Predicting Daily Demand in a Bike Sharing System

APMA 990 Final Project

Natalia Iwanski
April 23, 2014

1 Abstract

This project applies machine learning techniques to bike sharing data obtained for the city of Washington, DC. Bike sharing is an automated bike rental system in which rental stations are available 24/7 at numerous locations throughout a city. Users can borrow a bike at one station and return it at another within a given time frame. For Washington DC, data is available for each trip taken specifying its duration and location. This information is aggregated to a daily level to give a count of the number of trips taken each day over a three year period by both registered and casual users. K-nearest neighbour, linear regression and decision trees are all applied to the data to predict counts for randomly sampled test points over the three years. Linear regression with subset selection is found to give the lowest prediction error, however all three methods are able to detect time-dependent patterns in the data quite well. To obtain even better results, methods from time-series analysis can be applied, and also more data is necessary as the system is still relatively new and growing quickly.

2 Introduction

Bike sharing systems are growing in popularity throughout Canada and the United States because they provide congested cities with a convenient and affordable mode of transportation. The systems make biking more accessible as well as providing a sustainable travel option that reduces gas emissions [1]. Due to the fact that many bike sharing companies have to work closely with local municipalities and transportation authorities, there has been a trend to make bike sharing data publicly accessible to promote its growth and popularity. The bike sharing system within Washington, DC specifically, is run by a company called Capital Bikeshare. It has been operating trial bike rentals since 2008, with a full network of bikes being installed in 2010. Since then stations have also been extended to outlying suburbs, amounting to 200 stations in total, and over 1800 bikes [9]. Automated rentals are available 24/7 at each station, where casual users can purchase 24-hour or 3-day rentals, while longer term memberships can be purchased online by registered members. Rentals are only available for thirty minutes at a time, with longer trips incurring additional costs. Hence many users drop off a bike after thirty minutes and pick up a different one to continue to their destination.

Capital Bikeshare provides public access to their data through their website [6], including information about trip history, and station information. The trip history data provides information about every bike rental made within the system from September 2010 to the present, while the station information provides basic location and capacity attributes. For the purposes of this project trip history data is aggregated to a daily level, listing the number of registered and casual users that went through the system for any given day from January 1, 2011 to December 31, 2013. This is combined with station capacity information, daily weather conditions, and information regarding whether a particular day is a work day, weekend or holiday. Different methods are then applied to the data to predict the number of casual and registered users expected for a particular day. Counts of casual and registered users are predicted separately, due to the fact that they may use the system for different reasons. As a result different variables may have to be used for prediction. K-nearest neighbour, linear regression techniques and regression trees are all applied to the data and compared in regards to performance and prediction accuracy.

3 Data

Four different data sources were combined to make the final dataset, including trip history data, station information, holiday information and weather data.

- **Trip History Data:** contains the date, start time, end time, start station and end station for each trip made between 2011 and 2013. It also provides the type of member that made the trip and the specific bike ID. This was obtained from Capital Bikeshare's website [6].

- **Station Information:** contains the ID and location of each station in Washington and the surrounding area. It also provides the date each station was added (or removed from) the system, as well as its capacity, or the number of bike docks it has. This was also obtained from Capital Bikeshare's website.
- **Weather Data:** contains daily weather information between 2011-2013, including the average temperature, average visibility, average wind speed, the total amount of precipitation and a ranking of cloud cover. This was obtained from a website providing historical weather data called Weather Underground [7].
- **Holiday Data:** contains the dates of all statutory holidays in the District of Columbia between 2011 and 2013, obtained from [8].

All four data tables were joined in Micosoft Access to produce a final table with three years worth of daily data (1096 days) with the following input and output variables (categorical variables are marked with a $*$):

Input Variables	**Description**
Related to date:	
Year$*$	2011, 2012 or 2013
Day of the Week$*$	Categories 1-7, representing Sunday- Saturday
Holiday$*$	Coded as 1 if a holiday, 0 otherwise
Workday$*$	Coded as 1 if a workday, 0 otherwise
Season$*$	Coded as 1=winter, 2=spring, 3=summer, 4=fall
Related to station capacity:	
City Capacity	Total number of docks available at all stations in Washington DC.
Outskirts Capacity	Total number of docks available at all stations outside of Washington DC.
Related to weather:	
Mean Temperature	The average daily temperature in °C
Mean Visibility	The average daily visibility in km
Mean Wind Speed	The average daily wind speed in km/h
Precipitation	Amount of precipitation in mm
Cloud Cover$*$	Categories 0-8, 0=clear sky, 8=sky is not visible
Output Variables	**Description**
Registered Users	Number of trips made by registered users in one day
Casual Users	Number of trips made by casual users in one day

The training and testing sets were created based on a $70\% - 30\%$ split of the data. First, the data was kept in chronological order and 70% was chosen from 2011-2013 for training and 30% was chosen from the last part of 2013 for testing. For comparison, the data was also randomized, and the training and testing sets were chosen randomly based on a 70%- 30% split.

4 Initial Data Analysis

The output variables were examined first to get a sense of the patterns in the data. Figure 1 plots the counts of registered and casual users over all three years, or 1096 days. There is a strong relationship between time and trips taken, with bike demand peaking in the summer and dropping in the winter months. The counts also cover an extremely large range. Registered trips range from 20-8500 per day and casual trips range from 2-5300. After analyzing the data more, it was found that the counts of 2 and 20 correspond to the day that Hurricane Sandy hit the city, which was October 29, 2012.

Removing this outlier from the data, changed the ranges to 402-8500 and 9-5300 for registered and casual users respectively. This large range was anticipated to make predictions more difficult. Another factor was the growth of the system. It is clear from Figure 1 that the popularity of the bike sharing system grew from 2011-2013 and is likely still growing. Table 1 and Figure 1 show the yearly averages for both registered and casual users and we see that each year experiences significant growth.

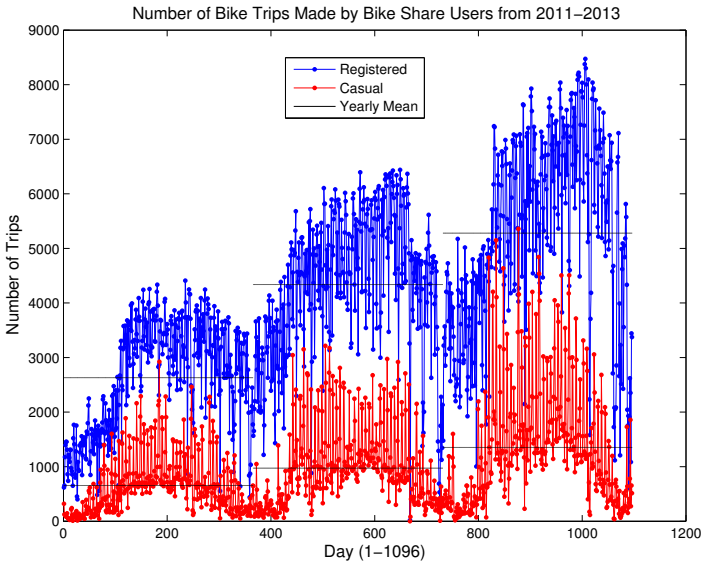


Figure 1: Number of trips made by registered and casual users from January 1st, 2011 to December 31st, 2013. We can see a clear relationship between time and trips, as well as significant yearly growth as the system becomes more popular. The yearly averages for each group of users is plotted in black.

	Yearly Averages		
	2011	2012	2013
Registered	2630	4335	5281
Casual	652	975	1355

Table 1: Yearly averages for registered and casual users.

Initially data for the three years included all trips from Washington, DC and the surrounding suburbs. However since entire cities were added in 2012 and 2013 this led to even more variation in the data and to inaccurate results. As a result the data analysis was restricted to trips beginning or ending in Washington. This still captures commuters coming from the suburbs into Washington, but removes all traffic between suburbs.

4.1 Average Trend

Due to the strong relationship between time and trips seen in Figure 1, an initial method using only time as input was applied. The training and testing sets were kept in chronological order and an average trend model was used to predict test points in 2013, similar to a method applied in [1]. For each test point in 2013, two data points were found in 2012 and 2011 corresponding to its month and day values. The difference in counts between these two days was then added to the count in 2012 as a prediction for 2013. Using this method was expected to capture the growth from year to year. Results are shown in Figure 2. We see that there are large prediction errors due to the fact that the

growth rate is not constant from year to year, and hence the rate between 2011 and 2012 is not the same as 2012 to 2013. Also, the model is more accurate for the registered users since they tend to make more regular trips regardless of factors such as weather. There is a great deal more prediction error for casual users because these trips are usually more unpredictable and dictated more by good weather and holidays.

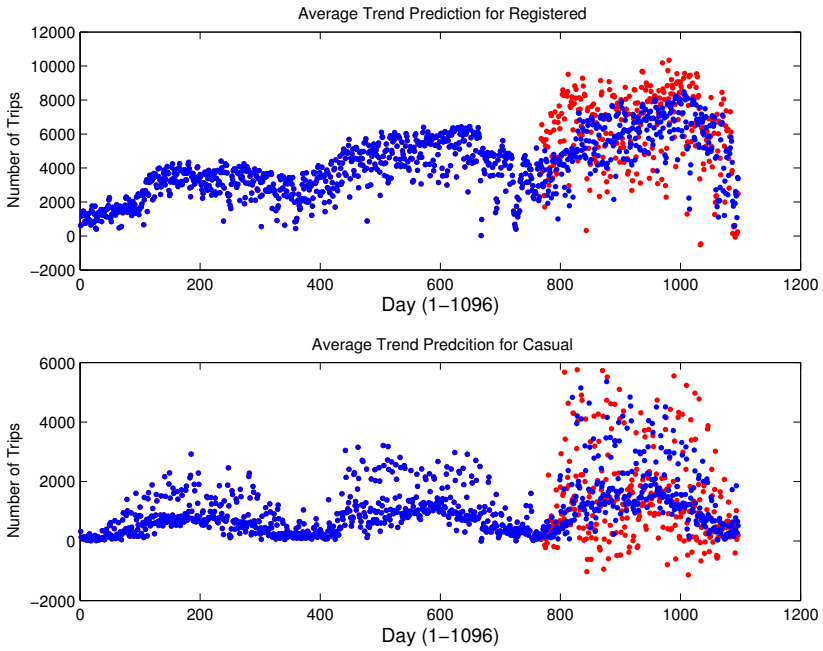


Figure 2: Actual and predicted data points using the average trend method. Mean squared error (MSE) for registered users was approximately 5,406,700, with an average absolute difference of 1836 trips between actual and predicted data points. For casual users MSE was 1,808,900, and the average difference was 1023 trips.

Consequently to obtain better predictions, more input variables need to be included. Also, because of the different growth rates, any method applied to the data with chronological training and testing sets will likely under or over predict. As a result randomized training and testing sets are used for all remaining methods in the project. Including more variables and randomizing the data allows methods such as k-nearest neighbour or linear regression to be applied. The following sections show results for these two methods and also regression trees.

5 K-Nearest Neighbour

The first method applied to the data is k-nearest neighbour (k-nn). For each observation x in the testing set, T_{test}, the method finds the k closest points to it in the training set T, denoted by $N_k(x)$ [2]. The predicted value $\hat{Y}(x)$ is then the average of the k output points, y_i, corresponding to the k-closest points. This implies $\hat{Y}(x)$ is defined as follows:

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i \tag{1}$$

Hence a crucial element of this method is the distance metric used to find the k-closest points to the test point. For purely numerical variables, Euclidean distance is used, however for categorical

variables, a different metric is typically applied, called Hamming distance. This requires categorical variables to be recoded into dummy variables first. A variable taking on G distinct values or categories, is represented as a G binary vector, with each component corresponding to a category, $g \in \{1, \ldots, G\}$. A component is equal to 1 if an observation corresponds to the gth category, and 0 otherwise. The Hamming distance between two binary vectors, v and w is then defined as the number of elements that differ between the two vectors, or

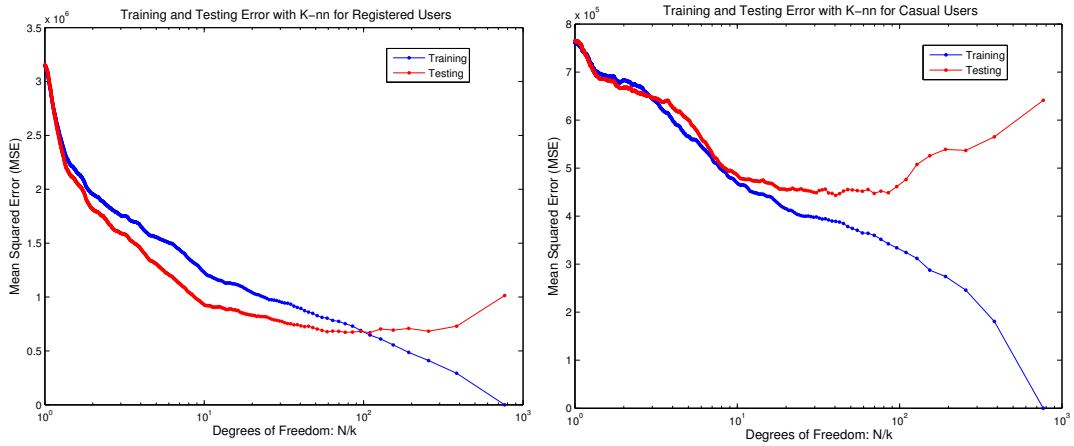
$$d(v, w) = \sum_i |v_i - w_i| \tag{2}$$

In Matlab, Hamming distance is defined as the percentage of elements that differ, so (2) is divided by the total number of elements in v or w.

For a combination of both numerical and categorical variables, defining a distance measure is more complex and is the subject of ongoing research [4], [3]. For this project, Euclidean distance is used for numerical attributes, Hamming distance is used for categorical attributes, and the two are combined into a single measure with a weighted sum. So for two input vectors x_1 and x_2, with n numerical components and m categorical components the distance metric is given by

$$d(x_1, x_2) = \frac{n}{n+m}\left(\sum_{i=1}^{n}(x_{1i} - x_{2i})^2\right)^{\frac{1}{2}} + \frac{m}{n+m}\frac{\sum_{i=1}^{m}|x_{1i} - x_{2i}|}{n+m} \tag{3}$$

Figure 3 shows the training and testing errors for both registered and casual users. The optimal value of k for registered users is found to be 7, which gives a mean squared testing error (MSE) of 669,810. Due to the fact that MSE can be difficult to interpret, the mean absolute error (MAE) is also calculated to give a better sense of how much predictions deviate from actual values. MAE for registered users is 623, so on average predictions are off by 620 trips. For casual users the optimal value of $k = 19$, MSE=443,190 and MAE=447.



(a) (b)

Figure 3: Training and testing mean squared error for registered and casual users. The optimal value of k for registered users is 7 and 19 for casual users.

Predictions are plotted with actual values over time in Figure 4 and also sorted according to trip size to see where the method gives the most error. K-nearest neighbour is able to detect the time-dependent pattern in the data very well, and is also able to capture the growth in the data. For casual users, the method is unable to predict the more extreme values, due to the fact that there may be other factors involved with casual bike use. Since k-nn averages over data points, it may be hard to

find values that are similar in terms of day of the week, holiday and weather which are likely the key predictors for casual use. Holiday dates change from year to year, and the weather is highly variable on these days as well. In general the method performs well for registered users but it greatly over and under predicts for several values. This could be a result of the variability in the data itself. Since the system is relatively new there are large swings evident even for days in the same month or week which probably influences the averages. As a result, a method which does not rely on taking averages may perform better, such as linear regression.

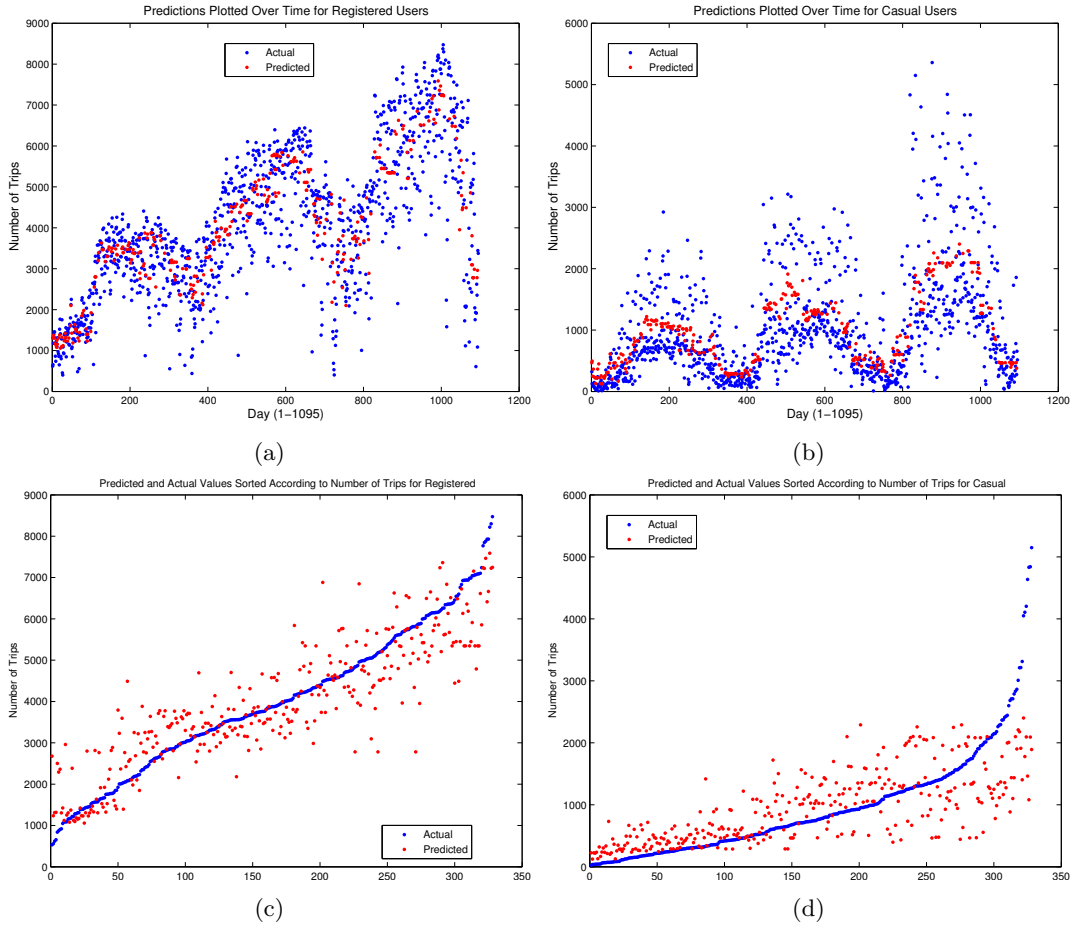


Figure 4: Predicted and actual values plotted against time in (a)-(b) and sorted according to trip counts in (d)-(e).

6 Linear Regression

Linear regression fits a linear model to the data by least squares. So given a matrix of input vectors, X, it predicts the output Y, via the model

$$\hat{Y} = X^T \hat{\beta} \tag{4}$$

where $\hat{\beta}$ includes the value $\hat{\beta}_0$ for the intercept term, and X has a column of ones. The coefficients β are found by minimizing the residual sum of squares for the model given by

$$RSS(\beta) = \sum_{i=1}^{N} \left(y_i - {x_i}^T \beta\right)^2 \tag{5}$$

which gives

$$\hat{\beta} = \left(X^T X\right)^{-1} X^T y \tag{6}$$

Finding a unique solution to the linear system in (4) depends on the linear independence of the input vectors in X. Results are unreliable when the columns of X are highly correlated. Building a correlation matrix of the input vectors in Table 2, we obtain the following values:

	City Cap.	Out. Cap.	Temp	Vis.	Precip.	Workday	2011	2012	2013	Sunday	Saturday	Summer	Cloud=8
City Cap.	1												
Out. Cap.	0.9	1											
Temp			1										
Vis.				1									
Precip.				-0.5	1								
Workday						1							
2011	-0.8						1						
2012							-0.5	1					
2013	0.8	0.8					-0.5	-0.5	1				
Sunday						-0.6				1			
Saturday						-0.6					1		
Summer			0.7									1	
Cloud=8				-0.5									1

Table 2: Correlation matrix for the input variables in the model. Only correlations greater than or equal to 0.5 are shown.

We see that city and outskirt capacity are highly correlated, since in most cases the addition of stations in the suburbs is also followed by the addition of stations or docks in the city. City capacity is also highly correlated with the years 2011 and 2013, which correspond to the years that had the smallest and largest number of bike docks. Mean temperature and the summer season are highly correlated, as well as precipitation,cloud cover and visibility. There is also a high correlation with workday, coded as 0 and 1, and the two weekend days Saturday and Sunday.

Due to these correlations the resulting matrix X is "close" to being rank deficient, which also produces a warning in Matlab. As a result methods which perform variable selection, or shrinkage methods need to be applied in order to reduce or eliminate the effects of these correlations. Four different regression models are applied including forward stepwise selection, ridge regression, lasso regression and a subset selection technique based on p-values and the F-statistic. All inputs are standardized so that results from the different models can be easily compared.

6.1 Forward Stepwise Selection

Forward Stepwise Selection (FSS) is a method for subset selection which tries to reduce the number of variables in a model, implying that it reduces variance. However, because it makes predictions with less variables, it also introduces more bias. The goal of subset selection is to find a good trade-off between the bias and variance to improve the results of linear regression. The model initially only contains the intercept term, and then at each step, the variable which reduces MSE or RSS the most is added on. When used on training data, this process will continue until all of the variables are back

in the model because including more predictors begins to overfit the model, reducing the error as a result. Hence cross-validation needs to be applied to determine the optimal number of variables.

Results of applying FSS and cross-validation to the bike share data are shown in Figure 7 and 8. The optimal number of variables for registered users is 7, which corresponds to city and outskirt capacity, mean temperature, mean visibility, precipitation, workday, and season. For casual users, city capacity, temperature, workday and season are found to be important predictors. The resulting values of β, MSE, and MAE are shown in Table 3. FSS is able to give a much better prediction error than k-nn, but compared to other regression methods, it does not perform quite as well. Predicted values are plotted over time and against actual values just as for k-nn, however since results for all regression methods are very similar, only the plots for the lasso are shown for registered users. For the most part lower trip counts tend to be over approximated, and higher trip counts are under approximated. For casual users there are more apparent differences. FSS tends to under predict the low trip counts, making many values in 2011 negative (Figure 8). This greatly contributes to the prediction error.

6.2 Ridge Regression

Ridge regression is classified as a shrinkage method, because it does not set any coefficients equal to zero, but instead shrinks coefficients towards zero if they are not as important in prediction. It imposes a penalty on the size of the coefficients, minimizing

$$\hat{\beta}^{ridge} = \underset{x}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} {\beta_j}^2 \right\} \tag{7}$$

Since no variables are eliminated, results are more difficult to interpret, especially if there are many variables. However, for casual users, ridge regression gives the lowest prediction error. Hence removing variables from a model is not always an improvement. Although there are lots of related variables in the bike sharing model such as cloud cover, visibility and precipitation, they all work towards reducing the error in the predictions. Examining Figure 8(e), we see that ridge regression is better able to predict lower trip values, and does not have the same problem as FSS in predicting negative values. Interestingly, ridge regression gives the highest prediction error for registered users, further confirming that the two groups of users exhibit different patterns and need to be kept separate when predicting their bike use.

6.3 Lasso Regression

Lasso is very similar to ridge regression, in that it also penalizes the size of coefficients, but due to the fact that it uses the 1-norm, it is able to set coefficients to zero. It minimizes the expression

$$\hat{\beta}^{lasso} = \underset{x}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\} \tag{8}$$

Larger values of λ push more coefficients towards zero, which can be seen in the profiles of the lasso coefficients in Figure 5. Only the profiles for registered users are shown because the plots are hard to interpret for the 31 variables in the model. Also, as λ approaches zero, the coefficients approach the values given by ordinary least squares. To find the optimal λ, cross validation can be applied, which is shown in Figures 7 and 8.

For registered users, lasso is able to give better results than FSS or ridge. It keeps several more variables in the model, such as holiday, particular days such as Sunday and Wednesday, and two indicators of cloud cover. However since these are categorical variables and are specified by a group of values, it would be beneficial to keep them together as the lasso algorithm proceeds. Keeping all 8 cloud cover variables together, or all four season variables, would allow the model to determine

whether season as a group is important or not. In [2], Hastie et al. mention that this is possible to do with an algorithm called grouped lasso, developed by Yuan and Lin [5]. The model is able to keep dummy variables together, shrinking and selecting the members of a group together. There are implementations of the algorithm in R, however there is no available code for Matlab, and the algorithm itself is not so straightforward to implement.

For casual users, lasso gives almost the same prediction error as ridge (Table 3), showing that different methods and different groups of variables can still give very similar predictions.

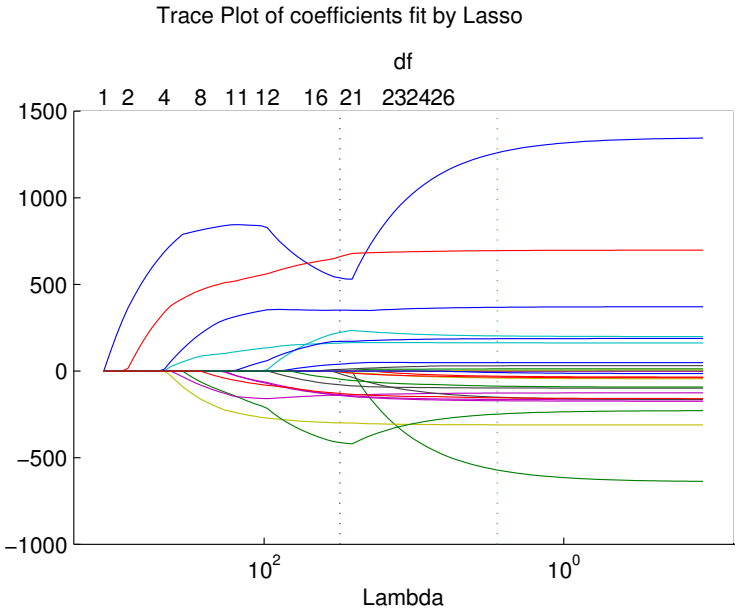


Figure 5: Profiles of lasso coefficients vs. values of λ for registered users

6.4 Matlab's `stepwisefit`

Since results from FSS left fewer variables in the model compared to lasso, another subset selection technique was applied to see how it differed between FSS and the lasso. Matlab has a built in subset selection technique called `stepwisefit`. Initially it begins with a constant model and then at each step variables are added or removed one by one based on their statistical significance. To test the statistical significance of a single variable, a z-score is typically computed, given by

$$z_j = \frac{\hat{\beta}_j}{\hat{\sigma}\sqrt{v_j}} \tag{9}$$

where $\hat{\sigma}$ denotes the estimated variance of β_j and v_j denotes the jth diagonal element of $(X^T X)^{-1}$. A z-score outside the $p = 0.05$ or $p = 0.95$ level is considered statistically significant which implies the null hypothesis of $\beta_j = 0$ can be rejected. For groups of variables, the F-statistic is used as a test of significance. It is given by

$$F = \frac{(RSS_0 - RSS_1)/(p_1 - p_0)}{RSS_1/(N - p_1 - 1)} \tag{10}$$

where the subscripts 0 and 1 represent the smaller and bigger models being compared, with p_0 and p_1 parameters. At each step the algorithm computes the p-value of an F-statistic that results in either adding or removing a variable from the model. Hence a variable can be added in one step and removed in another. Because all possible combinations of variables are not tested the process computes a locally optimal subset, not necessarily global.

For the bike sharing data the algorithm gives the lowest prediction error for registered users. For casual users it also performs well, but not as well as ridge or lasso. In terms of the number of variables it includes, it seems to find a balance between the results of FSS and the lasso, including slightly more than FSS, but slightly less than the lasso. The steps of the algorithm and their associated MSE are shown in Figure 6. Plots (a) and (b) show which variables are included in the model at each step.

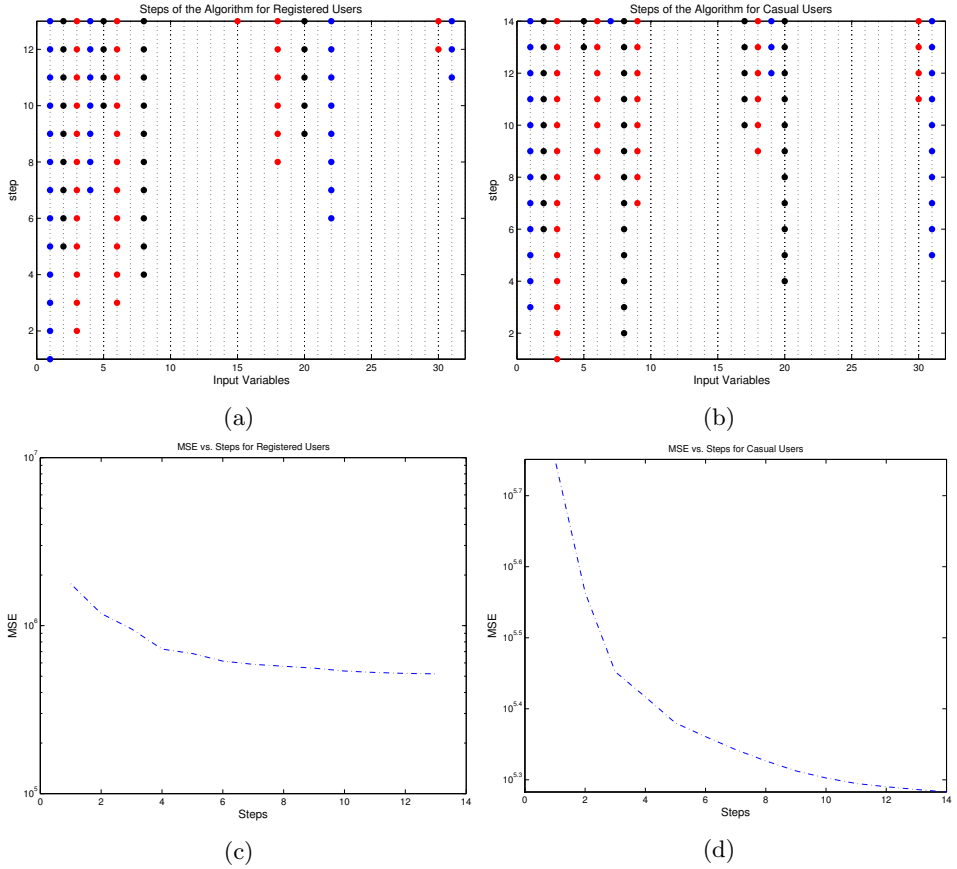


Figure 6: Steps of the `stepwise fit` algorithm for registered users ((a), (c)) and casual users ((b), (d)).

6.5 Discussion

Although there is a great deal of variation amongst the regression models, several key variables do emerge as good predictors for each group of users. For registered users city capacity has a strong positive coefficient, while outskirt capacity is usually negative. Temperature, and visibility give strong positive coefficients, while wind speed and rain have negative coefficients. Workday gives a large positive coefficient which supports the intuition that registered users are usually commuters. Spring and fall also seem to give significantly large positive coefficients maybe due to the fact that they see the most change in biking behaviour. For casual users, city capacity remains an important predictor, and

workday switches to a large negative coefficient, which confirms the idea that casual users typically use bikes for recreational use on weekends and during vacation season. The fact that winter has a strong negative coefficient further supports this idea, however holidays do not appear to be a good predictor.

Results for categorical variables are more difficult to interpret as usually only one or two categories are found to be important and the rest are set to zero. Using an algorithm such as grouped lasso would be beneficial as it would be able to indicate whether the season or cloud cover in general are important predictors or not. Another alternative would be to recode the categorical variables into binary variables. Since winter has a strong negative coefficient in some of the models, season could be coded as 1 for winter and 0 for all other seasons.

Since cross validation is applied to all of the models, results can change depending on the randomly chosen folds. But overall linear regression provides much better results than k-nn and for certain models, it is also able to better predict the lower and higher trip counts that k-nn had trouble with.

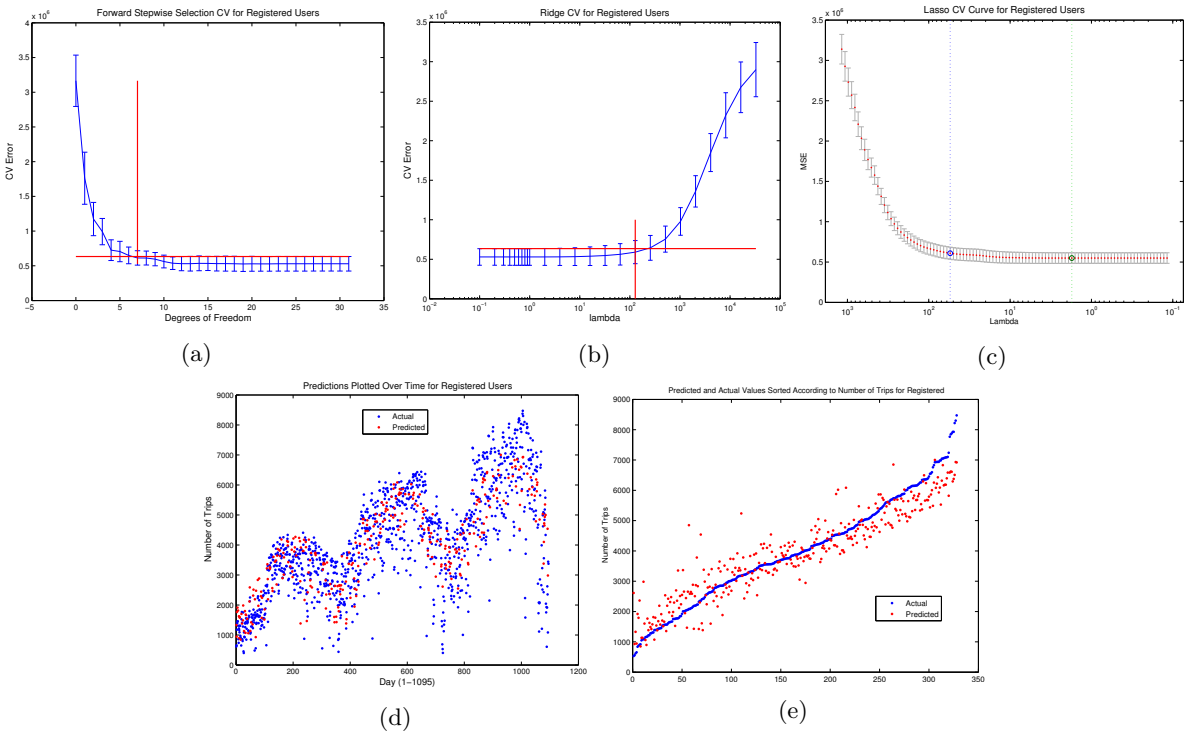


Figure 7: Cross Validation curves for FSS (a), Ridge (b), and Lasso (c). The optimal value is denoted by a red vertical line in (a) and (b) and by a green vertical line in (c). Predicted values are plotted against time in (d) and sorted according to trip size in (e) for Lasso regression. All figures refer to registered users.

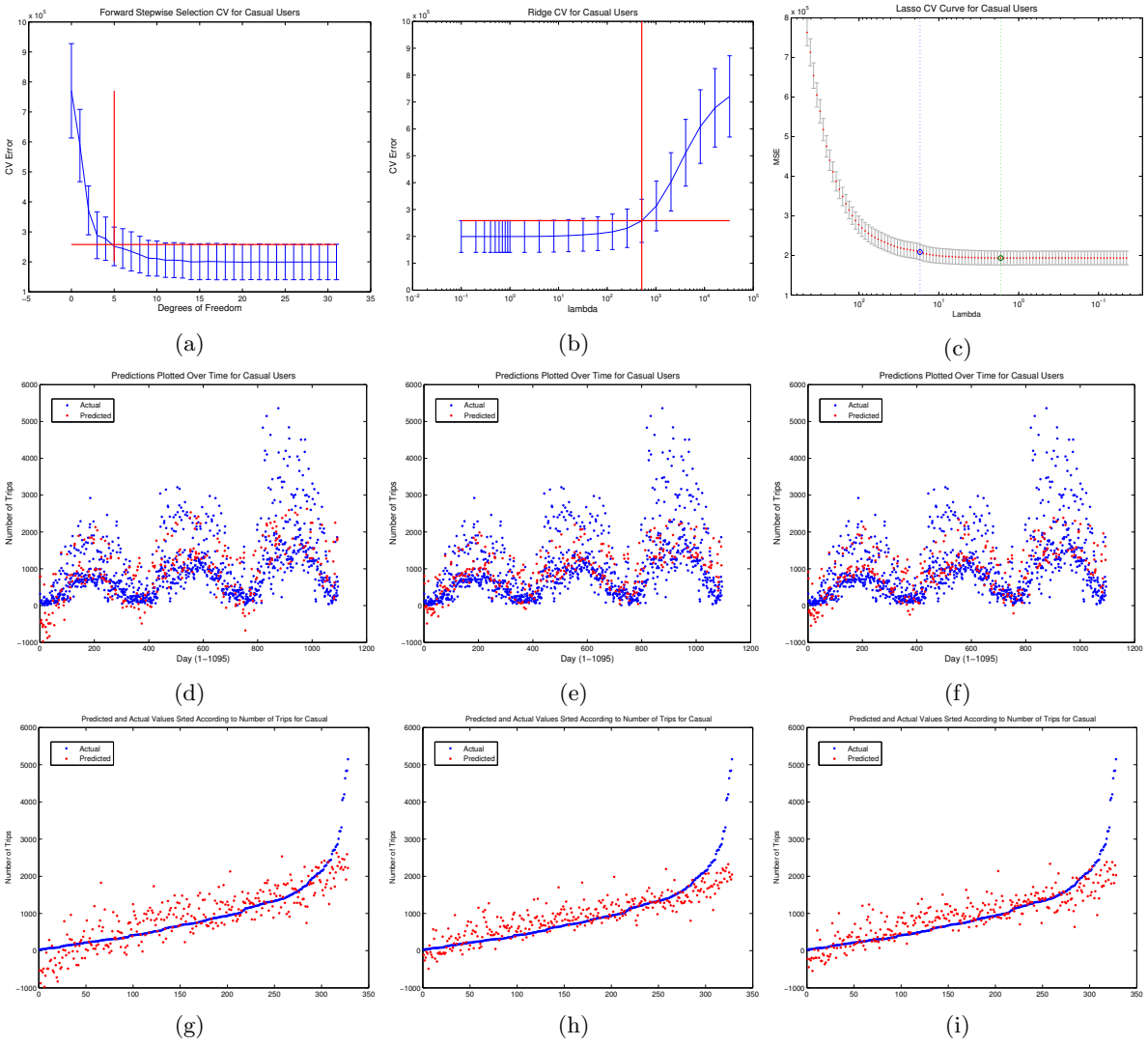


Figure 8: Cross Validation curves for FSS (a), Ridge (b), and Lasso (c). The optimal value is denoted by a red vertical line in (a) and (b) and by a green vertical line in (c). Predicted values are plotted against time ((d)-(f)), and sorted according to trip size in ((g)-(i)) for Lasso regression. All figures refer to casual users.

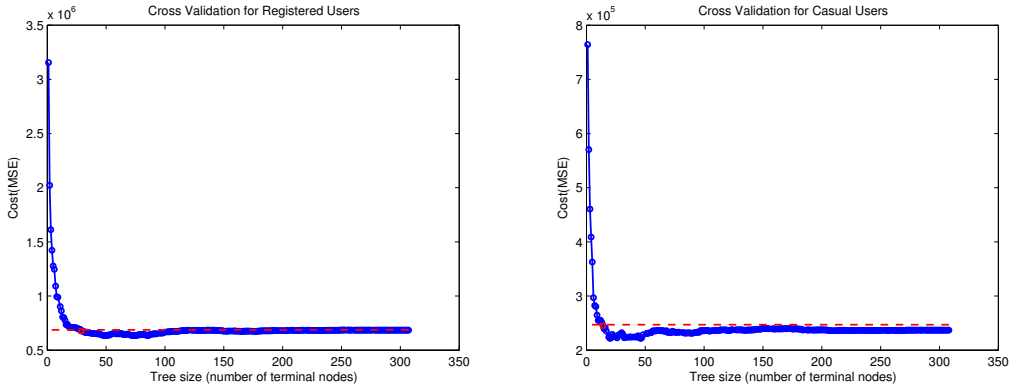
	Values of β for Registered Users				Values of β for Casual Users			
Variable	FSS	Ridge	Lasso	Stepwise Fit	FSS	Ridge	Lasso	Stepwise Fit
Intercept	4127.1	4127.1	4127.1	4127.1	1008	1008	1008	1008
City Capacity	1885.7	367.6	665.4	1837.5	313.9	95	163.5	919.9
Outskirt Capacity	-863.5	118.4	0	-779.7	0	15.3	0	-478.9
Mean Temperature	647.0	449.8	608	629.8	455.5	188.7	317.6	315
Mean Visibility	195.6	161.3	151.7	146.2	0	51.9	31.8	0
Mean Wind Speed	0	-157.8	-107.9	-157.1	0	-44.4	0	-47.3
Precipitation	-394.8	-231.4	-287.4	-327.8	0	-48.2	-40.4	-90.9
Holiday	0	-129.5	-42.5	0	0	12.9	0	-46.2
Workday	487.6	211	353.0	581.6	-449.3	-154.1	-333	-413.5
2011	0	-337.7	-326	0	0	-67.4	0	192
2012	0	64.4	0	0	0	-17.3	0	0
2013	0	266.6	125	0	0	83.2	79.1	0
Sunday	0	178.3	-103.1	0	0	70.6	0	0
Monday	0	10.6	0	0	0	-29.6	0	0
Tuesday	0	52.5	0	0	0	-60.3	0	0
Wednesday	0	84.6	14.4	67.2	0	-47.1	0	0
Thursday	0	55.1	0	0	0	-55.9	0	0
Friday	0	21.3	0	0	0	-5.6	0	68
Saturday	0	-46.1	0	178.6	0	128.3	66.4	92.6
Winter	0	-217.3	-149.7	0	0	-123.9	-99.6	-73.1
Spring	0	5.5	0	169.9	152.5	51.8	16.6	122.3
Summer	0	56.9	0	0	0	70.9	0	0
Fall	265	146.2	123	299.7	0	-3.7	0	0
Cloud Cover 0	0	19.6	0	0	0	14.7	0	0
Cloud Cover 1	0	-20.8	0	0	0	-2.1	0	0
Cloud Cover 2	0	33.1	0	0	0	35.7	0	0
Cloud Cover 3	0	31.2	0	0	0	4.7	0	0
Cloud Cover 4	0	29.9	0	0	0	11.8	0	0
Cloud Cover 5	0	51.9	0	0	0	20	0	0
Cloud Cover 6	0	38.4	0	0	0	21.2	0	0
Cloud Cover 7	0	-26.8	-7.5	-92.7	0	-18.1	0	-61.2
Cloud Cover 8	0	-124.5	-101.8	-163.6	0	-63.2	-60.3	-119.9
MSE	425,440	465,160	418,980	395,420	299,520	279,649	282,720	251,480
MAE	505.98	526.03	497.32	483.14	387.82	333	334.58	339.23

Table 3: Values of β for FSS, ridge, lasso and `stepwise fit`

7 Regression Trees

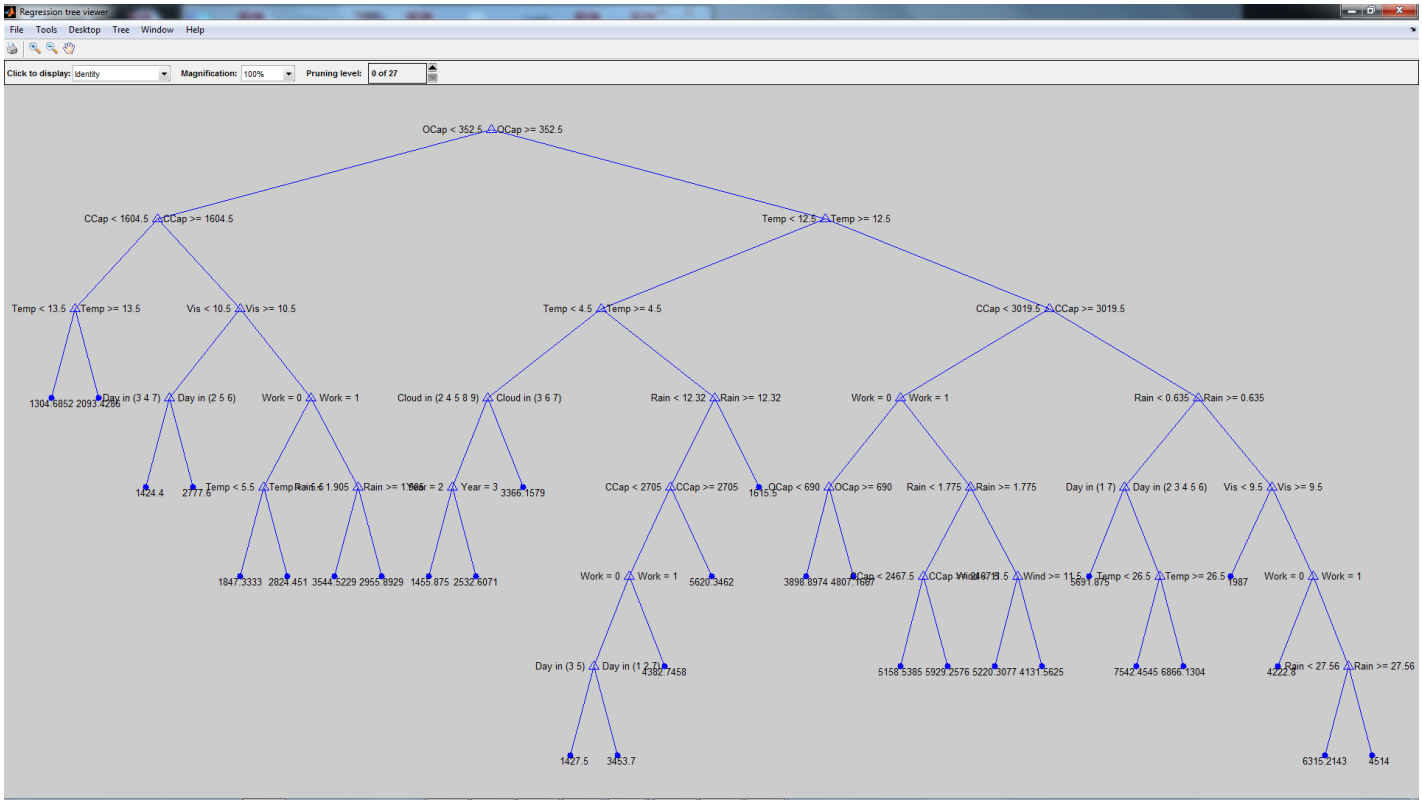
The last method applied to the data is a regression tree. Regression trees are especially effective with categorical data because splits in the tree are more naturally defined. Applying a tree to the bike sharing data is quite practical because it can be easily understood by someone with little background in statistics. Following the branches in the tree forms an easy checklist that can be used to determine the demand for a particular day. The weather forecast for the day, as well as the date and season can be used to follow the checklist to get a general idea about what can be expected on a particular day.

Since regression trees divide the input space into blocks and assign an averaged output value to each block, the method may have the same issues as k-nn. The large variation in the data from day to day may cause the averages to under or over estimate. After training and testing regression trees, results vary between registered and casual users. The tree performs worse than k-nn for registered members, however it does quite a bit better for casual users. The method is still outperformed by linear regression, but provides a useful way to identify important variables. To find optimal trees for the data, cross validation is used to find the best pruning levels for the trees. Figure 9 shows the cross validation curves for registered and casual users, plotting MSE vs. the number of terminal nodes. The resulting trees and their MSE are shown in Figure 10. In both trees the first several splits are based on the same key variables identified from linear regression. For registered users, city and outskirt capacity, temperature and workday all define major splits in the data. For casual users, workday is the first split made with larger trip counts located on the side corresponding to weekends and holidays. Hence the results from the regression trees further confirms the results from linear regression and k-nn.

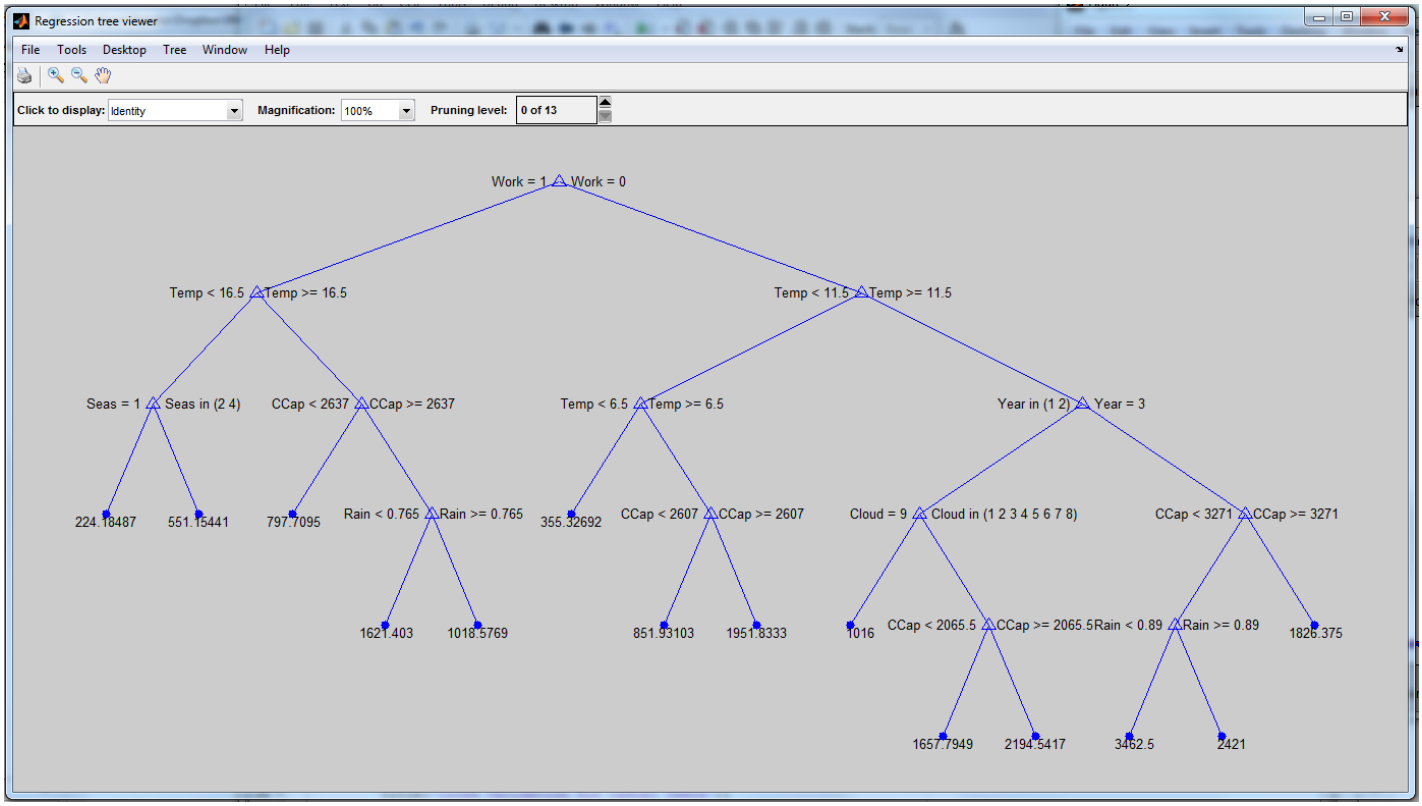


(a) (b)

Figure 9: Cross validation curves for regression trees for registered (a) and casual users (b). The optimal number of terminal nodes are represented by a red circle. For registered users, 29 terminal nodes is found to be optimal with a pruning level of 27. For casual users, the optimal number of terminal nodes is 14, with a pruning level of 13.



(a) MSE=865,170



(b) MSE=325,270

Figure 10: Optimal trees found for registered and casual users.

8 Discussion

Of the three methods applied, linear regression gave the best results since it was able to identify important predictors in the model. Since it finds a linear relationship between the predictors and output, it does not rely on taking averages of the data which causes larger errors for k-nn and regression trees. However all three gave prediction errors in similar ranges. The mean absolute error for registered users was typically between 500-600 trips and for casual users it was 300-400 trips. Since the maximum values of trips for registered and casual users is roughly 8500 and 5000, making a prediction that is 500 or 300 trips off is still quite reasonable. The fact that none of the methods could improve these ranges could indicate that this is the best that can be done with this type of data. There is a great deal of variation within each year and also across different years which contributes to the performance of the methods. For better results more data would have to be collected. Once demand for the system stabilizes the three methods applied will likely perform much better. However finding methods that are flexible enough to predict the growth of the system would also be very beneficial. To obtain more accurate results that capture the particular fluctuations in the data, predictions may have to be made at the station level. Including station specific information such as transit schedules, or proximity to businesses may capture more local bike trends that are not determined by the current variables in the model.

For k-nn, more investigation could be done about distance measures for mixed data. The metric itself could be changed or distance measures could be given larger weights for more important variables. With more data, the averages would most likely begin to give very good results. Another advantage of k-nn is that it is not affected by correlations between input variables, unlike linear regression.

All three methods also highlight the fact that bike use varies depending on the type of user. Registered users display more stable patterns, but their bike use is typically restricted to workdays. Casual users seem to be motivated by more recreational concerns, using the system primarily on weekends with good weather.

9 Conclusion

Overall, bike sharing data is well suited to regression since there are many variables that are linearly related to bike demand such as station capacity and temperature. For improved results with k-nn and regression trees, more years of data are probably necessary. There is a great deal of room for improvement with the model. More variables could be included, and predictions could be made at the station level including location information such as proximity to transit, to tourist attractions, to businesses etc. Predictions can also be broken down by hour, specifying rush hour information and also slow periods. Results could be used to create bike redistribution strategies since a large number of resources are used to redistribute bikes from full to empty stations. Other methods could be applied especially time series models, or Bayesian networks as in [1]. A comparison of demand in the city versus the suburbs would also be interesting since different factors may be at play in these two areas. This type of data is very amenable to machine learning techniques and as more and more bike systems are introduced in the US and Canada there will be growing opportunities to do interesting research with bike sharing networks.

References

[1] Froehlich, J., Neumann, J., and Oliver, N. (2009) Sensing and Predicting the Pulse of the City through Shared Bicycling, *21st International Joint Conference on Artificial Intelligence (IJCAI)*, Pasadena, California, pp. 1420-1426.

[2] Hastie, T., Tibshirani, R., and Friedman, J. (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction Second Edition* Springer, New York, NY

[3] LeMay, V., and Temesgen, H. (2005) Comparison of Nearest Neighbor Methods for Estimating Basal Area and Stems per Hectare Using Aerial Auxiliary Variables. *Forest Science* **51**(2): 109-119.

[4] Lourenco, F., Lobo, V., and Bacao, F. (2004). Binary-based similarity measures for categorical data and their application in self-organizing maps. *Proceedings of the JOCLAD*, Lisbon, Portugal.

[5] Yuan, M. and Lin, Y. (2007) Model selection and estimation in regression with grouped variables, *Journal of the Royal Statistical Society, Series B* **68**(1): 49-67.

[6] Capital Bikeshare Data: `http://capitalbikeshare.com/system-data`

[7] Historical Weather Data: `http://www.wunderground.com/history/`

[8] District of Columbia Holiday Schedules: `http://dchr.dc.gov/page/holiday-schedules`

[9] Capital Bikeshare Information: `capitalbikeshare.com/about`

10 Appendix: Description of Matlab Files

There are four main files from which each major method is called from:

- `RunAveTrend.m`
- `RunKnn.m`
- `RunLinearRegression.m`
- `RunRegTree.m`

`RunKnn.m` calls the functions `knn.m` and `form_dist_matrix.m`.

`RunLinearRegression.m` calls four scripts corresponding to each of the regression models:

- `FSS.m`
- `RIDGEReg.m`
- `LassoReg.m`
- `STEPMatlab.m`

Some of these scripts also call smaller functions as well.

`RunLinearRegression.m` can be executed all at once or in cell mode, with one cell being evaluated at a time for each regression method.

`RunRegTree.m` doesn't call any other functions. It produces cross validation results and trees from the single file.

All figures generated are placed in the folder `Output`