Hourly Usage Prediction for Capital Bikeshare in Washington,D.C

Hangyu Ren 112878345
Joel McGuire 113423225
Kegan Schifferdecker 112820772

Abstract

This paper shows the process behind how we were able to utilize predictive analytics in creating a model that can predict the hourly numbers for Capital Bikeshare in Washington DC. Predicting hourly usage of the bikeshare is a necessary component of solving the reallocation problem. To create our predictive models, we first prepared the data by checking for missing values, making appropriate transformations to skewed variables, testing if the test set and training set are identically distributed, engineering new features and training our model. Some of the most important features or variables were temperature and hour of the day. The results we got from our random forest model placed us with an out of sample prediction of .50996 RMSLE and put us in the 55th percentile of the competition.

Introduction

Bike sharing programs are services where bicycles are made available for shared use to people on a short time period basis for a price. Bike sharing has become popular throughout recent years due to factors such as a bike being an affordable method of transportation that avoids issues of traffic and parking, or wanting to reduce ones environmental impact. Due to this popularity, there are currently over 800 bike-sharing programs around the world (Fishman, 2016). Bike shares can be good for the places where they are installed, such as in DC where Fishman, Washington, & Haworth, (2014) found that bikeshares significantly replaced automobile use which reduced congestion and pollution in the city. A critical problem for bike-share systems is finding the optimal way to redistribute bikes from destination nodes to embarcation nodes (Chemla et al, 2013, Schuijbroek et al, 2017). While predictive analytics does not directly solve this problem, it will assist with understanding the system that is trying to be optimized.

Problem Definition and Formulation

The scope of the data is that it only includes information for the Capital Bikeshare program in Washington DC and Arlington. It includes two years worth of data from 2011 and 2012. The decision variables that we have for our models are how many bikes are demanded each hour for the whole bike sharing system. The data set that we are working with is an hourly count of bikes needed at a bikeshare location and it includes date, time, weather conditions, and what sort of day (holiday, workday, non-workday) for the columns[1]. We are trying to predict the

[1] A full description of all the variables used may be accessed in the appendix.

total bicycles used each hour. Our modeling approach it to utilize predictive analytics. More specifically we use a random forest method and elastic net method to create predictive models.

The data are rather clean. There doesn't appear to be any missing values and there are no absurd values, such as humidity being negative or wind speed being 100 mph. We had to do some feature engineering and transformation such aslog transforming skewed variables like wind speed and the count of bicycles rented per hour[2].

Datetime needed to be converted into an integer from a date object. Separate variables such as hour, day of week, day of month, and day of year were extracted from the date object. If riding a bike instead of a car is an aspirational activity, research suggests that extracting temporal features that capture the start of new periods (week, month, year) will add accuracy to ridership prediction (Dai et al, 2014). All variables with less than 50 unique values were assigned the data type of categorical.

Since the test data is selected non randomly (it's the last days of the month) there's reason to believe that the test and training data aren't i.i.d. This can be tested by seeing if our prediction of whether an observation belongs to the test or training data is better than random. This technique is called "Adversarial Validation"[3]. We found there was an AUC of .77 using (another) random forest classifier to predict whether an observation was from the test or training set. If the test and training data set were from the same distribution we would expect to see an AUC of .50 with maybe plus or minus .02. Since there's definite separability between the training and the test set it's best for us to train our model on the training data that's most likely to resemble the test data. This is especially pertinent for tree based models which are known to make erratic predictions when the input is outside the range of the values the model was trained on (Louppe 2014).

For our model we chose to use a random forest regressor. A random forest is a popular ensemble model, it's similar to bagged tree model except it's better at reducing variance because it chooses to use a subset of predictors so its individual tree models are not as correlated. It has some nice non parametric properties such as being very robust to outliers and makes no distributional assumption (Louppe 2014). This is in part why we chose not to spend too much time on outlier analysis.

The scoring metric or loss function for the competition is RMSLE, which is like the popular RMSE except it penalizes the underestimation more than the overestimation. It is pictured below.

---

[2] Log transformations and loss functions that include taking a logarithm are generally more prone to bugs in the modeling process due to incapacity to process negative or zero values.

[3] http://fastml.com/adversarial-validation-part-one/.

$$\varepsilon = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(log(pi + 1) - log(ai + 1))^2}$$

Solutions and Discussion

We trained our models models using scikit-learn[4], which is the standard for machine learning in Python. To tune the parameters of our models we randomly generated 100 different values from a uniform distribution and selected the value that minimized the five fold cross validated RMSLE[5]. This led us to an elastic net model with a value of 0.0337 for alpha which may suggest it is better to use Ridge over LASSO regression. It gave us an RMSLE of 1.086 on the validation set as a benchmark. For our Random Forest we arrived at the maximum depth of 90. For max features we went with the default which is the square root of the number of features provided at each nodes, which would be four. Number of estimators, which is the number of trees in the forest was selected to be 2,000. While more trees generally leads to a smaller error because of the ensemble voting, there's a tradeoff with computation. With a forest of 2,000 trees it took 57 minutes to train. On the validation set it scores a .29 which is quite a bit lower than our score on the test set. Our model is likely over fit.

For our random forest model, it is possible for a decision maker to input the variables of interest and get predictions, say if they were looking at demand for the next week they could use the weather forecasts to model what usage on the bikeshare will be. The most important variables are intuitive. Bad weather decreases usage to near zero and time of day by hour is highly related to usage, although these differ depending on the type of the user. Registered users appear to have more inelastic demand for the bike share and the peak hours are the same as the rush to and from work, casual users are much higher on weekends.

Conclusions

To test our model we sent our predictions to be graded by Kaggle for the competition, we achieved an RMSLE score of .50996 which was enough to place us in 1,807th place out of 3,251 teams. This is the 55th percentile, which indicates that while our model isn't spectacular or even in the top half of contestants it's a great improvement on the benchmark of 1.58455, and it's good enough to suggest we actually made a model that stands up to external testing.

We can recommend that the bikeshare operator pays attention to certain factors they are likely already paying attention to, like the weather or if it's a weekend, as those are the variables

[4] http://scikit-learn.org/stable/

[5]

that greatly determine how many bikes will be needed. We can also recommend the use of predictive analytics in solving the problem of how to optimally locate bikes in a bikeshare.

References

Chemla, D., Meunier, F., & Calvo, R. W. (2013). Bike sharing systems: Solving the static rebalancing problem. *Discrete Optimization*, *10*(2), 120-146.

Dai, H., Milkman, K. L., & Riis, J. (2014). The fresh start effect: Temporal landmarks motivate aspirational behavior. *Management Science*, *60*(10), 2563-2582.

Fishman, E. (2016). Bikeshare: A review of recent literature. *Transport Reviews*, *36*(1), 92-113.

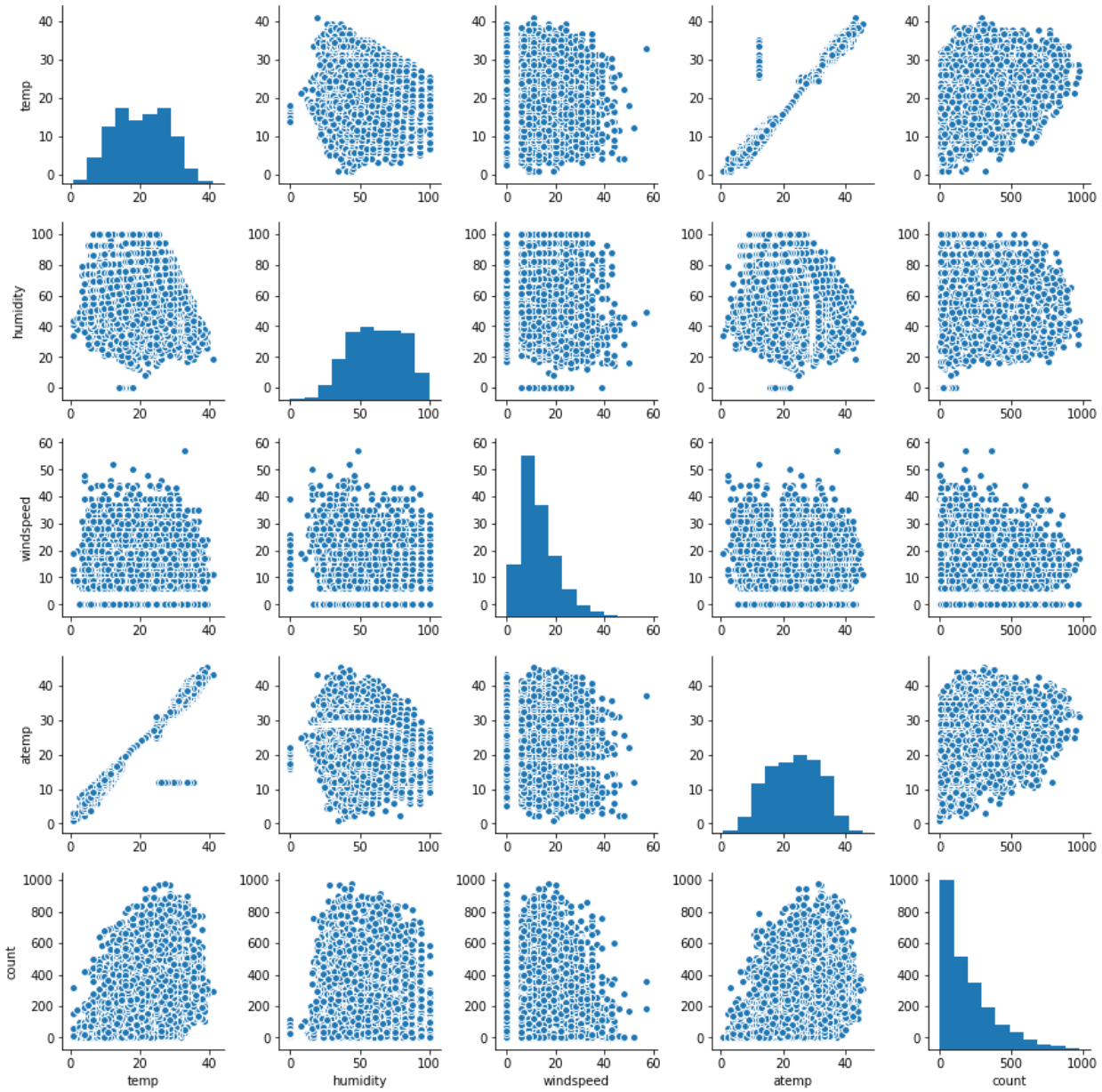
Fishman, E., Washington, S., & Haworth, N. (2014). Bike share's impact on car use: Evidence from the United States, Great Britain, and Australia. *Transportation Research Part D: Transport and Environment*, *31*, 13-20.

Kaggle.com. Bike Sharing Demand. https://www.kaggle.com/c/bike-sharing-demand, 2018.

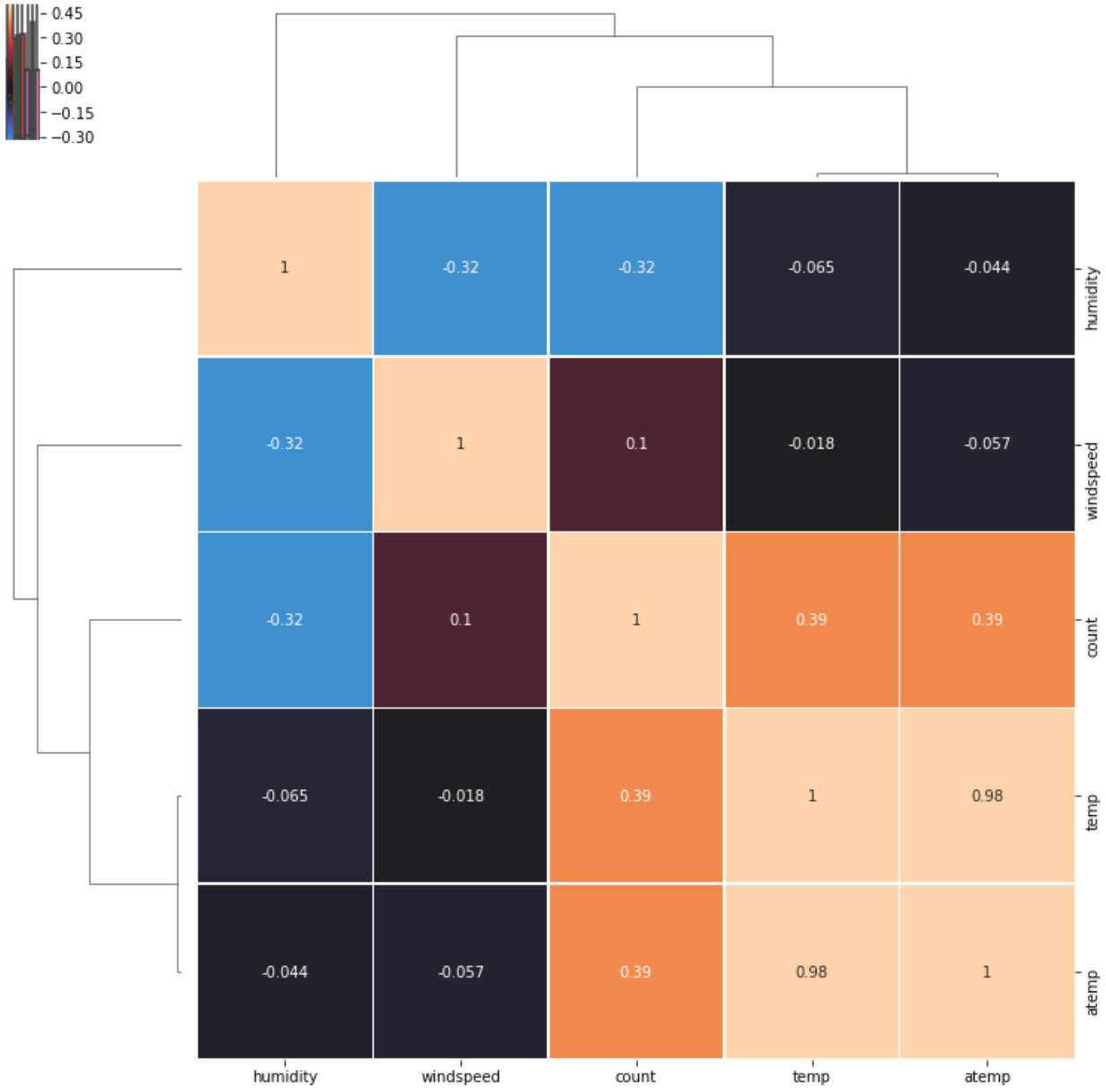
Louppe, G. (2014). Understanding random forests: From theory to practice. *arXiv preprint arXiv:1407.7502*.

Schuijbroek, J., Hampshire, R. C., & Van Hoeve, W. J. (2017). Inventory rebalancing and vehicle routing in bike sharing systems. *European Journal of Operational Research*, *257*(3), 992-1004.

Appendix



Above: Scatterplot matrix with histograms of variables along the diagonal.



Above: Correlation matrix.

Variables or Features

atemp	float64	What the temperature “feels like” in celsius.
count	float64	The number of bicycles rented per hour.
datetime	object	An object containing the hour, day, and year.
holiday	category	Logical. 1 = Holiday. 0 = not holiday.
humidity	int64	relative humidity
is_test	float64	Constructed logical.
p	float64	Probability an observation belongs to test.
season	category	1 = spring, 2 = summer, 3 = fall, 4 = winter

temp	float64	temperature in celsius
weather	category	1: Clear, Few clouds, Partly cloudy, Partly cloudy 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
windspeed	float64	Wind speed.
workingday	category	Logical 1 = work day, 2 = weekend
hour	category	1 - 24.
day.week	category	1 - 7. 1 is monday.
day.month	category	1 - 31.
month.year	category	1 - 12.