Forecast of Electricity Consumption

---Bomin Xie，Jing Zhang，
Kanika Sharma，Ziyao Wang

Introduction

Electricity is the only commodity that is produced and consumed simultaneously; therefore, a perfect balance between supply and consumption in the electricity power market must always be maintained. Forecasting electricity consumption is of national interest to any country since electricity is a key source of energy. A reliable forecast of energy consumption, production, and distribution meets the stable and long-term policy. The presence of economies of scale, focus on environmental concerns, regulatory requirements, and a favorable public image, coupled with inflation, rapidly rising energy prices, the emergence of alternative fuels and technologies, changes in life styles, and so on, has generated the need to use modeling techniques which capture the effect of factors such as prices, income, population, technology, and other economic, demographic, policy, and technological variables.

Underestimation could lead to under-capacity utilization, which would result in poor quality of service including localized brownouts, or even blackouts. While on the other hand, an overestimation could lead to the authorization of a plant that may not be needed for several years. The requirement is to ensure optimal phasing of investments, a long-term consideration, and rationalizing pricing structures and designing demand-side management programs, to meet the nature of short- or medium-term needs. The forecast further drives various plans and decisions on investment, construction, and conservation.

In order to carry out forecasting of electricity consumption, we shall be using a dataset collected on smart meter data with time series aggregated by four located industries.

Collecting and describing data

The dataset titled `DT_4_ind` shall be used. The numeric variable is as follows:

- `value`

The non-numeric variables are as follows:

- `date_time`
- `week`
- `date`
- `type`

Exploring data

The following packages need to be loaded as a first step to be carried out:

```
> install.packages("feather")
> install.packages("data.table")
> install.packages("ggplot2")
> install.packages("plotly")
> install.packages("animation")
> library(feather)
> library(data.table)
> library(ggplot2)
> library(plotly)
> library(animation)
```

Regression analysis

The regression model is formulated as follows:

$$y_i = \beta_1 d_{i1} + \beta_2 d_{i2} + \ldots + \beta_{48} d_{i48} + \beta_{49} w_{i1} + \ldots + \beta_{54} w_{i6} + \varepsilon_i$$

Variables (inputs) are of two types of seasonal dummy variables--daily $(d_1, \ldots, d_{48})$ and weekly $(w_1, \ldots, w_6)$. y_i is the electricity consumption at the time *i*, where $i = 1, \ldots, N$. $\beta_1 \ldots \beta_{54}$ are the regression coefficients to be estimated.

Printing the contents of the `AggData` data frame:

```
> AggData
```

The result is as follows:

```
                   date_time    value   week       date               type
     1: 2012-01-02 00:00:00 1590.210 Monday 2012-01-02 Commercial Property
     2: 2012-01-02 00:30:00 1563.772 Monday 2012-01-02 Commercial Property
     3: 2012-01-02 01:00:00 1559.914 Monday 2012-01-02 Commercial Property
     4: 2012-01-02 01:30:00 1584.671 Monday 2012-01-02 Commercial Property
     5: 2012-01-02 02:00:00 1604.281 Monday 2012-01-02 Commercial Property
    ---
 70076: 2012-12-31 21:30:00 3548.279 Monday 2012-12-31   Light Industrial
 70077: 2012-12-31 22:00:00 3488.161 Monday 2012-12-31   Light Industrial
 70078: 2012-12-31 22:30:00 3510.200 Monday 2012-12-31   Light Industrial
 70079: 2012-12-31 23:00:00 3533.678 Monday 2012-12-31   Light Industrial
 70080: 2012-12-31 23:30:00 3414.966 Monday 2012-12-31   Light Industrial
```

Transforming the characters of weekdays to integers: The `as.factor()` function is used to encode a vector as a factor. The `as.integer()` function creates the `AggData[, week]` object of the integer type:

```
> AggData[, week_num := as.integer(as.factor(AggData[, week]))]
```

Printing the contents of the `AggData` data frame after the change:

```
> AggData
```

As is clearly visible, the points are not normal as they are away from the red line. The measurements during the day were moved constantly by the estimated coefficient of the week variable, but the behavior during the day wasn't captured. We need to capture this behavior because weekends, especially, behave absolutely differently.

Improving regression analysis

Creating a linear model: The `lm()` function fits the linear models. `Load ~ 0 + Daily + Weekly + Daily:Weekly` is the new formula. Since `lm()` automatically adds to the linear model intercept, we define it now as `0`. `data = matrix_train` defines the data frame which contains the data:

```
> linear_model_2 <- lm(Load ~ 0 + Daily + Weekly + Daily:Weekly, data =
matrix_train)
```

Printing the contents of the `linear_model_2` data frame after the change:

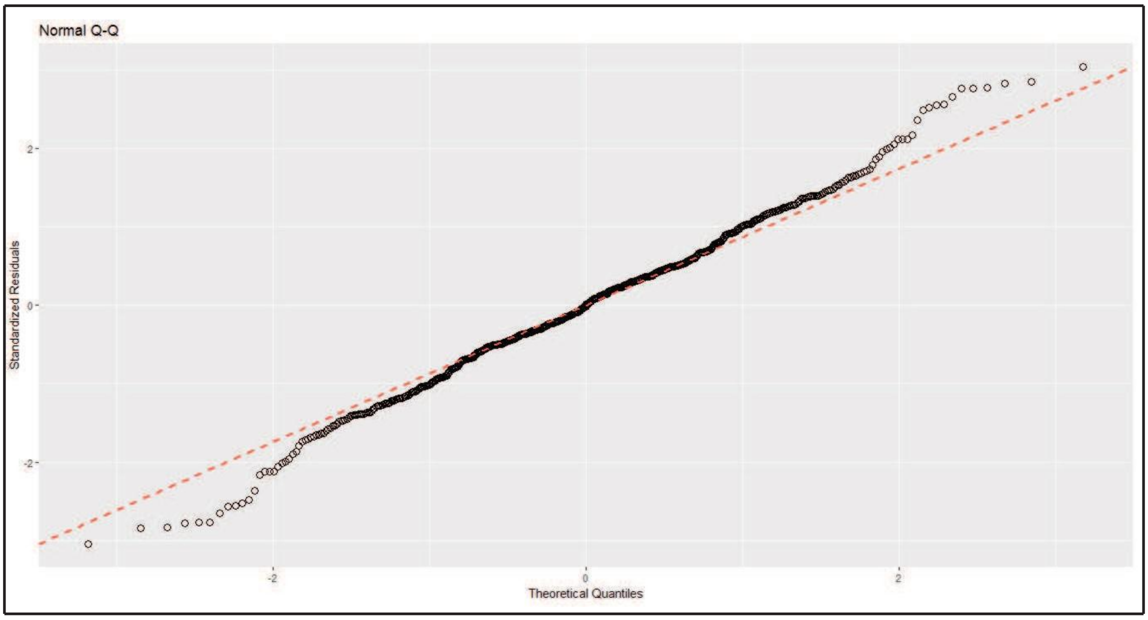
```
> linear_model_2
```

The result is as follows:

```
Call:
lm(formula = Load ~ 0 + Daily + Weekly + Daily:Weekly, data = matrix_train)

Coefficients:
        Daily1          Daily2          Daily3          Daily4          Daily5          Daily6          Daily7
      963.6868        910.1281        832.4827        767.9888        746.0964        709.4052        646.7310
        Daily8          Daily9         Daily10         Daily11         Daily12         Daily13         Daily14
      607.0952        593.1229        579.3818        571.4749        577.7818        575.1910        576.0727
       Daily15         Daily16         Daily17         Daily18         Daily19         Daily20         Daily21
      590.8817        596.6877        599.9783        596.8332        605.5058        617.2628        685.2319
       Daily22         Daily23         Daily24         Daily25         Daily26         Daily27         Daily28
      742.3322        951.7756       1158.7365       1339.7397       1530.8187       1680.5525       1772.6732
       Daily29         Daily30         Daily31         Daily32         Daily33         Daily34         Daily35
     1812.0905       1827.8779       1834.1506       1869.0104       1875.9046       1860.0126       1846.6024
       Daily36         Daily37         Daily38         Daily39         Daily40         Daily41         Daily42
     1789.6251       1731.6673       1682.2293       1586.0122       1571.6402       1496.8899       1358.2929
       Daily43         Daily44         Daily45         Daily46         Daily47         Daily48         Weekly2
     1245.7848       1166.1215       1091.9825       1066.7010        997.4765        955.2032       -355.3522
       Weekly3         Weekly4         Weekly5         Weekly6         Weekly7  Daily2:Weekly2  Daily3:Weekly2
      -42.9479       -383.8207        -47.4477         43.1739         38.2679         46.9318        138.6611
 Daily4:Weekly2  Daily5:Weekly2  Daily6:Weekly2  Daily7:Weekly2  Daily8:Weekly2  Daily9:Weekly2 Daily10:Weekly2
      211.0272        238.7233        273.2973        342.3193        382.9568        393.0218        420.0024
Daily11:Weekly2 Daily12:Weekly2 Daily13:Weekly2 Daily14:Weekly2 Daily15:Weekly2 Daily16:Weekly2 Daily17:Weekly2
      433.0777        432.5059        439.6293        428.7760        429.8100        424.0835        429.2114
Daily18:Weekly2 Daily19:Weekly2 Daily20:Weekly2 Daily21:Weekly2 Daily22:Weekly2 Daily23:Weekly2 Daily24:Weekly2
      441.2555        483.8489        489.4588        450.5552        461.8552        547.1952        614.2202
Daily25:Weekly2 Daily26:Weekly2 Daily27:Weekly2 Daily28:Weekly2 Daily29:Weekly2 Daily30:Weekly2 Daily31:Weekly2
      635.9273        645.8638        629.4519        538.6554        531.7929        506.9881        510.5959
Daily32:Weekly2 Daily33:Weekly2 Daily34:Weekly2 Daily35:Weekly2 Daily36:Weekly2 Daily37:Weekly2 Daily38:Weekly2
      516.8237        507.8630        524.8920        514.2205        531.7239        513.5249        515.8282
Daily39:Weekly2 Daily40:Weekly2 Daily41:Weekly2 Daily42:Weekly2 Daily43:Weekly2 Daily44:Weekly2 Daily45:Weekly2
      603.7991        572.5889        557.3761        561.1792        551.5947        539.1175        529.6320
Daily46:Weekly2 Daily47:Weekly2 Daily48:Weekly2  Daily2:Weekly3  Daily3:Weekly3  Daily4:Weekly3  Daily5:Weekly3
      491.0887        477.7638        443.8735          5.5692         10.2973         32.1318         32.9816
```

The result is as follows:



Building a forecasting model

We can define a function to return the forecast for a 1 week ahead prediction. The input parameters are `data` and `set_of_date`:

```
> predWeekReg <- function(data, set_of_date){
+ #creating the dataset by dates
+ data_train <- data[date %in% set_of_date]
+ N <- nrow(data_train)
+
+ # number of days in the train set
+ window <- N / period # number of days in the train set
+
+ #1, ..., period, 1, ..., period - daily season periods
+ #feature "week_num"- weekly season
+ matrix_train <- data.table(Load = data_train[, value],
+ Daily = as.factor(rep(1:period, window)),
+ Weekly = as.factor(data_train[, week_num]))
+
+ #creating linear model.
+ # formula - Load ~ 0 + Daily + Weekly + Daily:Weekly
+ # dataset - data = matrix_train
```

Plotting the forecast for a year

Plotting the results:

```
    > datas <- data.table(value = c(as.vector(lm_pred_weeks_1),
    AggData[(type == n_type[1]) & (date %in% n_date[-c(1:14,365)]),
value]),
    date_time = c(rep(AggData[-c(1:(14*48), (17473:nrow(AggData))),
date_time], 2)),
    type = c(rep("MLR", nrow(lm_pred_weeks_1)*ncol(lm_pred_weeks_1)),
    rep("Real", nrow(lm_pred_weeks_1)*ncol(lm_pred_weeks_1))),
    week = c(rep(1:50, each = 336), rep(1:50, each = 336)))

    > saveGIF({
    oopt = ani.options(interval = 0.9, nmax = 50)
    for(i in 1:ani.options("nmax")){
    print(ggplot(data = datas[week == i], aes(date_time, value, group =
type, colour = type)) +
    geom_line(size = 0.8) +
    scale_y_continuous(limits = c(min(datas[, value]), max(datas[,
value]))) +
    theme(panel.border = element_blank(), panel.background =
element_blank(),
    panel.grid.minor = element_line(colour = "grey90"),
    panel.grid.major = element_line(colour = "grey90"),
    panel.grid.major.x = element_line(colour = "grey90"),
    title = element_text(size = 15),
    axis.text = element_text(size = 10),
    axis.title = element_text(size = 12, face = "bold")) +
    labs(x = "Time", y = "Load (kW)",
    title = paste("Forecast of MLR (", n_type[1], "); ", "week: ", i, ";
MAPE: ",
    round(lm_err_mape_1[i], 2), "%", sep = "")))
    ani.pause()
    }
    }, movie.name = "industry_1.gif", ani.height = 450, ani.width = 750)
```