Short Term Load Forecasting

S Wajahat Ali

5/1/2020

The primary aim of this project is to forecaste hourly load for a day of a house using load data of the house of past year, recorded at a minute interval. Data used in this project can be downloaded from http://web.lums.edu.pk/~eig/precon.html (http://web.lums.edu.pk/~eig/precon.html).House 26 has been chosen for forecasting.

First, raw data will be cleaned and made ready for analysis. Then trends and other useful information will be extracted through data visulaization. Lastly, different forecating techniques will be used to predict the future values.

Following is the list of all the packages used in this project.

Preprocessing

```
## Read data
House26 = read.csv("House26.csv")
## Covert Date_Time column to posixct
House26$Date_Time = as.POSIXct(House26$Date_Time)
## Summary of the data
summary(House26)
```

```
##    Date_Time                     Usage_kW
##  Min.   :2018-06-01 00:00:00   Min.   :0.0005
##  1st Qu.:2018-08-31 05:59:45   1st Qu.:0.4851
##  Median :2018-11-30 11:59:30   Median :0.7576
##  Mean   :2018-11-30 11:59:30   Mean   :1.0076
##  3rd Qu.:2019-03-01 17:59:15   3rd Qu.:1.1946
##  Max.   :2019-05-31 23:59:00   Max.   :5.8536
```

The first column is the Date_Time, which shows that the data is a time series. The second column is Usage_kW, which represents the total electricity consumption of the household. The values are in kW, which is a unit of power. On average, 1008 W of electricity is consumed in the household.

```
## structure of the dataset
str(House26)
```

```
## 'data.frame':    525600 obs. of  2 variables:
##  $ Date_Time: POSIXct, format: "2018-06-01 00:00:00" "2018-06-01 00:01:00" ...
##  $ Usage_kW : num  3.84 3.87 4.02 4.02 4.01 ...
```

The dataset is of year 2019 which was not a leap year, so in 365 days there should be 525600 minutes. The dataset contains 525600 rows as expected. However, we need to look for any duplicate values or missing values in the dataset. We have to make sure that each time instant has a single row and no time instant is missing. To check this we will create a time series of our own for which we are sure that all rows are correct and then compare it to the dataset to check the integrity of the dataset.

```
## This dataframe contains all minutes between the specified time
All_minutes = data.frame(Date_Time = seq.POSIXt(
                      from = as.POSIXct("2018-06-01 00:00:00 PKT"),
                      to = as.POSIXct("2019-05-31 23:59:00 PKT"),
                      by = "min"))

## Merge the two dataframes.
House26 = merge(House26, All_minutes, by = "Date_Time", all.y = TRUE)
summary(House26)
```

```
##    Date_Time                     Usage_kW
##  Min.   :2018-06-01 00:00:00   Min.   :0.0005
##  1st Qu.:2018-08-31 05:59:45   1st Qu.:0.4851
##  Median :2018-11-30 11:59:30   Median :0.7576
##  Mean   :2018-11-30 11:59:30   Mean   :1.0076
##  3rd Qu.:2019-03-01 17:59:15   3rd Qu.:1.1946
##  Max.   :2019-05-31 23:59:00   Max.   :5.8536
```

The summary above shows that no rows have NA in Usage_kW column, which proves that the dataset has all Date and Time instances in the specified time.

The next thing we need to do is to convert the data from minute-interval to hour-interval. The tapply() function has been used to change the granularity of the data. All the minute interval samples in an hour are replaced by a single value, mean of the sixty values.

```
## Using the tapply() function to create hourly data and replace it in House26 dataframe
House26 = data.frame( Date_Time = seq.POSIXt(
                          from = as.POSIXct("2018-06-01 PKT"),
                          to = as.POSIXct("2019-05-31 23:00:00 PKT"),
                          by = "hour"),
                          Usage_kW = c(tapply(House26$Usage_kW,
                          (row(as.matrix(House26$Usage_kW))-1)%/%60, mean)))

summary(House26)
```

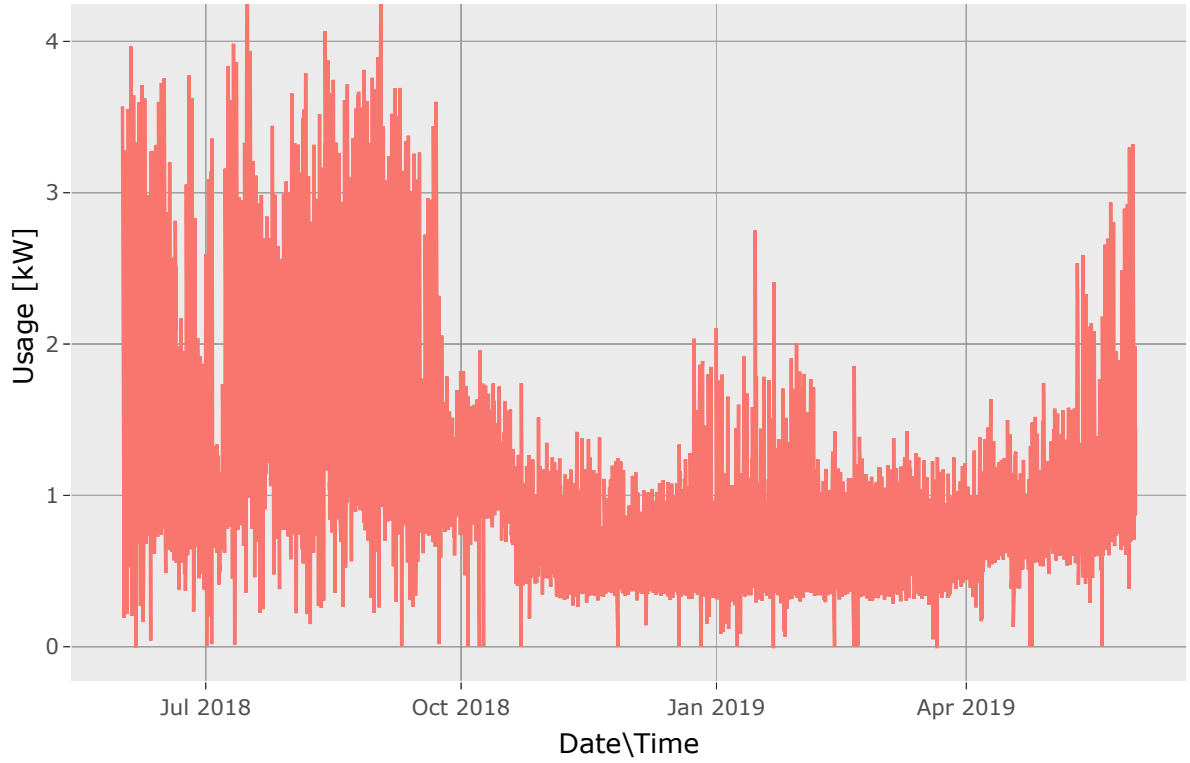
```
##    Date_Time                     Usage_kW
##  Min.   :2018-06-01 00:00:00   Min.   :0.0005
##  1st Qu.:2018-08-31 05:45:00   1st Qu.:0.5133
##  Median :2018-11-30 11:30:00   Median :0.7940
##  Mean   :2018-11-30 11:30:00   Mean   :1.0076
##  3rd Qu.:2019-03-01 17:15:00   3rd Qu.:1.1825
##  Max.   :2019-05-31 23:00:00   Max.   :4.5039
```

The data has been cleaned and is now ready for analysis.

Visualizing the Data

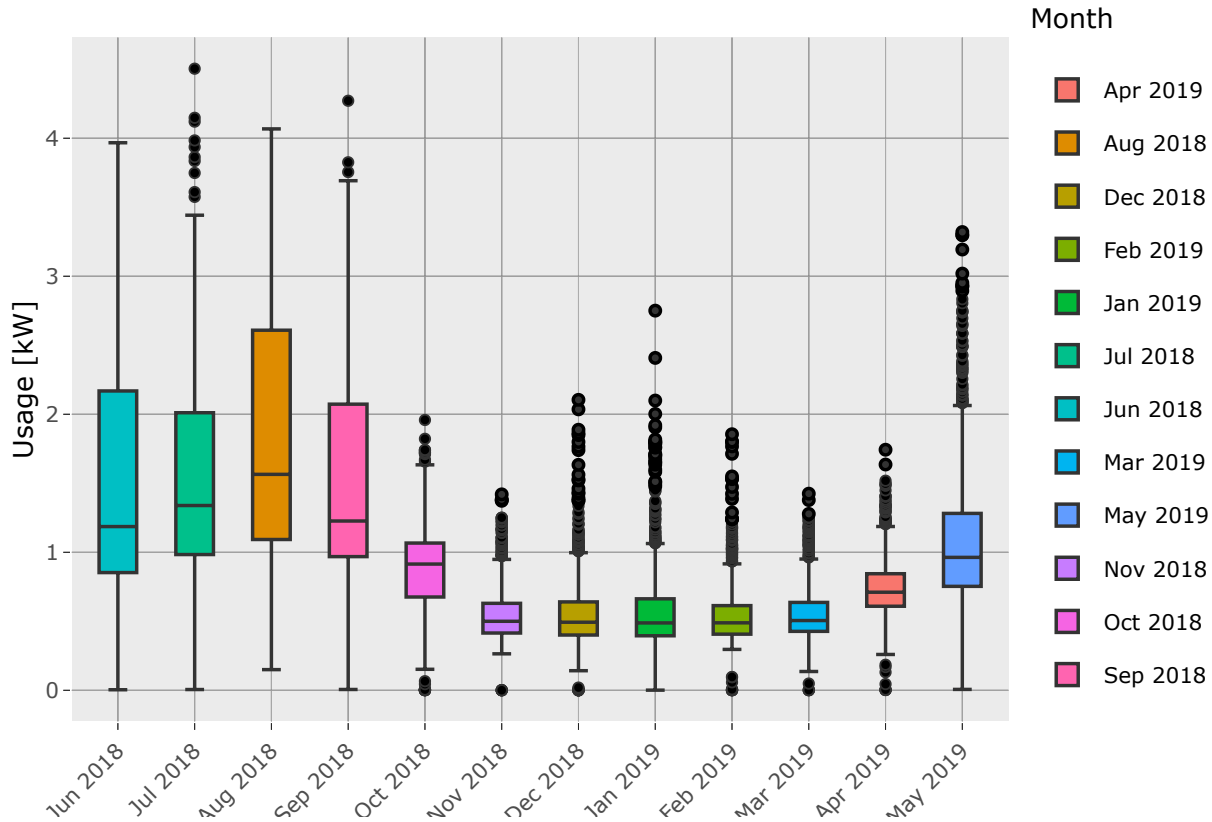
The plot below shows that electricity consumption is high for summer months and low for winter months. This means electricity consumption is highly dependent on the weather.

```
ggplotly(
 ggplot(House26, aes(x = Date_Time))+
 geom_line(aes(y = Usage_kW, color = "House 26"))+
               labs(x = "Date\\Time", y = "Usage [kW]")
 )
```



The following boxplot shows the monthly variation in the electricity consumption.

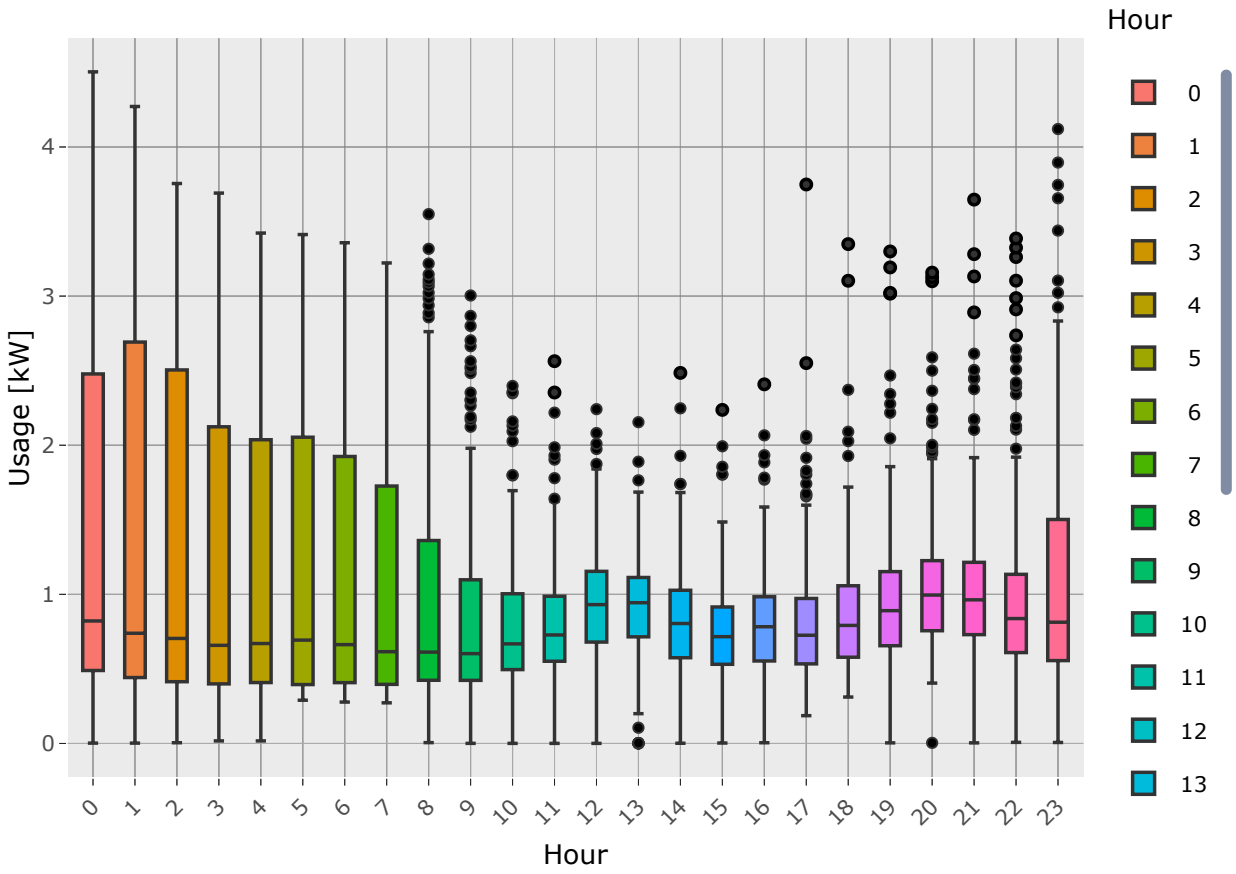
```
House26$Month = paste(months(House26$Date_Time, abbreviate = TRUE) ,
                      as.POSIXlt(House26$Date_Time)$year+1900)

ggplotly(
ggplot(House26)+
  geom_boxplot(aes(x = Month, y = Usage_kW, fill = Month))+
  labs(x="Month", y = "Usage [kW]")+
  theme(axis.text.x = element_text(angle = 45))+
  scale_x_discrete(limits= c(paste(month.abb[6:12], "2018"),
                             paste(month.abb[1:5], "2019")))
)
```



Month

The boxplot below shows the hourly variation in the electricity consumption. So, apart from seasonal variation electricity consumption also depends on the hour of a day, it is lower between 9 AM to 9 PM compared to 9 PM to 9 AM.

```
House26$Hour = as.character.Date(as.numeric(format(House26$Date_Time,'%H')))
ggplotly(
ggplot(House26)+
  geom_boxplot(aes(x = Hour, y = Usage_kW , fill = Hour))+
  labs(x="Hour", y = "Usage [kW]")+
  theme(axis.text.x = element_text(angle = 45)))
```



Removing the unwanted colums and renaming the rest.

```
House26 = House26[,c(1,2)]
colnames(House26) = c("Time", "hourly_load")
```

This code creates extra rows for 1st June, 2019, the day we want to forecast for

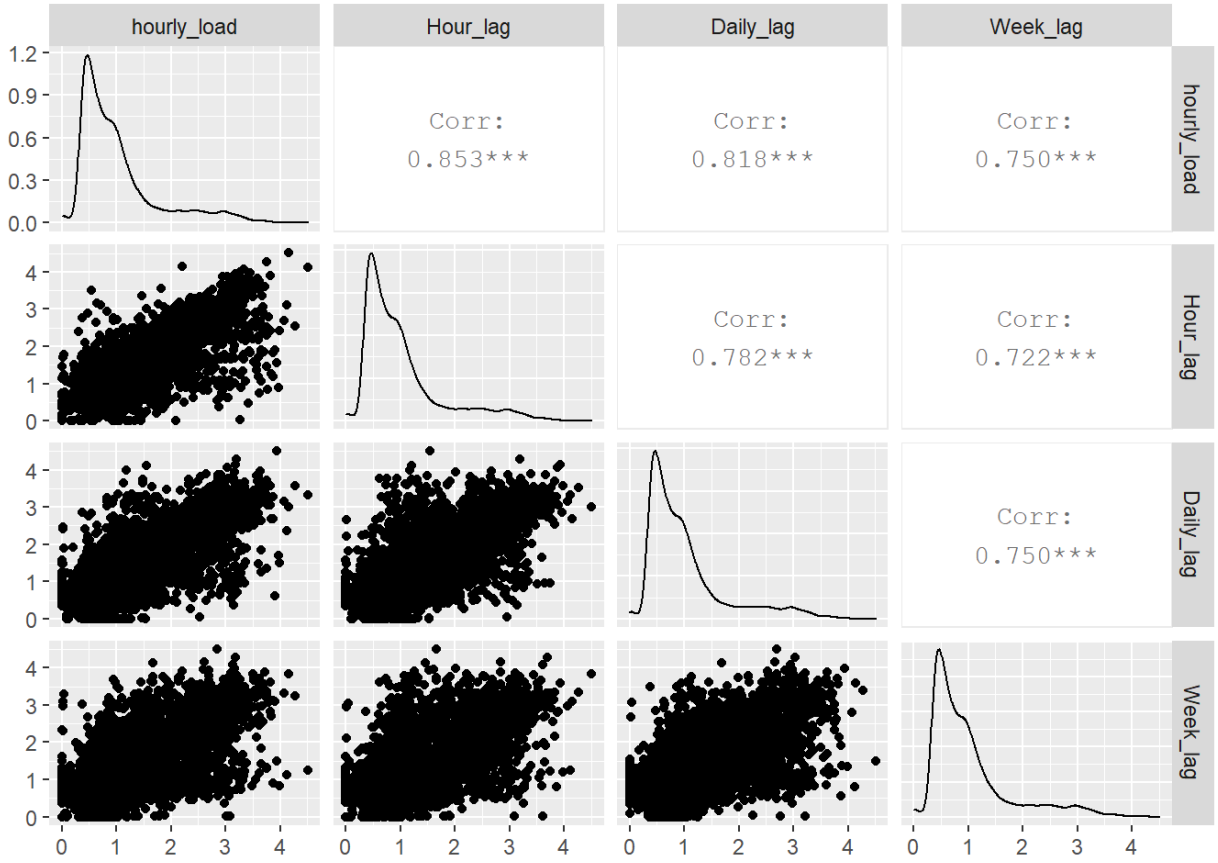
```
House26 <- rbind(House26,data.frame(Time = rep(NA, 24),
                                    hourly_load =rep(NA,24)))

House26$Time <- seq.POSIXt(from = as.POSIXct("2018-06-01 00:00:00 PKT"),
                           to = as.POSIXct("2019-06-01 23:00:00 PKT"),
                           by = "hour")
```

Creating Hourly, Daily and Weekly lags

```
House26$Hour_lag = c(rep(NA,1),House26$hourly_load[1:(nrow(House26)-1)])
House26$Daily_lag = c(rep(NA,24),House26$hourly_load[1:(nrow(House26)-24)])
House26$Week_lag = c(rep(NA,168),House26$hourly_load[1:(nrow(House26)-168)])
```

There is a high correlation between hourly_load and the rest of the independent variables. Hour_lag has the highest correlation of 0.853 with the hourly_load.

```
ggpairs(House26[,-1])
```



Hour_lag has the highest correlation with hourly_load, but we do not have Hour_lag values for the day we want to forecast load for. If we have to use Hour_lag, we will need to forecast for each hour separately and then use that result in forecasting the next hour. This chain forecasting increases the error in our final hour forecast results. So the next in line is Daily_lag, which is the most feasible to be used for STLF in our case.

```
House26_table = House26[(nrow(House26)-23): nrow(House26),]
 House26_table$Time <- strftime(House26_table$Time, format="%Y-%m-%d %H:%M:%S")
 House26_table
```

```
##                     Time hourly_load Hour_lag Daily_lag  Week_lag
## 11000 2019-06-01 00:00:00          NA 1.448917 3.3203750 0.9471083
## 21000 2019-06-01 01:00:00          NA       NA 2.9522400 1.9504000
## 31000 2019-06-01 02:00:00          NA       NA 2.9286050 1.6981917
## 41000 2019-06-01 03:00:00          NA       NA 3.1935667 1.3751250
## 51000 2019-06-01 04:00:00          NA       NA 2.1622133 1.5763167
## 61000 2019-06-01 05:00:00          NA       NA 2.2565900 1.3941433
## 71000 2019-06-01 06:00:00          NA       NA 2.4237983 1.3442233
## 8760  2019-06-01 07:00:00          NA       NA 2.5839683 1.3325433
## 9100  2019-06-01 08:00:00          NA       NA 2.4302150 0.9941283
## 10100 2019-06-01 09:00:00          NA       NA 1.5568500 0.9591483
## 11100 2019-06-01 10:00:00          NA       NA 1.1962567 1.1954950
## 12100 2019-06-01 11:00:00          NA       NA 0.7086150 0.8574933
## 13100 2019-06-01 12:00:00          NA       NA 0.8759450 0.8451950
## 14100 2019-06-01 13:00:00          NA       NA 0.9513400 0.8534717
## 15100 2019-06-01 14:00:00          NA       NA 0.9959883 1.3637300
## 16100 2019-06-01 15:00:00          NA       NA 0.7450867 0.8776683
## 17100 2019-06-01 16:00:00          NA       NA 0.8537967 0.8781500
## 18100 2019-06-01 17:00:00          NA       NA 0.7619217 1.0142717
## 19100 2019-06-01 18:00:00          NA       NA 1.0113517 0.9262517
## 20100 2019-06-01 19:00:00          NA       NA 1.4265567 1.1001183
## 21100 2019-06-01 20:00:00          NA       NA 1.9860583 1.2447117
## 22100 2019-06-01 21:00:00          NA       NA 1.5541933 0.9866233
## 23100 2019-06-01 22:00:00          NA       NA 0.8684533 0.7458450
## 24100 2019-06-01 23:00:00          NA       NA 1.4489167 0.7623667
```

Dividing the Data into two sets. One which we already have hourly load of, and other for which we need to predict the hourly load.

```
House26_Train = House26[1:8760,]
House26_Predicted = House26[-(1:8760),c(1,4,5)]
```

cleaning data for daily lag and weekly lag separately by removing NAs from the respective datasets.

```
House26_Train_Daily_lag = House26_Train[,c(1,2,4)]
House26_Train_Weekly_lag = House26_Train[,c(1,2,5)]
##cleaning NAs
House26_Train_Daily_lag = na.omit(House26_Train_Daily_lag)
House26_Train_Weekly_lag = na.omit(House26_Train_Weekly_lag)
```

Forecasting 1st June 2019 load with Linear Regression, SVM and ANN using both weekly lags and daily lags separately for House 26

1. Predicting hourly load using Daily lag and linear regression as forcasting method

```
## linear regression

smp_size = floor(0.80 * nrow(House26_Train_Daily_lag))
train_ind = sample(seq_len(nrow(House26_Train_Daily_lag)), size = smp_size)
train_House26 = House26_Train_Daily_lag[train_ind, ]
test_House26 = House26_Train_Daily_lag[-train_ind, ]

lm_model_Daily_lag_26 = lm(hourly_load~Daily_lag, data = train_House26)
```

```
  Calculating accuracy of the model
```

```
test_House26$LR_Daily_lag_predicted = predict(lm_model_Daily_lag_26, test_House26)
accuracy(test_House26$hourly_load, test_House26$LR_Daily_lag_predicted)
```

```
##                     ME      RMSE       MAE       MPE     MAPE
## Test set -0.005144957 0.4276546 0.2786189 -2.024712 29.99011
```

```
  Predicting hourly load for 1st June 2019
```

```
House26_Predicted$hourly_load_Predicted_LR_Daily_lag =
                                      predict(lm_model_Daily_lag_26,House26_Predicted)
```

2. Predicting hourly load using Daily lag and SVM as forcasting method

```
smp_size = floor(0.80 * nrow(House26_Train_Daily_lag))
train_ind = sample(seq_len(nrow(House26_Train_Daily_lag)), size = smp_size)
train_House26 = House26_Train_Daily_lag[train_ind, ]
test_House26 = House26_Train_Daily_lag[-train_ind, ]

svm_model_Daily_lag_26 = svm(hourly_load~Daily_lag, data = train_House26,
                             type = "eps-regression", kernel = "radial")
```

```
  Calculating accuracy of the model
```

```
test_House26$svm_Daily_lag_predicted = predict(svm_model_Daily_lag_26, test_House26)
accuracy(test_House26$hourly_load, test_House26$svm_Daily_lag_predicted)
```

```
##                     ME      RMSE       MAE      MPE     MAPE
## Test set -0.01569896 0.3995733 0.2456358 -3.32573 25.80534
```

```
  Predicting hourly load for 1st June 2019
```

```
House26_Predicted$hourly_load_Predicted_svm_Daily_lag =
                                  predict(svm_model_Daily_lag_26, House26_Predicted)
```

3. Predicting hourly load using Daily lag and ANN as forcasting method

```
# Scaling the Data
max = apply(House26_Train_Daily_lag[,-1] , 2 , max)
min = apply(House26_Train_Daily_lag[,-1], 2 , min)
scaled = as.data.frame(scale(House26_Train_Daily_lag[,-1],
                             center = min, scale = max - min))

# train - test  splitting the data
smp_size = floor(0.80 * nrow(scaled))
train_ind = sample(seq_len(nrow(scaled)), size = smp_size)
train_House26 = scaled[train_ind, ]
test_House26 = scaled[-train_ind, ]
ANN_model_Daily_lag_26 = neuralnet(hourly_load~Daily_lag, data = train_House26,
                        hidden = c(4,8,4), linear.output = TRUE)
```

Calculating accuracy of the model

```
scaled_prediction = compute(ANN_model_Daily_lag_26, test_House26[,c(1,2)])
test_House26$hourly_load_Predicted = (scaled_prediction$net.result *
                                  (max(House26_Train_Daily_lag$hourly_load, na.rm = T) -
                                   min(House26_Train_Daily_lag$hourly_load, na.rm = T))) +
                                   min(House26_Train_Daily_lag$hourly_load, na.rm = T)

test_House26$hourly_load_Actual = (test_House26$hourly_load *
                                   (max(House26_Train_Daily_lag$hourly_load, na.rm = T) -
                                    min(House26_Train_Daily_lag$hourly_load, na.rm = T))) +
min(House26_Train_Daily_lag$hourly_load, na.rm = T)

accuracy(test_House26$hourly_load_Actual,test_House26$hourly_load_Predicted )
```

```
##                  ME      RMSE       MAE      MPE     MAPE
## Test set 0.01091378 0.4105839 0.2690509 1.560362 28.44916
```

Predicting hourly load for 1st June 2019

```
predicted_load = compute(ANN_model_Daily_lag_26, House26_Predicted[,c(1,2)])

House26_Predicted$hourly_load_Predicted_ANN_Daily_lag = (predicted_load$net.result *
                                  (max(House26_Train_Daily_lag$hourly_load, na.rm = T) -
                                   min(House26_Train_Daily_lag$hourly_load, na.rm = T))) +
                                   min(House26_Train_Daily_lag$hourly_load, na.rm = T)
```

4. Predicting hourly load using Weekly lag and linear regression as forcasting method

```
smp_size <- floor(0.80 * nrow(House26_Train_Weekly_lag))
train_ind <- sample(seq_len(nrow(House26_Train_Weekly_lag)), size = smp_size)
train_House26 <- House26_Train_Weekly_lag[train_ind, ]
test_House26 <- House26_Train_Weekly_lag[-train_ind, ]

lm_model_Weekly_lag_26 = lm(hourly_load~Week_lag, data = train_House26)
```

Calculating accuracy of the model

```
test_House26$Weekly_lag_predicted = predict(lm_model_Weekly_lag_26, test_House26)
accuracy(test_House26$hourly_load, test_House26$Weekly_lag_predicted)
```

```
##                   ME    RMSE       MAE       MPE     MAPE
## Test set 0.01558207 0.46087 0.3115624 0.8767162 32.41884
```

```
  Predicting hourly load for 1st June 2019
```

```
House26_Predicted$hourly_load_Predicted_LR_Weekly_lag =
                                predict(lm_model_Weekly_lag_26, House26_Predicted)
```

5. Predicting hourly load using Weekly lag and SVM as forcasting method

```
smp_size <- floor(0.80 * nrow(House26_Train_Weekly_lag))
train_ind <- sample(seq_len(nrow(House26_Train_Weekly_lag)), size = smp_size)
train_House26 <- House26_Train_Weekly_lag[train_ind, ]
test_House26 <- House26_Train_Weekly_lag[-train_ind, ]

svm_model_Weekly_lag_26 = svm(hourly_load~Week_lag, data = train_House26,
                              type = "eps-regression", kernel = "radial")
```

```
  Calculating accuracy of the model
```

```
test_House26$svm_Weekly_lag_predicted = predict(svm_model_Weekly_lag_26, test_House26)
accuracy(test_House26$hourly_load, test_House26$svm_Weekly_lag_predicted)
```

```
##                   ME      RMSE       MAE       MPE     MAPE
## Test set -0.04634031 0.4532752 0.2865211 -6.998509 30.22411
```

```
  Predicting hourly load for 1st June 2019
```

```
House26_Predicted$hourly_load_Predicted_svm_Weekly_lag =
                                 predict(svm_model_Weekly_lag_26, House26_Predicted)
```

6. Predicting hourly load using Weekly lag and ANN as forcasting method

```
# Scaling the Data
max = apply(House26_Train_Weekly_lag[,-1] , 2 , max)
min = apply(House26_Train_Weekly_lag[,-1], 2 , min)
scaled = as.data.frame(scale(House26_Train_Weekly_lag[,-1],
                             center = min, scale = max - min))

# train - test  splitting the data
smp_size <- floor(0.80 * nrow(scaled))
train_ind <- sample(seq_len(nrow(scaled)), size = smp_size)
train_House26 <- scaled[train_ind, ]
test_House26 <- scaled[-train_ind, ]


ANN_model_Weekly_lag_26 = neuralnet(hourly_load~Week_lag, data = train_House26,
                                    hidden = c(4,8,4), linear.output = TRUE)
```

Calculating accuracy of the model

```
scaled_prediction = compute(ANN_model_Weekly_lag_26, test_House26[,c(1,2)])

test_House26$hourly_load_Predicted = (scaled_prediction$net.result *
(max(House26_Train_Weekly_lag$hourly_load, na.rm = T) -
                                  min(House26_Train_Weekly_lag$hourly_load, na.rm = T))) +
                                  min(House26_Train_Weekly_lag$hourly_load, na.rm = T)

test_House26$hourly_load_Actual = (test_House26$hourly_load *
                                  (max(House26_Train_Weekly_lag$hourly_load, na.rm = T) -
                                   min(House26_Train_Weekly_lag$hourly_load, na.rm = T))) +
                                   min(House26_Train_Weekly_lag$hourly_load, na.rm = T)


accuracy(test_House26$hourly_load_Actual,test_House26$hourly_load_Predicted )
```

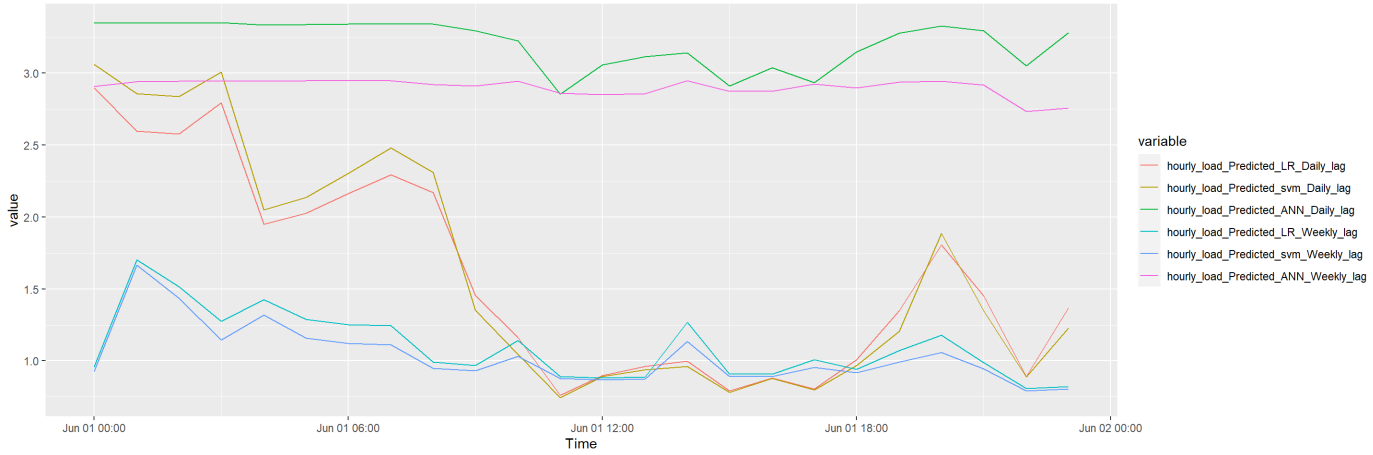
```
##                  ME      RMSE       MAE       MPE     MAPE
## Test set 0.01547734 0.4678632 0.2998003 0.9785318 29.96068
```

Predicting hourly load for 1st June 2019

```
predicted_load = compute(ANN_model_Weekly_lag_26, House26_Predicted[,c(1,3)])
House26_Predicted$hourly_load_Predicted_ANN_Weekly_lag = (predicted_load$net.result *
                                 (max(House26_Train_Weekly_lag$hourly_load, na.rm = T) -
                                  min(House26_Train_Weekly_lag$hourly_load, na.rm = T))) +
                                  min(House26_Train_Weekly_lag$hourly_load, na.rm = T)
```

Comparing all six of the models above

```
house26_plot =  melt(House26_Predicted[,-(1:3)])
house26_plot$Time = House26_Predicted$Time
ggplot(house26_plot, aes(x = Time))+
  geom_line(aes(y = value, color = variable))
```



The analysis of the graph above shows that ANN is not suitable for this forecast.However, Linear Regression and SVM have made a relatively better forecast, both resulted in similar prediction albiet different from those of daily lags in weekly lags.

Saving the forecasted data to a CSV file.

```
write.csv(House26_Predicted,"C:/Users/user/Desktop/STLF/House26_predicted.csv", row.names = TRUE)
```