Predicting Distribution of Dublin Bikes

Con O'Leary; olaoghac@tcd.ie

August 16, 2021

1 Evaluation

I believe **R**2** evaluation is appropriate for measuring success in this task. A prediction that is off by one bike is probably only going to be a fifth as much of an issue (to a hypothetical user availing of the model) as a prediction that is off by five bikes. As such, the fact that R**2 evaluation measures how correct the variance of predictions are—and that variance accounts equally for the frequency of errors as well as the magnitude of errors—means that error in the model is represented proportionally to how much of an issue is presented to the objective of the task.

2 Data procurement

There are considerable time gaps in *Dublinbikes 2020 Q1 usage data*. I recognised this accidentally, when I made a provisional graph that shows the percent bike occupancy of each station throughout the full duration of the data.

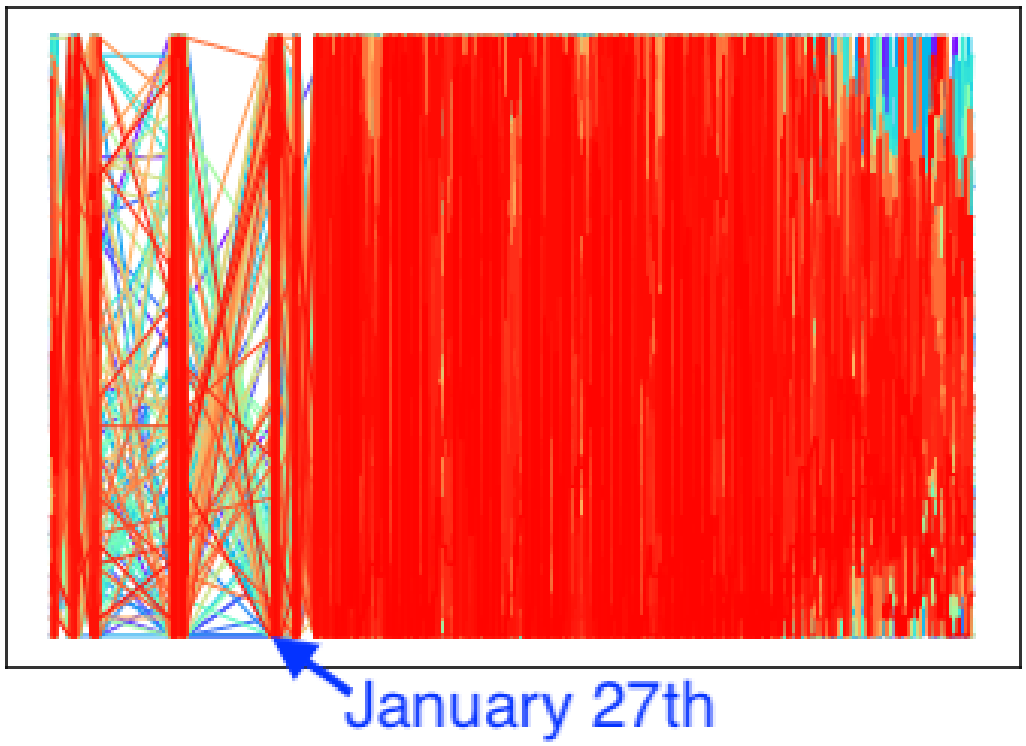

Figure 1: % bike occupancy of each station over the total duration

The sections of the x-axis that are atopped by red show the final station to be plotted (plotted in red) fluctuating in percent occupancy; they show where there is data. Where there is a mesh of colours the graphing library is adjoining lines across a data-less span. **The end of the second data-less span is the 27th of January 2020, and so I processed the information from then onwards.** The reason why I was adamant to exclude any dates before the 27th that did have full data is so that I would not have more data for certain days of the week than I have for others: I was worried this might lead to the models orientating themselves more around certain days of the week.

3 Baseline Approach

I first went about a linear regression. Noting the cyclical nature of the amount of bikes in a station, I derived polynomial features from the amount of bikes in a given station over the total duration. I disabled the shuffle parameter in my train-test split, as it seemed testing would be improper if, for example, we were looking for a prediction for the amount of bikes in station x in 30 minutes time when the prior training had included the amount of bikes in station x in 25 and 35 minutes time. Understandably, the x-axis of the training data spanning months and the x-axis of the testing data spanning weeks was not conducive at all, regardless of the degree of the polynomial features.

Next, with the same test/train division, I went about training 7 linear regressions—one for each day of the week. I trained the regressions independently, on their respective polynomial features, obtained from the share of data pertaining to that week day. When this approach fell through I didn't stick around long to figure out why, as I felt that would be counter-intuitive to the purpose of a baseline.

Finally, I abandoned the use of models and established a baseline directly from the data: the average bike occupancy per datapoint for each day of the week. The results of which were solid; scoring near-positive R**2 scores. Looking through a comparison of predicted and desired y-values I thought some form of regularisation would increase the accuracy. To regularise, I diluted my predictions with the mean y-value for the respective weekday of the given prediction. Optimising the coefficient that determines the degree to which the original predictions are diluted, I scored a marginally positive score for one of my stations, and a near-positive for the other. Interestingly this was my only approach that worked better for the more central, less residential Custom House Quay station. Seeing as my baseline scored an R**2 score of roughly zero, this, as far as I understand, means its squared sum error was roughly that of the mean line of the training data. To gauge the efficacy of my baseline, as well as to see the effect of the regularisation, I plotted the R**2 scores of my baseline (with a range of values for my regularistation coefficient) versus the R**2 scores of the per-station mean of the training data. Realising the latter performed approximately as well as the approach I had developed, I adopted it as my baseline.

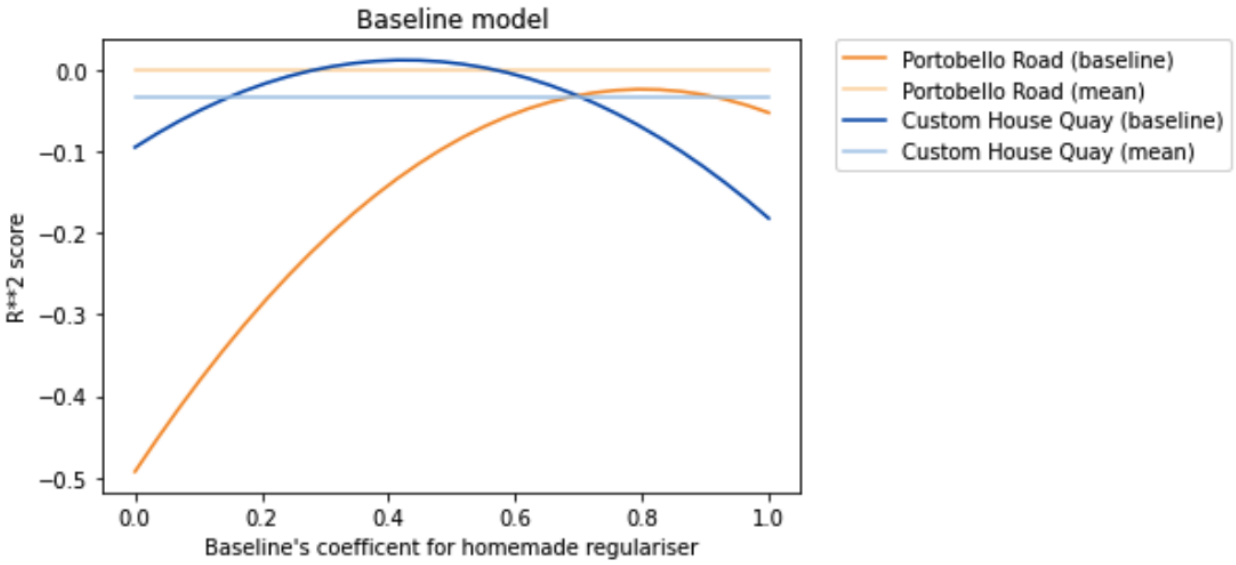

Figure 2: Optimising the regularisation of baseline predictions

4 Approach 1

The idea of my first approach was to identify patterns between bikes disappearing in stations and bikes turning up in other stations. As such I conceived a number of features: Chief among these is *bikes_changes_pastx*, which details for a given station at a given 5-minute interval the change in the number of bikes in the past x minutes. So that correlations between the changes detailed in *bikes_changes_pastx* could be found, time needed to be represented in features. The particular aspects of time that seemed to matter for the purposes of representing these patterns in bikes disappearing and turning up are the hour of day and the day of week. Figuring that an MLPRegressor might be a good means to represent these patterns, I made 7 inputs to represent the days of the week, and 23 to represent the time of the day. The 23 are represented as values between 0 and 1, such that 6,15 am would be represented by the 6th input having a value of 0.75, the 7th input having a value of 0.25, and the rest of the 23 inputs having a value of 0. Unlike this first time-related feature, I did not have the 7 inputs representing the days of the week transition gradually, but rather be 0 or 1: the degree of transition between the days is already represented by the first time-related feature. I also included a feature that is the percent fullness of the station—so that patterns between, for example, a station being quite empty and a large amount of bikes disappearing at a neighbouring station could be identified. Any features that did not already have values that ranged between 0 and 1 or -1 and 1 were normalised.

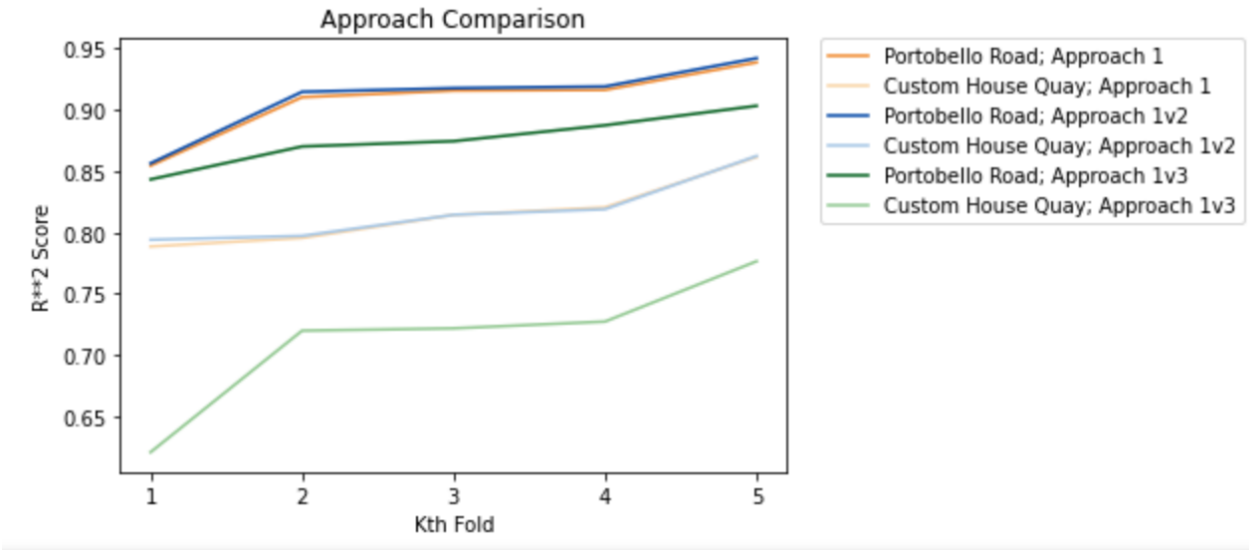

Figure 3: R**2 scores of variants of Approach 1

Variant 1 has the change in bike occupancy for the station in question over the past 5 and 10 minutes. Variant 2 is as Variant 1 but with the past 5, 10, 15, 20 and 25 minutes. Variant 3 has the change in bike occupancy for the station in question over the past 5 minutes, and the change in bike occupancy for all stations for the past 10, 15, 20 and 25 minutes.

While I had hoped the model would primarily derive its correlations from the various components of *bikes_changes_pastx*, instances of the model that did not have a whole kitchen sink of *bikes_changes_pastx* components thrown at them outperformed those that did. To be particular, the inclusion of (1) components pertaining to the change in amount of bikes over a duration greater than 10 minutes had no positive impact, and the inclusion of those greater than about 20 had a negative impact, and (2) the inclusion of components pertaining to the change in the amount of bikes in all stations had negative impacts. This was disappointing as I had hoped that a model having access to features that had the potential to describe inter-station bike transition patterns would discover such. Perhaps giving the changes in bike occupancy in all stations entailed too much irrelevant information. The way I would further this model if I had more time would be to only look at bike changes in the past x minutes for stations that can be travelled to and from in x minutes. What gives further credence to the merit of the aforementioned adjustment is the disparity between the results for the two stations when Approach 1v3 is applied: This disparity shows that certain stations take fine to having the data for all stations, whereas others do not, which could indicate that optimising the conditions under which data is granted for other stations (e.g. proximity) could hold great potential.

5 Approach 2

My second approach, at least as conceived in my mind, orientates around finding recurring behaviour in a station's bike occupancy changes. To identify these recurrences I fit a KNN algorithm with features, such that the neighbours of a given datapoint for a given station are the datapoints for that station where the conditions of the features most closely resemble that of the datapoint in question. Like the first approach, it uses components from *bikes_changes_pastx*, as well as the stations percent occupancy. Time is represented differently to first approach. If the hour of day were to represented as in the first approach, the KNN algorithm would regard 01:00 and 02:00 to be neighbourly, although it would not regard 23:00 and 00:00 to be neighbourly. As such, I represented the time of day as two different values, which are the xs and ys of a geometric circle.

This means 23:00 and 00:00 would be regarded by the algorithm as being as neighbourly as 01:00 and 02:00.

I think this approach took poorly to the components of *bikes_changes_pastx* pertaining to more than 10 minutes ago because, unlike the multi-layer perceptron model, the KNN algorithm does not weight coefficients for its features, so components pertaining to greater durations have the same weighting as the more-relevant past5 and past10 components.

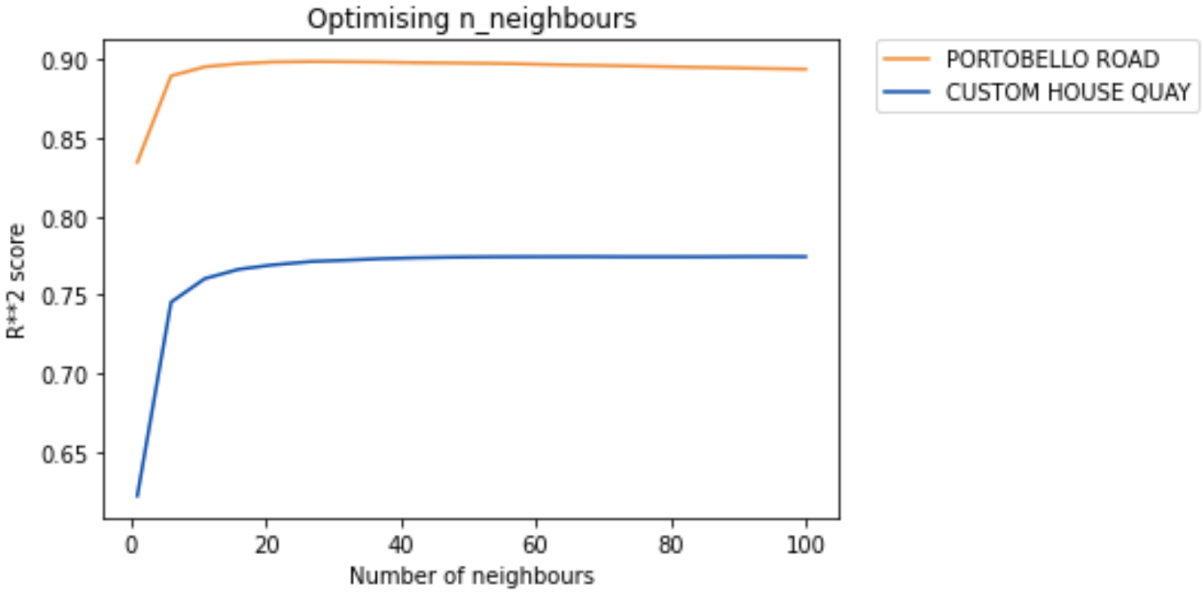

Figure 4: Optimising the number of neighbours

6 Conclusion

I was surprised that the KNN approach (approach 2) managed to perform nearly as well as the MLP approach.

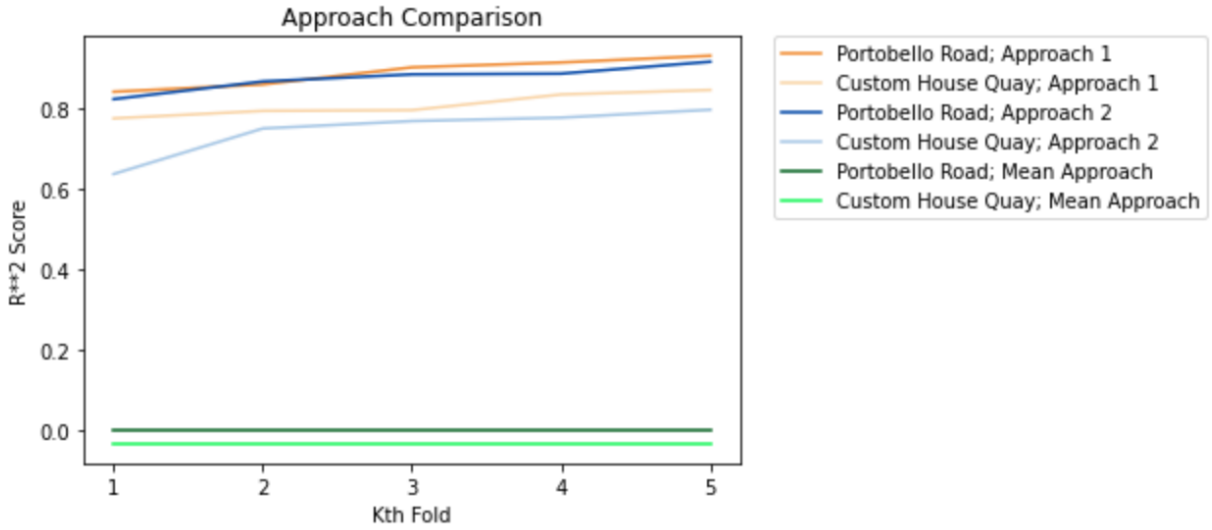

Figure 5: The two main approaches and the baseline approach

Either of the two approaches would probably suffice to drive a prediction feature for a Dublin Bikes app: the task does not require a particularly acute degree of accuracy, but rather consistency in being about right. The following graph plots the degree and severity of the errors of the two approaches when tasked to make the same predictions.

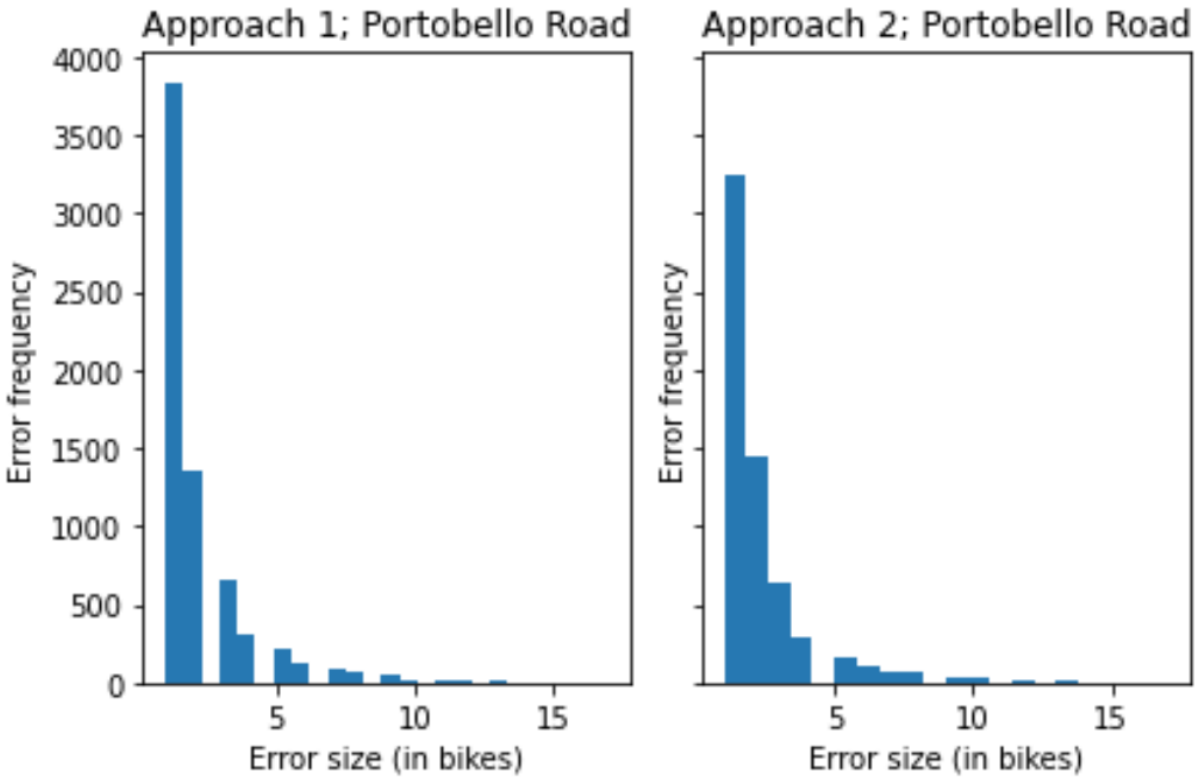

Figure 6: Of the erroneous predictions, the size and frequency of the error

In either case, it is evident by eye that roughly 90% of errors are off by between 1 and 4 bikes, which would be a nominal issues for users a great majority of the time. Nonetheless, I am certain R**2 scores in the high 90s could be achieved by a finer approach.