Exam - Time series forecasting

Khwansiri NINPAN

29/02/2020

Objective:

Forecast electricity consumption (kW) for 2/17/2010 with and without using outdoor temperature information

Data Files Information:

- 3 columns: Timestamp, Power(kW), Temps (C^o)
- 4603 observations for Timestamp and Temps (C^o):
  Information from 1/1/2010 1:15 to 2/17/2010 23:45
- 4507 observations for Power(kW):
  Information from 1/1/2010 1:15 to 2/16/2010 23:45

Part 1:

Forecast electricity consumption (kW) for 2/17/2010 without using outdoor temperature

Here, we will forecast electricity consumption on 2/17/2010 by using only previous informations of electricity consumption during 1/1/2010 to 2/16/2010.

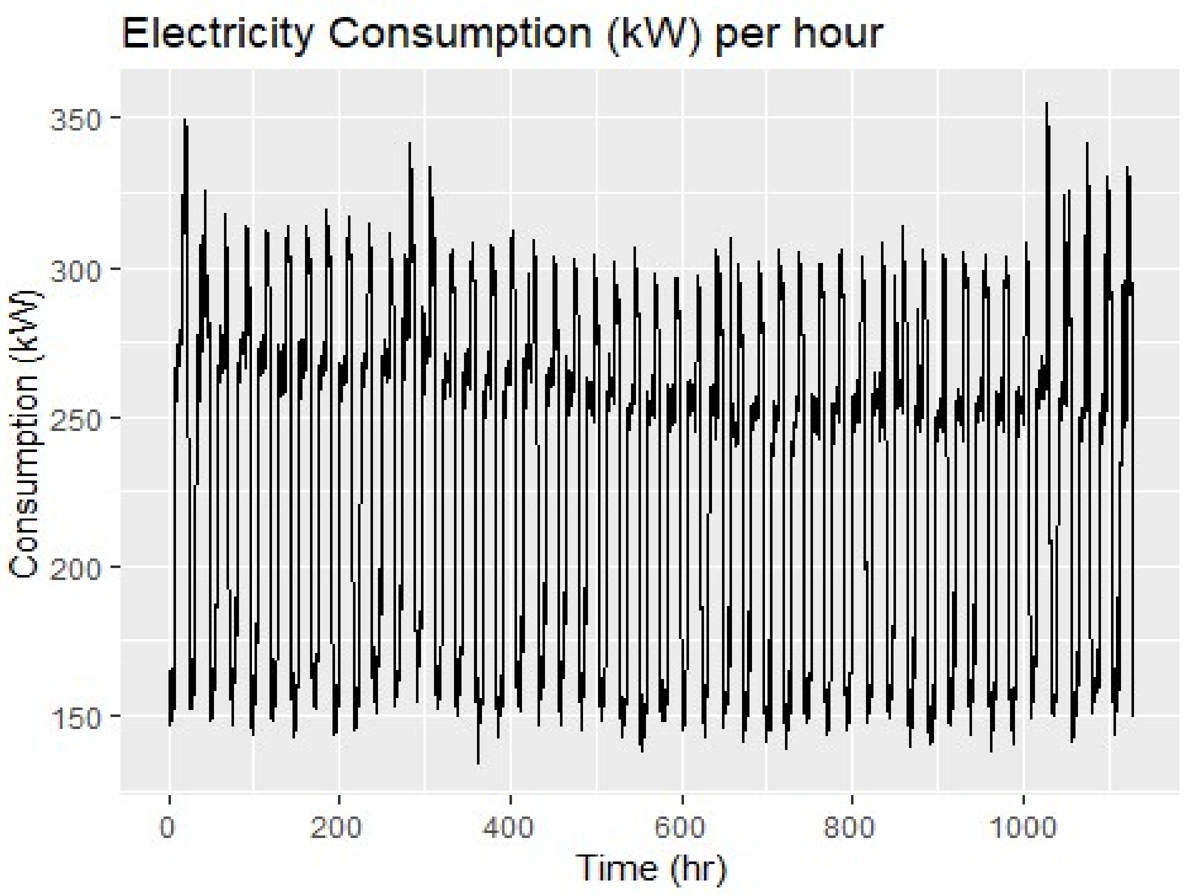
```
#Download data file
getwd()

## [1] "E:/DataScience/DSTI/TimeSeries/Exam/Final Report"

setwd("E:/DataScience/DSTI/TimeSeries/Exam")
data <- read.csv("TrainData.csv", stringsAsFactors=FALSE)
head(data, 5)   #Check data file

##    ï..Timestamp Power..kW. Temp..CÂ..
## 1 1/1/2010 1:15      165.1       10.6
## 2 1/1/2010 1:30      151.6       10.6
## 3 1/1/2010 1:45      146.9       10.6
## 4 1/1/2010 2:00      153.7       10.6
## 5 1/1/2010 2:15      153.8       10.6
```

Create time series objects with hourly seasonal pattern.
Since our data are observed every 15 minutes, to make an unit as hour, our frequency = 4 (60 minutes/15 minutes)
(Note: Information from 1/1/2010 to 2/16/2010 (Row 1-4508) for electricity consumption (Column 2))

```
consumption <- ts(data[1:4507,2], frequency = 4, start=c(1,2))
head(consumption)  #Check time series object

## [1] 165.1 151.6 146.9 153.7 153.8 159.0
```

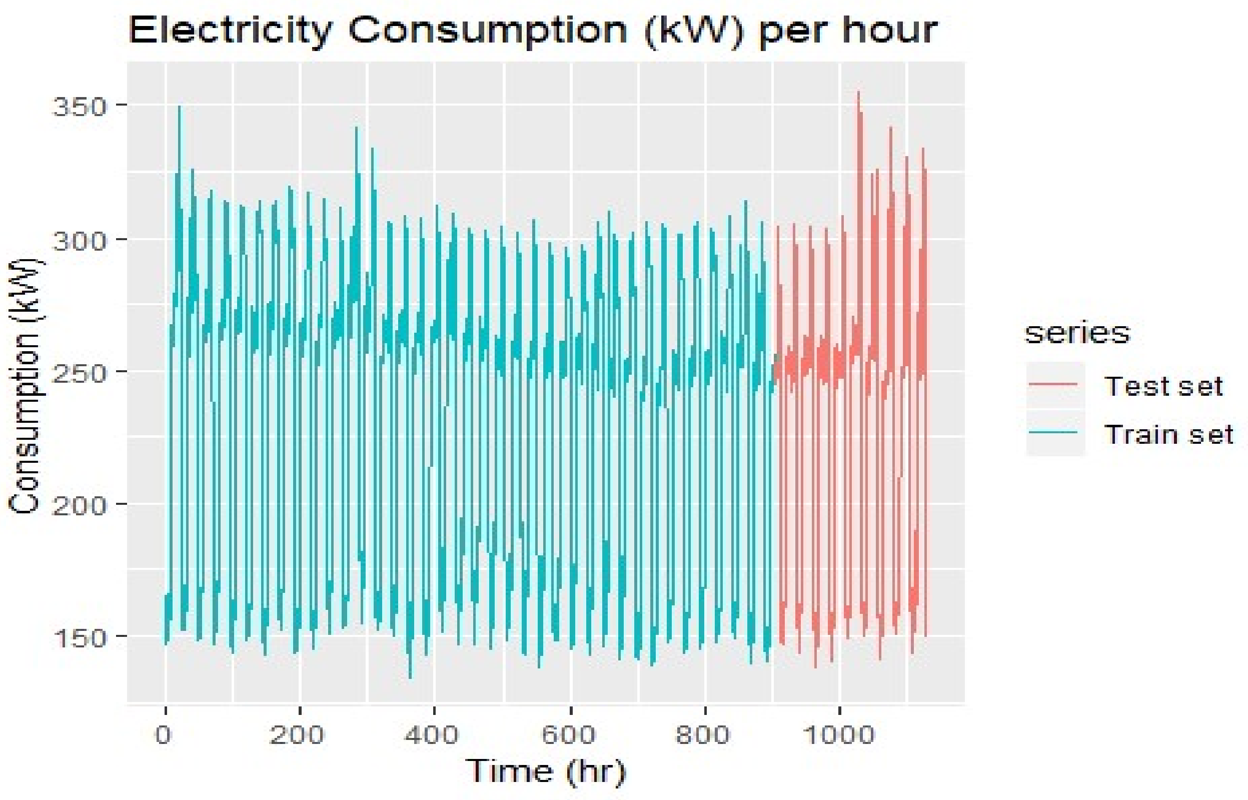
Plot data

```
library(forecast)
library(ggplot2)
autoplot(consumption) +
  ggtitle ('Electricity Consumption (kW) per hour') +
  xlab('Time (hr)') +
  ylab('Consumption (kW)')
```



Separate data into train (80%) and test set (20%) for further model evaluation

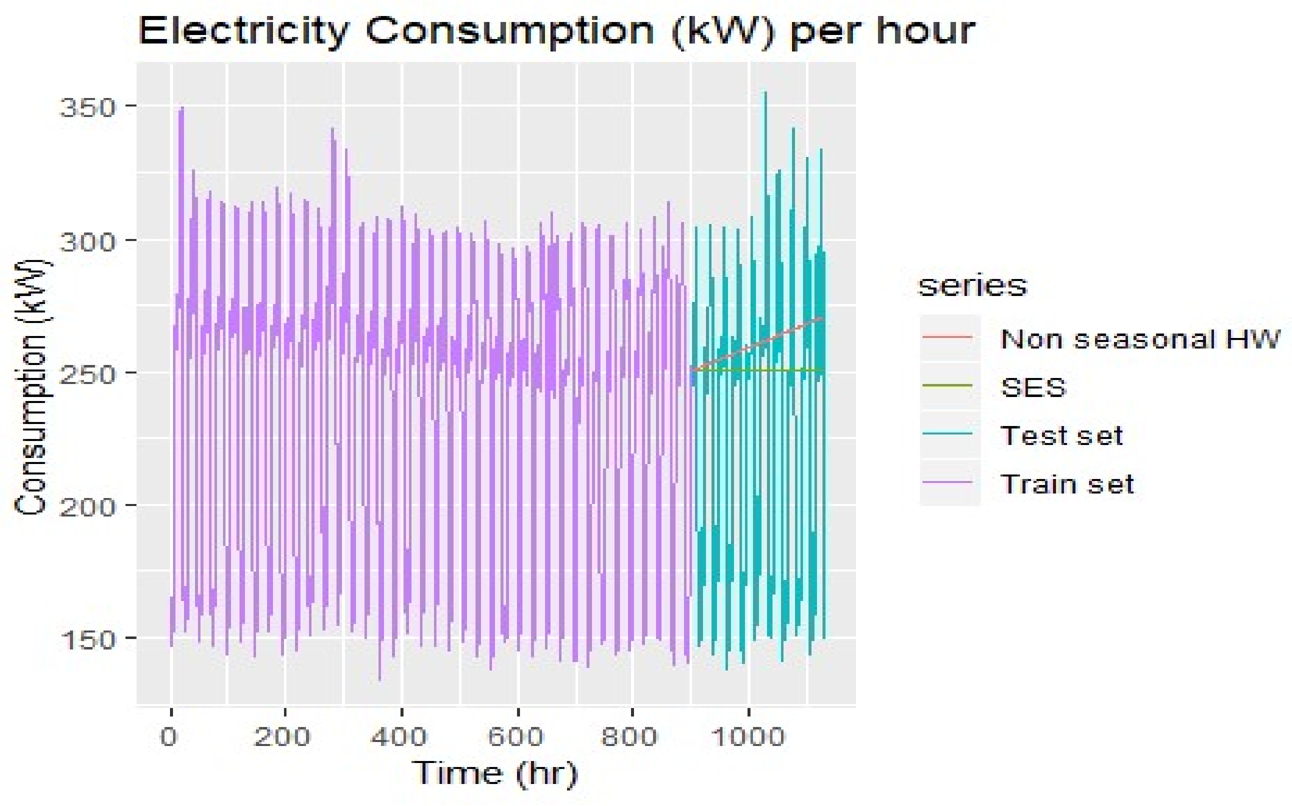
```
consum_train= window(consumption, start=c(1,2), end=c(902,4))
consum_test= window(consumption, start=c(903,1), end=c(1127,4))
autoplot(consum_train,series="Train set") +
 autolayer(consum_test,series='Test set')+
 ggtitle ('Electricity Consumption (kW) per hour') +
 xlab('Time (hr)') +
 ylab('Consumption (kW)')
```



Forecasting with Exponential Smoothing

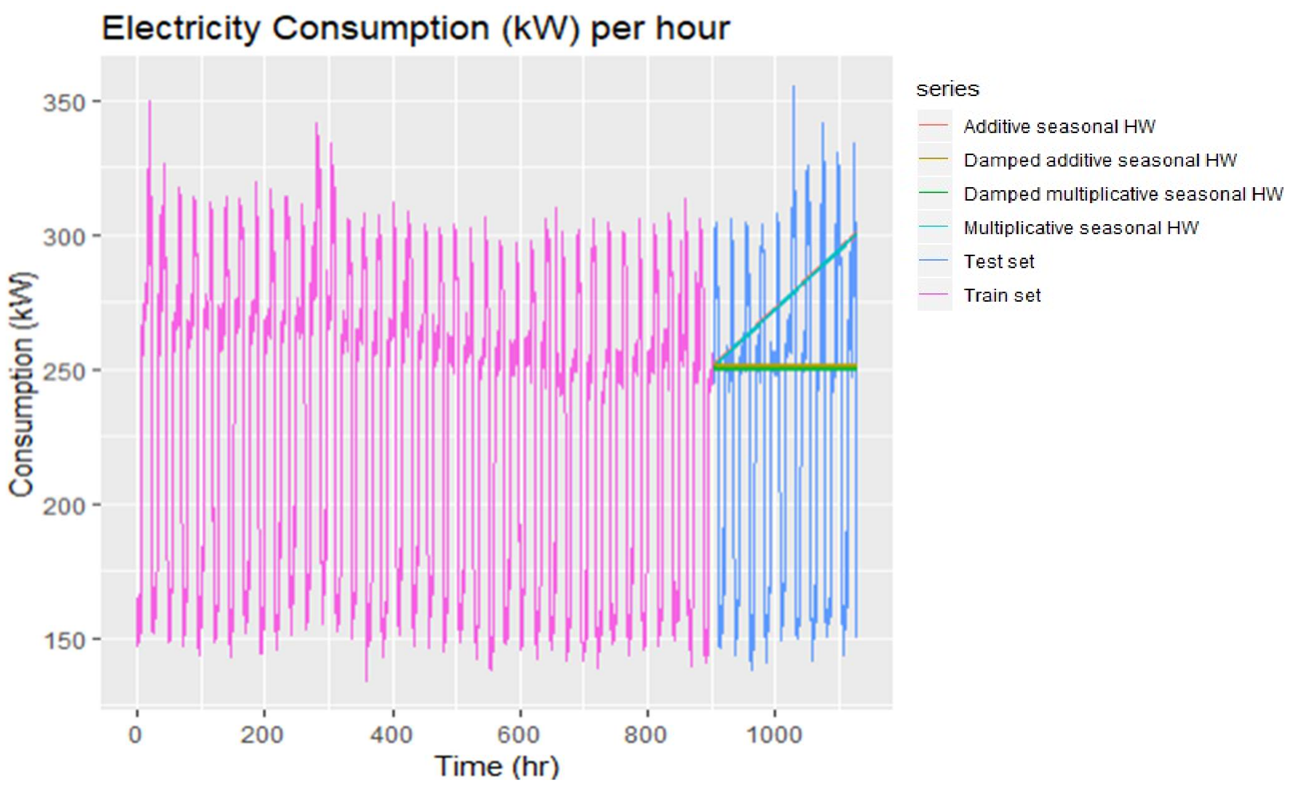
First, we consider forecasting model that not concern seasonal pattern.

```
#Simple Exponential Smoothing (SES) : Forecasting with a constant
#auto alpha selection, alpha = NULL
consum_SE = ses(consum_train,h=900, alpha=NULL)
#Non seasonal HW : Forecasting with a linear trend (auto alpha and beta selection)
consum_NHW = holt(consum_train,h=900,alpha=NULL,beta=NULL)
#2 methods in the same graph
autoplot(consum_train,series="Train set") +
 autolayer(consum_test,series='Test set')+
 autolayer(consum_SE$mean,series='SES')+
 autolayer(consum_NHW$mean,series='Non seasonal HW')+
 ggtitle ('Electricity Consumption (kW) per hour') +
 xlab('Time (hr)') +
 ylab('Consumption (kW)')
```



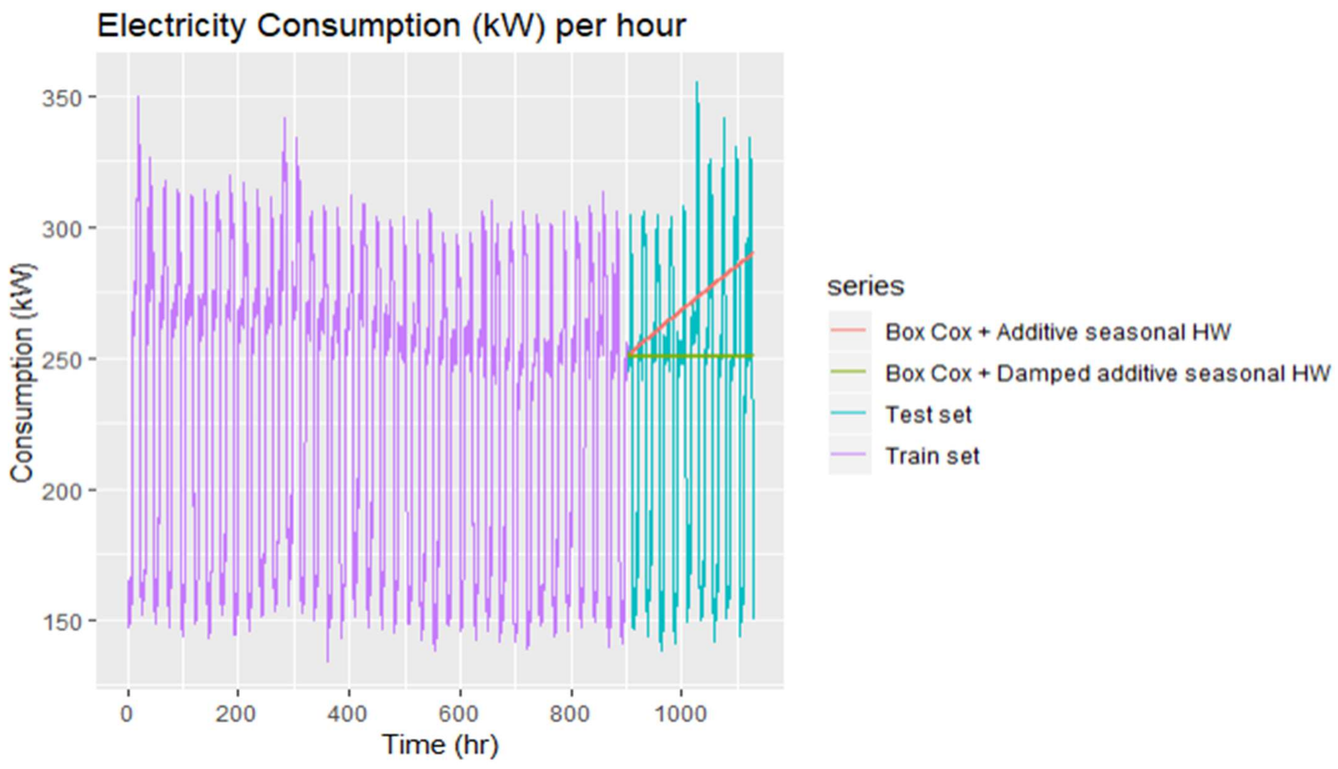
The models are not so good.
Let forecast by using both seasonal and linear trend.

```r
#Additive seasonal Holt-Winters model
consum_HW_add = hw(consum_train, seasonal='additive',h=900)
#Multiplicative seasonal Holt-Winters model
consum_HW_mul = hw(consum_train, seasonal='multiplicative',h=900)
#Damped additive seasonal Holt-Winters model
consum_DHW_add = hw(consum_train, seasonal='additive',h=900,damped=TRUE)
#Damped multiplicative seasonal Holt-Winters model
consum_DHW_mul = hw(consum_train, seasonal='multiplicative',h=900,damped=TRUE)

#Plot 4 models on the same graph
autoplot(consum_train,series="Train set") +
 autolayer(consum_test,series='Test set')+
 autolayer(consum_HW_add$mean,series='Additive seasonal HW')+
 autolayer(consum_HW_mul$mean,series='Multiplicative seasonal HW')+
 autolayer(consum_DHW_add$mean,series='Damped additive seasonal HW')+
 autolayer(consum_DHW_mul$mean,series='Damped multiplicative seasonal HW')+
 ggtitle ('Electricity Consumption (kW) per hour') +
 xlab('Time (hr)') +
 ylab('Consumption (kW)')
```



Still not so efficient.
Next, try to stabilize the variance by Box-Cox transformation in Additive HW model.

```
#Additive seasonal Holt-Winters model
consum_HW_addBC = hw(consum_train, seasonal='additive',h=900, lambda = 'auto' )
#Damped additive seasonal Holt-Winters model
consum_DHW_addBC = hw(consum_train, seasonal='additive',h=900,damped=TRUE, lambda =
'auto')

#Plot 2 models on the same graph
autoplot(consum_train,series="Train set") +
 autolayer(consum_test,series='Test set')+
 autolayer(consum_HW_addBC$mean,series='Box Cox + Additive seasonal HW')+
 autolayer(consum_DHW_addBC$mean,series='Box Cox + Damped additive seasonal HW')+
 ggtitle ('Electricity Consumption (kW) per hour') +
 xlab('Time (hr)') +
 ylab('Consumption (kW)')
```



The results show no significant improvement.
However, we can compute root mean square error (RMSE) of each model.

```
print(sqrt(mean((consum_SE$mean-consum_test)^2)))

## [1] 60.76483

print(sqrt(mean((consum_NHW$mean-consum_test)^2)))

## [1] 64.57402

print(sqrt(mean((consum_HW_add$mean-consum_test)^2)))

## [1] 73.46383

print(sqrt(mean((consum_HW_mul$mean-consum_test)^2)))

## [1] 73.40462

print(sqrt(mean((consum_DHW_add$mean-consum_test)^2)))

## [1] 60.91925

print(sqrt(mean((consum_DHW_mul$mean-consum_test)^2)))

## [1] 60.48178

print(sqrt(mean((consum_HW_addBC$mean-consum_test)^2)))

## [1] 70.18168

print(sqrt(mean((consum_DHW_addBC$mean-consum_test)^2)))

## [1] 60.84705
```

The model with the lowest error so far is Damped multiplicative Holt-Winters.
However, this is not because it is the best model (See that the prediction pattern is not correlate with pattern of test set). This low error is just because our calculation base on mean different and Damped multiplicative Holt-Winters gives us the linear model that situates around the middle of the graph.

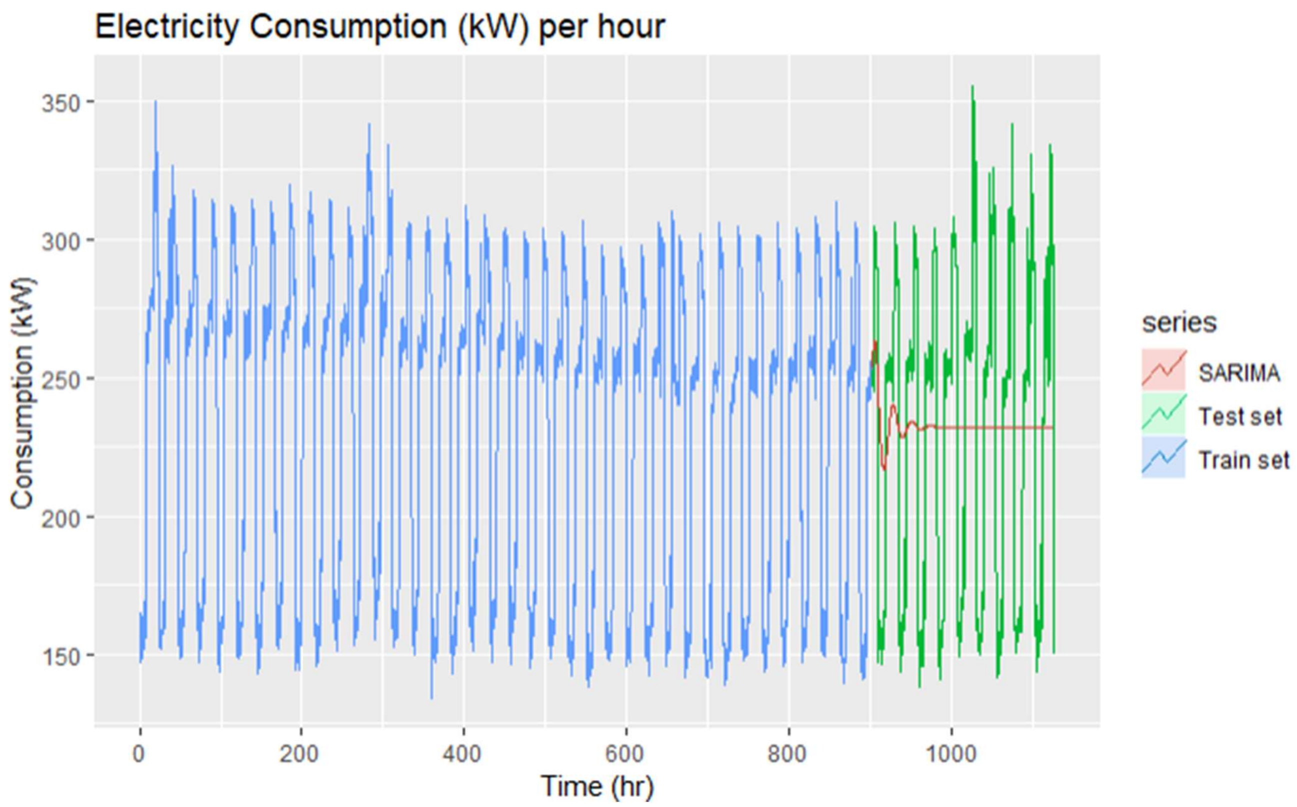
We should also consider that all alpha, beta, gamma and phi parameters used above are automatically chosen just to screen and see the pattern of each forecasting model. Therefore, if we really want to compare between each model, the parameters should be fixed. (For example, if you want to compare the effect of damped version to multiplicative Holt-Winters model, you should fix alpha, beta and gamma for both model (the only difference will be with or without phi parameter)).

But we will not consider them now since clearly we cannot use any exponential smoothing for our forecast.

Forecasting with ARIMA

Let's begin with automaticaly SARIMA model to see the possibility.

```r
consum_SARIMA = auto.arima(consum_train)
pred_consum_SARIMA = forecast(consum_SARIMA,h=900)
autoplot(consum_train,series="Train set") +
  autolayer(consum_test,series='Test set')+
  autolayer(pred_consum_SARIMA,series='SARIMA',PI=FALSE)+
  ggtitle ('Electricity Consumption (kW) per hour') +
  xlab('Time (hr)') +
  ylab('Consumption (kW)')
```



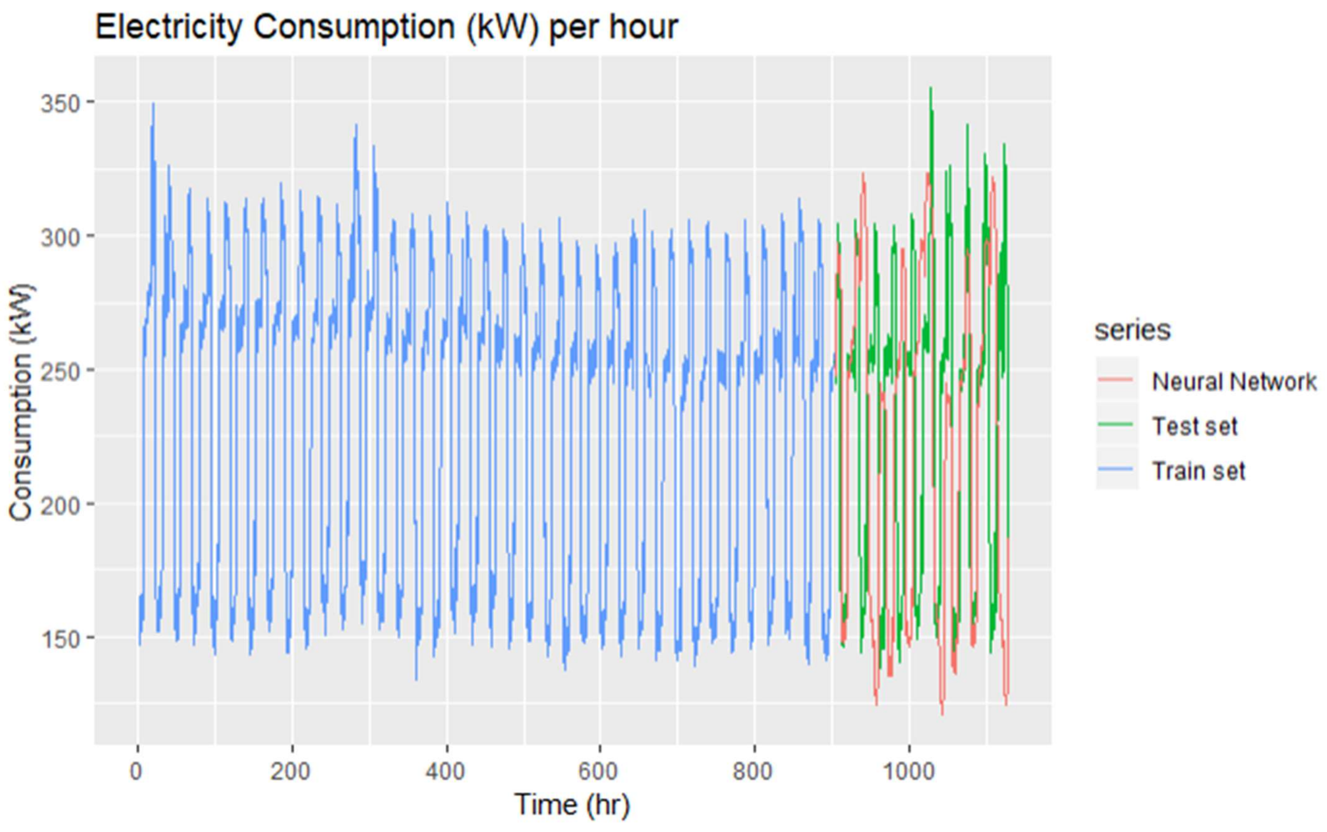
Not perfect but seems better than before.
Let's check model accuracy.

```r
print(sqrt(mean((pred_consum_SARIMA$mean-consum_test)^2)))

## [1] 56.392
```

This model shows less error but the prediction pattern are still not so good.
We should concern model that more flexible like Neural Network Auto-Regression.

Neural Network Auto-Regression

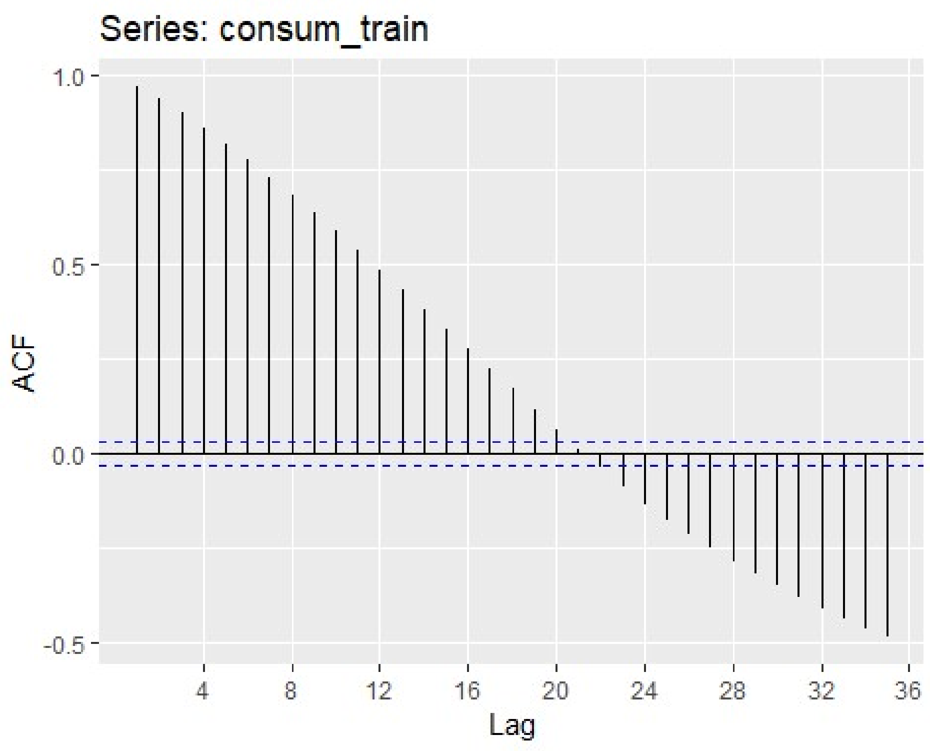
```
#First, using automatic choice for parameters p and k
consum_train_NN = nnetar(consum_train)
pred_consum_train_NN = forecast(consum_train_NN, h = 900)
autoplot(consum_train,series="Train set") +
  autolayer(consum_test,series='Test set')+
  autolayer(pred_consum_train_NN$mean,series='Neural Network')+
  ggtitle ('Electricity Consumption (kW) per hour') +
  xlab('Time (hr)') +
  ylab('Consumption (kW)')
```



```
#Check model accuracy
print(sqrt(mean((pred_consum_train_NN$mean-consum_test)^2)))

## [1] 83.40126
```

Even the error is higher than SARIMA model but the prediction pattern seems correct.
This correlates with the information from auto-correlation function (acf) below. See that auto-correlation declines slowly as the number of lags increases. This is a property of non-stationarity that will effect the efficiency of several forecasting models. It also possible that our data might has no seasonal but cyclic pattern. In that case, they cannot be modelized by usual linear model.

```
#Auto-correlation pattern
ggAcf(consum_train)
```

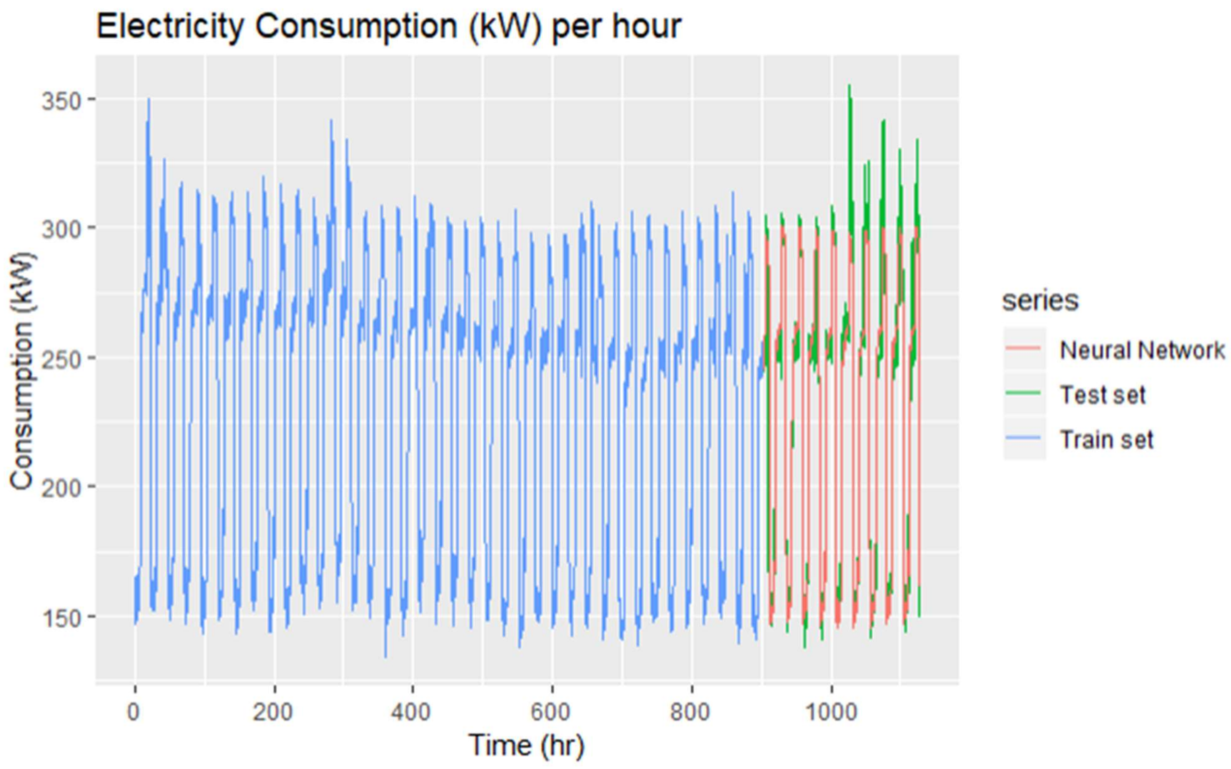


Now, let’s verify information from Neural network model for further adjustment.

```
#Check model information for further adjustment to improve the model
print(consum_train_NN)

## Series: consum_train
## Model:  NNAR(35,1,18)[4]
## Call:   nnetar(y = consum_train)
##
## Average of 20 networks, each of which is
## a 35-18-1 network with 667 weights
## options were - linear output units
##
## sigma^2 estimated as 46.39
```

Edit the model by adding more neuron and stabilize variance by Box-Cox transformation.

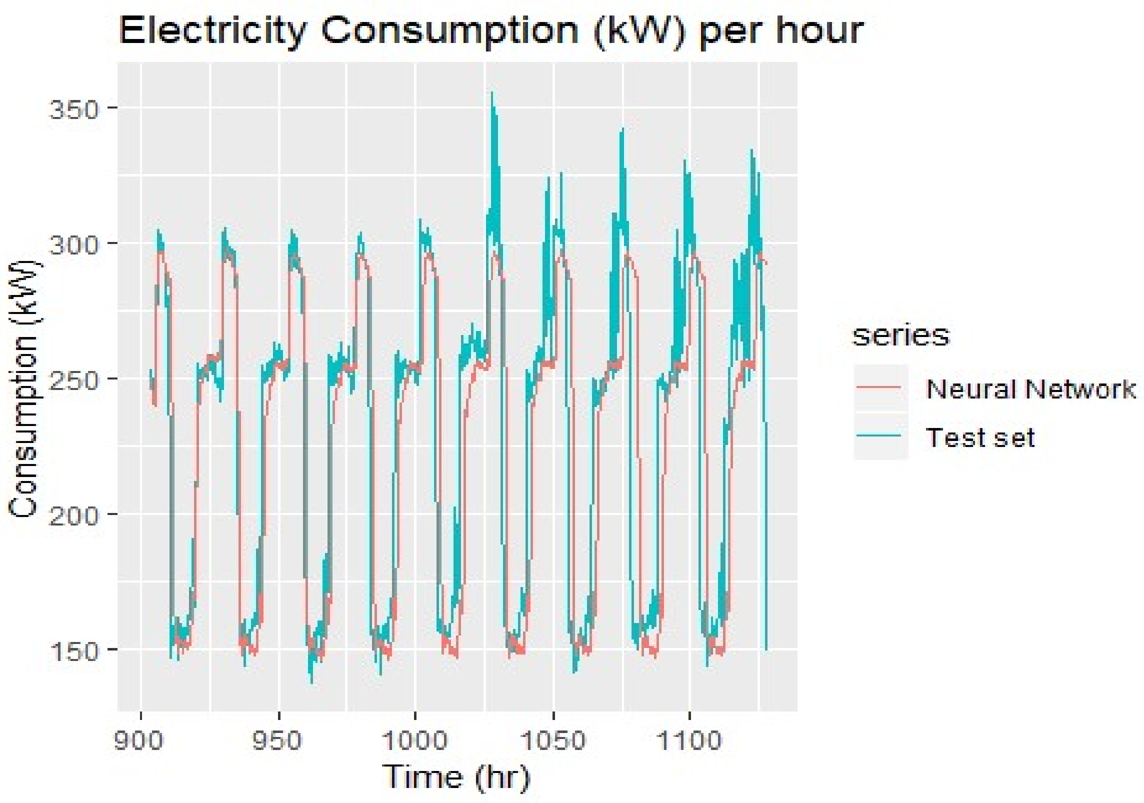
```
consum_train_NN2 = nnetar(consum_train,35,1,25,lambda='auto')
pred_consum_train_NN2 = forecast(consum_train_NN2, h = 900)
autoplot(consum_train,series="Train set") +
  autolayer(consum_test,series='Test set')+
  autolayer(pred_consum_train_NN2$mean,series='Neural Network')+
  ggtitle ('Electricity Consumption (kW) per hour') +
  xlab('Time (hr)') +
  ylab('Consumption (kW)')
```



```
#Check model accuracy
print(sqrt(mean((pred_consum_train_NN2$mean-consum_test)^2)))

## [1] 16.25184
```

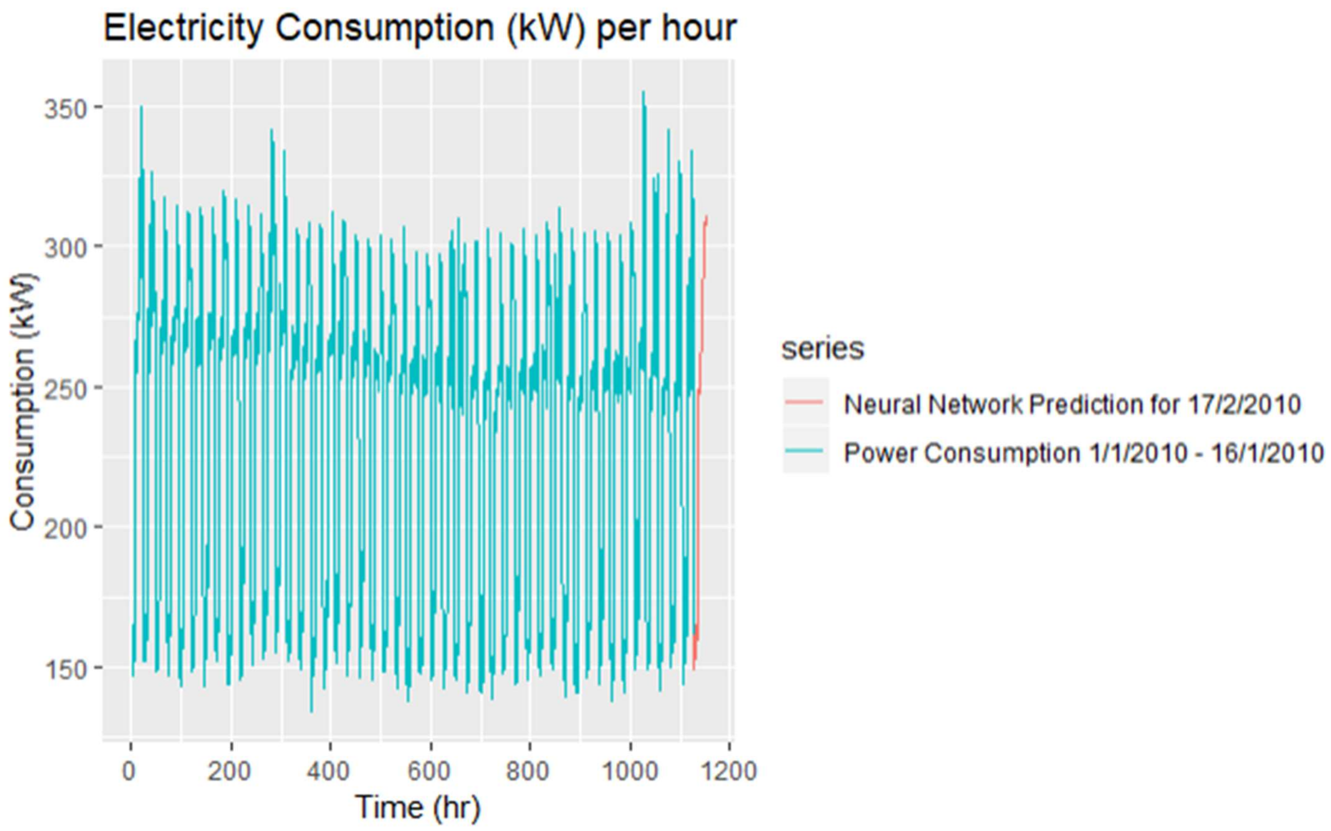
The error seems so much better.
Seems like we finally found the best model !
We can zoom in the prediction part to make it more clear.

```
autoplot(consum_test,series='Test set') +
  autolayer(pred_consum_train_NN2$mean,series='Neural Network')+
  ggtitle ('Electricity Consumption (kW) per hour') +
  xlab('Time (hr)') +
  ylab('Consumption (kW)')
```



Now we will predict electricity consumption of 17/2/2010 bases on the whole previous consumption information.
The prediction interval = 24 hr. of 17/2/2010, h = (24*60)/15 = 96 observations

```
consum_NN = nnetar(consumption,35,1,25,lambda='auto')
pred_consum_NN = forecast(consum_NN, h = 96)
autoplot(consumption,series="Power Consumption 1/1/2010 - 16/1/2010") +
  autolayer(pred_consum_NN$mean,series='Neural Network Prediction for 17/2/2010')+
  ggtitle ('Electricity Consumption (kW) per hour') +
  xlab('Time (hr)') +
  ylab('Consumption (kW)')
```



```
#Prediction results
Prediction = print(pred_consum_NN)
#Save prediction results to csv file
library("readr")
write_csv(Prediction,path="Prediction.csv")
```

Now, we will move to second objective of this forecasting.

Part 2:
Forecast electricity consumption (kW) for 2/17/2010 by using outdoor temperature

Start by download data file and make time series object as before.

```
temp <- ts(data[1:4507,3], frequency = 4, start=c(1,2))
head(temp)    #Check time series object

##   Qtr1 Qtr2 Qtr3 Qtr4
## 1      10.6 10.6 10.6
## 2 10.6 10.6 10.6
```

Extract data to make the regression and prediction

```
temp_forTRAIN=window(temp, start=c(1,2), end=c(902,4))
temp_forTEST=window(temp, start=c(903,1), end=c(1127,4))
```

Time series linear regression model

First of all, we check the effect of temperature to electricity consumption

```
fit_train=tslm(consum_train~temp_forTRAIN)
summary(fit_train)

##
## Call:
## tslm(formula = consum_train ~ temp_forTRAIN)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -121.103  -42.887    3.147   42.935  112.113
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   126.8451     3.5127   36.11   <2e-16 ***
## temp_forTRAIN   9.8531     0.3202   30.77   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 51.23 on 3605 degrees of freedom
## Multiple R-squared:  0.208,  Adjusted R-squared:  0.2078
## F-statistic: 946.8 on 1 and 3605 DF,  p-value: < 2.2e-16
```

The effect of temperature to electricity consumption are statisticaly significant.
We can add trend and seasonal pattern to this regression.

```
fit_train_TS=tslm(consum_train~temp_forTRAIN+trend+season)
summary(fit_train_TS)
```

```
##
## Call:
## tslm(formula = consum_train ~ temp_forTRAIN + trend + season)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -126.149  -42.694    2.596   43.334  122.641
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.363e+02  3.842e+00  35.485   <2e-16 ***
## temp_forTRAIN  1.052e+01  3.201e-01  32.862   <2e-16 ***
## trend         -9.350e-03  8.189e-04 -11.418   <2e-16 ***
## season2        8.304e-01  2.372e+00   0.350    0.726
## season3        6.519e-01  2.372e+00   0.275    0.783
## season4       -2.762e-01  2.372e+00  -0.116    0.907
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 50.35 on 3601 degrees of freedom
## Multiple R-squared:  0.2357, Adjusted R-squared:  0.2347
## F-statistic: 222.1 on 5 and 3601 DF,  p-value: < 2.2e-16
```

Seems like seasonal pattern play no role Let’s try to consider only trend.

```
fit_train_T=tslm(consum_train~temp_forTRAIN+trend)
summary(fit_train_T)
```

```
##
## Call:
## tslm(formula = consum_train ~ temp_forTRAIN + trend)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -125.799  -42.515    2.637   43.253  122.063
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.366e+02  3.556e+00   38.42   <2e-16 ***
## temp_forTRAIN  1.052e+01  3.200e-01   32.88   <2e-16 ***
## trend         -9.350e-03  8.186e-04  -11.42   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 50.33 on 3604 degrees of freedom
## Multiple R-squared:  0.2357, Adjusted R-squared:  0.2352
## F-statistic: 555.6 on 2 and 3604 DF,  p-value: < 2.2e-16
```

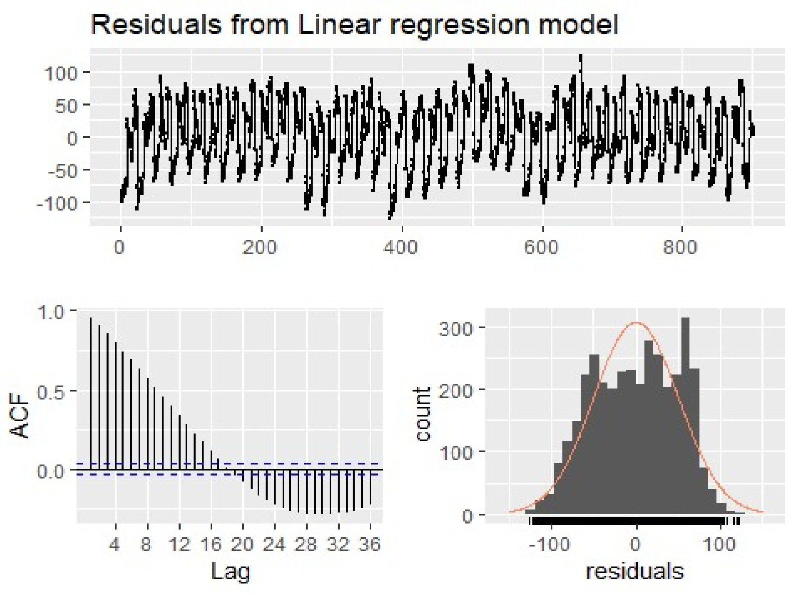
Compare all models

```
CV(fit_train)

##           CV          AIC         AICc          BIC        AdjR2
## 2.625386e+03 2.840046e+04 2.840046e+04 2.841903e+04 2.077773e-01

CV(fit_train_TS)

##           CV          AIC         AICc          BIC        AdjR2
## 2.539035e+03 2.827988e+04 2.827991e+04 2.832321e+04 2.346706e-01

CV(fit_train_T)

##           CV          AIC         AICc          BIC        AdjR2
## 2.535017e+03 2.827417e+04 2.827419e+04 2.829894e+04 2.352452e-01
```

Model with temperature and trend has the lowest AIC and the highest Adjusted R squared. Therefore, I will choose this model for further step.

Since time series linear regression model assume that the residuals are independent and identically distributed. So, we should check the residuals of our model first.

```
checkresiduals(fit_train_T,test="LB",plot=TRUE)
```



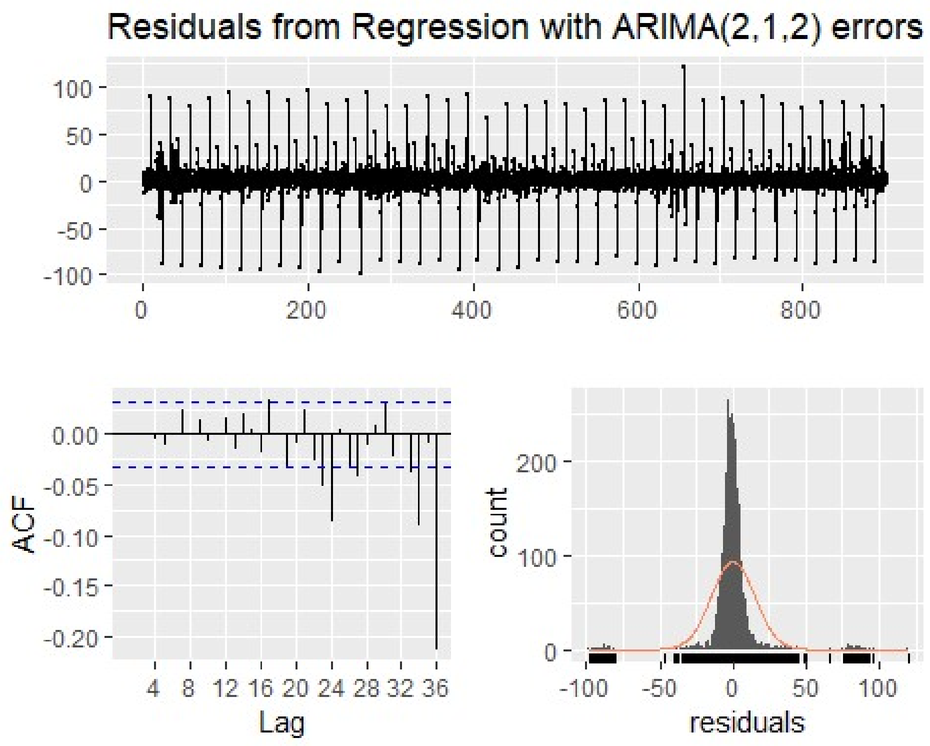
```
##
##  Ljung-Box test
##
## data:  Residuals from Linear regression model
## Q* = 17486, df = 5, p-value < 2.2e-16
##
## Model df: 3.   Total lags used: 8
```

Results show that the residual are dependent to each other.
Therefore, we cannot use this linear regression model.
The appropriate model in case the residual are not independent to each other are Dynamic regression model.

Dynamic regression model

Let's start with function with automatic selected parameters

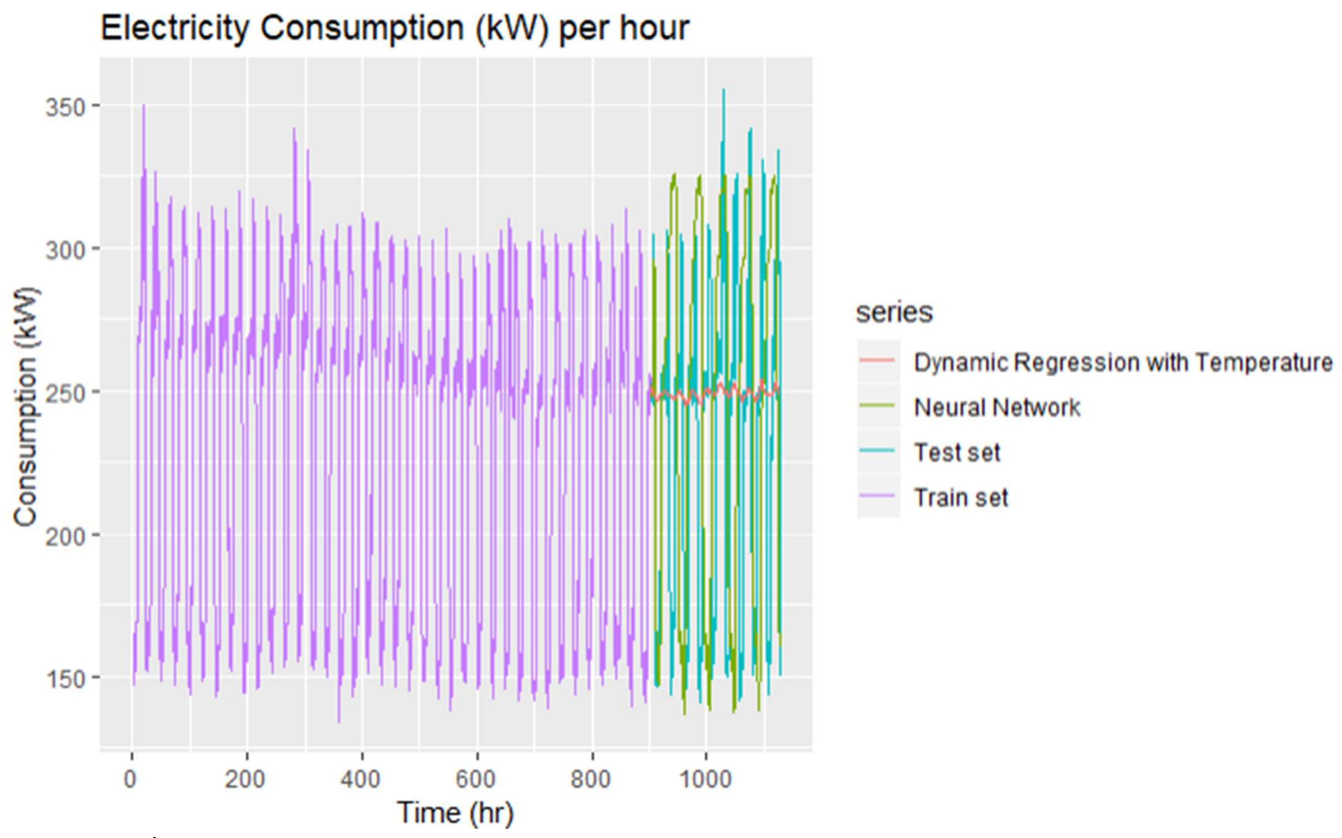
```
fit_train_T_ar = auto.arima(consum_train,xreg=temp_forTRAIN)
#Check autocorrelation of residuals
checkresiduals(fit_train_T_ar,test="LB",plot=TRUE)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(2,1,2) errors
## Q* = 2.551, df = 3, p-value = 0.4662
##
## Model df: 5.   Total lags used: 8
```

See that all the autocorrelations of the residuals have been modelled with this model.
The model being validated, now we can forecast the test set.

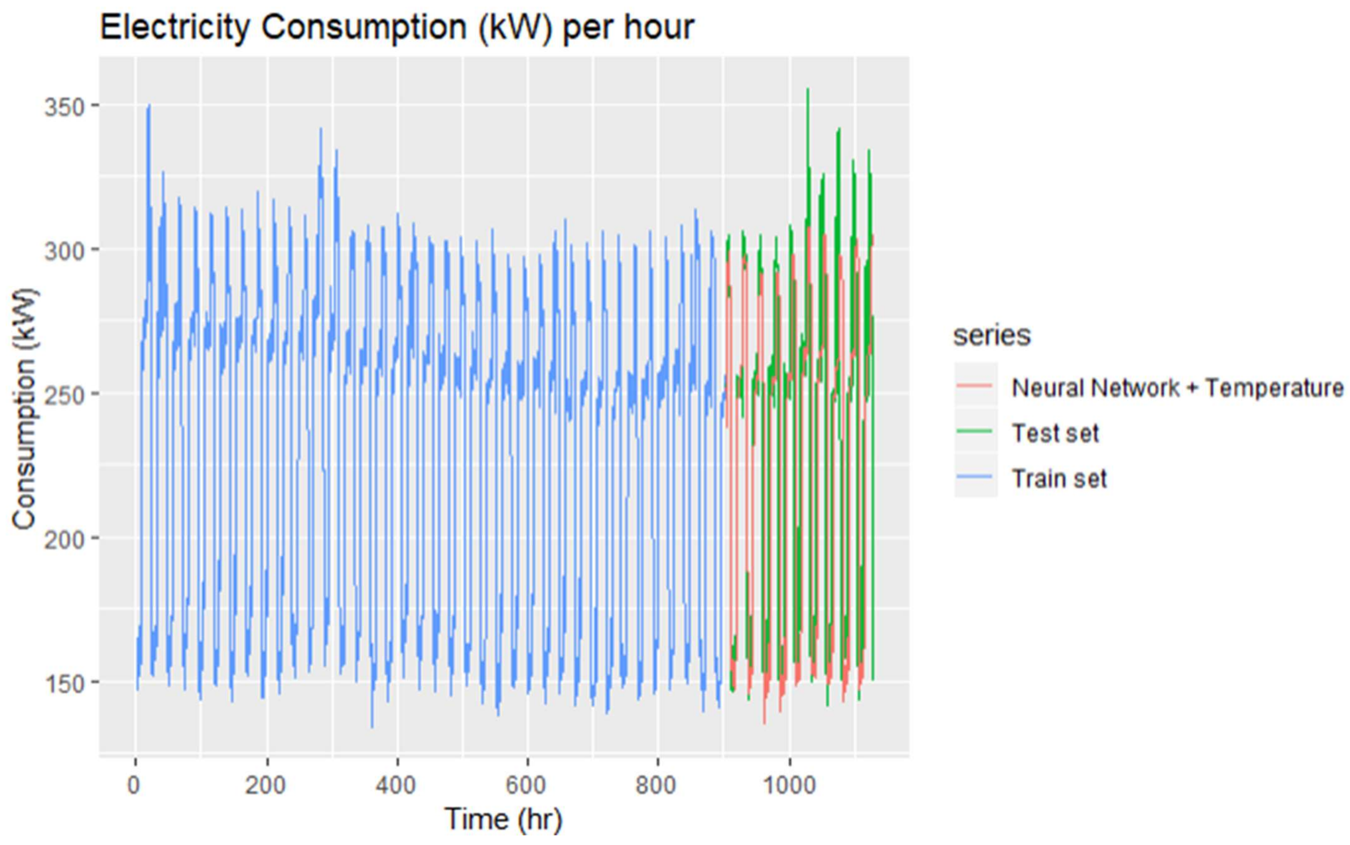
```
predict_test_T = forecast(fit_train_T_ar,xreg=temp_forTEST, h=900)
#Compare the prediction wit Neuron network model(without temperature)
autoplot(consum_train,series="Train set") +
  autolayer(consum_test,series='Test set')+
  autolayer(pred_consum_train_NN2$mean,series='Neural Network')+
  autolayer(predict_test_T$mean,series='Dynamic Regression with Temperature')+
  ggtitle ('Electricity Consumption (kW) per hour') +
  xlab('Time (hr)') +
  ylab('Consumption (kW)')
```



Not so good.

The best model that we have so far is Neural Network, we should try them with temperature variable.

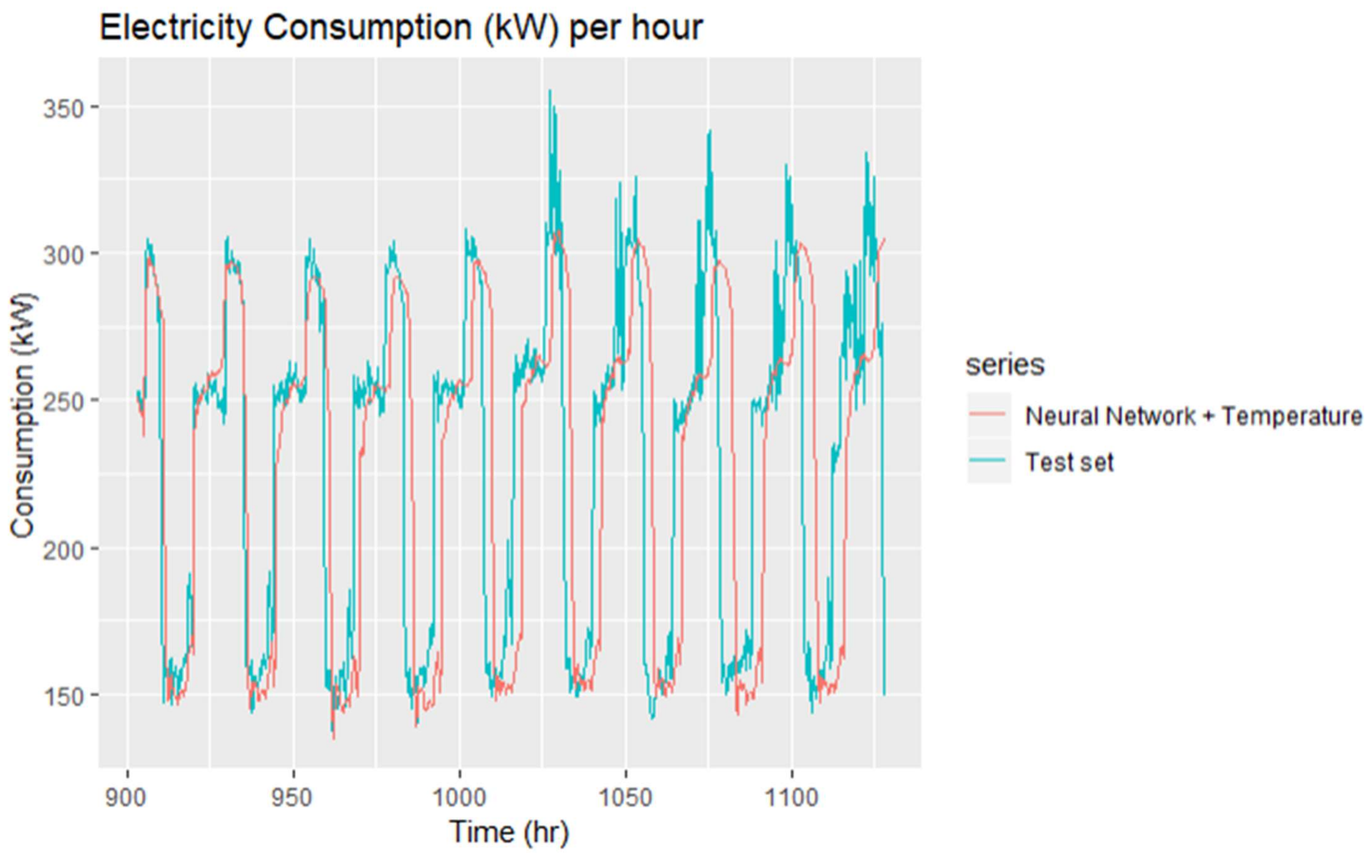
```r
consum_train_NN_T = nnetar(consum_train,35,1,25,lambda='auto',xreg=temp_forTRAIN)
pred_consum_train_NN_T = forecast(consum_train_NN_T, h = 900,xreg=temp_forTEST)
autoplot(consum_train,series="Train set") +
 autolayer(consum_test,series='Test set')+
 autolayer(pred_consum_train_NN_T$mean,series='Neural Network + Temperature')+
 ggtitle ('Electricity Consumption (kW) per hour') +
 xlab('Time (hr)') +
 ylab('Consumption (kW)')
```



```r
#Check model accuracy
print(sqrt(mean((pred_consum_train_NN_T$mean-consum_test)^2)))

## [1] 50.70654
```

We can zoom in the prediction.

```
autoplot(consum_test,series='Test set') +
 autolayer(pred_consum_train_NN_T$mean,series='Neural Network + Temperature')+
 ggtitle ('Electricity Consumption (kW) per hour') +
 xlab('Time (hr)') +
 ylab('Consumption (kW)')
```

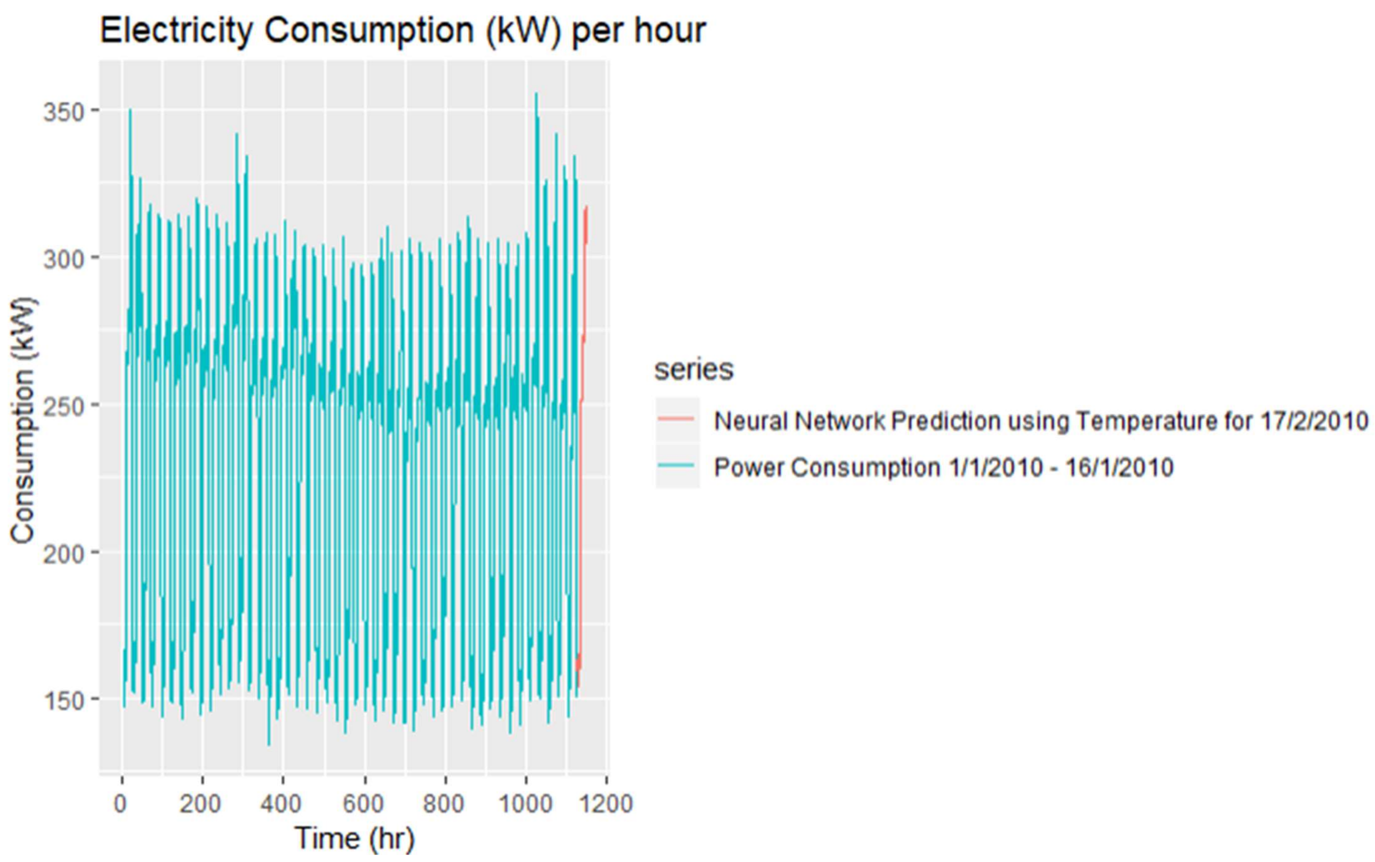


Not so bad but both pattern and error are worse than neural network forecast without temperature.

Let’s forecast power consumption on 17th Feb based on temperature of that day.

```
#Extract temperature of 17th Feb to make new time series object for forecast
temp_17 <- ts(data[4509:4603,3], frequency = 4, start=c(1,2))
head(temp_17)

##   Qtr1 Qtr2 Qtr3 Qtr4
## 1      11.1 11.1 11.1
## 2 11.1 10.6 10.6
```

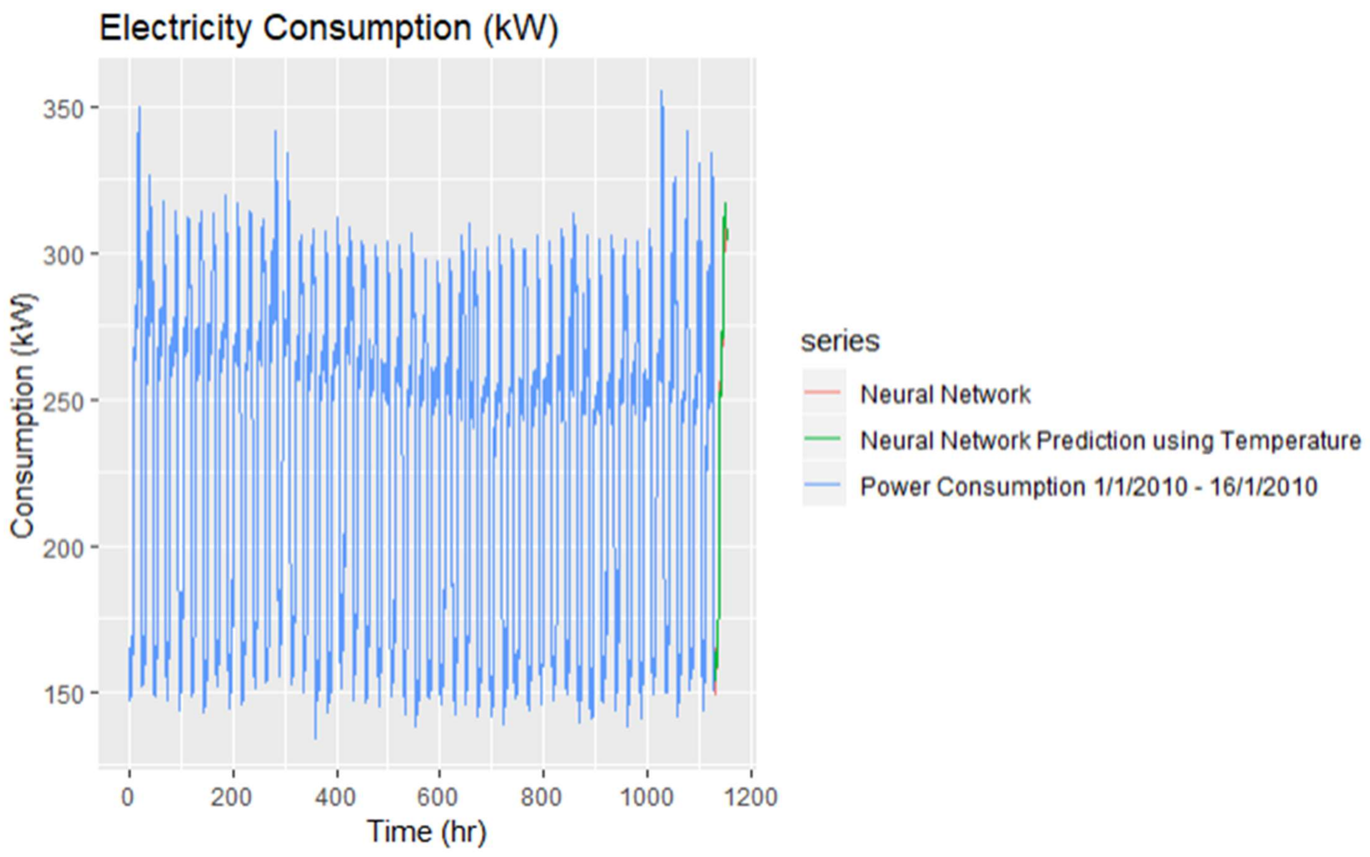
```
#Check time series object

consum_NN_T = nnetar(consumption,35,1,25,lambda='auto',xreg=temp)
pred_consum_NN_T = forecast(consum_NN_T, h = 96,xreg=temp_17)
autoplot(consumption,series="Power Consumption 1/1/2010 - 16/1/2010") +
 autolayer(pred_consum_NN_T$mean,series='Neural Network Prediction using Temperature f
or 17/2/2010')+
 ggtitle ('Electricity Consumption (kW) per hour') +
 xlab('Time (hr)') +
 ylab('Consumption (kW)')
```



```
#Prediction results
Prediction_T = print(pred_consum_NN_T)

#Save prediction results to csv file
library("readr")
write_csv(Prediction_T,path="Prediction with Temperature.csv")
```

We can compare both predictions in the same graph.

```
autoplot(consumption,series="Power Consumption 1/1/2010 - 16/1/2010") +
 autolayer(pred_consum_NN$mean,series='Neural Network')+
 autolayer(pred_consum_NN_T$mean,series='Neural Network Prediction using Temperature')
+
 ggtitle ('Electricity Consumption (kW)') +
 xlab('Time (hr)') +
 ylab('Consumption (kW)')
```



Conclusion:

The best forecast model we have is Neural Network auto-regression (NNAR) without consider outside temperature variable with root mean square error 16.25184.
The main advantage of NNAR are more flexible and can modelized non-linear relation like in our case.