Machine Learning Engineer Nanodegree

Capstone Project

Vladimir Baranov
January 12th, 2018

I. Definition

Project Overview

Machine learning is the science of helping computers discover patterns and relationships in data instead of being manually programmed. It's a powerful tool for creating personalized and dynamic experiences, and it's already driving everything from Netflix recommendations to autonomous cars. One of the main goals of machine learning is to improve the quality of service provided by a company [1]. This leads to better user experience, customer satisfaction and retention resulting in higher sales. At the same time knowing the needs of a user helps a company to reduce operational costs [2].

Airbnb is an online marketplace where people list, discover, and book accomodations around the world. It has collected various data about users. This data about the usage patterns of its present user base can be utilized to predict patterns about its future users to provide them with customized suggestions to serve Airbnb's customers better.

This project is a Kaggle competition that was run by Airbnb from Nov 2015 to Feb 2016 [3].

Problem Statement

In this project we are going to train a multiclass classifier in order to predict which country will be the first destination of a new user given a dataset containig information of a user, his activities on airbnb, summary statistics of a user and destination countries. We will solve this problem by fitting different classifiers on the cleaned dataset. Eventually we will choose the best model. The final result of our simulations will be a list of five most probable countries for every user.

Metrics

The evaluation metric for this project is $NDCG_k$ (normalized discounted cumulative gain) where $k = 5$.

DCG is a measure of ranking quality. DCG measures the gain of a country based on its position in the result list. The gain is accumulated from the top of the result list to the bottom, with the gain of each result discounted at lower ranks. Then DCG at each position for a chosen value of k is normalized across predictions.

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 and $IDCG_k$ 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.

When we predicting the countries, 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 $NDCG = \frac{2^0-1}{log_2(1+1)} + \frac{2^1-1}{log_2(2+1)} = \frac{1}{1.58496} = 0.6309$

II. Analysis

Data Exploration

The dataset was obtained from Kaggle [3]. It consists of 5 csv files.

train_users.csv - the training set of users containing 213451 rows of continious and categorical data

test_users.csv - the test set of users containing 62096 rows of continious and categorical data

- id: user id
- date_account_created: the date of account creation
- timestamp_first_active: timestamp of the first activity, note that it can be earlier than date_account_created or date_first_booking because a user can search before signing up
- date_first_booking: date of first booking
- gender
- age
- signup_method
- signup_flow: the page a user came to signup up from
- language: international language preference
- affiliate_channel: what kind of paid marketing
- affiliate_provider: where the marketing is e.g. google, craigslist, other
- first_affiliate_tracked: whats the first marketing the user interacted with before the signing up
- signup_app
- first_device_type
- first_browser
- country_destination: **target variable**

sessions.csv - web sessions log for users

- user_id: to be joined with the column 'id' in users table
- action
- action_type
- action_detail
- device_type
- secs_elapsed

countries.csv- summary statistics of destination countries in this dataset and their locations

age_gender_bkts.csv - summary statistics of users' age group, gender, country of destination

In this project we will use the first three files - training and testing sets as well as web sessions log. Let us have a look on transposed training and test sets: Out[73]:

	0	1	2	3	4
id	gxn3p5htnn	820tgsjxq7	4ft3gnwmtx	bjjt8pjhuk	87mebub9p4
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
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
date_first_booking	NaN	NaN	2010-08-02	2012-09-08	2010-02-18
gender	NaN	MALE	FEMALE	FEMALE	NaN
age	NaN	38	56	42	41
signup_method	facebook	facebook	basic	facebook	basic
signup_flow	0	0	3	0	0
language	en	en	en	en	en
affiliate_channel	direct	seo	direct	direct	direct
affiliate_provider	direct	google	direct	direct	direct
first_affiliate_tracked	NaN	NaN	NaN	NaN	NaN
signup_app	Web	Web	Web	Web	Web
first_device_type	Mac Desktop	Mac Desktop	Windows Desktop	Mac Desktop	Mac Desktop
first_browser	Chrome	Chrome	IE	Firefox	Chrome

```
Number of rows, columns in training set: (213451, 15)
```

Out[72]:

	0	1	2	3	4
id	5uwns89zht	jtl0dijy2j	xx0ulgorjt	6c6puo6ix0	czqhjk3yfe
date_account_created	2014-07-01 00:00:00	2014-07-01 00:00:00	2014-07-01 00:00:00	2014-07-01 00:00:00	2014-07-01 00:00:00
timestamp_first_active	2014-07-01 00:00:06	2014-07-01 00:00:51	2014-07-01 00:01:48	2014-07-01 00:02:15	2014-07-01 00:03:05
date_first_booking	NaN	NaN	NaN	NaN	NaN
gender	FEMALE	NaN	NaN	NaN	NaN
age	35	NaN	NaN	NaN	NaN
signup_method	facebook	basic	basic	basic	basic
signup_flow	0	0	0	0	0
language	en	en	en	en	en
affiliate_channel	direct	direct	direct	direct	direct
affiliate_provider	direct	direct	direct	direct	direct
first_affiliate_tracked	NaN	NaN	linked	linked	NaN
signup_app	Moweb	Moweb	Web	Web	Web
first_device_type	iPhone	iPhone	Windows Desktop	Windows Desktop	Mac Desktop
first_browser	Mobile Safari	Mobile Safari	Chrome	IE	Safari

```
Number of rows, columns in test set: (62096, 15)
```

The main dataset has 15 features. Training and testing sets contain values describing 213451 and 62096 different users, respectively. The first feature is 'id'. We will drop this feature later.

Here is the list of features in the main dataset:

- **date_account_created**: yyyy-mm-dd
- **timestamp_first_active**: yyyy-mm-dd hh:mm:ss
- **gender**: 'male','female', NaN
- **age**: continuous, NaN
- **signup_method**: 'basic', 'facebook', 'google', 'weibo'
- **signup_flow**: 0, 25, 12, 3, 2, 23, 24, 1, 8, 6, 21, 5, 20, 16, 15, 14, 10, 4 (categorical)
- **language**: 'en', 'zh', 'fr', 'es', 'ko', 'de', 'it', 'ru', 'ja', 'pt', 'sv', 'nl', 'tr', 'da', 'pl', 'no', 'cs', 'el', 'th', 'hu', 'id', 'fi', 'ca', 'is', 'hr', nan
- **affiliate_channel**: 'direct', 'sem-brand', 'sem-non-brand', 'seo', 'other', 'api', 'content', 'remarketing'
- **affiliate_provider**: 'direct', 'google', 'other', 'facebook', 'bing', 'craigslist', 'padmapper', 'vast', 'yahoo', 'facebook-open-graph', 'gsp', 'meetup', 'email-marketing', 'naver', 'baidu', 'yandex', 'wayn', 'daum'
- **first_affiliate_tracked**: nan, 'linked', 'omg', 'tracked-other', 'product', 'marketing', 'local ops'
- **signup_app**: 'Web', 'iOS', 'Android', 'Moweb'
- **first_device_type**: 'Mac Desktop', 'Windows Desktop', 'iPhone', 'iPad', 'Other/Unknown', 'Android Phone', 'Android Tablet', 'Desktop (Other)', 'SmartPhone (Other)'
- **first_browser**: 'Chrome', 'Safari', nan, 'Firefox', 'Mobile Safari', 'IE', 'Chrome Mobile', 'Android Browser', 'AOL Explorer', 'Opera', 'Silk', 'IE Mobile', 'BlackBerry Browser', 'Chromium', 'Mobile Firefox', 'Maxthon', 'Apple Mail', 'Sogou Explorer', 'SiteKiosk', 'RockMelt', 'Iron', 'Yandex.Browser', 'IceWeasel', 'Pale Moon', 'CometBird', 'SeaMonkey', 'Camino', 'TenFourFox', 'Opera Mini', 'wOSBrowser', 'CoolNovo', 'Avant Browser', 'Opera Mobile', 'Mozilla', 'SlimBrowser', 'Comodo Dragon', 'OmniWeb', 'Crazy Browser', 'TheWorld Browser', 'Flock', 'PS Vita browser', 'IceDragon', 'Googlebot', 'Conkeror', 'Arora', 'UC Browser', 'Stainless', 'Google Earth', 'IBrowse', 'NetNewsWire', 'Kindle Browser', 'Outlook 2007', 'Nintendo Browser', 'Palm Pre web browser', 'Epic'

There are two features here containing time and date. The feature 'age' is numerical. The rest of the dataset contains categorical data. Target value contains 12 different classes. Over half the train users did not book a destination ('NDF'), and the 'US' is the most common destination country.

```
Number of rows, columns in combined dataset: (275547, 15)


Target values:
NDF      124543
US        62376
other     10094
FR         5023
IT         2835
GB         2324
ES         2249
CA         1428
```

```
DE        1061
NL         762
AU         539
PT         217
Name: country_destination, dtype: int64
```

Statistical analysis of data:

```
                 age    signup_flow
count  158681.000000  275547.000000
mean       47.145310       4.291965
std       142.629468       8.794313
min         1.000000       0.000000
25%        28.000000       0.000000
50%        33.000000       0.000000
75%        42.000000       1.000000
max      2014.000000      25.000000
```

The percentage of missing data in training set:

```
id                          0.000000
date_account_created        0.000000
timestamp_first_active      0.000000
date_first_booking         58.347349
gender                     44.961139
age                        41.222576
signup_method               0.000000
signup_flow                 0.000000
language                    0.000000
affiliate_channel           0.000000
affiliate_provider          0.000000
first_affiliate_tracked    54.015676
signup_app                  0.000000
first_device_type           0.000000
first_browser              12.773892
dtype: float64
```

The percentage of missing data in test set:

```
id                           0.000000
date_account_created         0.000000
timestamp_first_active       0.000000
```

```
date_first_booking        100.000000
gender                     54.502705
age                        46.502190
signup_method               0.000000
signup_flow                 0.000000
language                    0.001610
affiliate_channel           0.000000
affiliate_provider          0.000000
first_affiliate_tracked    54.704007
signup_app                  0.000000
first_device_type           0.000000
first_browser              27.583097
dtype: float64


The percentage of missing data in combined set:


id                        0.000000
date_account_created      0.000000
timestamp_first_active    0.000000
date_first_booking        0.677340
gender                    0.471114
age                       0.424124
signup_method             0.000000
signup_flow               0.000000
language                  0.000004
affiliate_channel         0.000000
affiliate_provider        0.000000
first_affiliate_tracked   0.541708
signup_app                0.000000
first_device_type         0.000000
first_browser             0.161112
dtype: float64
```

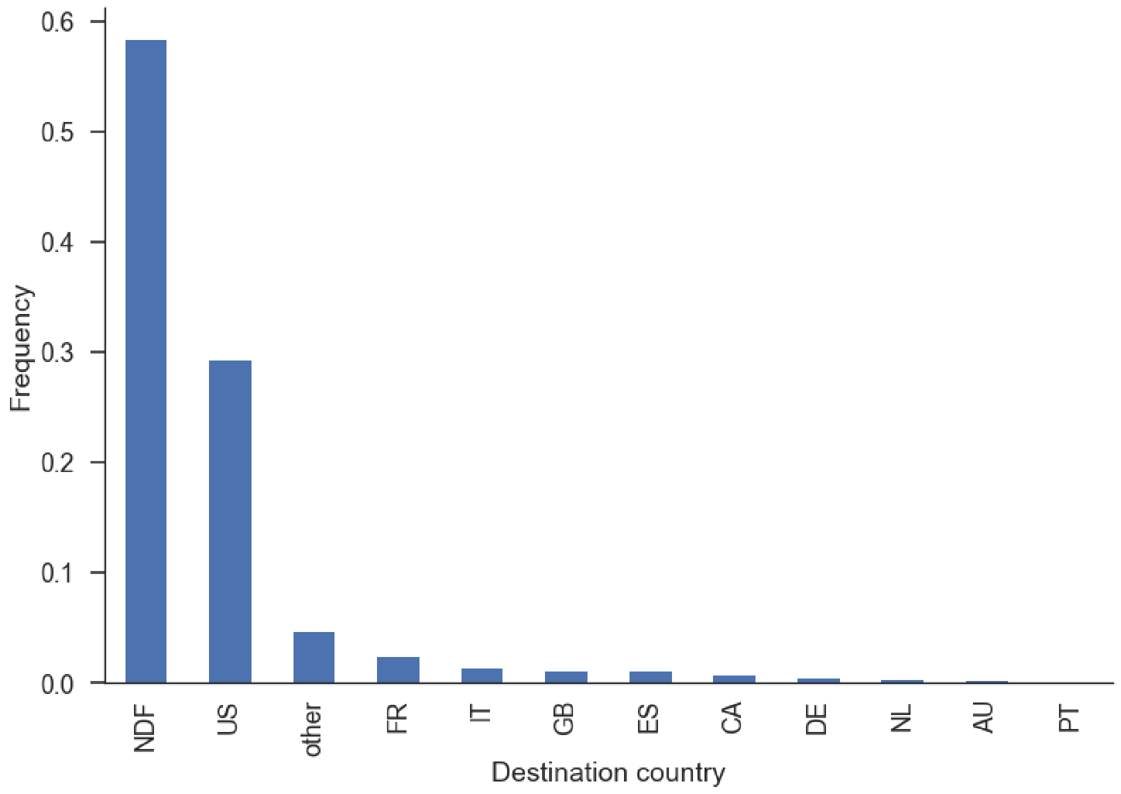
We will drop the feature 'date_first_booking' since it is absent in the test set. Features 'gender', 'age', and 'first_affiliate_tracked' contain about 50% of missing data, while there is 16% and less then 1% of missing data in the features 'first_browser' and 'language', respectively. Moreover, the feature 'age' has minumum and maximum values equal to 1 and 2014, respectively, which is beyond the airbnb policy. These abnormal values should be considered as outliers.

```
Number of rows, columns in sessions set: (213451, 15)
```

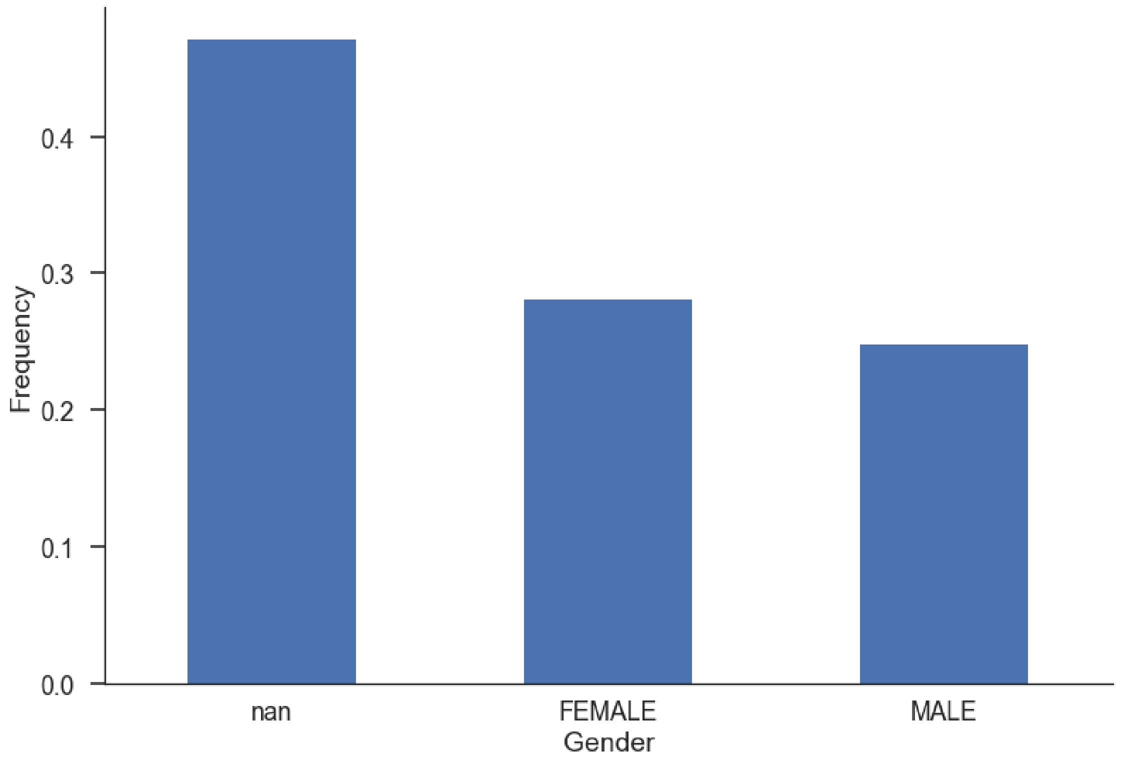
The sessions set has 6 features. It contains information of different user sessions in each line totalling to 10567737 lines.

Exploratory Visualization

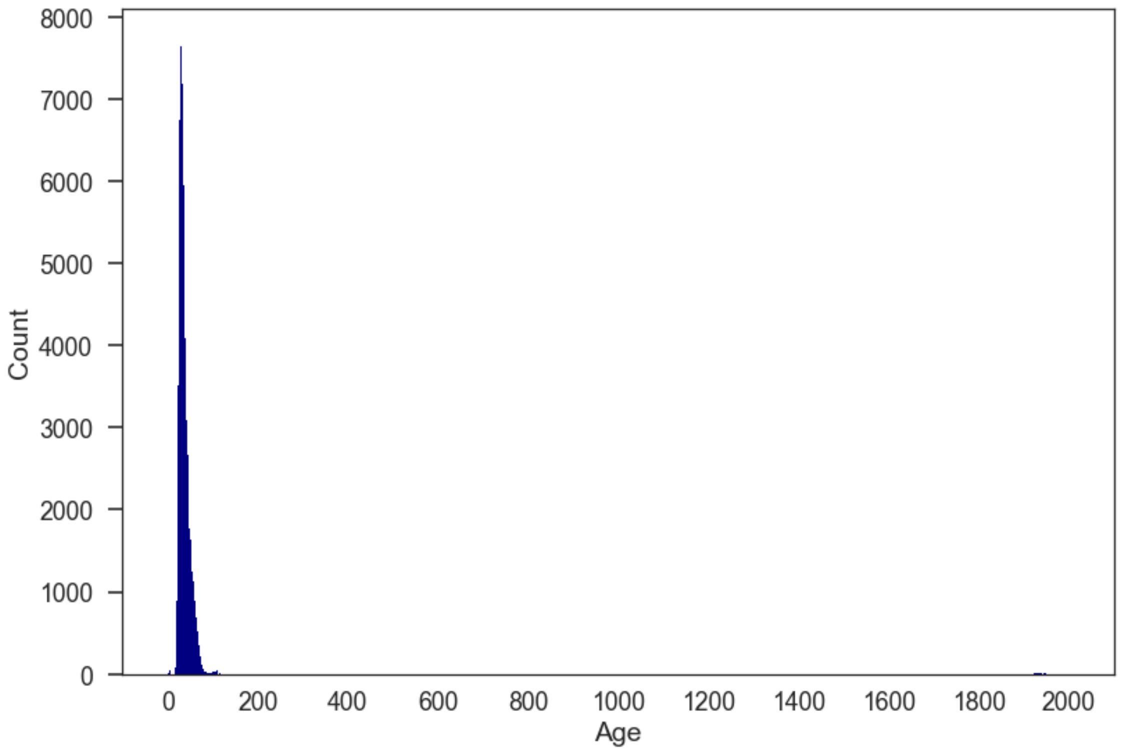
Let us visualize some properties of our dataset.



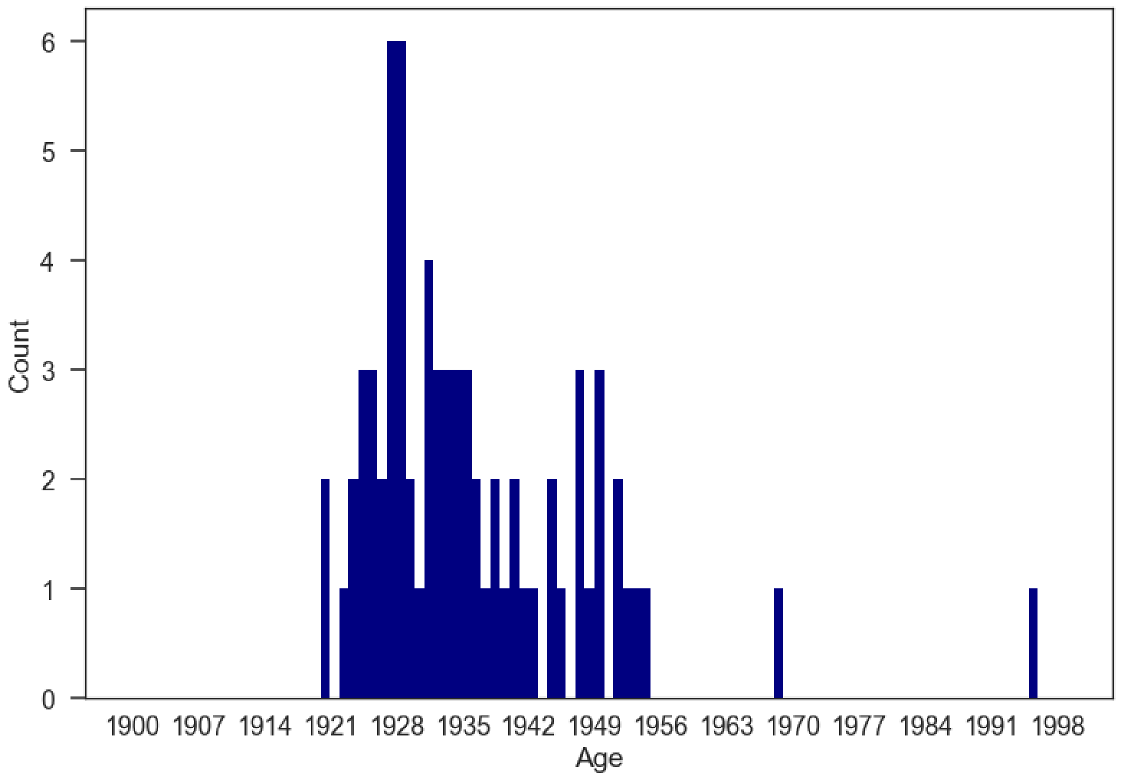
As we pointed out earlier, the major category is ’NDF’, followed by ’US’. This two categories correspond to about 90% of the data.



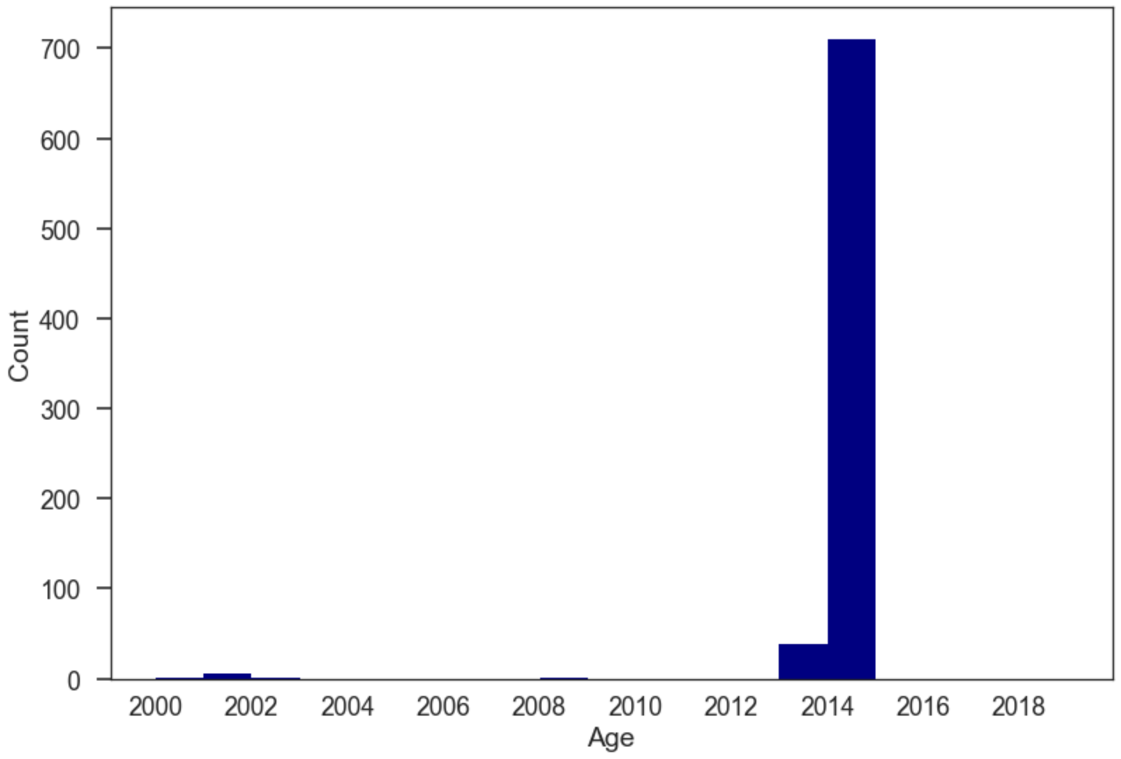
There is also about 50% of missing 'gender' data.



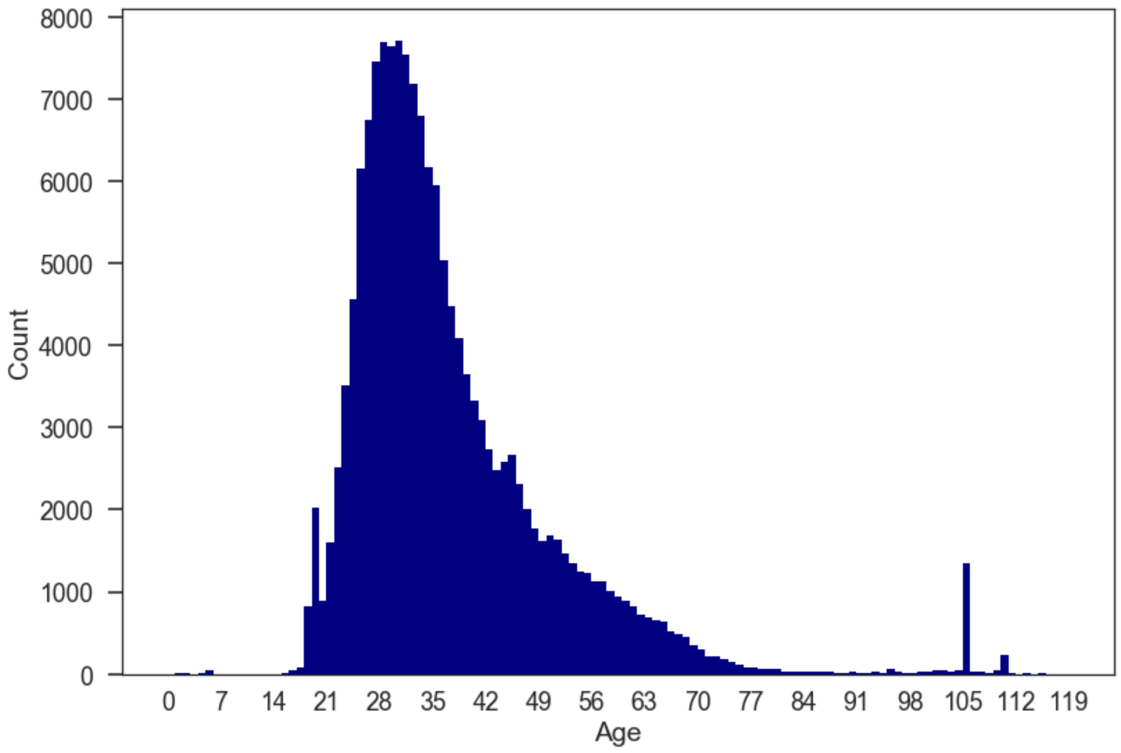
As we noticed before, ’age’ contains values of the order of thouthands. Let us look to that area closer.



The ages in between of 1920 and 1999 are most probably mistaken with the birth year. We will replace this values further.



There is a peak on the figure at the age=2014. This is definitely a mistake. We will delete these values.



Here we can see that the highest amount of users are about 30 years old. There is an unnatural peak at age=105. We will delete the values above 104 and below 15 as unreliable.

Algorithms and Techniques

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

1. First of all, we will clean the dataset. We will restrict 'age' to believable values and remove all the outliers.
2. Then we will exctract such features as hour, day, week, quarter, year, time of day, and season from 'date_account_created' and 'timestamp_first_active'.
3. Further, all the categorical data will be transformed using One Hot Encoding.
4. Next, we will train 4 different classifiers. Here is their summary.

The **Logistic regression** algorithm is the simplest classification algorithm used for classification tasks. In the case of multiclass logistic regression, with K classes this model takes the feature values $\mathbf{X}_i$ with corresponding weights $\boldsymbol{\beta}_i$ and calculates the probabilities Pr using the softmax function:
$\Pr(Y_i = k) = \frac{exp(\boldsymbol{\beta}_k \cdot \mathbf{X}_i)}{\sum_{c=0}^{k} exp(\boldsymbol{\beta}_c \cdot \mathbf{X}_i)}$.
Later the calculated probabilities used to find the target class. In general, the high probability class treated as the final target class [4]. Moreover, logistic regression returns well calibrated probability predictions by default as it directly optimizes log-loss [5].

Support Vector Machine (SVM) algorithm plots each data item as a point in n-dimensional space (where n is number of features) with the value of each feature being the value of a particular coordinate. Then, it performa classification by finding the hyper-plane that differentiate the two classes very well. Support Vectors are simply the coordinates of individual observation. SVM is a frontier which best segregates the two classes (hyper-plane/ line). In a case of a multi-class classification problem, one-vs.-all strategy is used. This strategy involves training a single classifier per class, with the samples of that class as positive samples and all other samples as negatives.

The advantages of SVM are [6]:

- SVMs are effective when the number of features is quite large.
- It works effectively even if the number of features are greater than the number of samples.
- Non-Linear data can also be classified using customized hyperplanes built by using kernel trick.
- It is a robust model to solve prediction problems since it maximizes margin.

We can illustrate the **AdaBoost** method on a following example.

Let's imagine we have to list out all the people in the city of San Francisco who are taller than 5'7, weigh less than 190 lbs, and are between the ages of 28 and 41. However we are supposed to do this without the help of machines. All we are allowed to do is take a look at the person and determine whether or not that person qualifies. How do we do it?

In order to improve the accuracy, we get three people to help us out. The first person is really good at guessing the height, the second person is really good at guessing the weight and the third person is really good at guessing the age. Individually, they may not be all that useful, because they can do only one simple task. But if we combine them together and filter out all the people, we have a very good chance of getting an accurate list of people who qualify. This is the concept behind AdaBoost. [7]

In other words, we first assign equal weights to all the training examples and choose a base algorithm. At each step of iteration, we apply the base algorithm to the training set and increase the weights of the incorrectly classified examples. We iterate n times, each time applying base learner on the training set with updated weights. The final model is the weighted sum of the n learners [8].

The advantages of AdaBoost are [9]:

- It's fast and simple method
- Does feature selection resulting in relatively simple classifier
- Not prone to ovefitting
- Fairly good generalization

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework, which produces an ensemble of weak decision tree learners via additive training.

The advantages of Gradient Tree Boosting are [10]:

- Natural handling of data of mixed type (= heterogeneous features)
- Predictive power
- Robustness to outliers in output space (via robust loss functions)
- XGBoost is 20x faster than scikit-learn's Gradient Boosting Classifier and is widely used in the winning solution on Kaggle.

5. As the final step, we will chose the best model according to NDCG score and predict top-5 countries for each user.

Benchmark

Since this is a Kaggle competition, we will use the top 10 percentile score from the leaderboard as the benchmark, which is equal to 0.8850 in private leaderboard and 0.8802 in the public one.

III. Methodology

Data Preprocessing

Data processing will be carried out in two steps: data cleaning and data transformation.

1. Data Cleaning

   In this section we will clean the data and make it suitable for machune learning algorithms.

   First, let us combine the training and testing sets in order to make data cleaning easier.

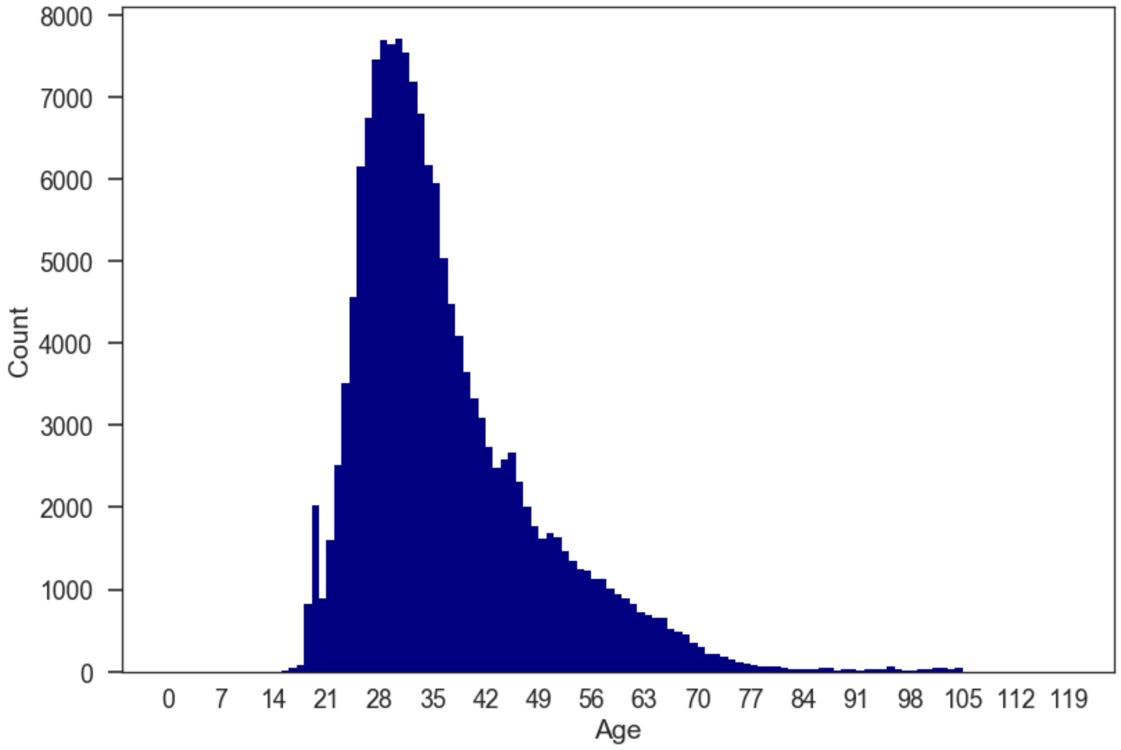
   As the following step, we drop the date_first_booking column. The reason for that is that the test set doesn't contain any information on this feature. Therefore it doesn't make any sense to take into account this feature in the training process.

Next, we clean the 'age' feature. We replace 'age' in the interval (1900,1999) as 2014-'age' since we believe that in this case 'age' represents the year of birth. We also replace the 'age' values less then 15 and higher then 104 with NaN as beyond reasonable. As the result, the 'age' feature gets the following shape.

```
C:\Users\Vladimir\Anaconda2\lib\site-packages\ipykernel_launcher.py:2: RuntimeWarning: invalid

C:\Users\Vladimir\Anaconda2\lib\site-packages\ipykernel_launcher.py:2: RuntimeWarning: invalid

C:\Users\Vladimir\Anaconda2\lib\site-packages\ipykernel_launcher.py:3: RuntimeWarning: invalid
  This is separate from the ipykernel package so we can avoid doing imports until
C:\Users\Vladimir\Anaconda2\lib\site-packages\ipykernel_launcher.py:4: RuntimeWarning: invalid
  after removing the cwd from sys.path.
```



As the final step of data cleaning, we replace NaN values. To fill the missing data of the 'language' feature we use the most frequent value. The amount of missing data here is a fraction of percent and such replacement will not affect the data a lot. The missinge data belonging to 'age' are replaced with -1. There is about 50% of missing data in the dataset. Therefore, using mean value of this feature would have unbalance the data too much. Last, the missing data of all other categorical data are replaced with 'missing'. As it was pointed out in the Data exploration section, the amount of the missing data in this case ranges from 16% to about the half.

```
2. Data transformation.
```

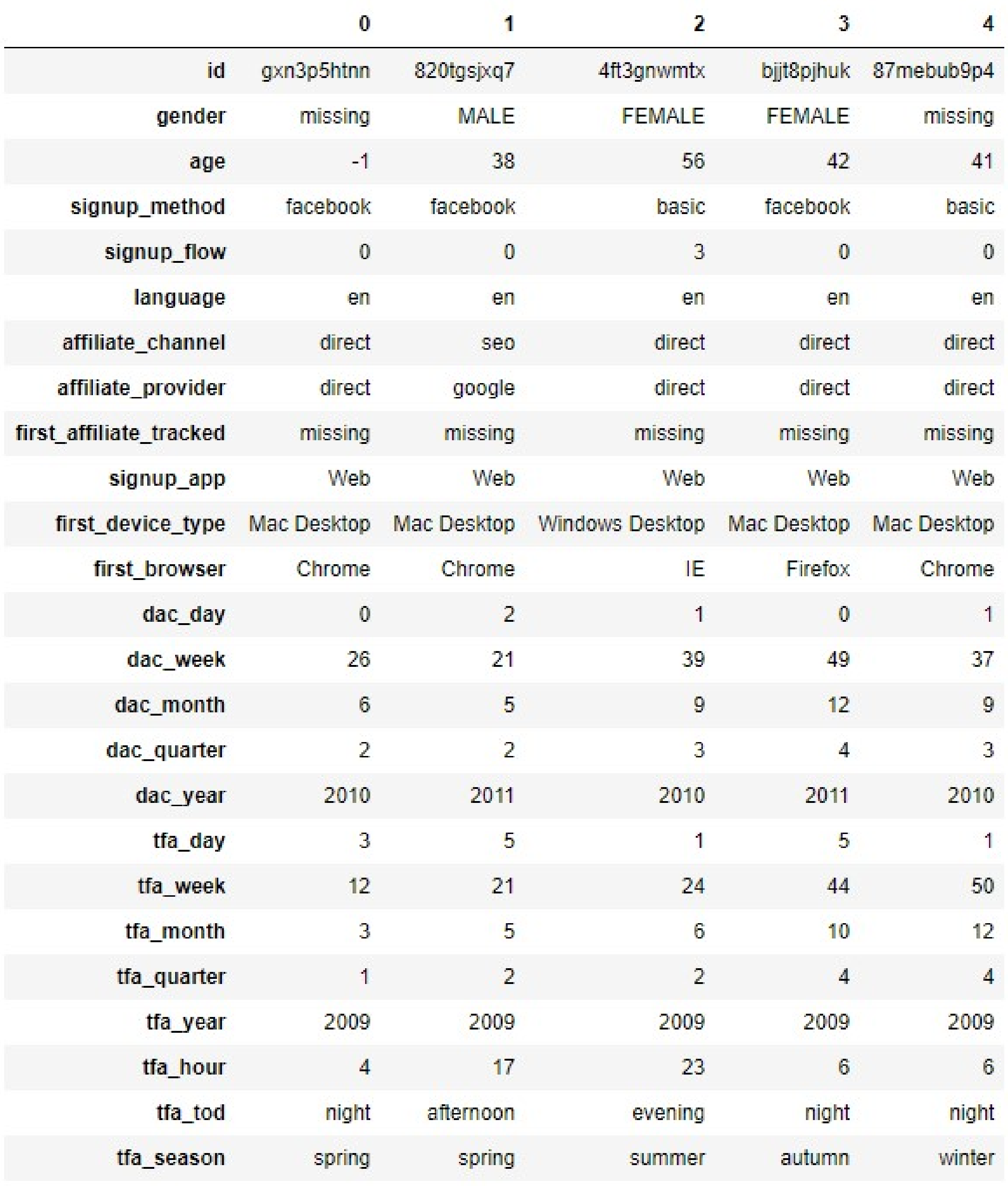
In this subsection we will work with with dates. First of all, we indicate that **date_account_created** and **timestamp_first_active** are dates when loading data, which lets the data to have the correct form: yyyy-mm-dd hh:mm:ss. Then we change the type of the features **date_account_created** and **timestamp_first_active** to datetime and create new features based on them. These new features include weekday, week, month, quarter, and year. Besides we create new features comprising hour, time of day and season based on **timestamp_first_active**. Finally we delete the initial date features and the column 'id'. The resulting featureset is the following:

- **gender**: 'male','female', NaN
- **age**: continuous, NaN
- **signup_method**: 'basic', 'facebook', 'google', 'weibo'
- **signup_flow**: 0, 25, 12, 3, 2, 23, 24, 1, 8, 6, 21, 5, 20, 16, 15, 14, 10, 4 (categorical)
- **language**: 'en', 'zh', 'fr', 'es', 'ko', 'de', 'it', 'ru', 'ja', 'pt', 'sv', 'nl', 'tr', 'da', 'pl', 'no', 'cs', 'el', 'th', 'hu', 'id', 'fi', 'ca', 'is', 'hr', nan
- **affiliate_channel**: 'direct', 'sem-brand', 'sem-non-brand', 'seo', 'other', 'api', 'content', 'remarketing'
- **affiliate_provider**: 'direct', 'google', 'other', 'facebook', 'bing', 'craigslist', 'padmapper', 'vast', 'yahoo', 'facebook-open-graph', 'gsp', 'meetup', 'email-marketing', 'naver', 'baidu', 'yandex', 'wayn', 'daum'
- **first_affiliate_tracked**: nan, 'linked', 'omg', 'tracked-other', 'product', 'marketing', 'local ops'
- **signup_app**: 'Web', 'iOS', 'Android', 'Moweb'
- **first_device_type**: 'Mac Desktop', 'Windows Desktop', 'iPhone', 'iPad', 'Other/Unknown', 'Android Phone', 'Android Tablet', 'Desktop (Other)', 'SmartPhone (Other)'
- **first_browser**: 'Chrome', 'Safari', nan, 'Firefox', 'Mobile Safari', 'IE', 'Chrome Mobile', 'Android Browser', 'AOL Explorer', 'Opera', 'Silk', 'IE Mobile', 'BlackBerry Browser', 'Chromium', 'Mobile Firefox', 'Maxthon', 'Apple Mail', 'Sogou Explorer', 'SiteKiosk', 'RockMelt', 'Iron', 'Yandex.Browser', 'IceWeasel', 'Pale Moon', 'CometBird', 'SeaMonkey', 'Camino', 'TenFourFox', 'Opera Mini', 'wOSBrowser', 'CoolNovo', 'Avant Browser', 'Opera Mobile', 'Mozilla', 'SlimBrowser', 'Comodo Dragon', 'OmniWeb', 'Crazy Browser', 'TheWorld Browser', 'Flock', 'PS Vita browser', 'IceDragon', 'Googlebot', 'Conkeror', 'Arora', 'UC Browser', 'Stainless', 'Google Earth', 'IBrowse', 'NetNewsWire', 'Kindle Browser', 'Outlook 2007', 'Nintendo Browser', 'Palm Pre web browser', 'Epic'
- **dac_day**: 0, 1, 2, 3, 4, 5, 6 (categorical)
- **dac_week**: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53 (categorical)
- **dac_month**: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 (categorical)
- **dac_quarter**: 1, 2, 3, 4 (categorical)
- **dac_year**: 2010, 2011, 2012, 2013, 2014 (categorical)
- **tfa_day**: 0, 1, 2, 3, 4, 5, 6 (categorical)
- **tfa_week**: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53 (categorical)
- **tfa_month**: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 (categorical)
- **tfa_quarter**: 1, 2, 3, 4 (categorical)
- **tfa_year**: 2009, 2010, 2011, 2012, 2013, 2014 (categorical)
- **tfa_hour**: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23(categorical)

- **tfa_tod**: ’afternoon’, ’evening’, ’morning’, ’night’(categorical)
- **tfa_season**: ’autumn’, ’spring’, ’summer’, ’winter’(categorical)

As a final step of the data transformation we one hot encode all categorical features except age. In order to make this transformation, we take one column with x categories and convert it into x columns, where each column represents one category in the original column. As the result, our dataset contains 347 features. After the transformation, we split the data back to training and test set and create numeric label for each of the 12 target labels. Now we are ready for the simulations.

Dataset after feature engineering: `Out[74]:`



	0	1	2	3	4
id	gxn3p5htnn	820tgsjxq7	4ft3gnwmtx	bjjt8pjhuk	87mebub9p4
gender	missing	MALE	FEMALE	FEMALE	missing
age	-1	38	56	42	41
signup_method	facebook	facebook	basic	facebook	basic
signup_flow	0	0	3	0	0
language	en	en	en	en	en
affiliate_channel	direct	seo	direct	direct	direct
affiliate_provider	direct	google	direct	direct	direct
first_affiliate_tracked	missing	missing	missing	missing	missing
signup_app	Web	Web	Web	Web	Web
first_device_type	Mac Desktop	Mac Desktop	Windows Desktop	Mac Desktop	Mac Desktop
first_browser	Chrome	Chrome	IE	Firefox	Chrome
dac_day	0	2	1	0	1
dac_week	26	21	39	49	37
dac_month	6	5	9	12	9
dac_quarter	2	2	3	4	3
dac_year	2010	2011	2010	2011	2010
tfa_day	3	5	1	5	1
tfa_week	12	21	24	44	50
tfa_month	3	5	6	10	12
tfa_quarter	1	2	2	4	4
tfa_year	2009	2009	2009	2009	2009
tfa_hour	4	17	23	6	6
tfa_tod	night	afternoon	evening	night	night
tfa_season	spring	spring	summer	autumn	winter

```
Dataset after OHE contains 275547 rows and 348 columns.
```

Implementation

This project is implemented in Python using the following libraries: - Numpy

- Pandas
- Matplotlib
- Seaborn
- Scikit-Learn
- XGBoost
- time, datetime

We first train 4 models with default parameters: AdaBoost, Logistic regression, LinearSVM, and XGboost. After the fitting procedure we find the probabilities of prediction of each label classe using **predict_proba** method on the training set. We obtain 5 the most probable ones for each user. We then compare the results using a custom NDCG scorer from kaggle scripts [11]. Then we find the most probable prediction and prepare a classification report showing how well our classifer works for this prediction. Finaly, we make a prediction **predict_proba** method on the test set and prepare a data file for submission to kaggle.

There is, however, a complication to the project. It appears that LinearSVM does not predict probabilities. At the same time SVM needs a lot of time to preform the calculation. Therefore, we wrap LinearSVM in calibrated calssifier, which gives us probabilistic classifier. Morever, among the chosen methods only logistic regression returns well calibrated probability predictions by default as it directly optimizes log-loss [5]. However, the tests with calibrated classifiers showed that calibration actually gives worse results than the original classifier. Therefore, we will calibrate the linear SVM classifier only, while leaving the other classifiers uncalibrated.

Here are the results of our calculations:

```
AdaBoostClassifier :

training time (min): 0.5562
prediction time (sec): 7.188
NDCG score: 0.821567014409
             precision    recall  f1-score   support

          0       0.00      0.00      0.00       539
          1       0.00      0.00      0.00      1428
          2       0.00      0.00      0.00      1061
          3       0.00      0.00      0.00      2249
          4       0.00      0.00      0.00      5023
          5       0.00      0.00      0.00      2324
          6       0.00      0.00      0.00      2835
          7       0.67      0.87      0.75    124543
          8       0.00      0.00      0.00       762
          9       0.00      0.00      0.00       217
         10       0.48      0.40      0.44     62376
         11       0.00      0.00      0.00     10094

avg / total       0.53      0.62      0.57    213451


C:\Users\Vladimir\Anaconda2\lib\site-packages\sklearn\metrics\classification.py:1113: Undefined
  'precision', 'predicted', average, warn_for)


LogisticRegression :

training time (min): 2.16046666667
prediction time (sec): 0.834
NDCG score: 0.818020364078
             precision    recall  f1-score   support

          0       0.00      0.00      0.00       539
          1       0.00      0.00      0.00      1428
          2       0.00      0.00      0.00      1061
          3       0.00      0.00      0.00      2249
          4       0.00      0.00      0.00      5023
          5       0.00      0.00      0.00      2324
          6       0.00      0.00      0.00      2835
          7       0.65      0.88      0.75    124543
          8       0.00      0.00      0.00       762
          9       0.00      0.00      0.00       217
         10       0.47      0.34      0.39     62376
         11       0.00      0.00      0.00     10094

avg / total       0.52      0.61      0.55    213451



CalibratedClassifierCV :

training time (min): 26.1393333333
prediction time (sec): 2.46
NDCG score: 0.806268535282
             precision    recall  f1-score   support

          0       0.00      0.00      0.00       539
          1       0.00      0.00      0.00      1428
          2       0.00      0.00      0.00      1061
          3       0.00      0.00      0.00      2249
          4       0.00      0.00      0.00      5023
          5       0.00      0.00      0.00      2324
          6       1.00      0.00      0.00      2835
          7       0.59      0.98      0.74    124543
          8       0.00      0.00      0.00       762
          9       0.00      0.00      0.00       217
         10       0.48      0.05      0.09     62376
         11       0.00      0.00      0.00     10094

avg / total       0.50      0.59      0.46    213451



C:\Users\Vladimir\Anaconda2\lib\site-packages\sklearn\metrics\classification.py:1113: Undefined
  'precision', 'predicted', average, warn_for)


XGBClassifier :

training time (min): 5.66936666667
prediction time (sec): 3.646
NDCG score: 0.826659971648
             precision    recall  f1-score   support

          0       0.00      0.00      0.00       539
          1       0.00      0.00      0.00      1428
          2       0.00      0.00      0.00      1061
          3       0.00      0.00      0.00      2249
          4       0.00      0.00      0.00      5023
          5       0.00      0.00      0.00      2324
          6       0.00      0.00      0.00      2835
          7       0.69      0.85      0.76    124543
          8       0.00      0.00      0.00       762
          9       0.00      0.00      0.00       217
         10       0.49      0.47      0.48     62376
         11       0.00      0.00      0.00     10094

avg / total       0.55      0.63      0.59    213451
```

As it is clearly seen from the classification reports, all the methods capture more or less well two most frequent classes at least for the most probable class. The NDCG score and scores from Kaggle are summarized in the following table:

Model	NDCG score	private score	public score
AdaBoost	0.82157	0.86427	0.86078
Logistic Regression	0.81802	0.86027	0.85711

Model	NDCG score	private score	public score
Callibrated Linear SVM	0.80627	0.85665	0.85351
XGBoost	0.82666	0.87017	0.86537

The best scores arew given by XGBoost. We will use it futher as the main classifier.

Refinement

In this section we will make some adjustments to our model. In order to improve the model, we utilize the sessions data by grouping users by **user_id** and **action_detail** and counting each unique **action_detail** event per **user_id**. This is then joined with our user data. All missing values are replaced with 0. This gives better results:

Model	NDCG score	private score	public score
XGBoost (no sessions)	0.82666	0.87017	0.86537
XGboost (sessions)	0.83225	0.88249	0.87796

```
XGBClassifier :

training time (min): 7.97806666667
prediction time (sec): 5.413
NDCG score: 0.832246467537
             precision    recall  f1-score   support

          0       0.00      0.00      0.00       539
          1       0.00      0.00      0.00      1428
          2       0.00      0.00      0.00      1061
          3       0.00      0.00      0.00      2249
          4       0.86      0.00      0.00      5023
          5       0.00      0.00      0.00      2324
          6       1.00      0.00      0.00      2835
          7       0.70      0.86      0.77    124543
          8       0.00      0.00      0.00       762
          9       0.00      0.00      0.00       217
         10       0.51      0.50      0.50     62376
         11       0.25      0.00      0.00     10094

avg / total       0.61      0.65      0.60    213451
```

Let us adjust the classifer parameters. One of the methods to solve the hyperparameter optimization problem is grid search. This algorithm searches through all possible hyper-parameter configurations, evaluates the error of each configuration, then returns the best configuration. The exhaustive search guarantee the best model configuration is returned.

We adjust the following parameters using the grid search technique: 'max_depth' : [3,6,9], 'learning_rate': [0.05, 0.1, 0.25], 'n_estimators': [50, 100, 200]. Suprisingly the results of the grid search showed that the best score can be obtained with minimal number of 'max_depth' and 'n_estimators':

mean: 0.68473, std: 0.04721, params: {'n_estimators': 50, 'learning_rate': 0.05, 'max_depth': 3}.

Here are the results of grid search:

```
NDCG score: 0.829469688931
             precision    recall  f1-score   support

          0       0.00      0.00      0.00       539
          1       1.00      0.00      0.00      1428
          2       0.00      0.00      0.00      1061
          3       0.00      0.00      0.00      2249
          4       0.00      0.00      0.00      5023
          5       0.00      0.00      0.00      2324
          6       1.00      0.00      0.00      2835
          7       0.71      0.85      0.77    124543
          8       0.00      0.00      0.00       762
          9       0.00      0.00      0.00       217
         10       0.50      0.52      0.51     62376
         11       0.00      0.00      0.00     10094

avg / total       0.58      0.64      0.60    213451


C:\Users\Vladimir\Anaconda2\lib\site-packages\sklearn\metrics\classification.py:1113: Undefined
  'precision', 'predicted', average, warn_for)
```

However, the scores in this case are slightly lower:

Model	NDCG score	private score	public score
XGBoost (no sessions)	0.82666	0.87017	0.86537
XGboost (sessions)	0.83225	0.88249	0.87796
XGboost (grid search)	0.82947	0.88048	0.87599

Also in both latter cases precision and recall for the most probable prediction are slightly better then in previous cases, but recall and f1-score are still 0 for every category except two most frequent ones.

Therefore, the best classifier among the examined ones is XGBoost with sessions data included. The results obtained with this classifer would put us in top 25% on Kaggle.

IV. Results

Model Evaluation and Validation

In this project XGboost showed itself as a fast, reliable, and prone to overfitting method. Including the sessions data helped us to improve the score even more. Here are the result scores of our simulations ordered from low to high:

Model	NDCG score	private score	public score
Callibrated Linear SVM	0.80627	0.85665	0.85351
Logistic Regression	0.81802	0.86027	0.85711
AdaBoost	0.82157	0.86427	0.86078
XGBoost (no sessions)	0.82666	0.87017	0.86537
XGboost (grid search)	0.82947	0.88048	0.87599
XGboost (sessions)	0.83225	0.88249	0.87796

The classifier showed good performance even though we used default model parameters. This classifier generalizes well to unseen data. It gave good results on both initial dataset and after adding sessions data. In order to do the sensitivity analysis we add random noise to the age feature in the training set: ′age′±5%. We also delete the data of 5% users chosen randomly. The results are the following:

```
XGBClassifier :

training time (min): 7.69171666667
prediction time (sec): 3.919
NDCG score: 0.83237573164
             precision    recall  f1-score   support

          0       0.00      0.00      0.00       510
          1       1.00      0.00      0.00      1344
          2       0.00      0.00      0.00      1010
          3       0.00      0.00      0.00      2144
          4       0.80      0.00      0.00      4776
          5       0.00      0.00      0.00      2224
          6       1.00      0.00      0.00      2698
          7       0.70      0.86      0.78    118266
          8       0.00      0.00      0.00       720
          9       0.00      0.00      0.00       208
         10       0.51      0.50      0.50     59283
         11       0.33      0.00      0.00      9593

avg / total       0.61      0.65      0.60    202776


C:\Users\Vladimir\Anaconda2\lib\site-packages\sklearn\metrics\classification.py:1113: Undefined
  'precision', 'predicted', average, warn_for)
```

Model	NDCG score	public score	private score
XGboost (sessions)	0.83225	0.87796	0.88249
XGBoost (sensitivity)	0.83237	0.87793	0.88260

These scores are almost the same meaning that our model is robust and small perturbations in training data do not affect the results.

We believe, therefore, that our model can be trusted since it is robust and it gave high score not only using our custom scorer, but also on Kaggle.

Justification

The final model gave slightly lower NDCG scores comparing to the benchmark:

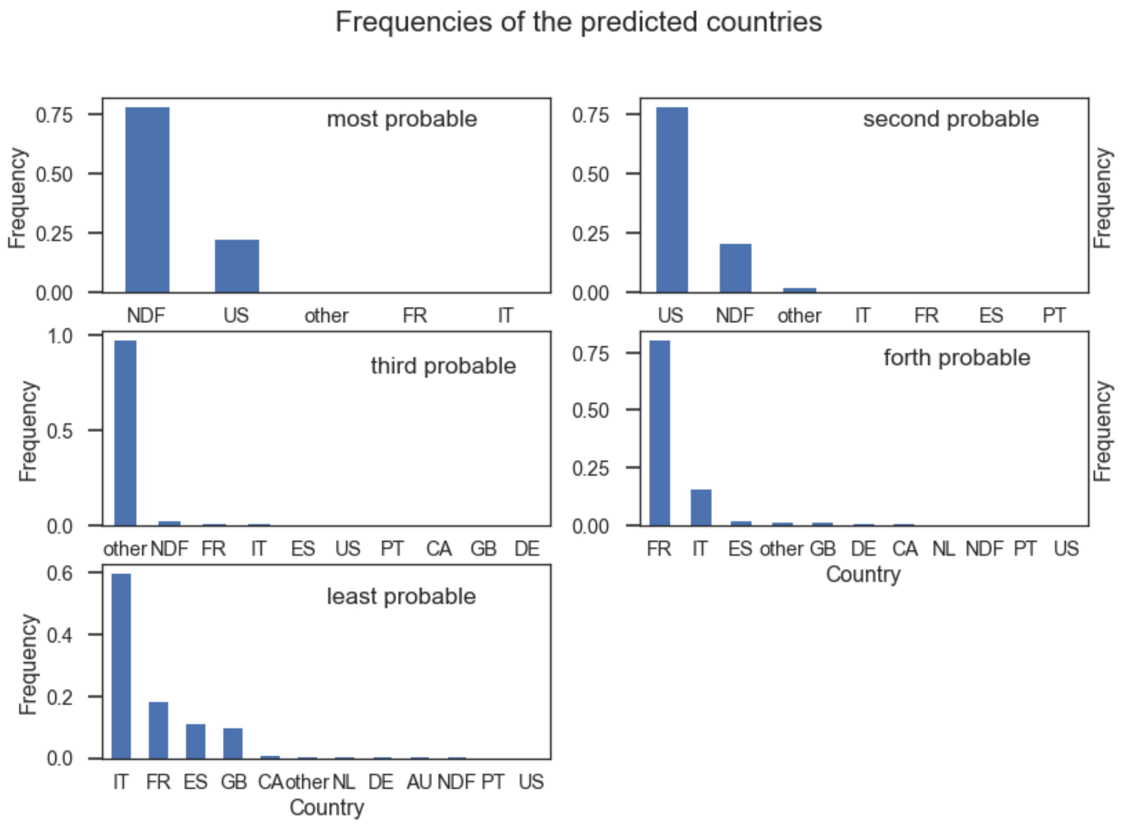
Model	NDCG score	public score	private score
Benchmark		0.8802	0.8850
XGboost (sessions)	0.83225	0.87796	0.88249

Despite that fact, we believe that our model predicts the labels well, since our result falls in top 17.63% of the solutions on Kaggle according to the private leaderboard. Therefore, we believe that the final solution is significant enough to have solved the problem.

V. Conclusion

Free-Form Visualization

Let us visualize frequencies of the predicted countries of different probabilities.



It is clearly seen from the figure, that the most probable countries for a new user to make a new booking at are 'NDF' and 'US'. The third probable country is 'other' which was predicted about 100% of times in this particular probability category. Finally, the least probable countries are 'FR', 'IT', 'ES', and 'GB'. This order exactly matches the order of frequency of the labels in the training set. Therefore, one could speculate that our model simply puts the countries in the order of decreasing frequency. We have tested this hypotesis with the following results:

```
NDCG score for most frequent countries: 0.806765442038
```

Model	NDCG score	public score	private score
Most frequent countries	0.80677	0.85669	0.85359
XGboost (sessions)	0.83225	0.87796	0.88249

Thus, our model does better than just predicting the most frequent destinations and therefore can be trusted.

Reflection

The goal of the project was to predict five the most likely countries for a new user of AirBnB to book an appartment in. While the training data were dated from 2010, the testing and the sessions dataset were from 2014. The harderst for us and probably the most important part of the project was data cleaning and transformation. The dataset contained some missing and unreliable data like age of the order of 2000.

Data cleaning was the top priority in the initial stage of the project. 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 datasets were done.

Then we applied different algorithms such as AdaBoost, Logistic Regression, Linear SVM, and XGBoost. The next step was to apply grid search to the model showed the best performance, which was XGBoost. We combined the trainig set with the sessions data, which resulted in higher NDCG score. However, the results of the grid search showed slightly lower scores.

In order to predict a list of five countries for each user, a probabilistic prediction was applied to the testing dataset for each user according the decreasing order of probability. This was then written to a CSV file
and submitted to Kaggle.

Despite the final model meets our expectations and, we believe, should be used in a general setting to solve these types of problems, there are some improvents possible, which we discuss further.

Improvement

We believe there is a certain number of improvements to do in order to increase the scores obtained by the model.

First of all, there is a number of XGBoost parameters we did not try to adjust with grid search. Some of them are subsample ratio of columns when constructing each tree and for each split, minimum loss reduction required to make a further partition on a leaf node of the tree, minimum sum of instance weight needed in a child, and others.

One could also try different optimization methods such as random search. Another suggestion would be to use the summary statistics of destination countries in the dataset and their locations as well as summary statistics of users age group, gender, country of destination.

The final suggestion would be to try obtaining more data on rare classes in order to improve the quality of their prediction.

VI. References

[1] Bernardo, Francisco, et al. "Interactive Machine Learning for End-User Innovation." American Association for Artificial Intelligence (2016).

[2] Lee, Wenke, et al. Journal of Computer Security **10**, 5 (2002).

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

[4] http://dataaspirant.com/2017/04/15/implement-logistic-regression-model-python-binary-classification/.

[5] http://scikit-learn.org/stable/modules/calibration.html.

[6] https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/.

[7] https://prateekvjoshi.com/2014/05/05/what-is-adaboost/.

[8] P. Harrington, Machine learning in action. Manning publication co. (2012).

[9] Y. Freund, R. E. Schapire, Journal of Japanese Society for Artificial Intelligence **14**, 771 (1999).

[10] https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/.

[11] https://www.kaggle.com/dietcoke/score-predictions-using-ndcg.