|  |
| --- |
| Prediction |
| MCDA 5580 Data Mining Assignment 4 Report |

|  |
| --- |
|  |

****

|  |  |
| --- | --- |
| ­­ | |
| Contents  [Executive Summary 2](#_Toc15293307)  [Data Prepare 2](#_Toc15293308)  [Analysis 4](#_Toc15293309)  [**Linear Regression, Support Vector Machine and Neural Network** 4](#_Toc15293310)  [**Time Series** 11](#_Toc15293311)  [Comparation 13](#_Toc15293312)  [Conclusion 14](#_Toc15293313)  [Reference 16](#_Toc15293314)  [Appendix 17](#_Toc15293315)  [Captions 27](#_Toc15293316) | |

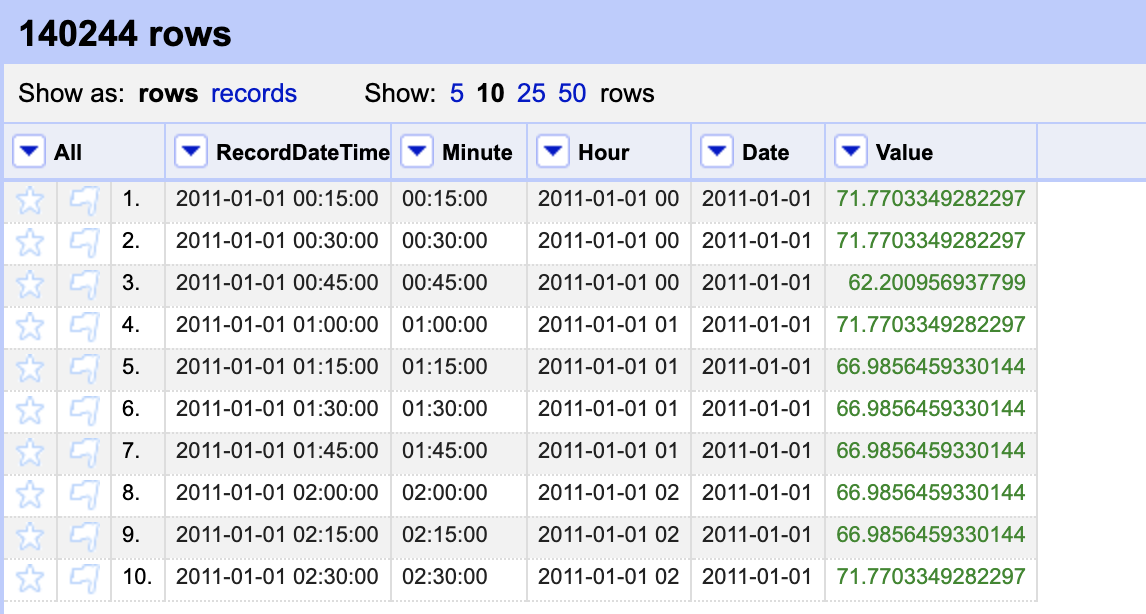
# Executive Summary

In this report, we will compare the three major prediction arithmetics: Linear Regression, Support Vector Machine and Neural Network. For data preparation, we use OpenRefine to create three columns which are minute, hour and date to store the minute, hour and date value for each usage reading.

# Data Prepare

Step.1

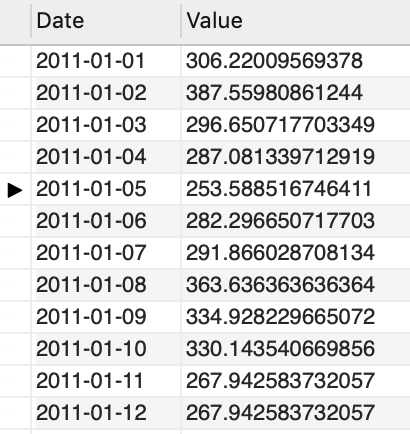
We add three columns based on date, hour and minute by using substring function in OpenRefine. By observation, there are 12 missing values. (140244-(365\*4+1)\*24\*4=12) So the dataset is not continuous and we can’t simply calculate the daily and hourly consumption.

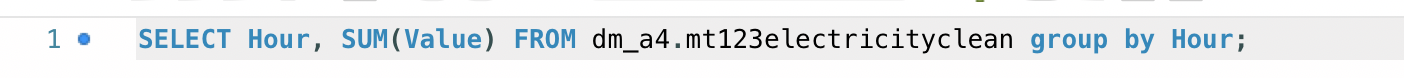


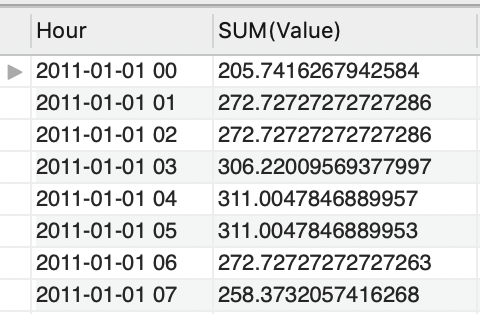
Step.2

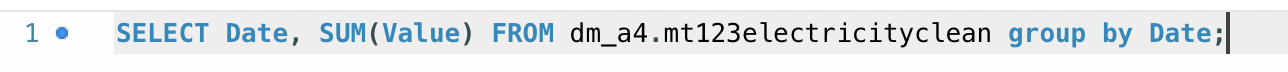
We use the following SQL queries to get three datasets. The first one is electricity consumption at 16:00 each day. The second and third are hourly and daily electricity consumption.

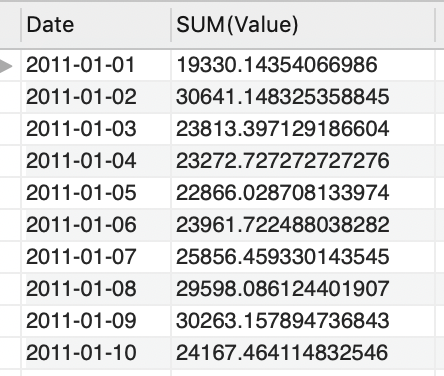






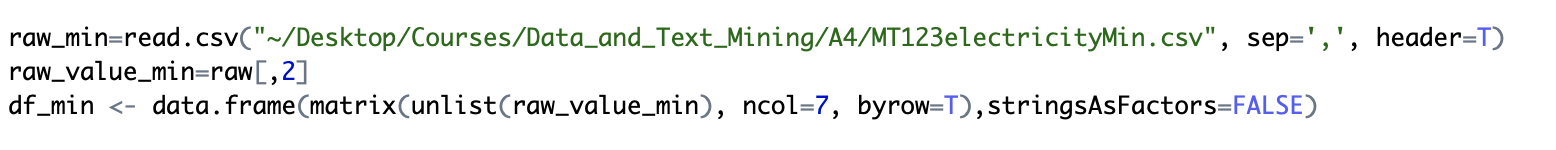


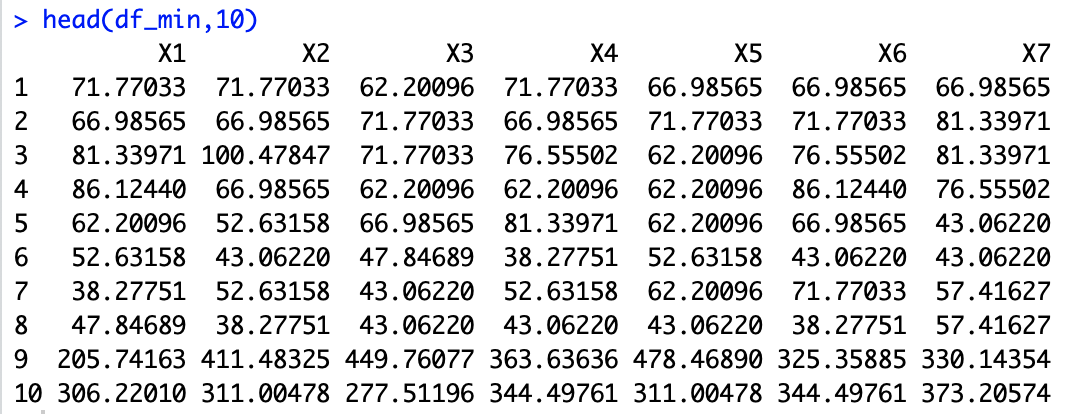




Step.3

We use the matrix function in R to reshape the data to a data frame with 7 or 24 columns. The number of columns is based on the consumption of periodic rules such as 7 days a week and 24 hours a day.



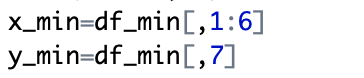


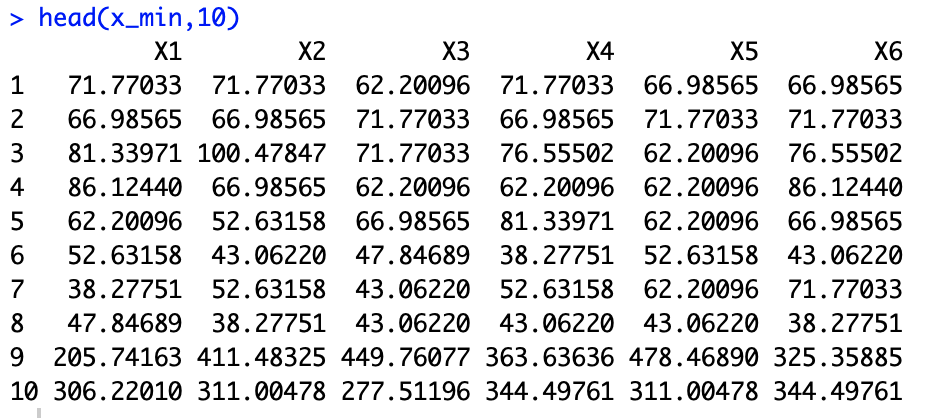
# Analysis

# **Linear Regression, Support Vector Machine and Neural Network**

Step.1

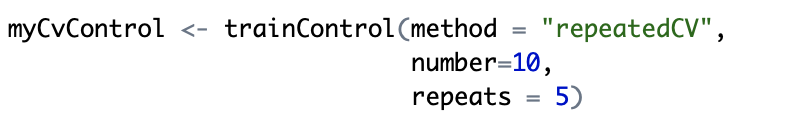
For the first analysis, we use the minute dataset for the electricity consumption of 16:00 every day. We choose the first six columns as inputs and the seventh column as output.





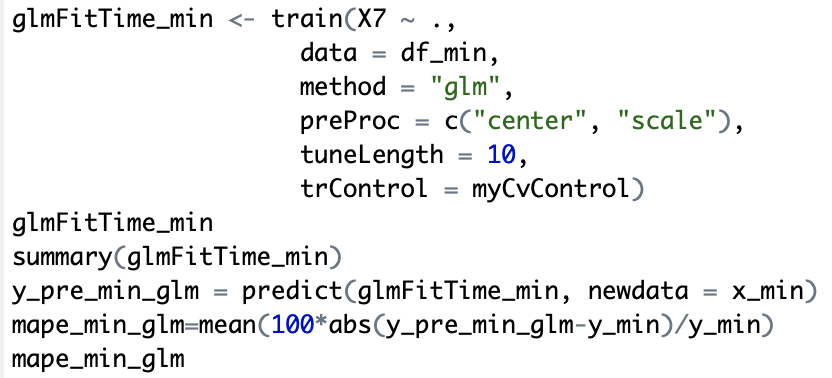
Step.2

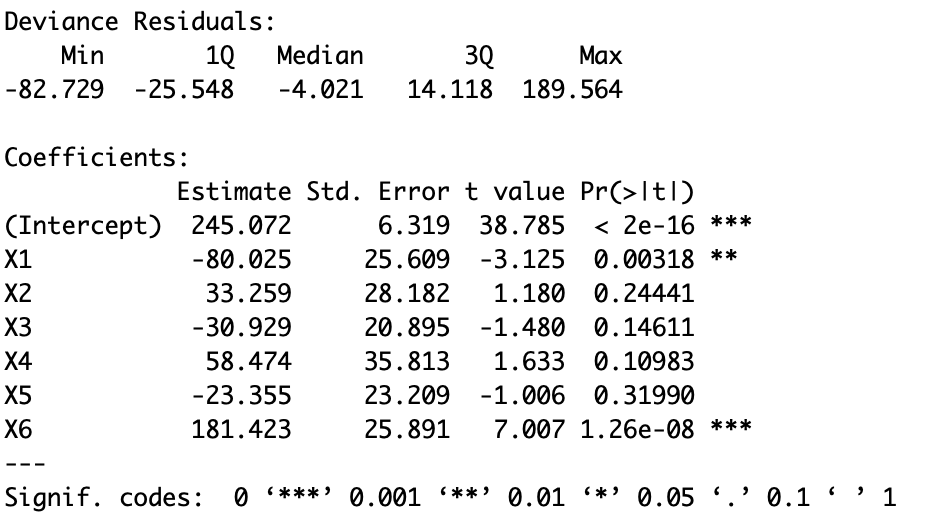
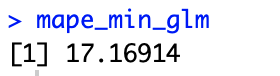
We set up the train control to cross-validate the model using 10 folds and 5 repeat times.



Step.3

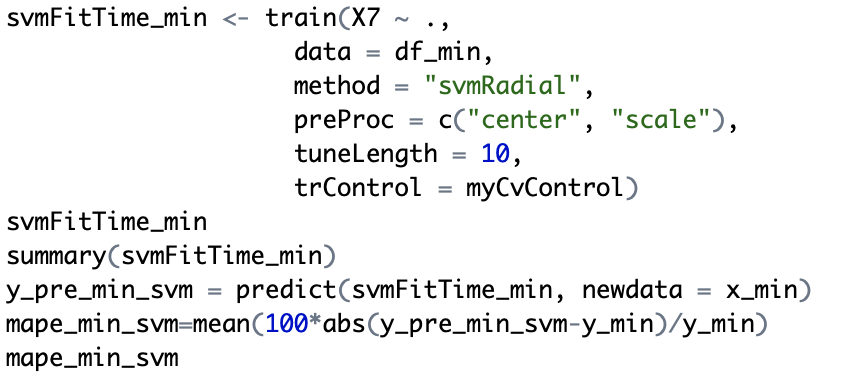
We use a linear regression to predict electricity consumption. The second, third, fourth and fifth columns are more significant in deciding the value of the seventh column. However, the MAPE value (mean absolute percentage error) is quite large in this model, around 17%.

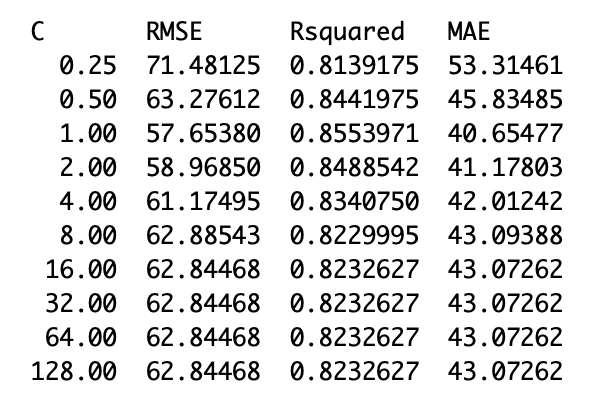
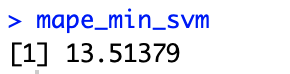


Step.4

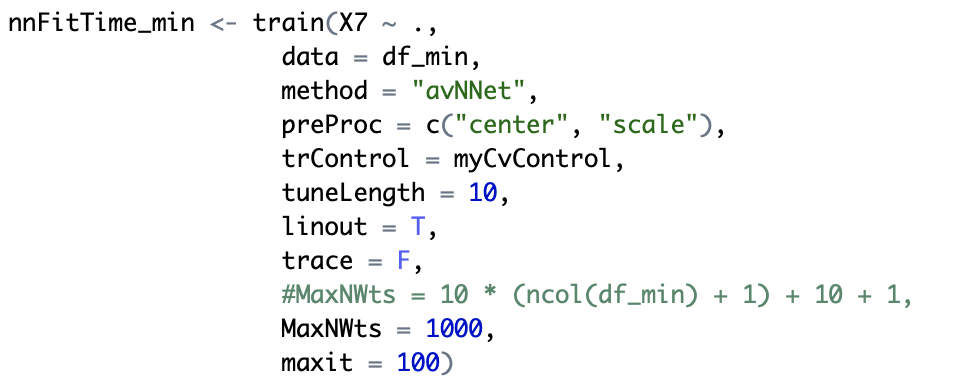
We use support vector machine to predict electricity consumption. In 10 fold of cross-validation, the RMSE values (root mean square error) are around 60 to 70 and the MAE values (mean absolute error) are around 40 to 50. The R-squared values are around 0.81 to 0.85, pretty close to 1. However, the MAPE value is around 13.5%, not accurate enough but better than a linear regression model.

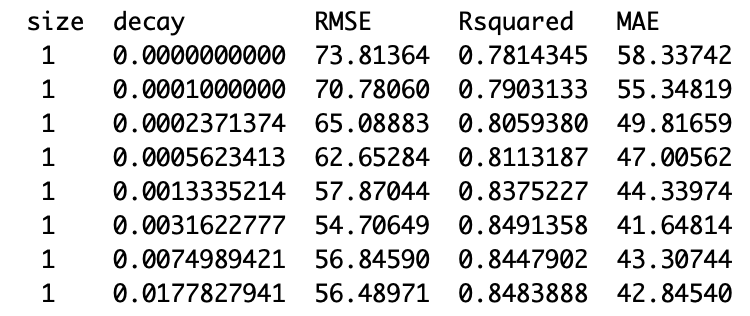
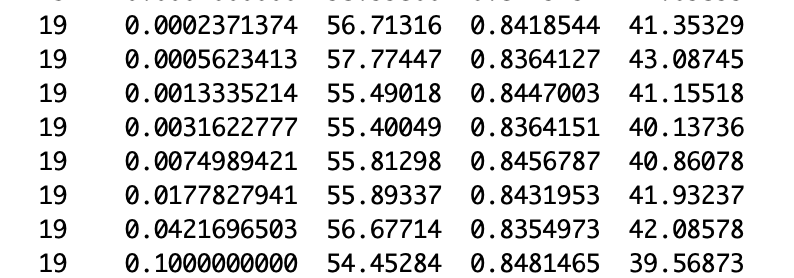


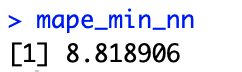
 

Step.4

We use a neural network to predict the electricity consumption with a large max number of weights value 1000 and max iteration of 100. From the start to end, the MAE value reduced from 58.34 to 39.57 and the R-squared value increased from 0.78 to 0.85. The RMSE value also reduced from 73.81 to 54.45. The MAPE value is 8.82%, significantly smaller than the previous two models.

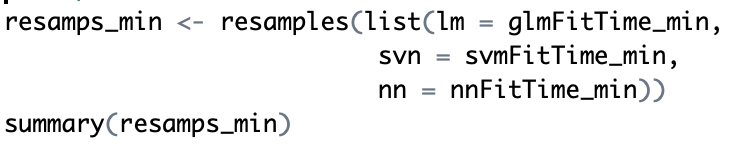


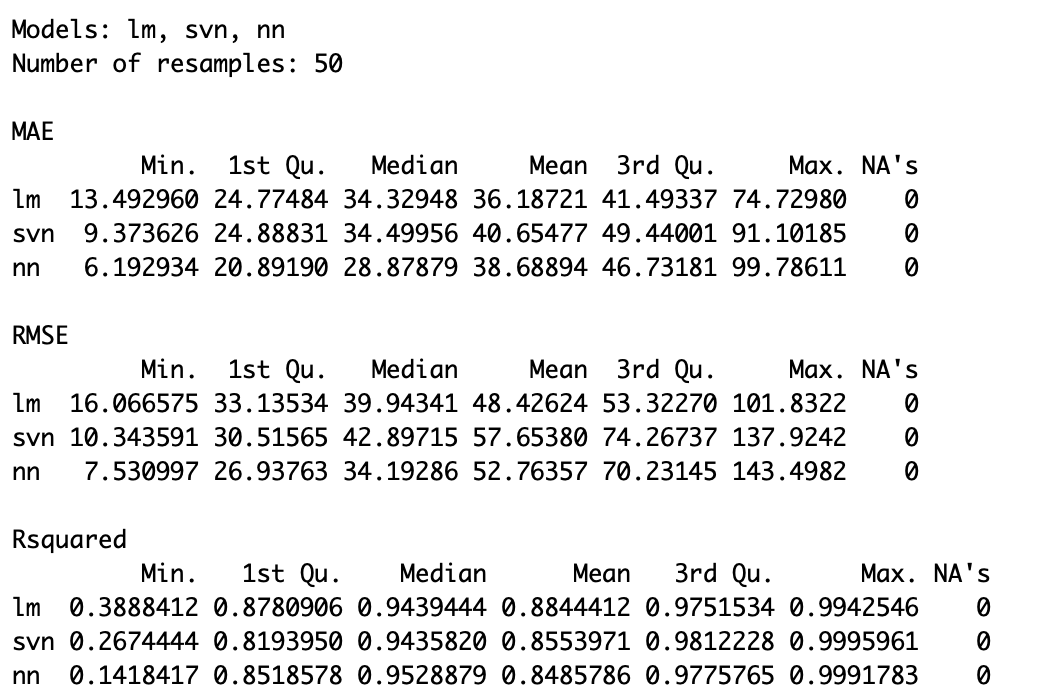
 



Step.5

We compare three models. The linear regression model has the best performance in MAE, RMSE and R-squared values. However, when it comes to the testing stage, the former two models perform much better than it and with better generalization performance. Because of the small datasets, the neural network model performs the best in predicting electricity consumption.

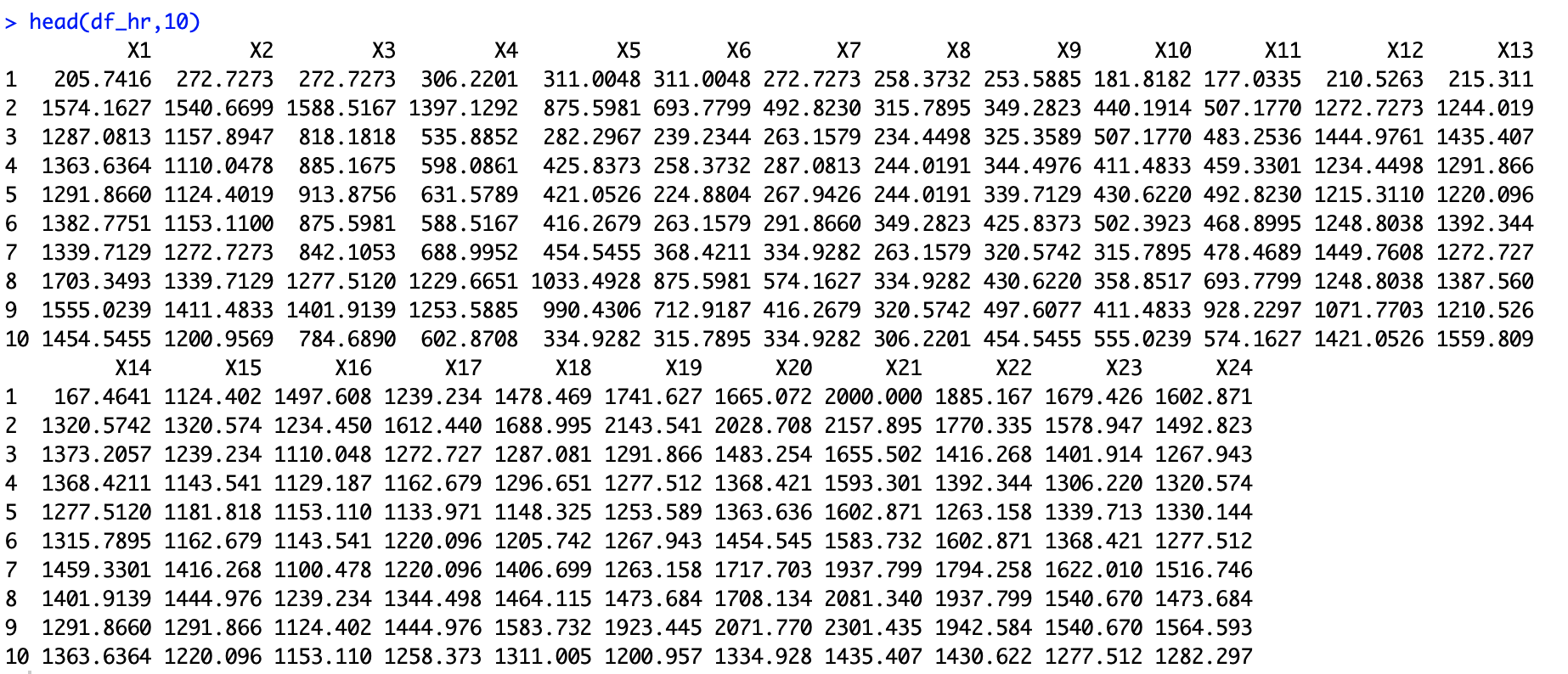




Step.6

We repeat the previous steps using hourly and daily datasets. The hourly dataset is reshaped into 24 columns because of 24 hours a day. The daily dataset is reshaped into 7 columns because of 7 days a week. The codes of this part are appended in the report and the results are shown below.

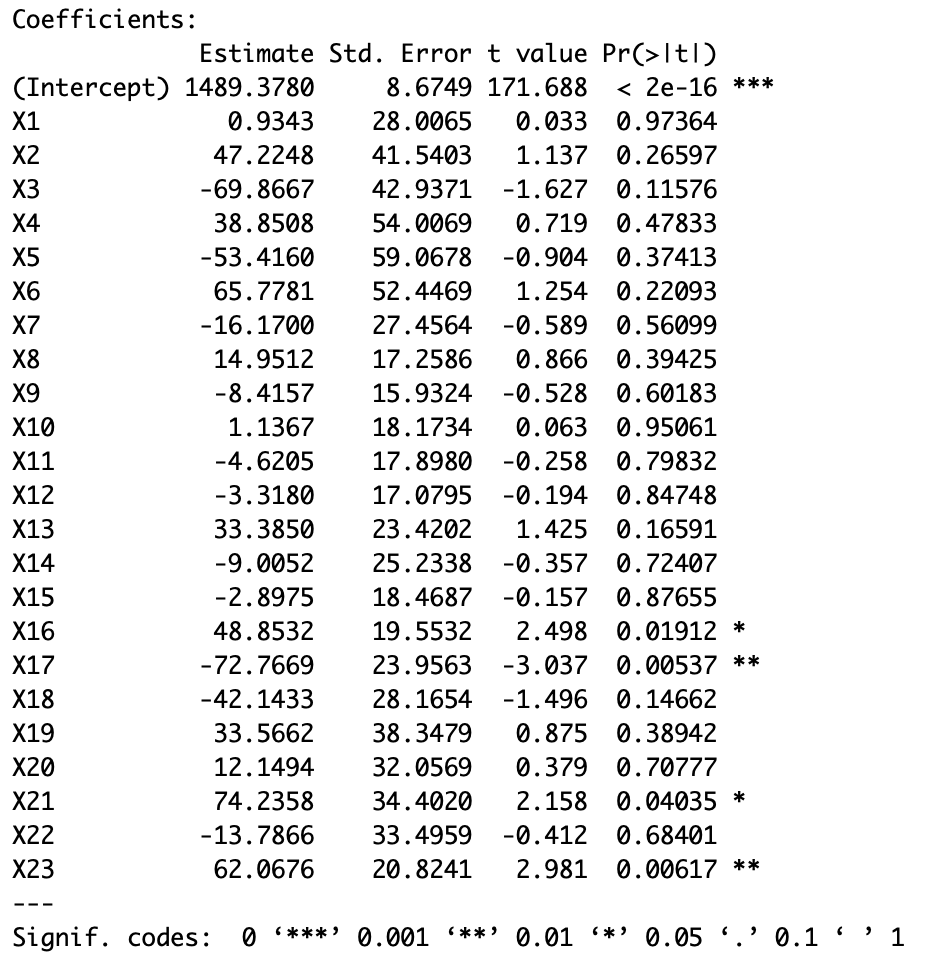
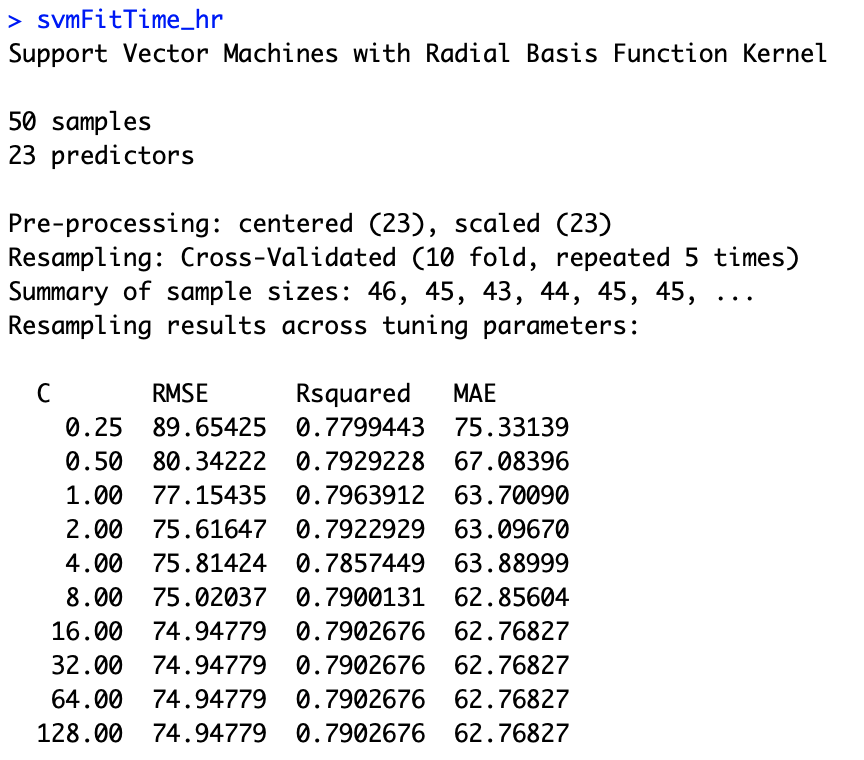


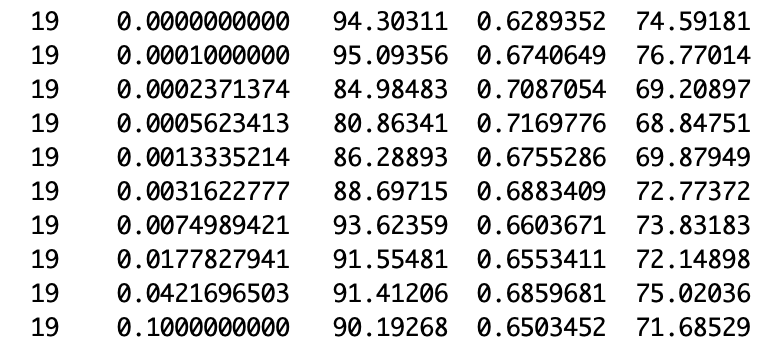
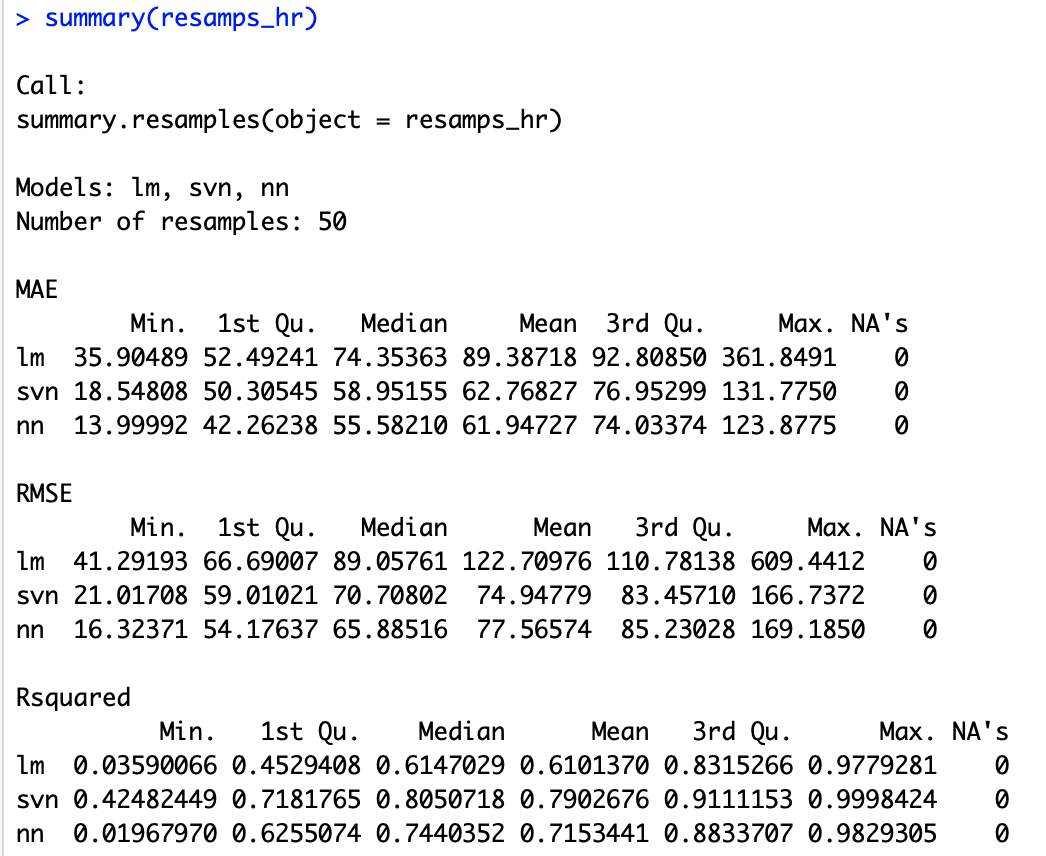




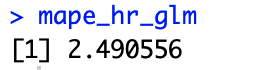
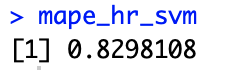
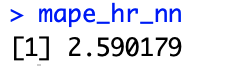


In the linear regression model, the 16th, 17th, 21st and 23rd columns are less significant in predicting hourly electricity consumption.

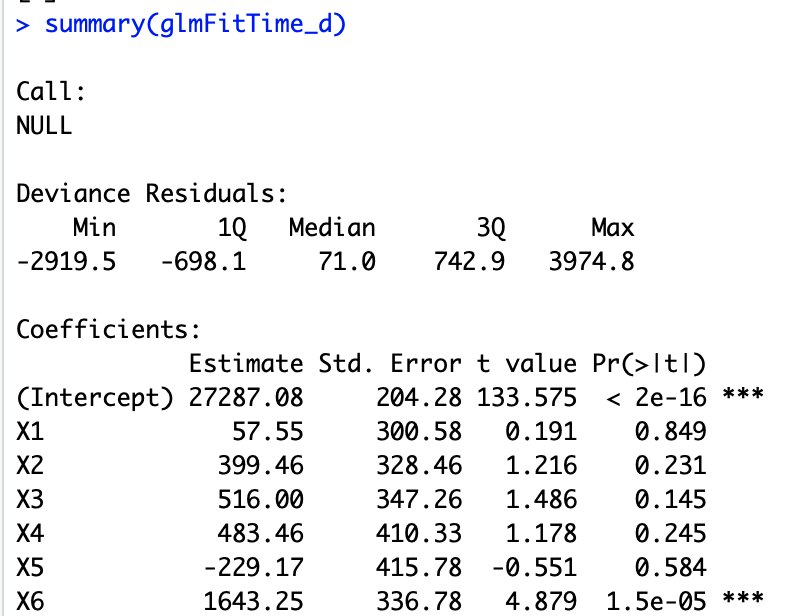
  

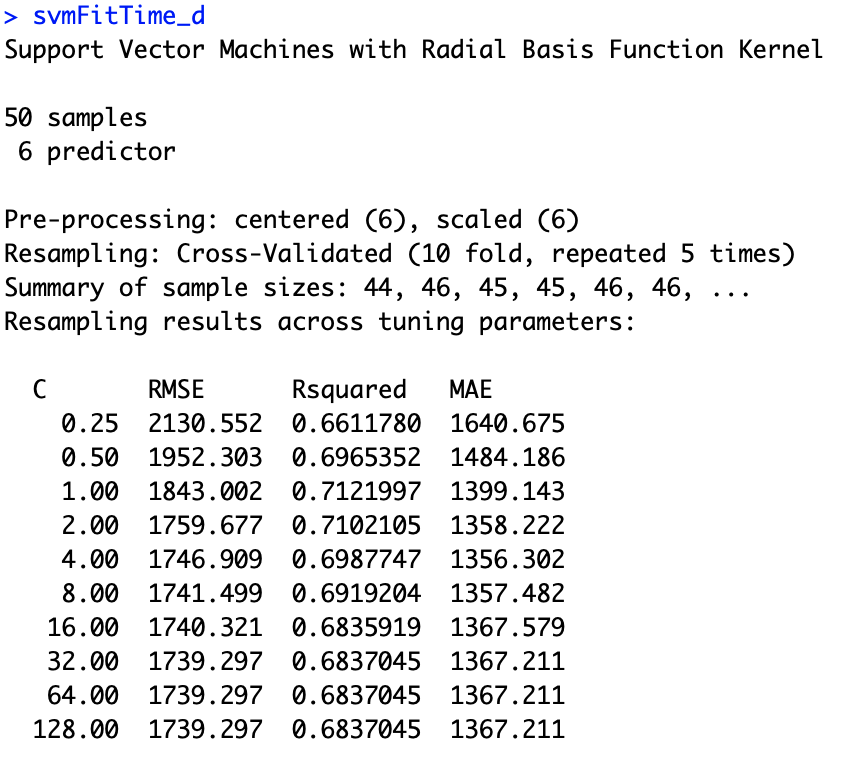
The MAPE value of three models in predicting hourly electricity consumption is 2.49%, 0.83% and 2.59%. The support vector machine model performs the best and we can see that with more inputs, the more accurate the models are.

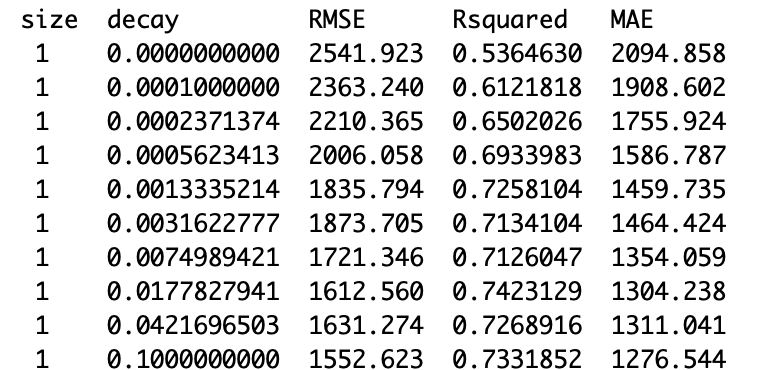
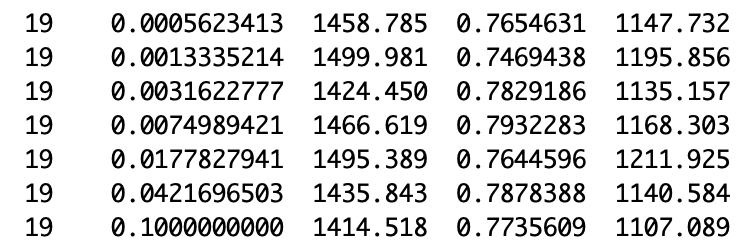
In a linear regression model of daily consumption, with the same number of input columns, the significant value is similar to the regression model of 15 minutes consumption.



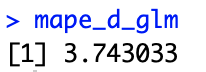
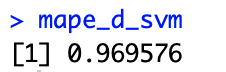
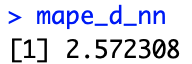
In support vector machine model of daily consumption, the R-squared values are also similar to the 15 minutes model, but more steady in performance.



Because of the limited iterations, the neural network model is not trained enough to perform better prediction than support vector machine model.

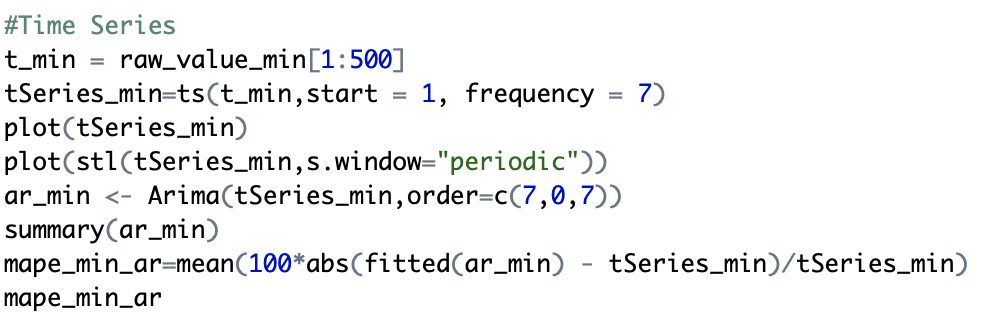
The MAPE values are as following:

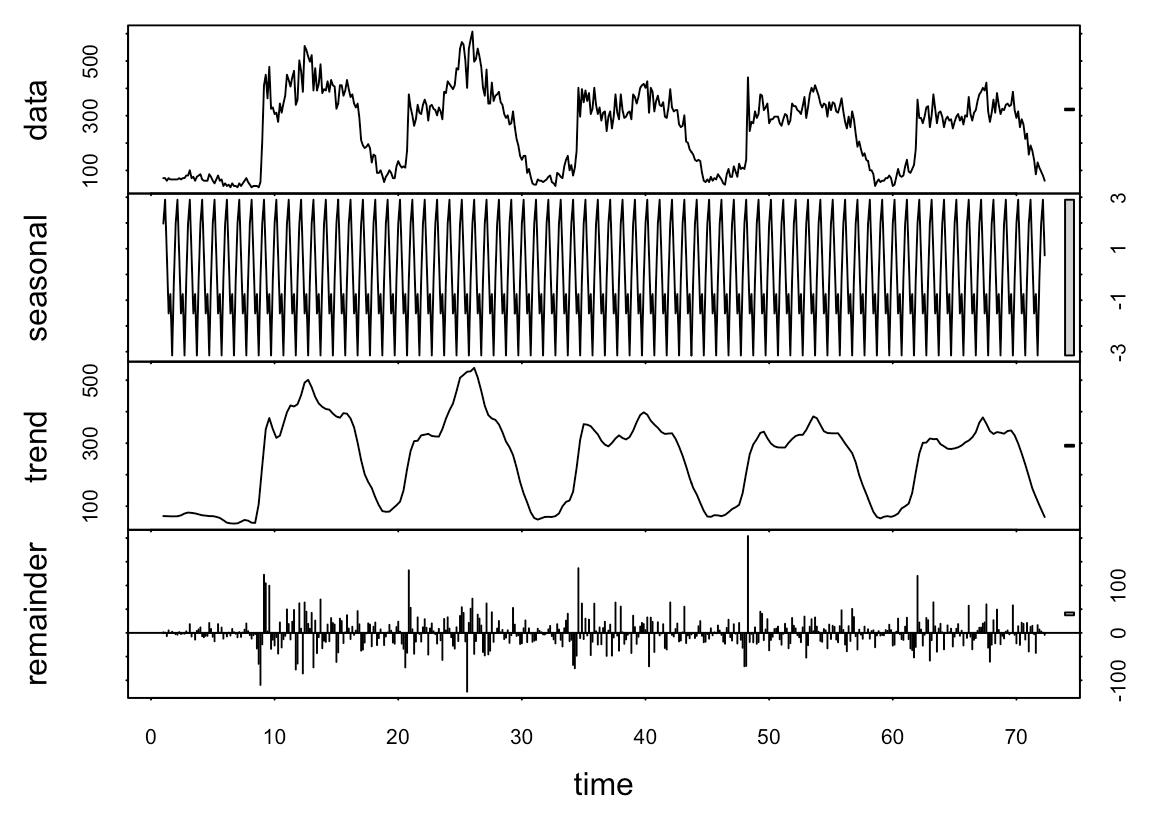
  

# **Time Series**

Step.1

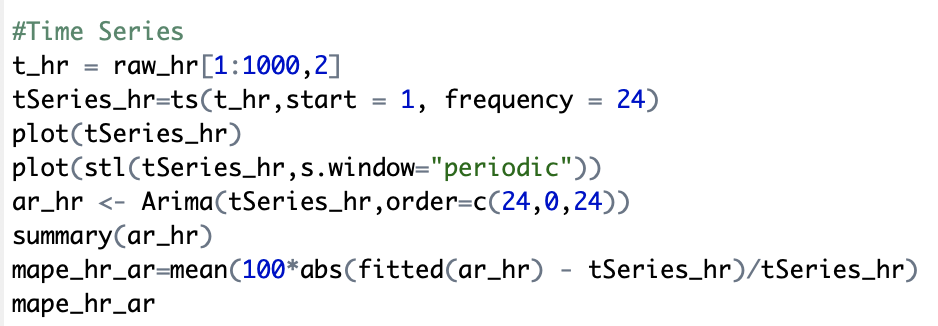
We use the first 500 rows of data to perform time series analysis on electricity consumption in the 15 minutes of 16:00 each day. We assume that there will be a pattern of 7 days a week and set the frequency, the lags of auto-regressive and the lags of moving the average to 7. We can see from the graph that by removing the seasonal data, the trend is still showing some periodic rule of 90 days. There may be another seasonal rule of three months. The MAPE value of ARIMA statistic model is 15.76%, slightly better than the linear regression statistic model.

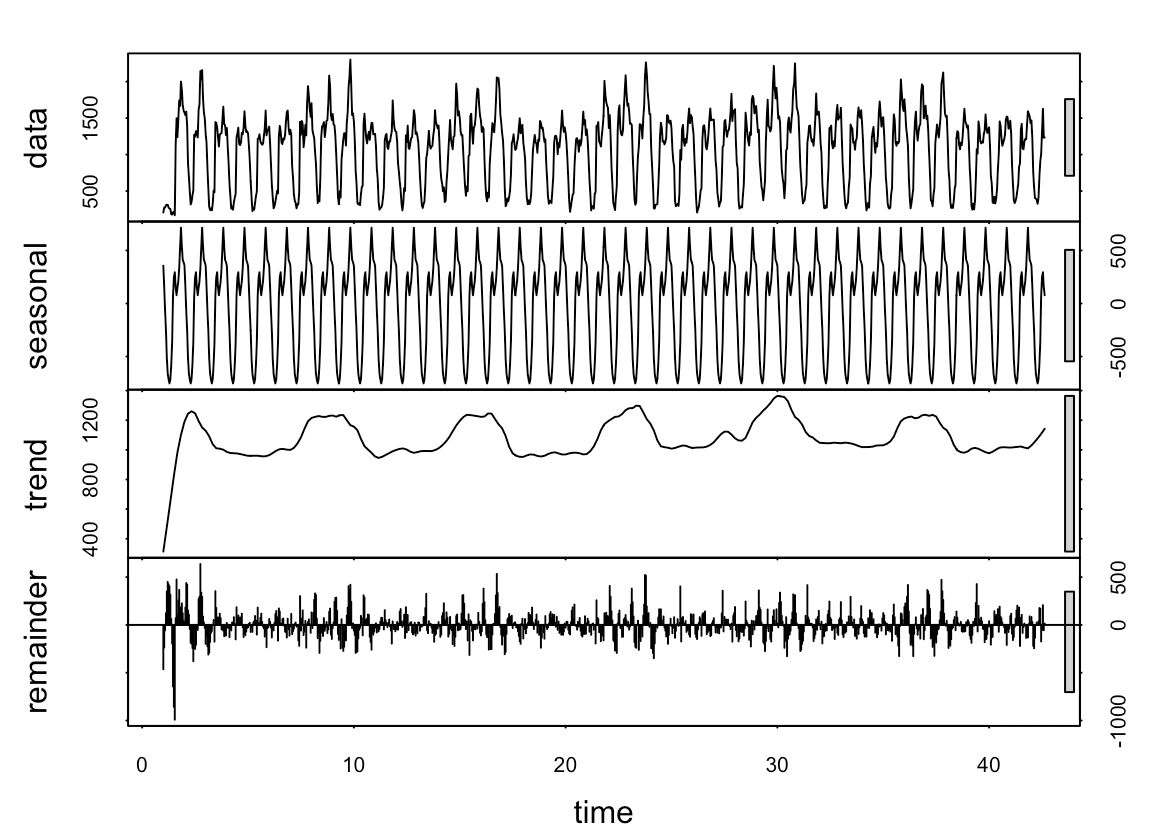
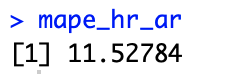


Step.2

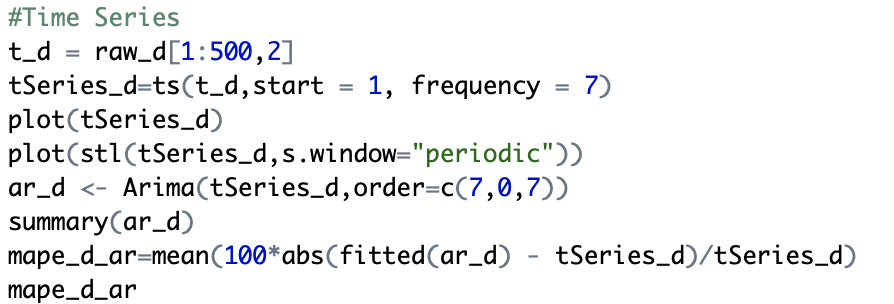
We use the first 1000 rows of hourly data to perform time series analysis. We assume that there will be a pattern of 24 hours a day and set the frequency, the lags of auto-regressive and the lags of moving the average to 24. We can see from the graph that the seasonal data is quite similar to the real data, so the consumption of the pattern is convincing. And we can still see a pattern of approximately 160 hours in trend data, so there might be another seasonal rule of a week. The MAPE value is 11.53%, significantly larger than the one from linear regression.

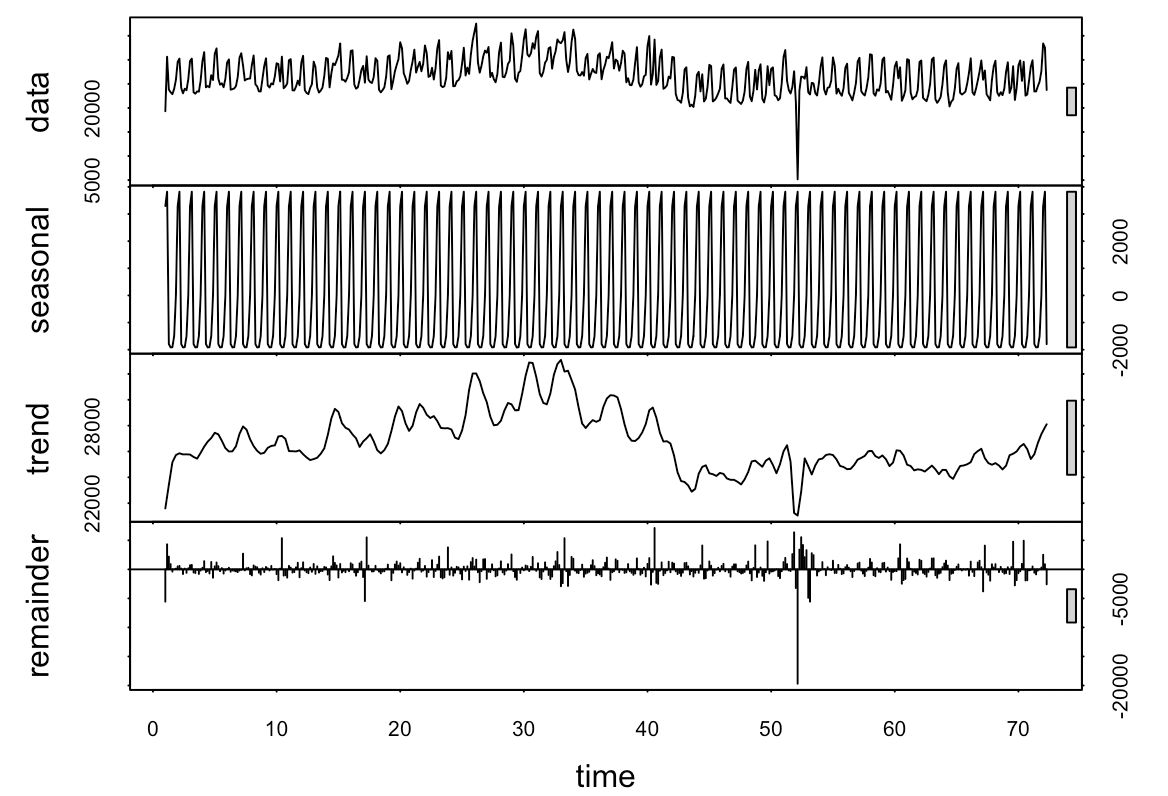
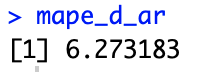


Step.3

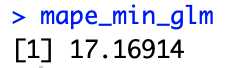
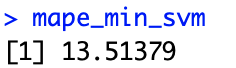
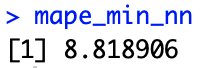
We use the first 500 rows of hourly data to perform time series analysis, same with the minutes one. We still assume that there will be a pattern of 7 days a week and set the frequency, the lags of auto-regressive and the lags of moving the average to 7. We can see from the graph that the seasonal data is quite similar to the trend we can see from the real data, so the consumption of the pattern is convincing. The MAPE value is 6.27%, slightly larger than the one from linear regression.

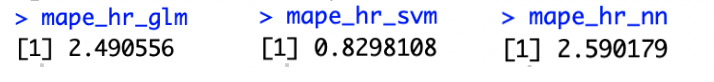


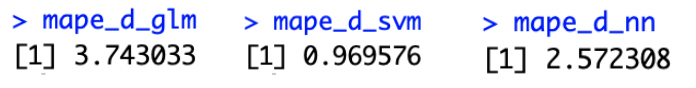
 

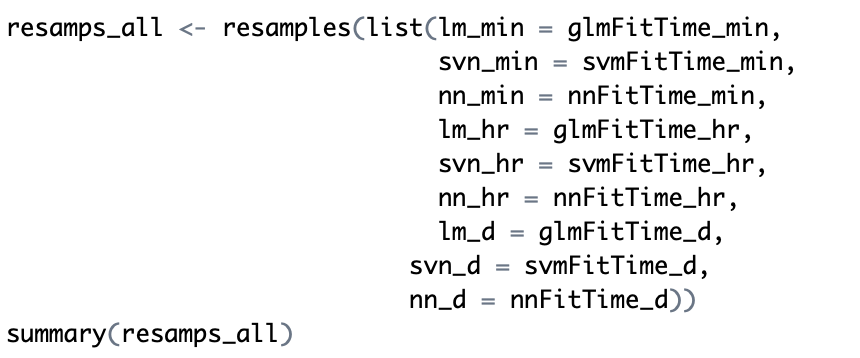
# Comparison

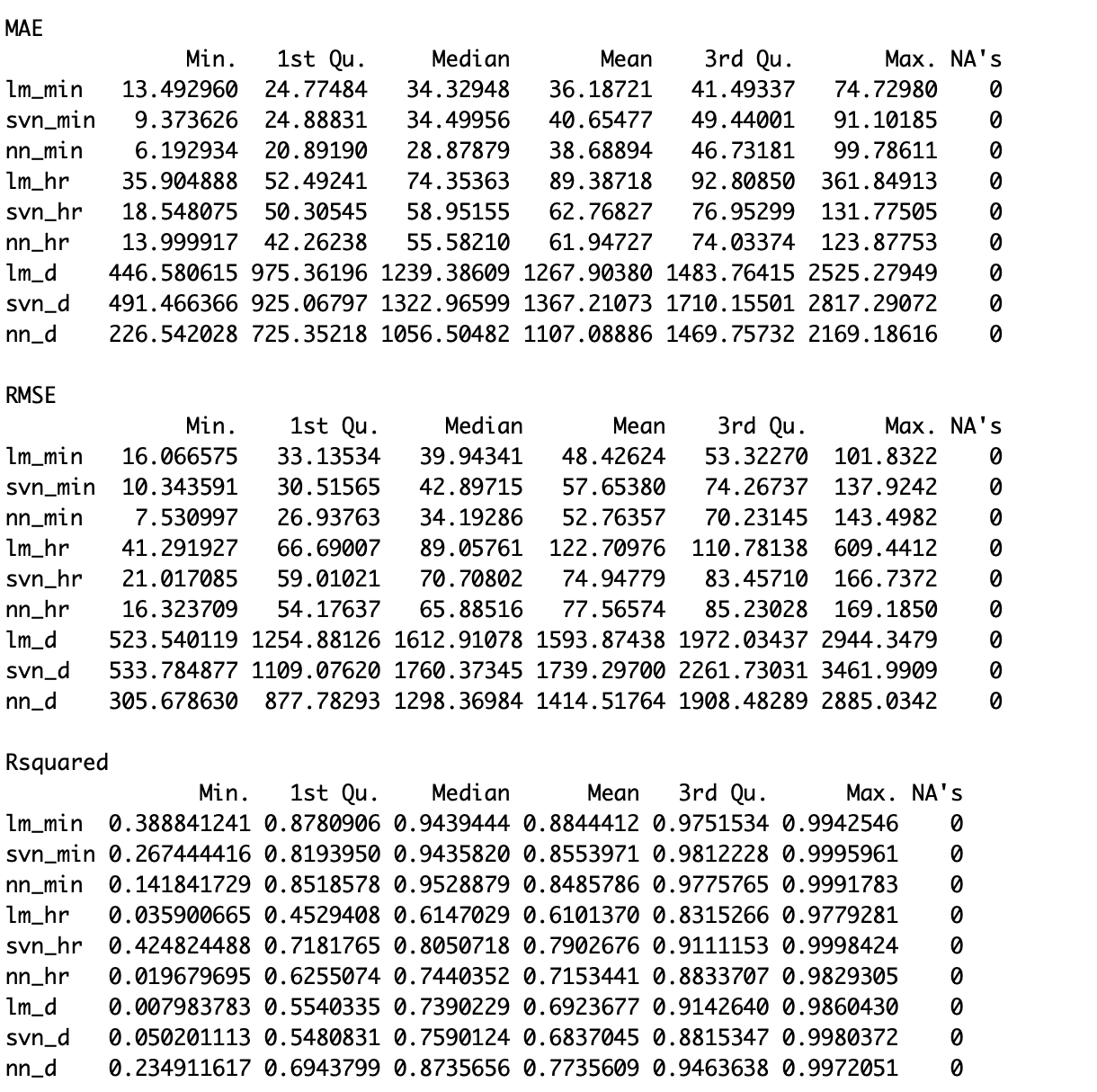
With small biased dataset (15 minutes datasets), the neural network can perform better prediction even with limited iterations. However, if the dataset is less biased, the support vector machine can perform considerably accurate prediction. Besides, it takes too much time to train the neural network model with more iterations and large datasets with a large number of input columns.









# Conclusion

We queried the database to retrieve the consumption on 16:00 each day, each hour and each day and import dataset into R. Matrix function in R have been used for reshape the dataset into a data frame with 7 or 24 columns and it is ready for further analysis. In the first run, we choose the consumption of 16:00 every day dataset. When we run the linear regression, the MAPE value is quite high around 17%. For the same dataset, when we run a support vector machine to predict the MAPE value is around 13.5%. When we run the neural network, we have a result of MAPE value around 8.82%. We repeated the three arithmetics with the hourly consumption the MAPE value for linear regression is 2.49%, for support vector machine is 0.83% and for the neural network is 2.59%. We repeated it again with the daily consumption dataset, the result of MAPE for linear regression is 3.74%, for support vector machine is 0.97% and for the neural network is 2.57%. The electric consumption partner meets our assumptions such as 24 hours a day, 7 days a week. When we remove the assumptions periotic rule to observing what left we still found another periotic rule in every 90 days.

# Reference

[1] *R Script Resource Package ‘RMySQL’* (July 15, 2019) available at

<https://smu.brightspace.com/d2l/le/content/68937/viewContent/504912/View>

# Appendix

**Data Preparation**

#15 min for 16:00 each day

raw\_min=read.csv("~/Desktop/Courses/Data\_and\_Text\_Mining/A4/MT123electricityMin.csv", sep=',', header=T)

head(raw\_min)

tail(raw\_min)

raw\_value\_min=raw[,2]

head(raw\_value\_min,10)

df\_min <- data.frame(matrix(unlist(raw\_value\_min), ncol=7, byrow=T),stringsAsFactors=FALSE)

df\_min = df\_min[1:50,]

head(df\_min,10)

x\_min=df\_min[,1:6]

y\_min=df\_min[,7]

head(x\_min,10)

#Hourly for each hour

raw\_hr=read.csv("~/Desktop/Courses/Data\_and\_Text\_Mining/A4/MT123electricityHour.csv", sep=',', header=T)

head(raw\_hr)

tail(raw\_hr)

raw\_value\_hr=raw\_hr[,2]

head(raw\_value\_hr,10)

df\_hr <- data.frame(matrix(unlist(raw\_value\_hr), ncol=24, byrow=T),stringsAsFactors=FALSE)

df\_hr = df\_hr[1:50,]

head(df\_hr,10)

x\_hr=df\_hr[,1:23]

y\_hr=df\_hr[,24]

#Daily for each day

raw\_d=read.csv("~/Desktop/Courses/Data\_and\_Text\_Mining/A4/MT123electricityDate.csv", sep=',', header=T)

head(raw\_d)

tail(raw\_d)

raw\_value\_d=raw\_d[,2]

head(raw\_value\_d,10)

df\_d <- data.frame(matrix(unlist(raw\_value\_d), ncol=7, byrow=T),stringsAsFactors=FALSE)

df\_d = df\_d[1:50,]

head(df\_d,10)

x\_d=df\_d[,1:6]

y\_d=df\_d[,7]

**Analysis**

library(caret)

library('tseries')

library("forecast")

#15 min for 16:00 each day

# Linear regression

glmFitTime\_min <- train(X7 ~ .,

data = df\_min,

method = "glm",

preProc = c("center", "scale"),

tuneLength = 10,

trControl = myCvControl)

glmFitTime\_min

summary(glmFitTime\_min)

y\_pre\_min\_glm = predict(glmFitTime\_min, newdata = x\_min)

mape\_min\_glm=mean(100\*abs(y\_pre\_min\_glm-y\_min)/y\_min)

mape\_min\_glm

# Support Vector Regression

svmFitTime\_min <- train(X7 ~ .,

data = df\_min,

method = "svmRadial",

preProc = c("center", "scale"),

tuneLength = 10,

trControl = myCvControl)

svmFitTime\_min

summary(svmFitTime\_min)

y\_pre\_min\_svm = predict(svmFitTime\_min, newdata = x\_min)

mape\_min\_svm=mean(100\*abs(y\_pre\_min\_svm-y\_min)/y\_min)

mape\_min\_svm

# Neural Network

nnFitTime\_min <- train(X7 ~ .,

data = df\_min,

method = "avNNet",

preProc = c("center", "scale"),

trControl = myCvControl,

tuneLength = 10,

linout = T,

trace = F,

#MaxNWts = 10 \* (ncol(df\_min) + 1) + 10 + 1,

MaxNWts = 1000,

maxit = 100)

nnFitTime\_min

summary(nnFitTime\_min)

y\_pre\_min\_nn = predict(nnFitTime\_min, newdata = x\_min)

mape\_min\_nn=mean(100\*abs(y\_pre\_min\_nn-y\_min)/y\_min)

mape\_min\_nn

# Compare models

resamps\_min <- resamples(list(lm = glmFitTime\_min,

svn = svmFitTime\_min,

nn = nnFitTime\_min))

summary(resamps\_min)

#Time Series

t\_min = raw\_value\_min[1:500]

tSeries\_min=ts(t\_min,start = 1, frequency = 7)

plot(tSeries\_min)

plot(stl(tSeries\_min,s.window="periodic"))

ar\_min <- Arima(tSeries\_min,order=c(7,0,7))

summary(ar\_min)

mape\_min\_ar=mean(100\*abs(fitted(ar\_min) - tSeries\_min)/tSeries\_min)

mape\_min\_ar

#Hourly for each hour

myCvControl <- trainControl(method = "repeatedCV",

number=10,

repeats = 5)

# Linear regression

glmFitTime\_hr <- train(X24 ~ .,

data = df\_hr,

method = "glm",

preProc = c("center", "scale"),

tuneLength = 10,

trControl = myCvControl)

glmFitTime\_hr

summary(glmFitTime\_hr)

y\_pre\_hr\_glm = predict(glmFitTime\_hr, newdata = x\_hr)

mape\_hr\_glm=mean(100\*abs(y\_pre\_hr\_glm-y\_hr)/y\_hr)

mape\_hr\_glm

# Support Vector Regression

svmFitTime\_hr <- train(X24 ~ .,

data = df\_hr,

method = "svmRadial",

preProc = c("center", "scale"),

tuneLength = 10,

trControl = myCvControl)

svmFitTime\_hr

summary(svmFitTime\_hr)

y\_pre\_hr\_svm = predict(svmFitTime\_hr, newdata = x\_hr)

mape\_hr\_svm=mean(100\*abs(y\_pre\_hr\_svm-y\_hr)/y\_hr)

mape\_hr\_svm

# Neural Network

nnFitTime\_hr <- train(X24 ~ .,

data = df\_hr,

method = "avNNet",

preProc = c("center", "scale"),

trControl = myCvControl,

tuneLength = 10,

linout = T,

trace = F,

#MaxNWts = 10 \* (ncol(df\_min) + 1) + 10 + 1,

MaxNWts = 1000,

maxit = 100)

nnFitTime\_hr

summary(nnFitTime\_hr)

y\_pre\_hr\_nn = predict(nnFitTime\_hr, newdata = x\_hr)

mape\_hr\_nn=mean(100\*abs(y\_pre\_hr\_nn-y\_hr)/y\_hr)

mape\_hr\_nn

# Compare models

resamps\_hr <- resamples(list(lm = glmFitTime\_hr,

svn = svmFitTime\_hr,

nn = nnFitTime\_hr))

summary(resamps\_hr)

#Time Series

t\_hr = raw\_hr[1:1000,2]

tSeries\_hr=ts(t\_hr,start = 1, frequency = 24)

plot(tSeries\_hr)

plot(stl(tSeries\_hr,s.window="periodic"))

ar\_hr <- Arima(tSeries\_hr,order=c(24,0,24))

summary(ar\_hr)

mape\_hr\_ar=mean(100\*abs(fitted(ar\_hr) - tSeries\_hr)/tSeries\_hr)

mape\_hr\_ar

#Daily for each day

# Linear regression

glmFitTime\_d <- train(X7 ~ .,

data = df\_d,

method = "glm",

preProc = c("center", "scale"),

tuneLength = 10,

trControl = myCvControl)

glmFitTime\_d

summary(glmFitTime\_d)

y\_pre\_d\_glm = predict(glmFitTime\_d, newdata = x\_d)

mape\_d\_glm=mean(100\*abs(y\_pre\_d\_glm-y\_d)/y\_d)

mape\_d\_glm

# Support Vector Regression

svmFitTime\_d <- train(X7 ~ .,

data = df\_d,

method = "svmRadial",

preProc = c("center", "scale"),

tuneLength = 10,

trControl = myCvControl)

svmFitTime\_d

summary(svmFitTime\_d)

y\_pre\_d\_svm = predict(svmFitTime\_d, newdata = x\_d)

mape\_d\_svm=mean(100\*abs(y\_pre\_d\_svm-y\_d)/y\_d)

mape\_d\_svm

# Neural Network

nnFitTime\_d <- train(X7 ~ .,

data = df\_d,

method = "avNNet",

preProc = c("center", "scale"),

trControl = myCvControl,

tuneLength = 10,

linout = T,

trace = F,

#MaxNWts = 10 \* (ncol(df\_min) + 1) + 10 + 1,

MaxNWts = 1000,

maxit = 100)

nnFitTime\_d

summary(nnFitTime\_d)

y\_pre\_d\_nn = predict(nnFitTime\_d, newdata = x\_d)

mape\_d\_nn=mean(100\*abs(y\_pre\_d\_nn-y\_d)/y\_d)

mape\_d\_nn

# Compare models

resamps\_d <- resamples(list(lm = glmFitTime\_d,

svn = svmFitTime\_d,

nn = nnFitTime\_d))

summary(resamps\_d)

resamps\_all <- resamples(list(lm\_min = glmFitTime\_min,

svn\_min = svmFitTime\_min,

nn\_min = nnFitTime\_min,

lm\_hr = glmFitTime\_hr,

svn\_hr = svmFitTime\_hr,

nn\_hr = nnFitTime\_hr,

lm\_d = glmFitTime\_d,

svn\_d = svmFitTime\_d,

nn\_d = nnFitTime\_d))

summary(resamps\_all)

#Time Series

t\_d = raw\_d[1:500,2]

tSeries\_d=ts(t\_d,start = 1, frequency = 7)

plot(tSeries\_d)

plot(stl(tSeries\_d,s.window="periodic"))

ar\_d <- Arima(tSeries\_d,order=c(7,0,7))

summary(ar\_d)

mape\_d\_ar=mean(100\*abs(fitted(ar\_d) - tSeries\_d)/tSeries\_d)

mape\_d\_ar