Forecasting Bike Rental Demand

Jimmy Du, Rolland He, Zhivko Zhechev

Abstract

In our project, we focus on predicting the number of bike rentals for a city bikeshare system in Washington D.C., as part of a Kaggle competition. Through our project, we identified several important feature engineering ideas that helped us create more predictive features. We then trained and tuned several different models with our data, and experimented with ensembling methods as well. In the end, we found that decision-tree based models perform well on the bikeshare data; in particular, using a conditional inference tree model yielded both the best cross-validation result and leaderboard performance.

1 Introduction

Predicting demand of ride sharing systems has become a common problem in the modern world, with companies such as Uber and Lyft emerging in the car-sharing service. Similarly, bike-sharing services have gained considerable traction in the U.S. in the past decade [1]. Heavy street traffic in busy cities and a desire for an environmentally friendly form of transportation make biking an attractive alternative to traveling by car. A limited supply of bikes, increasing demand for bikes, and cost of storage and relocation of bikes serve as motivation for forecasting the demand of bikes.

This paper examines the Capital Bikeshare program implemented in Washington D.C. We are provided hourly bike rental data with weather and date information spanning 2 years, and our goal is to predict the total number of bikes rented on an hourly basis. Our target optimization metric is the Root Mean Squared Logarithmic Error (RMSLE), which is computed as follows:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(log(p_i+1)-log(a_i+1))^2}$$

where n is the number of observations, p_i is our predicted count, and a_i is the actual count. We seek to identify the models that result in predictions which minimize this error.

We worked on this problem as part of a Kaggle competition. The data was split between a training and test set, and each day we were allowed to submit up to 5 sets of prediction values for the test set, and the RMSLE error results for each set of predictions would be tracked on a public leaderboard.

2 Data

2.1 Original Data

The raw training data consists of 10886 observations. Each training sample consists of 12 data entries – the first 9 contain the following features: date and hour, season, holiday, working-day, weather, temp, atemp (the temperature it feels like), humidity, and wind speed. Date and hour are provided as a single string, weather is a categorical variable with 4 levels representing different weather conditions, holiday is a binary indicator variable representing whether the particular day was a holiday, and working-day is a binary indicator variable signifying whether the day was a non-holiday weekday. In addition, the last 3 data entries are casual, registered, and count. Casual and registered represent the number of bike rentals made by non-registered and registered users, respectively. Count is the sum of casual and registered values and is the value that we seek to predict. The training data represents the first 20 days of each month for a 2 year period.

Similarly, the test data consists of 6439 observations with the same features. The registered, casual, and total count data is withheld, though, as that is what we are trying to predict. The test data represents the remaining days (after the first 20 days) of each month in the same 2 year period.

2.2 Featurization and Preprocessing

To start off, we converted the date and hour variable into 3 separate components, which were added as new features: month, day of week, and hour of day. From this dataset, we identified 6 ways the feature space could be altered in order to improve generalization error:

- **Converting categorical variables into binary variables and discretizing continuous variables**
  We discretize continuous variables like temperature by splitting them into buckets (where each bucket represents a range of values) and classifying an observation under one of these buckets. We do this primarily to improve the

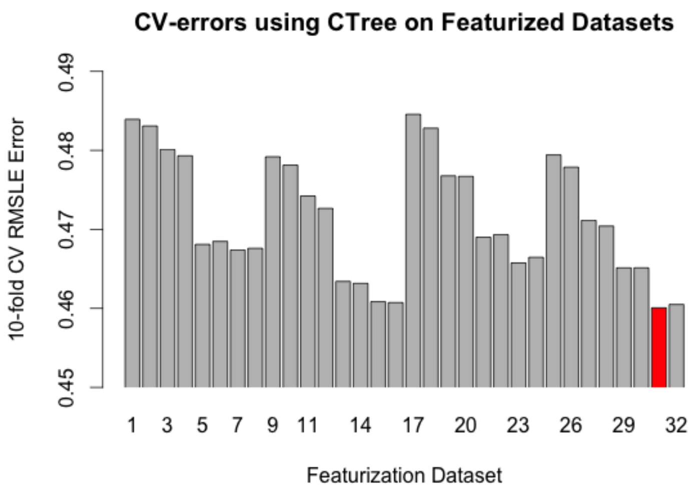

Figure 1: Plot of cross-validation errors by training a CTree model on 32 of our 64 featurized datasets. We ended up selecting the dataset represented by the bar highlighted in red.

performance of linear models, since variables such as temperature clearly do not have a linear relationship with bike rental count (each bucket, however, can be roughly approximated linearly). In addition, we also split categorical variables into binary indicator variables, as one of our models (GLMNet) could not handle categorical data.

- **Replacing the season variable by a variable indicating the month**
  There might be some differences between individual months of the same season, and therefore are not captured in the season variable. We conjectured that having a month variable instead could improve predictions.

- **Adding a new variable that represents the day of the week**
  We noticed some slight differences in the mean bikes rented between the different days, and wanted to see whether adding such a variable would positively influence our results.

- **Removing the temp variable**
  The features temp and atemp (temperature it feels like) are highly correlated, as one might expect, so removing one of them might reduce collinearity issues. We experimented with removing the temp variable, as we believed that atemp would be more relevant for an individual's decision of renting a bike.

- **Removing the holiday variable**
  We looked at the data and noticed that bike rentals did not seem to change when there was a holiday. Therefore, we tried removing the variable to see if it was just noise.

- **Adding "peak hour" indicator variables**
  Some hours clearly have more demand than other hours, so we looked at the data to identify peak hours for bike rental. We noticed that on weekdays, the peak hours for bike rentals were between 7-9 am and 5-7 pm. On weekends, the peak hours were anywhere between 10 am - 6 pm. Then we classified each observation depending on whether it fell into the weekday peak hours, the weekend peak hours, or none of them.

In order to select the best subset of these different featurizations, we wrote a Matlab script that generated all 64 featurized datasets, 1 for each subset of these featurization methods applied to the original data. We then used 10-fold cross-validation to compare the performance of all of these datasets (trained using a CTree model, which is described in the Models section), and selected the 3 datasets that yielded the lowest RMSLE error (Figure 1). We then submitted predictions to the leaderboard for these 3 datasets, and found the one that gave the best leaderboard performance. This particular dataset used only 2 of our featurization ideas: removing the holiday variable and adding the "peak hour" variables. We then performed the rest of our experiments and tests exclusively on this dataset.

3 Models

For each of the following models, we tuned the relevant hyper parameters via 10-fold cross-validation.

3.1 Linear Model

We used a basic linear regression model with no modifications in order to get some baseline predictions.

3.2 Generalized Linear Models with Elastic Net Regularization (GLMNet)

GLMNet [2] is an extension of GLM's we saw in class, by adding in additional regularization terms. This model solves the following problem:

$$\min_{\beta,\beta_0} \frac{1}{N}\sum_{i=1}^{N} w_i l(y_i, \beta_0 + \beta^T x_i) + \lambda[(1-\alpha)\|\beta\|_2^2/2 + \alpha\|\beta\|_1]$$

In the formula above, l gives the negative log-likelihood contribution from training example i and $\alpha \in [0,1]$ controls the elastic-net penalty. λ controls the overall strength of the penalty, and the regularization path is identified via coordinate descent. The model can be adapted to many distributions, including the Gaussian and the Poisson distributions. For our particular problem, we chose to use the Poisson distribution, given that we are predicting the value of a discrete occurence – the number of bikes that will be rented in a particular hour. In addition, by running some quick experiments, we found that the Poisson distribution did indeed give the lowest generalization error.

3.3 Generalized Boosted Models (GBM)

This model uses gradient boosting [8], which builds an additive decision-tree model in order to predict the outcome in a regression. The model greedily adds base learners from a select hypothesis class, and attempts to find a weighted combination of them that minimizes the training error. Though the performance of single base learners is generally poor, they can be combined to form a very strong learner – this process is known as boosting. GBM is a flexible model that can be adapted to a wide range of distributions, including the Poisson distribution, which again turns out to give the best generalization results out of the different distributions.

3.4 Principal Component Regression (PCR)

We created a new set of features using PCA (covered in class) [5], and ran a linear regression on a subset of these transformed features. It turned out that using all of the components gave the lowest generalization error.

3.5 Support Vector Regression (SVR)

This model works similarly to SVM's as described in class, but is adapted to handle regression. It attempts to approximate the value of a continuous value by using a loss function that is insensitive to the errors [6]. We found that using the Gaussian kernel with a squared error penalty performed the best for our data.

3.6 Random Forest (RF)

Random Forest [4] is a meta-algorithm that combines a large number of decision-tree models, each individually built on bootstrapped samples of the data. This process of sampling the data and combining the individual decision-trees is called bagging, and is able to reduce the variance of the predictions without increasing the bias. The final predictions are formed by taking the mean of the individual decision-tree predictions.

3.7 Conditional Inference Trees (CTree)

CTree [7] builds a decision-tree model by iteratively creating splits for the variable that is most significantly associated with the response variable, which is measured using p-values. The algorithm stops when no remaining variables have a significant p-value. The original idea to use CTree came from a forum post made by a fellow competitor [3].

3.8 Ensemble Learning – Stacking

We utilized a type of ensemble learning called stacking [9]. This involved holding out a small portion of the training dataset as the ensembling set, and forming predictions on this set with our best models after training on the rest of the data. Then we train a meta-model using these individual predictions as the new input data. Finally, we retrain the best models on all the training data and form predictions on the test data; afterwards, we apply the trained meta-model on those predictions to form our final predictions on the test set. For our data, we trained a stacking model using CTree and RF as sub-models and linear regression as the meta-model.

4 Results

We performed 10-fold cross-validation on our models, and submitted predictions for most of our models to the leaderboard. Below are the RMSLE error values (discussed in the introduction) obtained for each model:

Models	10-fold CV error	Leaderboard error
Linear	1.044	N/A
GLMNet	0.659	0.645
GBM	0.790	0.813
PCR	1.161	N/A
SVR	0.624	0.631
RF	0.503	0.547
CTree	0.460	0.508
Ensemble	N/A	0.521

Table 1: 10-fold cross-validation results and leaderboard performance using RMSLE metric

5 Discussion

Below we list some of the observations and interpretations we made from the results:

5.1 Performances of Linear Model and PCR

We expected the linear model to do relatively poorly, as it is an extremely inflexible model and tends to overfit the noise in a dataset with so many observations. We also expected PCA to not be particularly useful, and by extension, PCR, since our dataset has few features to begin with; thus, the dimensionality reduction benefit of PCA is lost for out data. While we guessed they wouldn't do very well, we still found it useful to create these models to serve as a basis of comparison, and to become more practiced with implementing regression models.

5.2 Variable Importance

We found that hour was by far the most important variable in our data, and contributed to the predictions significantly – this is quite reasonable as certain hours (peak hours) will clearly have significantly more bike rentals than other hours. In addition, the weather variables also had a small, but still positive, effect on our predictions. This indicates that weather can indeed help predict demand for bikes.

5.3 Tree-based Models

Two of the three tree based models did very well, namely CTree and RF. Our guess is that they were able to capture the subtle non-linear interaction effects between variables, since tree-based models are equipped to do this.

5.4 Cross-Validation vs Leaderboard Results

We observed that our cross-validation results were generally lower than the leaderboard errors. Therefore, we ended up getting slightly worse results when submitting to the leaderboard than we were expecting. This can be attributed to the fact that the training and test sets are a bit different, and the cross-validation estimates of error can often be a bit biased.

5.5 Regressing on Registered and Casual Separately

Since we are provided with the number of bike rentals that were made by registered and non-registered users (these 2 numbers add up to the total count), we also attempted to regress separately on registered and non-registered users, and add the result together. We expected that this would give us better results, since some of the features may influence non-registered and registered users differently. However, we ended up finding that simply regressing on total count had better cross-validation results (and performed better on the leaderboard). One explanation is that regressing on both components separately introduces the chance for more noise and greater variance, thereby making the predictions worse.

6 Conclusion

Using different combinations of models and featurizations, we found that hour is by far the most predictive feature for our problem, whereas weather variables play a much smaller, but still noticeable effect in contributing to accurate predictions. In addition, decision-tree models – in particular, CTree – were able to best capture the relationships within the data, which led to the best performance on the leaderboard.

7 Future Work

7.1 Using Multiple Featurized Training Sets

We narrowed down which of the 64 featurized training datasets to use by training all of them on a CTree model and using cross-validation to select the dataset with the lowest error. Then we exclusively used this dataset to test all of our models. However, the training set that achieved the lowest error with the CTree model may not be the optimal training set if we were to train with another model. Therefore, we can instead select a featurized dataset that does best for every different model we use.

7.2 Unsupervised Methods

In our project, we mainly used supervised learning algorithms. However, unsupervised methods such as clustering can often provide insight into some of the relationships in the data that can be otherwise missed. Therefore, it could also be worthwhile to perform some initial clustering techniques to identify some of the more meaningful relationships in the data.

7.3 Time-Series Analysis

The given training data only spans the first 20 days of each month, and the predictions are made for the remaining days of the month. In our project, we used the entire training set to make our predictions, although a realistic model should only be using data observed prior to or close to the time of the prediction. In order to usefully apply bikeshare prediction in the real world, using a sort of time-series analysis like this would be more useful for improved generalization.

References

[1] Bill Chappell. Firm buys big bike-share service; expansion and higher rates seen, 2014.

[2] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

[3] Brandon Harris. A simple model for kaggle bike sharing, 2014.

[4] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.

[5] Bjrn-Helge Mevik, Ron Wehrens, and Kristian Hovde Liland. *pls: Partial Least Squares and Principal Component regression*, 2013. R package version 2.4-3.

[6] David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, and Friedrich Leisch. *e1071: Misc Functions of the Department of Statistics (e1071), TU Wien*, 2014. R package version 1.6-3.

[7] Kurt Hornik Torsten Hothorn and Achim Zeileis. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15(3):651–674, 2006.

[8] Greg Ridgeway with contributions from others. *gbm: Generalized Boosted Regression Models*, 2013. R package version 2.1.

[9] David H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.