Comp6237 Data Mining Group Project

Kaggle Competition: Airbnb New User Bookings

Alun Meredith
am5e15@soton.ac.uk

Manuel Llamas
mlf1g15@soton.ac.uk

Nicola Vitale
nv5g15@soton.ac.uk

Olivia Wilson
oew1v07@soton.ac.uk

Miguel Ballesteros
mabm1e15@soton.ac.uk

1. INTRODUCTION

1.1 Task

As described in the Kaggle page, the Airbnb new user bookings competition consists of predicting the country where a new user will book their first travel experience. The Airbnb service allows booking accommodation in more than 190 countries from which the ten most popular foreign destinations and the US where subset. This contains the most common destination countries to be taken into account for the prediction, along with a special case when a user leaves the website without a transaction (NDF). This is a clear example of the 'cold start problem' where there is little initial data since no previous bookings have been made.

The available datasets contain basic user data with demographic fields, some application specific properties and user session activity while browsing for accommodation. Additionally, there is a test set and a submission sample that provide clarity on how the final outcome should look like. The code used to carry out this project is hosted on GitHub[1].

1.2 Evaluation

Kaggle evaluates the submissions using Normalised discounted cumulative gain (NDCG)

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

$$nDCG_k = \frac{DCG_k}{IDCG_k} \quad (2)$$

Where rel is the relevance of each country in the list (1 if the target country and 0 otherwise) and i is the length. This metric is rank based as predicting something second instead of first will score lower than predicting it first. The scores of some null hypothesis predictions and some forum scripts where used as a benchmark (table.3).

2. METHODOLOGY

2.1 Exploration

[1]https://github.com/miballeuk/airbnb

This section covers the exploration of the training data and its features, trying to understand the data and intuitions about them to inform the pre-processing and modeling. The data distribution and the initial challenges our models face are examined.

Tabling of each of the outputs introduces the fundamental problem of this data mining task (Figure 1). The number of users who either travel internally (US) or do not make a booking (NDF) dwarfs the other outputs with 88% of the observations combined. This means that a null hypothesis of always predicting NDF then US and no other countries achieves an NDCG score of 83.07%. The models must overcome two challenges; to correctly predict the classify US/NDF/foreign and secondly to capture the foreign destinations while maintaining a low false positive rate, otherwise this lowers the overall performance.

Relative NDF rate and US rate were computed for each feature against the rest of the population. Relative NDF rate is the difference between the percentage of NDF outcomes, while relative US rate is the difference between percentages of non-NDF outcomes which book within the US rather than externally. This reveals a number of patterns about these features. Unknown values for gender (this is also the case with age) are strongly tied to not making a booking, this may follow the logic that if users are not planning on making a booking they will not bother filling out sign up forms fully. The device type used also shows interesting patterns, Desktops are associated with low NDF rates, phones have high NDF rates and tablets are neutral. This may indicate that people browse the Airbnb from their phone like window shopping. Languages can have strong effects but there doesn't appear to be a general trend between non-English speaking and different outcomes.

The sessions data has 10M+ entries and six fields. A user ID links each record to the users data and there are only 0.03% missing values that can be discarded. There are three fields specifying the action, action type and action details, in addition to a couple of fields with device type and time spent in seconds. On average, each user performs 74 actions while browsing the site. The action fields have a high amount of repetitive value combinations, which suggest that merging them as a single field is a good bet and will not increase the number of unique values significantly. In the case of seconds elapsed, values are in the range of few seconds to

few days, possibly having a connection with the associated action. However, this field is candidate to be transformed manually into bins, defined using browsing based criteria.

When combining the training sessions and users data, about half of the sessions dataset is filtered out. There are also matching entries in the test users dataset, so any feature engineering will be useful for both the train and test operations. In particular, the combined dataset was filtered by no NDF users and therefore isolated to only those users that effectively booked a room. The idea was to find a particular action that confirms the booking operation. This was intuitively identified as 'booking_request' but with some inconsistencies since some users did not have this action even if a booking was complete. This suggest that some entries may be missing or that for a small set of users or devices the same operation is logged in a different way.

Initially, there was an attempt to identify correlations between the booking request and other actions, suggesting a sequence of booking steps for a specific country. Nevertheless, if the feature engineering creates one-hot encoding type features, those correlations will be visible by the model with no further actions.

2.2 Feature Engineering

Bag of Actions

Feature engineering is the fundamental process of using domain knowledge of the data to create features that make machine learning algorithms work. The more quality features we have from an event, the more likely is that we will be able to understand, explain and, eventually, predict similar events [1].

Having more features not necessary will lead to getting a better result. This premise was considered when applying the feature engineering actions. The feature engineering process started with brainstorming possible new features. Then, it was decided which ones could be relevant, we evaluated their performance in order to discard or update them. If necessary, this process was repeated until the work was done. Some of the methods used were combination, discretization, one-hot encoding, and filtering.

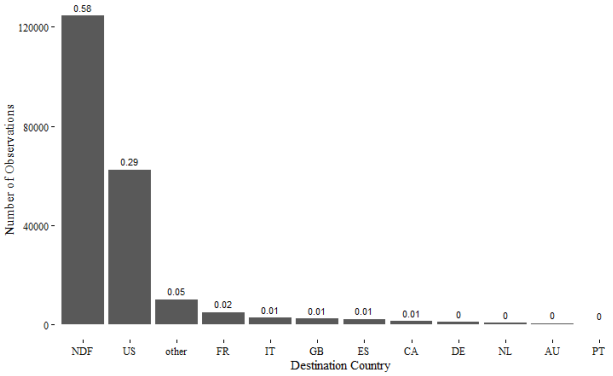

Figure 1: Number of observations for each destination of the training data. Labels show percentage of observations.

In this project we used two main datasets from Airbnb: users' profiles and users' sessions. The first step was filling 'unknown' and blank spaces with missing values for the sake of further feature engineering steps. Then the field first booking was removed considering that it doesn't exist in the test set.

It was found that the age variable had some wrong values that seemed more year of birth than age. Considering this, values above 1920 were converted into age using 2015 as reference (collected year). Other unexpected values below 14 or above 100 were changed to -1. Those are wrong values but definitely not missing, keeping in mind that missing values are crucial for this prediction. The variable age bucket was created grouping values in ranges of 5 years which are big enough to reduce unique values, but not overfitting.

Some dates were split into its components, one per field. This way the model can associate easily time sequence that make more sense. For example, the day 1 of a month is closer to the day 31 of the previous one and this might be better described by the week number. This is a great example of combining apparently useless features, that grouped with others can be very relevant [2]. Improvements were seen later in the models.

Another good example of combining variables is the lag from account created to first time active, in which 0 lag means a user effectively completing a booking. Missing values when filling the profile are a clear sign that the user is not willing to provide full details unless it needs to complete the booking process. Once every numeric variable were processed we focused on categorical variables, which we needed to transform into numeric so the algorithms could understand them. To do so we used one-hot encoding method, which creates independent fields for each unique value.

All of this successful feature engineering was being tested using XGBoost and Random Forest methods. However there were other cases with no such good results when testing. In fact, for a couple of dates values the conversion to separate features was not particularly useful.

All feature engineering operations applied to the sessions data cover the action fields and seconds elapsed. User id field is required to combine the data with the users data and does not provide prediction value. The field device type was candidate to directly be transformed using one-hot encoding.

Investigation showed that three action fields had 360, 11 and 156 unique values, combining to 457 unique combinations of these values. It was observed that with a high number of repetitive combinations, merging them in a bag of actions was a convenient reduction. This lowered the number of features to 351 when merged with the user data (on the user id field). The fields were combined to create a unique name, key for a later application of the one-hot encoding process.

Due to the high diversity of values in the seconds elapsed time field, a manual binning operation was applied to avoid a dimensionality expansion that could possibly lead to overfitting in the predictive model. The criteria used for binning

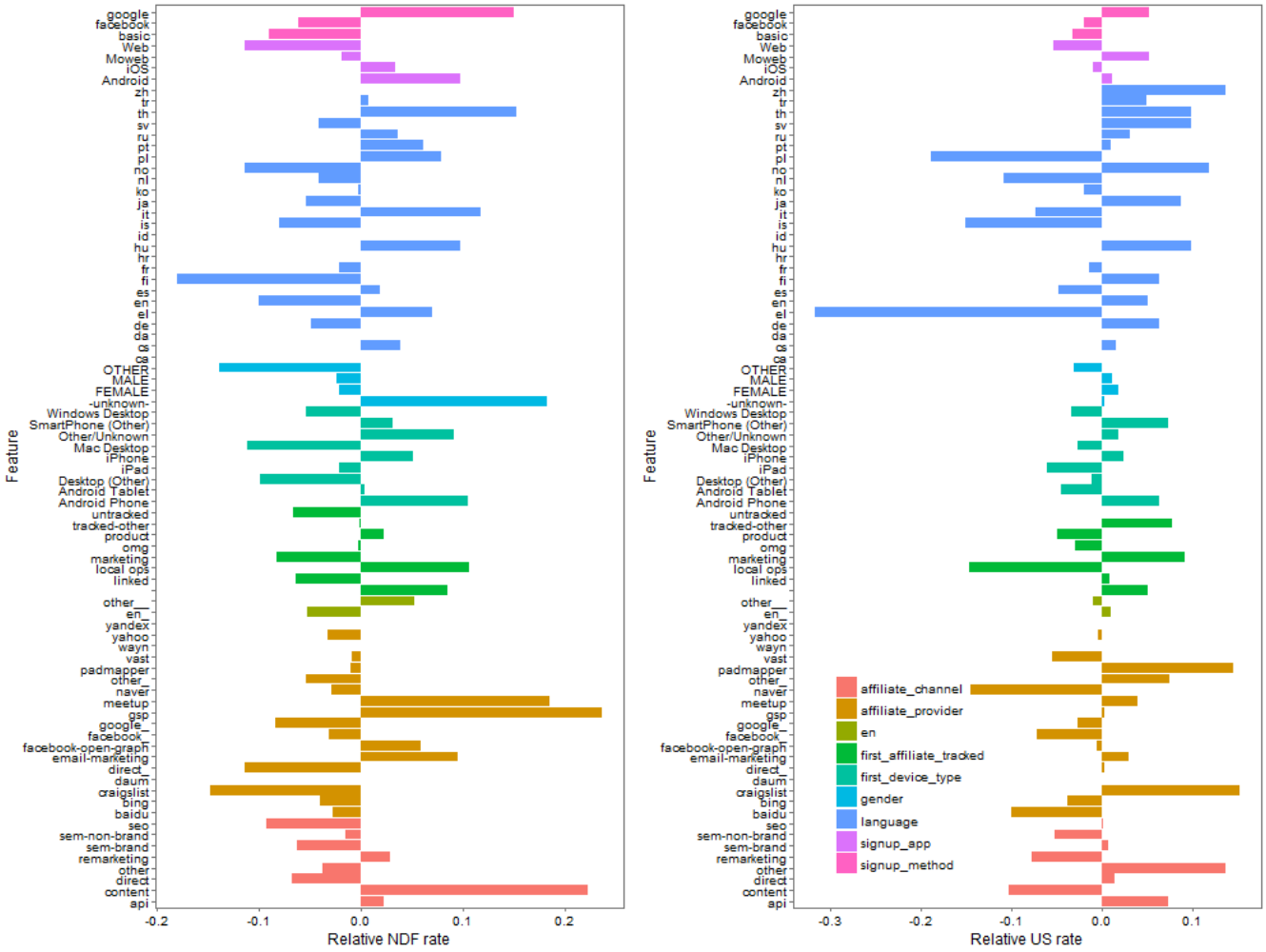

Figure 2: The relative rates of each feature against the rest of the population for NDF vs. All and US vs. External travel

was based on web site browsing behaviors estimating time ranges for common actions (Table 1).

A second 'bag of words' method was used by using a simpler model of using unique action categories as features rather than combining them, as well as not using the seconds elapsed data. Due to the merging process any users without sessions data were not included which significantly reduced the training dataset from approximately 213,000 to 73,000.

2.3 Models

XGBoost

XGBoost is an implementation in R of the extreme gradient boosting algorithm [3], originally developed by Friedman *et al.* [4]. It is a decision tree based approach which builds an ensemble of simple trees, adding new trees based on the errors of the previous ensemble. XGBoost was chosen because of its ability to train efficiently while maintaining the interpret-ability of a decision tree model. XGBoost is also a flexible model, able to operate both linearly and on factor variables and doesn't require one-vs.all methods to make predictions on multiple output factors.

When training XGBoost, factor variables must be one-hot encoded (OHE) and a number of other parameters of the training algorithm must be selected. The algorithm internally cross validates. 'Nrounds' is the maximum number of iterations (trees in the ensemble) if the model doesn't converge. 'Max Depth' is the maximum nodes deep of the tree. Gamma is the minimum loss reduction required to partition a leaf node into a decision. These two parameters restrict the complexity of the model while reducing interpretability, while improving performance by capturing more subtlety of the feature space notably when the feature space is large due to one-hot encoding deeper trees are required than if the tree were acting on linear features. However at a point the training data becomes over-fitted and no additional performance can be gained on the validation set.

In order to optimize the above parameters of the model, cross validation was done testing the performance of the model on validation sets for a variety of parameters while not being too computationally expensive.

Finally the error metric used was multiclass classification error (wrong classifications / all cases). This error metric doesn't capture the rank based approach that Kaggle evalu-

Range	Description
secs <= 0 || NA	No information available
0 < secs < 30	Within 30 seconds a user can load, read small amounts of data or fill a few-fields form.
30 < secs < 60	The user may take longer to read the page or fill a small form.
60 < secs < 300	The user is reading information in detail or filling a medium-sized form like the payments page.
300 < secs < 600	A page view taking this long can be caused by a user researching in a parallel window or simply a loss of continuity in the process.
600 < secs < 3600	The user definitely lost attention and continuity. A new action was fired, but it cannot be linked to the previous ones.
3600 < secs	The time is not reliable.

Table 1: Table showing criteria used for binning the seconds elapsed field.

ates the predictions it produces a reasonable approximation. The algorithm then outputs a classification probability for each of the output classes, which can be analyzed to produce a prediction file.

Random Forest

Random forests are an ensemble of decision trees each one training on a subset of data [5]. This method has a number of benefits in that it is not parameter heavy and can provide very good accuracy, as such it is often used in Kaggle competitions. The main parameter for a random forest is the number of decision trees, which is called `n_estimator`. This is investigated using cross-validation on the training data in 3.

Neural Network

Artificial Neural Network are applied in many situations in Data Mining. The libraries used in this project are built to train multi-layered perceptron for regression analysis, i.e. to approximate functional relationships between covariates and output variable. For this reason, ANN are sometimes referred as generalised linear models. In particular, for this research we used two libraries for ANN modeling implemented in R language: `nnet` and `neuralnet`. The first one has been used to implement the popular Stochastic Gradient Descent algorithm: different tunings for the learning rate have been tried and a decay parameter for the weights has been added to the model in a second instance. The second library has been used because of the possibility to implement the Resilient Gradient Descent as learning algorithm which include a modification on weights update at each iteration compared to the traditional version.

From the first trials it immediately appeared that this model was slow in dealing with the amount of data contained in the training set as provided by AirBnB. This penalised the feasible complexity of the network as it grows exponentially with increasing number of observations used in the training phase, number of layers and neurons in the network. Better results in terms of computational time have been obtained using a subset of the provided training set tuning the learning rate to large enough rates and eventually introducing a decay factor.

3. RESULTS

3.1 XGBoost

The XGBoost model produced the best results of the three models trained, with a score of 0.8777 (Table 3). Figure 2 shows how the model predicted on the training set for each rank. It predominantly picks along the null hypothesis (in order of frequency in the training set), especially at higher ranks. As the feature engineering and model improved we saw its ability to predict away from the null hypothesis increase. 99% of the targets are captured by the top 5 ranks at some point. This shows that the model is very good at identifying which countries a user may travel to but lacks the confidence to pick the correct one over the null hypothesis. This is reflected in the accuracy heat-map; although the model has low accuracy even when predicting non-US/NDF targets for rank 1 (high confidence) the sum of these is very high (% of each target in the top predictions). The notable exceptions to this are countries with low numbers of training examples, perhaps there was not enough data to train these targets.

	V1	V2	V3	V4	V5	targets
AU	0	2	8	88	188	152
CA	0	16	92	1083	2175	440
DE	0	4	35	394	643	250
ES	10	74	311	3186	9618	707
FR	32	292	2016	48422	17579	1435
GB	2	47	226	3427	11365	731
IT	17	134	792	13497	31184	979
NDF	50867	19275	2898	620	136	45041
NL	0	11	39	196	408	247
other	36	3361	67055	2832	447	3655
PT	0	0	3	49	69	83
US	22851	50599	340	21	3	20095
% correct	73	90	95	97	99	

Table 2: The number of times each target was predicted in each rank. % correct shows the percentage of observations in which the target was correctly identified in that position or earlier

Looking at the feature importance of the model (Figure 4) we see that the Sessions data dominates the most important features, especially the Booking Request action which intuitively is highly correlated with predicting non-NDF targets. Otherwise in the users data NA values as identified in the exploration are very important as well as signing up through Facebook, which was more important that the exploration suggests and the date variables. Date variables were weighted significantly higher by the Random Forest Model.

3.2 Random Forest

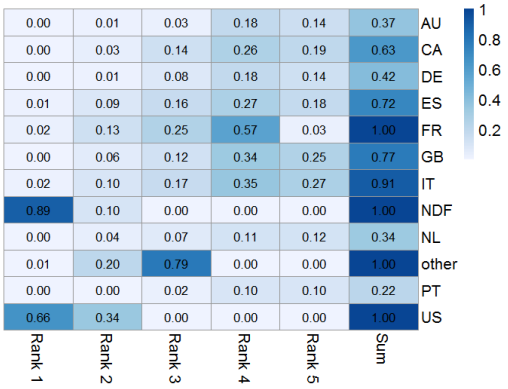

Figure 3: Accuracy of each of the predictions in table 2.

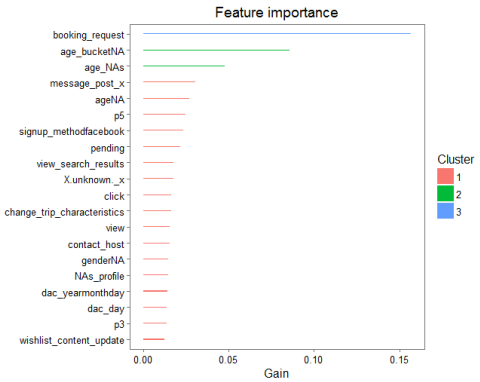

Figure 4: 20 Most Important features measured by the average gain of each feature in each tree of the XGB ensemble

Using cross validation (60% training 40% test) of the training data the best accuracy for each number of decision trees was found. The number of decision trees was restricted to a maximum of 150 for time constraint reasons. Figure 6 shows that with this data a large number of trees was required to obtain the best accuracy results, all datasets requiring either 142 or 150 learners. Using the respective number of learners, a single random forest was trained for each dataset and a confusion matrix showing errors of commission and omission was created. This is intriguing, as it confirms suspicions that there are many country destinations being classified as NDF or US for other countries. The NDCG for the user and session data, user data without dates, user hot encoded (testing data) were 0.71533, 0.84480 and 0.86216 respectively. This is interesting since other methods did not find the user hot encoded data as accurate as the user and session data.
These three random forests produced very different feature importances. For example, the user no dates data thought that the most important was age, first affiliate tracked, first browser (in that order), with age always being the most important. This could be due to the fact mentioned in Section 2 that an NA value for age was correlated with a country destination of NDF. Whereas, dac day, dac yearmonthday, tfa yearmonthday were always top with the one hot encoded data, which is at odds with the other classifiers for this dataset. It is unknown why the date columns were so important in this model which also featured the highest accuracy for decision trees. Finally the sessions data had a top three of view, dac day, and click, which is more understandable in that the more pages that someone has viewed the more likely they are to book somewhere.

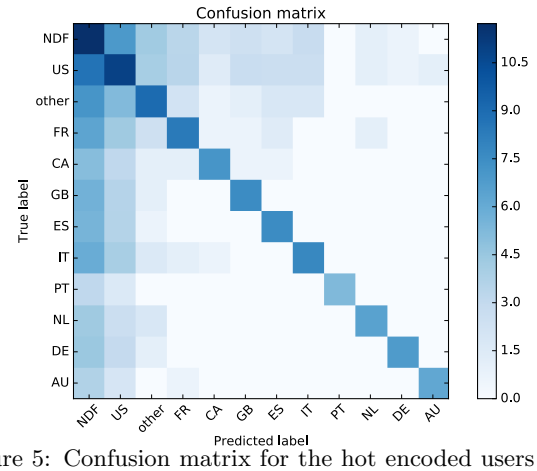

Figure 5: Confusion matrix for the hot encoded users data on a log scale. The first and most probable country was chosen from the prediction file.

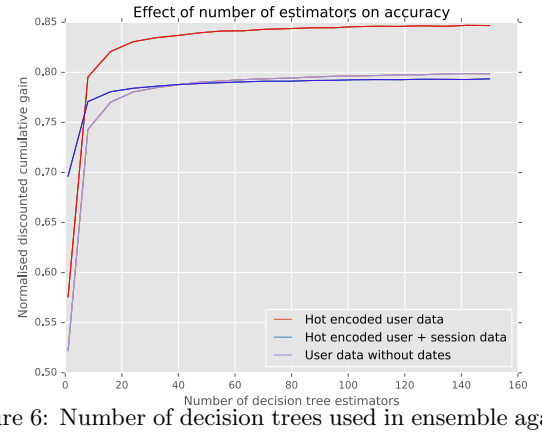

Figure 6: Number of decision trees used in ensemble against NDCG for the three different datasets. This was created using a 60% training and 40% test splits on the training data and run 10 times per n_estimator.

3.3 Neural Network

Regarding traditional SGD learning algorithm implemented in nnet it uses the following weights update rule:

$$w_i \leftarrow w_i - \eta * \frac{\partial E}{\partial w_i} \tag{3}$$

The best results have been obtained using this library when including in the model a learning rate etha of 0.3 and a decay factor lambda of 0.05 according to the following formula:

$$w_i \leftarrow w_i - \eta * \frac{\partial E}{\partial w_i} - \eta \lambda w_i \tag{4}$$

This combination and the application of a network formed of two layers of respectively 30 and 15 neurons, allowed to obtain the highest score with the neural network of 0.8513. The error function was computed as sum of squared errors (SSE) as follows:

$$E = \frac{1}{2} \sum_{l=1}^{L} \sum_{h=1}^{H} (o_{lh} - y_{lh})^2 \tag{5}$$

	NGDC
XGB	0.8777
NN	0.8513
RF	0.8622
NDF	0.6841
NDF_US	0.8307
XGB_Benchmark	0.8655
Null	0.6854

Table 3: NGDC scores for each trained model (NN, RF and XGB) and some benchmark predictions (always NDF, NDF/US, NDF/US/other/FR/IT and an XGB benchmark with minimal feature engineering [6]

Using neural net library the resilient gradient descent has been implemented. This variation consist in considering just the sign of the error derivative instead of its magnitude, so that the learning update remains constant and equal to the learning rate. Even if in literature this variation is reported to perform better in some cases this didn't applied to our project.

This library also offers plotting possibilities. Nevertheless, the topology of the network it is not comprehensible due to the large number of input contained in the engineered dataset. This library stores more statistics about the model in a list. Particularly interesting is the computation of generalized weights. The generalized weights are given for all covariates and their distribution provide us with information about the influence of the single covariate on the model.

4. CONCLUSION

The results of this project show that most of the learners do not have the confidence to learn a classifier that will reject the null hypothesis of NDF and US against any other country destination. Indeed, just learning these provides a relatively good accuracy. Even when learner did not choose the null hypothesis it was not confident in its prediction. It was also observed that user sessions data added value to two of the classifiers (XGBoost and Neural Networks), and significantly outperforming just the hot-encoded user profile data. However this was not the case with the random forests where the sessions data resulted in a lower test accuracy than the raw user profile data. This is likely due to the feature explosion of adding the sessions data and would need longer training and more estimators to get better results. However the computational requirements would have been too large for this project.

In all the classifiers it was found that dates of either account created or first activity had high importance. However it was not specific date fields it was the combined YearMonthDay fields that had high importance. It is hard to say why this might be the case but it may be spikes on particular dates like 4^{th} July. Also as previously mentioned, NA values also feature highly in the importances, especially in the case of age.

By applying most of the feature engineering techniques and obtaining a reasonably higher amount of fields, the models managed to boost the accuracy. There was room to experiment a bit with different features as the computational requirements were not that high in most cases. The sessions data and in particular the action related fields added a lot of noise with the exception of few of them that turned out to be very important. Finding and filtering those values required few iterations through the whole process, which lets us state that feature engineering is a permanent process requiring the constant feedback provided by the models.
Regarding the ANN implementation, this model has to be adapted when dealing with large amount of data both in terms of observations and in terms of variables since they heavily affect the complexity of the network. Relevant steps in this sense may be: finding a learning algorithm which is more suitable for the data include factors to improve weights update when dealing with a big number of features.

4.1 Future Work

Through the course of this project we encountered several challenges that would form the basis of future work. NDCG error functions are not present in the out of the box libraries used, a custom NDCG error function was built for the random forest model and this should be extended to the other two models. Equally the submission files used a naive approach of submitting the top 5 predictions for each user but this could be improved to optimize NDCG. The performance of the models could be improved in a number of ways, oversampling or bootstrapping methods could improve the model against targets with low numbers of observations, using neural network learning algorithms more suited to larger scale networks could improve this model and ensembling the methods together may may yield a stronger model maximizing the strengths of each method. Finally parsing the trees produced by the model could yield a better understanding of how the model is classifying the data and lead to a much stronger understanding of the underlying effects, e.g. find important features for the subset of predicting Spain.

References

[1] Analytics Vidhya, "A comprehensive guide to data exploration," 2016. (Accessed on 04/05/2016).

[2] A. Bouchard-Côté, "Feature engineering and selection," October 2009.

[3] T. Chen, T. He, and M. Benesty, *xgboost: Extreme Gradient Boosting*, 2016. R package version 0.4-3.

[4] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[5] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[6] indradenbakker, "Airbnb xgb benchmark," January 2016.