Bike Sharing Demand

Hossein Zareamoghaddam[1]

[1]Data is available at *https://www.kaggle.com/c/bike-sharing-demand*

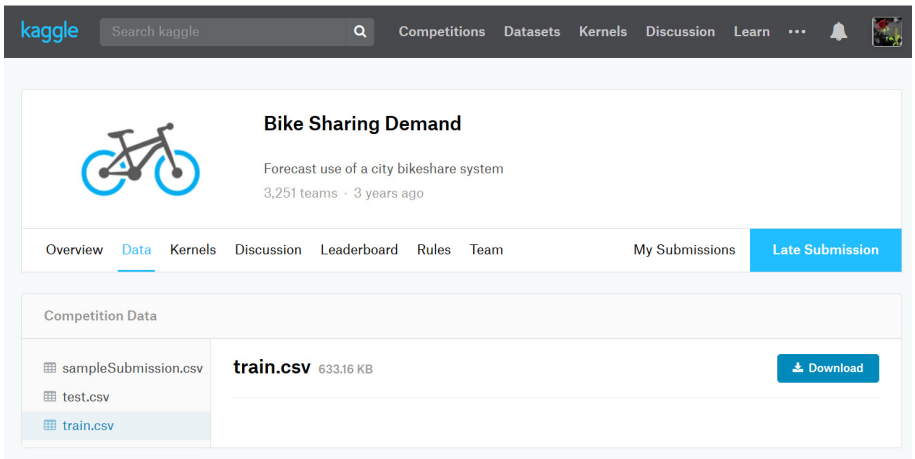
December 4, 2016

1 Introduction

In bike sharing competition, the participants were asked to combine historical usage patterns with weather data available in *train* dataset to forecast bike rental demand in the Capital Bikeshare program in Washington, D.C using information given in *test* set. The hourly rental data spanning were provided for two years while the training set was comprised of the first 19 days of each month, and the test set was the 20th to the end of the month. The data became available at Kaggle website by Hadi Fanaee Tork (in Fanaee-T, Hadi, and Gama, Joao, Event labeling combining ensemble detectors and background knowledge, Progress in Artificial Intelligence (2013): pp. 1-15, Springer Berlin Heidelberg) using data from Capital Bikeshare.

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.

The following information are provided for the training set with 10886 observations while the observed values for *casual*, *registered* and *count* are expected to be predicted for the test set with 6493 observations.



- *datetime:* hourly date + timestamp
- *season:* 1 = spring, 2 = summer, 3 = fall, 4 = winter
- *holiday:* whether the day is considered a holiday
- *workingday:* whether the day is neither a weekend nor holiday
- *weather:* 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
- *temp:* temperature in Celsius
- *atemp:* "feels like" temperature in Celsius
- *humidity:* relative humidity
- *windspeed:*wind speed
- *casual:* number of non-registered user rentals initiated
- *registered:* number of registered user rentals initiated
- *count:* number of total rentals

Using R-code 'str(train)', the format of the data and its features are shown.

```
> setwd("E:/GitHub/Bike Sharing Demand")
> train<-read.csv("train.csv")
> test<-read.csv("test.csv")
> str(train)
'data.frame': 10886 obs. of  12 variables:
$ datetime  : Factor w/ 10886 levels "2011-01-01 00:00:00",..: 1 2 3 4 5 6 ...
$ season    : int  1 1 1 1 1 1 1 1 1 1 ...
$ holiday   : int  0 0 0 0 0 0 0 0 0 0 ...
$ workingday: int  0 0 0 0 0 0 0 0 0 0 ...
$ weather   : int  1 1 1 1 1 2 1 1 1 1 ...
$ temp      : num  9.84 9.02 9.02 9.84 9.84 ...
$ atemp     : num  14.4 13.6 13.6 14.4 14.4 ...
$ humidity  : int  81 80 80 75 75 75 80 86 75 76 ...
$ windspeed : num  0 0 0 0 0 ...
$ casual    : int  3 8 5 3 0 0 2 1 1 8 ...
$ registered: int  13 32 27 10 1 1 0 2 7 6 ...
$ count     : int  16 40 32 13 1 1 2 3 8 14 ...
```

2 Data Processing and Visualization

Here, we look more carefully to the datasets. Fortunately, there is no missing values in the data sets. Both *train* and *test* data sets are manipulated for later analysis. To that end, for example, features like 'year', 'month', 'day' and 'hour' can be extracted from each set where 'datetime' shows date + hourly time stamp at the same column, formatted as "2011-01-01 00:00:00" for the first time point, i.e. midnight (00:00 AM) on January 01, 2011. Those features are separated using R-code below.

```
> ### Features tweaking
> # season, holiday, workingday and weather are factorized
> names <- c("season", "holiday", "workingday", "weather")
> train[,names] <- lapply(train[,names], factor)
> test[,names] <- lapply(test[,names], factor)
>
> #time
> train$hour<-(as.integer(substring(train$datetime,12,13)))
> test$hour<-(as.integer(substring(test$datetime,12,13)))
>
> train$time=NA
> test$time=NA
> train$time <- ifelse(train$hour >= 5 & train$hour < 12, 1,
+                      ifelse(train$hour >= 12 & train$hour < 17, 2,
+                             ifelse(train$hour >= 17 & train$hour < 22,3, 4 )))
> train$time<- as.factor(train$time)
> levels(train$time)<-c("Morning","Afternoon","Evening","Night")
>
> test$time <- ifelse(test$hour >= 5 & test$hour < 12, 1,
+                      ifelse(test$hour >= 12 & test$hour < 17, 2,
+                             ifelse(test$hour >= 17 & test$hour < 22, 3, 4 )))
> test$time<- as.factor(test$time)
> levels(test$time)<-c("Morning","Afternoon","Evening","Night")
>
> #days
> train$day<-factor(weekdays(as.Date(train$datetime)))
> test$day<-factor(weekdays(as.Date(test$datetime)))
>
> #months
> train$month<-months(as.Date(train$datetime))
> test$month<-months(as.Date(test$datetime))
>
> #year
> train$year<-factor(as.integer(substring(train$datetime,1,4)))
> test$year<-factor(as.integer(substring(test$datetime,1,4)))
```

The patterns of count, casual and registered of the train data set, the features which are going to be predicted for the test set, are presented in Figure 1 using the following

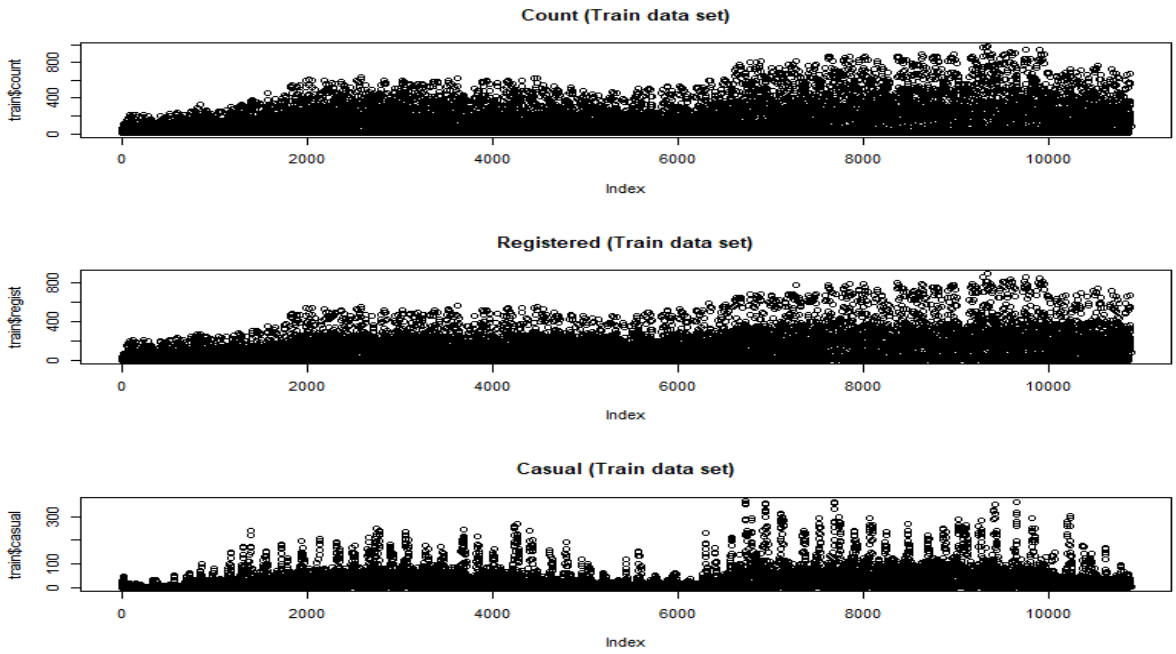


Figure 1: Predicted values of count, casual and registered in train set

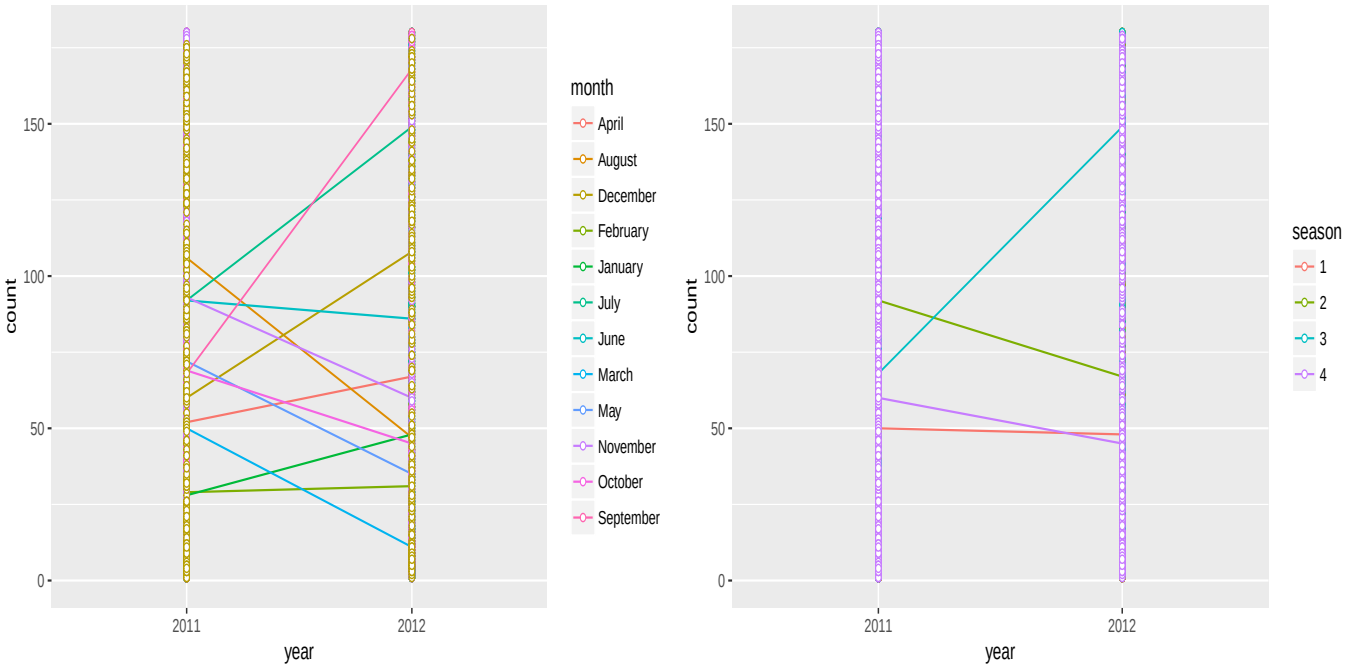
R-code.

```
> par(mfrow=c(3,1))
> plot(train$count,main = "count")
> plot(train$registered,main = "registered")
> plot(train$casual,main = "casual")
>
```

By making use of 'ggplot2' R-package, the bike rentals in two years of 2011 and 2012 are graphically represented by involving the affects of some other factors including month, season, day and holiday separately in Figure 2. One may notice, for example the daily or monthly, rental demand changes from 2011 to 2012. In panel (c) of Figure 2, the demand in 2012 was more than 2011 regardless of the demand time being a holiday or not. One of these plots is created using the following R-code. One may produce the remaining plots in Figure 2 using similar codes.

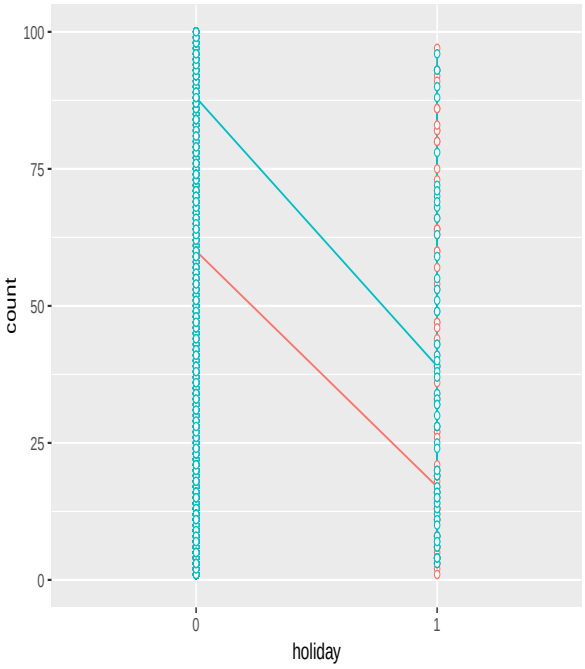
```
> library('ggplot2')
> ggplot(train, aes(x=year, y=count, colour=day, group=day)) +
+   geom_line() + geom_point(shape=21, fill="white") +
+   ylim(ymin=0,ymax=125)
```

The hourly trends of bike rentals of different years, days, seasons, holiday and weather conditions are comparable in Figure 3 panels (a)-(e). One may observe that the highly in demand hourly time periods were about 7:00 AM and 5:00 PM. These plots also confirm that season, day, year, holiday and weather have significant influences on hourly changes

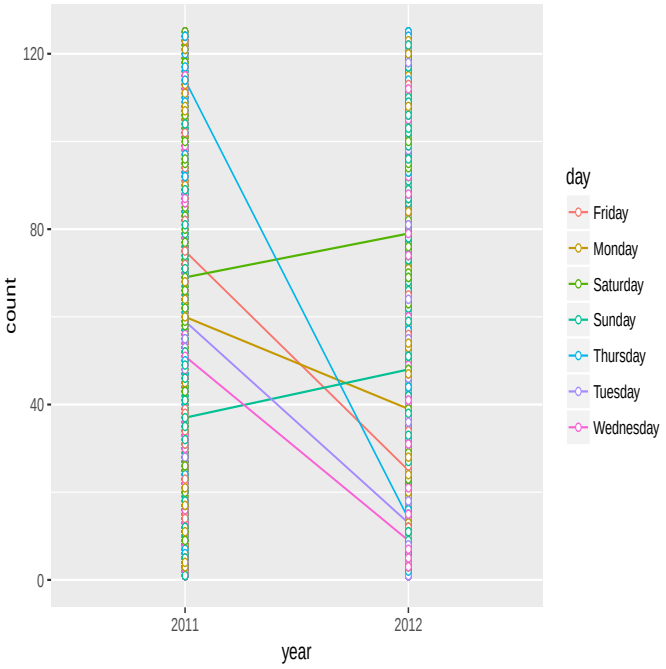


(a) count vs (year and month)

(b) count vs (year and season)



(c) count vs (year and holiday)



(d) count vs (year and day)

Figure 2: Changes of count by year and some other factors

of the number of rented bikes. Panel (f) represents the count changes versus temperature in being a holiday or not. That plot says that people did not like to biking in cold weather. One of the plots in Figure 3 is produced by the following R-code.

```
> # http://www.cookbook-r.com/Manipulating_data/Summarizing_data/
> summarySE <- function(data=NULL, measurevar, groupvars=NULL, na.rm=FALSE,
+                       conf.interval=.95, .drop=TRUE) {
+   library(plyr)
+   length2 <- function (x, na.rm=FALSE) {
+     if (na.rm) sum(!is.na(x))
+     else       length(x)
+   }
+
+   # This does the summary. For each group's data frame, return a vector with
+   # N, mean, and sd
+   datac <- ddply(data, groupvars, .drop=.drop,
+                  .fun = function(xx, col) {
+                    c(N    = length2(xx[[col]], na.rm=na.rm),
+                      mean = mean   (xx[[col]], na.rm=na.rm),
+                      sd   = sd     (xx[[col]], na.rm=na.rm)
+                    )
+                  },
+                  measurevar
+   )
+
+   # Rename the "mean" column
+   datac <- rename(datac, c("mean" = measurevar))
+
+   datac$se <- datac$sd / sqrt(datac$N)  # Calculate standard error of the mean
+
+   # Confidence interval multiplier for standard error
+   # Calculate t-statistic for confidence interval:
+   # e.g., if conf.interval is .95, use .975 (above/below), and use df=N-1
+   ciMult <- qt(conf.interval/2 + .5, datac$N-1)
+   datac$ci <- datac$se * ciMult
+
+   return(datac)
+ }
>
> # http://www.cookbook-r.com/Graphs/Plotting_means_and_error_bars_(ggplot2)/
> Myplot <- summarySE(train, measurevar="count", groupvars=c("year","hour"))
> pd <- position_dodge(0.1) # move them .05 to the left and right
> ggplot(Myplot, aes(x=hour, y=count, colour=year, group=year)) +
+   geom_errorbar(aes(ymin=count-ci, ymax=count+ci), colour="black", width=.1, positi
+   geom_line(position=pd) +
+   geom_point(position=pd, size=3)
>
```

The daily and monthly histogram plots of count in 2011 and 2012 are shown in Figure 4, where in both cases, the demand was higher in 2012 than 2011.

In order to predict the values of casual and registered and then add them together to get the values of count in test set, one may think of the possible relationship between casual and registered. The scatter plot in Figure 5 shows that these two features does not follow a particular pattern, but the scatter points look like a V-shape.

In Figure 6, some scatter plots of count, casual and registered versus different features are plotted to confirm the significant influence of selected factors on bike demands. The R-code to produce one of those plots is showing up here.

```
> install.packages("devtools")
> library(devtools)
> install_github("easyGgplot2", "kassambara")
> library("easyGgplot2")
> ggplot2.scatterplot(data=train, xName='casual', yName="registered",
+                     faceting=TRUE, facetingVarNames=c("time", "season"),
+                     facetingScales="free")
>
```

In order to have an initial idea about the relationship between selected factors , the corresponding pairwise scatter plots of the train data are shown in Figure 7 generated by

```
> pairNames<-c("temp","humidity","windspeed","hour","casual","registered","count")
>  pairs(train[,pairNames])
>
```

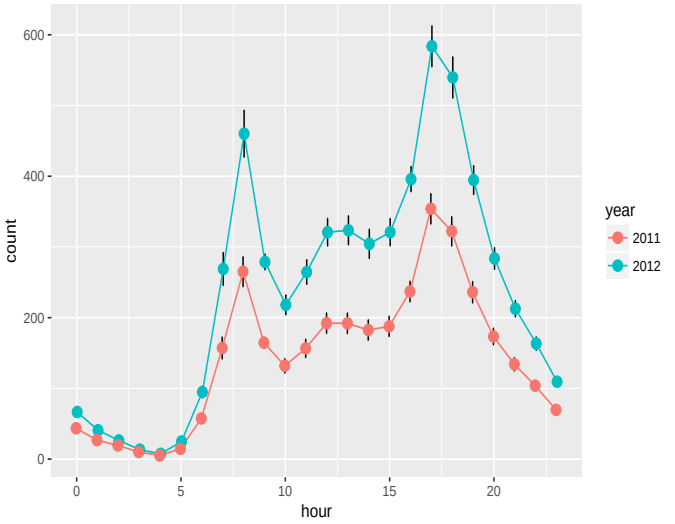
3 Predictive Analysis

As the values of feature 'count' is considered as the sum of the corresponding values in 'registered' and 'casual', one may find a model to predict the values for 'count' directly or find the predicted values of 'registered' and 'casual' for each row separately and then add them up as the associated 'count' value. Here, the second approach is considered.

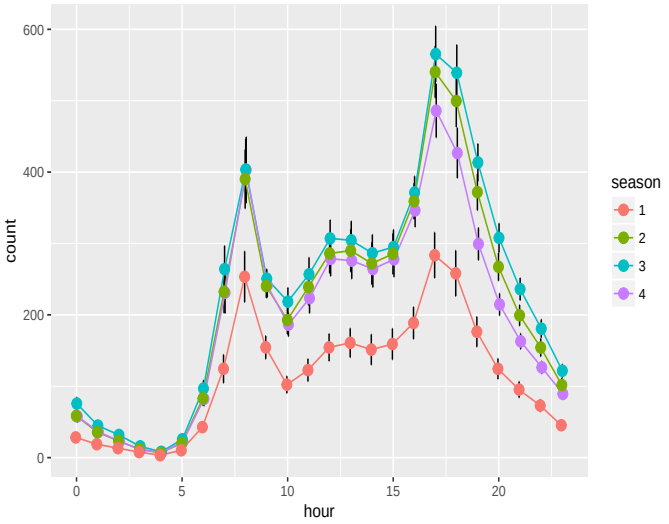
3.1 Linear Regression Model

The most widely use model-fitting technique perhaps is the linear regression model where the response (predicted) variable y is predicted by k different explanatory (predictors) variables $\{x_1, x_2, \ldots, x_k\}$ using the following regression model:

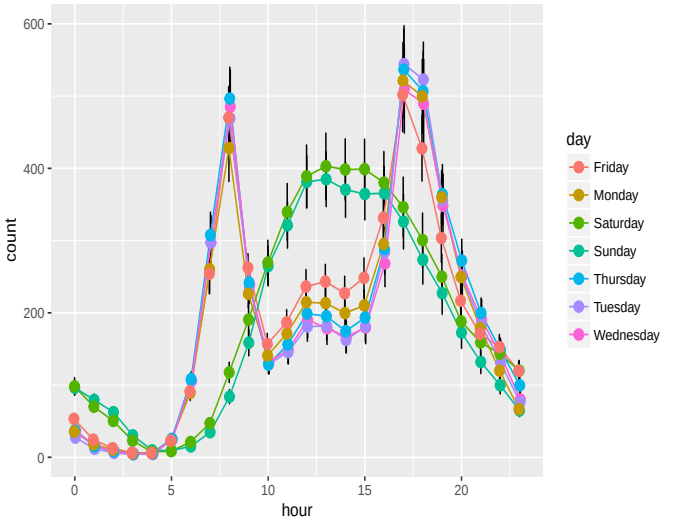
$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k + \epsilon,$$



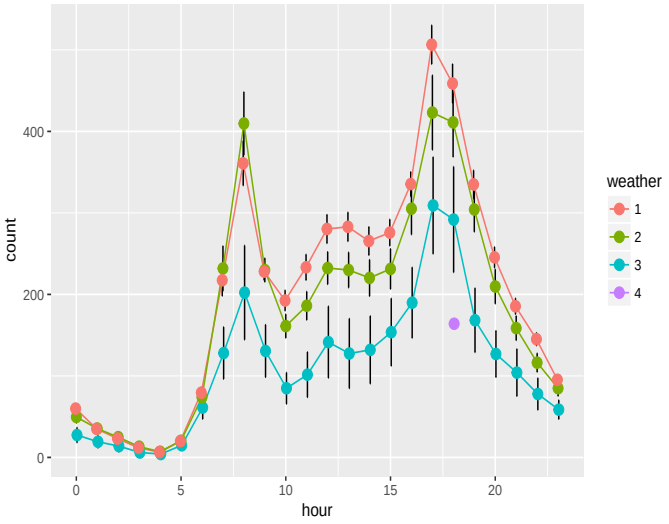
(a) count vs (year and hour)



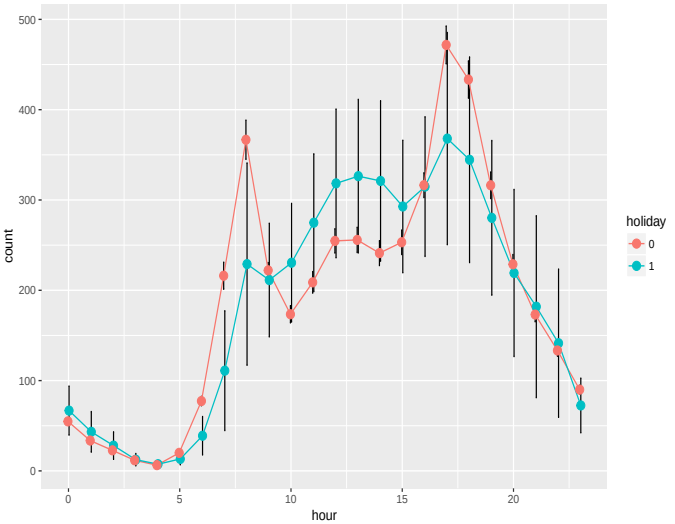
(b) count vs (season and hour)



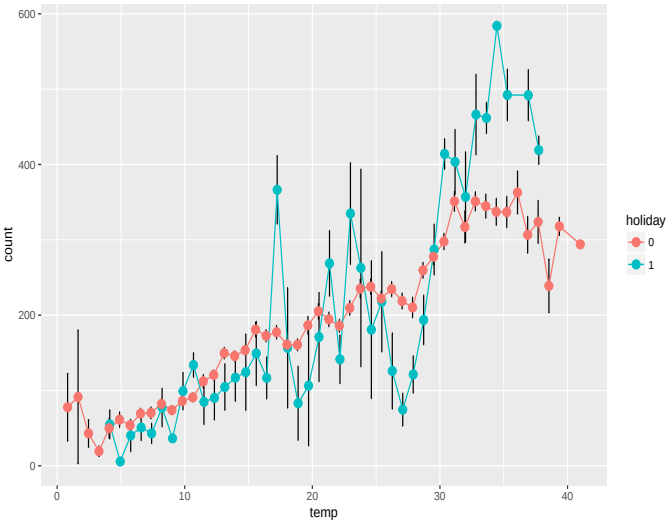
(c) count vs (day and hour)



(d) count vs (weather and hour)

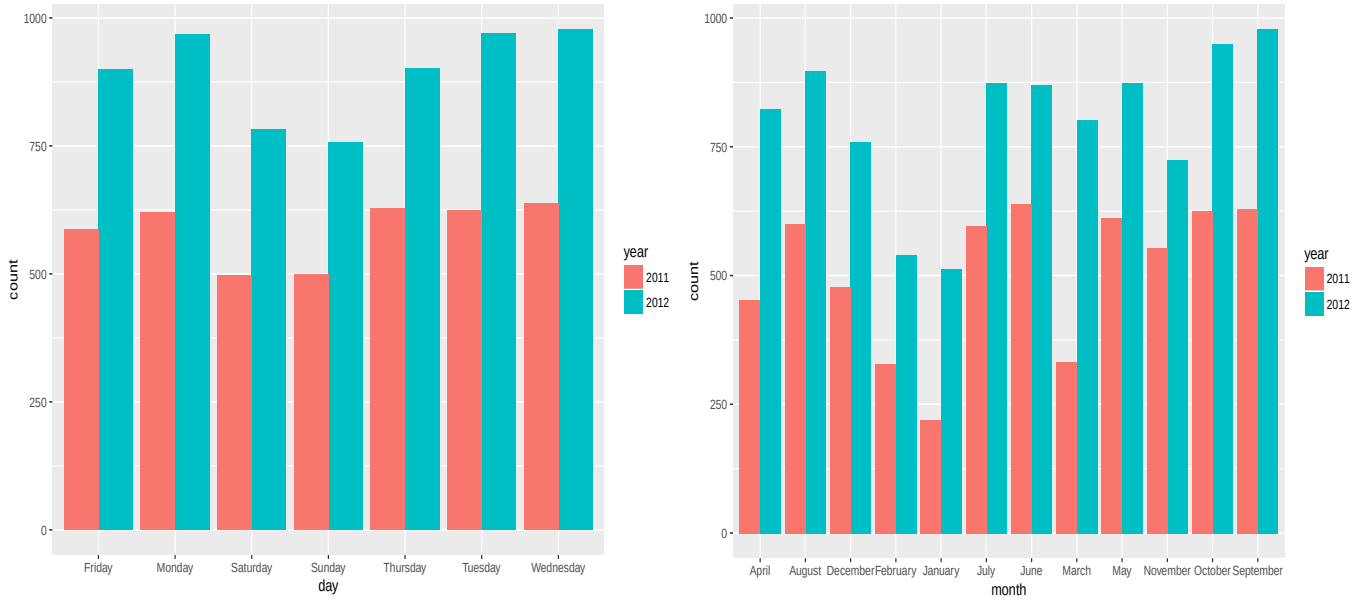


(e) count vs (holiday and hour)



(f) count vs (temp and holiday)

Figure 3: Trends of count versus two more factors



(a) count vs (year and day)

(b) count vs (year and month)

Figure 4: Histogram plots of count values in different years

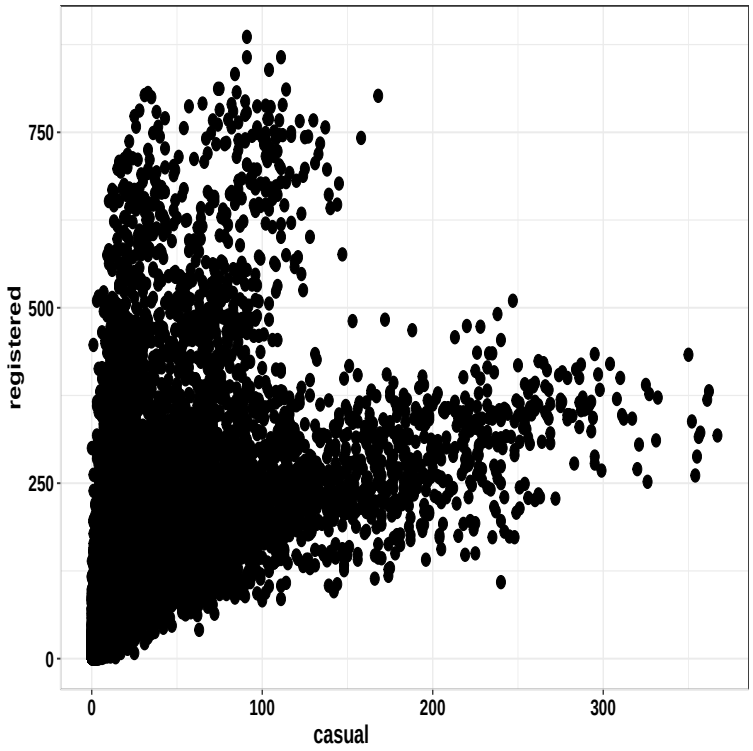
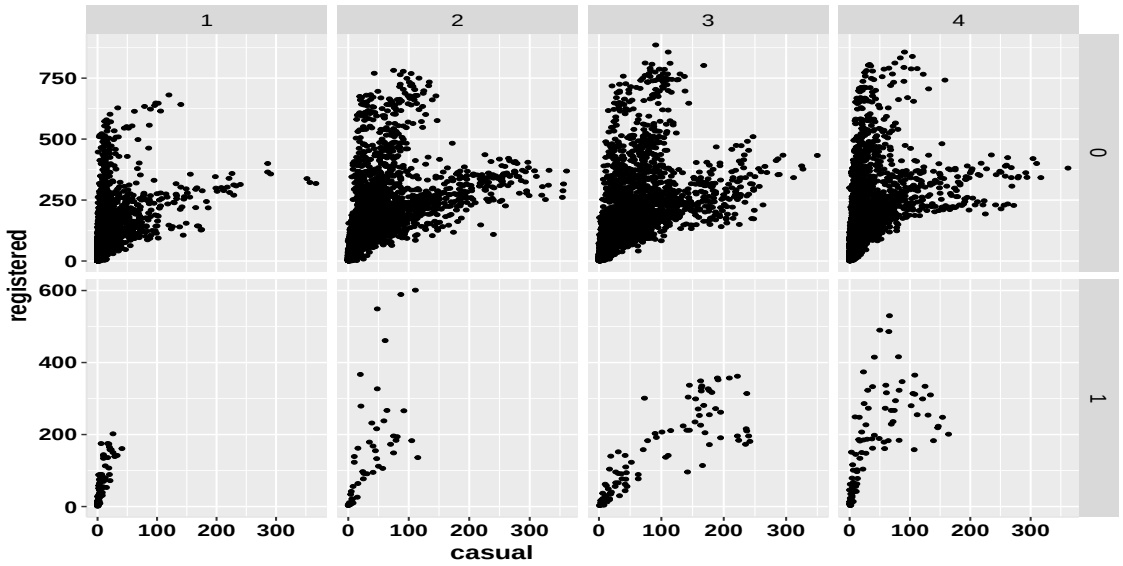
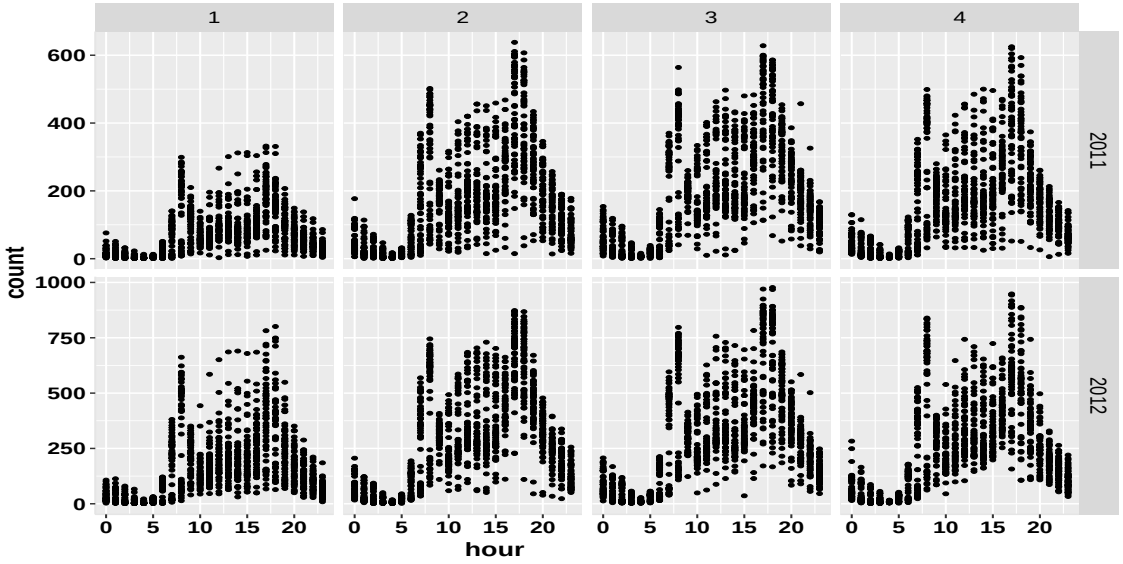


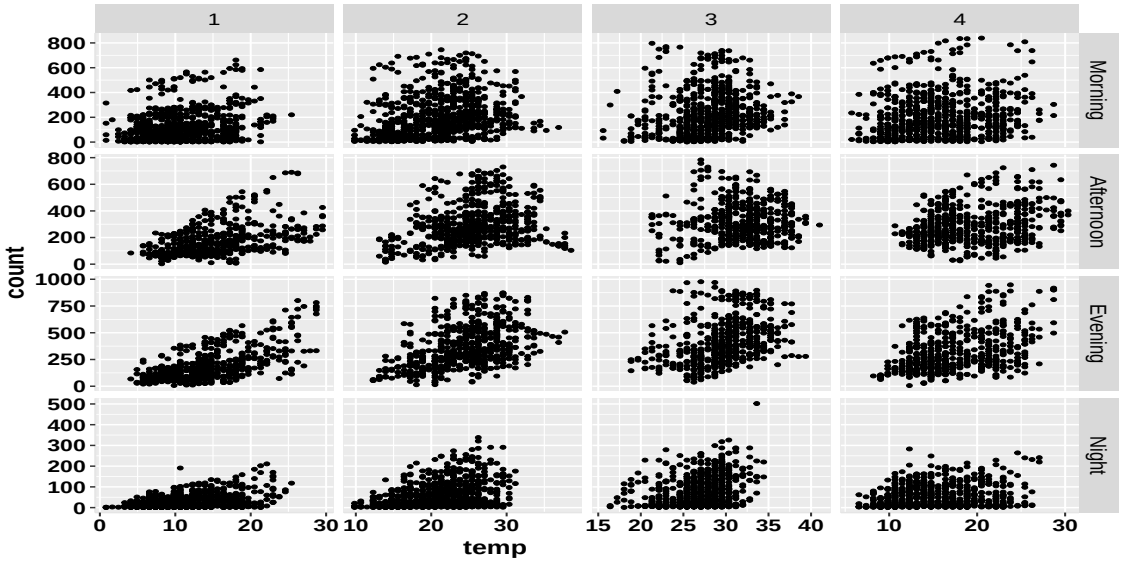
Figure 5: relationship between registered and casual



(a) casual vs registered of diff. season and holiday



(b) count vs hour of diff. year and season



(c) count vs temp of diff. time and season

Figure 6: Scatter plots of count vs three different features

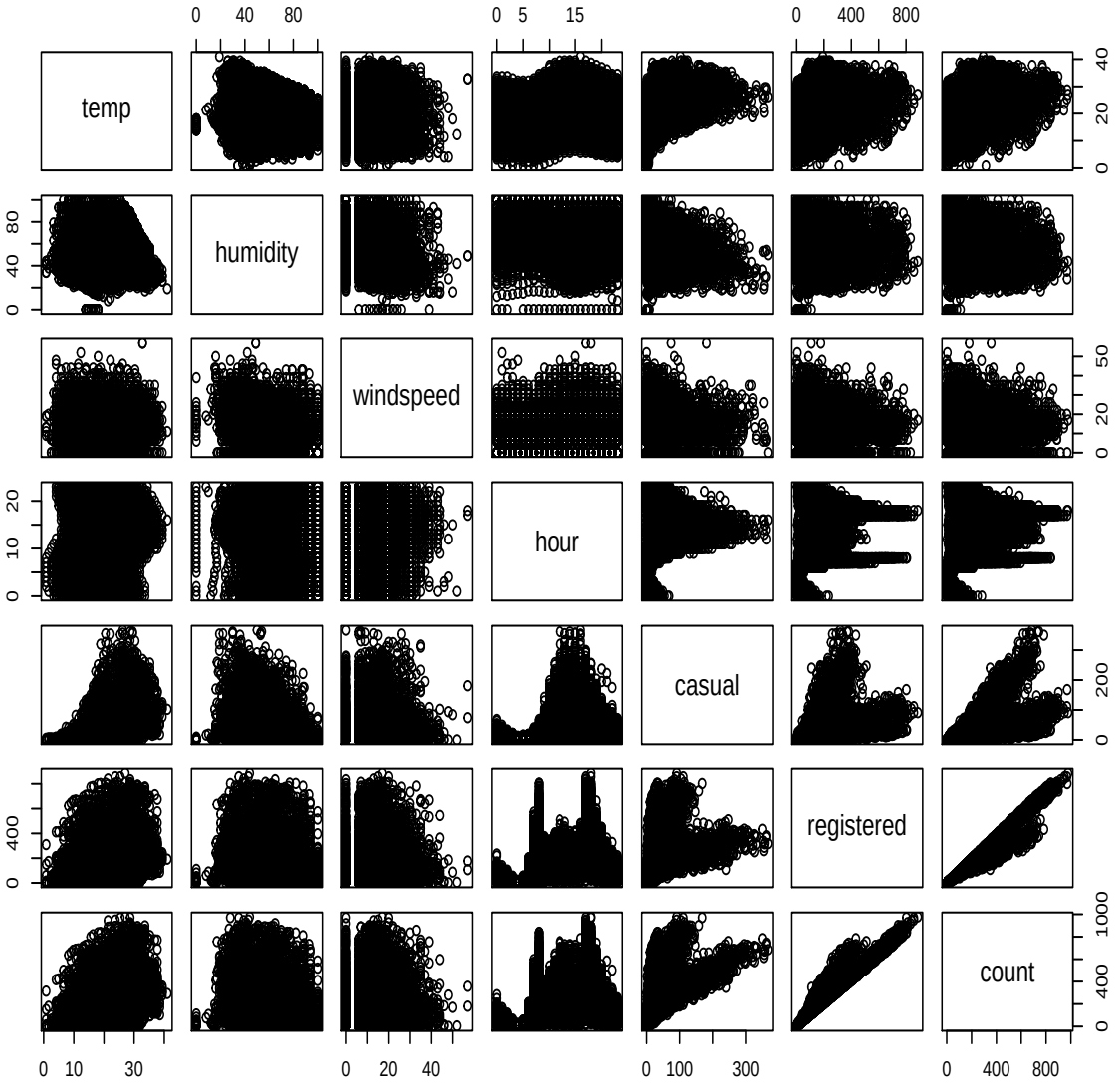


Figure 7: Pairwise relationship between selected features

where at first the unknown parameters β_i, $i = 0, \ldots, k$ are estimated from the train data set. Then, applying this model on the test set, the desired response values are estimated. In this report, the two different response variables are considered as 'registered' and 'casual' where all the remaining features (but 'count') are the potential predictors. So, we have two similar models by regressing either 'registered' and 'casual' on the predictors.

In both models, at first, the response variable is regressed on all other features (full model) and then, for variable selection, the reduced model will be the best reduced model in terms of AIC criteria (reduced model). Then, the reduced model is compared to the full model based on the likelihood ratio test using ANOVA r-code, ending up the reduced model is preferred. Then, Box-Cox transformation is applied to see whether it improve the prediction accuracy. The r-codes are shown below.

```
> library("bestglm")
> library("psych")
> lin.casual<-lm(casual~season+ holiday+ workingday+ weather+ temp+ atemp
+                + humidity+ windspeed+ hour+ time+ day+ month+ year, data=train)
>
> # summary(lin.casual)
> # stepcas<-step(lin.casual); # model selection
> # anova(stepcas) # model evaluation
> # reduced lm model
> lin.casual2<-lm(casual~ holiday+ weather+ temp+ humidity+ hour+ time+ day+
+         month+ year, data=train)
> #summary(lin.casual2)
> anova(lin.casual,lin.casual2) # compare full model to the reduced one
Analysis of Variance Table

Model 1: casual ~ season + holiday + workingday + weather + temp + atemp +
humidity + windspeed + hour + time + day + month + year
Model 2: casual ~ holiday + weather + temp + humidity + hour + time +
day + month + year
Res.Df      RSS Df Sum of Sq      F    Pr(>F)
1  10855 12062335
2  10857 12090896 -2    -28561 12.851 2.663e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
> # to use boxcox transformation, casual must be positive
> lin.casual22<-lm(casual+1~ holiday+ weather+ temp+
+     humidity+ hour+ time+ day+ month+ year, data=train)
> # boxcox(lin.casual22, plotit=T) # lambda = 0.1 is identified
> lin.casual22<-lm((((casual+1)^(0.1)-1)/0.1)~ holiday+
+     weather+ temp+ humidity+ hour+ time+ day+ month+ year, data=train)
> z.casual=predict.lm(lin.casual22,  newdata=test)
> lm.casual<-round((1+0.1*z.casual)^(10))
> # CHECK THE SIMILARITY of the valuaes for train$casual and lm.casual
```

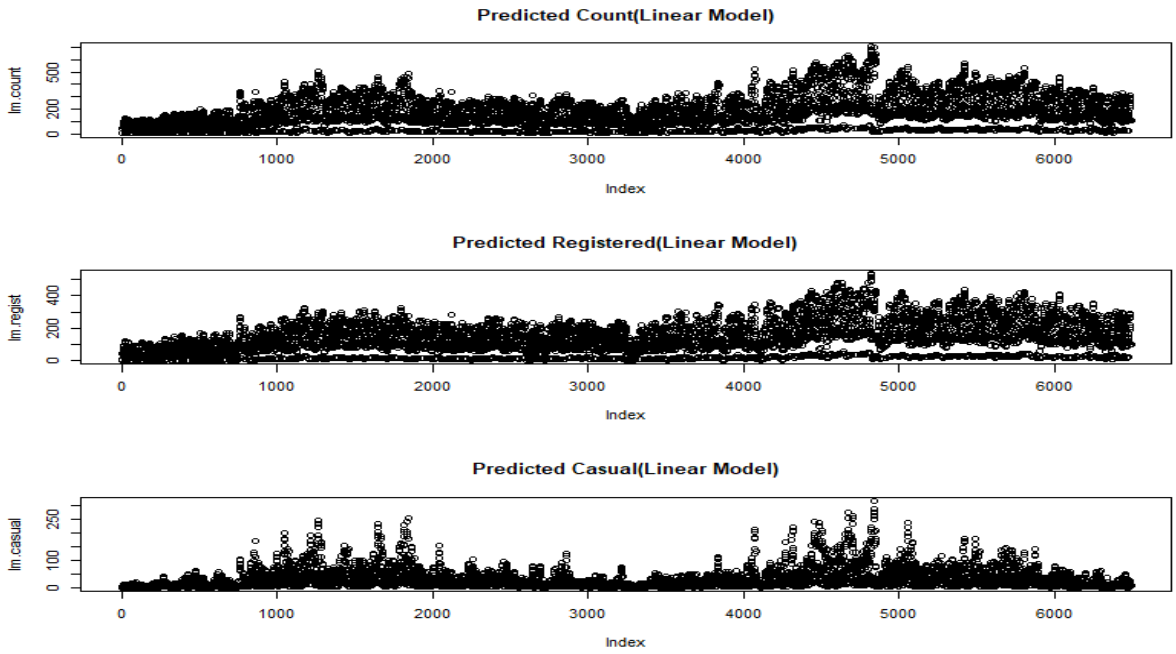


Figure 8: Predicted values of count, casual and registered in test set using linear model

```
> summary(lm.casual)
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.00    6.00   17.00   30.94   40.00  319.00
> summary(train$casual)
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.00    4.00   17.00   36.02   49.00  367.00
>
```

The values for 'registered' are similarly predicted. By adding these two obtained columns, the predicted values of 'count' are computed.

```
lm.count<-lm.regist+lm.casual
####create output file
lm.submit <- data.frame (datetime = test$datetime, count = lm.count )
write.csv(lm.submit, file = "lm_Prediction.csv", row.names=FALSE)
# result 0.87727
```

One may compare the predicted values obtained by linear model shown in Figure 8 to the values from the train set in Figure 1. There are some similarities between the two figures.

3.2 Negative Binomial

Negative binomial regression is a type of generalized linear model where the dependent variable is a count of the number of times an event occurs. In negative binomial regression, the log of the mean (μ) of the response values of y is determined by k regressors (predictors) under the following model

$$\log\mu = \beta_0 + \beta_1\, x_1 + \beta_2\, x_2 + \cdots + \beta_k\, x_k,$$

where the y values have the parameterized negative binomial distribution

$$P(Y = y) = \frac{\Gamma(y + \alpha^{-1})}{\Gamma(y+1)\Gamma(\alpha^{-1})}\Big(\frac{1}{1+\alpha\,\mu}\Big)^{\alpha^{-1}}\Big(\frac{\alpha\,\mu}{1+\alpha\,\mu}\Big)^{y},$$

$\alpha > 0$ being the heterogeneity parameter and the unknown regression coefficients β_i, $i = 0, \ldots, k$ are estimated by maximum likelihood estimation technique.

Then, I applied Negative Binomial technique as a GLM model using *glm.nb* code from 'MASS' R-package for *casual* and *registered* separately using the following code. Here, using reduced model, based on AIC criteria, did not improve the accuracy.

```
> library("MASS")
> nb.casual<-glm.nb(casual~season+ holiday+ workingday+ weather+ temp+ atemp
+        + humidity+ windspeed+ hour+ time+ day+ month+ year, data=train)
> nb.regist<-glm.nb(regeistered~season+ holiday+ workingday+ weather+ temp+
+      atemp+ humidity+ windspeed+ hour+ time+ day+ month+ year, data=train)
> nb.count<-nb.regist+nb.casual
> # Error 0.64515
```

One may see the predicted values computed by negative binomial regression model shown in Figure 9 and compare them to those of the train set in Figure 1.

3.3 Support Vector Regression

Support vector regression known as SVR is a modified version of the well-known classification technique of support vector machine (SVM) where the main objective of both methods is to maximize the distance function using a (non-linear) kernel function. Then, in the regression case, the model is represented as a function of the training points, not a function of features. One may find more information regarding SVM and SVR in the literature. To that aim, one needs *e1071* R-Package. The R-code for the prediction of 'casual' and 'registered' is shown below.

```
> library("e1071")
> Train1<-subset(train,select=-c(datetime,count,registered,casual))
> Test1<-subset(test,select=-c(datetime))
> Train1$casual <- train$casual
> reg.svm.cas = svm(formula = casual ~ ., data = train,
+                   type = 'eps-regression', kernel = 'radial')
>
> cas.svr = predict(reg.svm.cas, data=test)
> cas.svr=cas.svr*as.numeric(cas.svr>0)
```

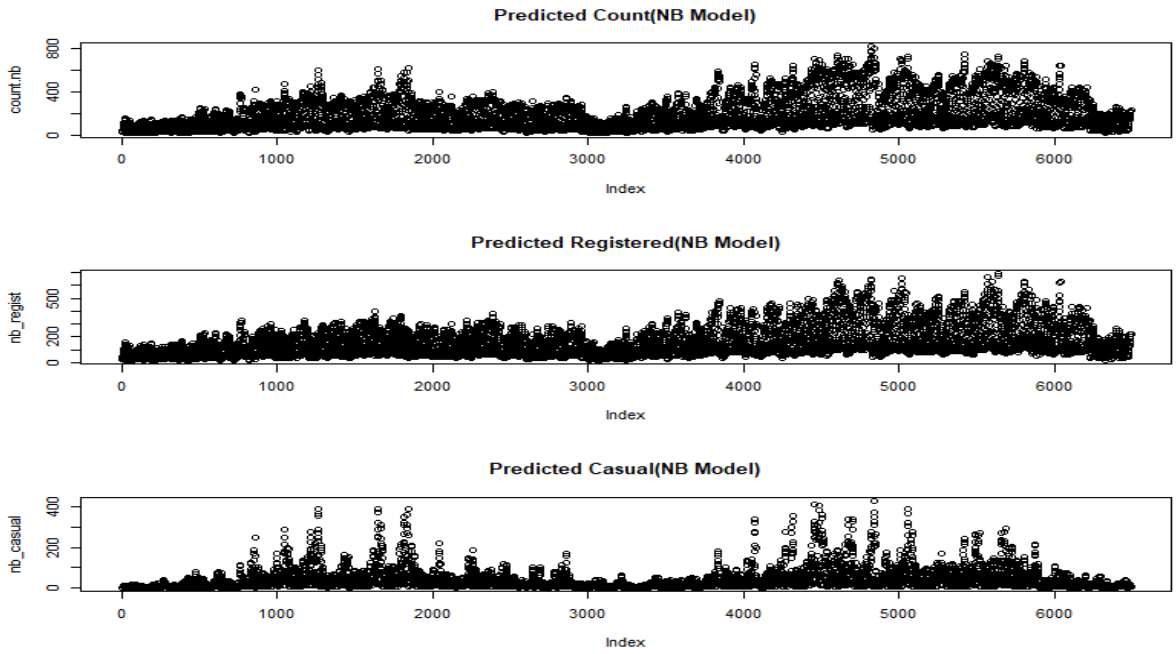


Figure 9: Predicted values of count, casual and registered in test set using negative binomial regression

```
> reg.svm.reg = svm(formula = registered ~ ., data = train,
+                   type = 'eps-regression', kernel = 'radial')
>
> reg.svr = predict(reg.svm.reg, data=test)
> reg.svr=reg.svr*as.numeric(reg.svr>0)
> count.svr=round(reg.svr+cas.svr)
```

One may compare the SVR predictions represented in Figure 10 to the those of the training set in Figure 1.

3.4 Random Forest

Random forests are a modified version of regression tree in which a single tree model has a certain level of accuracy for the purpose of prediction. In random forests regression, a large amount of such tree models are generated that the final predicted value is usually the average of predicted values determined by all available tree models. Therefore, this technique has a high level of accuracy in terms of prediction.

For random forest technique, the 'randomForest' R-package is required and the following R code was used to predict casual and registered amounts of the test set.

```
> library("randomForest")
> #fit casual   based on the others
> Train1<-subset(train,select=-c(datetime,count,registered))
> fit1=randomForest(casual~ .,data=Train1,ntree=500, mtry=5,
```

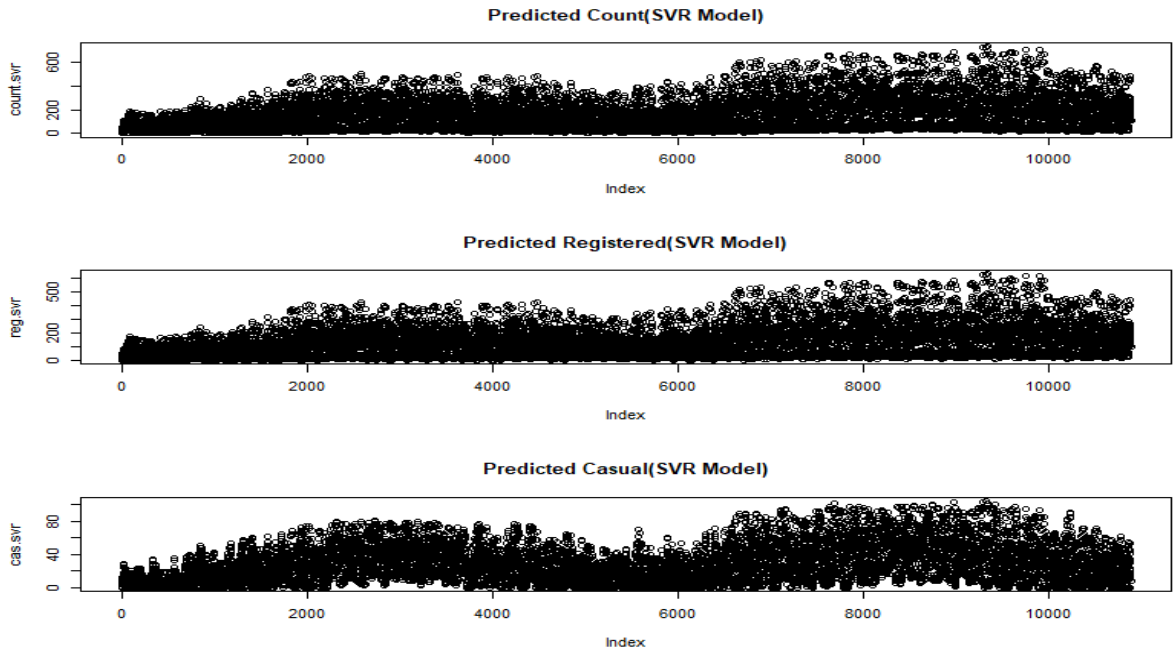


Figure 10: Predicted values of count, casual and registered in test set using support vector regression

```
+                    importance=TRUE)
> varImpPlot(fit1, main="Vars. select. for fitting casual(random forest)")
> casualFit <- randomForest(casual ~ hour + year + humidity + temp + atemp +
+    workingday + weekday, data=train, ntree=500, mtry=5, importance=TRUE)
> cas.rf <- round(predict(casualFit, test))
>
```

For this project, one may see the importance of each feature for predicting of 'registered' and 'casual' where year and hour definitely are not significant in these cases. The performance of the random forest prediction of 'count', 'casual' and 'registered' in Figure 12 could be compared to their corresponding in Figure 1.

3.5 Other Methods

Some other techniques like penalized regression methods including, LASSO, Elastic Net and SCAD, exponential regression, and so on might be used, while the selected techniques had good performances.

4 Conclusion

To sum up, I found out the accuracy of prediction using random forest technique is in higher level than the other mentioned methods; however, negative binomial (GLM) gives a good performance as well.

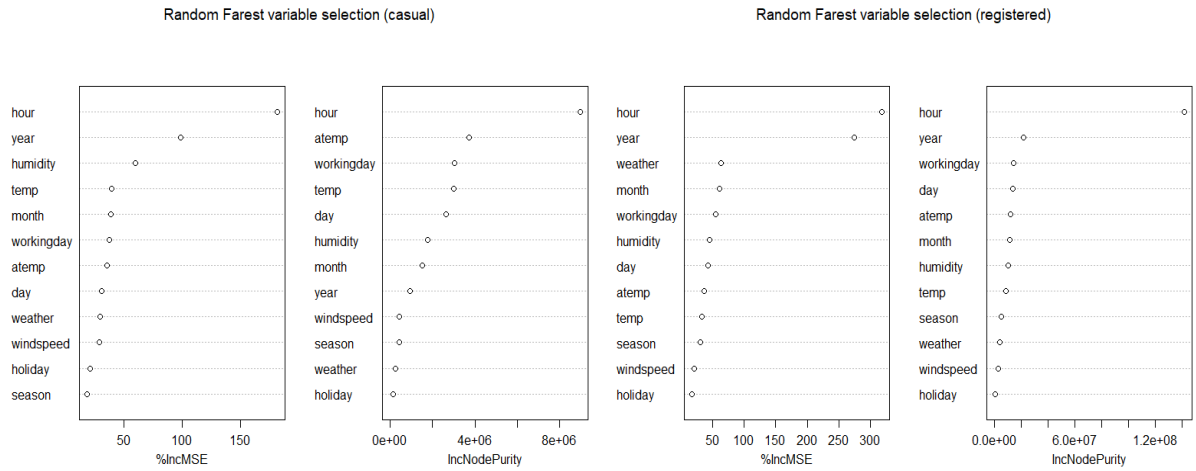


Figure 11: Variable selection using random forest techniques for both casual and registered

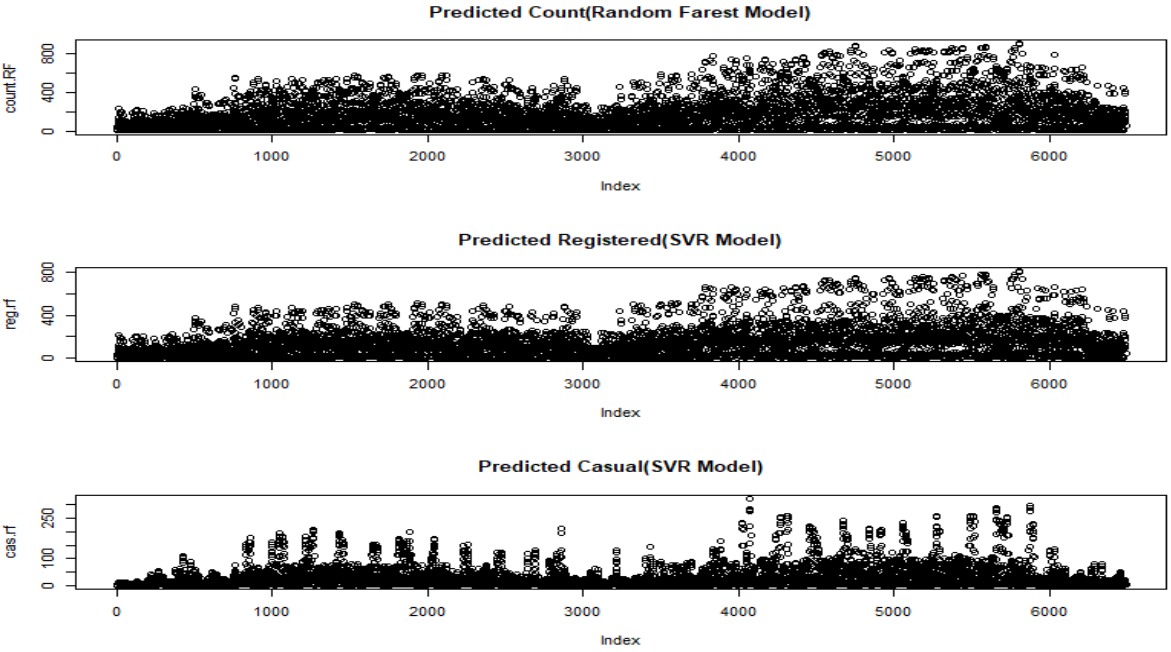


Figure 12: Predicted values of count, casual and registered in test set using random forest regression