# AirBnB New User Bookings, Winner's Interview: 3rd place: Sandro Vega Pons

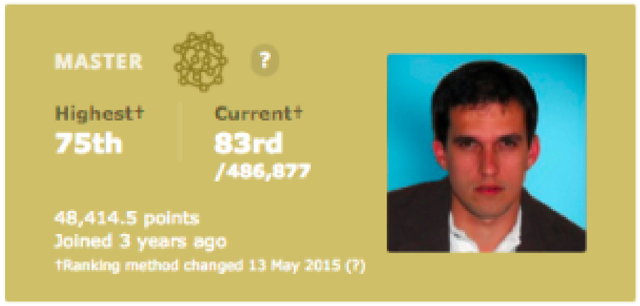
Kaggle Team (http://blog.kaggle.com/author/kaggleteam/) | 03.07.2016

AirBnB New User Bookings (https://www.kaggle.com/c/airbnb-recruiting-new-user-bookings) was a popular recruiting competition that challenged Kagglers to predict the first country where a new user would book travel. This was the first recruiting competition on Kaggle with scripts (https://www.kaggle.com/c/airbnb-recruiting-new-user-bookings/scripts) enabled. AirBnB encouraged participants to prove their chops through their collaboration and code sharing in addition to their final models. Sandro Vega Pons (https://www.kaggle.com/svpons) took 3rd place, ahead of 1,462 other competitors, using an ensemble of GradientBoosting, MLP, a RandomForest, and an ExtraTreesClassifier. In this blog, Sandro explains his approach and shares key scripts that illustrate important aspects of his analysis.

## The Basics

### What was your background prior to entering this challenge?

I currently work as a postdoctoral researcher at the NeuroInformatics Laboratory (https://nilab.fbk.eu/), FBK in Trento, Italy. My current research is mainly focused on the development of machine learning and network theory methods for neuroimaging data analysis. I hold a M.Sc. in Computer Science and a Ph.D. in Applied Mathematics.

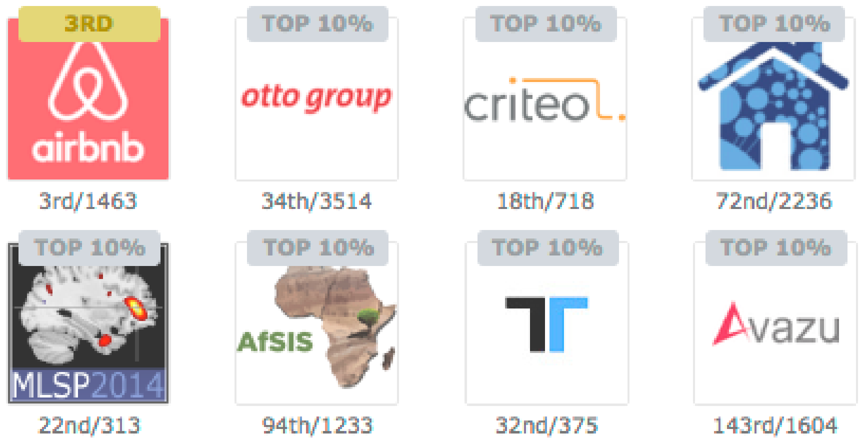


(https://www.kaggle.com/svpons)

SANDRO'S PROFILE ON KAGGLE (HTTPS://WWW.KAGGLE.COM/SVPONS) AND LINKEDIN (HTTPS://WWW.LINKEDIN.COM/IN/SANDROVEGAPONS)

### How did you get started competing on Kaggle?

I first heard about Kaggle around three years ago, when a colleague showed me the website. At that time I was starting to work with scientific and analytic Python packages (numpy, scipy, scikit-learn, etc.) and I found Kaggle competitions a perfect way of getting familiar with them. The first competition in which I seriously participated was Connectomics (https://www.kaggle.com/c/connectomics), which was somewhat related to my research activities in that time. From that moment on, I have tried to be always active in at least one competition, even though many times I cannot dedicate as much time as I would like to the competitions.



(https://www.kaggle.com/svpons)

SANDRO'S TOP 8 FINISHES

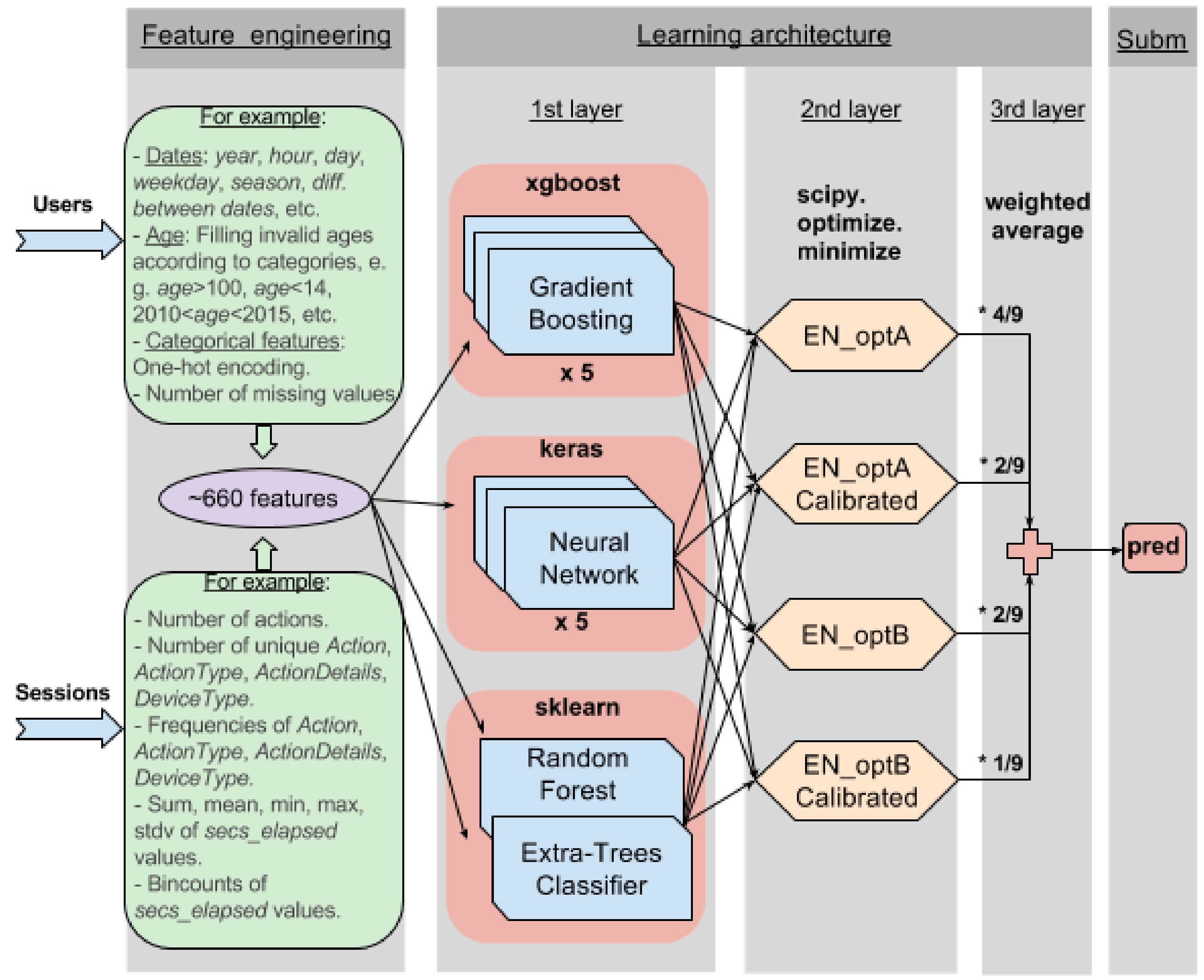
### What made you decide to enter this competition?

After ending up in the top 10% of some competitions, I was still looking for one in which I could finish in the top10, in order to get the master tier. I found this competition very interesting for many reasons, e.g.: there was room for creativity in feature engineering, the problem seemed to be complex enough (12-class classification, highly unbalanced, classes very overlapped, I was not very familiar with the evaluation measure, etc.) and the data was not too big (I could work on my laptop while on Christmas holidays). Then, after trying some initial ideas and seeing that the results were promising, I got completely hooked on this competition.

## Let's Get Technical

### What preprocessing and supervised learning methods did you use?

My solution was basically based on feature engineering in both *users* and *sessions data* and a 3-layers architecture for ensembling classifiers (see Figure 1).


Feature engineering
Learning architecture
Subm
Users
Sessions
For example:
- Dates: year, hour, day, weekday, season, diff. between dates, etc.
- Age: Filling invalid ages according to categories, e.g. age>100, age<14, 2010<age<2015, etc.
- Categorical features: One-hot encoding.
- Number of missing values
~660 features
For example:
- Number of actions.
- Number of unique Action, ActionType, ActionDetails, DeviceType.
- Frequencies of Action, ActionType, ActionDetails, DeviceType.
- Sum, mean, min, max, stdv of secs_elapsed values.
- Bincounts of secs_elapsed values.
1st layer
2nd layer
3rd layer
xgboost
Gradient Boosting
x 5
keras
Neural Network
x 5
sklearn
Random Forest
Extra-Trees Classifier
scipy. optimize. minimize
EN_optA
EN_optA Calibrated
EN_optB
EN_optB Calibrated
weighted average
* 4/9
* 2/9
* 2/9
* 1/9
pred

(http://blog.kaggle.com/wp-content/uploads/2016/03/learning_architecture-fig-1-1.png)

FIG. 1

**Feature Engineering**:

I extracted features from both *users* and *sessions* files. In the case of *users*, I applied a one-hot-encoding representation for categorical features and computed several features from *age* and *dates*, using different techniques for dealing with missing values. The *sessions* data was aggregated by user's id and different features based on counts, frequency, unique values, etc. were computed. The full feature engineering code can be found in this script (https://www.kaggle.com/svpons/airbnb-recruiting-new-user-bookings/feature-engineering).

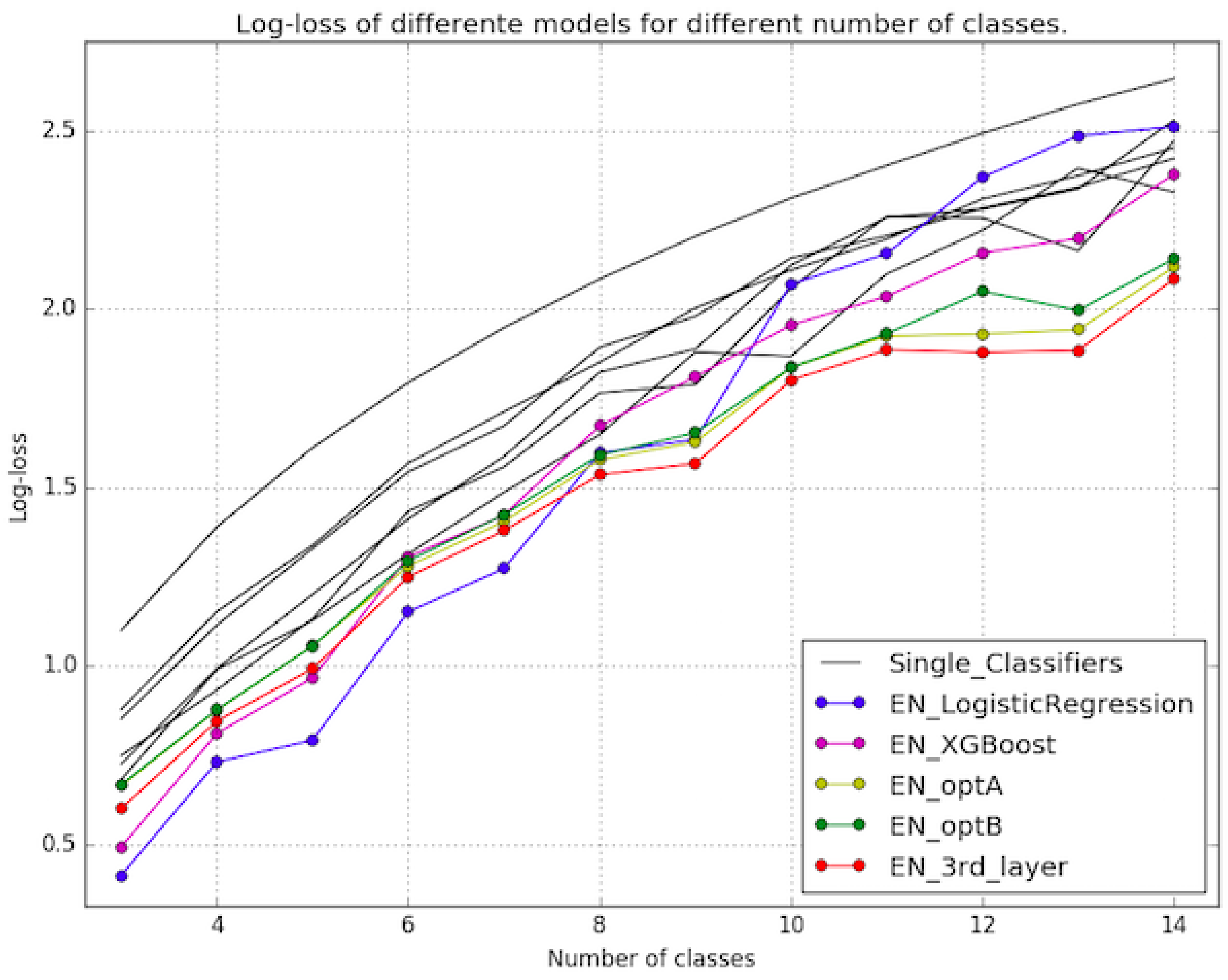
**Learning architecture**:

- **1st Layer**: I tried many classifiers at this level. Since the beginning, I got the best individual results with XGBoost (https://github.com/dmlc/xgboost) GradientBoosting. Later on, I was able to get comparable results with deep MLP implemented on Keras (https://github.com/fchollet/keras). I also used other classifiers (with a bit worse performance) implemented in scikit-learn (http://scikit-learn.org/stable/), like RandomForest, ExtraTreesClassifier and LogisticRegression. My best submission was computed by using 5 versions of GradientBoosting (trained with different parameters, class weights and subset of features), 5 versions of deep MLP (trained with different parameters, class weights and subset of features), a RandomForest and an ExtraTreesClassifier.
- **2nd Layer**: I implemented my own blending strategies based on the scipy.optimization package (http://docs.scipy.org/doc/scipy/reference/tutorial/optimize.html). A detailed description, source code of these ensembling methods, and how they can be used inside a 3-layer learning architecture can be found in this python notebook (https://www.kaggle.com/svpons/airbnb-recruiting-new-user-bookings/three-level-classification-architecture). I implemented two methods, called *EN_optA* and *EN_optB* in the script. I used the two methods and a calibrated version of them (using the CalibratedClassifierCV from scikit-learn) to end up with four ensemble predictions.
- **3rd Layer**: This last layer consists of a weighted average of the 4 predictions obtained in the previous layer.

## What was your most important insight into the data?

- Although the two problems are different, after some feature engineering on the data of this competition, the resulting classification problem, was somehow similar (number of classes, number of features, number of samples) to that of the Otto Group Product Classification Challenge (https://www.kaggle.com/c/otto-group-product-classification-challenge). The algorithms I used for that competition (ending up 34th from 3514 teams) were a good starting point.
- The two proposed ensembling techniques (*EN_optA* and *EN_optB*) minimize an objective function and the best results were obtained when minimizing the multiclass logloss. This means that in practice, the two methods basically became implementations of logistic regression. However, I always got better results with the proposed techniques rather than with the sklearn implementation of LogisticRegression. They also outperformed non linear classifiers like GradientBoosting (using multiclass logloss as objective function). I made a quick experiment in the last part of this notebook (https://www.kaggle.com/svpons/airbnb-recruiting-new-user-bookings/three-level-classification-architecture) in order to explore the behavior of the applied techniques in more detail. The idea was to compare the performance of the different ensembling methods on synthetic data for different number of classes. The results of this experiment (for the limited set of parameters values I explored) show that LogisticRegression (as implemented in sklearn) and GradientBoosting produce

better results for problems with a few number of classes. Nevertheless, as the number of classes increases, the proposed ensembling techniques become the best option.



(http://blog.kaggle.com/wp-content/uploads/2016/03/comp_ens-fig-2-1.png)

FIG. 2

- The further combination of the two proposed ensembling techniques and their calibrated versions, gave me a significant final boost. This means that they produce comparable results, but different enough to blend well. This was also verified in the experiment with synthetic data (Figure 2 above), where *EN_3rd_layer* always outperformed *EN_optA* and *EN_optB*.

## Were you surprised by any of your findings?

- I was happily surprised of being able to create deep MLP models producing results that were comparable to the best XGBoost ones. Moreover, I am always a bit surprised about how well deep learning models (like deep MLP) and tree-based models (like GradientBoosting) complement each other, and therefore how good they perform when ensembled. The combination of these two types of learning algorithms seems to be a recurrent choice in Kaggle competitions. This perhaps requires a more systematic study, beyond intuitive explanations based on the general understanding of the two types of models.
- I always got the best results when using multi-class log-loss as objective function. Specifically, for the the definition of *EN_optA* and *EN_optB* (second layer), I tried with different objective functions like *mlogloss*, *mlogloss@5* (multi-class log-loss after keeping only the 5 classes with highest probabilities), ndcg@5 (the competition evaluation metric) and *mlogloss + 2 - ndcg@5* (a combination of *mlogloss* and *ndcg* after converting *ndcg* to a dissimilarity measure). I was expecting the *ndcg* not to work well, since it is a non-smooth function and therefore not suitable for gradient based optimization. However, I was not sure whether some combination of *mlogloss* and *ndcg* could give any improvement.

## Which tools did you use?

Python (numpy (http://www.numpy.org/), pandas (http://pandas.pydata.org/), scipy (http://www.scipy.org/), xgboost (https://github.com/dmlc/xgboost), keras (https://github.com/fchollet/keras), scikit-learn (http://scikit-learn.org/stable/)).

## How did you spend your time on this competition?

At the beginning of the competition I focused on creating a very simple working pipeline that, by only using *users* data, was able to produce a meaningful submission. I made public this pipeline (it is based on simple feature engineering and XGBoost) on this script (https://www.kaggle.com/svpons/airbnb-recruiting-new-user-bookings/script-0-8655). After that, I focused on feature engineering on both *users* and *sessions* data and updated my model with the features computed from session data. I was then inactive for more than one month, but when I got back to the competition, I started working on other classifiers. The idea was to tune some classifiers that would potentially blend well with my best XGBoost model. During the final days I focused on ensembling. I first started with a more traditional stacked generalization approach based on LogisticRegression and GradientBoosting, but as I didn't get very good results, I moved on to building my own ensemble strategies.

## Words of Wisdom

## What have you taken away from this competition?

- The importance of keeping an eye on the competition forum and contributing to it (I didn't do much of that before this competition) when possible. Something similar can be said about competition scripts, they can be useful to others, but you can also get valuable feedback.
- Even though my NN models were very simple, I spent quite a bit of time with tools like Theano (https://github.com/Theano/Theano) and Keras. This gave me the opportunity to understand many concepts of deep learning that were not familiar to me.
- The importance of spending time in understanding the evaluation metric and how to optimize it. This competition is an example of why directly using an evaluation metric as objective function is not always the optimal solution.

## Do you have any advice for those just getting started in data science competitions?

- Plan the time that will be spent on the competition. Do not focus on being in the top of the leaderboard from the beginning, make a plan and work accordingly. For example, do not start with complicated ensemble techniques before squeezing out single classifiers.
- Try many ideas, but be careful with the reproducibility of the code (e.g. take care of seeds for random number generators).
- Do not move forward to complex ideas before having a deep understanding of the evaluation measure and a reliable validation strategy. Depending on the competition, this can be trivial or very tricky.

AIRBNB NEW USER BOOKINGS (HTTP://BLOG.KAGGLE.COM/TAG/AIRBNB-NEW-USER-BOOKINGS/)

RECOMMENDATION PROBLEM (HTTP://BLOG.KAGGLE.COM/TAG/RECOMMENDATION-PROBLEM/)