CS771A: Machine Learning: Tools, Techniques and Applications

Course Project

Bike Sharing Demand

Mentored by: Prof. Harish Karnick

Team Members

Rahul Yadav(11564)

Dhruv Singal(12243)

Srijan Saket(11728)

Peeyush Agarwal(12475)

Abstract

The Bike Sharing Demand challenge is a knowledge problem on Kaggle. The problem deals with analysis of the demand in the Capital Bikeshare program of Washington D.C. In particular, the problem asks the participants to combine historical usage patterns with weather data to forecast the bike rental demand. In this project we have proposed several methods to deal with the mentioned problem. We started with the analysis of the regressors that affect a bike sharing system. Since this is a regression problem the first model we tried is a multiple linear regression model. Then we moved to more complex non-parametric approaches like random forest which can handle data of mixed type and are robust to outliers unlike a simple regression model. To improve the results further we have used the Gradient Boosting Algorithm which is basically an ensemble of weak prediction models, like decision trees. Finally we have taken an ensemble of the results obtained from random forest and the gradient boosting to predict the count of bikes. These models can assist the planners of a bike sharing system as well as the users by predicting the bike demand at a macroscopic level incorporating certain factors.

Contents

A. Introduction

Bike sharing systems are a means of renting bicycles where the process of obtaining membership, rental, and bike return is automated via a network of kiosk locations throughout a city. Using these systems, people are able rent a bike from a one location and return it to a different place on an as-needed basis. Currently, there are over 500 bike-sharing programs around the world.

The data generated by these systems makes them attractive for researchers because the duration of travel, departure location, arrival location, and time elapsed is explicitly recorded. Bike sharing systems therefore function as a sensor network, which can be used for studying mobility in a city.

In this problem, we are provided hourly rental data spanning two years. For this competition, the training set is comprised of the first 19 days of each month, while the test set is the 20th to the end of the month. We are then asked to predict the total count of bikes rented during each hour covered by the test set, using only information available prior to the rental period.

This problem is of particular interest in various civic and logistical applications, as predicting the demand accurately can lead to better traffic management, appropriate logistical support for the bike sharing services and other areas as well.

B. Preprocessing:

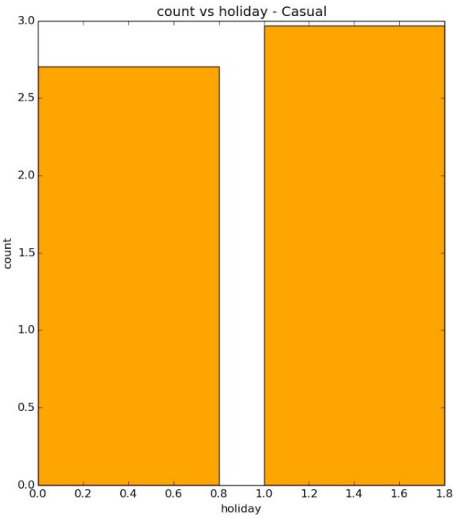
1. **Dataset**
   The dataset was provided by Hadi Fanaee Tork using data from Capital Bikeshare. It is hosted on UCI, and is accessible at
   http://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset

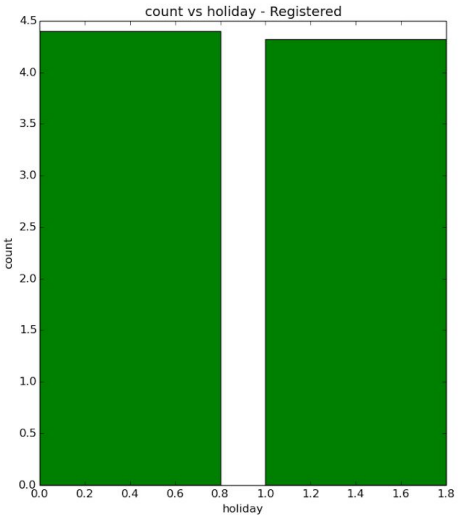
2. **Data Pre-processing and feature selection**

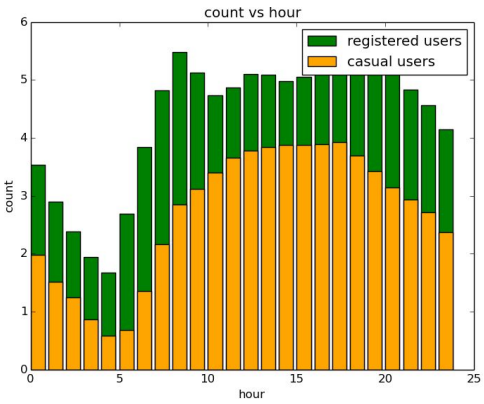
   The datetime attribute of the dataset was split into date (year, month, date, day of the week) and time (hour, minute, second). We made the assumption that only the hour of the day was the important attribute in the time set. We also converted the year attributed into an enumeration type attribute, to facilitate the training of the model.

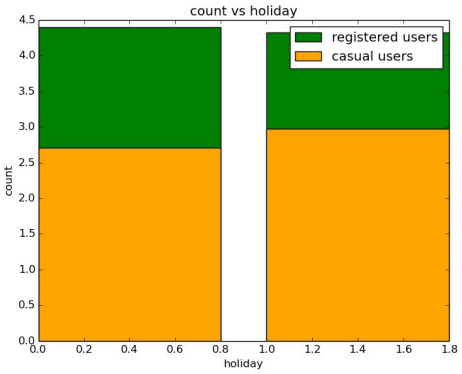
   We made the following observations in the dataset:

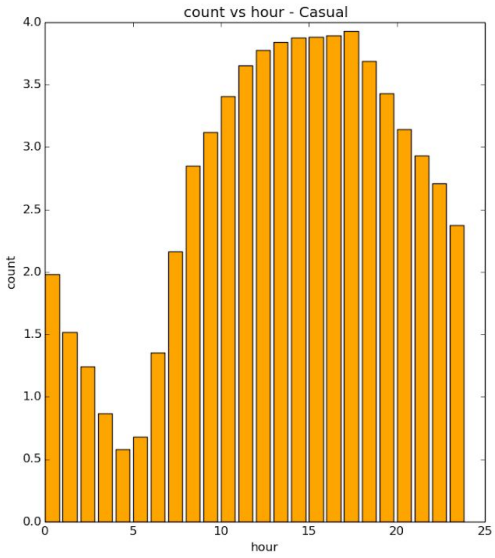
   A. The count of registered and casual users vary according to different distributions. This is evident from the following graphs:

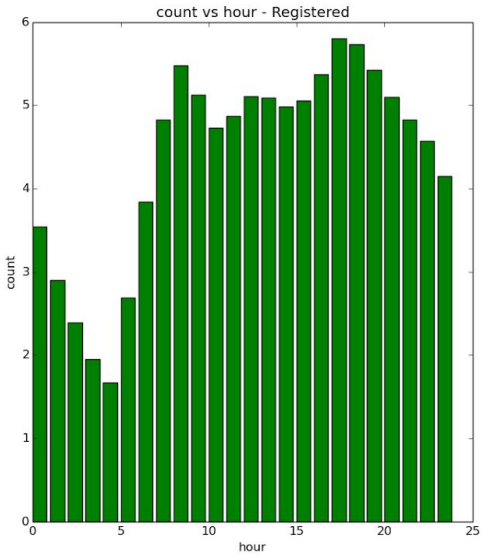






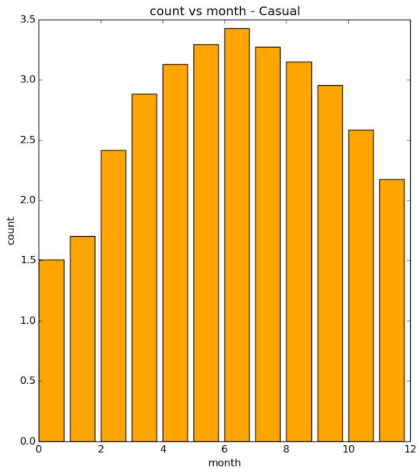


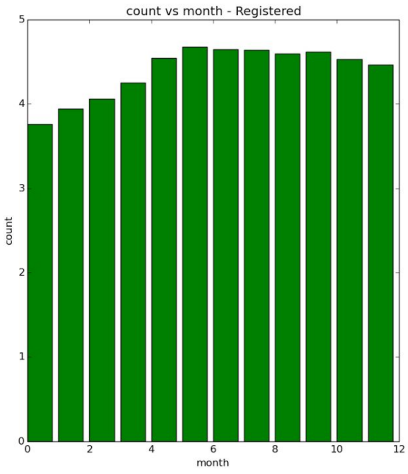


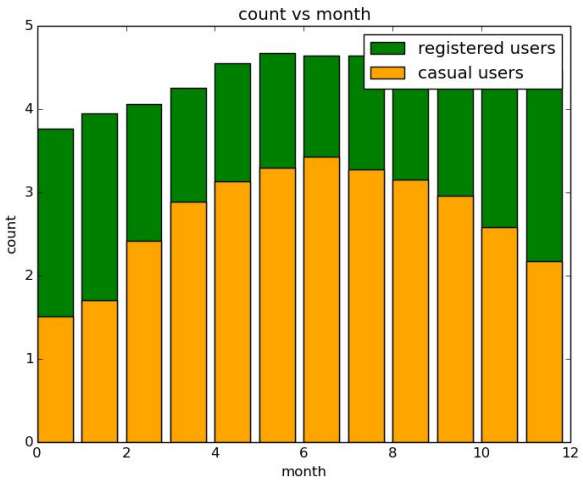


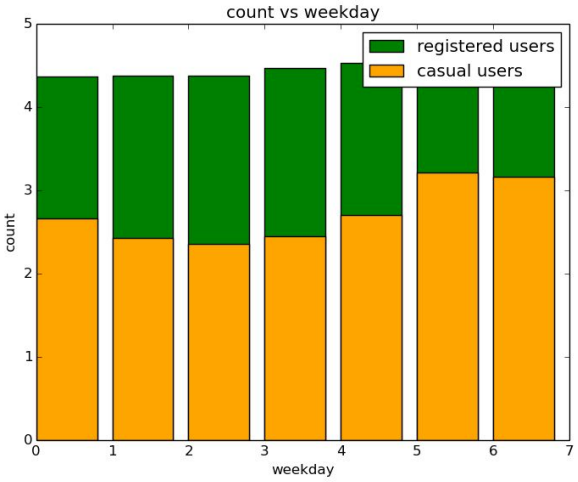
The hourly renting pattern clearly demonstrates that the demand from registered users tends to

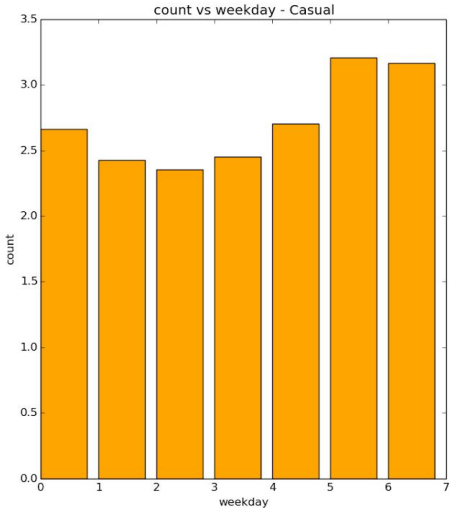
peak in morning and evening commute hours. With respect to holidays, it was observed that the demand for registered users actually declined during holidays, while the reverse was observed for casual users, which is pretty self-explanatory, considering that the registered users have a day off on holidays, and don’t want to commute to work using bikes.

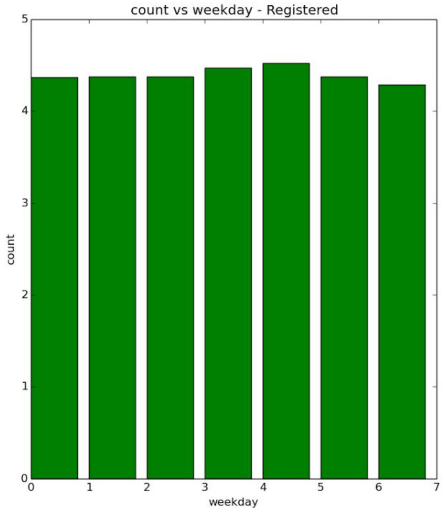






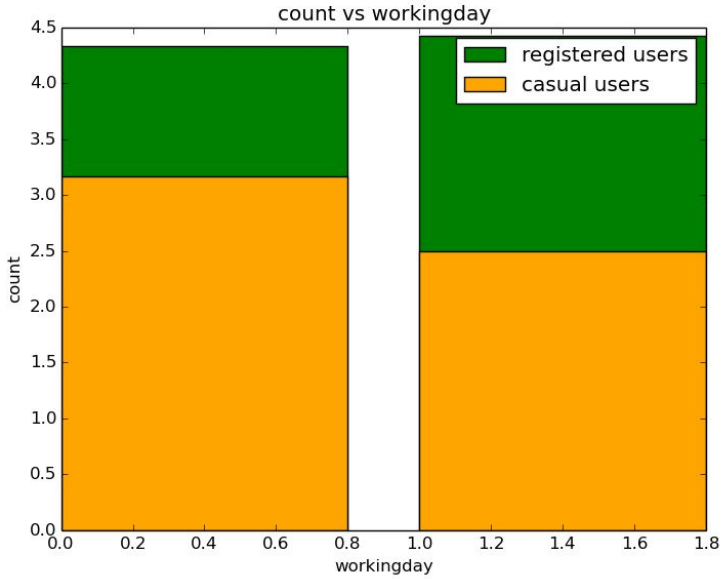


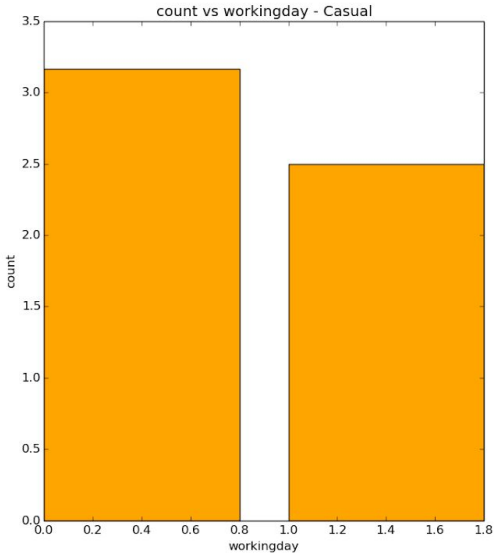


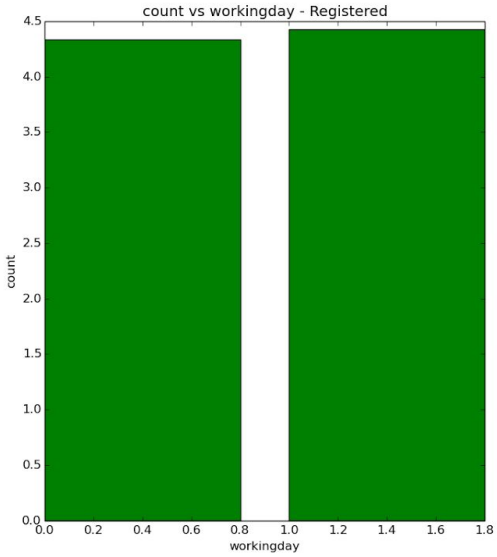


With respect to the months and weekdays, it's observed that casual users tend to rent more

during weekends and the demand from registered users actually fell down marginally during weekends. Casual users tend to rent more during spring and summer months, while the demand from registered users falls off during Christmas vacation period.







As is pretty evident from the all the graphs and observations, the renting behaviour of registered and casual users is inherently different.

Hence, the defining feature of our models was to train different models for registered and casual users and combine those results to give accurate predictions.

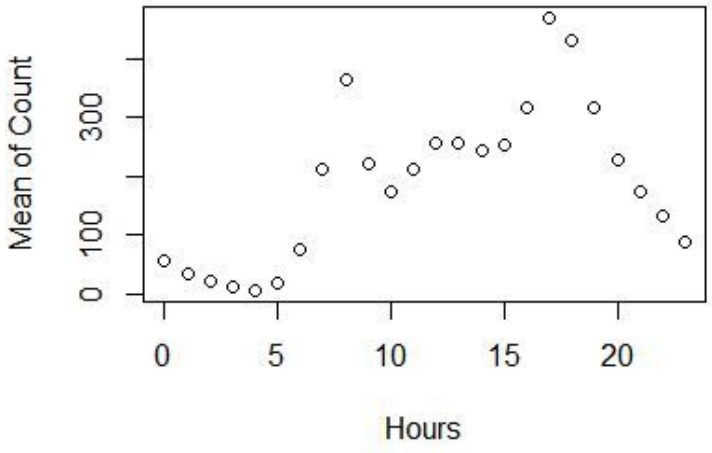
3. **Feature Engineering**

We have observed the train data set and based on the target variable (count,registered, casual) we have tried to obtain the few new variables which could help in our prediction task.

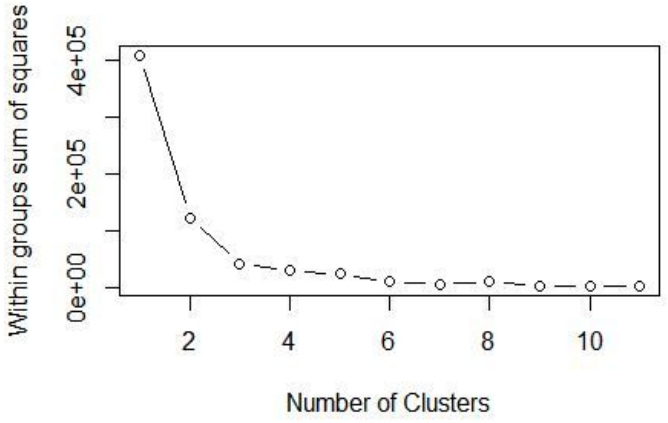
A). Inclusion of hourCluster variable:

We have observed the mean of “count” distributed with hours in such a way.

It reflects the hourly distribution of bike users so we have used this behaviour to create a variable using the K-mean clustering.



We have used the optimal cluster division using the cluster within sum of square and then finally created the daypart variable



B). Inclusion of isSunday Variable

We have also notices the sunday as least number of “count” in comparison on other days.

Day	“Count”

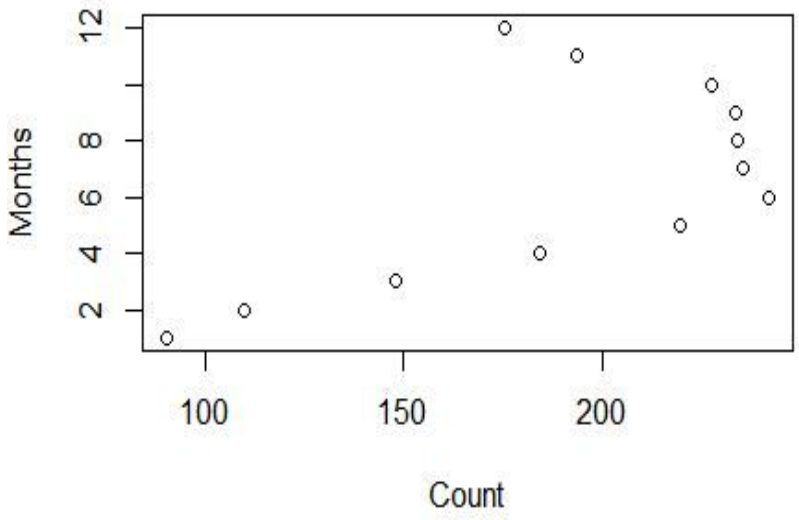
Friday	197.8443
Monday	190.3907
Saturday	196.6654
Sunday	**180.8398**
Thursday	197.2962
Tuesday	189.7238
Wednesday	188.4113

So we have used this information to group all days except in one group and created a dummy variables “Sunday” which takes “0” for sunday and “1” for other days.

C). Inclusion of monthCluster variable:

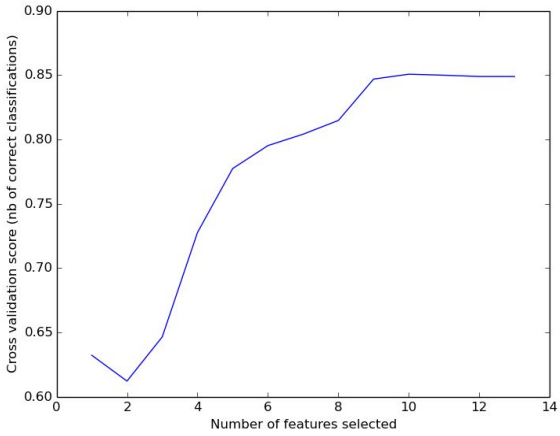
Similar to the above we graphically observed the distribution of “count“ on the month basis.

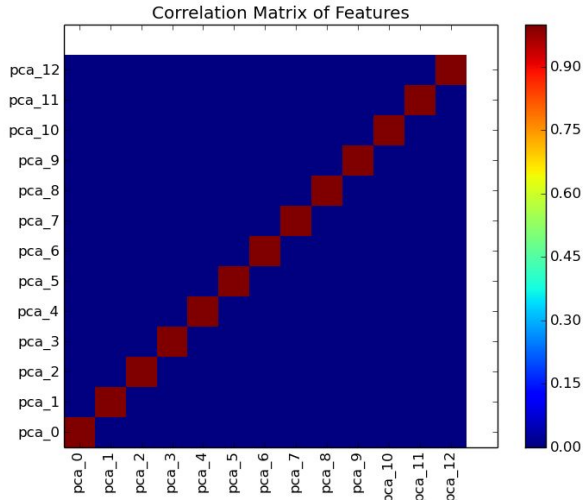
As evident from here we can clearly observe the grouping of months which might be helpful for us so we include this information as variable.

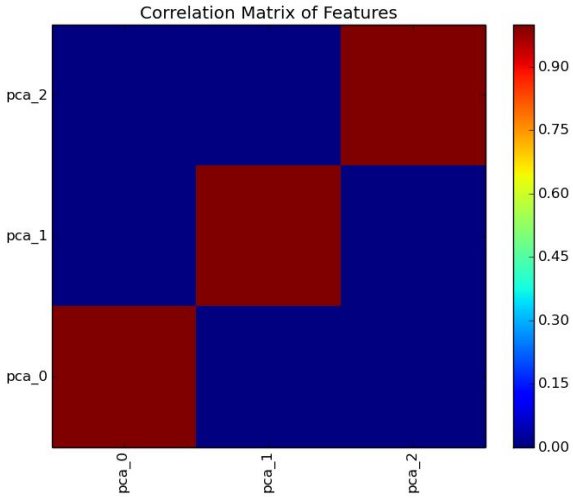


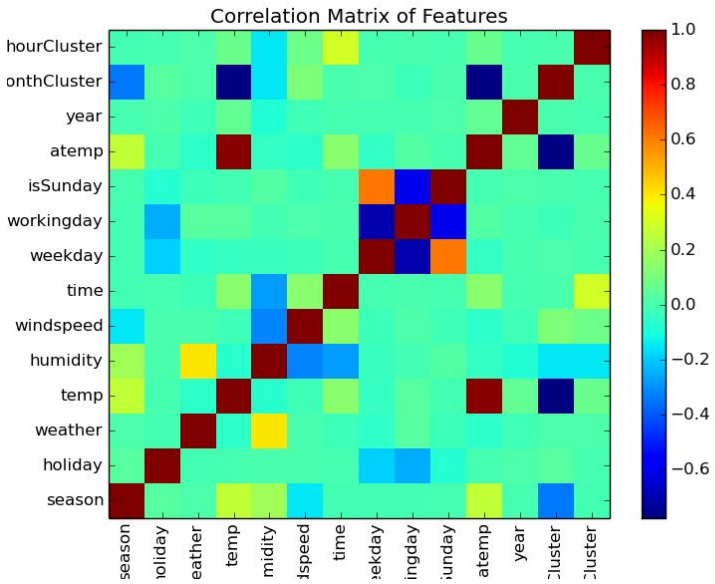
Here we have found the optimal cluster division and then used this information to create our new variable based on month clustering.

D) Recursive feature elimination using cross validation (RFECV) and feature selection:


Cross validation score (nb of correct classifications)
Number of features selected
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0
2
4
6
8
10
12
14


Correlation Matrix of Features
pca_12
pca_11
pca_10
pca_9
pca_8
pca_7
pca_6
pca_5
pca_4
pca_3
pca_2
pca_1
pca_0
0.90
0.75
0.60
0.45
0.30
0.15
0.00


Correlation Matrix of Features
pca_2
pca_1
pca_0
pca_0
pca_1
pca_2
0.90
0.75
0.60
0.45
0.30
0.15
0.00


Correlation Matrix of Features
hourCluster
onthCluster
year
atemp
isSunday
workingday
weekday
time
windspeed
humidity
temp
weather
holiday
season
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

The correlation confusion matrix between the 14 features (including hourCluster and monthCluster, isSunday) clearly demonstrates a set three subsets of features with some amount of correlation between them:

i) weekday, workingday, isSunday - moderate to high correlation
ii) windspeed, time, humidity - mild correlation
iii) atemp, temp - high correlation

On applying RFECV, puts the optimum number of variables required for modelling this problem at 10-11. Recursive feature elimination works by successively eliminating the worst performing variables (we used the Random Forest model to check the performance) and cross-validates it to check the performance of the model with the selected features.

Applying Principal Component Analysis (PCA) on the featureset, we got 12 uncorrelated variables, 3 of which account for 90% of the variance in the data. However, training the models using this feature set didn't prove to be good enough, which may be due to the important correlations incorporated in the original feature set.

Models And Results

1. Decision Tree:

Without Feature Engineering: 1.32522 - RMSLE
With Feature Engineering

A. Inclusion of sunday variable : 0.88789
B. Inclusion of sunday and daypart variables : 0.71930
C. Inclusion of sunday, daypart and month_cl variable : 0.45863

2. Multiple Linear Regression

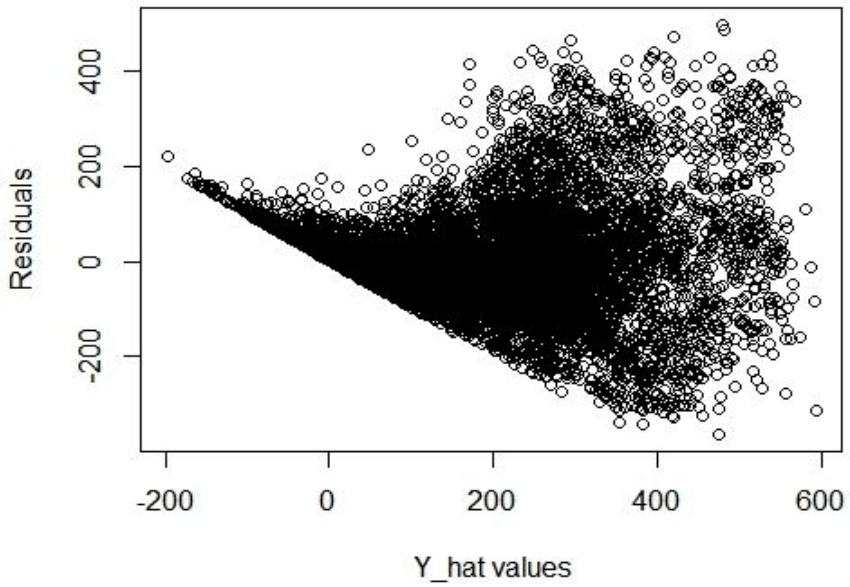
In order to apply this technique we started with the stepwise subset selection which starts with an intercept model, then adds a regressor at each step and simultaneously checks the significance of the previously added regressors. We got a subset of variables on applying this technique which we used for model fitting and prediction. Now there are certain underlying assumptions that needs to be fulfilled in order to apply this method. The conditions and their implementations are discussed below.

Multicollinearity check

We want our regressors not to depend on each other heavily. So we checked the multicollinearity of regressors on the basis of VIFs (Variance Inflation Factors) for the subset model and found that none of the regressors were highly correlated.

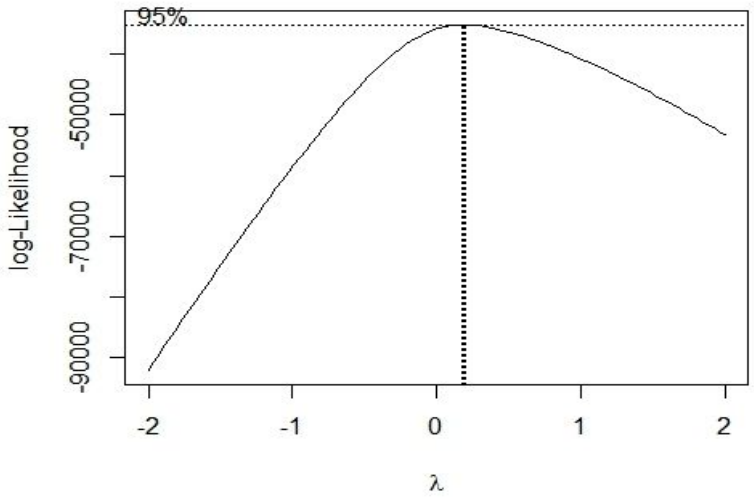
Homoscedasticity check

In a multiple linear regression model is assumed that the variances of error terms are constant. When we plotted the estimated residuals against the predicted response a pattern in the scatter plot was observed (as can be seen in the pic below) which suggests that the error variances are not constant.

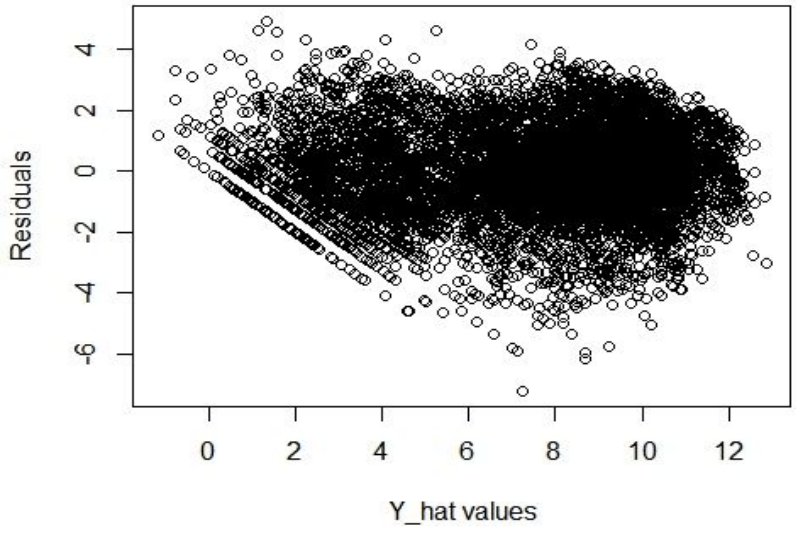


In order to fix this some transformation was needed such that the variances do not depend on any regressor or the dependent variable. For this we used the Box-Cox transformation which uses a transformation on the prediction variable.

For different values of the transforming parameter following graph was obtained:



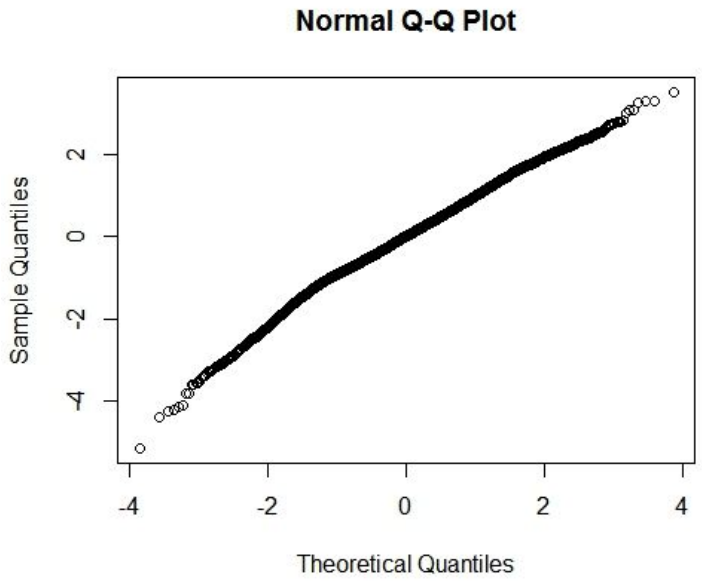
After applying the transformation when we again plotted the residuals against the predicted response following scatter plot was obtained:



Now this plot is random (has no pattern) and is centered about 0 (mean) which suggests that now the errors are homoscedastic.

Normality check

One of the main assumptions of a regression model is that the errors follow a normal distribution. We checked this assumption through a quantile-quantile plot:



Since there is minimal deviation we can convince ourselves that the errors follow a normal distribution.

After all the assumptions were met we again fitted the model and in this case following results were obtained:
RMSLE = **0.64**

3. Random Forest

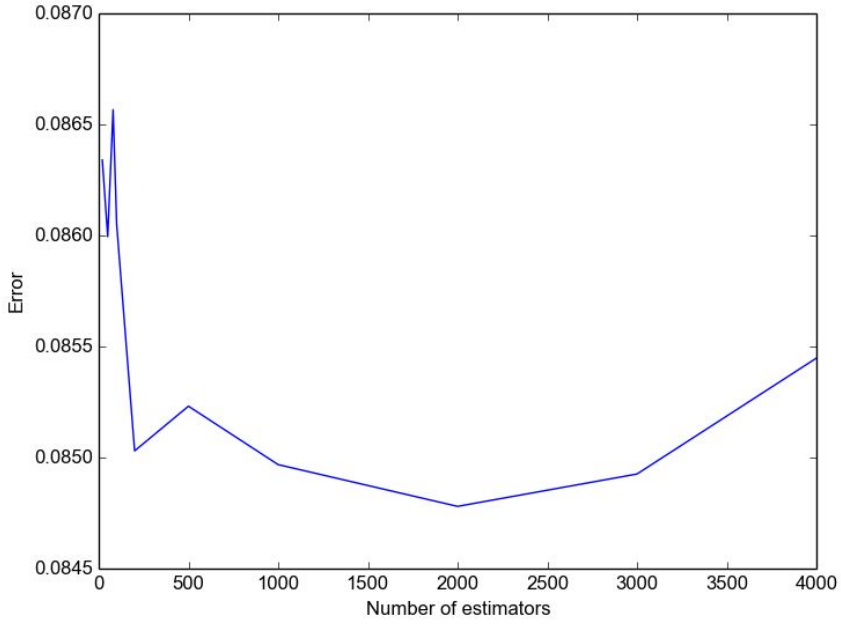
Predicting Casual and Registered users separately:
We observed during initial analysis of data that registered and casual users had different cycling behaviour. The behaviour for casual users was much more dependent on windspeed, temperature, season etc. Also, while the bike renting increased during morning and evening for registered users, it increased around evening for casual users. We thus decided to create separate models for the two types of users and combine their results to get the total count.

Error Measure:
We decided to optimize the Root Mean Squared Logarithmic Error (RMSLE). The intuition behind this is that the predictions are more useful if they are within a certain range of the original value. For example, if the predicted value is x, 1.2*x bikes may be kept for renting. So, a prediction of 140 when the actual renting was 120 or 160 is better than a prediction of 20 when the actual renting was 0 or 40. Also, Kaggle has chosen the same measure for evaluation on their leaderboard.
Considering this in mind, we decided to predict the log value of count.

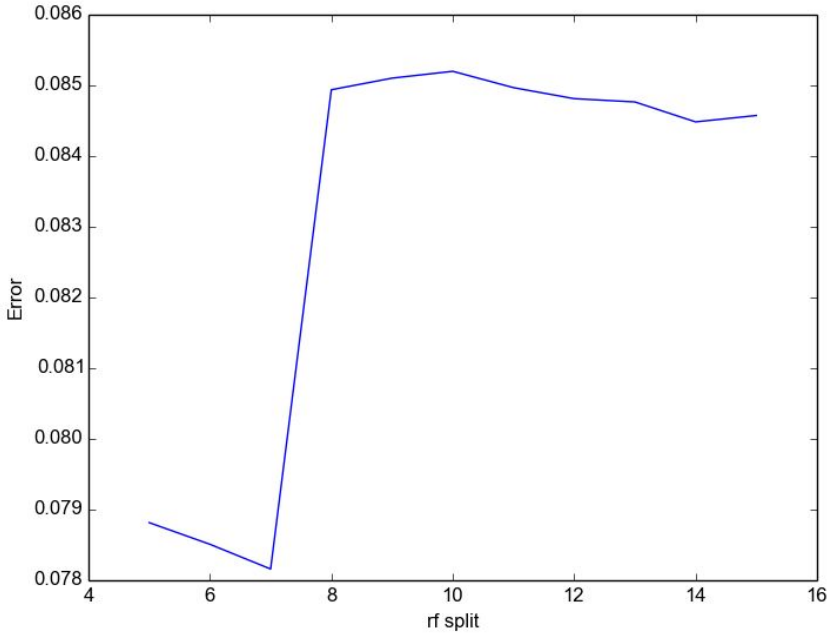
Cross Validation:
During the initial analysis of the dataset we found that there is a time dependence in the dataset. So, we decided that random test-train split will not represent the actual scenario very well. We decided to use the first 19 days of every month as train data and 20th day as test data. This helped us better evaluate our models thus helping in finding better parameters.

Parameter Tuning:
We tuned the number of trees in the forest and the minimum number of nodes required for splitting.

Using cross validation graph, we decided than 2000 was a good value for number of estimators/trees in the forest.



From cross validation, we decided than 7 was a good value for the minimum number of samples required to split an internal node.



Interesting observations with features:
We analyzed the importance of various variables while feature engineering which led to some interesting observations and insights.

We found that adding the variable 'isSunday' which represents if the day is sunday is useful if the number of trees is less (around 100). However, it becomes useless and contributes negatively to the model if the number of trees is increased to let's say 1000. With smaller number of trees, adding such features is more useful. This is because with large number of trees in the forest, some of the forests choose 'isSunday' as a splitting criterion (weekday = 0) even without an explicit feature for it. These trees contribute to the final prediction whenever the variable is important and better model the data.

Categorical Variables:
Some of the features in the dataset are nominal. For example 'weather', 'holiday', 'workingday', 'season' and 'hour' represent categories and so we decided to convert them to categorical before training our model.

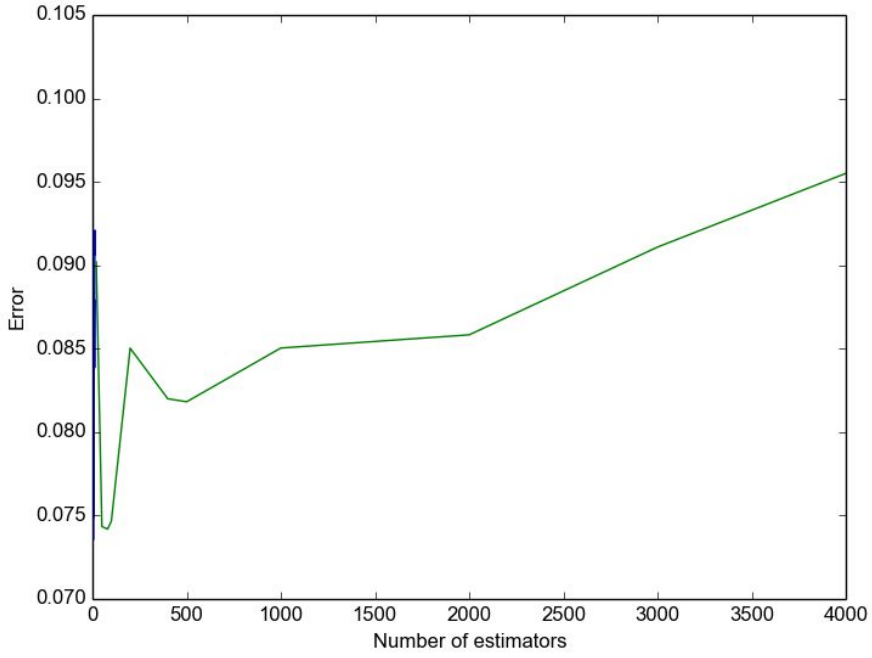
Data Standardization:
We standardized the features like 'windspeed', 'temp', 'humidity', 'atemp' etc. However, this did not impact our model. This was expected because variables are not compared with each other while applying random forest and thus standardization does not impact the model.

Principal Component Analysis:
We also tried applying PCA on the dataset before training our model. However, this led to poorer results. This was because good proportion of the signal was present in features with low variance. Also, the original features probably led to better decision trees which represented the actual scenario more closely.

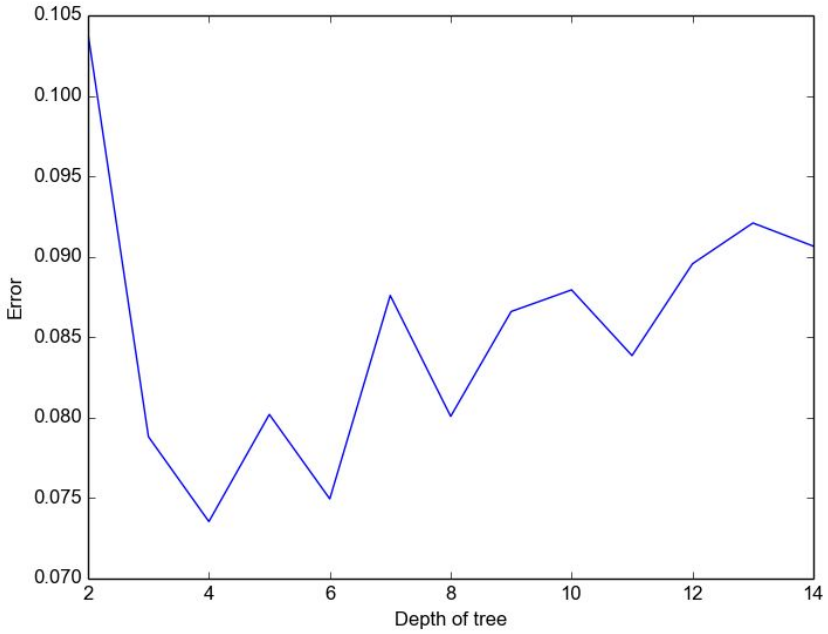
Our best model gives a cross validation RMSLE error of 0.08478.

4. Gradient Boosting
The analysis done in Random Forest was repeated for Gradient Boosting Regressors and similar results were obtained.

Using cross validation graph, we decided than 120 was a good value for number of estimators/trees. The model started to overfit after that and showed significant overfitting with more than 1000 estimators.



Using cross validation graph, we decided than 6 was a good value for the depth of tree. The model started to overfit after that and showed significant overfitting when the depth was more than 10.



Our best model gives a cross validation RMSLE error of 0.0749

5. Ensemble

Finally, we create an ensemble of Random Forest and Gradient Boosting Regressor. The combined model performs better than the individual models and gave better predictions/accuracies.

Conclusion

We managed to score a rank of **28** on the Kaggle challenge, and our best submission gave an RMSLE error of 0.36996 on Kaggle and a cross validation RMSLE error of 0.0747 on local data. This demonstrates the capability of tree regressors in real world problems and the power of ensembling in getting better models without overfitting. Feature engineering also played a key role in getting such results. The error confidence seems appropriate for the real life application to use the predictions.

References

- http://scikit-learn.org/stable/modules/ensemble.html
- Wikipedia, for articles on Gradient boosting, Random forests and error measures
- http://matplotlib.org/users/pyplot_tutorial.html
- http://scikit-learn.org/stable/modules/classes.html#module-sklearn.linear_model