Data Science W207.2 Final Project

Bike Sharing Demand

Matthew Holmes

Andrew Kabatznick

Grant Whittington

December 19, 2017

**Introduction/Business Case**

The purpose of this assignment is to apply the data science/machine learning process to a competition from Kaggle, a website devoted to machine learning and related public contests in which the public may try their hand at finding the best solution to a given problem. This assignment allows students to demonstrate the knowledge learned over the term in a complete real-world problem.

As beginning practitioners of machine learning, the team chose a problem with a manageable amount of data, limited factors and a simple well-defined goal. That problem is Bike Sharing Demand, defined in the link below:

<https://www.kaggle.com/c/bike-sharing-demand/data>

The goal is to predict the number of bikes which are rented in a given hour, based on data/time information and weather-related effects. This problem is somewhat of a toy case designed for people to cut their teeth on a data set. This aspect comes with benefits in terms of size and simplicity, but is not well defined in terms of goal or scope, and in some ways is ill defined to the type of techniques used in this course. For example, the scoring criterion is given, making model comparison easier and well defined. That equation defines RMSLE (Root Mean Squared Logarithmic Error) and is given below. As an error, the RMSLE is desired to be as small as possible for a good model.

Where pi, ai are the predicted and actual values for a test dataset, and n is the size of that dataset. Log refers to the natural log.

To make the problem more well suited to the project, a notional problem definition will be given, and the team will seek to solve that problem. When the actual problem conflicts with the notional one, the team will discuss what route was chosen, why, and what would have been done in a more ideal scenario.

The notional scenario involves a bicycle rental company which has been around for a few years and is looking to optimize their bike rentals over various sites in a city. From day to day, different amounts of bikes are rented based on a number of factors, and accurate prediction of that number will allow for bikes to be redistributed among sites, leading to a higher use rate and more profits. The initial dataset will be that of the rentals per hour at the site, and corresponding date/time information and related weather factors. After the initial model is created, “on-line” features will be implemented, allowing new data to be used in prediction of future data.

This scenario differs from that of the Kaggle competition in a few key ways. One, the Kaggle competition requests that test data only be predicted based on previous training data from a temporal standpoint. As such, it makes its “test set” the times relating to days at the end of the month, and its training set the days at the beginning of the month. Such a time-based request is outside the scope of this course and would require techniques beyond those covered. As such, that request will be ignored and the data will all be considered in the past and fair to use as one large dataset. The on-line aspect will be handled by using a subset of previous data as a factor to predict future data. This assumption has one downside in that aspects of the dataset are removed, making some methods or ideas difficult if not impossible to implement. This will be discussed more in future sections, such as factor generation.

Finally, all modeling will be done in an ipython jupyter notebook using various libraries imported from scikitlearn, pandas and matplotlib.

**Data Understanding (EDA)**

*Leaving this section to someone else*

**Data Preparation**

COME BACK AND ADD A BETTER TRANSITION STATEMENT BASED ON WHAT IS IN THE ABOVE SECTION

The data used for this problem is limited in scope and in a structured format of a CSV file. As a result, no special accommodations are needed to load it into the jupyter notebook. It is important to note at this time again that the training data is missing values for days at the end of the month, which will affect feature selection and generation. Those issues will be discussed in more detail below when applicable.

**Data Cleaning/Modification**

Some features in the training data set are in a format not immediately usable for machine learning techniques. Date/time is a string which contains the year, month, day and hour information in a single feature, and is split into a separate feature for each of those variables in integer format. Next, categorical variables for season and weather are one-hot encoded into logical features for the corresponding categories.

**Feature Generation**

There are a few options for implementing extra features based on the existing set, but the immediately obvious one is using the actual number of rentals from previous time periods as a predictor in the current time period. This not only helps satisfy the condition of having an “on-line” model, but also makes logical sense from a consumer standpoint. It is hard to predict based what will make a user more likely to rent a bike, but knowing that X number of bikes were rented in the last hour/day/week etc is likely to capture other effects not in the available features making the model stronger. The great downside of features of this type is that their implementation is limited due to the incomplete nature of the data. It is impossible to know how many bikes were rented in the previous day, or corresponding hour of the previous week when that data is not available. As such, the only feature added of this flavor is the number of rentals in the previous hour. This will not be available for the first hour of the first day of a new month, but the number of instances of this type are limited, 12 in a given calendar year, and the number of bike rentals during the early hours of the day are limited.

It is important to note that this new feature has limited practical application. It is not feasible to reallocate supply based on the demand in the previous hour. This feature is more of a proof of concept to demonstrate how such data can be used, and in a real-world instance with more complete data would be expanded. Also of note is how this type of “on-line” feature affects the model. It is not expected that the model be re-trained every time additional data is added, but rather that new data be implemented as a feature into the existing model. Perhaps once a month or other less frequent retraining of the model will be useful, but that would be another aspect of model creation to study and will not be included in this exercise.

**Modeling**

The nature of the problem dictates use of a supervised learning method with regression rather than classification output. These conditions lead to the following options as possibilities:

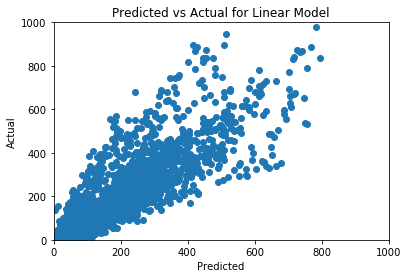
* Linear Model
* Bayesian Model
* Tree Model

Before any modeling can occur, the data set must be split into training, test and dev (where applicable). Based on each set being a desired percentage of the overall data, a random split was used, rather than using the first X % of the data for training as an example. This was done to more accurately capture the time component of the data and not overfit aspects. After some testing, a good split was found using 80% training data and 20% test data.

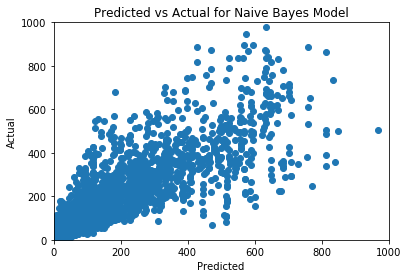
**Initial Model Exploration**

A model of each type was created with the standard settings, in order to get an initial idea of quality. In depth discussion of the creation of these models will not be discussed, only why they were or were not suitable to further investigation.

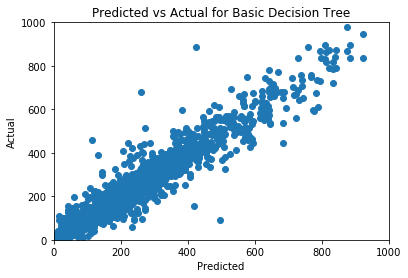
The linear model does not perform well. The limited number of factors does not lend well to effectively capturing the variability in the output. In addition, the model predicts some negative outputs, which is not possible, but also does not allow for use of RMSLE as an evaluation criterion to the quality of the model. Below is a plot of the actual vs predicted counts of the test data using the linear model. The most obvious aspects of this plot are the widening variance in results as the number of counts increases, and the fact that the model grossly underestimates the counts in many instances. Ideally the mean line of the data points would have slope one, but this is far from the case. In fact, the predicted counts never exceed 800, while actual accounts do to a reasonable number. For the above reasons the linear model is not selected for further study.



The Naïve Bayes model performs better than the linear model but still leaves some aspects to be desired. A plot of the predicted vs actual counts is given below. It improves upon the linear model but no longer predicting negative numbers of rentals, and also predicts counts in a range closer than actual values. However this comes at a cost of having more outliers than the linear model (greatly mis-predicted values), and still having a similar “cone” of variance. The RMSLE for this model on the test data is ~.71. For the above reasons this model was not selected for further study.



The tree based model by comparison preformed well. A plot of the predicted vs actual is given below and shows improvement in the variability of the predictions across the entire space. There are some outliers, but still fewer than that of the Bayesian model. There still appears to be some skew toward underpredicting counts, but that is less visible in this model than the two previous. As such this type of model is selected for further study and will likely see improvement. For comparison, the RMSLE value of this model on the test set is ~.375, showing market improvement over the Bayesian one.



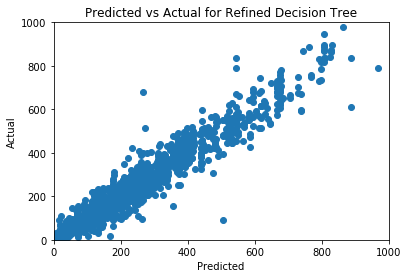
**Model Refinement**

In order to improve the model above, various options in the creation of the model are explored in order to find the best mix of model variability and bias. In laymen’s terms we want the model to have enough modeling power to capture the important features of the dataset, without having so much that it overfits and is poor on the test case. A grid search was performed over the max\_depth and max\_features areguments of the tree. By implementing the RMSLE equation as a custom scoring function, the search is insured to choose the best model for the evaluation of this problem. This technique also has another attractive quality, and that is the use of cross validation. This allows us to use the training data as both training and development without overfitting, allowing for a larger chunk of the total data to be used as training.

The result of these found the following factors as the best model:

* max\_features: 18
* max\_depth: 10
* number of folds (cv): 7

A plot of this model on the test set is given below. This model is similar to the unrefined tree model, but appears to be more compact around the mean line with fewer outliers. There is still some noticeable skew toward under prediction of values. The RMSLE value for this model on the test set is ~.332, showing improvement on the previous tree model.



Before moving on the issue of factor selection will be addressed. It is a common practice in modeling to select the most important factors to be used in prediction in order to reduce the effect of overfitting. For this problem, there are relatively few factors and none of the models appear to over predict aspects of the test cases. If anything, the models could use more variability in some cases. As such factor pruning and other similar techniques were not investigated.

**Evaluation and Deployment**

As discussed above, the best model found was a tree based one, which was further refined using grid searches over an array of parameters. Cross validation allowed for a larger amount of the data to be used in the creation of this model, while also protecting from overfitting. The advantages of this model are accuracy and greater level of predicted variance compared to the Bayes and Linear models. The downside of using a tree based model is that there is less clear meaning into the model. Implementation is easy however, as simple branching checks on factors will yield a final answer.

A feature of all models is the tendency for the predicted number of rentals to be lower than the actual ones. This fact may be more advantageous compared to overpredicting the actual counts, as this would mean that the number of bikes would more likely being too few rather than too many, indicating that all the bikes are rented. However, this may cause some customer dissatisfaction if many people who would otherwise rent a bike cannot. Either argument has merit, but it is important to note this detail.

Overall, the model generated does a reasonable job of predicting the complicated problem with a relatively low number of factors available. A more complete initial data set would allow for more factors to be generated and could only improve future models. In the notional sense, this model would be valuable to be able to predict and reallocate assets to various locations, ensuring that the percentage of bikes rented (and therefore profits) are as high as possible.