University of Bristol

Machine Learning

Prediction of the Number of Bicycles at Rental Stations

Stephen Thompson
Connor Williams

The Task

The goal in this assignment was to predict the number of available bicycles in all rental stations 3 hours in advance. There are at least two use cases for such predictions: first, a user plans to rent or return a bike in 3 hours' time and wants to choose a bike station which is has bikes available to rent or spaces to return a bike. Second, the bike rental company wants to avoid situations where a station is empty or full and therefore needs to move bikes between stations. For this purpose, they need to know which stations are more likely to be empty or full soon.

Phase 1

In this task we were given the data of 75 stations (Station 201 to Station 275) for the period of one month. We needed to be able to predict the number of bikes at each station at any given time for the next 3 months by firstly training a different model for every station and then training a general model which would work on all of the stations.

The Data

The first task would be to investigate the data to determine any obvious patterns or useful/ useless features. The features can be separated into 4 categories:

- Station information:
    - station, latitude, longitude, numDocks
- Temporal information:
    - timestamp, year, month, day, hour, weekday, weekhour, isHoliday
- Weather:
    - windMaxSpeed.m.s, windMeanSpeed.m.s, windDirection.grades, temperature.C, relHumidity.HR, airPressure.mb, precipitation.l.m2
- Counts & stats:
    - bikes_3h_ago, full_profile_3h_diff_bikes, full_profile_bikes, short_profile_3h_diff_bikes, short_profile_bikes, bikes.

The Problem

This task would ultimately be to provide a model which could answer the question *“given station X at time T, weather conditions W and the number of bikes here previously, how many bikes are likely to be parked here at time T′ ?”.* Part of this task is therefore regression, and to train a model with the target variable ‘bikes’. The other part of this task is feature selection so we can determine which features best give information on how many bikes are likely to be parked at a station in the future.

Software Selection

To decide which software tools to use for our model implementation we first had to examine what our needs were and the options available.

Needs

To achieve our goals in the assignment we needed a tool to aid in feature selection and also a one to implement the model. At an early stage we did not know which type of model we would use, so a tool with many options would be preferable. Because we needed to assemble the data from many files and then generate output in a specific format we also needed easy to use and automatable file I/O. The size of the data we would be processing is not too large so advanced implementation for parallel processing would not be worth the extra difficulty for this task. It would also be preferable to work in a language that we were comfortable with.

Options

- Weka - We considered Weka due to familiarity from using it during previous coursework. While it would have been a fine tool for the job we decided that we would prefer something that would be more customisable.
- Matlab - Matlab would have been a sufficient tool however we decided that the file I/O would be difficult considering the mixture of data types in the training and test data. There was also the issue that as it is proprietary software we would be bound to the lab machines preventing us working from home.
- SciKit-Learn - SKL is a machine learning library written in Python. It provides a plethora of models to use, has tools for feature selection and as we can implement our own I/O which makes it completely flexible to our needs. For these reasons we decided to use it for this project.

Regression model

There are plenty of regressors which would be fit for this task such as linear regression, ridge regression, decision trees, support vector regression etc. We decided that something simple such as Linear regression would be an appropriate choice for this task due to the fact that a good MAE has already been achieved by the baseline models. Linear regression, however, has many drawbacks when applied to modern data. It is sensitive to outliers and cross-correlations and is also subject to overfitting. Linear regression has zero bias but suffers from high variance. For this task it may be worth sacrificing some bias to achieve a lower variance therefore we have used SciKit-Learns's ridge regression which is a more robust version of linear regression, putting constraints on regression coefficients to make them much more natural and less subject to overfitting.

Feature Selection

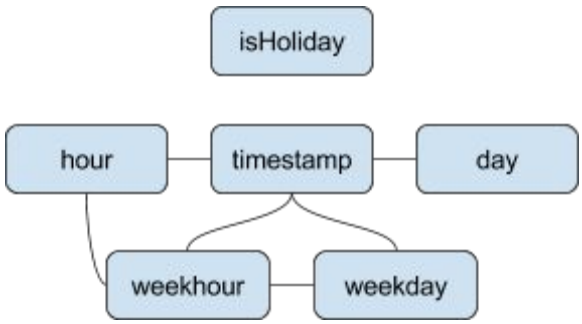
Before running the data through a regressor, it is important that we choose the best features to use. The aim is to to achieve a Goldilocks balance with the number of predictors to include:

- Too few: An underspecified model tends to produce biased estimates.
- Too many: An overspecified model tends to have less precise estimates.
- Just right: A model with the correct terms has no bias and the most precise estimates. [3]

It is immediately obvious from looking at the data that ‘month’ and ‘year’ are completely useless because they never change - all of the training data is from October 2014. These two features have no variance, rendering them useless.

When training a different model for each station, we can also remove some more features which have 0 variance which are ‘station’, ‘latitude’, ‘longitude’ and ‘numdocks’. However bear in mind that these features may still be useful for training a general model which will work on all stations.

The diagram below shows the relationship between the remaining temporal features (after removing ‘year’ and ‘month’). From it we can observe that there is an unnecessary overlap and we can simply work out ‘weekday’ and ‘hour’ from the single feature ‘weekhour’, meaning we can also remove ‘weekday’ and ‘hour’.



Relationship of remaining temporal features [1]

Another feature to remove from the data is ‘timestamp’. This feature has too many possible values for us to be able to distinguish any similarity between different time points. We do not need time as precise as seconds or even minutes if we are predicting bikes in 3 hours later.

It is not immediately obvious if any of the other features are definitely useful for our model or not, so we will not remove any more manually.

To distinguish which of the rest of the features we want to use, we use SciKit-Learn’s univariate feature selection which works by selecting the best features based on univariate statistical tests. We use this to select the K best features using the f_regression option which tests the effect of a single regressor, sequentially for many regressors in three steps:

1. The regressor of interest and the data are orthogonalized wrt constant regressors.
2. The cross correlation between data and regressors is computed.
3. It is converted to an F score then to a p-value [2]

Method 1: K=9

Individual Model

Selecting the 9 best features for a different model for each station is hard to keep track of because the features differ between stations however there are a few common ones.

bikes_3h_ago, short_profile_bikes, short_profile_3h_diff_bikes, full_profile_bikes, full_profile_3h_diff_bikes are the features which consistently obtain a good p-value and the other 4 features to be used differ between the rest of the 10 features left, indicating they are more relevant to some stations than to others.

General Model

Below is a table which orders the features by their p-values determined by SciKit-Learn's SelectKBest function. We can see that the common features from the individual models are also in the top 6 for the general model, as well as latitude which we did not use before due to it having 0 variance within stations.

Feature	P-Value
precipitation.l.m2	0.92
windDirection.grades	0.91
day	0.42
windMeanSpeed.m.s	0.048
windMaxSpeed.m.s	0.045
weekhour	0.023
temperature.C	0.00032
relHumidity.HR	0.00021
isHoliday	$8.34 * 10^{-4}$
station	$4.95 * 10^{-8}$
airPressure.mb	$2.95 * 10^{-11}$
numDocks	$8.57 * 10^{-178}$
latitude	0
full_profile_3h_diff_bikes	0
short_profile_3h_diff_bikes	0
longitude	0
full_profile_bikes	0
short_profile_bikes	0
bikes_3h_ago	0

Final results from using the 9 best features:

- Individual model average MAE = 2.718

- General model average MAE = 2.888

Method 2: K=4

Individual Model

Selecting the 4 best features for a different model for each station is again difficult to keep track of because the features differ between stations however the three common ones which appear regularly are bikes_3h_ago, short_profile_bikes, full_profile_bikes. The last feature used differs slightly between the other 3 profile information features.

General Model

The 4 best features as before are bikes_3h_ago, short_profile_bikes, full_profile_bikes and longitude.

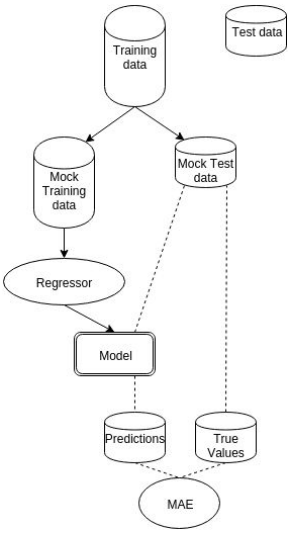
Final results from using the 4 best features:

- Individual model average MAE = 2.745
- General model average MAE = 2.904

The first method of feature selection worked better (where we used the 9 best features). This is simply down to the fact that we are using more data to train our model, resulting in better estimates in future. If we used any more than these 9 features we could sometimes observe worse performance on testing due to overfitting our model to our training data. This can be noticed by looking at the coefficients of the model and seeing that some coefficients are extremely small, meaning that the feature is not necessarily useful on future data.

Testing and Evaluation

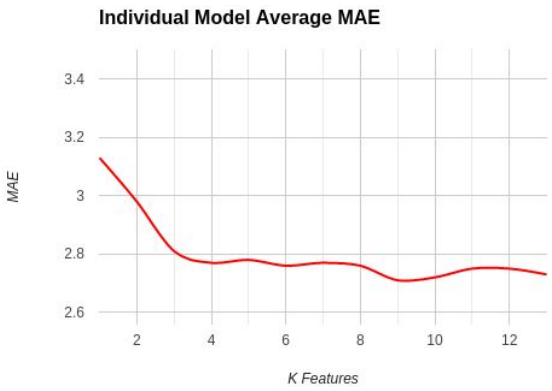
Without the test data, a way to test our generated models locally is to use a technique similar to K-fold validation. We split the training data up so that a random 30 instances for each station are used for the ‘mock’ test data and the other 715 instances for each station are used for training the model. Every time we run our script to generate a model, we first generate a different random set of ‘mock’ data.

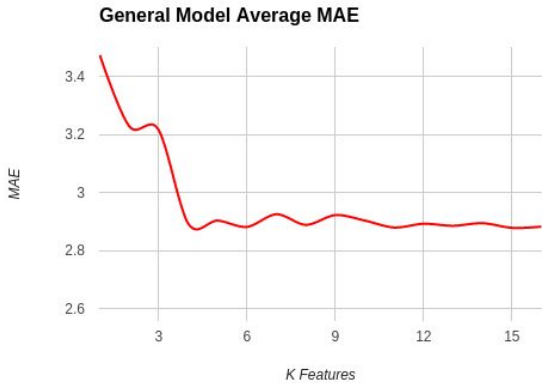


Once we have generated a model, we then run the mock test data through it to be left with some predictions. We calculate the mean absolute error between these predictions and the true values of ‘bikes’ to see how well the model performs. We also calculate the baseline for the training data as it is different to that of the test data. We calculated it to be 3.38 by simply predicting “bikes_3h_ago”. In the rest of the report this is the figure we evaluate against. We have found by submission

that beating this baseline in our tests results in a similar improvement over the baseline specified for the test data.

Using this testing method to plot the following graphs, we can observe that the feature selection used is appropriate. Using the 9 best features results in the best performance for the individual model and the 4 best features yield the best result for the general model.





Comparison of Methods

Both general and individual models have different strong points. A general model has a lot more data to train on and therefore would be less susceptible to noise and more likely to be accurate; it has lower bias but higher variance. Not only is the sample size larger, it also can use more features in comparison to the individual model for each station. This is because some features remain static for a single station i.e. station, latitude, longitude and numDocks. It also has the benefit of being able to be reused if a new station was built and needed predicting where as an individual model would need retraining for every new station.

Individual models have a much smaller set of data to operate on, both in features and sample size. This will cause more bias but less variance. Despite this, as they are modeled individually, they could reflect the characteristics of the specific station more effectively. e.g. some stations may be depleted in the morning and others in the day.

In this instance we found that the individual models provided better predictions. This will be due to the stations behaving differently enough that the advantages of the individual model outweigh the inaccuracy of the general model.

Phase 2

We were given a set of linear models trained on other stations (Station 1 to Station 200) with the training data from a whole year. Although these models were not trained on the stations to be predicted, they could still be used since there should be some similarity among different stations. To successfully use these models would reuse the knowledge learned from a whole year's data, which has potential to be powerful. The task was then to figure out how to predict the stations in Phase 1 by only using these pre trained models.

Using the provided models

We started by examining the data provided from the models to see how we might be able to reuse them. Unfortunately there was no information given other than the coefficients of a linear model. The would make it very difficult to try and draw parallels between the models given and singular stations we wished to predict e.g. trying to generate location specific models by collating models from nearby stations. Because of this and the success of our previous general model we decided to combine the 200 individual models supplied into one new general model by averaging the coefficients. We were provided with variety of models which we evaluated by using multiple random samples of the training data from stations 201-275 as testing data. Our results were as follows.

Model	M.A.E
Full Profiles	2.882
Full Profiles + Temperature	2.865
Short Profiles	2.826
Full and Short Profiles	2.877
Full and Short + Temperature	2.873

From this table we can see that the all of the models performed similarly but the short profiles model seemed the most effective.

When we evaluated our models versus the provided ones we found that the provided gave noticeably better predictions. The major reason for this is that the dataset that trained these models was significantly larger. This would make the model less susceptible to noise and bias but could result in higher variance. The benefit, in phase , of us being able to train for individual stations was that the model was tailored to the characteristics of each station. In phase 2 we found that the benefit was far outweighed by the extra data in the provided models.

Phase 3

The next stage of the project was to try and combine the models we had created in the first 2 phases in order to provide better predictions. We combined the models by taking a weighted average of the predictions from the previous models. We experimented on different weightings and then tested on random samples of the training data.

Phase 1 Weight	Phase 2 Weight	MAE
1	1	2.852
2	1	2.874
1	2	2.48

Unfortunately none of these provided better results that we had in phase 2. You can also see that as we increase the weighting of phase 2 the MAE decreases. The reason for this is that the predictions in phase 2 were better so diluting their accuracy with the less accurate predictions from phase 1 would just increase the MAE.

Future Improvements

We are relatively happy with the performance of our final models but there is still room for improvement. Firstly it would be possible to use model ensemble in all phases of our project. A linear classifier, such as the one we used, is a high bias model so boosting would help reduce the bias and provide better predictions. We could have done this in phase one by training multiple models on the subsets of the test data. In phase 2 we could also have done similar.

Although we did investigate using a decision tree classifier we found that it over fit the data and performed badly in tests. If we implemented a random forest we could have reduced the variance to overcome this.

Implementing model ensembles in the first 2 phases would also have made integrating them in phase 3 much more effective, resulting in better predictions.

Citations

[1] http://reframe-d2k.org/File:MoReBikeS_2015_paper_2.pdf
[2] http://scikit-learn.org/stable/modules/generated/
[3] http://blog.minitab.com/blog/adventures-in-statistics/how-to-choose-the-best-regression-model