Bike sharing model reuse framework for tree-based ensembles

Gergo Barta[1]

Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics, Magyar tudosok krt. 2. H-1117 Budapest, Hungary
barta@tmit.bme.hu
http://www.tmit.bme.hu

Abstract. The following paper describes the approach of team Dmlab in the 2015 ECML-PKDD Challenge on "Model Reuse with Bike rental Station data" using tree ensemble algorithms and adaptive training set construction.

Keywords: time series, forecasting, gradient boosting regression trees, ensemble models, bike sharing, competition, ECML-PKDD

1 Introduction

The ECML-PKDD Challenge on "Model Reuse with Bike rental Station data" is a novel competition announced in 2015. The proposed competition put focus on investigating model reuse potential in a bike sharing system. The challenge offered a unique approach to forecasting public bicycle availability, since competition participants needed to generate forecasts for previously unseen stations. This methodological difference promotes resourcefulness and offers a vast space for creative and even wild research ideas.

The report contains five sections:

1. Methods show the underlying models in detail with references.
2. Data description provides some statistics and description about the target variables and the features used throughout the competition.
3. Experiment Methodology summarizes the training and testing environment and evaluation scheme the challenge was conducted on.
4. Modeling details are presented in a the corresponding section.
5. Conclusions are drawn at the end.

2 Methods

Previous experience showed us that oftentimes multiple regressors are better than one[1]. Therefore I used an ensemble method that was successful in various other competitions: Gradient Boosted Regression Trees (GBR)[2–4] combined with Random Forests (RF). Experimental results were benchmarked and later extended using Ordinary Least Squares regression; a model widely used for time series regression. GBR and RF implementation was provided by Python's Scikit-learn[5] and OLS by the Statsmodels [6] library.

2.1 Random Forest

The Random forest is perhaps the best-known of ensemble methods, thus it combines simple models called base learners for increased performance. In this case multiple tree models are used to create a forest as introduced by Leo Breiman in 2001.

There are three key factors of forest creation:

1. bootstrapping the dataset
2. growing unpruned trees
3. limiting the candidate features at each split

These steps ensure that reasonably different trees are grown in each turn of iteration, which is key to effective model combination.

The bootstrapping step of the model creation carries out a random sampling of a dataset with N observations with replacement that results in N rows, but only ca. 63% of the data used. The probability that an observation x does not get into the sample S equals

$$P(x \notin S) = (1 - \frac{1}{n}) \approx e^{-1} = 0.368 \tag{1}$$

Pruning the trees would reduce variance between trees and thus considered inessential as the overfitting of individual trees is balanced anyway by the ensemble.

When growing trees a different set of features are proposed as candidates in finding the best split based on an information criteria like gini or enthropy. The subset of features are selected randomly further increasing the variance between trees.

The output of the trees is then combined by averaging the results based on some weights or by performing a majority vote in case of classification problems.

Random forests have very few vital parameters to tune, they are effectively non-parametric. The unique architecture provides many benefits and is widely recognized as a good initial approach to most problems. Unlike decision trees, the ensemble method's averaging property inherently finds a balance between high variance and bias. It is insensitive to many data related issues such as the large number and heterogeneity of features, outliers, missing data, and even an unbalanced target. Other than being a great out-of-the-box tool it offers various useful services. Random forest gives intrinsic evaluation of the results based on the data discarded by bootstrapping (called out-of-bag error), it also gives estimates what variables are important.

ExtraTrees is a slightly different Random forest variant suggested by Pierre Geurts, Damien Ernst and Louis Wehenkel in the article Extremely randomized trees in 2006. The extreme randomization comes from the fact that the variable splitting in each node is no longer based on finding the best split, but done in a completely random manner. This causes the trees grown to be even less data dependent, thus introducing extra variance between them.

2.2 Gradient Boosting Decision Trees

Gradient boosting is another ensemble method responsible for combining weak learners for higher model accuracy, as suggested by Friedman in 2000 [7]. The predictor generated in gradient boosting is a linear combination of weak learners, again we use tree models for this purpose. We iteratively build a sequence of models, and our final predictor will be the weighted average of these predictors. Boosting generally results in an additive prediction function:

$$f^*(X) = \beta_0 + f_1(X_1) + \ldots + f_p(X_p) \tag{2}$$

In each turn of the iteration the ensemble calculates two set of weights:

1. one for the current tree in the ensemble
2. one for each observation in the training dataset

The rows in the training set are iteratively reweighted by upweighting previously misclassified observations.

The general idea is to compute a sequence of simple trees, where each successive tree is built for the prediction residuals of the preceding tree. Each new base-learner is chosen to be maximally correlated with the negative gradient of the loss function, associated with the whole ensemble. This way the subsequent stages will work harder on fitting these examples and the resulting predictor is a linear combination of weak learners.

Utilizing boosting has many beneficial properties; various risk functions are applicable, intrinsic variable selection is carried out, also resolves multicollinearity issues, and works well with large number of features without overfitting.

3 Data description

The original competition goal was to predict hourly bike availability for a number of docking stations for a given hour on a given day. The provided dataset contained information about bike availability on hourly resolution for a roughly 2.5 year long period between 2012 and 2014 for 10 docking stations in the city of Valencia. In addition, 1 month worth of partial training data was provided for 190 other stations throughout the city. The evaluation in the intial phase was carried out on 25 previously unseen stations, leaving the last 50 stations for the final evaluation. Also, no additional data sources were allowed to be used for this competition.

3.1 Data preparation

Each station in the dataset has two very unique features originating from the bike sharing system; it's location given by GPS coordinates (*latitude*, *longitude*) and bicycle capacity (numDocks). In order to maintain model re-usability all capacity-related features were normalized by the *numDocks* attribute. These features include $bikes$, $bikes\_3h\_ago$, $full\_profile\_3h\_diff\_bikes$, $full\_profile\_bikes$,

$short\_profile\_3h\_diff\_bikes$ and $short\_profile\_bikes$. This transforms capacity-related attributes to take values from $[0, 1]$ range and capacity differences from $[-1, 1]$.

Throughout the datasets missing values appear for unknown reasons (eg. system malfunction, blackout, rolling window of different models) spanning various length. All rows with missing target label were discarded. Input variables were imputed simply with zero for $precipitation.l.m2$, $full\_profile\_3h\_diff\_bikes$, $short\_profile\_3h\_diff\_bikes$ or the mean of the two closest known values for $bikes\_3h\_ago$, $windMaxSpeed.m.s$, $windMeanSpeed.m.s$, $windDirection.grades$, $temperature.C$, $relHumidity.HR$ and $airPressure.mb$.

The feature engineering work created 9 new input attributes to enhance model performance. There were 4 new variables added containing baseline model outputs as suggested by the competition website; $base\_ago\_sprofdiff$, $base\_ago\_fprofdiff$, $base\_ago\_sprofdiff\_sprof$ and $base\_ago\_fprofdiff\_sprof$. This step essentially implements model stacking of the baseline outputs and machine learning methods. 5 basic date-related attributes were also extracted including month of year, day of week (in numerical format), day of year, day of month and week of year.

Constructing the model training set involved various tricks. The original deployment station data contained the month of October only which results on a weak modelling performance on the test data of November and December (especially with the holiday season around). It is highly desirable to have at least a full year's worth of training data at hand. Incidentally this is only provided for the 10 partial train stations. A Pearson correlation analysis was performed on the previously normalized *bikes* values to find suitable donors of training set extension.

Figure 1 shows interesting correlation patterns regarding bicycle rental. The stations in the top-left corner tend to move together, similarly so the larger block in the bottom-right. Caused by their location or functionality these stations act similarly acting as sources and sinks of bikes during a day of the city. Remember, the data has hourly granularity and most bike trips last less, which makes some of the source-sink pairs directly recognizable due to their very strong negative correlation (dark blues). Figure 2 depicts such a relation between stations 7 and 222 with an obvious weekend pattern in addition.

Table 1. Transforming stations with strong negative correlation

Input variable	**Transformation applied**
bikes	1 - bikes
bikes_3h_ago	1 - bikes_3h_ago
full_profile_bikes	1 - full_profile_bikes
short_profile_bikes	1 - short_profile_bikes
short_profile_3h_diff_bikes	(-1) * short_profile_3h_diff_bikes
full_profile_3h_diff_bikes	(-1) * full_profile_3h_diff_bikes

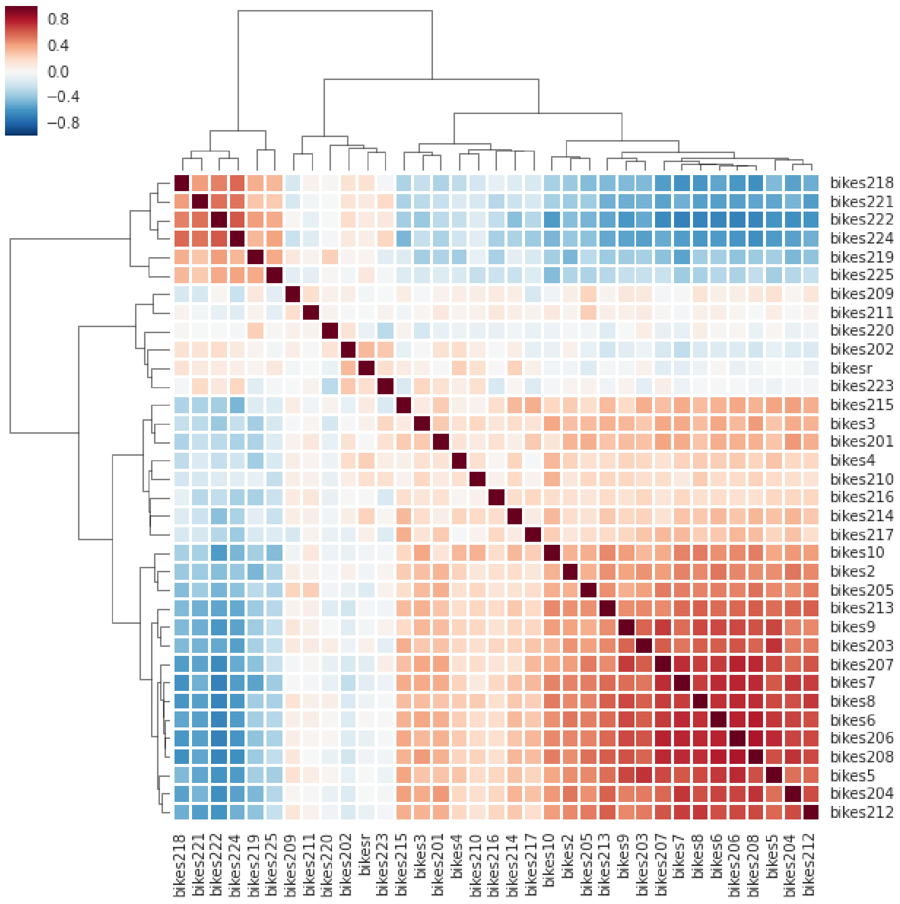

Fig. 1. Pearson correlation of stations 1-10 and 201-225

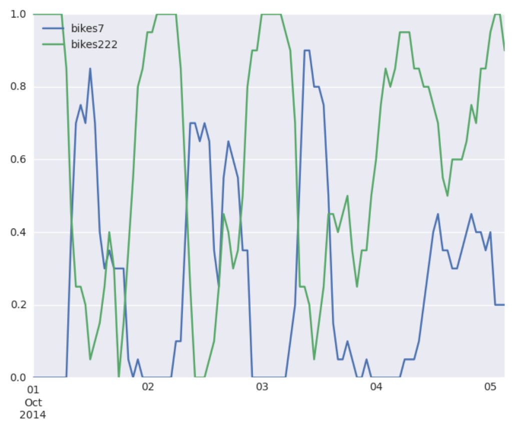

Fig. 2. Strong negative correlation between stations 7 and 222

As an extra twist in the training set construction negatively correlated stations were also included if no suitable addition was found previously. This step required capacity-related attributes to be swapped accordingly (see Table 1 for details). The resulting training set construction is summarized in Table 2, the value 'All' denote the usage of both stations 1-10 and their capacity-swapped negative counterparts. Optimal training set constructs listed in Table 2 were found by taking two aspects into account. Possible training set donors were identified by demonstrating an outstanding Pearson correlation with the target (taking the top decile of all station correlations). An optimal subset was finalized by the test result measurements shown in the following section.

4 Experiment Methodology

I used data from the second half of October 2014 as a validation set in my research methodology (unlike in the test set where specic dates were marked for evaluation for each station). To receive more generalized insights two batch of model evaluation run was performed; the first period starting 2014-10-17 and the second 2014-10-24. The MAE error measurements were averaged over the two.

The test set does not consist of a single continuous time interval, but rather 20 selected time points for each test station. This has two distinct effects, lagged and rolling features are unavailable (and against rules), but it is feasible to create a separate model for each test observation. Due to the training set construction often times many unrelated observations are included from loosely correlated stations. Nearest neighbor methods address this problem efficiently, but they usually offer mediocre performance in forecasting problems, thus a hybrid approach was pursued. To include relevant observations only a similar-

Table 2. Construction of the extended training set for the small test stations 201-225

Test station nr.	**Additional train data from**
201	7
202	1
203	5
204	7,6
205	6,7
206	6,7,8
207	7,9
208	6,7
209	All
210	10
211	None
212	8,7
213	8,6,7,5
214	6,5
215	5,7,6
216	5
217	7
218-225	All

ity check was introduced, implemented by Scikit-learn's KDTree module without limiting the choice of regression models. Distance measures were taken on different sets of attributes; with a backward selection starting with all variables, optimal performance was found when using baseline model outputs only (eg. full_profile_bikes, bikes_3h_ago, full_profile_3h_diff_bikes, short_profile_bikes, short_profile_3h_diff_bikes).

5 Modeling

Bike availability was forecasted using 3 robust and proven machine learning models; Ordinary Least Square Regression, Random Forest and Gradient Boosting Regression Trees. Experiments showed that model ensembles offered even lower error rates, the final model consists of the arithmetic mean of the 2 base model outputs GBR and RF. OLS was discarded from the ensemble due to inconsistent outputs observed. Both RF and GBR models are fairly robust against useless features, meaning almost no prior feature selection is required. OLS regression is very sensitive in this matter, here only the significant attributes are kept (T-stat p_value<0.05), but multicollinearity is still an issue.

All three models are quite robust and offer only a handful tuning parameters. Table 3 summarizes the parameter fine tuning carried out with a grid search approach. Unmentioned parameters were left at default.

Table 3. Grid search parameters and optimums found

Hyperparameter	**Considered values**	**Optimum**
KDTree neighbors	100,200,500,1000,2000,10000	2000
RF no. estimators	200,400,800,1000,2000	1000
GBR no. estimators	100,200,400,500,1000	500
RF maximum depth	5,7,9,None	7
GBR maximum depth	3,4,5,7	4

6 Conclusions and future work

The 2015 ECML-PKDD Challenge offered a novel way of forecasting; model reuse in time series analysis is an interesting open question and an approach worth investigating. My efforts in the contest were focused on complex data preparation and developing a specialized evaluation scheme. As well as providing accurate forecasts with the help of well-established estimators in the literature used in a fairly different context.

The methodology used in this paper can be easily applied in other domains of forecasting as well. Local validation indicated a mean absolute error of 2.476 over stations, while the full test results provided by the contest organizers had a somewhat lower MAE of 2.416.

During the competition I filtered the training set to better represent the characteristics of the day to be forecasted, which greatly improved model performance. Automating this process is also a promising and chief goal of ongoing research.

References

1. L. Rokach, "Ensemble-based classifiers," Artificial Intelligence Review, vol. 33, pp. 1-39, 2010.
2. S. a. S. L. Aggarwal, "Solar energy prediction using linear and non-linear regularization models: A study on AMS (American Meteorological Society) 2013–14 Solar Energy Prediction Contest," Energy, 2014.
3. H. B. e. a. McMahan, "Ad click prediction: a view from the trenches.," Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 1222-1230, 2013.
4. T. e. a. Graepel, "Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine.," Proceedings of the 27th International Conference on Machine Learning, pp. 13-20, 2010.
5. Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." The Journal of Machine Learning Research 12 (2011): 2825-2830.
6. Seabold, Skipper, and Josef Perktold. "Statsmodels: Econometric and statistical modeling with python." Proceedings of the 9th Python in Science Conference. 2010.
7. J. H. Friedman, "Greedy function approximation: a gradient boosting machine," Annals of Statistics, pp. 1189-1232, 2001.