Forecast using a Neural Network model

Lorena Romeo

2025-01-01

Purpose:

This is the bonus part, in which I will forecast the power consumption using a Neural Network model, taking into account also the Temperature regressor.

The first steps are the same of the main project presentation.

Load necessary libraries:

```
library(readxl)
library(openxlsx)
library(lubridate)
library(dplyr)
library(ggplot2)
library(forecast)
library(zoo)
library(writexl)
```

Load file

```
#load xlsx file
data_row = read.xlsx("C:/Users/Utilisateur/OneDrive/Desktop/2023-11-Elec-train.xlsx")
#plot(data_row)
head(data_row) #the first record on timestamp column is not in the good format, I need to convert it in
```

```
##            Timestamp Power.(kW) Temp.(C°)
## 1 40179.052083333336      165.1  10.55556
## 2      1/1/2010 1:30      151.6  10.55556
## 3      1/1/2010 1:45      146.9  10.55556
## 4      1/1/2010 2:00      153.7  10.55556
## 5      1/1/2010 2:15      153.8  10.55556
## 6      1/1/2010 2:30      159.0  10.55556
```

```
#str(data_row)
```

Work on date conversion

```r
convert_timestamp = function(ts) {
  if (grepl("^[0-9]+\\.[0-9]+$", ts)) {
    # Handle Excel serial date format
    excel_date = as.numeric(ts)
    origin = as.Date("1899-12-30")
    date = origin + excel_date
    return(as.POSIXct(date))
  } else {
    # Handle standard datetime format
    return(parse_date_time(ts, orders = "m/d/y H:M", tz = "UTC"))
  }
}

# Apply conversion function to Timestamp column
data_row$Timestamp = sapply(data_row$Timestamp, convert_timestamp)

#the timestamp is numeric now, I need to convert to datetime
# Convert numeric Unix epoch timestamps to POSIXct
data_row$Timestamp = as.POSIXct(data_row$Timestamp, origin = "1970-01-01", tz = "UTC")

#head(data_row)
```

Rename the variables for easiness

```r
#now that the dataset is clean I will rename the variables in a easier way
# Rename columns to valid names
elec = data_row %>%
  rename(
    Power_kW = `Power.(kW)`,
    Temp_C = `Temp.(C°)` # Use the exact name here
  )

#verify results
# str(elec)
# head(elec)
```

```r
elec <- elec %>%
  filter(!is.na(Power_kW))

#interpolation
elec <- elec %>%
  mutate(
    Power_kW = na.approx(ifelse(Power_kW == 0, NA, Power_kW))
  )
```

Create train and test dataset

```
# Set up the start and end dates fro train and test dataset
train_start = as.POSIXct("2010-01-01 01:15:00", tz = "UTC")
train_end = as.POSIXct("2010-02-18 23:45:00", tz = "UTC")
test_start = as.POSIXct("2010-02-19 00:00:00",  tz = "UTC")
test_end = as.POSIXct("2010-02-20 23:45:00",  tz = "UTC")

# Create training and test datasets
set.seed(123)
train_data = elec %>% filter(Timestamp >= train_start & Timestamp <= train_end)
test_data = elec %>% filter(Timestamp >= test_start & Timestamp <= test_end)


# str(train_data)
# str(test_data)

#check the boundaries of the datasets
# min(train_data$Timestamp)
# max(train_data$Timestamp)
# min(test_data$Timestamp)
# max(test_data$Timestamp)
```

Create Time Series objects

```
# Create time series objects (without temperature)
# Create time series objects
frequency = 96  # 96 intervals per day (15-minute intervals)
power_train = ts(train_data$Power_kW, start = c(1, 6), end = c(49, 96), frequency = frequency)
#I do not put the end in the test part because the boundary is already set in the code above
power_test = ts(test_data$Power_kW, start = c(50, 1),  frequency = frequency)

# #verify results
# str(power_train)
# str(power_test)
```

Create a first Neural Network Model using NNAR

This is a first attempt on creating a Neural Network model without fine tuning.

```
#create neural model
fit_nnar = nnetar(power_train, xreg = train_data$Temp_C)

#check predictions using test data
prev_nnar = forecast(fit_nnar, xreg = test_data$Temp_C, h = length(power_test))
```

Check quality of the model

```
# # Mean Absolute Error
# mae = mean(abs(prev_nnar$mean - power_test))
# print(mae)
#
# # Mean Squared Error
# mse = mean((prev_nnar$mean - power_test)^2)
# print(mse)

# Root Mean Squared Error
nnr1_rmse = sqrt(mean((prev_nnar$mean - power_test)^2))
print(nnr1_rmse)
```

```
## [1] 19.08972
```

```
# # Mean Absolute Percentage Error
# mape = mean(abs((prev_nnar$mean - power_test) / power_test)) * 100
# print(mape)
#
# # Symmetric Mean Absolute Percentage Error
# smape = mean(2 * abs(prev_nnar$mean - power_test) / (abs(prev_nnar$mean) + abs(power_test))) * 100
# print(smape)
```

From the RMSE I can suppose that with a bit of fine tune the model can have a better fit.

Retrieve the parameters of the auto generated model to understand how to fine tune it

I extract the parameters of the auto generated nnar and then decide what to change

```
# View specific model components
p_value = fit_nnar$p  # Number of non-seasonal lags
P_value = fit_nnar$P  # Number of seasonal lags
size_value = fit_nnar$size  # Number of neurons in the hidden layer
lambda_value = fit_nnar$lambda  # Box-Cox transformation parameter (if used)
repeats_value = fit_nnar$repeats  # Number of networks averaged


# Print these values
cat("Number of lags (p):", p_value, "\n")
```

```
## Number of lags (p): 25
```

```
cat("Number of seasonal lags (P):", P_value, "\n")
```

```
## Number of seasonal lags (P): 1
```

```r
cat("Number of neurons (size):", size_value, "\n")
```

```
## Number of neurons (size): 14
```

```r
cat("Box-Cox transformation (lambda):", lambda_value, "\n")
```

```
## Box-Cox transformation (lambda):
```

```r
cat("Number of repeats:", repeats_value, "\n")
```

```
## Number of repeats:
```

#Fit manually a new model and forecast

I can now try to fine tune the NN model.

```r
#1st attempt
fit2_nnar = nnetar(
  power_train,
  p = 15,
  P = 1,
  size = 15,
  repeats = 10,
  lambda = "auto",
  xreg = train_data$Temp_C
)
```

Make forecast based on test set and check quality of the model.

```r
# Make forecasts
prev_nnar2 = forecast(fit2_nnar, xreg = test_data$Temp_C, h = length(power_test))


# Evaluate model performance on adjusted test data
mse = mean((prev_nnar2$mean - power_test)^2)
print(mse)
```

```
## [1] 223.9655
```

```r
# Root Mean Squared Error
nnr2_rmse = sqrt(mean((prev_nnar2$mean - power_test)^2))
print(nnr2_rmse)
```

```
## [1] 14.96548
```

Fit manually another model and forecast

```r
#2nd attempt
fit3_nnar = nnetar(
  power_train,
  p = 15,
  P = 3,
  size = 15,
  repeats = 10,
  lambda = "auto",
  xreg = train_data$Temp_C
)
```

Make forecast based on test set and check quality of the model.

```r
# Make forecasts
prev_nnar3 = forecast(fit3_nnar, xreg = test_data$Temp_C, h = length(power_test))

# Evaluate model performance on adjusted test data
mse = mean((prev_nnar3$mean - power_test)^2)
print(mse)
```

```
## [1] 379.552
```

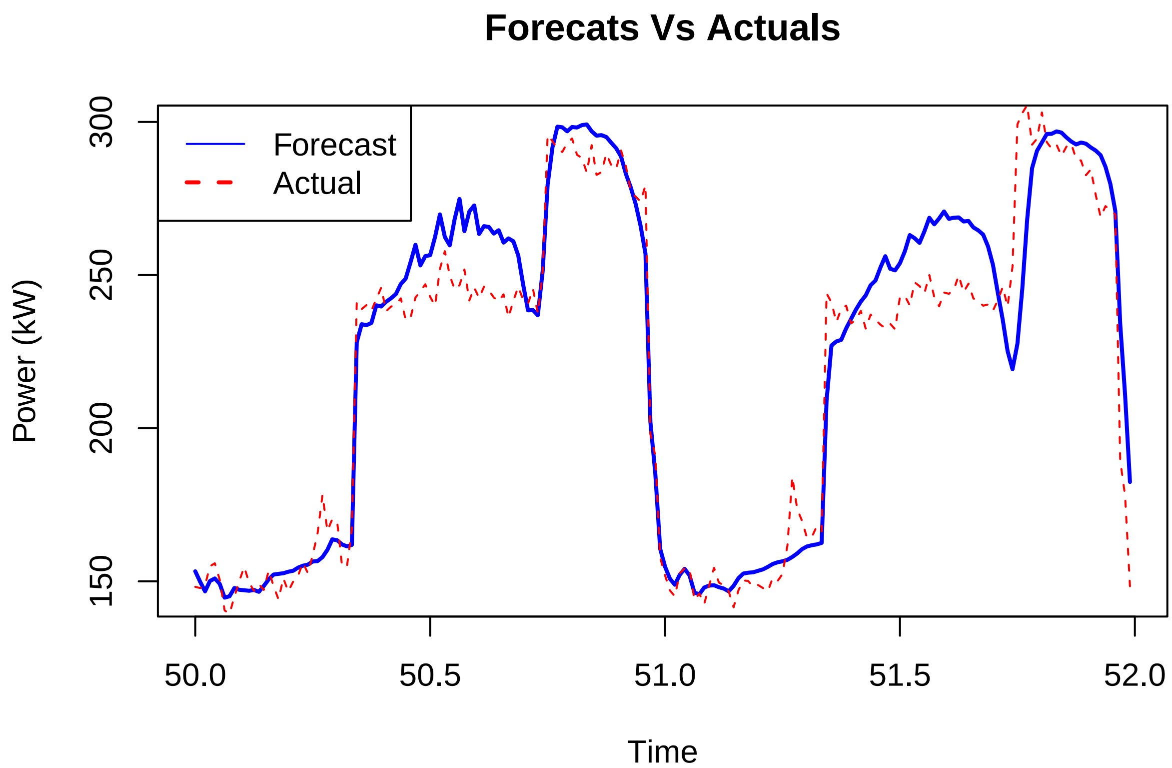
```r
# Root Mean Squared Error
nnr3_rmse = sqrt(mean((prev_nnar3$mean - power_test)^2))
print(nnr3_rmse)
```

```
## [1] 19.48209
```

From the RMSE of the three NN models I can see that the 2nd attempt (nnr2) is the best one. I will use it to forecast the power consumption on the 21st February.

Plot the best NN model with the test data

```r
#to see in details, plot only forecast
plot(prev_nnar2$mean, type = "l", col = "blue", lwd = 2,
     xlab= "Time", ylab = "Power (kW)", main = "Forecasts Vs Actuals")
#and add the actual (test values)
lines(power_test, col = "red", lwd = 1, lty = 2)
#add the legend
legend("topleft", legend = c("Forecast","Actual"),
       col = c("blue", "red"), lty = c(1,2), lwd = c(1,2))
```



Prepare to Forecast the Power kw of the 21st February

```
# I need to create a new ts object with the complete dataset

#set up the start and end dates for the complete dataset until 20th feb
start = as.POSIXct("2010-01-01 01:15:00", tz = "UTC")
end = as.POSIXct("2010-02-20 23:45:00",  tz = "UTC")

#create complete datasets until 20 feb 2010
complete_data = elec %>% filter(Timestamp >= start & Timestamp <= end)

#create time series objects (without temperature)
frequency = 96  # 96 intervals per day (15-minute intervals)
power_complete = ts(complete_data$Power_kW, start = c(1, 6), frequency = frequency)

#then I create a ts for the complete temperature
temp_next = ts(complete_data$Temp_C, start = c(1, 6), frequency = frequency)

#extract the relevant temperature series for the next 96 time points
temp_forecast = tail(complete_data$Temp_C, 96)  # Extract last 96 temperature observations

#I create a ts object for the temperature of the 21 Feb
temp_forecast_ts = ts(temp_forecast, frequency = 96)
```

Train the model using the complete dataset

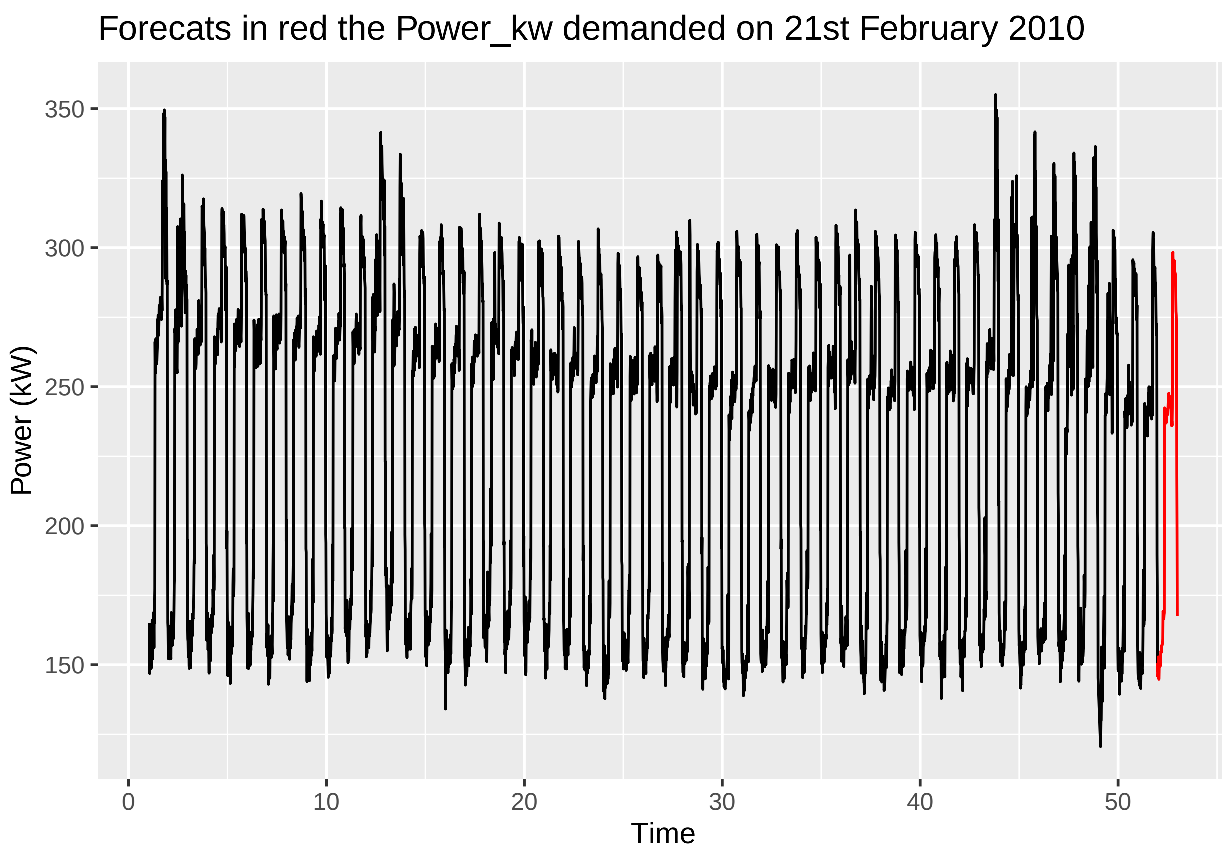
```r
#I train the model using complete power and temperature data (until 20 Feb 2010)
naar_complete = nnetar(
  power_complete,
  p = 15,
  P = 1,
  size = 15,
  repeats = 10,
  lambda = "auto",
  xreg = complete_data$Temp_C
)
```

Forecast the next day: 21st February

```r
#forecast the next 96 time points
nnar_next = forecast(naar_complete, h = 96, xreg = temp_forecast_ts)
```

Plot the Forecast together wtih the complete dataset

```r
# Plot the complete data and forecast
autoplot(power_complete) +
   autolayer(nnar_next$mean, series = "Forecast of 21st Feb with Neural Network", color = "red") +
   ggtitle("Forecats in red the Power_kw demanded on 21st February 2010") +
   ylab("Power (kW)") +
   xlab("Time")
```



Eventually print the Forecast results

```r
print(nnar_next)
```

```
##         Point Forecast
## 52.000       145.6907
## 52.010       150.4935
## 52.021       147.6592
## 52.031       152.8594
## 52.042       152.2445
## 52.052       150.9902
## 52.062       144.7835
## 52.073       147.1291
## 52.083       152.3461
## 52.094       151.2604
## 52.104       151.0461
## 52.115       150.4351
## 52.125       150.3203
## 52.135       149.5579
## 52.146       151.2756
## 52.156       153.8665
## 52.167       154.9969
## 52.177       154.3688
```

```
## 52.188        154.6456
## 52.198        156.0038
## 52.208        157.0506
## 52.219        156.8542
## 52.229        157.2407
## 52.240        157.5332
## 52.250        157.7289
## 52.260        159.3751
## 52.271        169.2357
## 52.281        168.8449
## 52.292        167.4386
## 52.302        166.7923
## 52.312        166.6251
## 52.323        167.9958
## 52.333        168.6610
## 52.344        240.1823
## 52.354        242.4280
## 52.365        238.1170
## 52.375        238.9614
## 52.385        241.2206
## 52.396        238.0453
## 52.406        236.9842
## 52.417        238.8281
## 52.427        236.9497
## 52.438        238.2555
## 52.448        237.7421
## 52.458        239.3256
## 52.469        240.3811
## 52.479        240.1133
## 52.490        239.8259
## 52.500        241.0943
## 52.510        242.2076
## 52.521        242.0155
## 52.531        244.1660
## 52.542        245.3850
## 52.552        245.4353
## 52.562        247.6843
## 52.573        246.2365
## 52.583        244.5160
## 52.594        244.7483
## 52.604        244.3141
## 52.615        244.5518
## 52.625        246.4486
## 52.635        245.1013
## 52.646        245.6720
## 52.656        244.0176
## 52.667        242.6935
## 52.677        240.7766
## 52.688        239.0332
## 52.698        236.4584
## 52.708        235.9669
## 52.719        237.5633
## 52.729        236.0683
## 52.740        242.4798
```

```
## 52.750       275.3447
## 52.760       296.3020
## 52.771       298.3884
## 52.781       294.5101
## 52.792       293.0158
## 52.802       294.3916
## 52.812       293.4311
## 52.823       293.7605
## 52.833       295.4606
## 52.844       293.1883
## 52.854       291.1890
## 52.865       291.3834
## 52.875       291.2492
## 52.885       290.6722
## 52.896       290.5390
## 52.906       289.1686
## 52.917       283.4425
## 52.927       277.1372
## 52.938       275.7809
## 52.948       272.4689
## 52.958       266.4824
## 52.969       218.2557
## 52.979       197.9135
## 52.990       167.6409
```

Export the results in excel

```
#correctly convert forecast power to a data frame
forecast_nnr = as.data.frame(nnar_next)
#extracting just the forecast points
point_forecast_nnr = forecast_nnr[["Point Forecast"]]
#create columns that I will use as 1st column in excel file
nnr_forecast_withtemp = as.data.frame(point_forecast_nnr)


#rename the columns to more descriptive names
colnames(nnr_forecast_withtemp) = c("NNR_With_Temperature")
#save the data frames to an Excel file
write_xlsx(nnr_forecast_withtemp, path = "C:/Users/Utilisateur/OneDrive/Desktop/Romeo_Lorena_NNR_second
```