Energy Forecasting

Sameer Lalit Lipsa

October 27, 2016

Table of Contents

## 1 Multiple-Linear Regression

Multiple linear regression is the most common form of linear regression analysis. As a predictive analysis, the multiple linear regression is used to explain the relationship between one continuous dependent variable from two or more independent variables. The independent variables can be continuous or categorical (dummy coded as appropriate).

### 1.1 Applying Multiple-Linear Regression to our dataset

#### 1.1.1 Dataset import and structure reset

We will import and check the structure of our cleansed dataset to be used to build our model

setwd("D:/Fall 16/ADS/Git Repo Assignemnt 2- Team 7/EnergyForecast")  
mergeData <- read.csv("MergedData.csv",header = TRUE)  
str(mergeData)

## 'data.frame': 8687 obs. of 22 variables:  
## $ X : int 1 2 3 4 5 6 7 8 9 10 ...  
## $ Date : Factor w/ 365 levels "2014-01-01","2014-01-02",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ hour : int 0 1 2 3 4 5 6 7 8 9 ...  
## $ Account : num 2.69e+10 2.69e+10 2.69e+10 2.69e+10 2.69e+10 ...  
## $ Channel : Factor w/ 1 level "MILDRED SCHOOL 1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Units : Factor w/ 1 level "kWh": 1 1 1 1 1 1 1 1 1 1 ...  
## $ month : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ day : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ year : int 2014 2014 2014 2014 2014 2014 2014 2014 2014 2014 ...  
## $ Day.of.Week : int 4 4 4 4 4 4 4 4 4 4 ...  
## $ weekday : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ Kwh : num 132 133 129 126 129 ...  
## $ PeakHour : int 0 0 0 0 0 0 0 1 1 1 ...  
## $ TemperatureF : num 23 21.9 21.9 21.9 21 21 21 19.9 23 24.1 ...  
## $ Dew\_PointF : num 5 3.9 3.9 3 3 3 3.9 5 6.1 5 ...  
## $ Humidity : num 46 46 46 44 46 46 48 52 48 44 ...  
## $ Sea\_Level\_PressureIn: num 30.2 30.2 30.2 30.3 30.3 ...  
## $ VisibilityMPH : num 10 10 10 10 10 10 10 10 10 10 ...  
## $ Wind\_SpeedMPH : num 8.1 11.5 12.7 11.5 9.2 11.5 10.4 10.4 10.4 16.1 ...  
## $ WindDirDegrees : num 290 290 240 250 260 270 250 230 260 300 ...  
## $ Conditions : Factor w/ 25 levels "Blowing Snow",..: 2 19 2 2 2 2 22 22 22 22 ...  
## $ Wind\_Direction : Factor w/ 18 levels "Calm","East",..: 17 17 18 18 16 16 18 14 16 17 ...

We would like to restructure some of the columns of the dataset and remove the unwanted columns, the datatypes of columns is understood intuitively by r, so we have to make sure the correct structure before proceeding for computation.

require(dplyr)  
mergeData <- mergeData %>% select(-c(X,Date,Account,Channel,Units,year))  
  
mergeData$hour <- as.numeric(mergeData$hour)  
mergeData$month <- as.numeric(mergeData$month)  
mergeData$Day.of.Week <- as.numeric(mergeData$Day.of.Week)  
mergeData$weekday <- as.numeric(mergeData$weekday)  
mergeData$PeakHour <- as.numeric(mergeData$PeakHour)  
mergeData$day <- as.numeric(mergeData$day)

We remove the categorical columns because it will not be useful in regression model and keeping subset of the data frame which only contains numeric features.

merge.sel <- subset(mergeData,select = -c(Conditions,Wind\_Direction))  
# Re-Check the structure  
str(mergeData)

## 'data.frame': 8687 obs. of 16 variables:  
## $ hour : num 0 1 2 3 4 5 6 7 8 9 ...  
## $ month : num 1 1 1 1 1 1 1 1 1 1 ...  
## $ day : num 1 1 1 1 1 1 1 1 1 1 ...  
## $ Day.of.Week : num 4 4 4 4 4 4 4 4 4 4 ...  
## $ weekday : num 1 1 1 1 1 1 1 1 1 1 ...  
## $ Kwh : num 132 133 129 126 129 ...  
## $ PeakHour : num 0 0 0 0 0 0 0 1 1 1 ...  
## $ TemperatureF : num 23 21.9 21.9 21.9 21 21 21 19.9 23 24.1 ...  
## $ Dew\_PointF : num 5 3.9 3.9 3 3 3 3.9 5 6.1 5 ...  
## $ Humidity : num 46 46 46 44 46 46 48 52 48 44 ...  
## $ Sea\_Level\_PressureIn: num 30.2 30.2 30.2 30.3 30.3 ...  
## $ VisibilityMPH : num 10 10 10 10 10 10 10 10 10 10 ...  
## $ Wind\_SpeedMPH : num 8.1 11.5 12.7 11.5 9.2 11.5 10.4 10.4 10.4 16.1 ...  
## $ WindDirDegrees : num 290 290 240 250 260 270 250 230 260 300 ...  
## $ Conditions : Factor w/ 25 levels "Blowing Snow",..: 2 19 2 2 2 2 22 22 22 22 ...  
## $ Wind\_Direction : Factor w/ 18 levels "Calm","East",..: 17 17 18 18 16 16 18 14 16 17 ...

The structure of the dataset is now ready for computation.

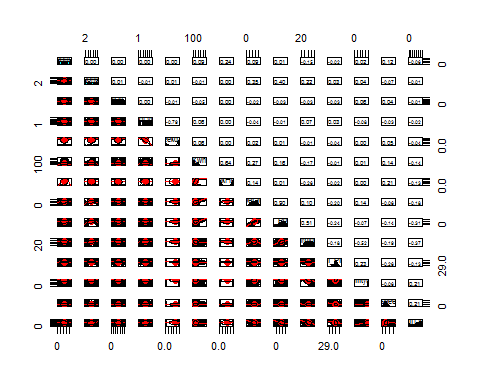
#### 1.1.2 Feature Transformation

There are multiple methods of feature creation and data transformation. Often, finding the right transformation of your data can reveal relationships that would be difficult see otherwise, and may also make it easier for your model to separate classes. In the simple case of creating linear models for regression, this can take the form of squaring terms, taking their logarithm, or other functional transformations that bring the fundamental problem back into a linear relationship through the transformation.

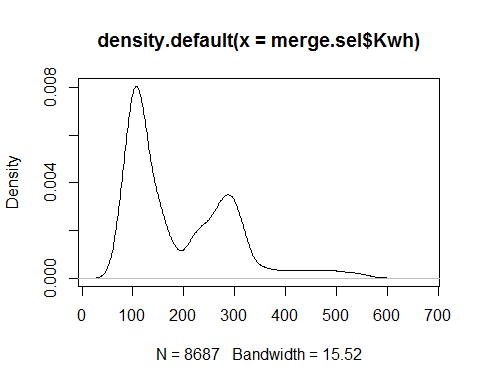
We will start by checking non linearity in the model. We will use the library psych

* psych: Adapted from the help page for pairs, pairs.panels shows a scatter plot of matrices (SPLOM), with bi variate scatter plots below the diagonal, histograms on the diagonal, and the Pearson correlation above the diagonal.

require(psych)  
pairs.panels(merge.sel,col="red")

 Using this plot we can identify the non linearity. To take a closer look we plot month.

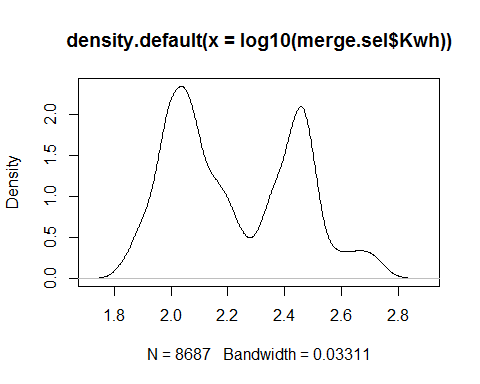
plot(density(merge.sel$Kwh))



As seen from the distribution of KwH, we will use a log scale to convert the feature, which is nothing but our dependent variable.

We plot again with transformed predictor i.e. KwH to check if the distribution still looks skewed as in last plot

plot(density(log10(merge.sel$Kwh)))

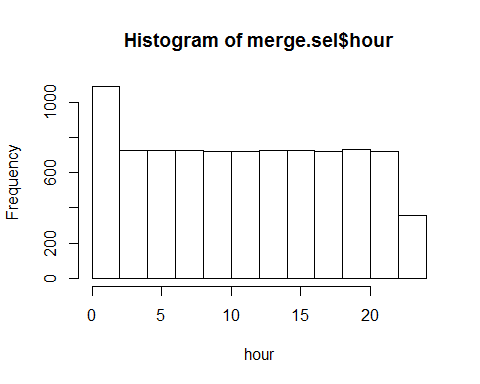


The plot looks very much distributed to we will update the transformed predictor value in our dataset as well.

merge.sel$Kwh <- log10(merge.sel$Kwh)

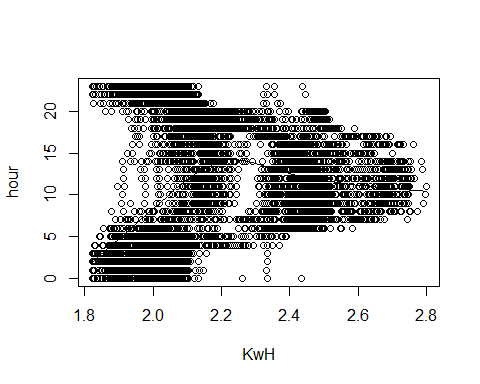
Our next focus is to check other feature with respect to kwH and see the relationship. Lets check the histogram of Day and also the relationship with kwH

hist(merge.sel$hour, xlab = 'hour')



We plot this graph again using scatter.smooth() function. \* scatter.smooth():Plot and add a smooth curve computed by loses to a scatter plot.

scatter.smooth(x=merge.sel$Kwh,y=merge.sel$hour,xlab = 'KwH', ylab = 'hour')



In r we have other methods for feature transformation, we find lambda transformation to be useful to demonstrate with Day Of Week parameter.

* forecast library: Methods and tools for displaying and analyzing uni variate time series forecasts including exponential smoothing via state space models and automatic ARIMA modelling.
* BoxCox.lambda(): It is procedure to identify an appropriate exponent (lambda) to transform the data to improve its normality.

We use lambda transformation for Day Of Week and update the transformed value to our dataset.

library(forecast)  
lambda <- BoxCox.lambda(merge.sel$Day.of.Week)  
merge.sel$Day.of.Week <- BoxCox(merge.sel$Day.of.Week,lambda = lambda)

#### 1.1.3 Feature Selection

Selecting the right features in your data can mean the difference between mediocre performance with long training times and great performance with short training times, thus we do feature selection.

We will use leaps library for feature selection. leaps() performs an exhaustive search for the best subsets of the variables in x for predicting y in linear regression, using an efficient branch-and-bound algorithm. It is a compatibility wrapper for regsubsets does the same thing better.

library(leaps)

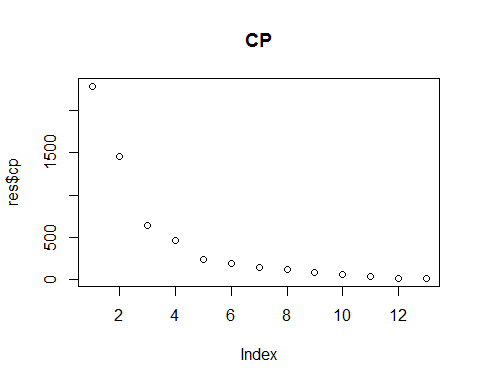
#### 1.1.3.1 Feature Selection Exhaustive Search

* regsubsets(): Model selection by exhaustive search, forward or backward stepwise, or sequential replacement

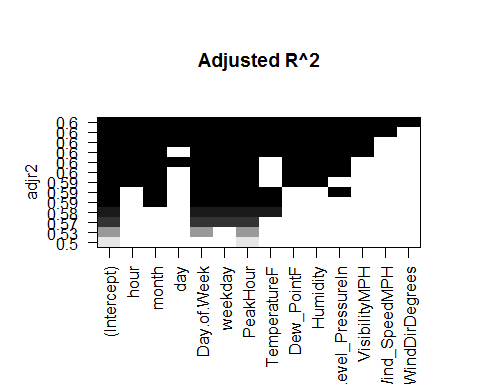
regsubsets.out <- regsubsets(Kwh ~ ., data = merge.sel, nvmax = 14,method = "exhaustive")  
res <- summary(regsubsets.out)

The regsubsets()function has a built-in plot command which can display the selected variables for the "best" model with a given model selection statistic. The top row of each plot contains a black square for each variable selected according to the optimal model associated with that statistic. Examples using the R2 (unadjusted), adjusted R2, Mallow's Ck, and the BIC are shown as below.

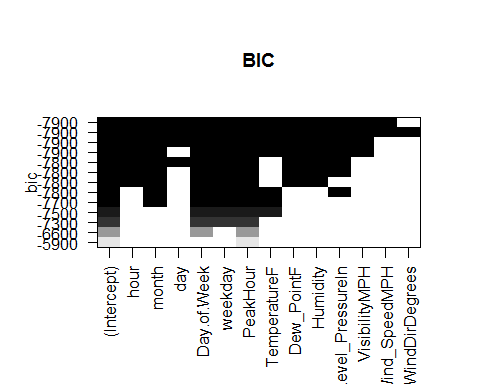
plot(res$cp, main = "CP")



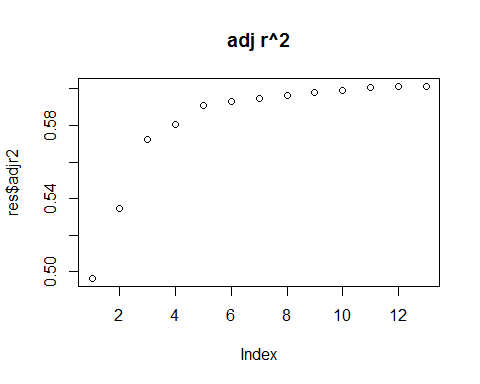
plot(regsubsets.out, scale = "adjr2", main = "Adjusted R^2")



plot(regsubsets.out, scale = "bic", main = "BIC")



plot(res$adjr2, main = "adj r^2")



coef(regsubsets.out,10)

## (Intercept) hour month   
## 0.986612992 -0.001518070 -0.006809980   
## Day.of.Week weekday PeakHour   
## 0.175084366 0.120601290 0.308369670   
## TemperatureF Dew\_PointF Humidity   
## -0.006847200 0.009347132 -0.004853796   
## Sea\_Level\_PressureIn VisibilityMPH   
## 0.041667612 -0.006239505

res$adjr2[10]

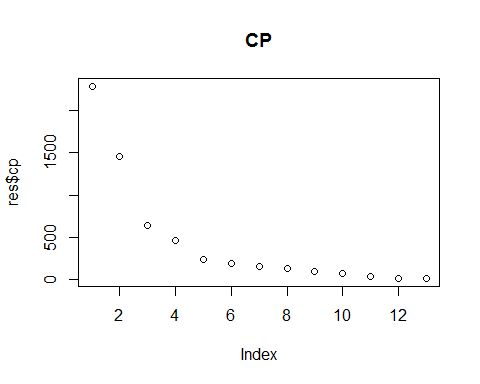
## [1] 0.5993363

#### 1.1.3.2 Feature Selection Forward Search

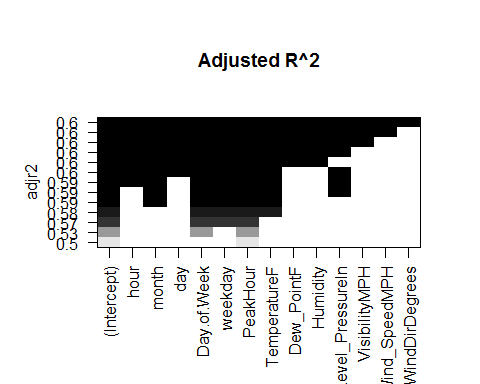
regsubsets.out <- regsubsets(Kwh ~ ., data = merge.sel, nvmax = 14,method = "forward")  
res <- summary(regsubsets.out)

Statistics:

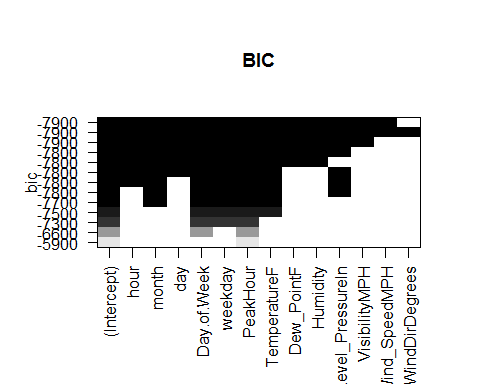
plot(res$cp, main = "CP")



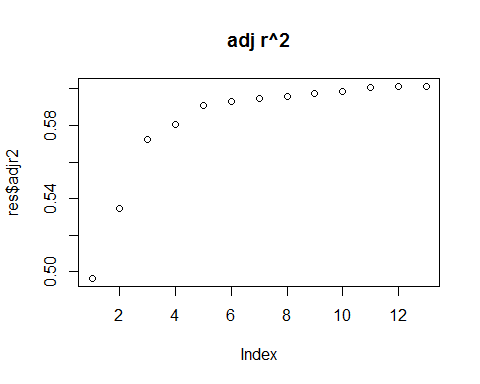
plot(regsubsets.out, scale = "adjr2", main = "Adjusted R^2")



plot(regsubsets.out, scale = "bic", main = "BIC")



plot(res$adjr2, main = "adj r^2")



coef(regsubsets.out,10)

## (Intercept) hour month   
## 1.0777456959 -0.0014628085 -0.0070198703   
## day Day.of.Week weekday   
## -0.0009535587 0.1773663822 0.1218297525   
## PeakHour TemperatureF Dew\_PointF   
## 0.3124898552 -0.0039129652 0.0060353058   
## Humidity Sea\_Level\_PressureIn   
## -0.0030071928 0.0323572775

res$adjr2[10]

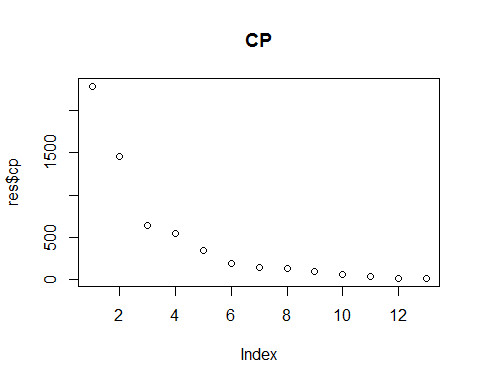
## [1] 0.5984361

#### 1.1.3.3 Feature Selection Backward Search

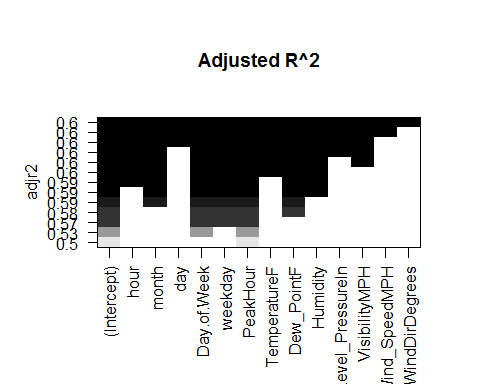
regsubsets.out <- regsubsets(Kwh ~ ., data = merge.sel, nvmax = 14,method = "backward")  
res <- summary(regsubsets.out)

Statistics:

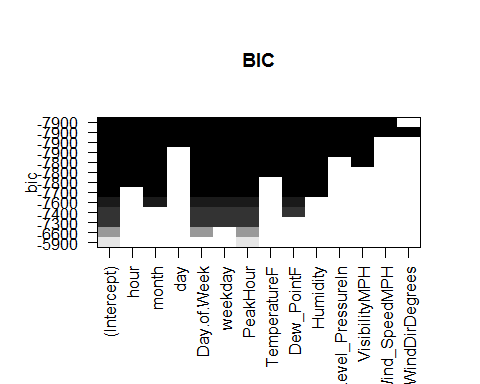
plot(res$cp, main = "CP")



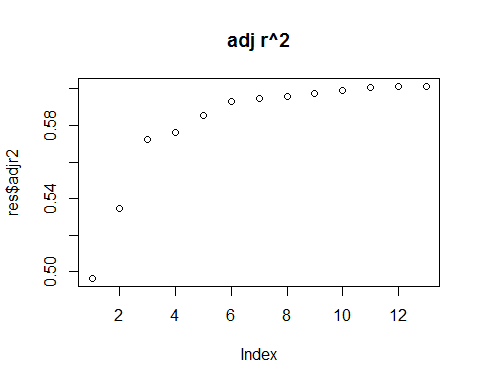
plot(regsubsets.out, scale = "adjr2", main = "Adjusted R^2")



plot(regsubsets.out, scale = "bic", main = "BIC")



plot(res$adjr2, main = "adj r^2")



coef(regsubsets.out,10)

## (Intercept) hour month   
## 0.986612992 -0.001518070 -0.006809980   
## Day.of.Week weekday PeakHour   
## 0.175084366 0.120601290 0.308369670   
## TemperatureF Dew\_PointF Humidity   
## -0.006847200 0.009347132 -0.004853796   
## Sea\_Level\_PressureIn VisibilityMPH   
## 0.041667612 -0.006239505

res$adjr2[10]

## [1] 0.5993363

#### 1.1.3.4 Feature Selection Manually selecting features

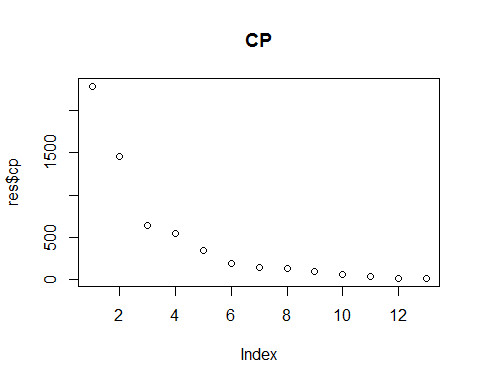
regsubsets.step <- regsubsets(Kwh~hour + month + day + Day.of.Week + weekday + PeakHour +  
 TemperatureF + Dew\_PointF + Humidity +  
 VisibilityMPH, data = merge.sel, nvmax =14 )  
summary(regsubsets.step)

## Subset selection object  
## Call: regsubsets.formula(Kwh ~ hour + month + day + Day.of.Week + weekday +   
## PeakHour + TemperatureF + Dew\_PointF + Humidity + VisibilityMPH,   
## data = merge.sel, nvmax = 14)  
## 10 Variables (and intercept)  
## Forced in Forced out  
## hour FALSE FALSE  
## month FALSE FALSE  
## day FALSE FALSE  
## Day.of.Week FALSE FALSE  
## weekday FALSE FALSE  
## PeakHour FALSE FALSE  
## TemperatureF FALSE FALSE  
## Dew\_PointF FALSE FALSE  
## Humidity FALSE FALSE  
## VisibilityMPH FALSE FALSE  
## 1 subsets of each size up to 10  
## Selection Algorithm: exhaustive  
## hour month day Day.of.Week weekday PeakHour TemperatureF  
## 1 ( 1 ) " " " " " " " " " " "\*" " "   
## 2 ( 1 ) " " " " " " "\*" " " "\*" " "   
## 3 ( 1 ) " " " " " " "\*" "\*" "\*" " "   
## 4 ( 1 ) " " " " " " "\*" "\*" "\*" "\*"   
## 5 ( 1 ) " " "\*" " " "\*" "\*" "\*" "\*"   
## 6 ( 1 ) " " "\*" " " "\*" "\*" "\*" " "   
## 7 ( 1 ) "\*" "\*" " " "\*" "\*" "\*" " "   
## 8 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" " "   
## 9 ( 1 ) "\*" "\*" " " "\*" "\*" "\*" "\*"   
## 10 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## Dew\_PointF Humidity VisibilityMPH  
## 1 ( 1 ) " " " " " "   
## 2 ( 1 ) " " " " " "   
## 3 ( 1 ) " " " " " "   
## 4 ( 1 ) " " " " " "   
## 5 ( 1 ) " " " " " "   
## 6 ( 1 ) "\*" "\*" " "   
## 7 ( 1 ) "\*" "\*" " "   
## 8 ( 1 ) "\*" "\*" " "   
## 9 ( 1 ) "\*" "\*" "\*"   
## 10 ( 1 ) "\*" "\*" "\*"

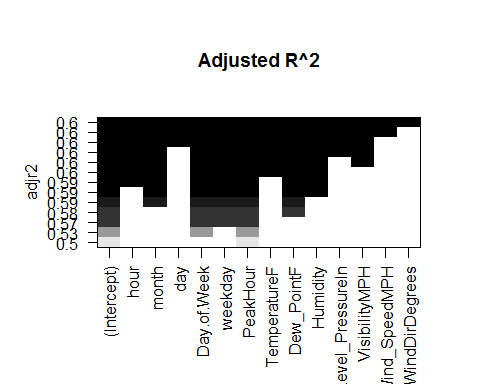
resstep <- summary(regsubsets.step)

Statistics:

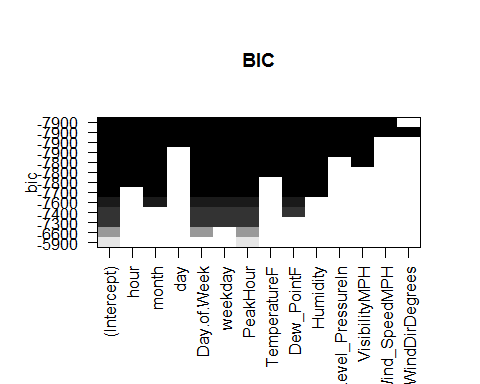
plot(res$cp, main = "CP")



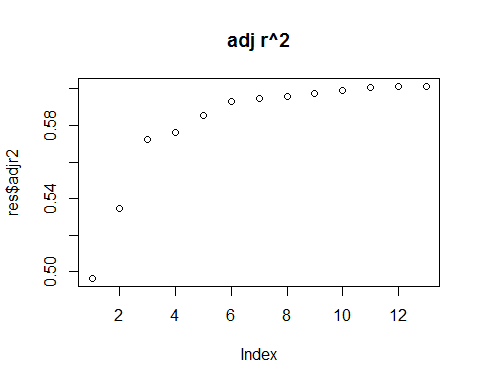
plot(regsubsets.out, scale = "adjr2", main = "Adjusted R^2")



plot(regsubsets.out, scale = "bic", main = "BIC")



plot(res$adjr2, main = "adj r^2")



coef(regsubsets.out,10)

## (Intercept) hour month   
## 0.986612992 -0.001518070 -0.006809980   
## Day.of.Week weekday PeakHour   
## 0.175084366 0.120601290 0.308369670   
## TemperatureF Dew\_PointF Humidity   
## -0.006847200 0.009347132 -0.004853796   
## Sea\_Level\_PressureIn VisibilityMPH   
## 0.041667612 -0.006239505

res$adjr2[10]

## [1] 0.5993363

#### 1.1.4 Develop a linear model

Now we Develop a linear model using variables selected from Backward approach. The model will be built using the training sample of the data. The model will be validated using the validation sample of the data.

##### 1.1.4.1 Split Validation

Split the data into training and validation samples. We will use (train.size)% for training and (100-train.size)% for validation

* set.seed(): Set the seed of R's random number generator, which is useful for creating simulations or random objects that can be reproduced.

We select 80% data for training and rest 20% for the testing and validation.

set.seed(2017)  
train.size <- 0.8  
train.index <- sample.int(length(merge.sel$Kwh),round(length(merge.sel$Kwh)\*train.size))  
train.sample <- merge.sel[train.index,]  
train.val <- merge.sel[-train.index,]

train.sample contains our training data and train.val contains our remaining validation data.

Multiple regression model uses a simple formula: Kwh = B0 + B1xTemperatureF + B2xPeakHour + B3xDew\_PointF

We will perform additional tests on training data. We will use a stepwise selection of variables by backward elimination. We will consider all possible candidate variables and eliminate one at a time.

* lm():lm is used to fit linear models. It can be used to carry out regression, single stratum analysis of variance and analysis of co-variance.

fit <- lm(Kwh ~ hour + month + Day.of.Week + weekday + PeakHour + TemperatureF + Dew\_PointF +  
 Humidity + Sea\_Level\_PressureIn + VisibilityMPH , data = train.sample)

By cheeking the summary of fit we would get to know the fit of our linear model.

summary(fit)

##   
## Call:  
## lm(formula = Kwh ~ hour + month + Day.of.Week + weekday + PeakHour +   
## TemperatureF + Dew\_PointF + Humidity + Sea\_Level\_PressureIn +   
## VisibilityMPH, data = train.sample)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.50077 -0.10272 0.00021 0.08930 0.49639   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.0740476 0.2314694 4.640 3.55e-06 \*\*\*  
## hour -0.0014895 0.0002559 -5.820 6.16e-09 \*\*\*  
## month -0.0067638 0.0005458 -12.392 < 2e-16 \*\*\*  
## Day.of.Week 0.1760833 0.0048277 36.473 < 2e-16 \*\*\*  
## weekday 0.1216719 0.0046563 26.130 < 2e-16 \*\*\*  
## PeakHour 0.3080442 0.0037442 82.272 < 2e-16 \*\*\*  
## TemperatureF -0.0064596 0.0012238 -5.278 1.34e-07 \*\*\*  
## Dew\_PointF 0.0089394 0.0012977 6.889 6.12e-12 \*\*\*  
## Humidity -0.0046908 0.0005996 -7.823 5.93e-15 \*\*\*  
## Sea\_Level\_PressureIn 0.0381529 0.0074093 5.149 2.69e-07 \*\*\*  
## VisibilityMPH -0.0060852 0.0009974 -6.101 1.11e-09 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.1422 on 6939 degrees of freedom  
## Multiple R-squared: 0.6011, Adjusted R-squared: 0.6006   
## F-statistic: 1046 on 10 and 6939 DF, p-value: < 2.2e-16

By plotting fit it provide checks for heteroscedasticity, normality, and influential observations.

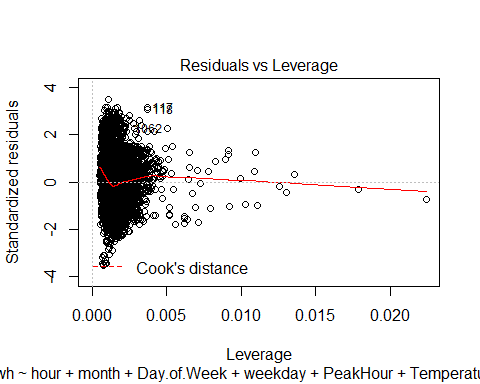
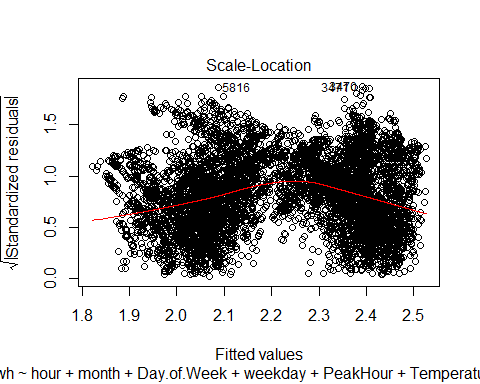
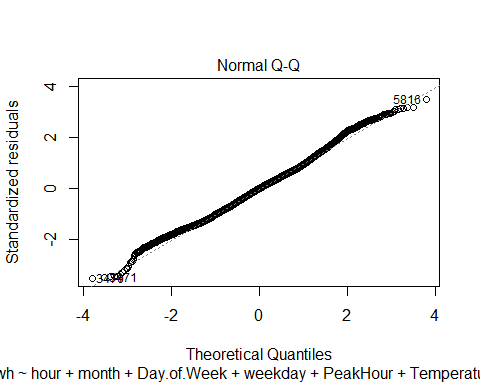
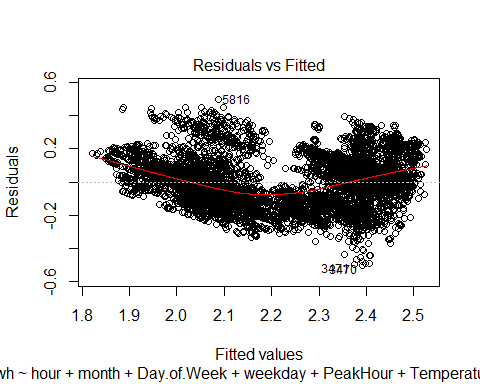
Residuals vs Fitted: This plot shows if residuals have non-linear patterns. There could be a non-linear relationship between predictor variables and an outcome variable and the pattern could show up in this plot if the model doesn't capture the non-linear relationship.

Normal Q-Q: This plot shows if residuals are normally distributed. Do residuals follow a straight line well or do they deviate severely? It's good if residuals are lined well on the straight dashed line.

Scale-Location: It's also called Spread-Location plot. This plot shows if residuals are spread equally along the ranges of predictors. This is how you can check the assumption of equal variance (homoscedasticity). It's good if you see a horizontal line with equally (randomly) spread points.

Residuals vs Leverage: This plot helps us to find influential cases (i.e., subjects) if any. Not all outliers are influential in linear regression analysis (whatever outliers mean).

plot(fit)

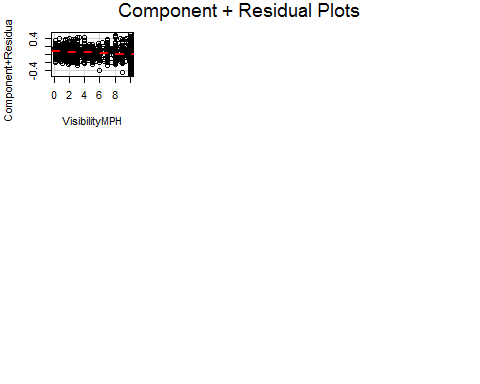
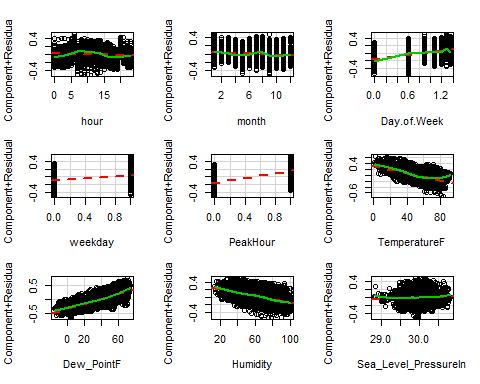


Lets check the distribution of errors around the model using car library. \* car library: It is used to compute Correlation and Linear Regression.

library(car)

Let us check the correlation and regression for fit.

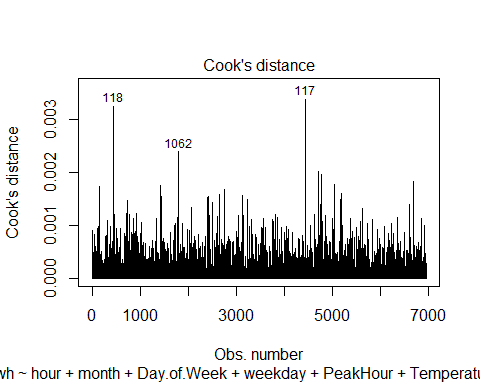
car::crPlots(fit)



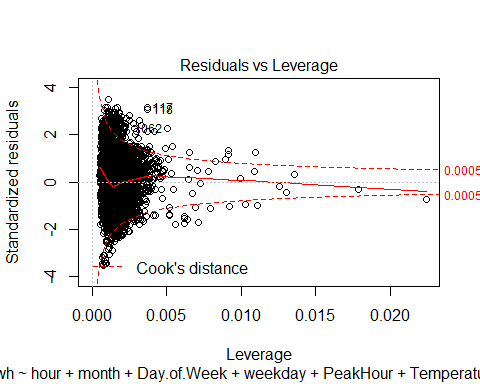
Red Line is the model and green line shows the fit of the variables.

Cook's distance is useful for identifying outliers in the X values (observations for predictor variables). It also shows the influence of each observation on the fitted response values. An observation with Cook's distance larger than three times the mean Cook's distance might be an outlier.

# Cook's D plot, cutoff as 4/(n-k-1)  
cutoff <- 4/((nrow(train.sample)-length(fit$coefficients)-2))  
# Identify D values > cutoff  
plot(fit,which=4, cook.levels = cutoff)



# Check, how important are these observations for the construction of the model  
plot(fit,which=5, cook.levels = cutoff)



After removing the not fitting values we will again check for fit our model.

fit <- lm(Kwh ~ hour + month + Day.of.Week + weekday + PeakHour + TemperatureF + Dew\_PointF +  
 Humidity + Sea\_Level\_PressureIn + VisibilityMPH , data = train.sample)

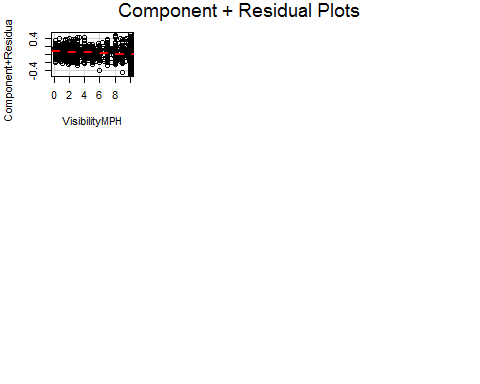
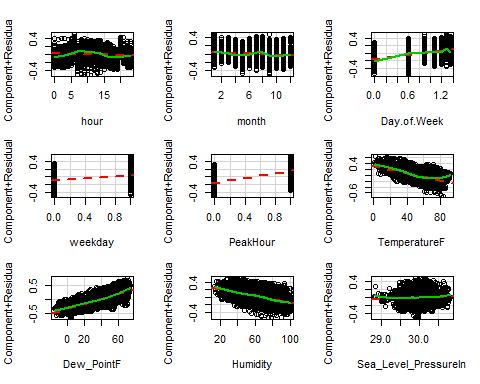
Checking the summary of fit again, we would get to know the new fit of our linear model.

summary(fit)

##   
## Call:  
## lm(formula = Kwh ~ hour + month + Day.of.Week + weekday + PeakHour +   
## TemperatureF + Dew\_PointF + Humidity + Sea\_Level\_PressureIn +   
## VisibilityMPH, data = train.sample)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.50077 -0.10272 0.00021 0.08930 0.49639   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.0740476 0.2314694 4.640 3.55e-06 \*\*\*  
## hour -0.0014895 0.0002559 -5.820 6.16e-09 \*\*\*  
## month -0.0067638 0.0005458 -12.392 < 2e-16 \*\*\*  
## Day.of.Week 0.1760833 0.0048277 36.473 < 2e-16 \*\*\*  
## weekday 0.1216719 0.0046563 26.130 < 2e-16 \*\*\*  
## PeakHour 0.3080442 0.0037442 82.272 < 2e-16 \*\*\*  
## TemperatureF -0.0064596 0.0012238 -5.278 1.34e-07 \*\*\*  
## Dew\_PointF 0.0089394 0.0012977 6.889 6.12e-12 \*\*\*  
## Humidity -0.0046908 0.0005996 -7.823 5.93e-15 \*\*\*  
## Sea\_Level\_PressureIn 0.0381529 0.0074093 5.149 2.69e-07 \*\*\*  
## VisibilityMPH -0.0060852 0.0009974 -6.101 1.11e-09 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.1422 on 6939 degrees of freedom  
## Multiple R-squared: 0.6011, Adjusted R-squared: 0.6006   
## F-statistic: 1046 on 10 and 6939 DF, p-value: < 2.2e-16

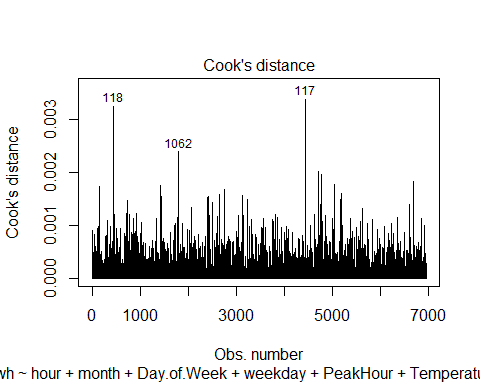
Also checking the correlation and regression for fit.

car::crPlots(fit)

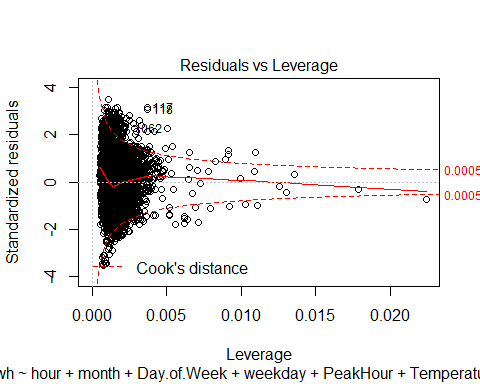


Plotting Cook's D plot to recheck the fit.

# Cook's D plot, cutoff as 4/(n-k-1)  
cutoff <- 4/((nrow(train.sample)-length(fit$coefficients)-2))  
# Identify D values > cutoff  
plot(fit,which=4, cook.levels = cutoff)



# Check, how important are these observations for the construction of the model  
plot(fit,which=5, cook.levels = cutoff)



# By observing above plots, we remove the records which are indicated to be not fitting.  
train.sample <- train.sample[-which(rownames(train.sample) %in% c("1062","117","8170")),]

Now one more time we are curious to know fit, so lets use lm()

fit <- lm(Kwh ~ hour + month + Day.of.Week + weekday + PeakHour + TemperatureF + Dew\_PointF + Humidity + Sea\_Level\_PressureIn + VisibilityMPH , data = train.sample)

We print the final Print the Regression Coefficients.

summary(fit) #R-squared: 0.5646, Adjusted R-squared: 0.5639

##   
## Call:  
## lm(formula = Kwh ~ hour + month + Day.of.Week + weekday + PeakHour +   
## TemperatureF + Dew\_PointF + Humidity + Sea\_Level\_PressureIn +   
## VisibilityMPH, data = train.sample)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.50090 -0.10228 0.00011 0.08932 0.49711   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.0480865 0.2315551 4.526 6.10e-06 \*\*\*  
## hour -0.0014996 0.0002557 -5.866 4.68e-09 \*\*\*  
## month -0.0067437 0.0005453 -12.366 < 2e-16 \*\*\*  
## Day.of.Week 0.1763925 0.0048232 36.572 < 2e-16 \*\*\*  
## weekday 0.1218559 0.0046514 26.198 < 2e-16 \*\*\*  
## PeakHour 0.3082507 0.0037395 82.432 < 2e-16 \*\*\*  
## TemperatureF -0.0066501 0.0012232 -5.437 5.62e-08 \*\*\*  
## Dew\_PointF 0.0091545 0.0012971 7.057 1.86e-12 \*\*\*  
## Humidity -0.0048027 0.0005996 -8.010 1.33e-15 \*\*\*  
## Sea\_Level\_PressureIn 0.0393247 0.0074123 5.305 1.16e-07 \*\*\*  
## VisibilityMPH -0.0062168 0.0009972 -6.234 4.82e-10 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.142 on 6936 degrees of freedom  
## Multiple R-squared: 0.6024, Adjusted R-squared: 0.6018   
## F-statistic: 1051 on 10 and 6936 DF, p-value: < 2.2e-16

We can see that we managed to increase adjusted r square value by 1% which is very insignificant.

Now we can evaluate the final linear model. We will find all predicted values for both a training and a validation set.

trainPred = subset(train.sample, select = -c(Kwh))  
validPred = subset(train.val, select = -c(Kwh))  
train.sample$Pred.Kwh <- predict(fit, newdata = trainPred)  
train.val$Pred.Kwh <- predict(fit, newdata = validPred)

Check how good is the model on the training set ~ correlation^2,RME, MAE and MAPE

val.corr <- round(cor(train.sample$Pred.Kwh,train.sample$Kwh),2)  
val.corr

## [1] 0.78

Lets check the data on how exactly has it predicted the Kwh values.

head(train.sample[,c("Kwh","Pred.Kwh")])

## Kwh Pred.Kwh  
## 8029 2.472844 2.417694  
## 4666 2.632559 2.430555  
## 4075 2.554053 2.352952  
## 2507 2.026533 2.058286  
## 6687 1.826852 2.102360  
## 6710 1.938870 2.134597

As we can see there is high correlation between actual and predicted. Lets calculate other parameters for the model. Note:Bringing back the transformed kwH with reverse transformation.

accuracy(10^train.val$Pred.Kwh, 10^train.val$Kwh)

## ME RMSE MAE MPE MAPE  
## Test set 11.32739 71.82698 50.65441 -5.83553 27.28802

write.csv(accuracy(10^train.val$Pred.Kwh, 10^train.val$Kwh),   
 file = "PerformanceMetrics\_split.csv")

We will also print coefficients of our model

coef(fit)

## (Intercept) hour month   
## 1.048086491 -0.001499607 -0.006743690   
## Day.of.Week weekday PeakHour   
## 0.176392538 0.121855852 0.308250693   
## TemperatureF Dew\_PointF Humidity   
## -0.006650124 0.009154493 -0.004802746   
## Sea\_Level\_PressureIn VisibilityMPH   
## 0.039324700 -0.006216770

write.csv(coef(fit),file = "RegressionOutputs.csv")

We have processed our model using split validation, we can use cross validation to check for better results.

##### 1.1.4.2 Cross Validation

Cross-validation is one of the most widely-used method for model selection, and for choosing tuning parameter values. It calculates the estimated K-fold cross-validation prediction error for generalized linear models.

* caret library: The caret package (short for classification and regression training) contains functions to streamline the model training process for complex regression and classification problems.
* klaR library: Miscellaneous functions for classification and visualization.

library(caret)  
library(klaR)

We set the 80% of data as train data train.sample and test data as train.val.

train.size <- 0.8  
train.index <- sample.int(length(merge.sel$Kwh),round(length(merge.sel$Kwh)\*train.size))  
train.sample <- merge.sel[train.index,]  
train.val <- merge.sel[-train.index,]

We now control the training data.

* trainControl(): To modify the resampling method, a trainControl function is used. K is controlled by the number argument and defaults to 10

train\_control <- trainControl(method="cv", number=3)

Lets train the model now.

model <- train(Kwh ~ hour + month + Day.of.Week + weekday + PeakHour + TemperatureF + Dew\_PointF +  
 Humidity + Sea\_Level\_PressureIn + VisibilityMPH , data = train.sample,  
 trControl=train\_control,method = "lm")

We summarize results to have look at our model.

print(model)

## Linear Regression   
##   
## 6950 samples  
## 10 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (3 fold)   
## Summary of sample sizes: 4633, 4632, 4635   
## Resampling results:  
##   
## RMSE Rsquared   
## 0.1436707 0.5928099  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE  
##

We now pridict the Kwh values using this model

trainPred = subset(train.sample, select = -c(Kwh))  
validPred = subset(train.val, select = -c(Kwh))  
train.sample$Pred.Kwh <- predict(model, newdata = trainPred)  
train.val$Pred.Kwh <- predict(model, newdata = validPred)

We will now check the accuracy of our prediction.

accuracy(10^train.val$Pred.Kwh, 10^train.val$Kwh)

## ME RMSE MAE MPE MAPE  
## Test set 14.3247 71.49257 49.56543 -3.871741 25.79423

write.csv(accuracy(10^train.val$Pred.Kwh, 10^train.val$Kwh),   
 file = "PerformanceMetrics\_cross.csv")

We can see that accuracy has ben improved compared to split validation.