**Energy Forecasting using Multilinear Regression**

Team 1 :

Amulya Aankul

Nand Govind Modani

Saksham Agrawal

Introduction

The data we have is of the energy consumption of a school over the span of one year. Our objective is to predict the energy use in the city of Boston using this data along with several variables from the weather data. For prediction of power consumption, we have to use Multi-linear Regression along with several other techniques and evaluate its performance. The flow chart of our project is below:

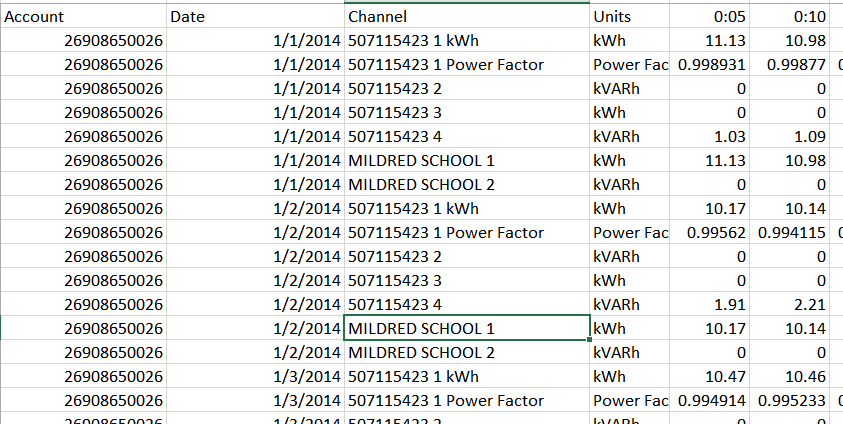


Data wrangling and Data cleaning

The steps involved in data wrangling is summarized below:-

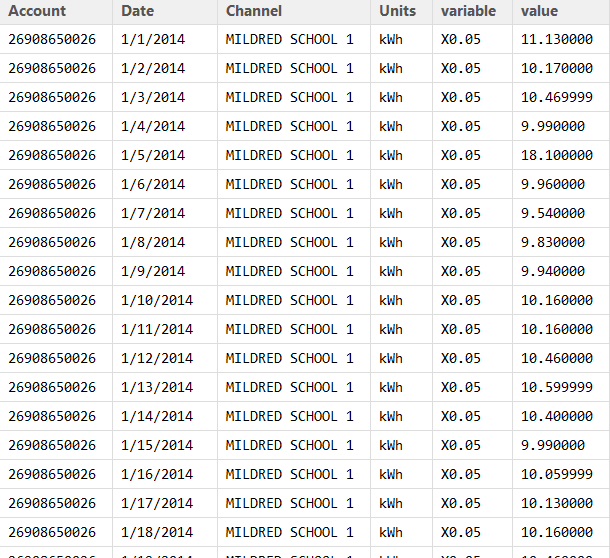
1) Filter the unwanted and repeated rows

* The first step in data wrangling was to identify what rows and columns are actually useful for us in the analysis. We noticed that the power was divided into 3 categories: Power factor, kWh and kVARh. After going through the data and other information regarding the power usage we came to a conclusion that Power factor and kVARh must be ignored as it hardly adds to the total power. Moreover, we don’t get charged for the Reactive power. This leaves us with rows where channel is MILDRED SCHOOL 1. Hence we filter out the rest of the channel values.

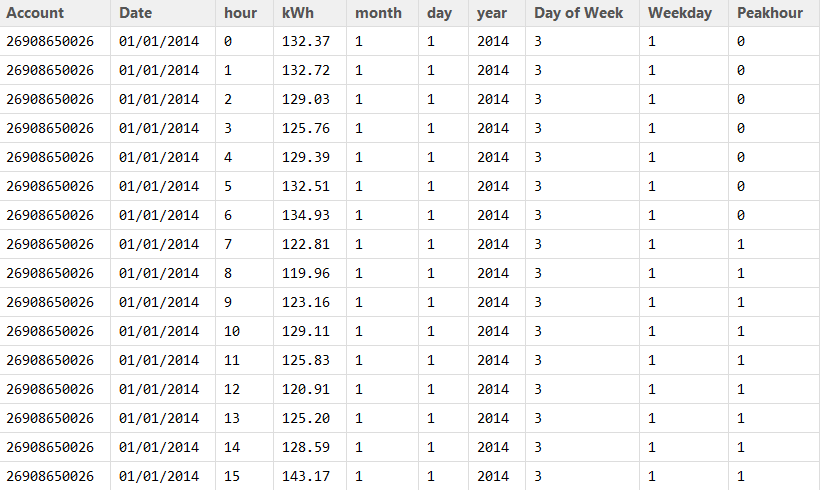


2) Reshaping the structure from wide format to long format

Since we had to perform calculations on the columns, it was convenient to convert the column values into rows. This made aggregation much efficient and easier.

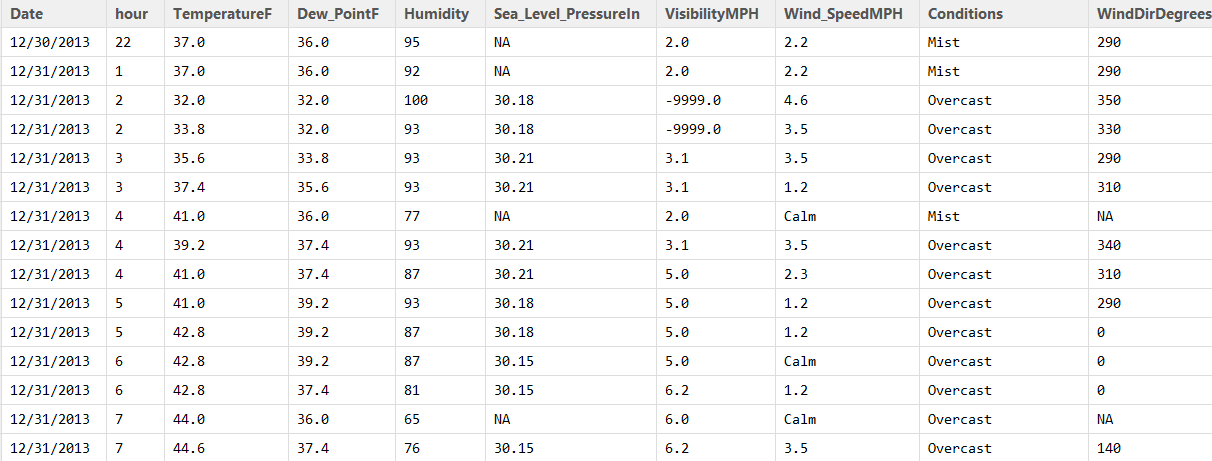


3) Aggregating to hourly values:

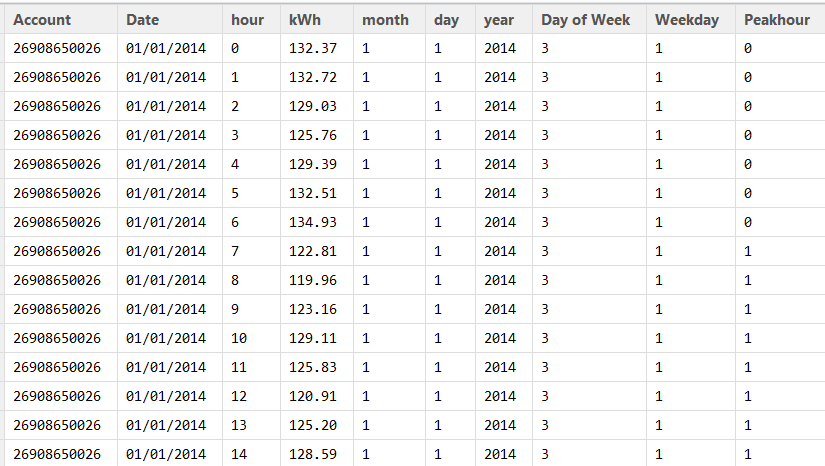


4) Merging the forecast data and aggregated power usage data

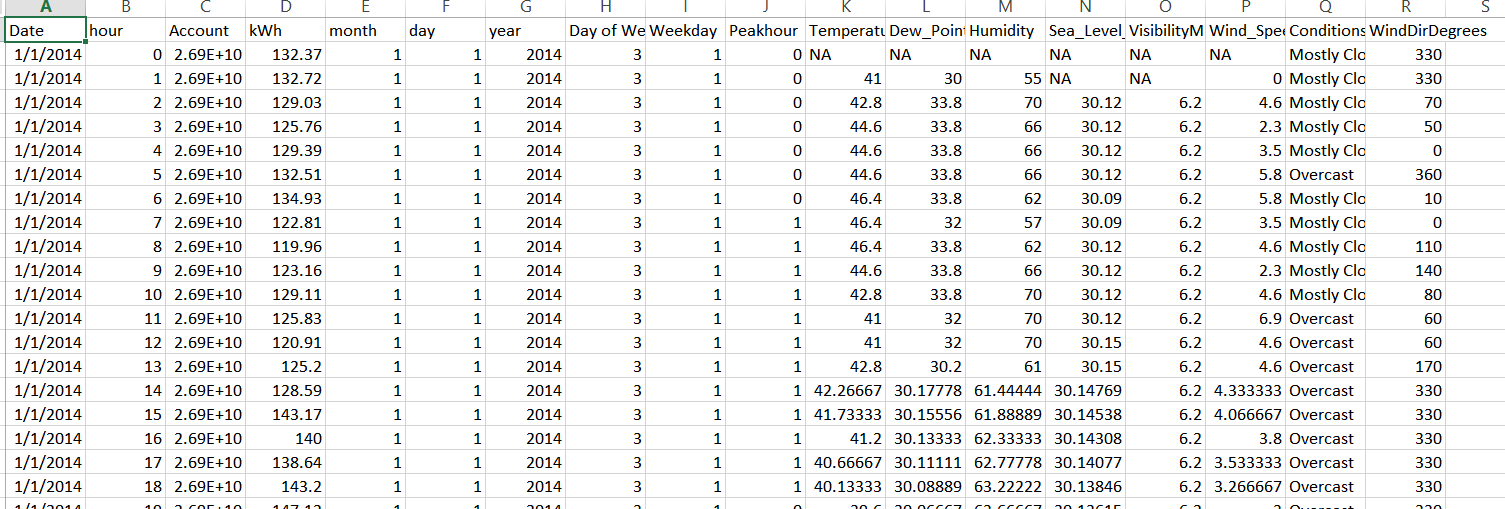
Weather data:



Aggregated Data:

’

Merged Data on Date, Account and hour:



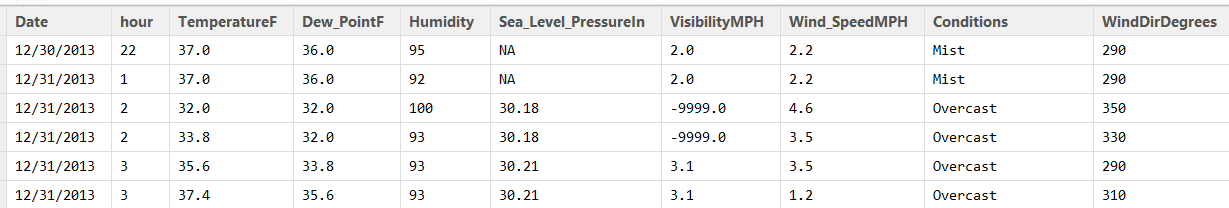
5) Removing the duplicate, outliers and NA values:

We identified the outliers from Boxplot method and removed NA values using the following logic:

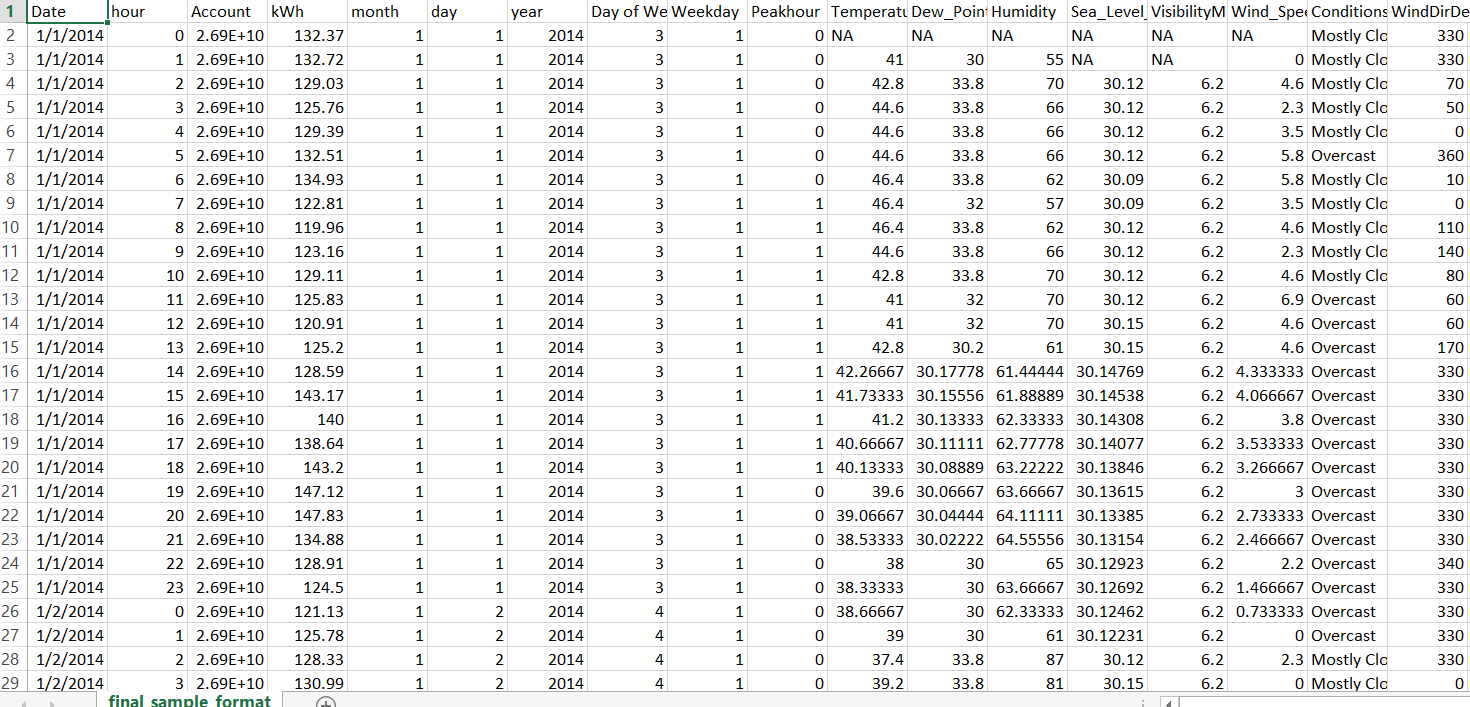
* Missing values in numerical columns like TemperatureF, Dew\_PointF, Humidity, Sea\_Level\_PressureIn, VisibilityMPH and Wind\_SpeedMPH were handled by using prediction by Interpolation method.

* Missing values in Conditions column was handled by using Last observation carried forward (LOCF) method.
* Missing values in WindDirDegrees were simply filled up using random values as the values were random.

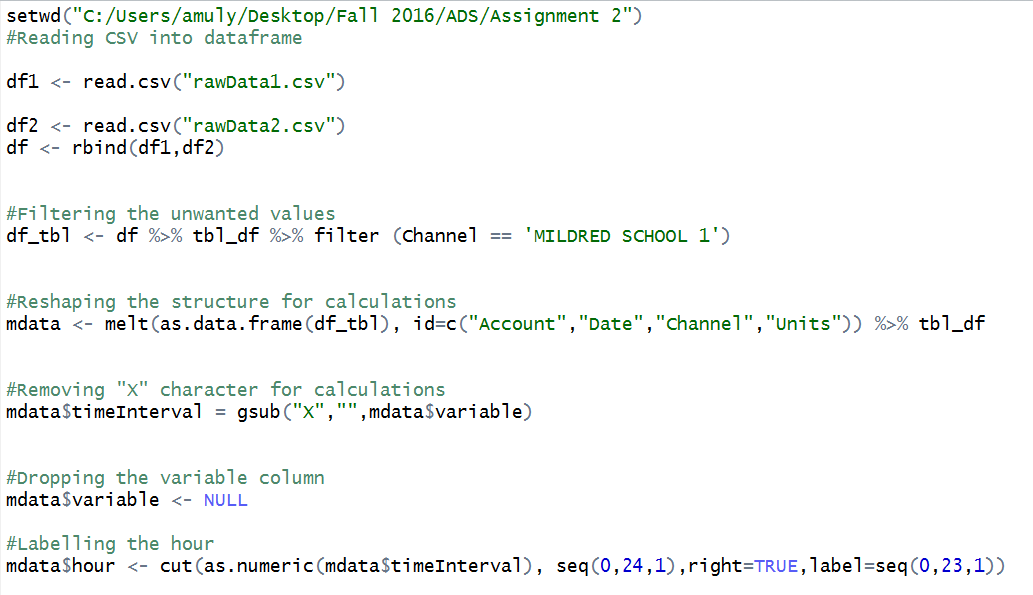
Due to the categorical values in the row, it was difficult to aggregate the duplicate key values, so we simply decided to consider the last value.

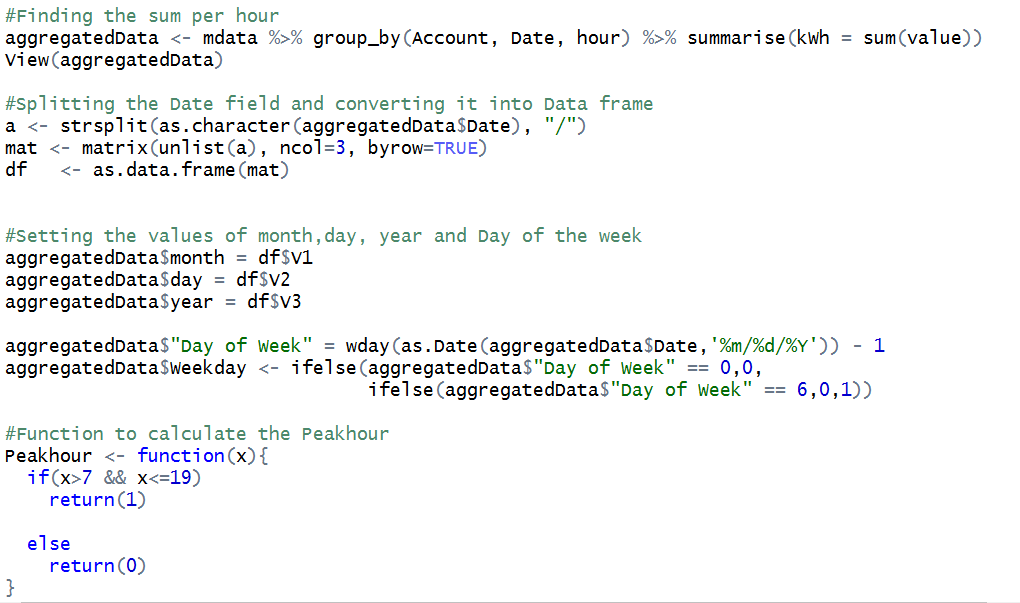


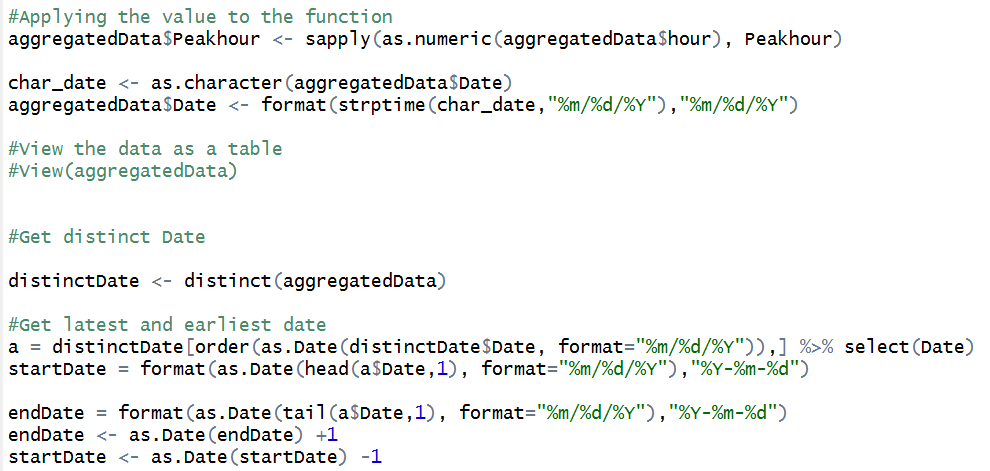
The clean data looks something like this, and is ready for modelling:

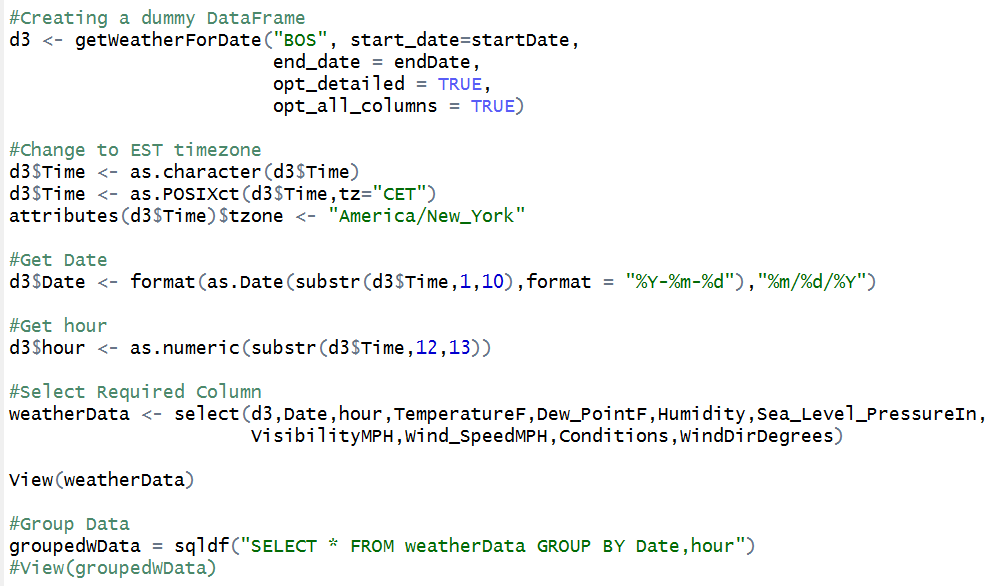


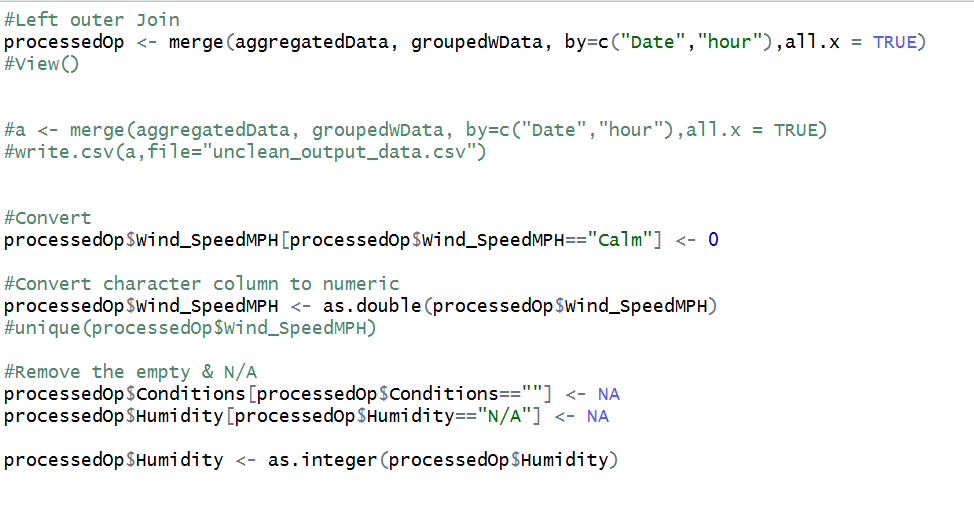
Code:

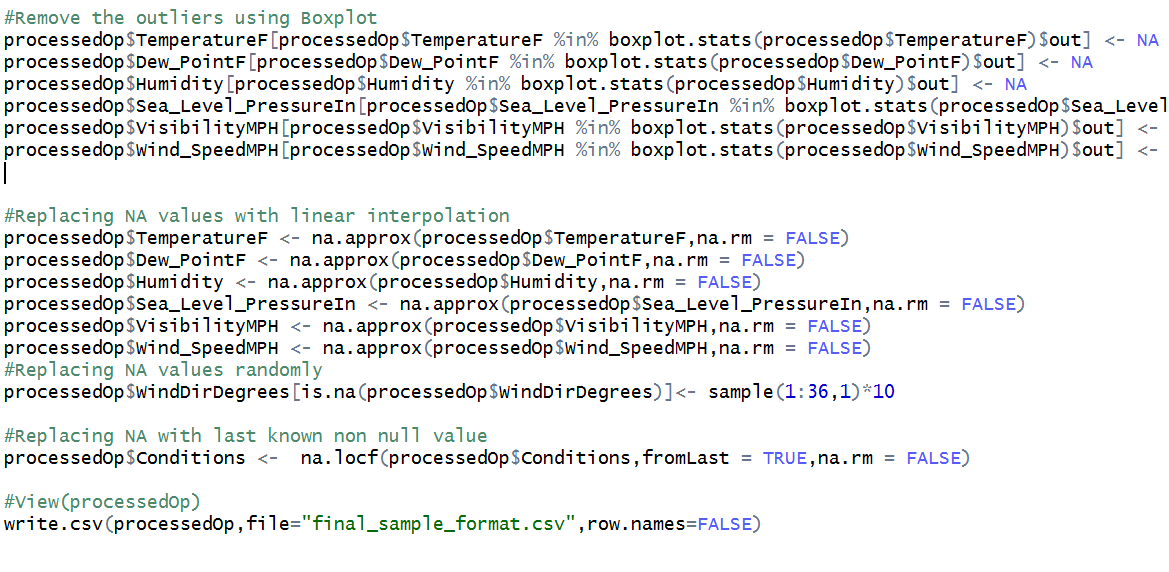












**Multi-Linear Regression**

**WORKFLOW**

Regularization (Ridge, LASSO)

Feature Selection

Co-efficient & RMSE

Model Creation

Feature Transformation

Traditional Techniques

Correlation between variables

Feature Engineering

**Tuning**

|  |  |
| --- | --- |
| **INITIAL FEATURES** | **TYPE** |
| hour | Categorical |
| Account | *Constant* |
| kWh | *Output* |
| month | Categorical |
| day | Categorical |
| year | *Constant* |
| Day.of.Week | Categorical |
| Weekday | Categorical |
| Peakhour | Categorical |
| TemperatureF | Numerical |
| Dew\_PointF | Numerical |
| Humidity | Numerical |
| Sea\_Level\_PressureIn | Numerical |
| VisibilityMPH | Numerical |
| Wind\_SpeedMPH | Numerical |
| Conditions | Categorical |
| WindDirDegrees | Numerical |

**Feature Transformation of Categorical Variables**

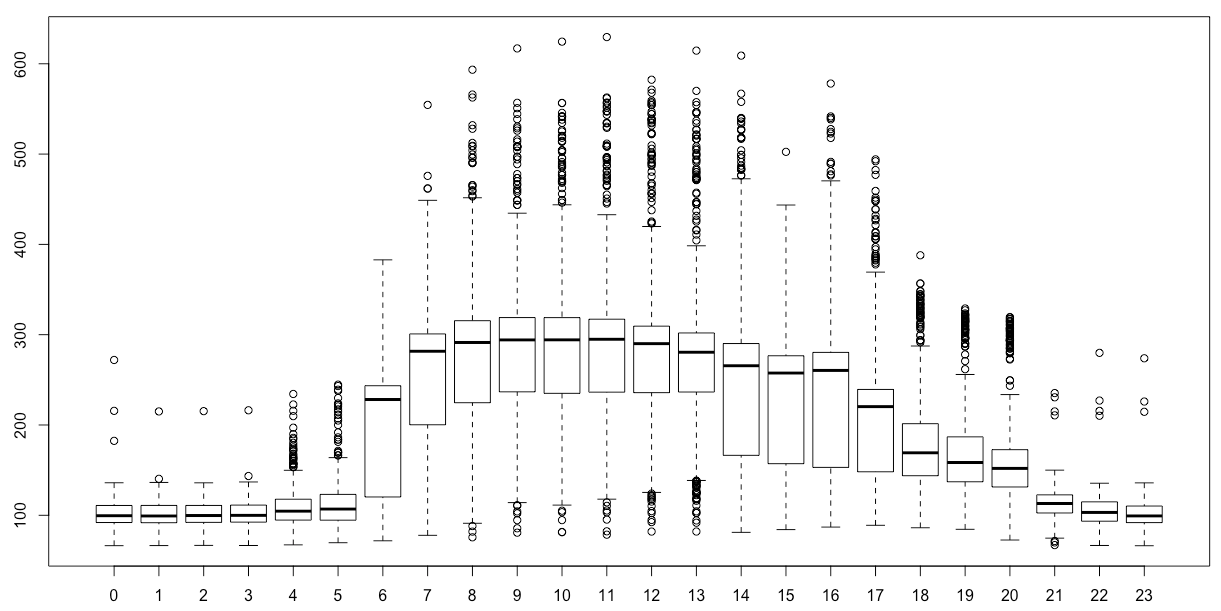
Since we have high number of levels in some categorical variables, so we have to transform them to feed it to a regression model, as it can cause over fitting.

**Hour & Peak hour**

Currently, hour has 24 levels of categories in it. So feeding all of them or even some of them increases the no. of variables. The box plot below shows that power consumption in between 7am to 7pm is distinctly high as compared to that of 7pm to 7am, i.e. power consumption in

Peak Hour can be easily separated from non-peak hour.

So we dropped the “hour” column and using the information from “Peak Hour” column.

****

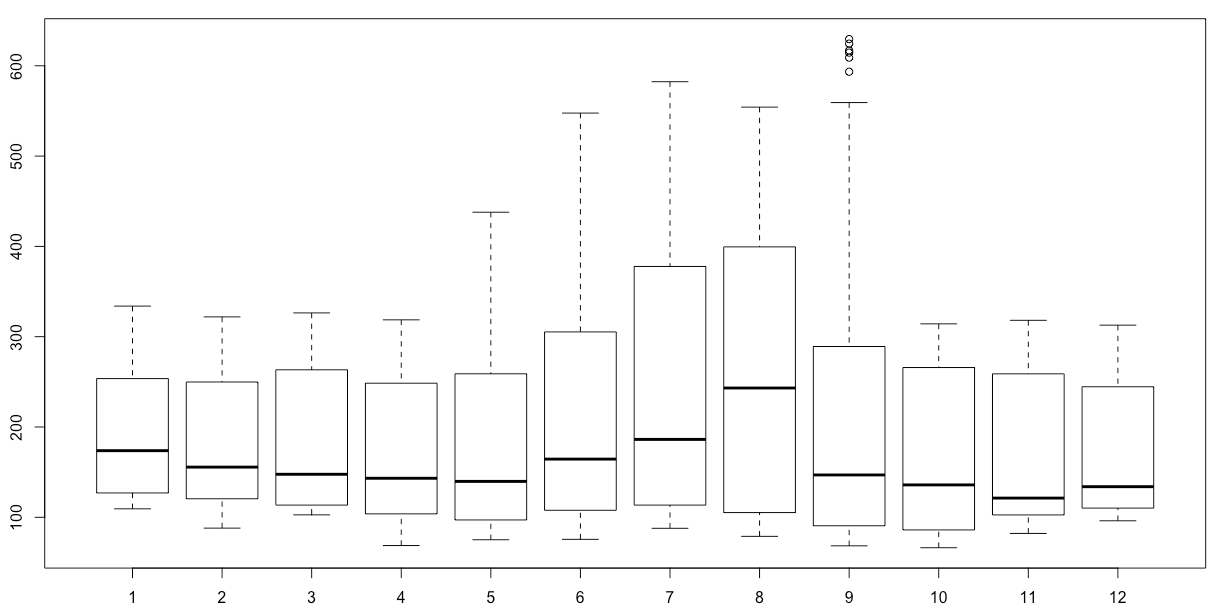
**Month**

“Month” has 12 levels of categories which again are high in itself to feed into any regression model. Keeping in mind that this data is of a “School” and after looking at the pattern of below plot, as similar to that of “hour”, we can reduce the categories of “month” to three, viz.

month 1 – month 4 - Spring

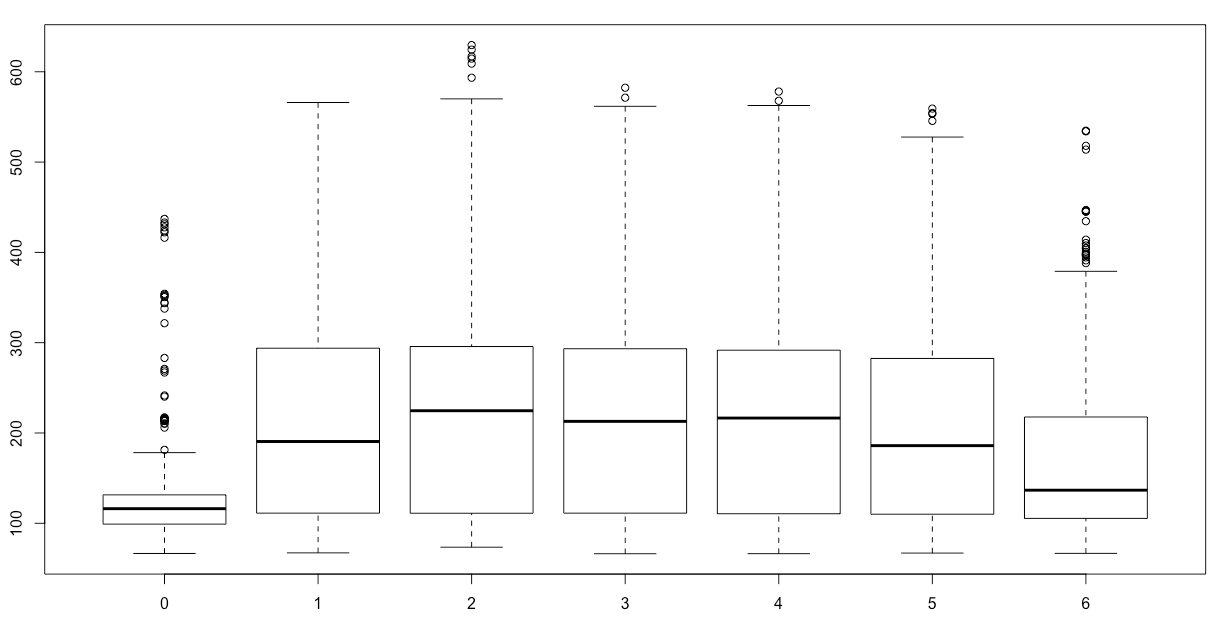
month 5 – month 8 - Summer

month 9 – month 12 - Fall

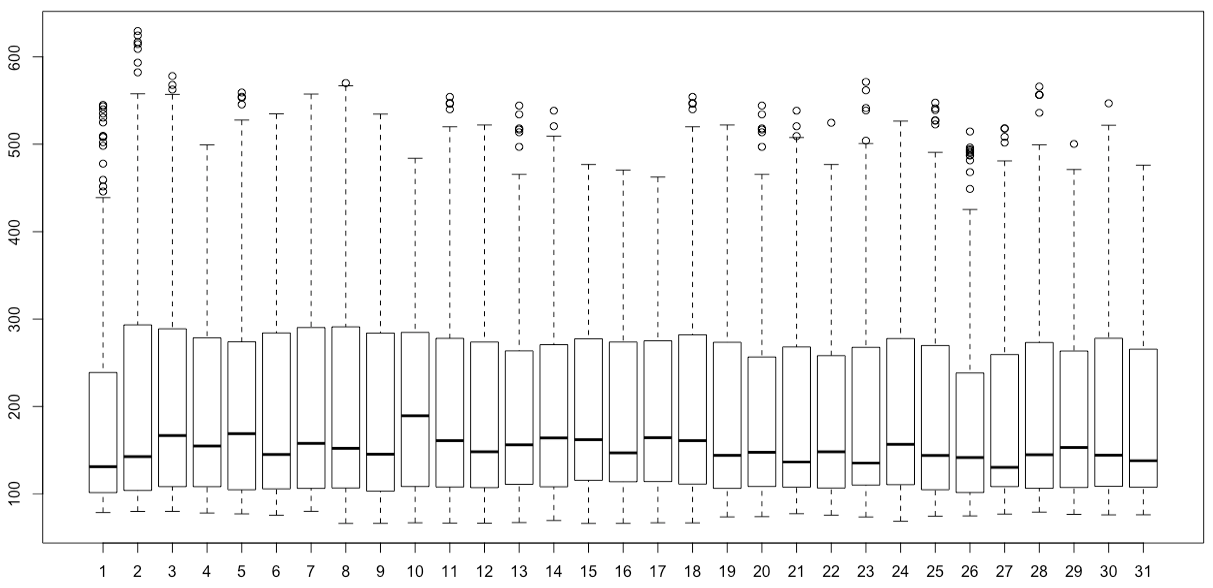


**Day & Day.of.Week & Weekday**

Pattern of “Day.of.Week” vs kWh clearly shows difference in levels for “Day.of.Week 0” and “Day.of.Week 6” i.e. Weekday or not. Hence, it is wise to reduce the levels of “Day.of.Week” from 7 to 2 and directly use “Weekday”.

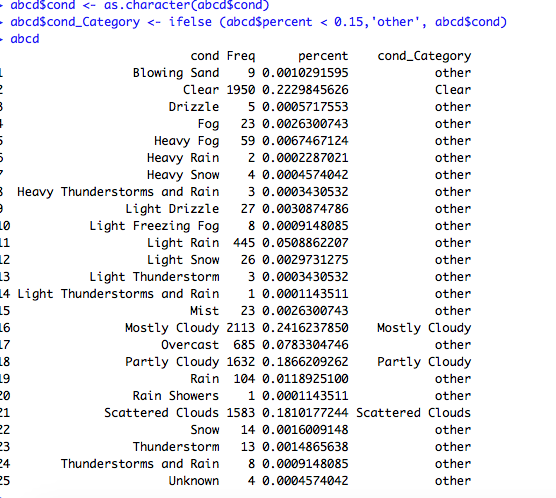


Pattern of “Day” shows high randomness in relation to “kWh”, this is due to the fact that any “day” out of 31 levels can be a “WeekDay” for a “Month”. So instead of introducing 31 levels or even a subset of “day” categories, we can use “WeekDay” as it is more reflective of power consumption.



**Conditions**

Feature engineering for “Conditions” is the trickiest part as there are no fixed levels of categories. Originally, amount of missing data in “Conditions” was around 36% and around 21 categories in “Conditions” are occupying only 10% of rows. So, only 3-5 categories are present that can help in creating a prediction model. To handle this, we decided to take into account only those categories of “Conditons” which occupy more than 15% of rows and renaming all other categories as “Other”. This way we can handle any new “Conditions” value. In below screenshot, we can see the same calculation and its result in the right most column.



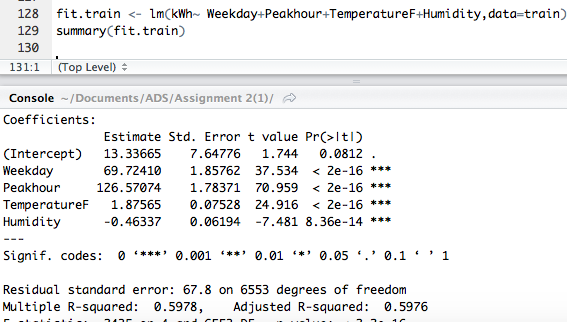
**Feature Transformation of Numerical Variables**

Although, we have only 7 numerical variables but we can reduce them to make a better fitting and generalized predictive model.

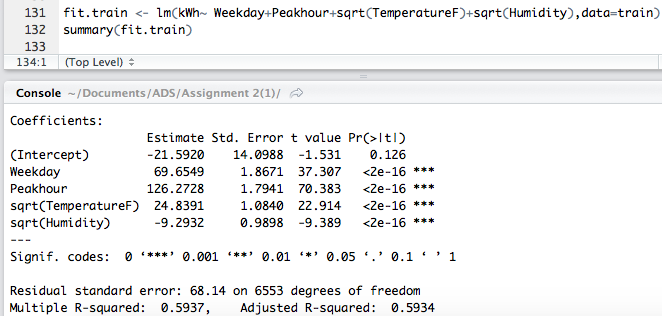
**Temperature & Humidity**

Once we fed, all numerical variables and above transformed categorical variables in Lin. Regression Model, we saw that co-efficient of “TemperatureF” is very low and hence isn’t creating much of an impact. But after seeing the relation in a scatter plot, we transformed “**TemperatureF” to “SQRT(TempeatureF)**” and found a much better co-efficient.

Same Pattern was found in “Humidity”, so we performed the same transformation: **“Humidity” to “SQRT(Humidity)”**

****

*Model Summary* ***Before Transformation*** *in Temperature & Humidity*

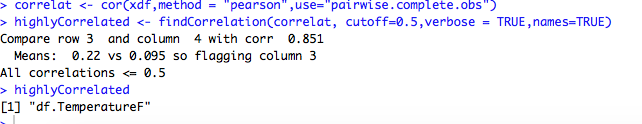


*Model Summary* ***After Transformation*** *in Temperature & Humidity*

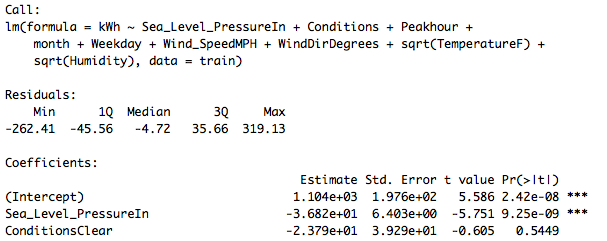
**FEATURE SELECTION**

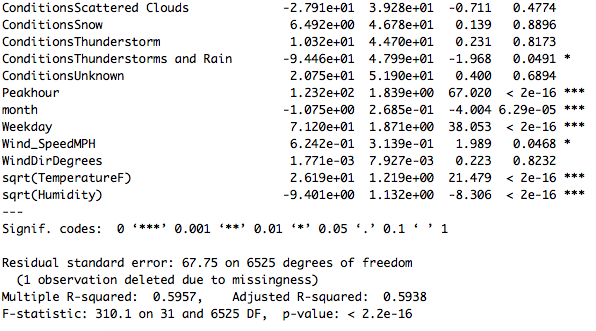
First step, to choose Features is to see if we have highly co-related variables.

After calculating correlation matrix for the features, we found that “TemperatureF” is highly correlated to “DewPointF”. So, this way we ruled out “DewPointF” from the model and selected “TemperatureF”

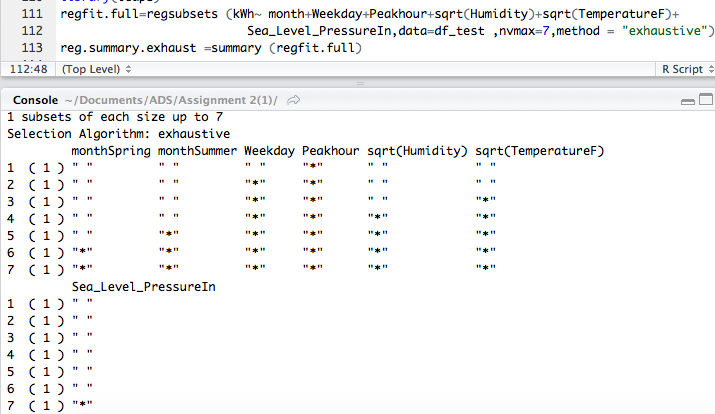


Feeding all variables to see the model by far and try to remove variables with high p-value. So further we can see in below model that “Conditions” , “WinDirDegrees’ and “Wind\_SpeedMPH” can be dropped from the model.

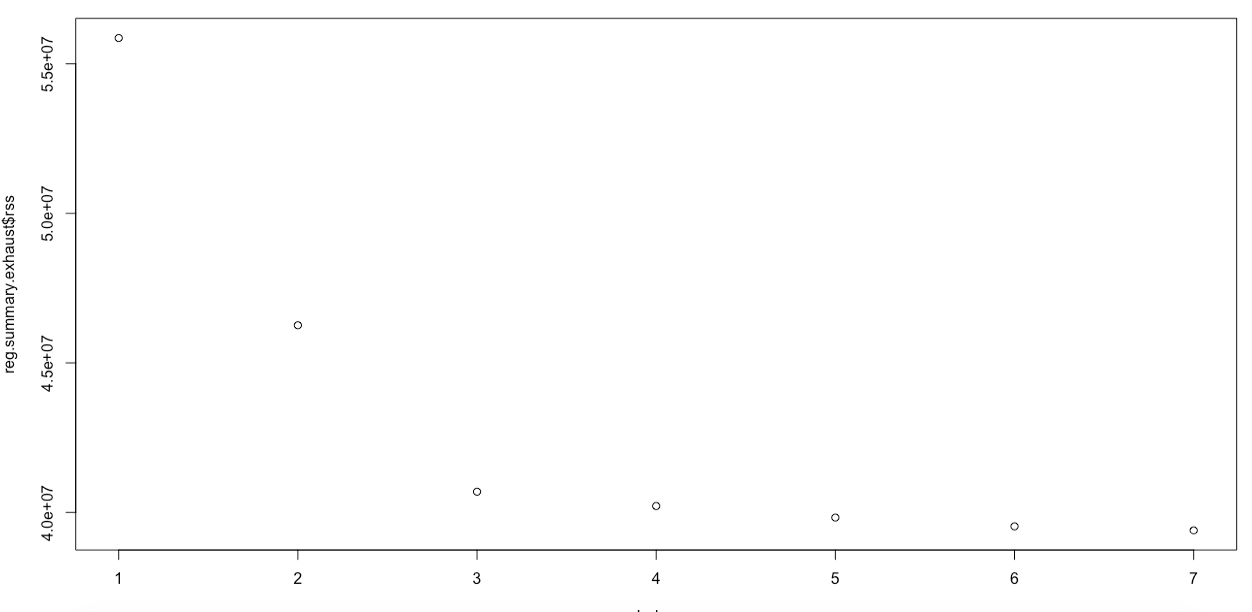




Next Step, to create best implementation of regression, from by far deduced model. We are using **exhaustive feature selection** method to further improve the model. Below diagram shows the selectivity of each variable.



Now let’s look at the RSS plot to figure out the no. of variables majorly impacting the prediction. Below graph tells us that 3 variables are enough in our model to predict the value.



**MODEL**

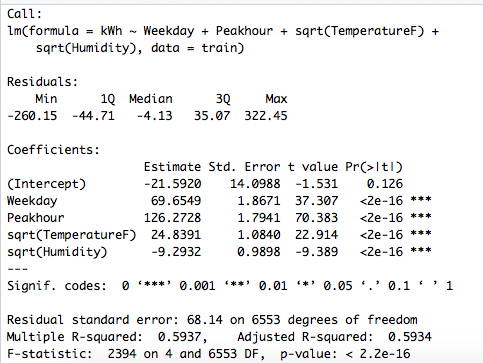
After performing feature engineering, we came to conclusion that 4 variables are enough in our model and the selected model is

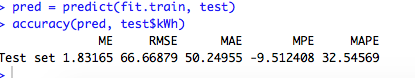
*kWh ~ Weekday + Peakhour+ sqrt(TemperatureF) + sqrt(Humidity)*

Following is the Performance evaluation of our model:

**Adjusted R-sq = 59.34 %**

**RMSE = 66.66879**





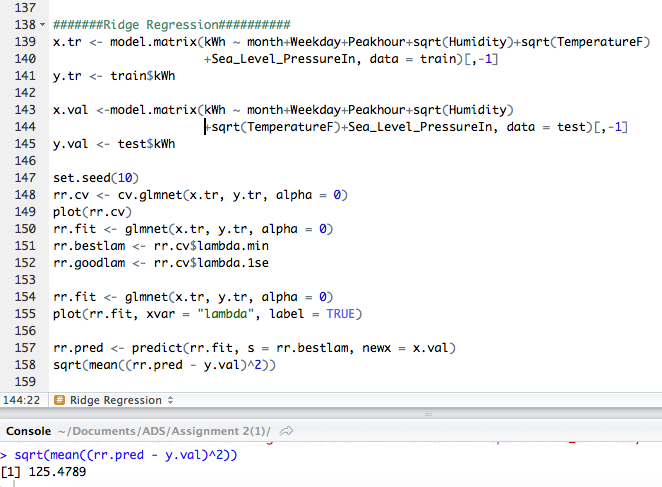
**REGULARIZATION**

We got no improvement in RMSE after applying regularization.

Ridge Regression

We can see that RMSE got deteriorated after applying Ridge Regularization.

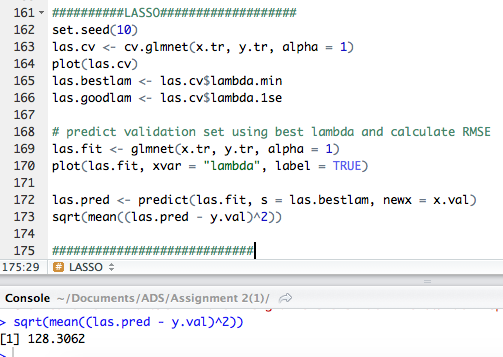
**RMSE = 125.4789**



LASSO Regression

We can see that RMSE got deteriorated after applying LASSO Regularization.

**RMSE = 128.3062**



Forecast

**Step 1 :** Get data from Forcastdata.csv and store it in a data frame

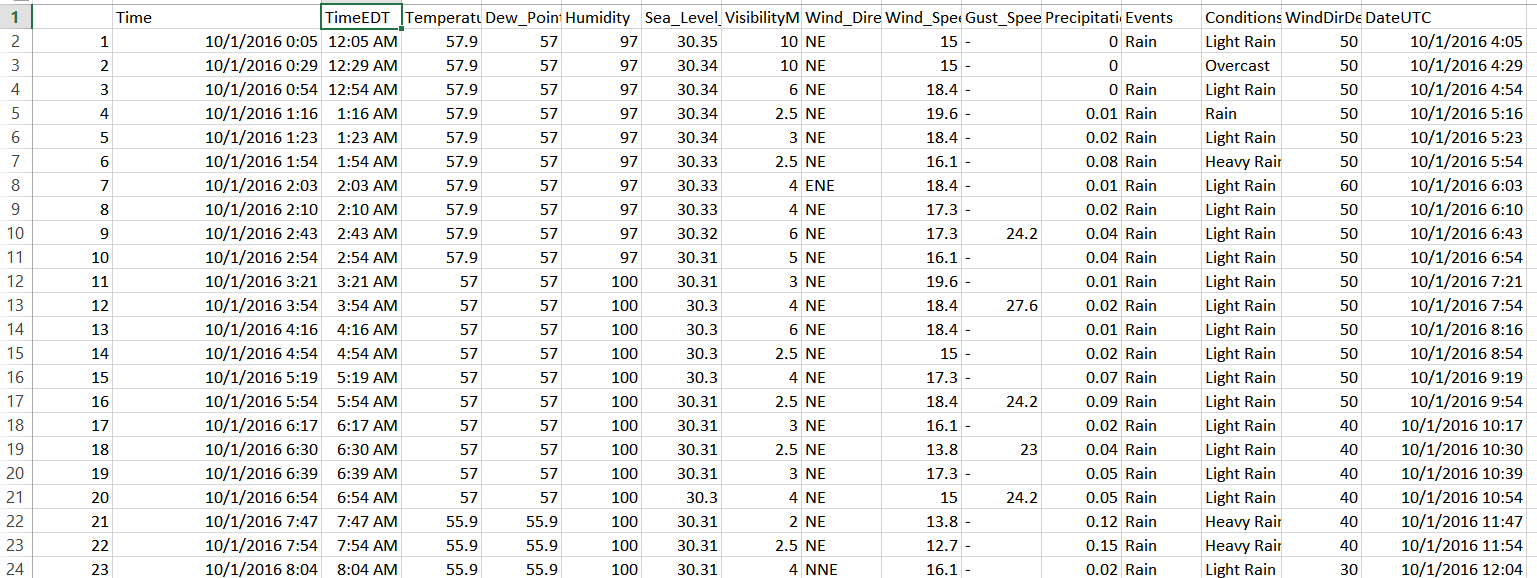
**Step 2 :** Select the variables required for the model from Part 2

**Step 3 :** Get the derived variable as done in part 1

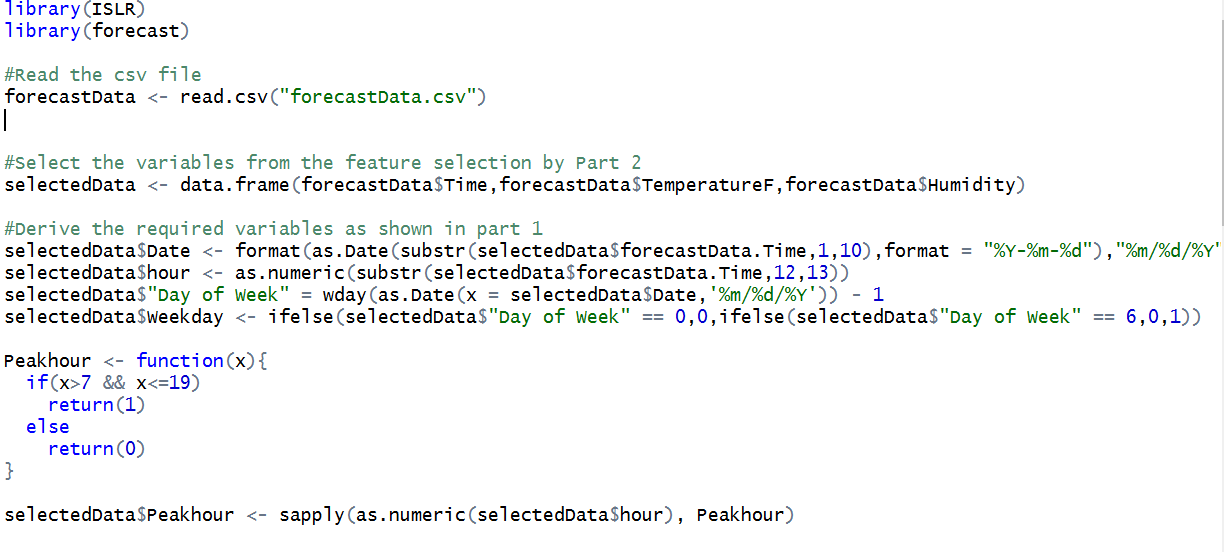
**Step 4 :** Run Linear Regression

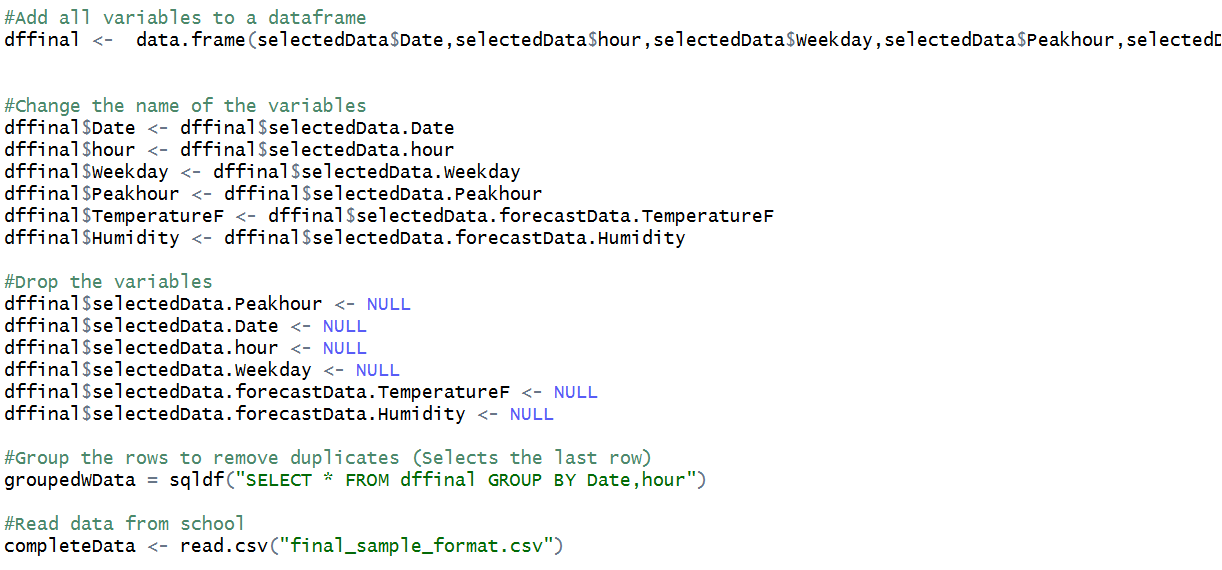
**Step 5 :** Get the predicted value and write it into csv file

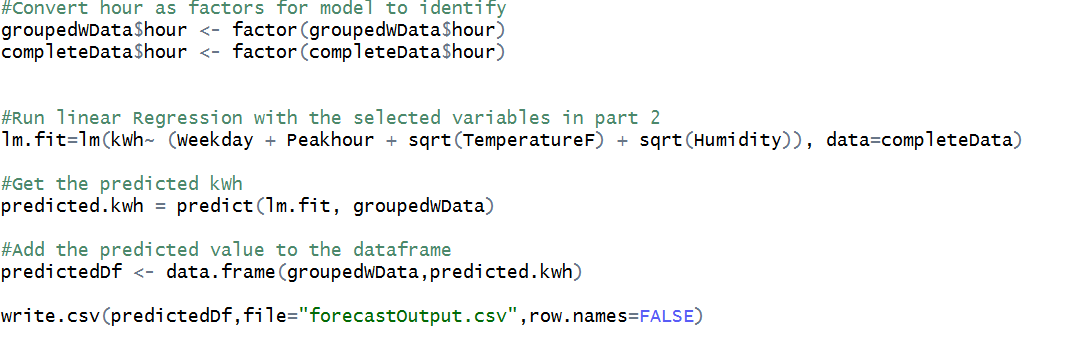
Input file:-



Code:-







Output:-

