# Classified: TalkingData

*Classified is Jordi Beernink, Thijs Werrij, Roel Bouman, Gerdriaan Mulder, and Jeffrey Luppes*

Radboud University

## 1 Introduction

This report details the work done on the TalkingData Competition data set by the Classified team. Although the competition closed in September of 2016, the data set remained public and open to submissions, but the leaderboard was frozen as the competition closed.

**Problem Description** The goal of the competition is to predict mobile behaviour based upon app usage, phone brand and location information. As an abstraction upon this, the goal is to accurately predict to which gender and age group the owner of a mobile device belongs. The predictions are accepted as twelve categories: six age groups for each gender. Submissions to the competition are evaluated and ranked using the multi-class logarithmic loss metric.

**Data set** The data set is hosted on Kaggle and consists of eight csv files totalling 1.2 GB. In order to generate a single data set these files were merged based on the device ID, which generated a data set of 4.2 GB. The data was fairly sparse, as many devices only had a few associated events, while others had up to thousands. This was true also for the phone brand types with some phones appearing only a few times while others had a much larger market share.

## 2 Approach

**Pre-Processing** The pre-processing step was arguably the most important phase in this competition. After merging the csv files, the data was still incredibly sparse: over two-thirds of all devices had no events linked to them. Prediction was prioritized based on the device properties and the installed applications.

**Feature Extraction** All features present in the data set were aggregated into feature vectors based on the age/gender group. Additionally, some features were created by looking at what day of the week certain something was used. Using a clever One-hot-encoding, sparse matrices, and the pickle library present in Python.

### 2.1 Classifiers

Initially a single pipeline was created using Random Forests, before being extended with other classification algorithms. As nearly all implementation followed the Scikit-learn API, the pipeline was easily extendable.

**Random Forests** To establish a baseline, the Random Forest Classifier from the Scikit-learn package was used. As initial testing with Random Forests yielded a score below that of the benchmark submission, further analysis using Random Forests was abandoned early on.

**Logistic Regression** The Logistic Regression classifier from the Scikit-learn package in Python was used for classification. This provided quite decent results, especially when compared to other submissions on Kaggle. The best working solver for the large sparse data set was Limited-memory BFGS.

**XGBoost** Extreme Gradient Boosting [1] was a more novel method used in classification. While computational intense, and harder to use, it was well suited for the problem.

**Deep Learning/Neural networks** With the use of Keras, a Neural Network library with a Tensorflow back-end, a neural network model was made. The network consists of several dense layers, dropout layers, and layers that use the well known Parametric Rectified Linear Unit (PReLU) [2].

## 3 Results

The results of the classifiers are displayed in the following table.

| Classifier | Mult-class logloss | Kaggle rank (private leaderboard) |
|---|---|---|
| Benchmark submission | 2.48491 | 1557 |
| Random Forests | 3.261 | 1667 |
| XGBoost tree | 2.280 | 996 |
| XGBoost Linear | 2.269 | 776 |
| Logistic Regression | 2.265 | 666 |
| Neural Network (Keras) | 2.251 | 645 |

## 4 Discussion

The neural network outperformed the other classifiers by a small margin, but the relatively simple logistic regression model did come extremely close. This is still better than over 60% of the submissions.

## 5 Other ideas

The main focus point for improvement seems to be the pre-processing and the selection of features. A large amount of optimization of the classification was done, without impacting the score much.

A number of concepts for features were considered but not explored. Perhaps classification can be improved by mapping users to urban areas or specific cities. Additionally, many more features could have been involved that involve distance, density or time. Sadly, the contest did not allow for data enrichment with data outside of the competition.

## 6 Individual Contributions

By alphanumeric ordering:

- Jordi Beernink: *Project leader, Pre-processing, XGBoost implementation*
- Roel Bouman: *Classification Pipeline, Random Forests and Logistic Regression*
- Jeffrey Luppes: *XGBoost implementation, Report writing*
- Gerdriaan Mulder: *Pre-processing, installation on the lilo servers*
- Thijs Werrij - *Deep Learning*

### 6.1 Github Repository

All code accompanying this report may be retrieved from `https://github.com/RoelBouman/Classified/tree/first_competition` where it has been bundled as a release. The README on that page also has extensive material on how to run the code on the lilo.science.ru.nl servers where it has been prepared for a demo.

# Bibliography

[1] Chen Tianqi *: TalkingData - Linear Model on Apps and Labels*, https://www.quora.com/What-is-the-difference-between-the-R-gbm-gradient-boosting-machine-and-xgboost-extreme-gradient-boosting, Quora, 15th September 2015.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun *: Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*, http://arxiv.org/abs/1502.01852, CoRR, 2nd March 2015.