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. While the competition closed in September of 2016, the data set remained public and open to submissions though the leaderboard was frozen.

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 in which gender and age group the owner of a mobile device resides. 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 algorithm.

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. This data was fairly sparse, as many devices only had a few associated events, while others ranged in the thousands. This was true also for the phone brand types with some phones appearing only a few times while others had quite the market share.

2 Approach

Pre-Processing The pre-processing step was the most tedious phase. After merging the .csv files together the data was still incredibly sparse: over 2/3rds of all devices had no events linked to them. Prediction priority was given to 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 using Random Forests was created, before being extended with other classification algorithms.

Random Forests To establish a baseline for a Random Forest Classifier from the Scikit-learn package was implemented, as it was presumed that it could serve as a starting point for more complex models. After scoring even worse than the benchmark submission of each class having an equal percentage it was quickly made clear that the sparse nature of the data was something that worked against the trees.

Logistic Regression Unexpectedly good performance came from the Logistic Regression classifier. The best working solver for the large sparse data set was Limited-memory BFGS.

XGBoost Extreme Gradiant Boosting [1] was used to much better effect than Random Forests. While computational intense, and harder to implement well, it was well suited for the problem.

Deep Learning/Neural networks With the use of Keras, a library for Theano and Tensorflow, a neural network model was made. The network consists of several dense layers, dropout layers, and layers that use 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 well, but the relative simple logistic regression model did come extremely close. This is still better than over 60% of the submissions.

5 Other ideas

The main vector for improvement seems to be the pre-processing and the selection of features because a large amount of work was already done with optimizing the pipeline without significantly impacting the score in a positive way.

A number of concepts were considered but not explored. A lot information could have come from mapping users to urban areas or specific cities. Additionally, many more features could have been involved that involve distance, density or time. It was not allowed to enrich the data with outside data.

6 Individual Contributions

By alphanummeric ordering:

- Jordi Beernink: *Project leader, Pre-processing, XGBoost implementation*
- Roel Bouman: *Classification Pipeline and RF implementation*
- 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 alsohas 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.