EXAMENSARBETE INOM TEKNIK, GRUNDNIVÅ,
15 HP
STOCKHOLM, SVERIGE 2017

Predicting Airbnb user's desired travel destinations

HUGO ULFSSON

KTH
SKOLAN FÖR DATAVETENSKAP OCH KOMMUNIKATION

EXAMENSARBETE INOM TEKNIK, GRUNDNIVÅ,
15 HP
STOCKHOLM, SVERIGE 2017

Förutsäga nya Airbnb användares första resemål

HUGO ULFSSON

KTH
SKOLAN FÖR DATAVETENSKAP OCH KOMMUNIKATION

Abstract

The purpose of this report is to go through how to predict user's intentions with machine learning and describe our thought process when working on the different stages of solving this type of problem. This will be done by solving Airbnb's Kaggle problem where they wanted Kaggle users to predict where their users were most likely going to travel to based on data from their website. We go through the different choices we made while cleaning and prepairing the provided datasets and the reasoning behind these. The prediction is made using XGBoost and its boosted decision tree algorithm with several different approaches to how we go about preparing the data for training. Finally we upload the results for validation on the Kaggle challenge page and we discuss the strengths and weaknesses behind every approach and discuss what more could have been done to further improve the result.

Abstract

Syftet med denna rapport är att gå igenom hur man kan förutsäga användares avsikter med maskininlärning och beskriva vår tankeprocess när man arbetar på de olika stadier som krävs för att lösa denna typ av problem. Detta kommer att göras genom att lösa Airbnbs Kaggle-problem där de ville att Kaggle-användare skulle förutsäga var deras nya användare troligtvis skulle resa till baserat på data från deras hemsida. Vi går igenom de olika val som vi gjorde när vi rengjorde och förberedde de tillhandahållna dataseten och resonemanget bakom dessa. Förutsägelsen görs med hjälp av XGBoost och dess boosted decision tree algorithm med flera olika tillvägagångssätt för hur vi arbetade med att förbereda data för träning. Slutligen laddar vi upp resultaten för validering på Kaggles utmaningssida och vi diskuterar styrkor och svagheter bakom varje tillvägagångssätt samt diskuterar vad mer kunde ha gjorts för att ytterligare förbättra resultatet.

1 CONTENTS

PREDICTING AIRBNB USER'S DESIRED TRAVEL DESTINATIONS

Machine learning applications today are largely used by companies like Google to enhance the user experience when using their products, for example in the process of identifying and classifying emails as spam. Another application is making sure that advertisements reach users who are most likely interested in what is being advertised. If we have enough data we can make very precise predictions. Many companies today are using machine learning to get to know their users on a deeper level by analysing data about them coming from different sources. There is a lot of data being collected every time we visit a website and different companies are joining forces on the collection of data which increases the amount of data that companies have available for user profiling (Webb & Micheal J Pazzani, 2001). This can be done by for example interaction between Facebook or google and the webpage we are visiting.

1.1 MACHINE LEARNING

Machine learning is the ability to have computers "learn" from data and make predictions from what it learnt, it is what we get when we combine the practice of pattern recognition in computers with artificial intelligence. We must usually go through two basic stages during a normal machine learning process, the first stage is where we have the computer learn from data and the second is where the computer can start to make predictions from new data. Several different techniques can be used during the learning part: Supervised, Unsupervised and Reinforced.

- **Supervised** is where we give the computer a set of data with every correct outcome listed and we have the computer find patterns to why the data produces a certain outcome, every outcome is known beforehand.
- **Unsupervised** is where we give the computer some data but no correct solution is presented to it and it's up to the computer to generate a solution by for example splitting the data into groups based on similar properties.
- **Reinforced learning** is when the computer learns by trial and error, every solution the computer presents is tested and awarded with positive or negative feedback which the computer then uses to further improve on its solution.

1.2 PROBLEM

The problem that this report is supposed to tackle is how we create a model for predicting user intentions based off of user data such as the personal information they entered themselves when creating an account or data collected by the site on what users did during their time on the site, this question will be answered through solving an actual problem provided by the Kaggle website, namely Airbnb's new user booking challenge. The challenge is to predict and rank the 5 most likely travel destinations of every user, there are 5 datasets of varying importance tied to this problem. When a prediction is done of all users it is uploaded to the Kaggle website which calculates its correctness with a NDCG scoring algorithm.

1.3 Existing research

The amount of research for a subject like this which hasn't seen commercial use until just recently is staggering, machine learning have been a subject since mid-1900's and have seen almost constant development since[1]. The research we are going to focus on and use during this report are mainly targeted at the basics of machine learning and applying it while solving a user intention prediction problem. Literature chosen concerning the basics of machine learning includes "Machine Learning for User Modelling" (Webb & Micheal J Pazzani, 2001), "User Intentions Modelling in Web Applications Using Data Mining" (Chen, o.a., 2002), "The Changing Science of Machine Learning" (Langley, 2011) and "C4.5: Programs for Machine Learning" (Quinlan, 1993). When we dove deeper into what actual algorithm we were going to use we found some interesting literature about this as well: "Survey of Multi-Class Classification Methods" (Aly, 2005), "Supervised Machine Learning: A Review of Classification Techniques" (Kotsiantis, 2007) and "XGBoost: A Scaleable Tree Boosting System" (Chen & Guestrin, 2016). We also looked into different methods for validating our model: "A Survey of Cross-Validation Procedures" (Arlot & Celisse, 2010) and a report covering the topic of NDCG scoring for measuring the quality of our prediction (Ravikumar, Tewari, & Yang).

2 Method

When we first start working on a problem like this we have to make several decisions based on what we want to accomplish. What programming language are we going to use, which machine learning approach is best suited for the problem and what algorithm is likely going to be most accurate. Personally we chose to do this by using python as our programming language because it had good support for our dataset format and a decent amount of libraries aimed at machine learning applications, other very viable languages that we considered were R and Matlab. Our machine learning approach will be supervised machine learning because the dataset are designed for this method since we have access to a training set with the correct outcomes and a test set without outcomes. The algorithm that we are going to use have to be fast and not use up a lot of memory since we have a very large amount of data to process. We also have to consider the fact that we want multi classification which means that we want to basically find out how probable it is that the user want to travel to all destinations and not just which destination have the highest probability. A good algorithm for this is gradient boosted decision trees which uses several smaller decision trees with scored outcomes and each outcome gets a different score when ran through the tree. The specific algorithm chosen is XGBoost which stands for extreme gradient boost, this is chosen over other implementations of boosted trees because of its superiority in speed (Kotsiantis, 2007).

---

[1] Machine learning saw a dip in popularity during the AI-winter because the subject got over hyped during early 1980's and rapidly lost popularity until mid-1990 (Langley, 2011)

2.1 SUPERVISED MACHINE LEARNING

As explained above, supervised machine learning is a technique where the computer uses test data with known outcomes to build a model which it can apply to data where the outcome is unknown to predict the outcome of this data. The model produced by this learning process can be split into two different categories: Classification and Regression. A classification type model is a model that aims to assign a label to a set of data like predicting if an email is spam or not. Regression models are used to predict continuous measurements, this may be used to predict how the weather will change based on current weather data.

The workflow when working with supervised machine learning is as follows (Kotsiantis, 2007):

- Data Analysis: The data must be analysed to help the computer understand it as best as possible, during this part we should format the data into a more understandable format and remove unnecessary data entries.
- Algorithm decision: What algorithm we decide to work with depends on what our system limitations are, some algorithms use a lot less memory but may take longer to produce a model than others, some algorithms produce better models for predictions but their reasoning may be too hard to understand which makes parameter tuning harder.
- Model Validation: When the chosen algorithm have produced a model we must validate the model's accuracy, there are several approaches to this and one of them are cross validation. Cross validation is a method where we split the training set into N number of folds, we then remove the outcomes of one of the folds and use this as a test set and train a model on the rest of the data. This process is then repeated for each fold to get a general score of how the model performed (Arlot & Celisse, 2010).
- Parameter tuning: If the model's accuracy was deemed too low we adjust the algorithms parameters and use these new parameters to train another model, this can be done by for example changing the tree size if we chose to work with the decision tree algorithm. We chose to take a more automated approach to this and use grid search when constructing models, we load the grid search with a set of parameters in different ranges and the grid search checks which combination of parameters produce the most accurate result.
- Applying the model: Once we have an accurate model we can now use it to predict the outcomes of new data.

2.2 XGBOOST ALGORITHM

The algorithm we choose to use was created by Tianqi Chen from University of Washington and is an implementation of gradient boosted decision trees. A standard decision tree algorithm will create one tree based on all features to try and predict the outcome, it achieves this by continuously splitting the data into 2 or more groups and leading those groups into more splits on each branch, for example it might split gender into male, female or unknown as the first node in the tree each decision will branch out to new decision nodes or a leaf node which is the final destination for the users in this case, which leaf they end up in will tell the classifier which destination they are most likely going to choose. The downside of normal decision trees is the size required to explain the data fully when for example every gender needs its own decision node describing its age to fully be able to profile the users.

A boosted tree uses the same concept of a tree with decision nodes and final leaf nodes but the difference is the size, a boosted tree uses smaller trees that only explain a fraction of the features in each tree. Several of these trees are then created and every leaf node contains a score, the score from all sub trees are summed up to achieve the final score. The gradient part of the algorithm comes from the creation of every new tree, when the algorithm creates a new tree it analyses the previous trees and creates a new tree that attempts to fix errors in the previous ones.

Datasets today become larger and larger which pose a problem for the existing hardware where for instance the memory is very limited which requires good memory handling from the machine learning algorithms to not suffer too much from common out of memory issues. XGBoost uses several different techniques to tackle these problems, most notably Cache aware access and out-of-block computation, cache aware access means that the algorithm plans ahead to actively reduce the amount of cache misses when new data needs to be constantly fetched from memory and out-of-block computation enables the computer to store parts of the data on disk instead of keeping everything in the main memory, this is supported by efficient use of threads for preloading compressed data from the disk that it partly decompresses when needed. XGBoost automatically makes use of all available cores for computing (Chen & Guestrin, 2016).

2.3 ANALYSING THE DATA

There are 5 datasets supplied by Airbnb for the competition to help us with the prediction (see Appendix A for full reference of the dataset formats):

- Train_users_2: This dataset contains data on 213.451 users from Airbnb, all data is anonymized meaning that all data used for identification is replaced by an ID field. The different data points in this field contains data gathered from the first recorded visit of the user and the data entered during his/her signup process. For the training set we also have data on which country the user first chose to travel to.
- Test users: Test users contains 62.096 users on which the prediction is supposed to be made, it uses the same format as Train_users with the exception of the date_first_booking and country_destination fields.
- Sessions: Contains records of user's actions on the website, each user is identified with a user_id field which corresponds to the id field in the user datasets. Each record in this dataset contains data on what action was performed with what device type and the time in seconds since the last reported action. Only contains data on users from 2014.
- Age_gender_bkts: This dataset groups users in the training set into age groups of 5 years difference and shows information about each group's gender and decided country destination.
- Countries: A summary of the different country destinations and various data on their location and which language is primarily spoken there.

2.4 CLEANING THE DATASETS

Before we proceed with cleaning it is important to really understand what each field tells us and how it is related to the outcome. In the following section we will list the different fields with the highest importance and what tactics we can use to make them more comprehensible for the computer. In this report we will do this process by analysing the real world relation between what these features tell us and how they relate to the outcome, another approach to this is having the computer analyse the data to find which features have the highest correlation to the outcome (Blum & Langley, 1997).

2.4.1 The date fields

In the user datasets we have 3 different fields containing data on different dates: date_account_created, timestamp_first_active and date_first_booking. The date_first_booking field only carries a value for users who booked a travel which makes it useless for prediction since none of the test users will have a value for this, we will simply drop this field because it cannot possible do anything but confuse the computer. The other date fields holds data on when the user first performed any action on the site and when the user chose to sign up with an account. These fields can tell us a lot about the outcome since travel destination are often chosen based on season.

Before we clean these fields we must change the format to a format that we can more easily modify. This data is stored as strings but we want to convert them to date objects which are easily modifiable, this is effectively done by a `to_datetime()` function in the Pandas[2] library which takes a set of columns and converts the data in these into `datetime` objects following a specified format.

We must split the dates into smaller parts and encode them using the one-hot technique for the computer to be able to see relevant patterns in the data, we do this since the XGBoost algorithm is based solely on numerical comparisons which means that it takes user data and checks whether or not the users data is higher than a certain number and use this for the classification. We must also take into account that the seasons cycle throughout the year which is why we chose one-hot encoding[3] for the dates since we do not want the computer to discover a pattern in the size of the month. The size of the year is still relevant since popularity of some travel destinations change over time which means we want to store this number as it is. The smaller parts of the dates can be ignored, this is because they cycle too much to have anything to do with the final outcome and removing these decreases the amount of features used in the training model.

2.4.2 The age field

Users' ages control the outcomes a lot and are therefore a very important field to clean properly. At first glance we can see that most users does not have an age and there is a lot of users with an age over 1000

[2] Pandas library was our library of choice for manipulating the datasets, it does this in a similar way how databases work with data organized in tables.

[3] One-hot encoding is an encoding technique were we split a feature into several features, one for each possible value of that feature and use 1's and 0's to indicate which value every data point have of that feature. This is an excellent way of representing features with a low range of possible values that's originally not stored as numerals.

which we must figure out a way to handle. We can split the users with unbelievable ages into two groups that we will handle differently, the users whose ages we can correct and the ones we cannot. Some users have ages of for example 1956 which is probably people who entered their birth year as their age, we could subtract this from the year of those users first activity to calculate what their age most likely is. There is also some people with ages like 2012 or 2014 which is people who most likely entered the current year as age, there is nothing we can do about this except remove their age and place them in the group of people without an age. We can assume that the vast majority of users will be between age 15 and 95 and people who have entered ages outside of this span probably lied about their age for one reason or another. Since the outcome is not linearly related to users' age we should store the ages using one-hot encoding however this pose a problem with the amount of added features that's largely redundant. To solve this we will bucket the ages which means that we group the ages in smaller groups, a group size of 5 years seems reasonable here but we could also have group sizes of varying length. We place the users who lied about their age in the lowest tier group starting at 5-10 years old and the users without any age data in the next group from 10-15, the users with legit ages are then placed in their corresponding group.

All other fields in the user datasets can be cleaned pretty effectively by representing them using one-hot encoding since they are all represented by strings, the importance of these will have to be decided by the training algorithm when it assigns weights to the features.

Since the XGBoost algorithm is fairly adept at noticing which fields are important and which are not we can keep these features just in case they hold some relation to the outcome.

The next problem we must tackle is what do we do with the session dataset, it contains by far the most data out of all the interesting datasets however it only contains data on users active during 2014 which is only about half of all users and this includes all test users which makes the remaining training users only around 30% of the original set of training users. This pose a big problem for training the model, we will go more into detail on this later.

The sessions dataset is structured differently than the user datasets since there are multiple rows for each user instead of just 1, we must condense these into just 1 row per user so we can join it with the user dataset. We must therefore find a good way to compress up to hundreds of rows into a single row without dropping too much valuable information.

2.4.3 Action fields

There are 3 fields in the session dataset that have to do with the users' actions: action, action_type and action_detail. Since these fields hold values as Strings we should store them using one-hot encoding, the amount of different actions would require a lot of added columns which is not necessary. Handling all 3 features separately would add 475 new features when encoding them and with some research we find that there is only 375 unique combinations of actions in the session's dataset. We can also see that some combinations are used more frequently than others and that some are rarely used at all. If we put all the action combinations least used into a group of combinations we can greatly reduce the amount of added features during one-hot encoding. By choosing to group every action used less than .01% of the time we reduce the total amount of needed features from 375 to 88. Pure one-hot encoding is not very usable in this case since one user could've performed the same action more than once and we would only be able to tell whether or not a user performed an action at least once. A better solution to this would be to count all actions a user performed and store that with the actions instead. Finally we normalize the counts to get a number between 0 and 1 which represent how many times a user performed an action in relation to the total amount of actions performed by that user.

2.4.4 Device field

The attribute we want to bring out of this feature is how much each user used each device, this field can tell us a lot about the user since what devices we use today are largely personal preference just like where we would like spend our vacation. To bring out which devices the user prefers we will group the devices in the same way we solved the actions except instead of counts we will make use of the seconds_elapsed field to measure how much time each user spent using the different devices. The seconds_elapsed field is fairly inconsistent with a lot of outliers because of user inactivity or other reasons, this makes the device usage times not very accurate, one solution to this is that we set a max for how long a user could have used a device between actions, this max can be found by statistically analyzing the distribution of the seconds_elapsed field and limiting the data to a certain number of standard deviations from the mean.

After normalizing these fields we now have a sessions dataset containing one row per user since the start of 2014 and it is now ready to join with the user datasets.

2.5 THE PROBLEM WITH THE SESSIONS DATASET RESTRICTION

As we briefly wrote about earlier the sessions dataset only contains users with an activity after 2014 which only covers 135.483 users, that is only half of the total amount of users however every user in the test set exist in the sessions which makes the session coverage only cripple the training set. What this means is that if we want to use the sessions dataset we either have to discard about 130.000 users or come up with some way to use those users even though they have no records in the sessions dataset. There are 3 different solutions to this problem, each having their pros and cons:

- Ignoring the session's dataset, this approach makes use of the entire user dataset which in turn means that we will have users with overlapping account creation dates of the ones in the test set, this may provide a more accurate prediction because the distribution of travel destinations will take into account the late summer months. With this method we will lose all session data which contains most of the data.
- Ignoring earlier users, if we were to choose this approach we would have a lot more data on the users but we drop more than 130.000 users. The loss of that amount of training points might seem like the biggest problem with this approach but it is not, the biggest problem is that the only users left in the training set are users created during the winter to early summer months and the users we want to predict only contains users from late summer months, realistically these 2 groups should have a different distribution of travel destinations since the attractiveness of some destination largely depend on the seasons.
- Making use of both, we could outer-join the session's dataset with the users set and fill in every missing value with 0, this would indicate that those users used no devices and took no actions. The result of this would be that we now have the full user set to train the classifier but the classifier will put less weight on the data from the sessions dataset since most users will have identical data in those columns.

2.6 THE REMAINING DATASETS

There are some interesting features in both of the remaining datasets that can be used to achieve a more accurate prediction if joined with the other datasets. These sets have been constructed based on outside data and data taken from the original dataset.

2.6.1 Age-gender bkts

The age-gender bkts is a dataset completely constructed from the data in the train_users dataset which means that it is not bringing any new data into the training data, we could join this data with our data since we use the same bucketing ranges on the age field and give the classifier more precise data on these groups.

2.6.2 Countries

In countries we find some interesting data on languages spoken in the different destinations and since we know the language preferred by each user we could use this data to more accurately predict their desired destination based on for example age and levensthein distance. For example older people might more often choose to travel to a country where they speak a language closer to their own. Another interesting field in this set is the latitude position of each country since we can through this determine which hemisphere the country is located on and at what rate this country is affected by seasons, for example a country close to equator will not change much during the seasons but the closer to the poles a country is the more drastically it will experience changes in climate during the different seasons. Since we have no way of knowing the users origin country we cannot accurately use distance between countries as a predictor, we cannot use their language to derive their location either since the different languages are spoken all around the globe.

2.7 RUNNING THE MACHINE LEARNING CODE

Before running the dataset through the machine learning module several preparations have to be made. To start off we will split the user dataset that at this moment contains both the training users and the users who are going to the used for the prediction. Extracting the training set is rather easy because we can just drop every row which have no value for country_destination.

The test set on the other hand is trickier but we solve it by extracting a list of user ids from the original test user set and performing an inner join of this set and the total set which will naturally drop all rows with user ids not in the test-user id group.

This is not the only thing we have to do concerning the datasets, we must also extract which features we want to predict and which features we want to base our prediction on. The first step is to convert the id row into an index which will make the module ignore this when training the model. Since the model works better with numeral values instead of strings we must encode the country_destination to be represented by a number instead of a string. When working with machine learning algorithms it is normal to call the features we want to predict for y and the features used for prediction X.

At this point we can start preparing the model and as we earlier stated we are going to use XGBoost and its boosted trees algorithm, this model will create several trees and try to improve the accuracy of these by cross validation. After cross validation the training process runs again and modifies the parameters according to a predefined parameter set, the most notable parameters being size, depth, number of trees and learning rate which means how much the module is allowed to change the weights after each iteration of trees. After each combination of parameters have been tested the model with the best accuracy is chosen as the final model. This model is then serialized and saved to file along with the encoder used for encoding the country_destination field.

2.8 Using the model to make a prediction

The process of making a prediction once we have the model is very simple, the only preparation we have to do is configure the test-dataset the same way we did the training one. The prediction will produce a multi-dimensional list where every row corresponds to a user on the same index in the test-dataset and every column corresponds to a destination and the values are the probability that the user would pick that destination.

The competition required a certain format on the submission where we had to list every user and the five most likely destinations which then after uploading were evaluated using their NDCG system.

2.9 Algorithm used during evaluation

To evaluate our solution we are using Kaggles submission system. The system gives us a score between 0-1 where 1 is perfect solution. Kaggle uses NDCG (Normalized Discounted Cumulative Gain) which is a system for measuring the quality of rankings.

In the Airbnb challenge we have focused on we were predicting 5 different ranked choices for each user in the dataset. The score of the predictions are calculated from the position of the correct prediction. Each user prediction gets a normalized score between 0 and 1 and the total score is then calculated from the average of all user prediction scores. The score is only calculated from the position of the correct prediction and any predictions with lower probability will not reduce the score of the user prediction.

Formula for calculating NDCG = $\sum_1^i \frac{2^{rel}-1}{log_2(i+1)}$ where k is the number of predictions and rel is either 0 or 1, its 1 for the correct prediction and 0 otherwise. For example if the correct destination is predicted as the most probable destination then we would get a score of 1 and if the correct destination was not part of the predicted destinations we get a score of 0, if the correct destination was predicted as the 3rd most likely destination we would get a score of $\frac{2^1-1}{log2(4)} = 0{,}5$ (Ravikumar, Tewari, & Yang).

3 RESULTS

In this section we will provide results based on the Kaggle ranking and score of different attempts, we will also describe what decisions were made during cleaning and if there is any parameters that differ from the standard set. The code used during each attempt can found in the public git repository https://github.com/ulfsson1992/Kaggle-Airbnb-Challenge.

3.1.1 First attempt – sessions

submission.csv 0.87703
17 hours ago by HugoUlfsson

Parameters used during grid search process:

Parameter	Value
Max_depth	[3, 4, 5]
Learning_rate	[0.1, 0.3]
N_ estimators	[25, 50]

With a score of 0.877 and this being my first try also makes this my best attempt, on the leader board we would place around #300 where #1 have a score of 0.882. The parameters with the best score was a max depth of 5, 0,1 learning rate and 50 estimators. Training the classifier and cross-validating every combination of parameters took 29 minutes. The dataset contained a total of 222 features where 77 came from the session's dataset.

For this attempt we decided to only use users who had a record in the session's dataset which limited us to users created during 2014. This choice reduced the amount of total users from 275547 down to 135911. There was several things we did not like about this approach:

The resulting training set was now limited to users created in January to June and the test set where all created between July and October. This is bad because travel destinations largely depend on what time of year it is, you generally do not want to travel to places like Sweden during January to May because of the dull weather during winter. Since we now have a training set based during winter to early summer they will probably have a different distribution of travel destinations than the users we want to use during prediction.

Since all users were created during 2014 we can ignore the year part of the date columns. The month part is also not usable since no users in the training set have the same month values as users in the test set, this coupled with the fact that we also made the choice that any smaller date parts than months are unusable due to their high cycle rate we could entirely remove the date features. This did not seem like a good approach since we feel like the date column influence the outcome more than any others but at the time we could not think of any better solutions since we also wanted to use the entire session dataset.

3.1.2 Second attempt – users

submission.csv 0.86938
a few seconds ago by HugoUlfsson
add submission details

This attempt scored considerably lower while taking twice the time to train the model used which means that the amount of data points have a larger impact on the time than the amount of features used. Same set of parameters where used for this attempt. The dataset contained a total of 202 features.

The difference between this attempt and last attempt was that this time we did not make use of the session's dataset and instead used the entire training set of users. When using all of the users we had date time stamps that overlapped the test set. Since we used users with a wider range of values on the year and month feature we could now make use of those features to improve on the prediction.

3.1.3 Third attempt – combined all users and session data

submission.csv 0.87680
7 days ago by HugoUlfsson
Session and user data

The score of this attempt was only slightly lower than the first attempt and since this model used by far the most data points combined with the highest number of features it took 133 minutes to train the model. We used the same setup of parameters for this attempt as were used in the previous ones.

In this attempt we outer-joined the session's data with the users which means that we got a complete set of users where the ones who were missing data from the sessions were still present. The missing data got replaced by 0's which for the computer meant that they took no actions and used no devices.

The dataset contained a total of 275 different features where 77 came from the session's dataset.

4 DISCUSSION

In this section we will discuss the results and problems with the process we used to achieve these. We also write about what could have been done to further improve the prediction.

4.1 THE RESULTS

By analysing all results achieved by users on the leader board we can clearly see that there is not much difference between one of the worst[4] and the best[5] solutions when it comes to accuracy, this is because 87,57% of the users choose to travel to United States or does not choose to travel at all.

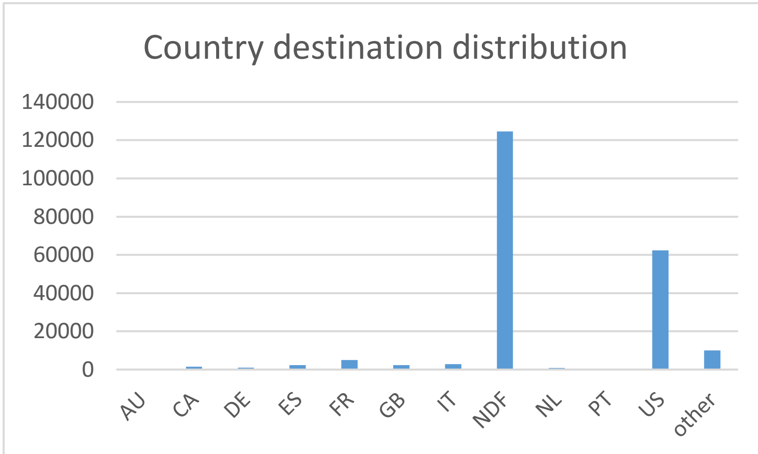

Figure 1 shows the distribution of country destination of users in the training dataset

Our 3 different attempts scored a lot different from what we had expected, at first glance we felt that there was too much scrambled data in the session's dataset to be of any use during the training process and that the data on the users where far more important for making a good prediction. XGBoost contains a function that plots which features where most valuable in construction of the model, we did not know about this feature while working on solving the problem but after seeing the plot for the

[4] The rank 1000th solution scored 0.86707

[5] The winner of the competition scored 0.88697

model used during the first attempt it became clear as to why the sessions dataset where of that high value.

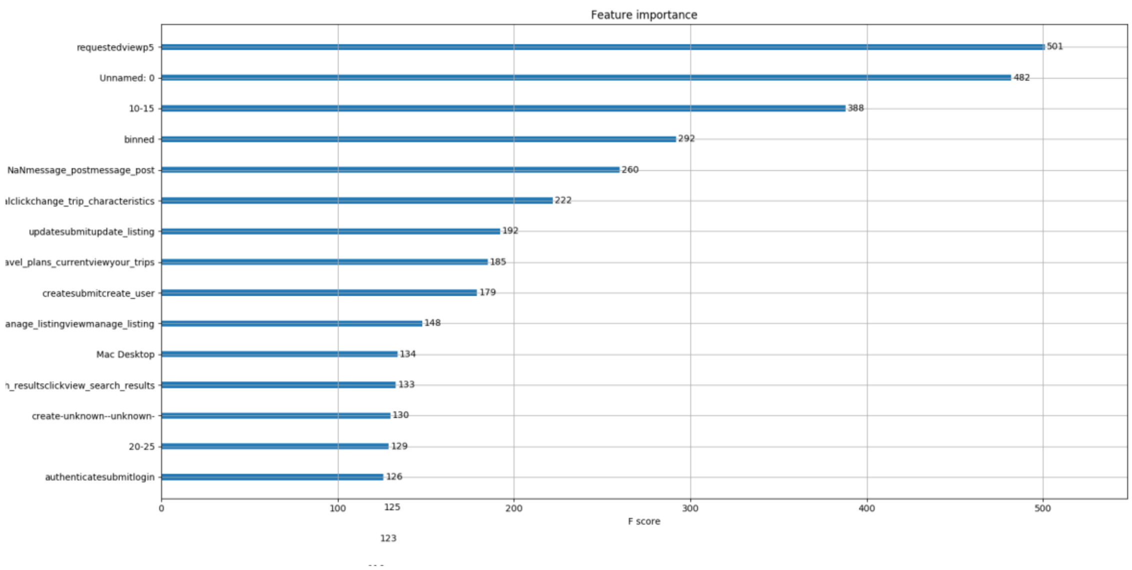

Figure 2 shows the values of the 15 most valuable features used when creating the model in the first attempt, note that all fields except 3 come from which actions where performed by the users. F-score tells us how many times those features where used as decision nodes in the model tree.

The names for the different actions and their descriptions are rather cryptic and it is hard to tell exactly what each action does, the 2 most valuable features are actions where one is called 'requestviewp5' and the other is called 'unnamed: 0'. When seeing these numbers we could make a guess that these 2 actions are closely tied to the confirmation process of travel booking. Figure 2 shows the importance the data from the session's dataset which explains why our second attempt[6] scored that much lower than our first attempt[7].

Another interesting thing to note is the difference between the first attempt and the third where we used the entire session's dataset as well as the full user dataset, the difference between those attempts in score is 0.00023 but it contained the same data and more as the first attempt. Looking at the feature importance values for this model we can see that users' age becomes the most deciding feature while the different actions remain important. The fact that the actions still hold that much value after introducing 130.000 more users without any data shows that they did not add any valuable information

---

[6] The second attempt made use of the full user dataset and ignored all data contained in the session's dataset.

[7] Our first attempt only contained data from users created during 2014 who had a record in the session's dataset.

and all we achieved by adding those users was confusing the training process and giving the different action combinations a lower weight for deciding the outcome.

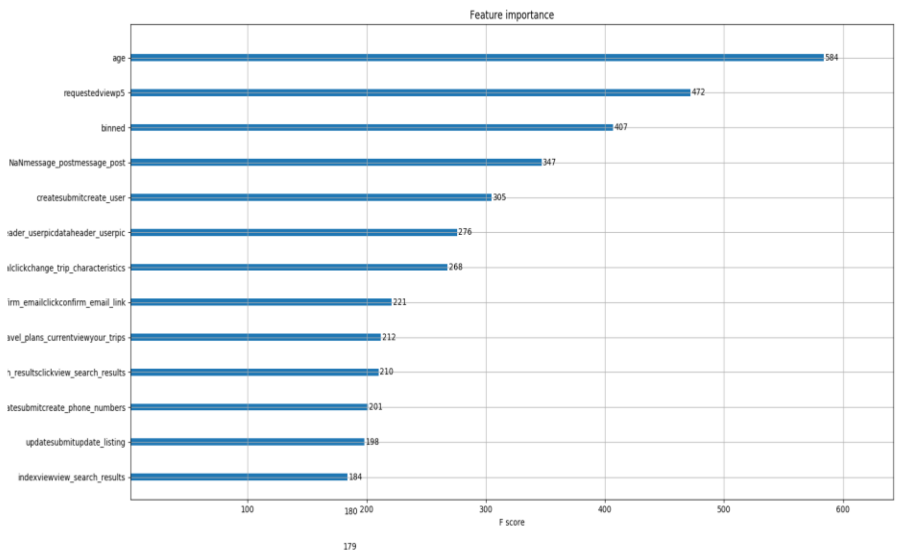

Figure 3 shows feature importance for the model used in attempt 3 where every user where present as well as using the entire session dataset

4.2 THE PROBLEM WITH THE DATASET DISTRIBUTIONS

The main problem with the datasets is the fact that we are only supplied session data for users created during 2014 which is bad since analysing feature importance shows us that the session data contains the most valuable information out of the different datasets. This in turn means that we cannot use the majority of users if we want to efficiently make use of the session dataset. As to why they choose to limit the session dataset in this way is probably because of size limitations since that dataset was by a wide margin the largest one, however it would probably have been better to provide session data for a smaller amount of users spanning over a longer time instead of just cutting it the way they did.

Another problem is only considered a problem because of the limitations brought by the session data coverage and that is the fact that we have no seasonal overlap with the training and test set of users covered by the session data. It comes naturally that the test users are newer users than the ones in the training set because that is the way it works in the real life, we use data from old users to improve the experience of new users. There is no way for us to see the difference in travel destination for the test and training users since we cannot see the actual country destination distribution of the test users, we can however see the difference between earlier users during that period and the period covered by the 2014 training users.

The following graph shows the difference in distribution of travel destinations for users in the training set with an account created during 2014 and the group of users in the months July to October in 2010 to 2013:

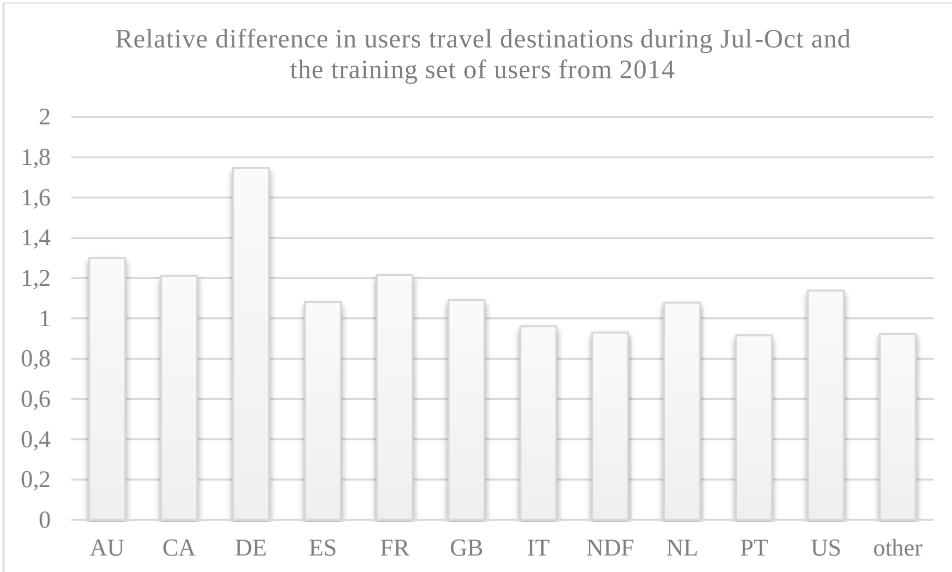

Figure 4 shows the relative difference between users in the training group with an account created during the late summer months of July to October and the users in the training dataset with a record in the session's dataset.

We can clearly see that a greater percentage of the users choose to travel somewhere during the autumn months as depicted by the fact that the NDF-row is proportionally lower during that time. Every travel destination except for Portugal, Italy and other also increase in popularity during this time. This problem would not have been as bad if they would have solved the session coverage problem the way we proposed in the previous paragraph, that way we would have had access to users spanning all seasons over several years. It is however unrealistic to expect the same distribution of travel destinations to repeat every year since the popularity of some destinations will always change based on outside factors.

4.3 Possible improvements

There are several different ways we could have improved the result on the prediction that we did not go over in the method section or have time to properly attempt because of time constraints. The only type of action we used while preparing the datasets was modifying the data in the users and session datasets and majority of the modifications was purely aimed at making the current data more understandable for the computer, for example one-hot encoding some features. We used some data modification techniques where we partly changed the data but we did not add any new data. Another thing we could have dealt with is analysing the importance of the different features to minimize the amount of features used in the final model, this would have sped up the training process and made it easier to improve on other parts of the prediction. We did not spend any time optimizing the parameters used during training, we used a default set of parameters and let a module test out the most optimal combination of parameters.

4.3.1 Adding data

There are a lot of ways we could add data from outside sources to improve the classification, the datasets we used only contained data on the users and which letter combination they chose. We could use data on popularity of the different destinations from various surveys. One problem with adding data like this is the fact that we need to format it according to our dataset, the data inside the age_gender and countries dataset already use a format similar to the datasets used in the prediction and contains a lot of interesting data.

The age_gender dataset contains data on different groups of users grouped by age buckets similar to the ones we used and we could have used this data to add the probability that their age/gender/language group have of booking a travel to the different destinations. For example if English speaking females between age 45 and 50 have a 56% probability of travelling to the United States we could add a feature for all users called prob_US and set the value for the users in the target group to 0,56. This value could be modified to account for changing popularity of the different destinations.

The countries dataset contains a couple of interesting feature groups for describing the different destinations, those features being destination location and language spoken. In a realistic scenario we would probably have access to an approximate location of the users but this data was most likely removed when they anonymized the users, this makes it a bit harder for us to make use of the location. In theory if we knew where the user is located we could calculate the distances to the different destinations and add these as features. The spoken languages in every country is coupled with the Levensthein distance from English, this could be calculated for every language on the list and added as features. We could calculate the average physical distance and language distance of users travel destinations for every group in the age_gender dataset to further improve the usefulness of this data.

4.3.2 Feature analysis

As we showed in the results section the XGBoost module contains a pretty neat method for plotting the importance of the various features, we could have used this to find which features could be filtered out entirely or condensed. By condensed we mean that we could reduce a number of related features into one new feature, for example the features concerning affiliate providers in the user datasets always ranked among the lowest in feature value so we could reduce all of the affiliate features into one: affiliate_used.

4.3.3 Parameter tuning

The last bit of improvements we can do on the model is tuning the parameters used during training, the different parameters used vary a lot for every type of machine learning algorithm, with boosted decision trees the most notably are tree size, number of trees and learning rate. These are the parameters that directly affect the model, then there are some parameters that affect the performance during training and such. The approach we chose to use when picking our parameters was running a grid search with a set of parameters, the values of the different parameters used in the grid search was chosen around the standard value of that parameter. The grid search then trained a model with each combination of parameters and tested these with cross validation, the model that scored best was finally chosen as the model used for prediction. The problem with our approach was that we were satisfied with the results of the grid search, we didn’t pay attention to the fact that the combination of parameters picked where always the same: 50 trees, 0.1 learning rate and a tree size of 5 splits. What we should have done was further testing with parameters outside the ranges used, in this case we should have tested with more and bigger trees along with slower learning.

5 CONCLUSION

Going into this problem with no previous knowledge of machine learning the slogan for the board game Othello: “a minute to learn, a lifetime to master” feels pretty accurate when talking about machine learning, just solving this type of problem and getting an accurate enough solution comes easy but if the aim is to get the best possible solution there is a million things we have to know about. We feel that with the experience we had coming into this and various time constraints that the results provided are good enough however we are confident that with the knowledge we now have on how to treat the data and perform the training process we could have achieved a much better result. Working on this project have opened our eyes for how much machine learning is actually being used already by most commercial sites to cater for their users, the future for machine learning is bright.

References

Aly, M. (2005). *Survey on Multiclass Classification Methods.*

Arlot, S., & Celisse, A. (2010). A survey of cross-validation procedures. *Statistics Surveys*, 40-79. doi:10.1214/09-SS054

Blum, A. L., & Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial intelligence*, 245-271.

Chen, T., & Guestrin, C. (2016). XGBoost: A Scaleable Tree Boosting System. *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785-794). San Francisco, California, USA: ACM. doi:10.1145/2939672.2939785

Chen, Z., Lin, F., Liu, H., Liu, Y., Ma, W.-Y., & Wenyin, L. (2002). User Intention Modeling in Web Applications Using Data Mining. *World Wide Web: Internet and Web Information Systems*, pp. 181-191.

Kotsiantis, S. B. (2007). Supervised Machine Learning: A Review of Classification Techniques. *Informatica*, 249-268.

Langley, P. (2011). The changing science of machine learning. *Machine Learning*, pp. 275-279. doi:10.1007/s10994-011-5242-y

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning.* Morgan Kaufman Publishers Inc.

Ravikumar, P., Tewari, A., & Yang, E. (n.d.). *On NDCG Consistency of Listwise Ranking Methods.*

Webb, G. I., & Micheal J Pazzani, D. B. (2001). *Machine Learning for User Modeling.* Kluwer Academic Publishers.

6 APPENDIX A – TABLES

6.1 USERS DATASET FORMAT

Applies to both user datasets

Field name	*Description*
ID	User identification used when joining with the session's dataset and validating the prediction.
Date_account_created	Date when the user's account was created, not every user have created an account.
Timestamp_first_active	The timestamp of the user's first registered action on the website.
Date_first_booking	The date when the user booked a vacation through the site.
Gender	Gender of the user.
Age	User's age.
Signup_method	Airbnb provides several options for signing up as a user on their website.
Signup_flow	The page on the site from which the user chose to sign up.
Language	Preferred language of the user.
First_affiliate_tracked	The first recorded advertisement provider that brought the user to the site.
Affiliate_channel	Which method of advertisement was used with first_affiliate_tracked
Affiliate_provider	Where the advertisement was clicked.
Signup_app	What application was used during the sign up.
First_device_type	The first recorded device used by the user.
Country_destination	The destination first chosen by the user.

6.2 SESSIONS DATASET FORMAT

Field name	*Description*
User_id	Identification of the user who performed the recorded action.
Action	What action where performed by the user.
Action_type	The type of the performed action.
Action_detail	Details of the performed action.
Device_type	The device used when the action was performed.
Secs_elapsed	Seconds elapsed since the last recorded action.

6.3 AGE - GENDER BKTS FORMAT

Field name	*Description*
Age_bucket	Which age bucket of 5 years this group of users belong to
Country_destination	The country destination of this group
Gender	The gender of this group
Population in thousands	Group size
Year	The year this group's users were created

6.4 COUNTRIES FORMAT

Field name	*Description*
Country_destination	Currently described country
Lat_destination	Latitude coordinates of the destination
Lng_destination	Longitude coordinates of the destination
Distance km	Distance in kilometres from the United states
Destination_km2	The area of the country
Destination_language	The language spoken in the country
Language_levensthein_distance	The levensthein distance between the spoken language and english