SAINT MARY'S UNIVERSITY SINCE 1802

One University. One World. Yours.

Data and Text Mining

MCDA5580

Assignment – 4

Team Members:

NAME	A#	Email
Manoj Bandaru	A00433174	Manoj.bandaru@smu.ca
Aditya Tandon	A00432835	Aditya.Tandon@smu.ca
Khagesh Pandya	A00431429	Khagesh.pandya@smu.ca

Submitted to

Trishla Shah

Contents

Table of Figures

1. Executive Summary

The task was to perform time series prediction on three different time crunches i.e. daily/hourly/15-minute electricity consumption. Also, to compare linear regression (glmFitTime), SVM (svmFitTime), Neural Networks regression techniques (nnFitTime) and Time Series Model (tSeries)– ARIMA (Auto Regressive Integrated Moving Average), to check which model works best and give the least mean absolute percentage error loss, or which model can predict value the best.

2. Summary

The Data is about the electricity consumption for 15-minute period, using python and SQL we have clubbed the consumption and made 2 more csv files of Hourly and Daily consumption. The data given has Time Stamp and the consumption of electricity of that time. Further, the data has been transposed in order to check which regression model can give us the closest value, by identifying then trends, using K-Fold Cross Validation, which in our case is 10-fold cross validation using trainControl method.

3. Methodology

- Transform the dataset
- Perform K-Fold Cross Validation and divide into K Folds.
- Use 3 different Prediction Models – Linear Regression, Support Vector Machine (Regression) with Radial Kernel and Neural Network
- Train a Time Series Model. Use the ARIMA Auto Regressive Integrated Moving Average to identify the underlying trends. Moving average is used in order to take the average of the data points, that defines a new trend, and give the value according to the current trend.
- Compare the Mean Absolute Percentage Error loss of all the models.

4. Prediction on Fifteen min data frequency

The data we used for 15 mins, was directly given to us in the MTElectricity comma separated file. We then transformed the dataset dividing into 1hour 30 min data, and we had to predict for the 1 hour 45th minute usage. Dividing the data into 7 parts, made the total of 22 thousand plus rows, which became difficult to both train on an SVR model and the Neural Network model, therefore, as a limitation of that, we took about 1000 rows of data, which is 11 days of data, and fed them to all our models.

4.1 Linear Regression

```
> # Linear regression
> glmFitTime <- train(X7 ~ .,
+                     data = data,
+                     method = "glm",
+                     preProc = c("center", "scale"),
+                     tuneLength = 10,
+                     trControl = myCvControl, na.action=na.exclude)
> glmFitTime
Generalized Linear Model

1000 samples
   6 predictor

Pre-processing: centered (6), scaled (6)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 899, 900, 900, 901, 899, 902, ...
Resampling results:

  RMSE      Rsquared   MAE
  47.60183  0.8553963  33.72856

> summary(glmFitTime)

Call:
NULL

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-130.59   -27.59    -7.11    21.06   373.42

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  273.522      1.512 180.891  < 2e-16 ***
X1            -9.353      4.229  -2.212 0.027221 *
X2             2.017      4.913   0.411 0.681487
X3             9.259      4.920   1.882 0.060123 .
X4            17.743      4.670   3.800 0.000154 ***
X5            11.964      4.633   2.583 0.009947 **
X6            86.384      3.999  21.601  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 2286.385)

    Null deviance: 15674880  on 999  degrees of freedom
Residual deviance:  2270381  on 993  degrees of freedom
AIC: 10582

Number of Fisher Scoring iterations: 2

> y_hat = predict(glmFitTime, newdata = data_x)
> mean(100*abs(y_hat-y)/y)
[1] 14.17386
```

Figure 1 Linear Regression For 15 Minute Data Point

We got the MAPE of 14.17, which is a good score, that means it couldn't identify the trends. One reason of that could be, that we divided the 15 min data into 7 parts, rather than giving it either an hour of data, by cutting down into 4 parts.

Note: Because of Limited processing time, we couldn't train on the whole dataset and took only 1000 rows of data, which is about 11 days of data.

4.2 Support Vector Machine (Regression)

```
> # Support Vector Regression
> svmFitTime <- train(X7 ~ .,
+                     data = data,
+                     method = "svmRadial",
+                     preProc = c("center", "scale"),
+                     tuneLength = 10,
+                     trControl = myCvControl, na.action = na.exclude)
> svmFitTime
Support Vector Machines with Radial Basis Function Kernel

1000 samples
   6 predictor

Pre-processing: centered (6), scaled (6)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 900, 901, 898, 900, 900, 900, ...
Resampling results across tuning parameters:

  C       RMSE      Rsquared   MAE
    0.25  47.14834  0.8585722  33.28622
    0.50  46.57132  0.8612128  32.88418
    1.00  46.81267  0.8596271  33.09829
    2.00  47.47991  0.8555239  33.63067
    4.00  48.70789  0.8481561  34.34413
    8.00  50.60664  0.8367663  35.52806
   16.00  52.88319  0.8230141  37.18928
   32.00  55.61441  0.8063930  39.13885
   64.00  60.08027  0.7795103  41.86040
  128.00  65.92669  0.7443473  45.54679

Tuning parameter 'sigma' was held constant at a value of 0.6581754
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were sigma = 0.6581754 and C = 0.5.
> summary(svmFitTime)
Length  Class   Mode
     1   ksvm     S4
> y_hat = predict(svmFitTime, newdata = data_x)
> mean(100*abs(y_hat-y)/y)
[1] 11.51948
```

Figure 2 SVM For 15-Minute Data Point

We got a MAPE of about 11.51 after training the dataset having every 15 minutes of data, for about 11 days, which is better than that of the Linear Regression Model.

Here we used the svmRadial Kernel, which is used for regression, based on how far a single training example has influence on the training of the model. It finds a perfect fit, and creates different planes for every independent feature, and hence creates lots of dimensions. It is called a Radial Kernel, because it creates a radius of influence for the training set.

Note: Because of Limited processing time, we couldn't train on the whole dataset and took only 1000 rows of data, which is about 11 days of data.

4.3 Neural Networks

```
> # Neural Network
> nnFitTime <- train(X7 ~ .,
+                    data = data,
+                    method = "avNNet",
+                    preProc = c("center", "scale"),
+                    trControl = myCvControl,
+                    linout = T,
+                    trace = F,
+                    MaxNWts = 10 * (ncol(data) + 1) + 10 + 1,
+                    maxit = 500, na.action = na.exclude)
Warning message:
executing %dopar% sequentially: no parallel backend registered
> nnFitTime
Model Averaged Neural Network

1000 samples
   6 predictor

Pre-processing: centered (6), scaled (6)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 900, 901, 901, 899, 901, 899, ...
Resampling results across tuning parameters:

  size  decay  RMSE      Rsquared   MAE
  1     0e+00  55.41193  0.8173331  40.76109
  1     1e-04  53.42860  0.8246783  38.78595
  1     1e-01  48.20249  0.8511169  34.16244
  3     0e+00  46.66729  0.8593641  33.35620
  3     1e-04  46.70797  0.8596590  33.31082
  3     1e-01  45.31230  0.8677548  32.44115
  5     0e+00  46.96049  0.8580914  33.65643
  5     1e-04  46.13281  0.8629574  32.93909
  5     1e-01  45.28730  0.8679446  32.47903

Tuning parameter 'bag' was held constant at a value of FALSE
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were size = 5, decay = 0.1 and bag = FALSE.
> summary(nnFitTime)
            Length Class      Mode
model       5      -none-     list
repeats     1      -none-     numeric
bag         1      -none-     logical
seeds       5      -none-     numeric
names       6      -none-     character
terms       3      terms      call
coefnames   6      -none-     character
xlevels     0      -none-     list
xNames      6      -none-     character
problemType 1      -none-     character
tuneValue   3      data.frame list
obsLevels   1      -none-     logical
param       4      -none-     list
> y_hat = predict(nnFitTime, newdata = data_x)
> mean(100*abs(y_hat-y)/y)
[1] 12.39855
```

Figure 3 Neural Networks 15 Minute Data Points

MAPE for a neural network for regression, is 12.39 which is less than that of the SVR but is certainly more from the simple linear regression model.

4.4 ARIMA

```
> ar <- Arima(tSeries,order=c(7,0,7))
> summary(ar)
Series: tSeries
ARIMA(7,0,7) with non-zero mean

Coefficients:
         ar1     ar2      ar3     ar4     ar5     ar6      ar7      ma1      ma2     ma3      ma4      ma5      ma6
      0.7101  0.3025  -0.2538  0.3909  0.2568  0.2863  -0.7201  -0.0581  -0.2118  0.2377  -0.1698  -0.4330  -0.5315
s.e.  0.1268  0.1802   0.1618  0.1624  0.1855  0.1617   0.1096   0.1243   0.1150  0.1049   0.1200   0.1223   0.1209
         ma7      mean
      0.3311  264.9556
s.e.  0.0455    8.6764

sigma^2 estimated as 2001:  log likelihood=-5214.48
AIC=10460.96   AICc=10461.51   BIC=10539.48

Training set error measures:
                    ME     RMSE      MAE       MPE     MAPE     MASE        ACF1
Training set -0.417584 44.39163 29.97349 -5.013877 14.45029 0.595306 0.002287931
> #ar <- arima(tSeries,order=c(7,0,7))
> mean(100*abs(fitted(ar) - tSeries)/tSeries)
[1] 14.45029
```

Figure 4 ARIMA for 15 Minutes Data Points

Note: Because of Limited processing time, we couldn't train on the whole dataset and took only 1000 rows of data, which is about 11 days of data.

MAPE for the ARIMA Model, is 14.45 which is almost equivalent to the MAPE of Linear Regression Model, as the data here is not too much, therefore the time series model couldn't recognize the patterns in the dataset.

5. Prediction on Hourly data

We took the data by hour from the date timestamp, by adding a lag of 1 row to the data, as when we are looking at the dataset, it starts from 00:15 am, and when it goes to the 1:00 am, it is still counting for the same hour value, which is 00:45 to 1:00 am. Therefore, we added a lag in the dataset in order to get the exact value. After which we transformed it into 24 column dataset, where we predicted the 24th hour of the day, using 4 different models, mentioned below.

5.1 Linear Regression

```
X19          21.696      6.848    3.168 0.001565 **
X20          59.411      6.925    8.579  < 2e-16 ***
X21          31.816      6.250    5.091 4.04e-07 ***
X22          66.892      4.781   13.991  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 7935.477)

    Null deviance: 50314708  on 1457  degrees of freedom
Residual deviance: 11379474  on 1434  degrees of freedom
AIC: 17255

Number of Fisher Scoring iterations: 2

> y_hat = predict(glmFitTime, newdata = data_x)
> mean(100*abs(y_hat-y)/y)
[1] 13.25321
```

Figure 5 Linear Regression On 1 Hour Data Points

MAPE value for hourly data for Linear Regression model is 13.25, which is good according to the number of features given to it, which is 23.

5.2 Support Vector Machine (Regression)

```
Support Vector Machines with Radial Basis Function Kernel

1462 samples
  23 predictor

Pre-processing: centered (23), scaled (23)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 1314, 1313, 1311, 1313, 1313, 1312, ...
Resampling results across tuning parameters:

  C       RMSE       Rsquared   MAE
    0.25  112.05247  0.6432481  73.80703
    0.50  106.05570  0.6802870  72.41085
    1.00   98.36755  0.7241431  70.95279
    2.00   92.84058  0.7506658  69.97300
    4.00   93.21109  0.7464966  70.68681
    8.00   95.61555  0.7336492  72.46855
   16.00   98.87628  0.7171037  74.75066
   32.00  104.26334  0.6897857  78.98681
   64.00  109.22451  0.6651282  82.84287
  128.00  112.61380  0.6484894  85.52198

Tuning parameter 'sigma' was held constant at a value of 0.05159204
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were sigma = 0.05159204 and C =
> summary(svmFitTime)
Length  Class   Mode
     1   ksvm     S4
> y_hat = predict(svmFitTime, newdata = data_x)
> mean(100*abs(y_hat-y)/y)
[1] 13.31616
```

Figure 6 SVM For 1 Hour Data Points

MAPE value for hourly data for SVR model is 13.31, which is not better than that of Linear Regression model. One of the reasons for that could be, there are a lot of features, and hence a lot dimensions that had to be made by the SVR model.

5.3 Neural Network

```
Model Averaged Neural Network

1462 samples
  23 predictor

Pre-processing: centered (23), scaled (23)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 1313, 1313, 1313, 1312, 1311, 1312, ...
Resampling results across tuning parameters:

  size  decay  RMSE       Rsquared   MAE
  1     0e+00  140.92476  0.5073426  104.88291
  1     1e-04  132.90842  0.5390940   97.36162
  1     1e-01   97.29824  0.7349333   73.70113
  3     0e+00  121.33914  0.5994066   90.10151
  3     1e-04  117.57516  0.6239628   86.78688
  3     1e-01  105.60387  0.6890446   77.52565
  5     0e+00  110.43105  0.6628779   82.94550
  5     1e-04  107.05304  0.6844380   80.19765
  5     1e-01  101.44066  0.7094142   76.32592

Tuning parameter 'bag' was held constant at a value of FALSE
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were size = 1, decay = 0.1 and bag = FALSE.
> summary(nnFitTime)
            Length Class      Mode
model        5     -none-     list
repeats      1     -none-     numeric
bag          1     -none-     logical
seeds        5     -none-     numeric
names       23     -none-     character
terms        3     terms      call
coefnames   23     -none-     character
xlevels      0     -none-     list
xNames      23     -none-     character
problemType  1     -none-     character
tuneValue    3     data.frame list
obsLevels    1     -none-     logical
param        4     -none-     list
> y_hat = predict(nnFitTime, newdata = data_x)
> mean(100*abs(y_hat-y)/y)
[1] 13.27223
```

Figure 7 Neural Networks 1 hour Data Points

MAPE value for hourly data for Neural Network model is 13.27, which is better than that of SVR, but slightly worse than Linear Regression Model, because of a lot of features on which the model was trained.

5.4 ARIMA

```
Series: tSeries
ARIMA(7,0,7) with non-zero mean

Coefficients:
          ar1      ar2     ar3     ar4     ar5      ar6      ar7     ma1     ma2     ma3      ma4      ma5      ma6
      -0.3698  -0.0290  0.5695  0.7621  0.1592  -0.1878  -0.7872  1.5505  1.6324  0.9368  -0.0998  -0.6665  -0.7682
s.e.   0.0339   0.0306  0.0093  0.0038  0.0320   0.0312   0.0084  0.0341  0.0203  0.0216   0.0201   0.0309   0.0145
          ma7       mean
      -0.0932  1122.8860
s.e.   0.0084     3.5845

sigma^2 estimated as 28812:  log likelihood=-229763.7
AIC=459559.4   AICc=459559.4   BIC=459694.8

Training set error measures:
                     ME     RMSE      MAE       MPE     MAPE      MASE       ACF1
Training set 0.04800679 169.7044 131.0947 -3.728342 14.45686 0.6239156 0.00618209
> #ar <- arima(tSeries,order=c(7,0,7))
> mean(100*abs(fitted(ar) - tSeries)/tSeries)
```

Figure 8 ARIMA For 1 Hour Data Points

MAPE for a time series model for time series forecasting, is 14.45 which is the highest amongst all the models.

6. Prediction on daily data

Same as we added a lag to the 1 hour data, we had to do the same thing to the 1 day data, as value from 23:45 pm – 00:00 am should be counted in the same day, but when we separate the data using day function, it doesn't count that part in the same day. Therefore, we created a lag to get the exact data.

6.1 Linear Regression

```
> # Linear regression
> glmFitTime <- train(X7 ~ .,
+                     data = data,
+                     method = "glm",
+                     preProc = c("center", "scale"),
+                     tuneLength = 10,
+                     trControl = myCvControl, na.action=na.exclude)
> glmFitTime
Generalized Linear Model

208 samples
  6 predictor

Pre-processing: centered (6), scaled (6)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 187, 188, 187, 188, 186, 188, ...
Resampling results:

  RMSE      Rsquared   MAE
  2013.625  0.6186899  1509.088

> summary(glmFitTime)

Call:
NULL

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-6508.5  -1318.8   -135.8    886.2   7737.4

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  26834.2      141.4 189.803  < 2e-16 ***
X1            2592.0      202.5  12.799  < 2e-16 ***
X2           -1081.8      193.8  -5.583 7.60e-08 ***
X3             316.2      255.5   1.238   0.2172
X4             816.1      389.8   2.093   0.0376 *
X5            -865.7      396.1  -2.185   0.0300 *
X6            1154.4      201.0   5.742 3.41e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 4157495)

    Null deviance: 2134030384  on 207  degrees of freedom
Residual deviance:  835656494  on 201  degrees of freedom
AIC: 3769.2

Number of Fisher Scoring iterations: 2

> y_hat = predict(glmFitTime, newdata = data_x)
> mean(100*abs(y_hat-y)/y)
[1] 5.440118
```

Figure 9 Linear Regression On 1 Day Data Point

MAPE value for daily data for Linear Regression model is 5.44, which is good according to the number of features given to it, which is 7.

6.2 Support Vector Machine (Regression)

```
> # Support Vector Regression
> svmFitTime <- train(X7 ~ .,
+                     data = data,
+                     method = "svmRadial",
+                     preProc = c("center", "scale"),
+                     tuneLength = 10,
+                     trControl = myCvControl, na.action = na.exclude)
> svmFitTime
Support Vector Machines with Radial Basis Function Kernel

208 samples
  6 predictor

Pre-processing: centered (6), scaled (6)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 188, 186, 187, 187, 186, 188, ...
Resampling results across tuning parameters:

  C       RMSE      Rsquared   MAE
    0.25  2039.487  0.6192628  1444.152
    0.50  1964.127  0.6369677  1391.262
    1.00  1902.954  0.6550236  1358.171
    2.00  1897.208  0.6572796  1350.026
    4.00  1906.672  0.6551459  1358.264
    8.00  1990.518  0.6312533  1426.809
   16.00  2100.253  0.5995857  1505.020
   32.00  2233.013  0.5640628  1627.782
   64.00  2411.675  0.5192001  1772.256
  128.00  2609.556  0.4714447  1913.713

Tuning parameter 'sigma' was held constant at a value of 0.4568381
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were sigma = 0.4568381 and C = 2.
> summary(svmFitTime)
Length  Class   Mode
     1   ksvm     S4
> y_hat = predict(svmFitTime, newdata = data_x)
> mean(100*abs(y_hat-y)/y)
[1] 2.449999
```

Figure 10 SVM On 1 Day Data Point

MAPE value for daily data for Support Vector Regression model is 2.449, which is the best so far from all the datasets.

6.3 Neural Network

```
> # Neural Network
> nnFitTime <- train(X7 ~ .,
+                    data = data,
+                    method = "avNNet",
+                    preProc = c("center", "scale"),
+                    trControl = myCvControl,
+                    linout = T,
+                    trace = F,
+                    MaxNWts = 10 * (ncol(data) + 1) + 10 + 1,
+                    maxit = 500, na.action = na.exclude)
Warning message:
In nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,  :
  There were missing values in resampled performance measures.
> nnFitTime
Model Averaged Neural Network

208 samples
  6 predictor

Pre-processing: centered (6), scaled (6)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 187, 188, 185, 188, 188, 188, ...
Resampling results across tuning parameters:

  size  decay  RMSE      Rsquared   MAE
  1     0e+00  3016.424  0.3988102  2566.756
  1     1e-04  2829.187  0.5111862  2396.590
  1     1e-01  2139.590  0.6070790  1645.562
  3     0e+00  3089.132  0.2727500  2605.037
  3     1e-04  2875.604  0.3657369  2445.260
  3     1e-01  2012.971  0.6120715  1486.046
  5     0e+00  3000.741  0.2768715  2556.029
  5     1e-04  2872.443  0.3779841  2426.483
  5     1e-01  2023.150  0.6078886  1464.219

Tuning parameter 'bag' was held constant at a value of FALSE
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were size = 3, decay = 0.1 and bag = FALSE.
> summary(nnFitTime)
            Length Class      Mode
model       5      -none-     list
repeats     1      -none-     numeric
bag         1      -none-     logical
seeds       5      -none-     numeric
names       6      -none-     character
terms       3      terms      call
coefnames   6      -none-     character
xlevels     0      -none-     list
xNames      6      -none-     character
problemType 1      -none-     character
tuneValue   3      data.frame list
obsLevels   1      -none-     logical
param       4      -none-     list
> y_hat = predict(nnFitTime, newdata = data_x)
> mean(100*abs(y_hat-y)/y)
[1] 4.478118
```

Figure 11 Neural Network On 1 Day Data Points

MAPE for a neural network for regression, is 4.47 which is less than that of the SVR but is certainly more, but is good, and hence is able to read the underlying patterns.

6.4 ARIMA

```
> ar <- Arima(tSeries,order=c(7,0,7))
> summary(ar)
Series: tSeries
ARIMA(7,0,7) with non-zero mean

Coefficients:
          ar1      ar2      ar3      ar4      ar5      ar6     ar7     ma1     ma2     ma3     ma4     ma5     ma6
      -0.0881  -0.0884  -0.0878  -0.0887  -0.0884  -0.0879  0.9113  0.5242  0.5032  0.5079  0.5052  0.5082  0.5038
s.e.   0.0204   0.0204   0.0204   0.0204   0.0204   0.0205  0.0204  0.0570  0.0579  0.0573  0.0573  0.0572  0.0582
          ma7        mean
      -0.4801  26908.1331
s.e.   0.0573    301.5613

sigma^2 estimated as 4064820:  log likelihood=-13196.36
AIC=26424.72   AICc=26425.1   BIC=26509.31

Training set error measures:
                   ME     RMSE      MAE        MPE     MAPE      MASE      ACF1
Training set 10.72881 2005.763 1278.051 -0.9927072 5.354858 0.4306871 0.1050636
> #ar <- arima(tSeries,order=c(7,0,7))
> mean(100*abs(fitted(ar) - tSeries)/tSeries)
[1] 5.354858
```

Figure 12 ARIMA For Daily Consumption

MAPE for the time series model ARIMA is 5.35 which is very good, and we can clearly see that it is able to identify the underlying trends.

7. Comparison of Models

7.1 Fifteen Minute Data

```
> resamps <- resamples(list(lm = glmFitTime,
+                           svn = svmFitTime,
+                           nn = nnFitTime))
> summary(resamps)

Call:
summary.resamples(object = resamps)

Models: lm, svn, nn
Number of resamples: 50

MAE
        Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
lm  26.78157 31.62394 33.75444 33.72856 35.94578 39.18632    0
svn 28.28049 30.80126 32.54828 32.88418 34.95863 39.06701    0
nn  27.29848 30.95010 32.26975 32.47903 34.58279 37.26297    0

RMSE
        Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
lm  35.40301 43.24563 47.85732 47.60183 50.98489 63.10898    0
svn 35.64396 41.86451 44.53427 46.57132 51.19483 63.51883    0
nn  35.61694 41.26928 45.16549 45.28730 48.99396 61.30256    0

Rsquared
         Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
lm  0.7658718 0.8310063 0.8603724 0.8553963 0.8852730 0.9405901    0
svn 0.7625018 0.8302696 0.8732187 0.8612128 0.8925077 0.9272789    0
nn  0.7783622 0.8457673 0.8721244 0.8679446 0.8999302 0.9223511    0
```

Figure 13 Comparison of Error of different models on 15 mins data

7.2 Hourly Data

```
Call:
summary.resamples(object = resamps)

Models: lm, svn, nn
Number of resamples: 50

MAE
        Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
lm  60.58382 68.91600 70.44209 70.54778 73.25410 79.87565    0
svn 60.14922 66.78285 70.56537 69.97300 72.98424 79.77236    0
nn  65.69131 70.49549 73.53683 73.70113 76.80327 89.95452    0

RMSE
        Min.  1st Qu.   Median     Mean   3rd Qu.     Max. NA's
lm  79.81476 87.16004 90.11802 89.83922  92.57417 105.6415    0
svn 78.07697 88.05005 91.80100 92.84058  96.88953 122.8222    0
nn  83.47499 91.08637 93.80945 97.29824 101.92896 123.7647    0

Rsquared
         Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
lm  0.6257546 0.7241459 0.7611524 0.7607772 0.7888391 0.8738348    0
svn 0.6583645 0.7314730 0.7577652 0.7506658 0.7695518 0.8178474    0
nn  0.6475052 0.7078270 0.7369938 0.7349333 0.7611449 0.8412747    0
```

Figure 14 Comparison of Error of different models on Hourly Data

7.3 Daily Data

```
> # Compare models
> resamps <- resamples(list(lm = glmFitTime,
+                           svn = svmFitTime,
+                           nn = nnFitTime))
> summary(resamps)

Call:
summary.resamples(object = resamps)

Models: lm, svn, nn
Number of resamples: 50

MAE
        Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
lm  990.9301 1176.825 1464.296 1509.088 1835.965 2285.784    0
svn 908.0529 1127.614 1331.446 1350.026 1482.922 1939.846    0
nn  979.9995 1267.640 1474.850 1486.046 1681.158 2228.126    0

RMSE
        Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
lm  1231.815 1635.855 1917.650 2013.625 2403.501 3392.018    0
svn 1140.361 1582.753 1889.494 1897.208 2177.172 2712.915    0
nn  1262.366 1715.505 1958.386 2012.971 2271.414 3093.972    0

Rsquared
         Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
lm  0.1862563 0.4817656 0.6314426 0.6186899 0.7621157 0.9236415    0
svn 0.2876720 0.5319013 0.6702400 0.6572796 0.7787421 0.8814308    0
nn  0.1820971 0.5229479 0.6668595 0.6120715 0.7383322 0.8555959    0
```

Figure 15 Comparison of Error Loss of different models of Daily Data

8. Time Series Analysis
8.1 Fifteen mins Data

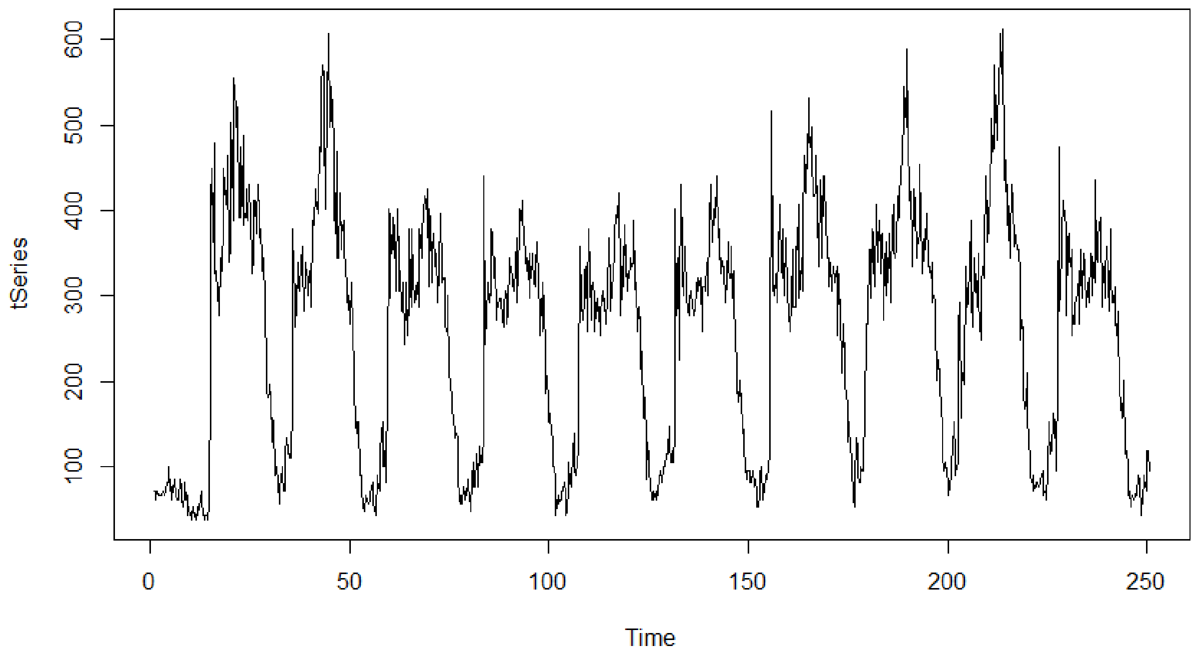


Figure 16 Time Series on 15 minute Data

There has been a lot of seasonality in the 15 mins data, for 11 days, as the electricity consumption changes from morning to evening and hence, we can see a repetetive pattern over the span of 11 days.

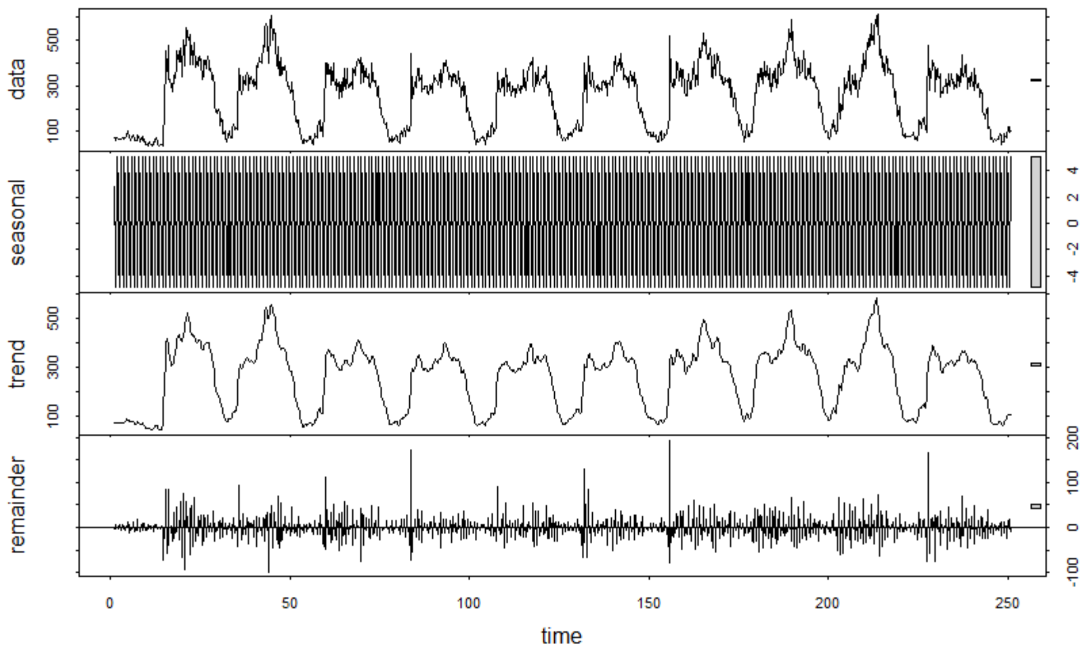


Figure 17 Seasonal, Trend, Remainder graph on 15 min data

As we can see the underlying trend from the data and trend graph, there was more usage of electricity on the starting two days, that were 1st Jan, 2011 and 2nd Jan, 2011, which later on was repeated after 4 days, with an increasing trend starting from the next Friday, going till the next Sunday. Hence we can see, how the Time Series is useful in understanding the trends and getting better insights.

8.2 Daily data

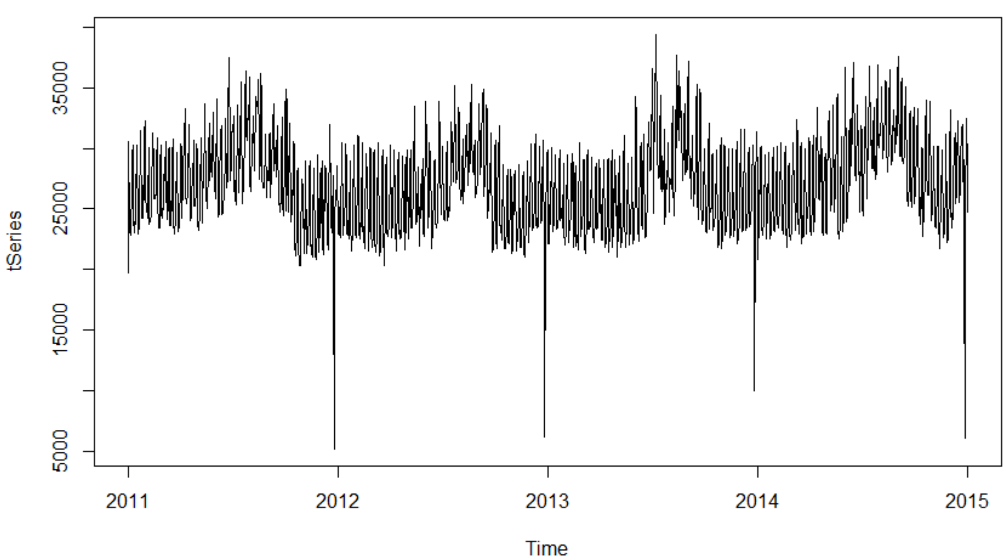


Figure 16 Time Series on Daily Data

From the 1 day data graph, we can see, there has been an increase in the consumption near the month October and November, which is consistent all through 4 years, and a drop in december every year, because of the holidays in that month.

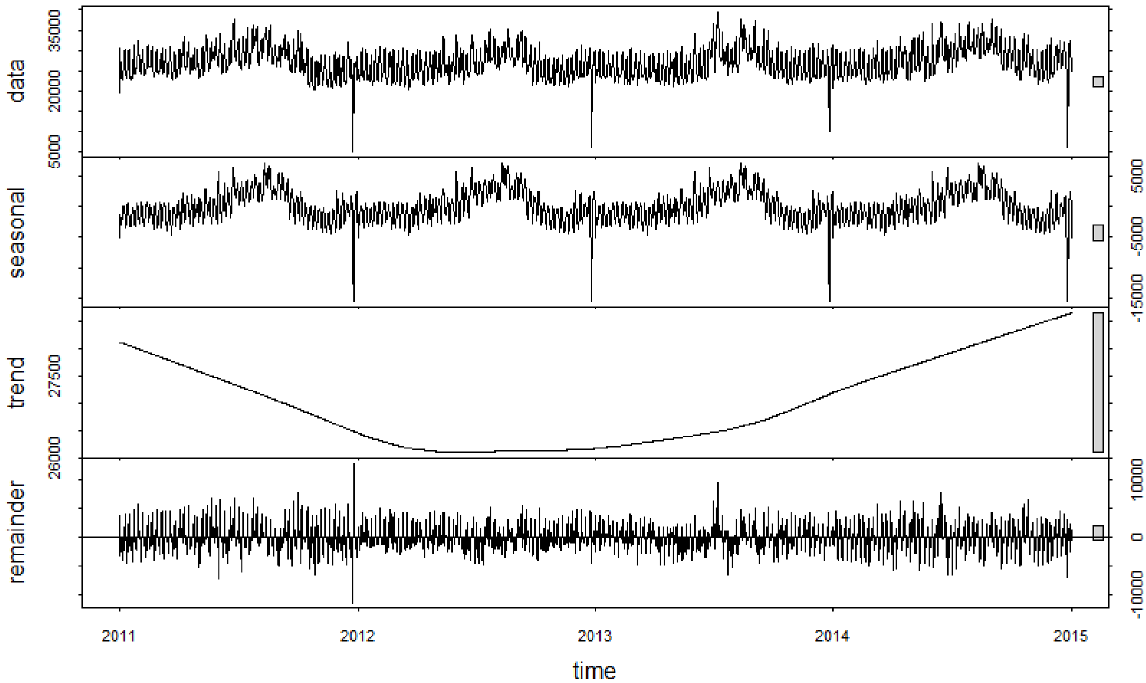


Figure 17 Seasonal, Trend, Remainder graph on daily data

As stated above, the time series is clearly able to identify the underlying trend, as we can see the seasonality in the graphs above.

8.3 Hourly data

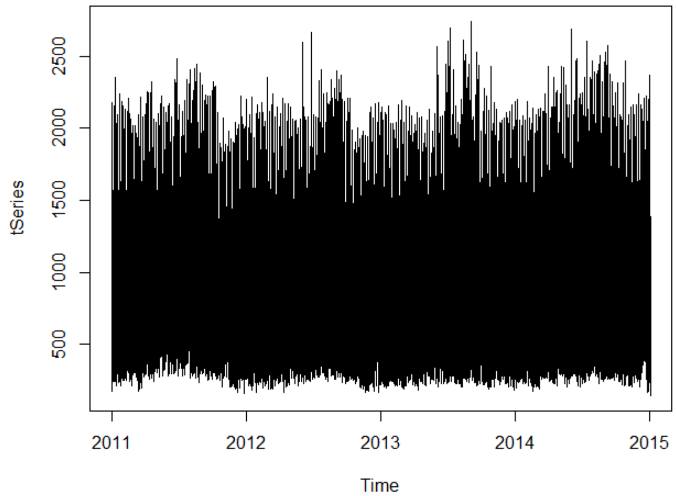
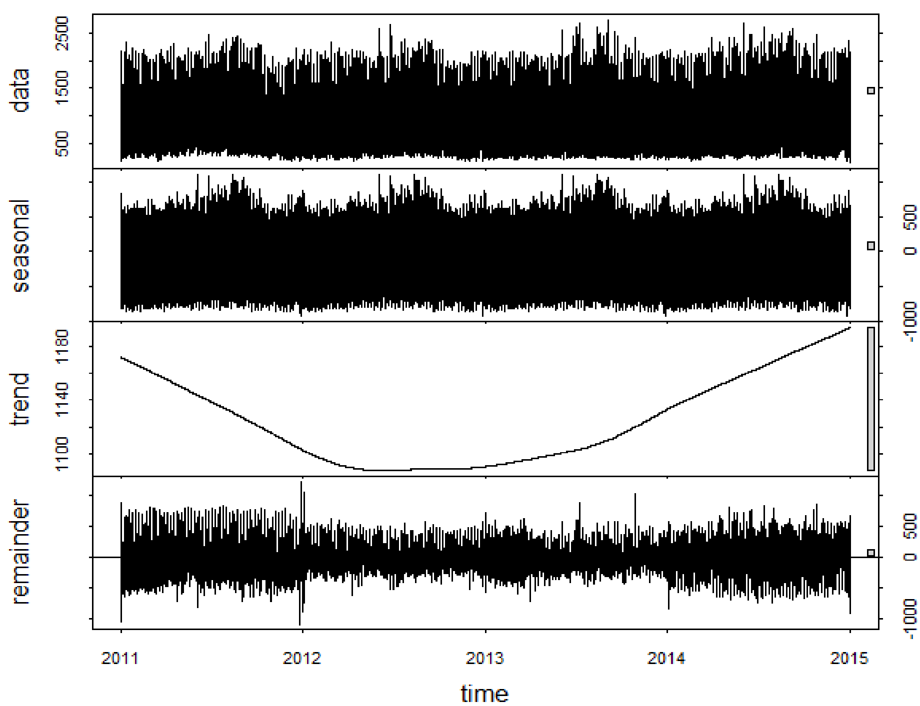


Figure 18 Hourly Data Time Series Trend

Since the chart is very cluttered, it is difficult to get an idea of what the trends are, but we could have get the exact trends, if we could do bootstrapping using the Bootstrap library, and used resampling in this case. Log wouldn't have worked since the data is not right skewed, if it would have been, we could have used that.



9. Conclusion:

Time/ MAPE	Linear Regression	SVM	Neural Networks	Arima	Best Model
15 Minutes	14.17	11.51	12.39	14.45	SVM
1 Hour	13.25	13.31	13.27	14.45	Linear Regression
1 Day	5.44	2.44	4.47	5.35	SVM

According to this table, we can see, that SVR (Regressor of Support Vector Machines) is the best model, which uses the Radial Kernel, and it gives the least error amongst all the other models.

ARIMA is used to find the trend underlying the dataset, but in this case, we can see SVM (Regression) works the best.

10. References

[1] Scikit-learn.org. (2019). *RBF SVM parameters — scikit-learn 0.21.3 documentation*. [online] Available at: https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html [Accessed 28 Jul. 2019].

[2] v8.7, f. (2019). *forecast package* | *R Documentation*. [online] Rdocumentation.org. Available at: https://www.rdocumentation.org/packages/forecast/versions/8.7 [Accessed 30 Jul. 2019].

[3] GeeksforGeeks. (2019). *Python* | *Pandas DatetimeIndex.dayofyear - GeeksforGeeks*. [online] Available at: https://www.geeksforgeeks.org/python-pandas-datetimeindex-dayofyear/ [Accessed 30 Jul. 2019].

[4] Rdocumentation.org. (2019). *step function* | *R Documentation*. [online] Available at: https://www.rdocumentation.org/packages/stats/versions/3.6.1/topics/step [Accessed 29 Jul. 2019].

[5] Pandas.pydata.org. (2019). *pandas.DatetimeIndex.dayofweek — pandas 0.25.0 documentation*. [online] Available at: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DatetimeIndex.dayofweek.html [Accessed 27 Jul. 2019].

[6] Statsoft.com. (2019). *How To Identify Patterns in Time Series Data: Time Series Analysis*. [online] Available at: http://www.statsoft.com/Textbook/Time-Series-Analysis [Accessed 19 Jul. 2019].