‘More Bikes’: Experiments in Univariate Regression

Tim Lawson

University of Bristol

tim.lawson@bristol.ac.uk

1 Task description

The assignment is to predict the number of available bikes at 75 rental stations in three hours’ time for a period of three months from November 2014, i.e., a supervised univariate regression problem. It is divided into three sub-tasks, which differ in the information that is available. For sub-task 1, the number of available bikes at each of the 75 stations for the month of October 2014 is provided (section 4). This sub-task may be approached by building a separate model for each station or a single model for all 75 stations. For sub-task 2, a set of linear models that were trained on the number of available bikes at each of a separate set of 200 stations for a year are provided (section 5). Finally, for sub-task 3, both sources of information may be used. Additionally, the number of available bikes at each of ten stations between June 2012 and October 2014 is provided for analysis (fig. 3).

The evaluation data was not available to participants, but the score achieved on a held-out test set is reported on the task leaderboard.[1] The predictions are evaluated by the mean absolute error (MAE) between the predicted and true numbers of available bikes over the period of three months, beginning in November 2014. Hereafter, the MAE is referred to as the ‘score’.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{1}$$

This report begins with a description of the general approach taken to the assignment in section 2. Section 3 presents a preliminary analysis of the data. Finally, sections 4 and 5 detail the results of the methods applied to each of the sub-tasks.[2]

2 Approach

The experiments described in this report were performed with the `scikit-learn` Python package (Pedregosa et al. 2011). In each case, preprocessing and feature selection were performed by *estimators* that implemented the *transformer* interface, prediction was performed by estimators that implemented the *predictor* interface, and estimators were composed into `Pipeline` objects over which hyperparameter search was performed (Buitinck et al. 2013, pp. 4–9).

Generally, standard k-fold cross-validation is disfavoured for time-series data due to the inherent correlation between successive folds (Bergmeir et al. 2018). Instead, nested time-series cross-validation[3] with ten folds was used to evaluate the models, which is illustrated in fig. 1. The best estimator for each model class was determined by grid search[4] with the mean absolute error as the scoring function,

[1] `https://www.kaggle.com/c/morebikes2023/leaderboard`

[2] The code that produced these results is available at `tslwn/more-bikes`.

[3] See `sklearn.model_selection.TimeSeriesSplit`.

[4] See `sklearn.model_selection.GridSearchCV` and `sklearn.model_selection.HalvingGridSearchCV`.

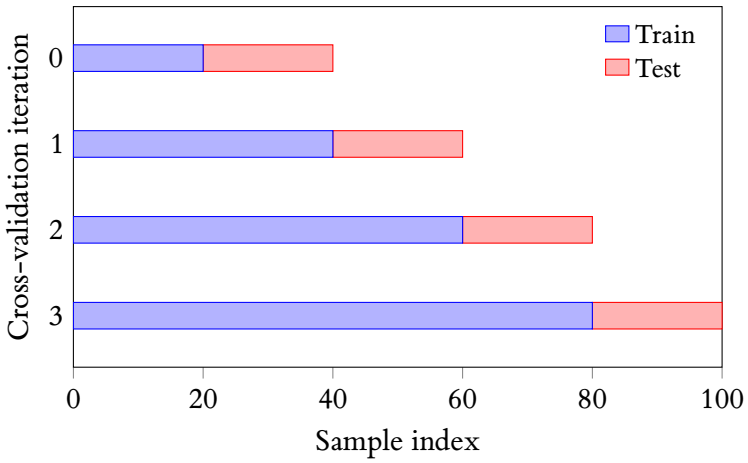

Figure 1: A visualization of the nested time-series cross-validation behaviour that I used, after the visualization of `sklearn.model_selection.TimeSeriesSplit`.

following the task description. Finally, the statistical significances of the differences between the mean scores of the best estimators were assessed by paired t-tests of the scores on the ten cross-validation folds. These methods are described in more detail in sections 4 and 5.

3 Data analysis

The data is at hourly intervals with $n = 54385$ instances across 75 stations. A summary of its features and the distributional characteristics of the non-temporal quantitative features are given in tables 1 and 2. This analysis revealed that many of the features, including the target variable `bikes`, are missing for $n = 73$ instances, which were excluded in each of the pipelines. Additionally, the 'profile' features, i.e., the features derived from the numbers of available bikes at preceding times, are not defined for the first week of instances at each station. Hence, they are missing for approximately $\frac{1}{4}$ of the instances. The meteorological features are constant for all stations at a given timestamp. Some features have a natural range – in particular, the number of available bikes at a given station is bounded by zero and the number of docks at that station (section 3.2).

3.1 Feature selection

Intuitively, features that have zero variance are not informative for regression analysis and were automatically excluded.[5] In the case of sub-task 1, the available data is limited to the month of October 2014; therefore, these included the month and year. The 'station' features (table 1) are constant for all instances at a given station, so were likewise excluded in this case. Finally, the `precipitation` feature is zero for all instances.

Correlations between features are undesirable in regression analysis (Alin 2010). Therefore, I sought to identify and exclude redundant features. To determine these, I computed the Pearson correlation coefficients between pairs of quantitative features (fig. 2), which yielded the following:

- `bikes_avg_full` and `bikes_avg_short` are fully correlated ($r = 1.00$);
- `bikes_3h_diff_avg_full` and `bikes_3h_diff_avg_short` are fully correlated ($r = 1.00$);
- `wind_speed_max` and `wind_speed_avg` are highly correlated ($r = 0.96$).

Hence, the second of each of these pairs of features was also excluded for sub-task 1. For sub-task 2, the 'profile' features were retained because they are inputs to the pre-trained linear models (table 6).

[5] See `sklearn.feature_selection.VarianceThreshold`.

Category	Feature	Data type	Kind
Station	station	int	ordinal
	latitude	float	
	longitude	float	
	docks	int	
Temporal	timestamp	int	
	year	int	
	month	int	
	day	int	
	hour	int	
	weekday	str	ordinal
	weekhour	int	
	is_holiday	bool	categorical
Meteorological	wind_speed_max	float	
	wind_speed_avg	float	
	wind_direction	float	
	temperature	float	
	humidity	float	
	pressure	float	
	precipitation	float	
Bikes	bikes	int	
	bikes_avg_full	float	
	bikes_avg_short	float	
	bikes_3h	int	
	bikes_3h_diff_avg_full	float	
	bikes_3h_diff_avg_short	float	

Table 1: A summary of the features and the target variable (bikes). Except where indicated, the features are quantitative. The features have been renamed to be easier to read.

Feature	n/a	Mean	Variance
wind_speed_avg	73	4.69	21
wind_direction	365	170.23	7,553.8
temperature	73	21.71	10.7
humidity	73	65.94	279.7
pressure	73	1,002.26	1,808.27
bikes	73	7.35	43.15
bikes_avg_full	12,264	7.31	35.82
bikes_avg_short	12,264	7.31	35.82
bikes_3h	292	7.34	43.17
bikes_3h_diff_avg_full	12,483	$4.1 \cdot 10^{-3}$	22.4
bikes_3h_diff_avg_short	12,483	$4.1 \cdot 10^{-3}$	22.4

Table 2: The numbers of missing values (n/a) and distributional characteristics of the non-temporal quantitative features and the target variable (bikes). Features that have zero variance for the first case of sub-task 1 are excluded (section 3.1).

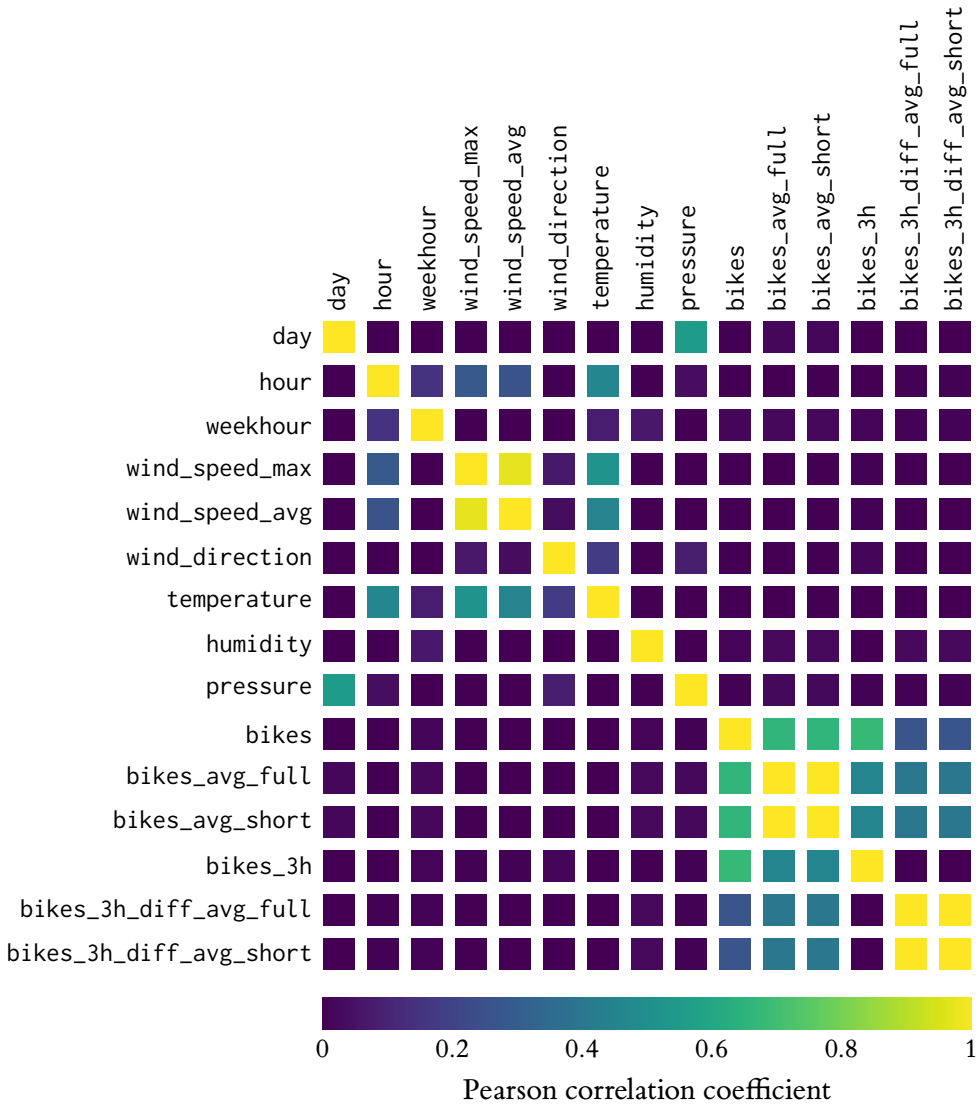

Figure 2: The Pearson correlation coefficients between pairs of the quantitative features and the target variable (`bikes`). The ordering follows table 1. Features that have zero variance for the first case of sub-task 1 are excluded (section 3.1). The `timestamp` feature is also excluded because it is naturally correlated with the other temporal features.

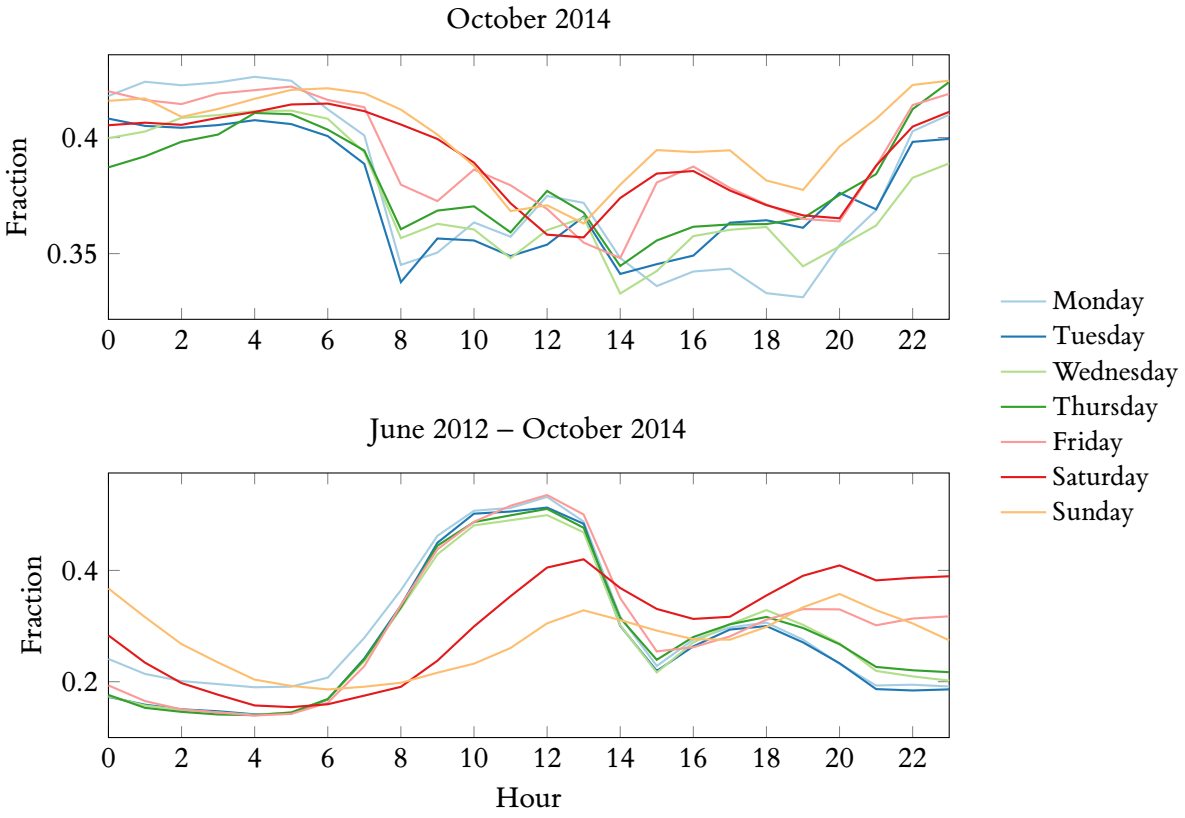

Figure 3: The average fraction of available bikes at each hour of the day, separated by the day of the week. In the top chart, the data is limited to the month of October 2014 for the 75 stations to predict. In the bottom chart, the additional data from the first ten stations over a year.

3.2 Feature engineering

The bounds on the number of available bikes at a given station were enforced for sub-task 1 by predicting the *fraction* of bikes, i.e., the number of bikes divided by the number of docks at the station. I implemented this by extending the `TransformedTargetRegressor` meta-estimator to permit data-dependent transforms.[6] Another possible transformation of the target variable would have been to predict the *change* in the number of available bikes, but I did not explore this option.

Initially, I investigated introducing derived temporal features, e.g., by discretizing the hour of the day into 'day' and 'night' intervals. However, as shown in fig. 3, the average fraction of available bikes at each hour of the day differs substantially between the training data and the data from the first ten stations. Hence, I considered that a feature of this kind would be unlikely to generalize well to the test data. Additionally, having elected to investigate tree models for sub-task 1 (section 4), which partition the spaces of quantitative features (Flach 2012, p. 155), discretization may be unnecessary.

4 Sub-task 1: Decision-tree regressors

4.1 Model classes

Gradient-boosted decision trees are a popular choice of model class for time-series forecasting problems (Bojer and Meldgaard 2021). In particular, `scikit-learn` provides an optimized implementation inspired by LightGBM (Ke et al. 2017).[7] An advantage of this model class is that it supports missing values, which are evident in the data (table 2). For both cases of sub-task 1, I chose to investigate the

[6] See `sklearn.compose.TransformedTargetRegressor`.
[7] See `sklearn.ensemble.HistGradientBoostingRegressor`.

Parameter	Values
criterion	absolute_error
max_depth	None, 10, 20, 50
min_samples_leaf	5, 10, 20
max_leaf_nodes	None, 7, 15, 31

(a) Decision tree

Parameter	Values
loss	absolute_error
scoring	neg_mean_absolute_error
l2_regularization	0.1, 0.2, 0.5
max_depth	None, 5, 10, 20, 50
max_iter	10, 20, 50, 100, 200, 500
max_leaf_nodes	None, 15, 31, 63
min_samples_leaf	1, 2, 5, 10, 20, 50

(b) Gradient-boosted decision tree

Table 3: The parameter grids over which hyperparameter search was performed for sub-task 1. Except where stated, the default values of the parameters were used. Scoring criteria were fixed to the mean absolute error, following the task description. The best parameters for the case of a single model for all stations are underlined.

performance of decision trees and ensemble methods based on decision trees. Additionally, I predicted the arithmetic mean of the fraction of available bikes as a baseline.

The best estimator for each model class was determined by grid search with ten-fold cross-validation (section 2). However, with a separate model for each of 75 stations, I found that it was computationally expensive to perform grid search over a wide range of hyperparameters. Hence, I constrained the search space by first searching a wider range for the case of a single model for all stations, then varying the parameters about the values of the best estimator for the case of a separate model for each station. The resultant parameter grids for decision trees and gradient-boosted decision trees are given in table 3.

4.2 Results

The mean scores achieved by the best estimator for each model class over the cross-validation folds and stations are listed in table 4. With a separate model for each station, there is greater variance in the mean score because there are 75 times as many samples, with fewer instances per cross-validation fold. Intuitively, the baseline achieved a lower score with a separate model for each station. In both cases of sub-task 1, all model classes achieved a lower score than the baseline.

To determine whether the differences between the mean scores achieved by the best estimators for each model class were statistically significant, I performed paired t-tests of the scores on the ten cross-validation folds. I did not compare the scores between estimators for the different cases of the sub-task because the samples are different, i.e., not paired. With a separate model for each station, both individual decision trees and gradient-boosted decision trees performed significantly better than the baseline for roughly $\frac{2}{3}$ of the stations (table 5a). However, gradient-boosted decision trees only performed significantly better than individual decision trees for eight stations. With a single model for all stations, the results were more decisive – the best gradient-boosted decision tree performed significantly better than the best individual decision tree (table 5b).

Model	μ	σ^2
Baseline	4.43	3.82
Decision tree	3.38	3.05
Gradient-boosted decision tree	3.23	2.54

(a) Separate model for each station

Model	μ	σ^2
Baseline	5.45	0.06
Decision tree	3.11	0.05
Gradient-boosted decision tree	2.60	0.08

(b) Single model for all stations

Table 4: The mean scores achieved by the model classes for each case of sub-task 1. In table 4a, μ is the mean of the scores for the 75 stations and ten cross-validation folds. In table 4b, μ is the mean of the scores for the ten cross-validation folds only.

Model 1	Model 2	μ_t	$t > 0$	$p < 0.05$
Baseline	Decision tree	2.957	71	47
Baseline	Gradient-boosted decision tree	3.399	73	49
Decision tree	Gradient-boosted decision tree	0.452	47	8

(a) Separate models for each station. The average t-statistic (μ_t), number of positive t-statistics ($t > 0$), and number of significant p-values ($p < 0.05$) for paired t-tests of the scores achieved by the best estimators for each model class on the ten cross-validation folds.

Model 1	Model 2	t-statistic	p-value
Baseline	Decision tree	25.2	1.16×10^{-9}
Baseline	Gradient-boosted decision tree	29.4	2.99×10^{-10}
Decision tree	Gradient-boosted decision tree	13.2	3.43×10^{-7}

(b) Single model for all stations. The t-statistic and p-value for paired t-tests of the scores achieved by the best estimators for each model class on the ten cross-validation folds.

Table 5: The results of paired t-tests of the scores achieved by the best estimators for each model class for the cases of sub-task 1. A positive t-statistic indicates that 'Model 2' achieved a lower mean score than 'Model 1'.

Model	temperature	bikes_3h	bikes_avg_full	bikes_avg_short	bikes_3h_diff_avg_full	bikes_3h_diff_avg_short
rlm_full		✓	✓		✓	
rlm_full_temp	✓	✓	✓		✓	
rlm_short		✓		✓		✓
rlm_short_full		✓	✓	✓	✓	✓
rlm_short_full_temp	✓	✓	✓	✓	✓	✓
rlm_short_temp	✓	✓		✓		✓

Table 6: The features used by the different kinds of linear models for sub-task 2. The features are listed in table 1 and follow the same ordering.

5 Sub-task 2: Ensembles

The available information for sub-task 2 is a set of linear models that were trained on the data for a separate set of 200 stations. For each station, there are six models that were generated by the `rlm` function from the R package MASS.[8] Each of the six kinds of model uses a different set of features, which are listed in table 6. For this sub-task, I automatically excluded zero-variance and highly correlated features but retained the 'profile' features (section 3.1). Unlike the gradient-boosted decision tree implementation in `scikit-learn`, the linear models do not support missing values, so I imputed these with the mean of the feature value over the training data.[9] The models are pre-trained, so I used stacked generalization to combine their predictions (Wolpert 1992).[10] This method employs a meta-estimator that trains a second-level regressor on the predictions of a set of first-level regressors (the pre-trained linear models).

As discussed in section 3.1, the 'full' and 'short' features are fully correlated, and the latter were excluded for sub-task 1. Therefore, I first investigated the relative performance of the different kinds of models. A box plot of the mean scores achieved by the individual pre-trained of each kind is shown in fig. 4. This demonstrates that individual `rlm_short` and `rlm_short_temp` models generally achieved lower scores. However, simple stacked generalization with each of the kinds of models, i.e., using the default parameters of `sklearn.ensemble.StackingRegressor`, achieved very similar scores, so I elected to include all of them in the stacked generalization. The best second-level regressor was determined by grid search with ten-fold cross-validation (section 2).

References

Alin, Aylin (2010). "Multicollinearity". In: *WIREs Computational Statistics* 2.3, pp. 370–374.

Bergmeir, Christoph, Rob J. Hyndman, and Bonsoo Koo (2018). "A note on the validity of cross-validation for evaluating autoregressive time series prediction". In: *Computational Statistics & Data Analysis* 120, pp. 70–83.

Bojer, Casper Solheim and Jens Peder Meldgaard (2021). "Kaggle forecasting competitions: An overlooked learning opportunity". In: *International Journal of Forecasting* 37.2, pp. 587–603.

[8] See `https://www.rdocumentation.org/packages/MASS/versions/7.3-54/topics/rlm`.

[9] See `sklearn.impute.SimpleImputer`.

[10] See `sklearn.ensemble.StackingRegressor`.

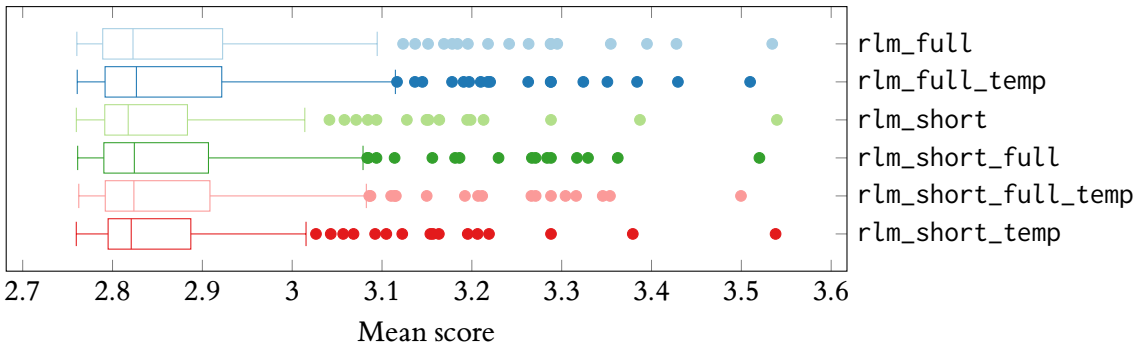

Figure 4: A box plot of the mean scores achieved by the different kinds of models for sub-task 2 over the ten cross-validation folds of the data from the 75 stations for the month of October 2014, i.e., the same data as for the second case of sub-task 1. The kinds of models are listed in table 6.

Buitinck, Lars, Gilles Louppe, and Mathieu Blondel (2013). "API design for machine learning software: experiences from the scikit-learn project". In: *ECML/PKDD 2013 Workshop: Languages for Data Mining and Machine Learning*.

Flach, Peter (2012). *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. 1st ed. Cambridge University Press.

Ke, Guolin et al. (2017). "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc.

Pedregosa, Fabian et al. (2011). "Scikit-learn: Machine Learning in Python". In: *The Journal of Machine Learning Research* 12, pp. 2825–2830.

Wolpert, David H. (1992). "Stacked generalization". In: *Neural Networks* 5.2, pp. 241–259.