Energy Forecast Analysis

Team 4

Ashish Dass, Anamika Jha, Shruti Narain

October 28, 2016

Abstract

In this assignment we have focused on developing a model designed for power consumption forecasting using multiple linear regressions. The scripts and the modeling have been done using R. The powers consumed for a single account based on different factors are forecasted. The forward, backward and step wise selection of variables are done to reach to an optimum model for prediction. Use of "Prune Trees" are done to get a visualization of the decision rules of dependent variables.

1. Sample Section

A part of the sample data comes from the data collected in 2014 at the Mildred school in the form of Raw-Data1.csv and Rawdata2.csv. Other half of the data has been extracted online form https://www.wunderground.com/weather. The data from both the sources has been combined using R script and the below modification has been done.

1.1 Data from csv files

Power: The power column contains values in 3 different units (kWh, kVARh and Power factor). The expected result was to get the power in kWh so for every hour of every single day kVARh multiplied with power factor gives kWh power consumption. Hence the value obtained from this multiplication summed with the kWh value gives us the total power consumed for that hour.

- After the power consumption has been obtained for every day, the power value given for every 5 minutes is clubbed together to obtain the value for an hour.

Now that we have a column for power consumption of every day for every hour, we have added separate columns for hour, month, day, year, day of week, weekday and peak hour all of them derived from the date and hour column using the below logic:

- Hour: has been assigned 0-23 values starting from 00:00 to 23:59 based on the values hour column we achieved after clubbing every 5 mins of data.
- Month: January to December has been assigned values 1-12 respectively derived from the date column.
- Day: values assigned from 1-31 based on the date column.
- Year: the year value derived from date column.
- Day of Week: Assigned values 0-6 from Sun-Sat respectively to values taken from date column.
- Weekday: value 1 if the day is a weekday else 0 (with use of chron library).
- Peak hour: for 7 am to 7 pm value 1 has been assigned else 0.

1.2 Data from wunderground website

The rest of the columns have been retrieved from wunderground website with the use of devtools library and "Ram-N/weatherData" github install. The data that this website provides us for Boston city is not consistent with the format that we need. Hence data cleaning has been performed to make it consistent with our already created date and time columns.

- Some erroneous value in wind_speedMPH column were replaced with 0.
- Humidity columns had N/A in the records which were replaced by 0.
- erroneous values in Wind_Direction were replaced with NA.
- The clean data was then grouped using Date and hour and mean of Humidity, Visibility, Wind Speed, Seal level pressure, Temperature, Wind Direction degrees were calculated using summarize function.
- Top most values for wind direction and Condition were considered.

The two data sets are then merged based on the hour and date columns.

2.Multiple-Linear Regression

Of all the columns we have in our SampleOutput.csv we did variable selection based on the forward, backward and step wise approach.

Backward Selection:

```
install.packages("leaps")
library(leaps)
#### Backward selection
regfit.bwd=regsubsets(power_~.,data=matWeatherFinal3,nvmax=11, method
="backward")
B=summary(regfit.bwd)
names(B)
B
B$rss
B$adjr2
coef(regfit.bwd,6)
```

```r
### Regression (Subset selection)
### Needed package and datasets

matWeatherFinal3=na.omit(matWeatherFinal3) # Get rid of NAs
install.packages("leaps")
library(leaps)

#### Backward selection
regfit.bwd=regsubsets(power_~.,data=matWeatherFinal3 ,nvmax=11, method="backward")

B=summary(regfit.bwd)
names(B)
B
B$rss
B$adjr2
coef(regfit.bwd,6)

par(mfrow=c(2,2))
plot(B$rss ,xlab="Number of Variables ",ylab="RSS", type="l")
plot(B$adjr2 ,xlab="Number of Variables ", ylab="Adjusted RSq",type="l")
coef(regfit.bwd ,6)

```

18:1 (Top Level) R Script

Console E:/Sem4-Fall 2016/ADS/Assignment 2 Materials/

```
10  ( 1 ) " "          " "          " "
11  ( 1 ) " "          " "          " "
12  ( 1 ) " "          " "          " "
> B$rss
 [1] 462880793 443825151 423655917 402579078 379305392 356678985 336713251 316293857
 [9] 297164183 278849873 265368671 254489824
> B$adjr2
 [1] 0.03777876 0.07728140 0.11910887 0.16283365 0.21113773 0.25810695 0.29955247
 [8] 0.34195168 0.38167740 0.41971580 0.44770440 0.47028288
> coef(regfit.bwd,6)
(Intercept)    hour__8     hour__9    hour__10    hour__11    hour__12    hour__13
   364.8176   262.6524    278.5281    287.2542    291.9627    282.7587    275.0793
```

- The first mean squared value comes from all the variables selected.
- From step 2, one variable is being removed at a time.
- The variables are dropped one by one till we reach our best result.

	Sno	Mean R squared	Removed	Equation
Step1	1	0.7211		power_ ~ hour__ + month__ + day__ +dayofWeek__ + Weekday__ + Wind_Direction + Conditions + Temperature + Dew_PointF + Humidity + Sea_Level_PressureIn + VisibilityMPH + Wind_SpeedMPH + WindDirDegrees + peakOfhour_
Step 2	1	0.6308	hour__	power_ ~ hour__ + month__ + day__ +dayofWeek__ + Wind_Direction + Conditions + Temperature + Dew_PointF + Humidity + Sea_Level_PressureIn + VisibilityMPH + Wind_SpeedMPH + WindDirDegrees + peakOfhour_
	2	0.6797	month__	
	3	0.7145	day__	
	4	0.7069	dayofWeek__	
	5	0.7211	Weekday__	
	6	0.7171	Wind_Direction	
	7	0.7189	Conditions	
	8	0.721	Temperature	
	9	0.7205	Dew_PointF	
	10	0.7198	Humidity	
	11	0.7166	Sea_Level_PressureIn	
	12	0.7211	VisibilityMPH	
	13	0.721	Wind_SpeedMPH	
	14	0.7211	WindDirDegrees	
	15	0.7211	peakOfhour	

Step 3	1	0.6308	hour__
	2	0.6797	month__
	3	0.7145	day__
	4	0.6046	dayofWeek__
	5	0.7171	Wind_Direction
	6	0.7189	Conditions
	7	0.721	Temperature
	8	0.7205	Dew_PointF
	9	0.7198	Humidity
	10	0.7211	Sea_Level_PressureIn
	11	0.721	VisibilityMPH
	12	0.721	Wind_SpeedMPH
	13	0.7211	WindDirDegrees
	14	0.7211	peakOfhour_
Step 4	1	0.6308	hour__
	2	0.679	month__
	3	0.7145	day__
	4	0.6046	dayofWeek__
	5	0.7171	Wind_Direction
	6	0.7189	Conditions
	7	0.7209	Temperature
	8	0.7204	Dew_PointF

Summary of the most optimum model obtained using backward selection on train data:

```r
install.packages("car")
library(car)
samp <- floor(0.75* nrow(matWeatherFinal))
samp
set.seed(123)
training.index <- sample(seq_len(nrow(matWeatherFinal)), size= samp)
train <- matWeatherFinal[training.index, ]
test <- matWeatherFinal[-training.index, ]
h<-matWeatherFinal

na.omit(test)
na.omit(train)
#check sea level and temperature collinearity
lm.fit = lm(power_ ~ hour__ + month__ + day__ +dayofWeek__ +
              Wind_Direction + Temperature + Dew_PointF +
              Humidity + Sea_Level_PressureIn ,data=train)
summary(lm.fit)
RegressionOutput <- summary(lm.fit)$coefficients[,1]
write.csv(RegressionOutput, file = "RegressionOutput1.csv")

vif(lm.fit)
install.packages("forecast")
```

24:26 (Top Level) R Script

Console E:/Sem4-Fall 2016/ADS/Assignment 2 Materials/

```
Temperature            -1.8866     1.0855  -1.738 0.082239 .
Dew_PointF              4.3813     1.1648   3.761 0.000170 ***
Humidity               -2.9834     0.5135  -5.810 6.53e-09 ***
Sea_Level_PressureIn  -15.5664     8.3306  -1.869 0.061727 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 128.2 on 6426 degrees of freedom
  (52 observations deleted due to missingness)
Multiple R-squared:  0.7177,	Adjusted R-squared:  0.7137
F-statistic: 179.5 on 91 and 6426 DF,  p-value: < 2.2e-16
```

Regression output:

```
samp
set.seed(123)
training.index <- sample(seq_len(nrow(matWeatherFinal)), size= samp)
train <- matWeatherFinal[training.index, ]
test <- matWeatherFinal[-training.index, ]
h<-matWeatherFinal

na.omit(test)
na.omit(train)
#check sea level and temperature collinearity
lm.fit = lm(power_ ~ hour__ + month__ + day__ +dayofWeek__ +
              Wind_Direction + Temperature + Dew_PointF +
              Humidity + Sea_Level_PressureIn ,data=train)
summary(lm.fit)
RegressionOutput <- summary(lm.fit)$coefficients[,1]
write.csv(RegressionOutput, file = "RegressionOutput1.csv")

vif(lm.fit)
install.packages("forecast")
library(forecast)
pred=predict(lm.fit,test)
PerformanceMetrics <- accuracy(pred,train$power_)
```

```
18:17  (Top Level) ÷                                          R Script
```

Console E:/Sem4-Fall 2016/ADS/Assignment 2 Materials/

```
Multiple R-squared:  0.7177,    Adjusted R-squared:  0.7137
F-statistic: 179.5 on 91 and 6426 DF,  p-value: < 2.2e-16

> RegressionOutput <- summary(lm.fit)$coefficients[,1]
> RegressionOutput
          (Intercept)               hour__1               hour__2
          669.1230718             2.6196710            13.4885581
              hour__3               hour__4               hour__5
           10.0119812            17.9064844            37.0014994
              hour__6               hour__7               hour__8
          236.7689732           356.0020129           378.0166407
              hour__9              hour__10              hour__11
          390.5932896           392.3482593           394.3028698
             hour__12              hour__13              hour__14
```

Performance matrix:

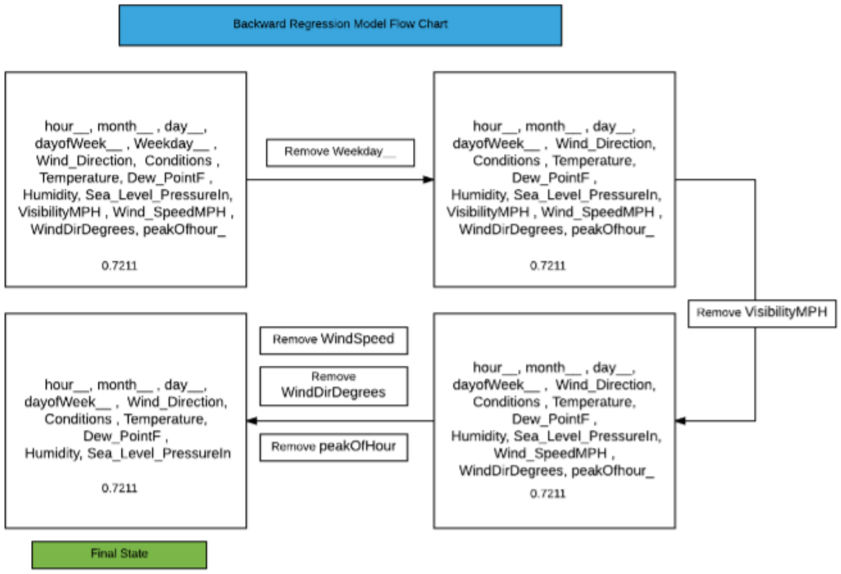
```
na.omit(test)
na.omit(train)
#check sea level and temperature collinearity
lm.fit = lm(power_ ~ hour__ + month__ + day__ +dayofWeek__ +
              Wind_Direction + Temperature + Dew_PointF +
              Humidity + Sea_Level_PressureIn ,data=train)
summary(lm.fit)
RegressionOutput <- summary(lm.fit)$coefficients[,1]
write.csv(RegressionOutput, file = "RegressionOutput1.csv")

vif(lm.fit)
install.packages("forecast")
library(forecast)
pred=predict(lm.fit,test)
PerformanceMetrics <- accuracy(pred,train$power_)

write.csv(t(PerformanceMetrics), file = "PerformanceMetrics1.csv")

lm.fit1 = lm(power_ ~ ., data=train)
summary(lm.fit1)

pred=predict(lm.fit1,test)
```

```
25:9  (Top Level) ÷                                          R Script ÷
```

Console E:/Sem4-Fall 2016/ADS/Assignment 2 Materials/

```
           39.8053023             5.1920003            17.2130871
     Wind_DirectionSE           Temperature            Dew_PointF
           31.9707317            -1.8866461             4.3812549
             Humidity  Sea_Level_PressureIn
           -2.9833748           -15.5663799
> t(PerformanceMetrics)
      Test set
ME   -14.29269
RMSE 982.01770
MAE  267.09981
MPE  -31.12927
MAPE  71.48662
>
```

Flowchart:



Step wise Selection:

Summary of the most optimum model obtained using step wise selection on train data:

```
install.packages("car")
library(car)
samp <- floor(0.75* nrow(matWeatherFinal))
samp
set.seed(123)
training.index <- sample(seq_len(nrow(matWeatherFinal)), size= samp)
train <- matWeatherFinal[training.index, ]
test <- matWeatherFinal[-training.index, ]
h<-matWeatherFinal

na.omit(test)
na.omit(train)
#check sea level and temperature collinearity
lm.fit = lm(power_ ~ Temperature +  month__ +  day__ + hour__ +dayofweek__, data=t
summary(lm.fit)
RegressionOutput <- summary(lm.fit)$coefficients[,1]
write.csv(RegressionOutput, file = "RegressionOutputForward.csv")

vif(lm.fit)
install.packages("forecast")
library(forecast)
```

16:1 (Top Level) R Script

Console E:/Sem4-Fall 2016/ADS/Assignment 2 Materials/

```
dayofweek__3 232.3008      6.0787  38.215  < 2e-16 ***
dayofweek__4 226.2204      6.0986  37.094  < 2e-16 ***
dayofweek__5 207.5756      6.1180  33.929  < 2e-16 ***
dayofweek__6 104.9973      6.1347  17.115  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 129.7 on 6446 degrees of freedom
  (52 observations deleted due to missingness)
Multiple R-squared:  0.7102,    Adjusted R-squared:  0.707
F-statistic: 222.5 on 71 and 6446 DF,  p-value: < 2.2e-16

>
```

Performance Matrix:

```r
test <- matWeatherFinal[-training.index, ]
h<-matWeatherFinal

na.omit(test)
na.omit(train)
#check sea level and temperature collinearity
lm.fit = lm(power_ ~ Temperature +  month__ +  day__ + hour__ +dayofweek__, data=t
summary(lm.fit)
RegressionOutput <- summary(lm.fit)$coefficients[,1]
write.csv(RegressionOutput, file = "RegressionOutputStepwise.csv")

vif(lm.fit)
install.packages("forecast")
library(forecast)
pred=predict(lm.fit,test)
PerformanceMetrics <- accuracy(pred,train$power_)

write.csv(t(PerformanceMetrics), file = "PerformanceMetricsForward.csv")

lm.fit1 = lm(power_ ~ ., data=train)
summary(lm.fit1)
```

21:2 (Top Level) R Script

Console E:/Sem4-Fall 2016/ADS/Assignment 2 Materials/

```
day__        1.324796 30        1.004699
hour__       1.243753 23        1.004753
dayofweek__ 1.246360  6         1.018522
> pred=predict(lm.fit,test)
> PerformanceMetrics <- accuracy(pred,train$power_)
> t(PerformanceMetrics)
       Test set
ME     9.321017
RMSE 343.005441
MAE  249.902608
MPE  -27.334111
MAPE  68.733958
```

Regression Output:

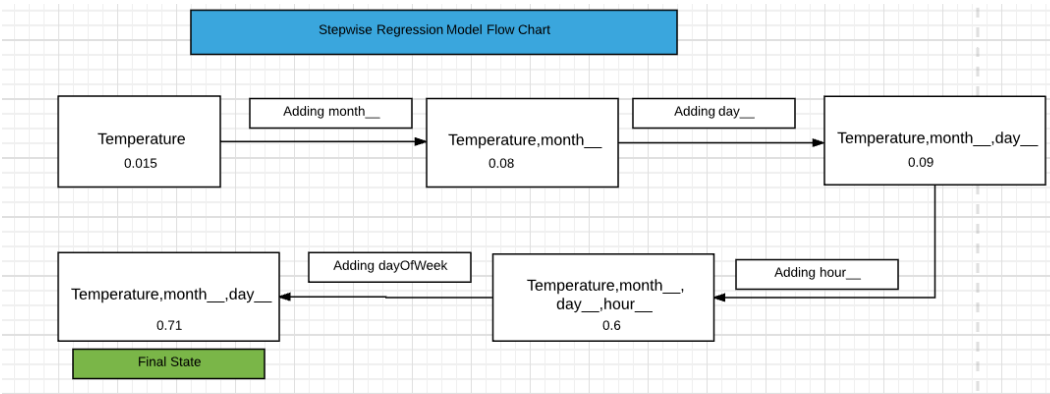
```r
samp
set.seed(123)
training.index <- sample(seq_len(nrow(matWeatherFinal)), size= samp)
train <- matWeatherFinal[training.index, ]
test <- matWeatherFinal[-training.index, ]
h<-matWeatherFinal

na.omit(test)
na.omit(train)
#check sea level and temperature collinearity
lm.fit = lm(power_ ~ Temperature +  month__ +  day__ + hour__ +dayofweek__, data=t
summary(lm.fit)
RegressionOutput <- summary(lm.fit)$coefficients[,1]
write.csv(RegressionOutput, file = "RegressionOutputStepwise.csv")

vif(lm.fit)
install.packages("forecast")
library(forecast)
pred=predict(lm.fit,test)
PerformanceMetrics <- accuracy(pred,train$power_)
```

17:61 (Top Level) R Script

Console E:/Sem4-Fall 2016/ADS/Assignment 2 Materials/

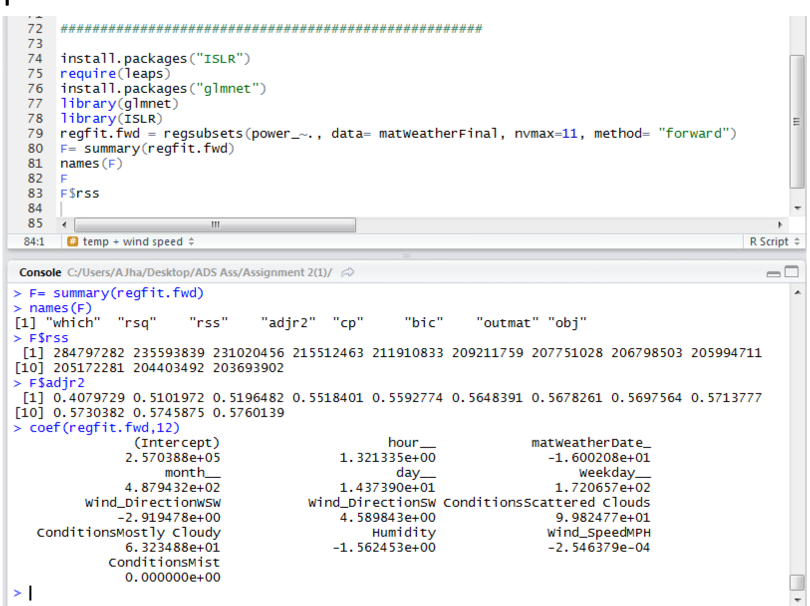
```
Residual standard error: 129.7 on 6446 degrees of freedom
  (52 observations deleted due to missingness)
Multiple R-squared:  0.7102,	Adjusted R-squared:  0.707
F-statistic: 222.5 on 71 and 6446 DF,  p-value: < 2.2e-16

> RegressionOutput <- summary(lm.fit)$coefficients[,1]
> RegressionOutput
 (Intercept)  Temperature     month__2     month__3     month__4     month__5
 -24.4577501    1.7877639  -25.1623248  -22.0503418  -77.0812015  -89.8381607
    month__6     month__7     month__8     month__9    month__10    month__11
 -18.7332802   50.6586782   86.0343823  -46.9085329  -97.1714806  -63.0722209
   month__12       day__2       day__3       day__4       day__5       day__6
 -81.0270548   79.5594709   39.2386569   15.5880244   54.3827346   36.2389357
      day__7       day__8       day__9      day__10      day__11      day__12
```

Flowchart:



Forward Selection:

install.packages("ISLR")
require(leaps)
install.packages("glmnet")
library(glmnet)
library(ISLR)
regfit.fwd = regsubsets(power_~., data= matWeatherFinal, nvmax=11, method= "forward")
F= summary(regfit.fwd)
names(F)
F



```
########################################################

install.packages("ISLR")
require(leaps)
install.packages("glmnet")
library(glmnet)
library(ISLR)
regfit.fwd = regsubsets(power_~., data= matWeatherFinal, nvmax=11, method= "forward")
F= summary(regfit.fwd)
names(F)
F
F$rss
```

```
> F= summary(regfit.fwd)
> names(F)
[1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
> F$rss
 [1] 284797282 235593839 231020456 215512463 211910833 209211759 207751028 206798503 205994711
[10] 205172281 204403492 203693902
> F$adjr2
 [1] 0.4079729 0.5101972 0.5196482 0.5518401 0.5592774 0.5648391 0.5678261 0.5697564 0.5713777
[10] 0.5730382 0.5745875 0.5760139
> coef(regfit.fwd,12)
              (Intercept)                     hour__              matWeatherDate_
             2.570388e+05               1.321335e+00                -1.600208e+01
                  month__                      day__                    weekday__
             4.879432e+02               1.437390e+01                 1.720657e+02
        Wind_DirectionWSW         Wind_DirectionSW  ConditionsScattered Clouds
            -2.919478e+00               4.589843e+00                 9.982477e+01
   ConditionsMostly Cloudy                  Humidity               Wind_SpeedMPH
             6.323488e+01              -1.562453e+00                -2.546379e-04
           ConditionsMist
             0.000000e+00
>
```

- Started with Temperature alone as a dependent variable.
- Repeated the process for all the dependent variables.
- We kept adding one more variable at a time to check any improvement in the summary results.
- These steps were repeated till we reached our best results.

Example 1:

```
fit <- lm(power_ ~ Temperature , data=matWeatherFinal)

summary(fit)
```

```
Call:
lm(formula = power_ ~ Temperature, data = matWeatherFinal)

Residuals:
   Min     1Q Median     3Q    Max 
-312.4 -185.9 -113.6  180.1 2423.9 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 394.66679    4.11112    96.0   <2e-16 ***
Temperature   0.80414    0.06331    12.7   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 236.8 on 8419 degrees of freedom
Multiple R-squared:  0.0188,	Adjusted R-squared:  0.01869 
F-statistic: 161.3 on 1 and 8419 DF,  p-value: < 2.2e-16
```

Example 2:

```
fit <- lm(power_ ~ Temperature , data=matWeatherFinal)

summary(fit)
```

```
Call:
lm(formula = power_ ~ Dew_PointF, data = matWeatherFinal)

Residuals:
   Min     1Q Median     3Q    Max 
-309.7 -186.3 -115.3  180.4 1587.8 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 414.26357    3.51105 117.989   <2e-16 ***
Dew_PointF    0.55620    0.06254   8.894   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 237.9 on 8419 degrees of freedom
Multiple R-squared:  0.009308,	Adjusted R-squared:  0.009191 
F-statistic:  79.1 on 1 and 8419 DF,  p-value: < 2.2e-16
```

Summary of the most optimum model obtained using forward selection on train data:

```
install.packages("car")
library(car)
samp <- floor(0.75* nrow(matWeatherFinal))
samp
set.seed(123)
training.index <- sample(seq_len(nrow(matWeatherFinal)), size= samp)
train <- matWeatherFinal[training.index, ]
test <- matWeatherFinal[-training.index, ]
h<-matWeatherFinal

na.omit(test)
na.omit(train)
#check sea level and temperature collinearity
lm.fit = lm.fit = lm(power_ ~ Temperature +  month__ +  day__ + hour__ +dayofWeek_
summary(lm.fit)
Regressionoutput <- summary(lm.fit)$coefficients[,1]
write.csv(Regressionoutput, file = "RegressionoutputForward.csv")

vif(lm.fit)
install.packages("forecast")
library(forecast)

16:1  (Top Level)  R Script
```

```
Console E:/Sem4-Fall 2016/ADS/Assignment 2 Materials/
dayofWeek__4          226.6737     6.0962  37.183  < 2e-16 ***
dayofWeek__5          207.2720     6.1146  33.898  < 2e-16 ***
dayofWeek__6          104.5757     6.1320  17.054  < 2e-16 ***
Sea_Level_PressureIn   23.3449     7.4471   3.135 0.001728 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 129.6 on 6445 degrees of freedom
  (52 observations deleted due to missingness)
Multiple R-squared:  0.7106,	Adjusted R-squared:  0.7074 
F-statistic: 219.8 on 72 and 6445 DF,  p-value: < 2.2e-16

> 
```

Regression Output:

```r
training.index <- sample(seq_len(nrow(matWeatherFinal)), size= samp)
train <- matWeatherFinal[training.index, ]
test <- matWeatherFinal[-training.index, ]
h<-matWeatherFinal

na.omit(test)
na.omit(train)
#check sea level and temperature collinearity
lm.fit = lm.fit = lm(power_ ~ Temperature +  month__ +  day__ + hour__ +dayofWeek_
summary(lm.fit)
RegressionOutput <- summary(lm.fit)$coefficients[,1]
write.csv(RegressionOutput, file = "RegressionOutputForward.csv")

vif(lm.fit)
install.packages("forecast")
library(forecast)
pred=predict(lm.fit,test)
PerformanceMetrics <- accuracy(pred,train$power_)

write.csv(t(PerformanceMetrics), file = "PerformanceMetricsForward.csv")
```

16:17 (Top Level) R Script

Console E:/Sem4-Fall 2016/ADS/Assignment 2 Materials/

```
> RegressionOutput <- summary(lm.fit)$coefficients[,1]
> RegressionOutput
    (Intercept)     Temperature        month__2        month__3
   -733.7741813       2.0649232     -26.2435660     -23.8853171
       month__4        month__5        month__6        month__7
    -84.0064189     -98.7122738     -30.1017513      38.7935028
       month__8        month__9       month__10       month__11
     73.5675632     -60.2127855    -103.5471478     -67.1640057
      month__12          day__2          day__3          day__4
    -86.4487545      81.3099195      41.6924738      17.2645892
         day__5          day__6          day__7          day__8
     56.6081316      38.2805195      47.5385017      50.8267705
         day__9         day__10         day__11         day__12
```

Performance Matrix:

```r

na.omit(test)
na.omit(train)
#check sea level and temperature collinearity
lm.fit = lm.fit = lm(power_ ~ Temperature +  month__ +  day__ + hour__
summary(lm.fit)
RegressionOutput <- summary(lm.fit)$coefficients[,1]
write.csv(RegressionOutput, file = "RegressionOutputForward.csv")

vif(lm.fit)
install.packages("forecast")
library(forecast)
pred=predict(lm.fit,test)
PerformanceMetrics <- accuracy(pred,train$power_)

write.csv(t(PerformanceMetrics), file = "PerformanceMetricsForward.csv

lm.fit1 = lm(power_ ~ ., data=train)
summary(lm.fit1)

pred=predict(lm.fit1,test)
```

27:1 (Top Level)

Console E:/Sem4-Fall 2016/ADS/Assignment 2 Materials/

```
Warning in install.packages :
  package ‘forecast’ is in use and will not be installed
> library(forecast)
> pred=predict(lm.fit,test)
> PerformanceMetrics <- accuracy(pred,train$power_)
> t(PerformanceMetrics)
        Test set
ME      45.69453
RMSE  1860.27492
MAE    286.47067
MPE    -21.71344
MAPE    74.42619
> write.csv(t(PerformanceMetrics), file = "PerformanceMetricsForward.csv")
>
```

3. Prediction based on forecast data

- The Sample output is generated from Raw datasets merged, after data cleaning and feature selection (after trying Forward, Backward and Stepwise) techniques.
- The feature selection, led us to split the Sample Output into 75% training data set and 25% testing dataset, after trying a possible combinations of 60-40, 80-20, etc.
- Then after fitting the data into the multiple linear regression, we used package “Forecast” to use its predict() function to predict the data on the clean Forecast data missing on the Power/kwh(which was to be predicted).

The summary and the performance matrix of the final model selected using step wise regression

```
fDataNew ×   Run as 3rd - COMPLETE_PART3_A2_CL... ×   forecastPrediction ×   Run as 4th - COMPLETE_FORECASTIN... * ×
Source on Save                                                           Run   Source

################## CHECKING FIT OF 1 - THE BEST MODEL #########

lm.fit = lm(kWh ~ Temperature + month + day + `Day of Week` + hour,data=train) # removed pe
summary(lm.fit)

################### Getting all coefficients  #########
summary(lm.fit)$coefficients[,]

29:1   Getting all coefficients                                                                R Script
```

```
Console C:/Users/310250154/Desktop/ADS/Assignments/assignment 2/Assignment 2(1)/
4482              Scattered Clouds      240.00000
7251                 Mostly Cloudy      310.00000
4532                 Mostly Cloudy      110.00000
7420                    Light Rain      100.00000
4811                 Mostly Cloudy      100.00000
3467                 Mostly Cloudy      220.00000
7774                 Partly Cloudy      360.00000
5804                      Overcast      240.00000
3383                      Overcast      135.00000
151                       Overcast      350.00000
 [ reached getOption("max.print") -- omitted 5771 rows ]
> lm.fit = lm(kWh ~ Temperature + month + day + `Day of Week` + hour,data=train) # removed peakh
our and weekday #humidity is making a diff of 0.001
> summary(lm.fit)

Call:
lm(formula = kWh ~ Temperature + month + day + `Day of Week` +
    hour, data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-572.04  -67.31   -4.41   57.08  737.76

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)    33.80554   13.59021   2.487 0.012890 *
Temperature     0.06962    0.03822   1.822 0.068548 .
month2        -22.52104    7.92725  -2.841 0.004512 **
month3        -11.71157    7.74514  -1.512 0.130553
month4        -48.37865    7.92839  -6.102 1.11e-09 ***
month5        -36.56030    7.73785  -4.725 2.35e-06 ***
```

```
hour20          126.29005   11.21701  11.259  < 2e-16 ***
hour21           34.66418   11.08578   3.127 0.001774 **
hour22           12.87687   11.14207   1.156 0.247847
hour23           11.40202   11.17354   1.020 0.307554

Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 129.2 on 6446 degrees of freedom
  (52 observations deleted due to missingness)
Multiple R-squared:  0.7141,    Adjusted R-squared:  0.711
F-statistic: 226.8 on 71 and 6446 DF,  p-value: < 2.2e-16

>
```

- with the use of vif() [variance inflation factors] function we have determined the collinearity between dependent variables:

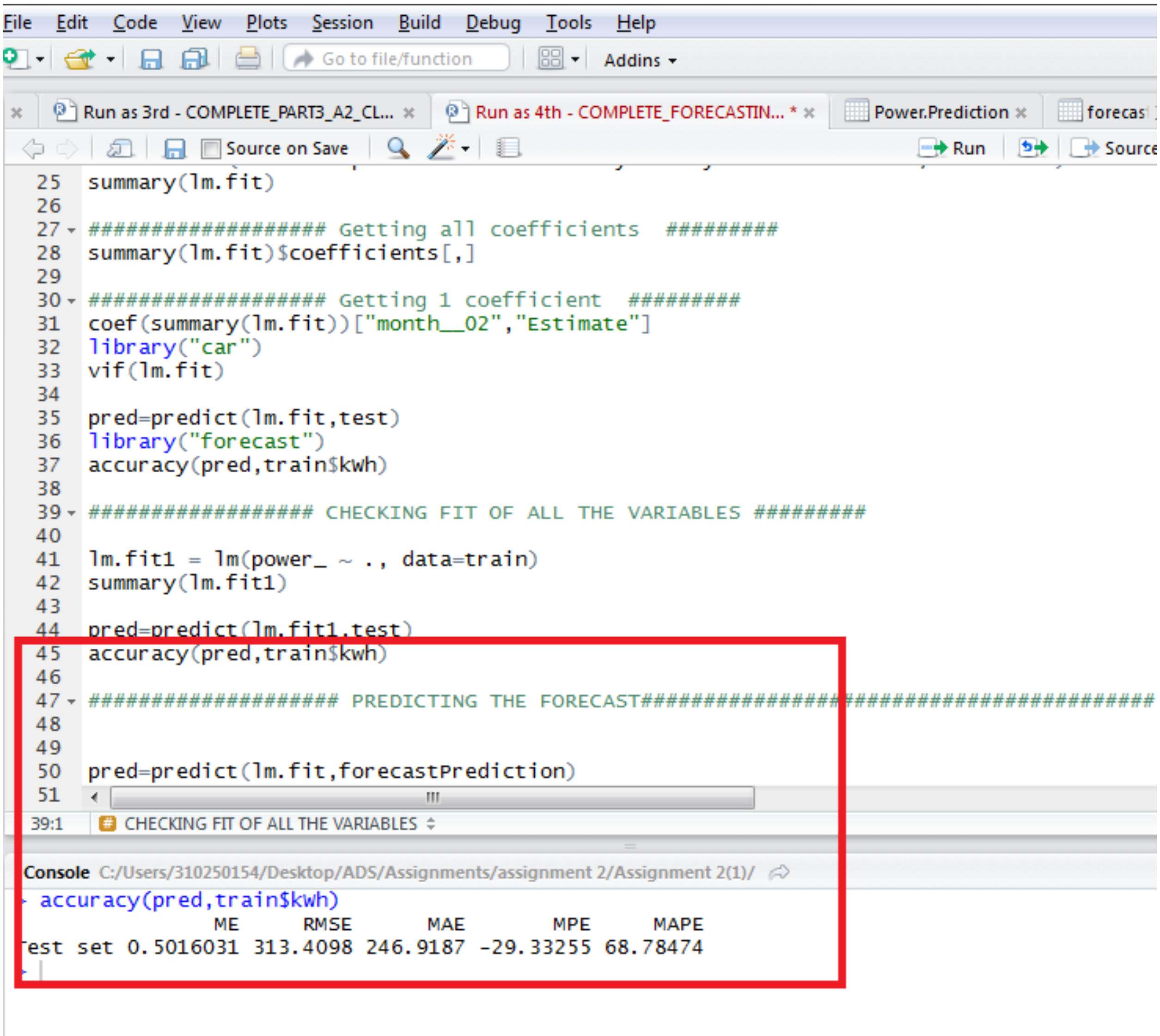
- Values more than 5 denotes collinearity between the variables. In our case all values are between 0-1 hence the dependent variables we have selected are no collinear.

```r
summary(lm.fit)$coefficients[,]

################### Getting 1 coefficient   #########
coef(summary(lm.fit))["month__02","Estimate"]
library("car")
vif(lm.fit)

pred=predict(lm.fit,test)
library("forecast")
accuracy(pred,train$kwh)

################## CHECKING FIT OF ALL THE VARIABLES ########

lm.fit1 = lm(power_ ~ ., data=train)
summary(lm.fit1)

pred=predict(lm.fit1,test)
accuracy(pred,train$kwh)

#################### PREDICTING THE FORECAST################
```

35:1 Getting 1 coefficient

Console C:/Users/310250154/Desktop/ADS/Assignments/assignment 2/Assignment 2(1)/

```
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 129.2 on 6446 degrees of freedom
  (52 observations deleted due to missingness)
Multiple R-squared:  0.7141,    Adjusted R-squared:  0.711
F-statistic: 226.8 on 71 and 6446 DF,  p-value: < 2.2e-16

> vif(lm.fit)
                  GVIF Df GVIF^(1/(2*Df))
Temperature   1.167016  1        1.080285
month         1.210154 11        1.008708
day           1.267092 30        1.003953
`Day of Week` 1.216142  6        1.016441
hour          1.052748 23        1.001118
>
```



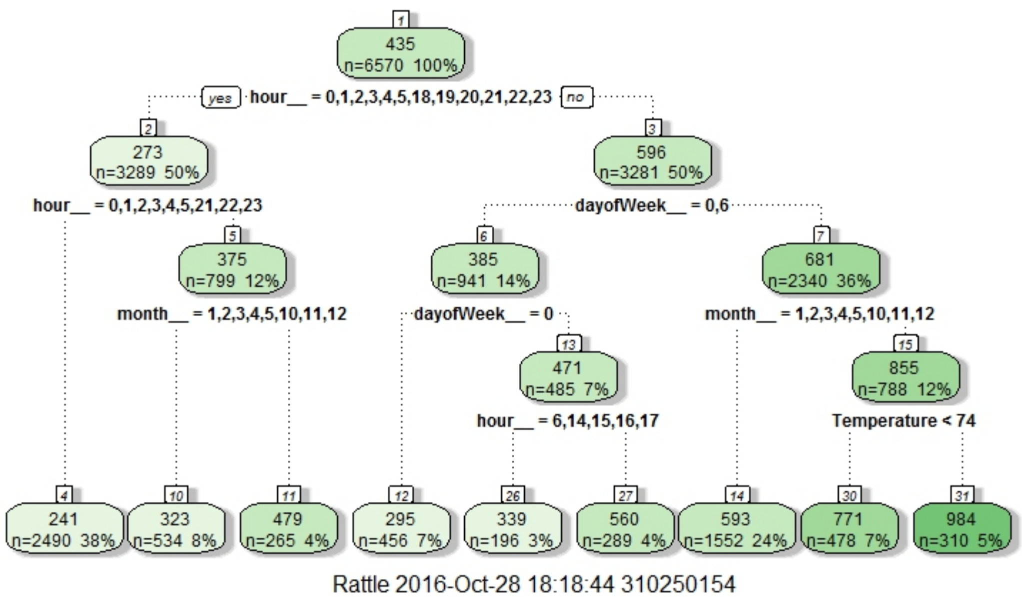
Prune Tree:

Prune Tree is used to avoid overfitting the data. The size of the tree can be selected in order to minimize the cross validated error.

- Prune the tree to the desired size using
  prune(fit, cp=)
- printcp( ) is used to examine the cross-validated error results, select the complexity parameter associated with minimum error, and place it into the **prune( )** function

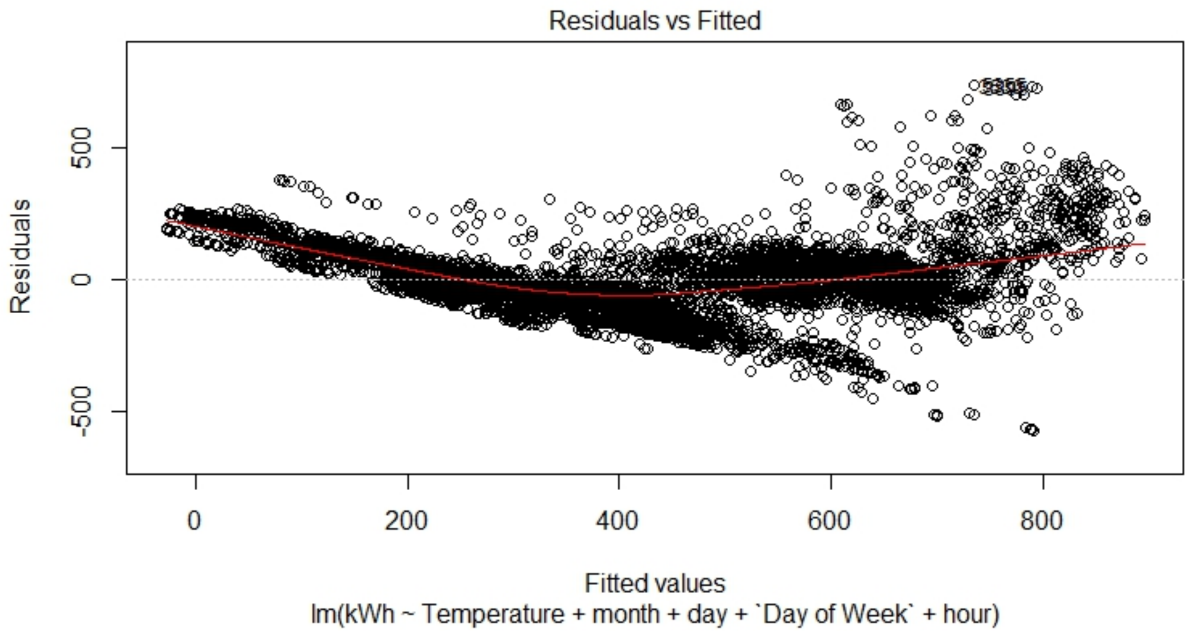
The prune tree results for our model is pasted below:

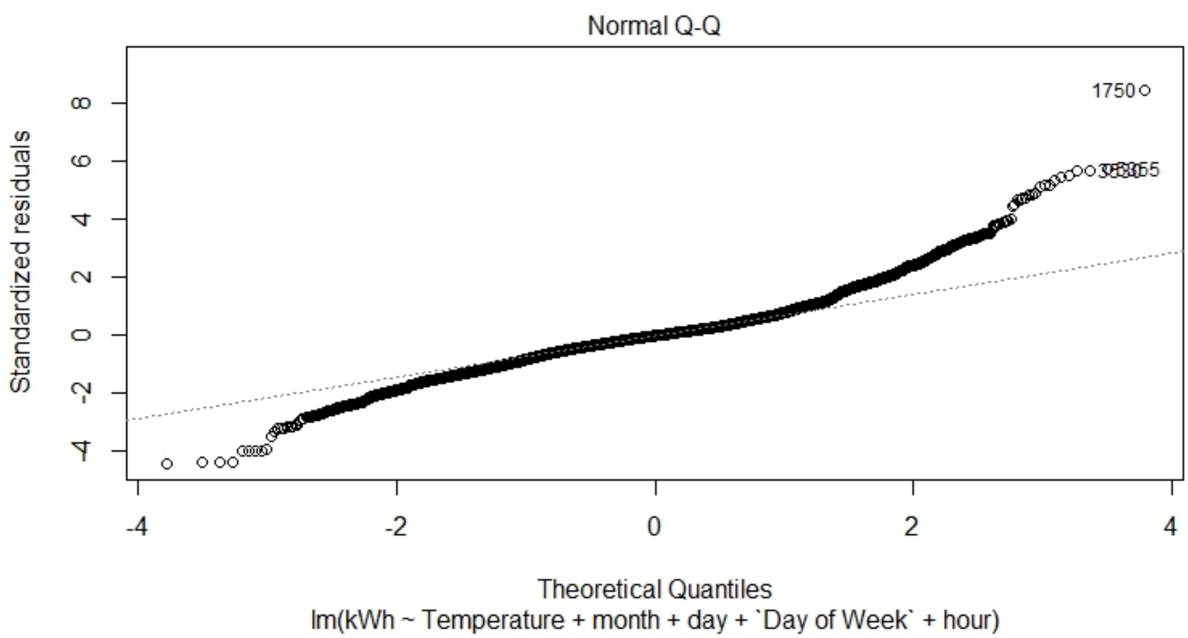
```
install.packages("rpart.plot")
install.packages("rpart")
library(rpart)
library(rattle)


rt <- rpart(power_ ~ Temperature +  month__ +  day__ + hour__ +dayofWeek__ , data=train)

test.pred.rtree <- predict(rt,test)


printcp(rt)
min.xerror <- rt$cptable[which.min(rt$cptable[,"xerror"]),"CP"]
min.xerror
rt.pruned <- prune(rt,cp = min.xerror)
fancyRpartPlot(rt.pruned)

```

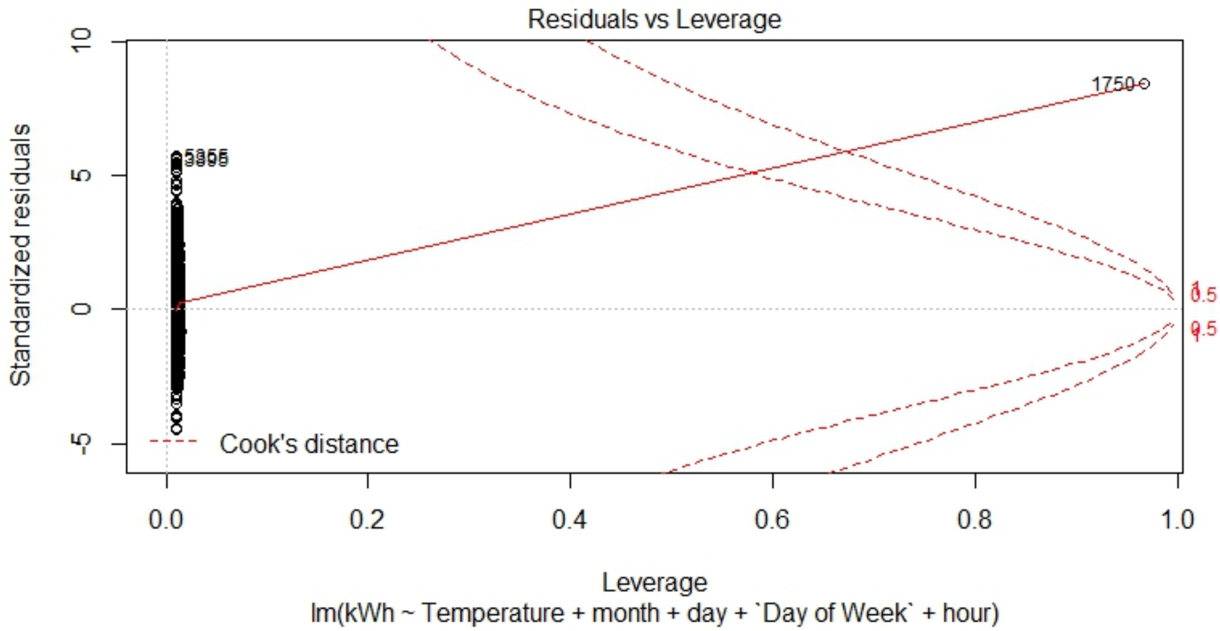


- This tree is giving an overview of decision rules for predicting our outcome.
- We see that hour has the maximum weightage and hence it is the root of the decision/prune tree.
- Subsequent levels explain the weightage in the model respectively.

Plots based on the best fit model:







Residuals vs Leverage
Standardized residuals
10
5
0
-5
1750
Cook's distance
1
0.5
0.5
1
0.0
0.2
0.4
0.6
0.8
1.0
Leverage
lm(kWh ~ Temperature + month + day + `Day of Week` + hour)