate which country someone will make their first booking in.

From **date_account_created** column, we extract the day, week, month, quarter and year the account was created in. From the **timestamp_first_active** column, we extract the hour, day, week, month, quarter and year first active.

From both those columns, we obtain the days to being active after account was created.

As before, the originals columns are then dropped.

b. Training and Prediction

Since we had merged the training and testing dataset earlier, we will need to outline our training dataset. "Inner" joining the country_destination column of the training dataset with the combined dataset will achieve this purpose.

We then define the labels (country_destination) and the data (the earlier dataset with the country_destination column dropped).

The algorithm used to train the model is called XGBoost. In the case of Decision Trees, it uses a technique called “Boosting”, in which it builds trees iteratively such that each tree ‘learns’ from earlier trees.
To do this the algorithm builds a first tree – typically a shallower tree than if you were going to use a one tree approach – and makes predictions using that tree. Then the algorithm finds the records that are misclassified by that tree, and assigns a higher weight of importance to those records than the records that were correctly classified. The algorithm then builds a new tree with these new weightings. This whole process is repeated as many times as specified by the user. Once the specified number of trees have been built, all the trees built during this process are used to classify the records, with a majority rules approach used to determine the final prediction.

Initially, the model was run and fit without any parameters.

To make predictions, as done with the training dataset, the test data was extracted using the code below:

```
#Prediction

df_test.set_index('id', inplace=True)
df_test = pd.merge(df_test.loc[:,['date_first_booking']], df_train_test, how='left', left_index=True, right_index=True,
X_test = df_test.drop('date_first_booking', axis=1, inplace=False)
X_test = X_test.fillna(-1)
id_test = df_test.index.values

y_pred = model.predict_proba(X_test)
```

We have used the **predict_proba** method here. This is done because of the way Kaggle will assess the results for this particular competition. Rather than just assessing one prediction for each user, Kaggle will assess up to 5 predictions for each user. In order to maximize the score, we will use the predicted probabilities that **predict_proba** produces to select the 5 best predictions.

Finally, the code to write these into a CSV file is produced. Submitting this to Kaggle gets an NDCG value of 0.87007.

492	↑62	David Rempel	0.87008	1	Fri, 22 Jan 2016 19:57:03	
493	↑12	PSR ‡	0.87008	3	Tue, 09 Feb 2016 14:15:38 (-22.8h)	
494	↑113	dmi3kno	0.87007	8	Fri, 05 Feb 2016 12:22:20 (-50.1d)	
-		**PratheerthPadman**	**0.87007**	-	**Fri, 29 Jul 2016 08:36:57**	Post-Deadline

Post-Deadline Entry
If you would have submitted this entry during the competition, you would have been around here on the leaderboard.

495	↑46	Anonymous 98782	0.87007	13	Thu, 11 Feb 2016 11:00:00 (-47.9d)
496	↑76	Mr The Dave	0.87007	10	Thu, 11 Feb 2016 05:52:20 (-25.2h)

3.3 Refinement

For refining the model, GridSearchCV was used. The parameters that were experimented with are: max_depth, learning_rate and n_estimators.

The max_depth values were initially set as 2, 3, 4, 5, 6 and 7; The learning rates as 0.1, 0.2, 0.3 and n_estimators as 25,50 and 75.

The result was that it took nearly 6 hours to train the model. On reducing the values of the involved parameters, I was able to bring down the training time to a reasonable ~95 minutes without taking any loss in the results.

The final values of the hyper parameters in GridSearchCV were:

max_depth = 4, 5
learning_rate = 0.1, 0.3
n_estimators = 75, 100

The parameters chosen by the algorithm were:

learning_rate = 0.1
max_depth = 4
n_estimators = 100

On submission of the revised model, the score obtained was 0.87049 which is a marginal improvement over the old score.

377	↑150	kingofhearts ‡	0.87050	34	Wed, 10 Feb 2016 13:07:39 (-5.8d)	
378	↑217	NikSeldon	0.87050	39	Thu, 11 Feb 2016 20:14:19 (-43.8d)	
379	↑224	泽映 彭	0.87049	10	Fri, 15 Jan 2016 03:30:43 (-7.8d)	
380	↑276	Anonymous 99274	0.87049	10	Tue, 02 Feb 2016 12:58:01 (-0.1h)	
-		**PratheerthPadman**	**0.87049**	-	**Fri, 29 Jul 2016 19:34:18**	**Post-Deadline**

Post-Deadline Entry
If you would have submitted this entry during the competition, you would have been around here on the leaderboard.

381	↑523	mquad	0.87047	2	Wed, 09 Dec 2015 23:47:27
382	↑60	karimo ‡	0.87046	10	Thu, 11 Feb 2016 23:10:21 (-24.1h)

4. RESULTS

4.1 Model Evaluation & Validation

XGBoost or “eXtreme Gradient Boosting" was chosen to solve the problem for a variety of reasons. Speed was one of the primary reason. For a large dataset, XGBoost employs parallel processing which makes it faster than other boosting algorithms.

It also bypasses the over fitting in Decision Trees which would otherwise work well for a classification problem.

XGBoost allows user to run a cross-validation at each iteration of the boosting process and thus it is easy to get the exact optimum number of boosting iterations in a single run.

The following parameters were kept constant to obtain the desired results:

a. Objective - This defines the loss function to be minimized
b. Subsample - Denotes the fraction of observations to be randomly samples for each tree. Lower values make the algorithm more conservative and prevents over fitting but too small values might lead to under-fitting
c. colsample_bytree - Denotes the fraction of columns to be randomly samples for each tree.
d. Seed - random seed number used for results reproducibility

The parameters which were tuned are:

a. Max_depth - The maximum depth of the tree.
b. Learning_rate - Step size shrinkage used to prevent over fitting
c. N_estimators - The number of trees to be built.

4.2 Justification

The final model gave an NDCG score of 0.87 which is far superior to the benchmark score of 0.68411 established earlier.

In the final solution, each user is predicted to book a country for the first time, in decreasing order of probability which is precisely the problem the competition proposed to solve.

5. CONCLUSION

5.1 Free-Form Visualization

For this section, rather than intuitively providing a graph about the data, I would like to draw attention to the imperfections in the data even when it is provided by a 'high end' source.

Till date in the MLND, Udacity has provided us with near perfect data which were almost always neatly arranged and ready for a learning algorithm to be run on. For the first time, the student is exposed to real world data and the cleaning and transformation of data to be performed were jarring.

There were, as the images of snippets of the data below would testify, quite a few columns with missing data. Also included were date and timestamps in differing formats. The inherent problem of having free-text fields was brought into light in the age column where on the one hand some users would enter their year of birth instead of their age and on the other, the users either mistakenly or intentionally entered absurd values to depict ages.

date_account_created	2010-06-28	2011-05-25	2010-09-28	2011-12-05	2010-09-14	2010-01-01	2010-01-02
timestamp_first_active	20090319043255	20090523174809	20090609231247	20091031060129	20091208061105	20100101215619	20100102012558

age
38
56
42
41
46
47
50
46
36
47
37

5.2 Reflection

The basis of the project was to predict the most likely countries a new user of AirBnB would book in. While the training data started from 2010, the testing and the sessions dataset only began from 2014.

Even when receiving datasets from (AirBnB) and through (Kaggle) reputable sources, there needs to be a fair amount of cleanup.
Cleaning up the dataset was the top priority in the initial stages. For this purpose, the training and testing data were combined.

Later, to improve the models understanding and ability to predict the outcomes, data transformation and feature engineering of the combined as well as the sessions datasets were done.

The training and testing datasets were extracted from the merged datasets and one of the most popular training algorithms on Kaggle, XGBoost was applied. GridSearchCV was then applied to find the most appropriate hyper-parameters.

A probabilistic prediction was applied to the testing dataset to predict a list of 5 countries for each user according the decreasing order of probability. This was then written to a CSV file which was based on the sample solution given by AirBnB.

The most difficult aspect of the project, for me personally, was deciding what to do with the Sessions dataset. Given the structure of the data and what the secs_elapsed column meant and showed, it seemed logical that the first action should to distinguish between the users' primary and secondary device.

5.3 Improvement

There are two areas to improve that would most certainly improve the NDCG score.

The amount of trees to be built (n_estimators) is capped at 100 at the moment. Earlier, I had capped it at 25, 50 and 75 respectively. Every time the GridSearchCV algorithm had picked out the higher value in search of the best score. It can be assumed that, up to a point, the greater the number of trees, the better will the model be trained. In the interest of keeping the training time under a reasonable amount, the number was capped at 100.

The other area of improvement would be incorporating the remaining data in the Sessions dataset to the training model. The columns that were ignored were the 'action', 'action_type' and 'action_detail' column. These actions could be possibly encoded and incorporated into the final dataset to obtain that would be marginally higher.

References:

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

2. https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/

3. http://xgboost.readthedocs.io/en/latest/model.html