ENERGY FORECASTING

Mohamed Almusallam

Portland State University
ETM 510: Demand Forecasting
Instructors: Dr. Tim Anderson and Dr. Sule Balkan
March 21, 2017

Introduction:

Generating and managing the electrical power is one of the important aspects of the electrical grid. The technical process might be easier to manage if the power load or the power demand is known ahead of time. Some information might be available, but they may not be enough. For example, the power demand usually does not equal to zero, so the lowest possible generated power could be known and it could not equal to zero. However, the highest possible demand might not be always available, and the variation might not be known. Preparing for the future is also important. If population increases, there is a higher probability that power demand will also be increasing. Planning new generation plants is important. To manage existing generation, or to plan for new generation plants, the energy demand might be a major contributor. Forecasting power is one of the ways that are used to get the power demand in the future.

There are multiple ways to get a rough idea of the energy demand. One way is by looking at historical data, and processing that data. The historical data of the New England region, which includes Connecticut, Maine, Massachusetts, New Hampshire, Rhode Island, and Vermont, will be used to establish a forecasting model. The only historical data that will be used are the 2015 and the 2016 data. There are three different forecasting methods used to predict the energy demand. The first method is the linear interpolation, the second method is the Arima model, and the third method is the neural networks. The prediction from each method will be submitted to the GEFCom2017 competition.

The Linear Interpolation Method:

The first method that was used is the linear interpolation method. As noted before, demand forecast for the year of 2017 in the New England region depends on the recorded power demand in the years 2015 and 2016. This method assumes that the relationship between the power demand in 2015 and the power demand in 2016 is linear. This assumption means that if the power demand in 2016 is lower than the power demand in 2015, then the predicted demand for the year 2017 will even be lower than the power demand in 2016.

There are a few ways to establish a linear relationship between the power demand provided in the two historical years to forecast the demand in 2017. One of the ways was to calculate the growth rate. $The\ growth\ rate = \frac{present-past}{past} * 100$. In this case, a forecast for the month of March 2017 was needed, March 2015 would be compared with March 2016 and the average growth rate would be calculated. The power demand forecast for March 2017 will be the based on the demand of March 2016 multiplied by the average growth rate. This approach is one way to treat the data linearly. However, this is not the method that was implemented in the submission for the GEFCom2017 competition.

Another way to get a forecast linearly is to break the data into hours. The month of March has 31 days which means that the month has 744 hours. The historical data provided a power demand for each hour. This model will compare the power demand from one hour in March 2015 with the power demand associated with the same hour in March 2016. For example, this model will compare the power demand from 1st of March 2015 at 1 AM to the power demand from 1st of March 2016. A line could be drawn between the two points. The equation of the line is the following: $y = ax + b$, where y represents the power demand, x represents the year, a represents the slope of the line, and b represents the y axis intercept. Since the x-axis represents

the year, the line between the two points could be extended to the year of 2017. The y-axis point associated with the year 2017 will be the power forecast for the year 2017 at that specific hour. Figure 1 is an example of how the model is constructed. The figure compares the power demand in March 1st at 1 AM. The equation of the line was between 2015 and 2016 data points was first calculated, then using that equation to get the projected 2017 power demand.

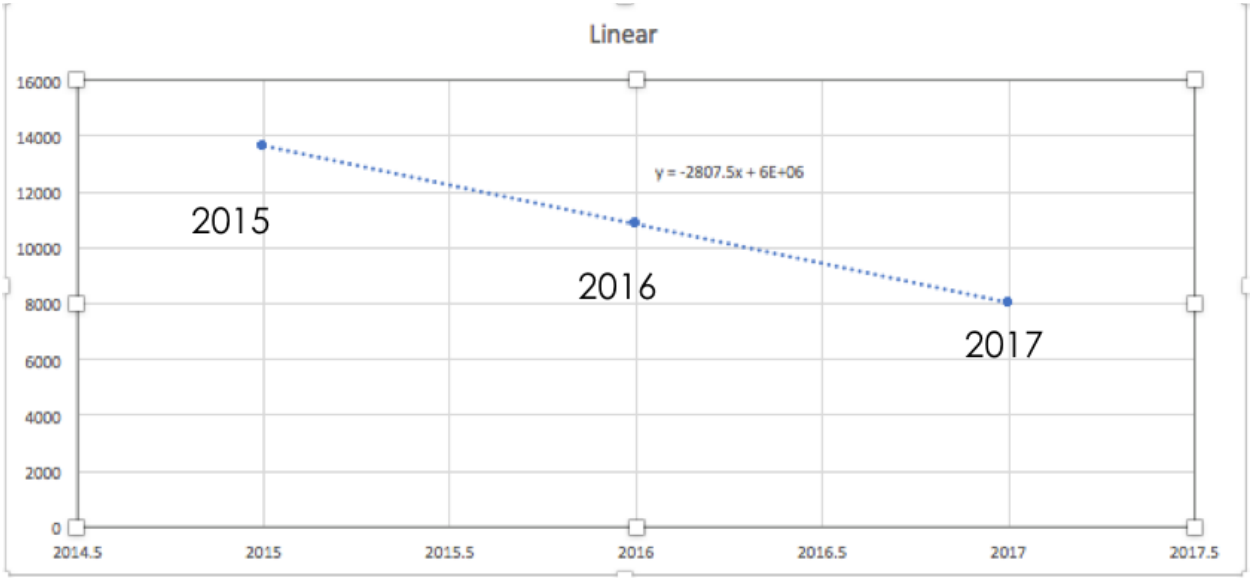

Figure 1: Linear Method line example

The software program R was used to read all the historical data from excel sheets. Then the program will divide the data into hours, and the code will compare the power demand associated with each year individually to slope (a) and the y-axis intercept (b) from the line equation $y = ax + b$. After having the slope and the y-axis intercept, the code will calculate the power demand for the 2017. The R code will be repeating the same process with the help of the for loop. In each iteration, the code will pick a different hour from the historical data, and a new slope (a) and y-axis intercept (b) will be calculated, which means a new power prediction point.

Figure 2, shows the March 2015 and 2016 power demand along with the 2017 forecast for New England region (Total demand). The blue line represents the demand for 2015, the

orange line represents the demand for 2016, and the grey line represents the projected demand for 2017. Figure 2, shows that the linear method is not very accurate and might be a bad forecasting method for this application. After examining the historical data, the lowest possible power demand is about 9700 MW. It is possible for the forecast to predict power demand that is lower than 9700 MW, however predicting half of that suggests that the forecast is not very accurate. The lowest predicted power demand was about 5000 MW, which is impossible to have when looking at the total demand in the New England region. Another problem with the linear method results is trend. Looking at the 2015 and the 2016 data suggests that the power demand is declining, which in theory might indicate that the power demand in 2017 might be slightly lower than the previous year. However, the forecast does not show any type of trend declining trend, which could be an indicating that the forecast is not accurate at all.

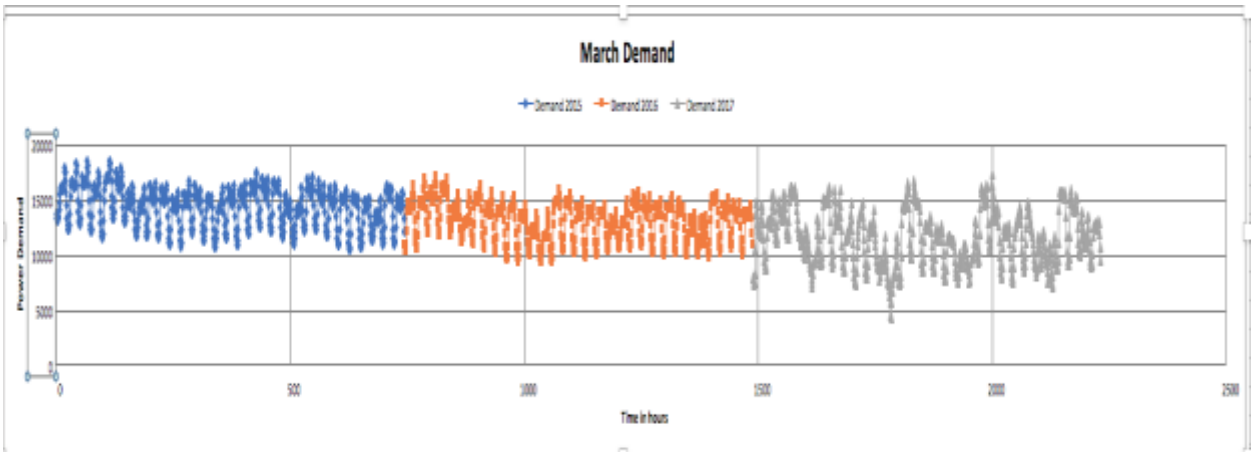

Figure 2: Linear Method March Forecast

As noted before, one surprising finding in this method is the very low predicted demand. This point appeared on March 13th at 8 AM, which is surprising. The power demand for a day usually starts to increase in the morning at around 7 AM, and the power starts to decrease at night at around 9 PM. Figure 3, shows the power demand for the March 13th. The problem here

was because the method compares a point from 2015 with a point from 2016, and if the difference between the two points is large then the difference between 2016 and 2017 will also be large. At 8 AM, the difference between 2015 and 2016 was large and the slope was negative, so the forecast will have a decreasing power demand in the morning.

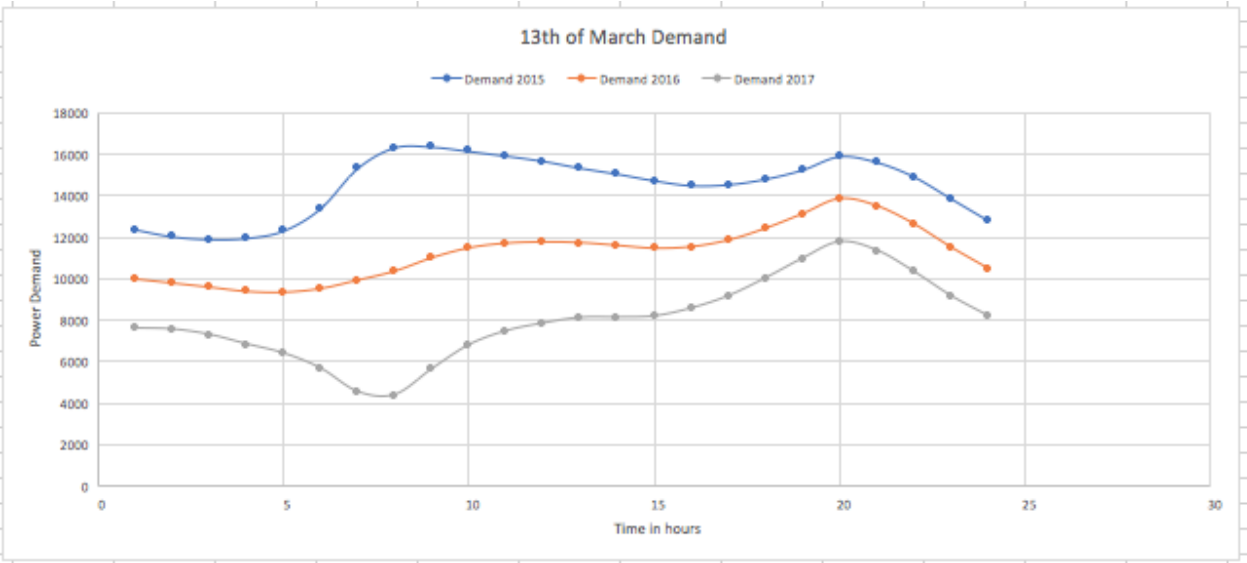

Figure 3: Linear Method March 13th

The ARIMA Method:

The second method that was used is the ARIMA method. The ARIMA method is one of the methods that is used to forecast time series data (Hyndman & Athanasopoulos, 2016). The ARIMA model equation is usually represented by the following (p, d, q), where p represents the autoregressive order, (d) represents the number of differencing needed to make the data stationary, and the (q) represents the moving average order ("Introduction to ARIMA models," n.d.).

In order to use the ARIMA method, the data should have a few properties. The data used in the ARIMA method should be stationery (Hyndman & Athanasopoulos, 2016). Meaning that the data should not have a trend, and it should not depend on time. Also, the data should not have

any type of season in it. The way to make the data stationary is by differencing (Hyndman & Athanasopoulos, 2016). The current observation minus the previous observation is one of the steps to make the data stationary. The log of the power demand historical data will be taken before differencing the data in order to for the data to be stationary on variance (Upadhyay, 2015).

In the linear interpolation, the data was divided into hours, and there was a linear model for each forecast. In the ARIMA model, looking at individual hours are not enough to construct a model for the forecast. The 2015 and 2016 power demand data were divided into days rather than hours. A day from 2015 will be combined with the same day from 2016 to get the model that projected demand for 2017. For example, the program will take March 1st from the 2015 data set and combine it with March 1st from the 2016 data set, and the ARIMA model will be created using these two days.

It is hard to determine the required ARIMA(p, d, q) term for each individual model. The R software has a package that is called the forecast package. One of the functions that is provided in this package is the auto.arima( ) function (Hyndman, O'Hara-Wild, Bergmeir, Razbash, & Wang, 2017). The function will examine the data, which is in this case the day from 2016 followed by the day from 2017, and provide the necessary Arima model terms (p, d, q). After getting the model, another function from the same package, which is called forecast( ), will be used to get the predication of the power demand for 2017. The R code will be repeating the same process for 31 days. There will be 31 different Arima models for each state in the New England region.

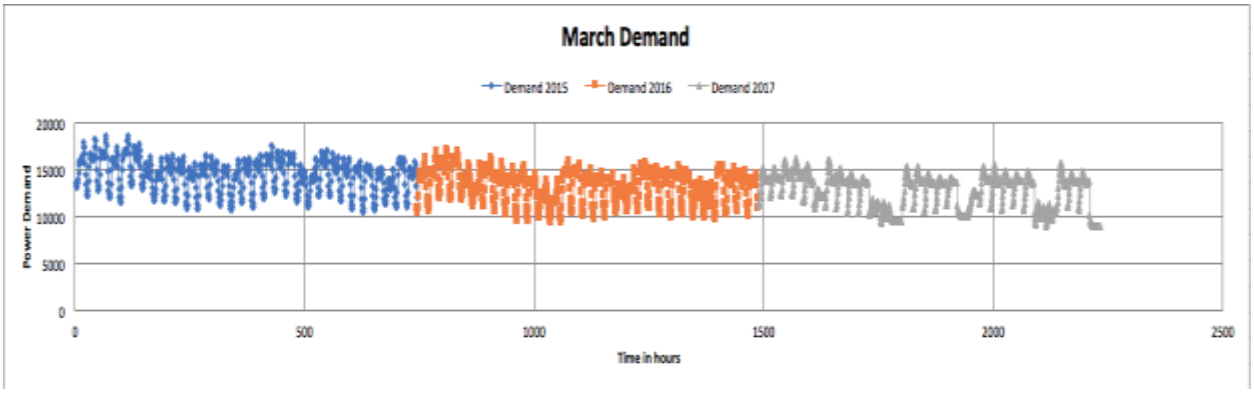

Figure 4: ARIMA Model March 2017 Forecast

Figure 4, shows the power demand forecast for March 2017. The grey line which is the demand forecast seems to be following a pattern that is very similar to the previous years. The prediction in this model does not have any values that are very low or very high unlike the linear interpolation model. However, the power demand for some days in the predication is lower than usual. One of the reasons behind the low power demand in some days is the usage of the auto.arima() function. The predication for each day of the month of March has a unique Arima model. That model was constructed by the auto.arima function. If the data are close, meaning there is no clear trend in the data, the auto.arima model might assume that the data is stationary, and differencing the data is not needed. The auto.arima function might also decide not to include a constant to the prediction, which might have some impact on the prediction.

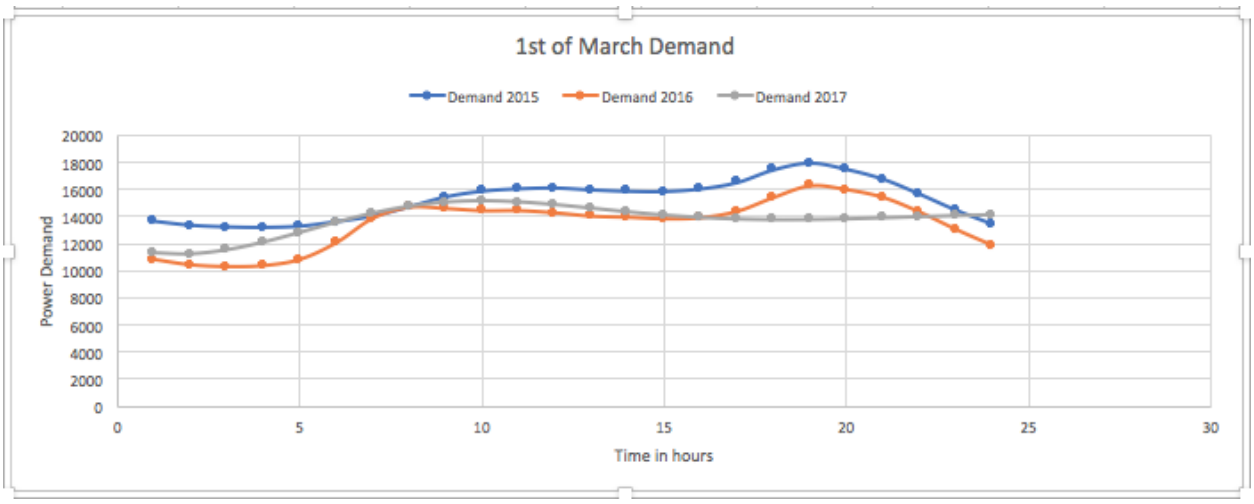

Figure 5: ARIMA Model 1st of March Forecast

Figure 5, compares the power demand obtained from 2015 and 2016 with the power forecast of 2017. The typical power demand for a day has a specific pattern as seen in the 2015 and 2016 demands. The 2017 forecast does not follow the same pattern. The forecast will start low at the morning, then it will start increasing until noon, then the forecast will remain about constant for the rest of the day. This is true for every forecast that was produced by the Arima model. This issue might be created because the Arima model that was used is the non-seasonal Arima model. The seasonal Arima model has more terms that are related to seasonality. We have tried to create a seasonal Arima model in the software R, but the software will usually not accept the seasonal terms of the Arima model, so the code only uses the non-seasonal Arima model.

The Neural Networks Method:

The last method that was used is neural networks. The neural networks consist of connected neurons that can process data (Stergiou & Siganos, n.d.). The neural networks are useful in many application since it can read data and determine and the patterns and the trend in the data (Stergiou & Siganos, n.d.). The neural network usually is constructed by layers ("Basic Introduction To Neural Networks," n.d.). The first layer of the neural network is the input layer, and the last layer of the neural network is usually the output layer. Between the input layer and the output layer, there exist a layer that is called the hidden layer. All the connected neurons are placed in the hidden layer. There are connections between the inputs, the neurons, and the outputs of the system. Each connection might carry a coefficient or weight, so the input signal of the neuron might be multiplied by the weight. The output signal of the neuron depends on the neuron function. For example, one type of neurons is called perceptrons. The output of perceptrons can only be 1 or 0 (Nielsen, 2017). If the input signal of a neuron multiplied by the assigned weight is larger than some defined value, the output of the neuron will be a 1, otherwise the output of the neuron will be a 0 (Nielsen, 2017). There are other neuron functions that are being used, and they might be producing a different output.

The previous forecasting methods depends only on the historical power demand data. After studying the historical data, temperature has some influence over the power demand. The power demand in 2016 decreased on average by 20%, while the temperature in 2016 higher than the temperature of 2015 on average by 9%. There is some relationship between temperature and power demand. The neural network allows to have multiple inputs to create a single output.

To forecast the month of April, a model needs to be created. Neural networks are good at estimating a function. If the inputs are known, and the output is also known, neural network

could calculate the required weights and number of neurons needed to get from inputs to output. To create the model the output of the network is the 2016 power demand and the following were used as inputs:

- 2015 Date (04/01/2015)
- 2015 Time of day
- 2015 Temperature
- 2015 Power demand
- 2016 Date (04/01/2016)
- 2016 Time of day
- 2016 Temperature

After the model is created inputs can be changed to obtain the required forecast. In this case, any 2015 input will be replaced with 2016 data, and any 2016 input will be replaced with 2017 data. The following represents the new set of inputs that are going to produce the desired forecast:

- 2016 Date (04/01/2016)
- 2016 Time of day
- 2016 Temperature
- 2016 Power demand
- 2017 Date (04/01/2016)
- 2017 Time of day
- 2017 Temperature (not available, the average of 2015 and 2016 temperatures was used)

The R software has a package called caret. One of the functions provided in this package is the train function. The train function will be used to model the of the neural network. After specifying the output and the inputs of the network, the train function will determine the required number of neurons, and calculate the weights. After obtaining the model the inputs will be changed as specified previously, and the predict function in R will be used to predict the 2017 forecast.

Figure 6, shows the results of the of the month of April. The results using the neural networks are much better than any method used previously. The power forecast did not include any values that are low or high, and power demand for each day seems to be consistent.

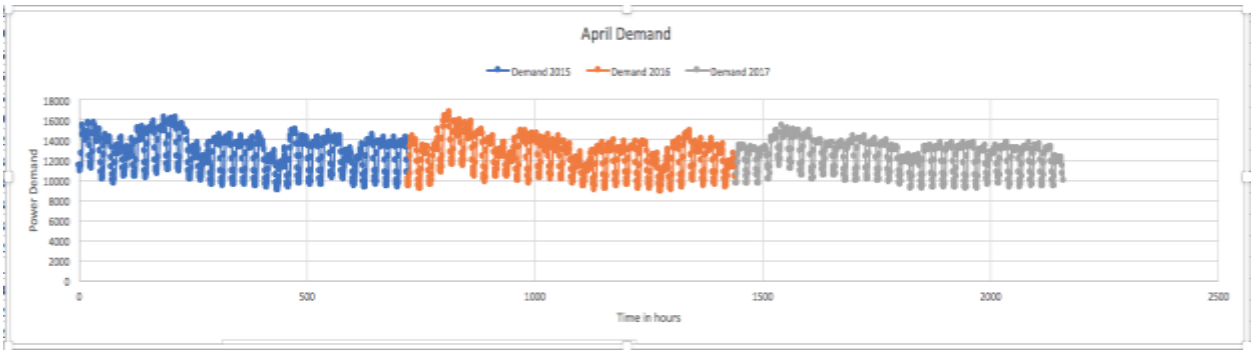

Figure 6: Neural Network Method April Forecast

Figure 7, shows the power demand and the power forecast of April 1st. As can be seen, the forecast follows a typical day pattern unlike the forecast produced by the ARIMA method. Also, the forecast seems to be lower than both 2015 and 2016 demands. The neural network was trained to get 2016 power demand given some 2015 inputs. The 2016 power demand was slightly lower than the 2015 demand, so the neural networks might assume that 2017 might also be lower than 2016.

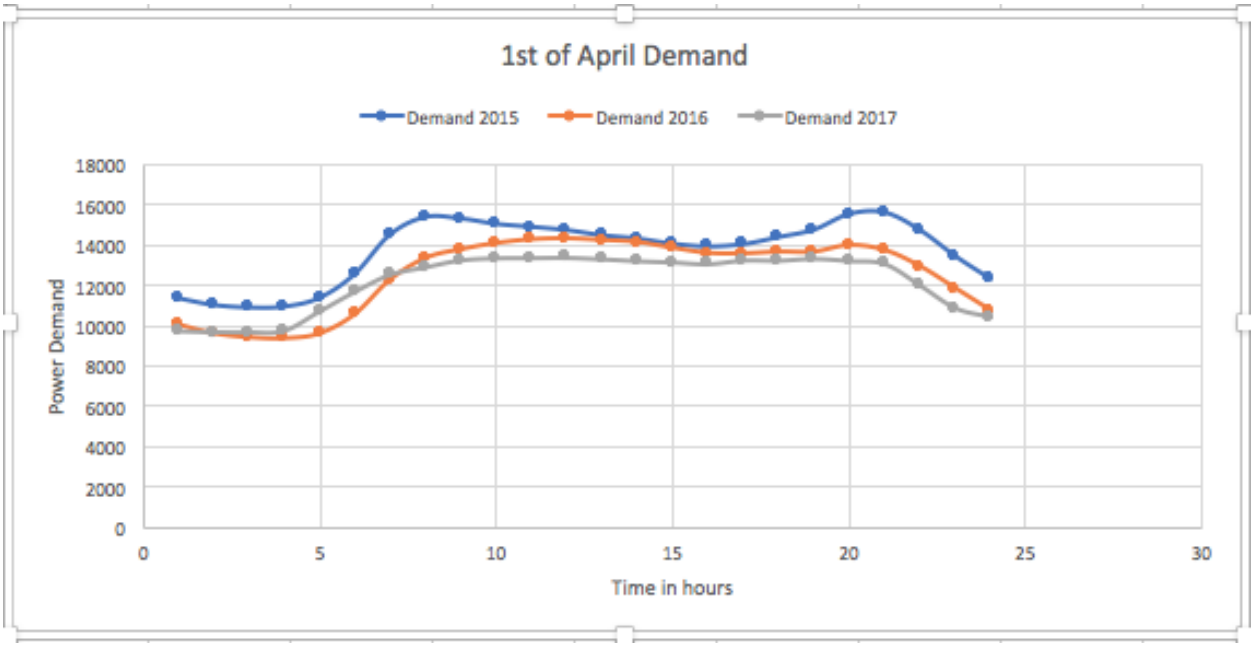

Figure 7: Neural Networks April 1st Demand and Forecast

Conclusion:

The three methods that are used to forecast energy are the linear, Arima, and the neural networks. By looking at the predictions from all of the methods, the most accurate model would be the neural networks. The neural networks method gave the flexibility to add external variables that could shape the output. The neural net model does not differentiate between weekdays and weekend, or between regular days and holidays, but the neural net might allow to add new variables and inputs to the model such as weekends. There are other methods, not discussed in this report, that could be used for forecasting. One of the methods is the Random forest, which is a type of machine learning, like the neural networks. Random forest might produce results that are similar to the neural net, but the data will be processed differently. The only struggle that I experienced in this project is choosing a forecasting method. The options are wide, and specific methods are designed to work with specific applications. The R software made choosing a method a little bit easier, since many methods are available as functions, which gave access to most method results.

References:

A Basic Introduction To Neural Networks. (n.d.). Retrieved from http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

Hyndman, R. J., & Athanasopoulos, G. (2016). *Forecasting: principles and practice*. Retrieved from https://www.otexts.org/fpp

Hyndman, R. J., O'Hara-Wild, M., Bergmeir, C., Razbash, S., & Wang, E. (2017, February 23). Package 'forecast'. Retrieved from https://cran.r-project.org/web/packages/forecast/forecast.pdf

Introduction to ARIMA models. (n.d.). Retrieved from https://people.duke.edu/~rnau/411arim.htm

Nielsen, M. (2017, January). Neural networks and deep learning. Retrieved from http://neuralnetworksanddeeplearning.com/

Stergiou, C., & Siganos, D. (n.d.). Neural Networks. Retrieved from https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html

Upadhyay, R. (2015, June 28). Step-by-Step Graphic Guide to Forecasting through ARIMA Modeling using R – Manufacturing Case Study Example (Part 4). Retrieved from http://ucanalytics.com/blogs/step-by-step-graphic-guide-to-forecasting-through-arima-modeling-in-r-manufacturing-case-study-example/

Appendix:

The linear method R code:

```
library(XLConnect)  # Read and write Excel files
library(tibble)     # A better way of managing dataframes
library(dplyr)      # Data manipulation
library(ggplot2)    # Graphics

wk = loadWorkbook("2015.xls")
wk2 = loadWorkbook("2016.xls")

ISO_NE_CA_2015 <- select(as_tibble(readWorksheet(wk, sheet="ISONE CA")),
                  Date, Hour, DEMAND, DryBulb, DewPnt)
ISO_NE_CA_2016 <- select(as_tibble(readWorksheet(wk2, sheet="ISO NE CA")),
                  Date, Hr_End, RT_Demand, Dry_Bulb, Dew_Point)

 Demand2015 <- ISO_NE_CA_2015["DEMAND"]
 Demand2016 <- ISO_NE_CA_2016["RT_Demand"]
 DryBulb2015 <- ISO_NE_CA_2015["DryBulb"]
 DryBulb2016 <- ISO_NE_CA_2016["Dry_Bulb"]
 DewPnt2015 <- ISO_NE_CA_2015["DewPnt"]
 DewPnt2016 <- ISO_NE_CA_2016["Dew_Point"]
 size <- dim(Demand2015)
 ave_Demand <- c()
 ave_DryBulb <- c()
 ave_DewPoint <- c()
 Demand2017 <- c()
 for (i in 1:size[1]){
  ave_Demand[i] <- ((Demand2015[i,1]+Demand2016[i,1])/2)
  ave_DryBulb[i] <- ((DryBulb2015[i,1]+DryBulb2016[i,1])/2)
  ave_DewPoint[i] <- ((DewPnt2015[i,1]+DewPnt2016[i,1])/2)

  m <- (Demand2016[i,1]-Demand2015[i,1])
  b <- (Demand2016[i,1]-m*2016)
  Demand2017[i] <- round((m*2017 + b),digits = 2)
 }

 wk4 <- loadWorkbook("output1.xlsx",create = TRUE)
 createSheet(wk4, name="Linear")
 input <-matrix(data =Demand2017,nrow = 8760,ncol = 1,byrow = TRUE)
 input2 <- data.frame('Hour' = c(1:8760),'Demand2015'=Demand2015)
 input3 <- data.frame('Demand2016'=Demand2016,'Demand2017'= input)
 writeWorksheet(wk4,input2,sheet="Linear",startRow = 1,startCol = 1)
 saveWorkbook(wk4)
 writeWorksheet(wk4,input3,sheet="Linear",startRow = 1,startCol = 3)
 saveWorkbook(wk4)
```

The Arima Model R code:

```
Sys.setenv(JAVA_HOME='C:\\Program Files\\Java\\jre1.8.0_121')


library(XLConnect)  # Read and write Excel files
library(tibble)     # A better way of managing dataframes
library(dplyr)      # Data manipulation
library(ggplot2)    # Graphics

wk = loadWorkbook("2015.xls")
wk2 = loadWorkbook("2016.xls")


ISO_NE_CA_2015 <- select(as_tibble(readWorksheet(wk, sheet="NEMASSBOST")),
                         Date, Hour, DEMAND, DryBulb, DewPnt)

ISO_NE_CA_2016 <- select(as_tibble(readWorksheet(wk2, sheet="NEMA")),
                         Date, Hr_End, RT_Demand, Dry_Bulb, Dew_Point)

  Demand2015 <- ISO_NE_CA_2015["DEMAND"]
  Demand2015 <- Demand2015[1418:2161,1]
  Demand2016 <- ISO_NE_CA_2016["RT_Demand"]
  Demand2016 <- Demand2016[1442:2185,1]
  DryBulb2015 <- ISO_NE_CA_2015["DryBulb"]
  DryBulb2015 <- DryBulb2015[1417:2160,1]
  DryBulb2016 <- ISO_NE_CA_2016["Dry_Bulb"]
  DryBulb2016 <- DryBulb2016[1417:2160,1]
  DewPnt2015 <- ISO_NE_CA_2015["DewPnt"]
  DewPnt2015 <- DewPnt2015[1417:2160,1]
  DewPnt2016 <- ISO_NE_CA_2016["Dew_Point"]
  DewPnt2016 <- DewPnt2016[1417:2160,1]

  DayDemand <- c()
  ForecastDemand <- c()
  u <-1
  k <- 1
  b <- 1
for (day in 1:31){

    for(oo in 1:48){
      if (oo <= 24){
      DayDemand[oo] <- Demand2015[oo,1]
      b <- b+1
      }
      if (oo > 24){
```

```
      DayDemand[oo] <- Demand2016[k,1]
      k <- k+1
     }
   }
wwww <-ts(DayDemand,start=1,frequency=24)


diffr <- diff(log10(as.numeric(wwww)))


#par(mfrow = c(1,2))
#acf(ts(diffr),main = "ACF Tractor Sales")
#pacf(ts(diffr),main = "PACF Tractor Sales")

require(forecast)


ARIMAfit <- auto.arima((log10(as.numeric(wwww))), approximation =
FALSE,trace=FALSE)


pred <- predict(ARIMAfit,n.ahead = 24)

kkkk <- 10^(pred$pred)


for (uu in 1:24){
 ForecastDemand[u] <- kkkk[uu]
 u <- u+1
}


}

wk4 <- loadWorkbook("output5.xlsx",create = TRUE)
 createSheet(wk4, name="Linear")

 input <- data.frame('Demand2017' = ForecastDemand)
 writeWorksheet(wk4,input,sheet="Linear",startRow = 1,startCol = 1)
 saveWorkbook(wk4)

plot(wwww)
lines(10^(pred$pred))
lines(10^(pred$pred+2*pred$se))
lines(10^(pred$pred-2*pred$se))
```

The Neural Network R code:

```
Sys.setenv(JAVA_HOME='C:\\Program Files\\Java\\jre1.8.0_121')

library(XLConnect)  # Read and write Excel files
library(tibble)     # A better way of managing dataframes
library(dplyr)      # Data manipulation
library(ggplot2)    # Graphics

wk = loadWorkbook("2015.xls")
wk2 = loadWorkbook("2016.xls")

ISO_NE_CA_20151 <- select(as_tibble(readWorksheet(wk, sheet="ISONE CA")),
                  Date, Hour, DEMAND, DryBulb, DewPnt)
ISO_NE_CA_20152 <- select(as_tibble(readWorksheet(wk, sheet="ME")),
                  Date, Hour, DEMAND, DryBulb, DewPnt)
ISO_NE_CA_20153 <- select(as_tibble(readWorksheet(wk, sheet="NH")),
                  Date, Hour, DEMAND, DryBulb, DewPnt)
ISO_NE_CA_20154 <- select(as_tibble(readWorksheet(wk, sheet="VT")),
                  Date, Hour, DEMAND, DryBulb, DewPnt)
ISO_NE_CA_20155 <- select(as_tibble(readWorksheet(wk, sheet="CT")),
                  Date, Hour, DEMAND, DryBulb, DewPnt)
ISO_NE_CA_20156 <- select(as_tibble(readWorksheet(wk, sheet="RI")),
                  Date, Hour, DEMAND, DryBulb, DewPnt)
ISO_NE_CA_20157 <- select(as_tibble(readWorksheet(wk, sheet="SEMASS")),
                  Date, Hour, DEMAND, DryBulb, DewPnt)
ISO_NE_CA_20158 <- select(as_tibble(readWorksheet(wk, sheet="WCMASS")),
                  Date, Hour, DEMAND, DryBulb, DewPnt)
ISO_NE_CA_20159 <- select(as_tibble(readWorksheet(wk, sheet="NEMASSBOST")),
                  Date, Hour, DEMAND, DryBulb, DewPnt)

ISO_NE_CA_20161 <- select(as_tibble(readWorksheet(wk2, sheet="ISO NE CA")),
                  Date, Hr_End, RT_Demand, Dry_Bulb, Dew_Point)
ISO_NE_CA_20162 <- select(as_tibble(readWorksheet(wk2, sheet="ME")),
                  Date, Hr_End, RT_Demand, Dry_Bulb, Dew_Point)
ISO_NE_CA_20163 <- select(as_tibble(readWorksheet(wk2, sheet="NH")),
                  Date, Hr_End, RT_Demand, Dry_Bulb, Dew_Point)
ISO_NE_CA_20164 <- select(as_tibble(readWorksheet(wk2, sheet="VT")),
                  Date, Hr_End, RT_Demand, Dry_Bulb, Dew_Point)
ISO_NE_CA_20165 <- select(as_tibble(readWorksheet(wk2, sheet="CT")),
                  Date, Hr_End, RT_Demand, Dry_Bulb, Dew_Point)
ISO_NE_CA_20166 <- select(as_tibble(readWorksheet(wk2, sheet="RI")),
                  Date, Hr_End, RT_Demand, Dry_Bulb, Dew_Point)
ISO_NE_CA_20167 <- select(as_tibble(readWorksheet(wk2, sheet="SEMA")),
```

```
                Date, Hr_End, RT_Demand, Dry_Bulb, Dew_Point)
ISO_NE_CA_20168 <- select(as_tibble(readWorksheet(wk2, sheet="WCMA")),
                Date, Hr_End, RT_Demand, Dry_Bulb, Dew_Point)
ISO_NE_CA_20169 <- select(as_tibble(readWorksheet(wk2, sheet="NEMA")),
                Date, Hr_End, RT_Demand, Dry_Bulb, Dew_Point)


 Demand2015 <- c()

 HOLD <- ISO_NE_CA_20151["DEMAND"]
 Demand2015 <- HOLD[2162:2881,1]
 Demand2015 <- data.frame(Demand2015)
 HOLD <- ISO_NE_CA_20152["DEMAND"]
 Demand2015[,2] <- HOLD[2162:2881,1]
 HOLD <- ISO_NE_CA_20153["DEMAND"]
 Demand2015[,3] <- HOLD[2162:2881,1]
 HOLD <- ISO_NE_CA_20154["DEMAND"]
 Demand2015[,4] <- HOLD[2162:2881,1]
 HOLD <- ISO_NE_CA_20155["DEMAND"]
 Demand2015[,5] <- HOLD[2162:2881,1]
 HOLD <- ISO_NE_CA_20156["DEMAND"]
 Demand2015[,6] <- HOLD[2162:2881,1]
 HOLD <- ISO_NE_CA_20157["DEMAND"]
 Demand2015[,7] <- HOLD[2162:2881,1]
 HOLD <- ISO_NE_CA_20158["DEMAND"]
 Demand2015[,8] <- HOLD[2162:2881,1]
 HOLD <- ISO_NE_CA_20159["DEMAND"]
 Demand2015[,9] <- HOLD[2162:2881,1]

 Demand2016 <- c()


 HOLD <- ISO_NE_CA_20161["RT_Demand"]
 Demand2016 <- HOLD[2162:2881,1]
 Demand2016 <- data.frame(Demand2016)
 HOLD <- ISO_NE_CA_20162["RT_Demand"]
 Demand2016[,2] <- HOLD[2162:2881,1]
 HOLD <- ISO_NE_CA_20163["RT_Demand"]
 Demand2016[,3] <- HOLD[2162:2881,1]
 HOLD <- ISO_NE_CA_20164["RT_Demand"]
 Demand2016[,4] <- HOLD[2162:2881,1]
 HOLD <- ISO_NE_CA_20165["RT_Demand"]
 Demand2016[,5] <- HOLD[2162:2881,1]
 HOLD <- ISO_NE_CA_20166["RT_Demand"]
```

```
Demand2016[,6] <- HOLD[2162:2881,1]
HOLD <- ISO_NE_CA_20167["RT_Demand"]
Demand2016[,7] <- HOLD[2162:2881,1]
HOLD <- ISO_NE_CA_20168["RT_Demand"]
Demand2016[,8] <- HOLD[2162:2881,1]
HOLD <- ISO_NE_CA_20169["RT_Demand"]
Demand2016[,9] <- HOLD[2162:2881,1]


DryBulb2015 <- c()
HOLD <- ISO_NE_CA_20151["DryBulb"]
DryBulb2015 <- HOLD[2162:2881,1]
DryBulb2015 <- data.frame(DryBulb2015)
HOLD <- ISO_NE_CA_20152["DryBulb"]
DryBulb2015[,2] <- HOLD[2162:2881,1]
HOLD <- ISO_NE_CA_20153["DryBulb"]
DryBulb2015[,3] <- HOLD[2162:2881,1]
HOLD <- ISO_NE_CA_20154["DryBulb"]
DryBulb2015[,4] <- HOLD[2162:2881,1]
HOLD <- ISO_NE_CA_20155["DryBulb"]
DryBulb2015[,5] <- HOLD[2162:2881,1]
HOLD <- ISO_NE_CA_20156["DryBulb"]
DryBulb2015[,6] <- HOLD[2162:2881,1]
HOLD <- ISO_NE_CA_20157["DryBulb"]
DryBulb2015[,7] <- HOLD[2162:2881,1]
HOLD <- ISO_NE_CA_20158["DryBulb"]
DryBulb2015[,8] <- HOLD[2162:2881,1]
HOLD <- ISO_NE_CA_20159["DryBulb"]
DryBulb2015[,9] <- HOLD[2162:2881,1]


DryBulb2016 <- c()

HOLD <- ISO_NE_CA_20161["Dry_Bulb"]
DryBulb2016 <- HOLD[2162:2881,1]
DryBulb2016 <- data.frame(DryBulb2016)
HOLD <- ISO_NE_CA_20162["Dry_Bulb"]
DryBulb2016[,2] <- HOLD[2162:2881,1]
HOLD <- ISO_NE_CA_20163["Dry_Bulb"]
DryBulb2016[,3] <- HOLD[2162:2881,1]
HOLD <- ISO_NE_CA_20164["Dry_Bulb"]
DryBulb2016[,4] <- HOLD[2162:2881,1]
HOLD <- ISO_NE_CA_20165["Dry_Bulb"]
DryBulb2016[,5] <- HOLD[2162:2881,1]
```

```
  HOLD <- ISO_NE_CA_20166["Dry_Bulb"]
  DryBulb2016[,6] <- HOLD[2162:2881,1]
  HOLD <- ISO_NE_CA_20167["Dry_Bulb"]
  DryBulb2016[,7] <- HOLD[2162:2881,1]
  HOLD <- ISO_NE_CA_20168["Dry_Bulb"]
  DryBulb2016[,8] <- HOLD[2162:2881,1]
  HOLD <- ISO_NE_CA_20169["Dry_Bulb"]
  DryBulb2016[,9] <- HOLD[2162:2881,1]


  temp17 <- DryBulb2015

  for (zz in 1:9){
    for (z in 1:720){
      temp17[z,zz] <- (DryBulb2015[z,zz]+DryBulb2016[z,zz])/2
    }
  }


library(caret)
Date15 <- c()
Date16 <- c()
Date17 <- c()
hour15 <- c()
hour16 <- c()
hour17 <- c()
t <- 0
u <- 1
for (dayall in 1:30){
  for (hourall in 1:24){
    Date15[u] <- 03012015 +t
    Date16[u] <- 03012016 +t
    Date17[u] <- 03012017 +t
    hour15[u] <- hourall
    hour16[u] <- hourall
    hour17[u] <- hourall
    u <- u+1
  }
  t <- t +10000
}
eee_ <- Demand2015


for (w in 1:10) {
```

```
em <-
data.frame(Date15,hour15,DryBulb2015[,w],Demand2015[,w],Date16,hour16,DryBulb2016[,w]
,Demand2016[,w])
maxs <- apply(em, 2, max)
mins <- apply(em, 2, min)
colnames(em) <-
c("Date1","Hour1","temp1","Demand1","Date2","Hour2","temp2","Demand2")

scaled_in <- as.data.frame(scale(em, center = mins, scale = maxs - mins))

colnames(scaled_in) <-
c("Date1","Hour1","temp1","Demand1","Date2","Hour2","temp2","Demand2")
library(nnet)

fit3 <-
train(Demand2~Date1+Hour1+temp1+Demand1+Date2+Hour2+temp2,data=scaled_in,linout =
TRUE, method = 'nnet')

yyyy <-
data.frame(Date16,hour16,DryBulb2016[,w],Demand2016[,w],Date17,hour17,temp17[,w])
colnames(yyyy) <-  c("Date1","Hour1","temp1","Demand1","Date2","Hour2","temp2")

maxs <- apply(yyyy, 2, max)
mins <- apply(yyyy, 2, min)

scaled_out <- as.data.frame(scale(yyyy, center = mins, scale = maxs - mins))
colnames(scaled_out) <-  c("Date1","Hour1","temp1","Demand1","Date2","Hour2","temp2")


eee <- predict(fit3,scaled_out)
eee_[,w] <- eee*(max(yyyy$Demand1)-min(yyyy$Demand1))+min(yyyy$Demand1)

}
A <- c()
ave <- c()
stan <- c()
ten <- eee_
wk4 <- loadWorkbook("copy1.xls")
for (c in 1:10){
  if (c < 10){
  for (o in 1:720) {
    A[1] <-  Demand2015[o,c]
    A[2] <-  Demand2016[o,c]
    A[3] <-  eee_[o,c]
```

```
   ave[o] <- eee_[o,c]
   stan[o] <- sd(A)
   ten[o,1] <- ave[o]+(stan[o]*-1.2816)/(sqrt(3))
   ten[o,2] <- ave[o]+(stan[o]*-0.8416)/(sqrt(3))
   ten[o,3] <- ave[o]+(stan[o]*-0.5244)/(sqrt(3))
   ten[o,4] <- ave[o]+(stan[o]*-0.2533)/(sqrt(3))
   ten[o,5] <- eee_[o,c]
   ten[o,6] <- ave[o]+(stan[o]*0.2533)/(sqrt(3))
   ten[o,7] <- ave[o]+(stan[o]*0.5244)/(sqrt(3))
   ten[o,8] <- ave[o]+(stan[o]*0.8416)/(sqrt(3))
   ten[o,9] <- ave[o]+(stan[o]*1.2816)/(sqrt(3))

  }
  }
colnames(ten) <-  c("Q10","Q20","Q30","Q40","Q50","Q60","Q70","Q80","Q90")


  if (c == 1){
     writeWorksheet(wk4,ten,sheet="TOTAL",startRow = 1,startCol = 3)
     saveWorkbook(wk4)
  }

  if (c == 2){
     writeWorksheet(wk4,ten,sheet="ME",startRow = 1,startCol = 3)
     saveWorkbook(wk4)
}
  if (c == 3){
     writeWorksheet(wk4,ten,sheet="NH",startRow = 1,startCol = 3)
     saveWorkbook(wk4)
}
  if (c == 4){
     writeWorksheet(wk4,ten,sheet="VT",startRow = 1,startCol = 3)
     saveWorkbook(wk4)
}
  if (c == 5){
     writeWorksheet(wk4,ten,sheet="CT",startRow = 1,startCol = 3)
     saveWorkbook(wk4)
}
  if (c == 6){
     writeWorksheet(wk4,ten,sheet="RI",startRow = 1,startCol = 3)
     saveWorkbook(wk4)
}
  if (c == 7){
```

```
    writeWorksheet(wk4,ten,sheet="SEMASS",startRow = 1,startCol = 3)
    saveWorkbook(wk4)
}
 if (c == 8){
    Wc <- ten
    writeWorksheet(wk4,ten,sheet="WCMASS",startRow = 1,startCol = 3)
    saveWorkbook(wk4)
}
 if (c == 9){
    NEMA <- ten
    writeWorksheet(wk4,ten,sheet="NEMASSBOST",startRow = 1,startCol = 3)
    saveWorkbook(wk4)
 }


 if (c == 10){
  ten <- Wc + NEMA
    writeWorksheet(wk4,ten,sheet="MASS",startRow = 1,startCol = 3)
    saveWorkbook(wk4)
 }


}
```