---
title: "Time Series - EXAM"
author: "Samd Guizani"
date: "`r Sys.Date()`"
output: word_document

Assignment

- File 2023-11-Elec-train.xlsx contains electricity consumption (kW) and outdoor air temperature for a building., measured every 15 minutes, from 1/1/2010 1:15 to 2/20/2010 23:45.

- In addition, outdoor air temperature are available for 2/21/2010. The goal is to forecast electricity consumption (kW) for 2/21/2010.

- Two forecasts should be returned, in one Excel file entitled YourName.xlsx, with exactly two columns (one columns per forecast) and 96 rows:

    1) first one without using outdoor temperature

    2) the second one using outdoor temperature.

Working directory and imports

```{r}
setwd("~/DSTI_MSc DS and AI/02-Foundation/06-Time Series/Exam")

library(readxl)
library(forecast)
library(funtimes) # trend tests
library(ggplot2)
library(imputeTS) # imputing missing data in a time series (interpolation)
library(randomForest)
library(xgboost)
library(e1071) # SVM
library(vars) # VAR model
```

Functions

```{r}
# Forecast Time Series Using a Machine Learning Model

forecast_ML = function(fit, newdata, h)
  #' @description Generates a time series forecast using a machine learning model (e.g., random forest or XGBoost). Iteratively predicts future values based on previous predictions and updates the input data matrix accordingly.
  #'
  #' @param fit A trained machine learning model (e.g., random forest, XGBoost) with a `predict` method.
  #'
  #' @param newdata A matrix of shape (1 x n) used as input to make the initial prediction. The matrix is updated iteratively for subsequent predictions.
  #'
  #' @param h An integer specifying the forecast horizon (number of future steps to predict).
  #'
  #' @return
  #' A numeric vector of length `h` containing the forecasted time series.
{
  prev = rep(NULL, h)
  for (t in 1:h) {
    prev[t] = predict(fit, newdata = newdata)
```

````
    newdata = matrix(c(newdata[-1], prev[t]), 1)
  }
  return(prev)
}

## Root Mean Squared Error

RMSE = function(y_act, y_prd)
{
  return(sqrt(mean((y_act - y_prd)^2)))
}
```

# Load data and explore

## Plot time series and evaluate trends and seasonality patterns

```{r}
data = read_excel('2023-11-Elec-train.xlsx')
data$Timestamp <- as.POSIXct(data[[1]], format = "%m/%d/%Y %H:%M")
data$Timestamp[1] <- as.POSIXct("1/1/2010 1:15", format = "%m/%d/%Y %H:%M") # fix import issue with 1st timestamp.

ts_power = ts(data$`Power (kW)`[1:(dim(data)[1] - 96)], start = 1, freq = 96) # last 96 obs are NA, to be forecasted

autoplot(ts_power)
autoplot(ts_power) + xlim(c(45, 50)) # focus on unusual data
ggseasonplot(ts_power) # seasonal plot with daily period
ggseasonplot(ts(ts_power, freq = 7 * 96)) # seasonal plot with weekly period

ts_temperature = ts(data$`Temp (C°)`, start = 1, freq = 96)
autoplot(ts_temperature)
ggseasonplot(ts_temperature)
```

#### Notes:

-   Power shows a daily and weekly periodic pattern. Possibly a slight decreasing trend. Variance seems constant over time. Unusual zero values on day 49 (i.e. 2/18/2010) and unusual peak of power consumption on day 28 (i.e. 1/28/2010).

-   Temperature shows a daily periodic pattern and an increasing trend.

```{r fig.height=4.5, fig.width=8}
i = 1 + 96 * 0
n_days = 10
ts_temporary = ts(data$`Power (kW)`[i:(i+96*n_days-1)], start = 1+(i-1)/96, freq = 96)
ggseasonplot(ts_temporary) #+ xlim(0.75,1)

```

#### Notes:

-   Power daily pattern is comparable during 6 of the week days, but the 7th day has a specific pattern (earlier decrease on 01/03/2010, 01/10/2010, 01/17/2010, etc...)

## Replace unusual values by interpolation

```{r}
# focus on power values at zero
loc_0s = which(ts_power == 0)
ts_power_impute = ts_power
ts_power_impute[loc_0s] = NA
ts_power_impute = imputeTS::na_interpolation(ts_power_impute, option = 'linear')
````

```
autoplot(ts_power) +
  autolayer(ts_power_impute) +
  xlim(c(45, 50))

ts_power_impute[loc_0s]
```

Notes:

-   Replacing 0 values by interpolated values seems reasonable.

Time series decomposition and differentiating

```{r}
plot(decompose(ts_power_impute)) # daily period
plot(decompose(ts(ts_power_impute, frequency = 7*96))) # weekly period
```

Notes:

-   Decomposing based on **daily** period: still a seasonal pattern in the random series (period of 7 days, i.e. weekly) as well as in the trend component.

-   Decomposing based on **weekly** period: trend component looks smooth with no seasonal pattern. Random component still shows daily pattern (-\> information need to be modeled)

```{r}
ggtsdisplay(diff(ts_power_impute, lag = 96, differences = 1)) # daily period
ggtsdisplay(diff(diff(ts_power_impute, lag = 96, differences = 1),
                 lag = 1,
                 differences = 1)) # daily period + diff with lag 1

ggtsdisplay(diff(ts_power_impute, lag = 7 * 96, differences = 1)) # weekly period
ggtsdisplay(diff(diff(ts_power_impute, lag = 7 * 96, differences = ),
                 lag = 1,
                 differences = 1)) # weekly period + diff with lag 1

```

Notes:

-   Differentiating with a lag = 1 day period: still observe a weekly seasonal pattern (see time series plot)

-   Differentiating twice (with a lag = 1 day period + lag = 1 for de-trending): still observe a weekly seasonal pattern (see time series plot)

-   Differentiating with a lag = 1 week period: periodic pattern no longer observed, but a trend is still visible (see time series plot).

-   Differentiating twice (with a lag = 1 week period + lag = 1 for de-trending): time series centered on 0, no visible trend. ACF/PACF show significant autocorrelation values (-\> information to be modeled)

Modeling, without co-variates

```{r}
# Converting ts_power_impute to daily period
y_daily = ts(ts_power_impute, frequency = 96)
y_daily_train = head(y_daily, length(y_daily) - 96)
y_daily_test = tail(y_daily, 96) # last day kept as test set

# Converting ts_power_impute to weekly period
y_weekly = ts(ts_power_impute, frequency = 7*96)
```

```
y_weekly_train = head(y_weekly, length(y_weekly) - 96)
y_weekly_test = tail(y_weekly, 96) # last day kept as test set
```

Holt-Winters, Daily period

```{r}
# exec_t_start = Sys.time()
#
# fit = hw(y_daily_train, h=96, seasonal = "additive")
# fit |>  summary()
#
# ggtsdisplay(fit$residuals)
# checkresiduals(fit, plot = TRUE)
#
# exec_t_end = Sys.time()
# print(exec_t_end - exec_t_start)
```

Notes:

-   Holt-Winters model fitting fails due to too high frequency (96).

SARIMA (auto), Daily period

```{r}
# Auto SARIMA, daily period
exec_t_start = Sys.time()

fit = auto.arima(y_daily_train)
fit |>  summary()

ggtsdisplay(fit$residuals)
checkresiduals(fit, plot = TRUE)

exec_t_end = Sys.time()
print(exec_t_end - exec_t_start)
```

```{r}
# saveRDS(fit, file = "ARIMA_auto_(5,0,0)(0,1,0)[96].rds")
```

Notes:

-   ACF shows significant autocorrelation at 96 (= 1 day period) and PACF shows exponentially decreasing autocorrelation for daily periods -\> try adding seasonal MA (Q = 1)

-   A trend may still exist -\> try differentiating (d = 1)

-   Some autocorrelation values are significant within the 1st period on ACF and PACF - \> try changing the order p and q

SARIMA (manual), Daily period

```{r}
# SARIMA, daily period
exec_t_start = Sys.time()

fit = Arima(y_daily_train, order = c(5,1,5), seasonal = c(0,1,1))
fit |>  summary()

ggtsdisplay(fit$residuals)
checkresiduals(fit, plot = TRUE)
```

```
exec_t_end = Sys.time()
print(exec_t_end - exec_t_start)
```

```{r}
# saveRDS(fit, file = "ARIMA_man_(5,1,5)(0,1,1)[96].rds")
```

NNetAR, Daily period

```{r}
exec_t_start = Sys.time()

fit = nnetar(y_daily_train)
fit |>  summary()

e = fit$residuals
print(paste0("Train RMSE: ", sqrt(mean(e^2, na.rm = TRUE))))
ggtsdisplay(fit$residuals)
checkresiduals(fit, plot = TRUE)

exec_t_end = Sys.time()
print(exec_t_end - exec_t_start)
```

```{r}
# saveRDS(fit, file = "NNetAR_daily.rds")
```

SARIMA (auto), Weekly period

```{r}
# Auto ARIMA, weekly period
exec_t_start = Sys.time()

fit = auto.arima(y_weekly_train)
fit |>  summary()

ggtsdisplay(fit$residuals)
checkresiduals(fit, plot = TRUE)

exec_t_end = Sys.time()
print(exec_t_end - exec_t_start)
```

```{r}
# saveRDS(fit, file = "ARIMA_auto_(5,1,2)(0,1,0)[672].rds")
```

SARIMA (manual), Weekly period

```{r}
# # ARIMA, weekly period
# exec_t_start = Sys.time()
#
# fit = Arima(y_weekly_train, order = c(5,1,2), seasonal = c(0,1,1))
# fit |>  summary()
#
# ggtsdisplay(fit$residuals)
# checkresiduals(fit, plot = TRUE)
#
# exec_t_end = Sys.time()
# print(exec_t_end - exec_t_start)
```

```{r}
# saveRDS(fit, file = "ARIMA_auto_(5,1,2)(0,1,1)[672].rds")
```

NetAR, Weekly period

```{r}
exec_t_start = Sys.time()

fit = nnetar(y_weekly_train)
fit |>  summary()

e = fit$residuals
print(paste0("Train RMSE: ", sqrt(mean(e^2, na.rm = TRUE))))
ggtsdisplay(fit$residuals)
checkresiduals(fit, plot = TRUE)

exec_t_end = Sys.time()
print(exec_t_end - exec_t_start)
```

```{r}
# saveRDS(fit, file = "NNetAR_weekly.rds")
```

```{r}

# x = co2
#
# forecastfunction = function(x, h){forecast(Arima(x, order=c(1,0,0)), h=h)}
#
# e = tsCV(co2, forecastfunction, h = 12, window = 5, xreg = NULL, initial = 150)



# #Fit an AR(2) model to each rolling origin subset
# far2 <- function(x, h){forecast(Arima(x, order=c(2,0,0)), h=h)}
# e <- tsCV(lynx, far2, h=1)
#
# #Fit the same model with a rolling window of length 30
# e <- tsCV(lynx, far2, h=1, window=30)
#
# #Example with exogenous predictors
# far2_xreg <- function(x, h, xreg, newxreg) {
#   forecast(Arima(x, order=c(2,0,0), xreg=xreg), xreg=newxreg)
# }
#
# y <- ts(rnorm(50))
# xreg <- matrix(rnorm(100),ncol=2)
# e <- tsCV(y, far2_xreg, h=3, xreg=xreg)


```

```{r}
# autoplot(window(fit$fitted, start = 30)) + autolayer(window(ts_power_impute, start =
30))
```

```{r}
# exec_t_start = Sys.time()
#
# forecastfunc = function(x, h)
# {
```

````
#   forecast(Arima(x, order=c(0,1,0), seasonal=c(0,1,0)),
#             h=h,
#             window = 2 * 96 * 7, # condider 2 weeks history to build a model
#             initial = 3000)
# }
#
# e = tsCV(y_daily_train, forecastfunc, h=96)
#
# exec_t_end = Sys.time()
# print(exec_t_end - exec_t_start)
```

## ML data prep

```{r}
# next observation based on last day
df_daily = as.vector(y_daily_train)[1:(96+1)]
for (i in 1:(length(y_daily_train)-(96+1)))
{
  df_daily = rbind(df_daily, as.vector(y_daily_train)[(i+1):(i+96+1)])
}

# next observation based on last week
df_weekly = as.vector(y_weekly_train)[1:(7*96+1)]
for (i in 1:(length(y_weekly_train)-(7*96+1)))
{
  df_weekly = rbind(df_weekly, as.vector(y_weekly_train)[(i+1):(i+7*96+1)])
}

# next 96 observations based on 2 last week
df_2weeks = as.vector(y_weekly_train)[1:(2*7*96+96)]
for (i in 1:(length(y_weekly_train)-(2*7*96+96)))
{
  df_2weeks = rbind(df_2weeks, as.vector(y_weekly_train)[(i+1):(i+2*7*96+96)])
}
```

## ML - Random Forest, Daily period

```{r}
exec_t_start = Sys.time()

fit = randomForest(x = df_daily[,-(96+1)], y = df_daily[, (96+1)])
fit |>  summary()

e = ts(fit$y - fit$predicted, start = c(1,1), frequency = 96)
print(paste0("Train RMSE: ", sqrt(mean(e^2, na.rm = TRUE))))
ggtsdisplay(e)
checkresiduals(e, plot = TRUE)

exec_t_end = Sys.time()
print(exec_t_end - exec_t_start)

# saveRDS(fit, file = "RF_daily.rds")
```

## ML - XGBoost, Daily period

```{r}
exec_t_start = Sys.time()

fit = xgboost(data = df_daily[,-(96+1)], label = df_daily[, (96+1)],
              max_depth = 10,
              eta = 0.5,
              nrounds = 100,
````

````
              objective = "reg:squarederror")
fit |>  summary()

e = ts(df_daily[, (96+1)] - predict(fit, newdata = df_daily[,-(96+1)]), start = c(1,1),
frequency = 96)
print(paste0("Train RMSE: ", sqrt(mean(e^2, na.rm = TRUE))))
ggtsdisplay(e)
checkresiduals(e, plot = TRUE)

exec_t_end = Sys.time()
print(exec_t_end - exec_t_start)

# saveRDS(fit, file = "XGBoost_daily.rds")
```

## ML - Random Forest, Weekly period

```{r}
exec_t_start = Sys.time()

fit = randomForest(x = df_weekly[,-(7*96+1)], y = df_weekly[, (7*96+1)])
fit |>  summary()

e = ts(fit$y - fit$predicted, start = c(1,1), frequency = 96)
print(paste0("Train RMSE: ", sqrt(mean(e^2, na.rm = TRUE))))
ggtsdisplay(e)
checkresiduals(e, plot = TRUE)

exec_t_end = Sys.time()
print(exec_t_end - exec_t_start)

# saveRDS(fit, file = "RF_weekly.rds")
```

## ML - XGBoost, Weekly period

```{r}
exec_t_start = Sys.time()

fit = xgboost(data = df_weekly[,-(7*96+1)], label = df_weekly[, (7*96+1)],
              max_depth = 10,
              eta = 0.5,
              nrounds = 100,
              objective = "reg:squarederror")
fit |>  summary()

e = ts(df_weekly[, (7*96+1)] - predict(fit, newdata = df_weekly[,-(7*96+1)]), start =
c(1,1), frequency = 96)
print(paste0("Train RMSE: ", sqrt(mean(e^2, na.rm = TRUE))))
ggtsdisplay(e)
checkresiduals(e, plot = TRUE)

exec_t_end = Sys.time()
print(exec_t_end - exec_t_start)

# saveRDS(fit, file = "XGBoost_weeky.rds")
```

## ML - PLS, 2 weeks history to forecast next day

```{r}
library(pls)
```

```{r}
````

```
exec_t_start = Sys.time()

fit = plsr(df_2weeks[,(2*7*96+1):(2*7*96+96)] ~ df_2weeks[,1:(2*7*96)],
           scale = TRUE,
           validation = "CV")
fit |>  summary()

# Cross-validation results
validation_mse <- fit$validation$PRESS
avg_mse <- colMeans(validation_mse)

# Optimal components minimizing average MSE
optimal_ncomp <- which.min(avg_mse)

e = fit$residuals[,,optimal_ncomp]
print(paste0("Train RMSE: ", sqrt(mean(e^2, na.rm = TRUE))))
# ggtsdisplay(e)
# checkresiduals(e, plot = TRUE)

exec_t_end = Sys.time()
print(exec_t_end - exec_t_start)

# saveRDS(fit, file = "PLS_2weeks.rds")
```

Notes:

- PLS: Very long fitting time (approx. 6h). And produces a huge model object (+11 GB)

- Train RMSE = 8.41923 (optimal_ncomp = 144)

Model performance comparison on test set

Models based on daily period

```{r}
# Build a list of models
models_list = list()
models_list$SARIMA_500_010_96 = readRDS('ARIMA_auto_(5,0,0)(0,1,0)[96].rds')
models_list$SARIMA_515_010_96 = readRDS('ARIMA_man_(5,1,5)(0,1,1)[96].rds')
models_list$NNetAR_daily = readRDS('NNetAR_daily.rds')
models_list$RF_daily = readRDS('RF_daily.rds')
models_list$XGBoots_daily = readRDS('XGBoost_daily.rds')

# Make predictions with each model and store RMSE
previsions_list = list()
rmsep_list = list()
horizon = 96
newdata_ML = tail(y_daily_train, horizon)

for (name in names(models_list))
{
  cat(paste0("Forcasting model:", name, "\n"))

  if(grepl("RF", name)
     | grepl("XG", name)) # Use forecast_ML function with ML models
  {
    prevision = forecast_ML(models_list[[name]], newdata = matrix(newdata_ML,1), horizon)
    prevision = ts(prevision, start = c(50,92), frequency = 96)
  }
  else # Use forecast function with ts models
  {
    prevision = forecast(models_list[[name]], h = horizon)
    prevision = prevision$mean
  }
```

````
  previsions_list[[name]] = prevision
  rmsep_list[[name]] = RMSE(y_daily_test,prevision)

  cat(paste0("Test set RMSE: ", rmsep_list[[name]], "\n\n"))
}

# Plots
autoplot(y_daily_test) +
  autolayer(previsions_list$SARIMA_500_010_96) +
  autolayer(previsions_list$SARIMA_515_010_96) +
  autolayer(previsions_list$NNetAR_daily) +
  autolayer(previsions_list$RF_daily) +
  autolayer(previsions_list$XGBoots_daily)
```

### Models based on weekly period

```{r}
# Build a list of models
models_list = list()
models_list$SARIMA_512_010_672 = readRDS('ARIMA_auto_(5,1,2)(0,1,0)[672].rds')
models_list$NNetAR_weekly = readRDS('NNetAR_weekly.rds')
models_list$RF_weekly = readRDS('RF_weekly.rds')
models_list$XGBoots_weekly = readRDS('XGBoost_weeky.rds')

# Make predictions with each model and store RMSE
previsions_list = list()
rmsep_list = list()
horizon = 96
newdata_ML = tail(y_weekly_train, 7*horizon)

for (name in names(models_list))
{
  cat(paste0("Forcasting model:", name, "\n"))

  if(grepl("RF", name)
     | grepl("XG", name)) # Use forecast_ML function with ML models
  {
    prevision = forecast_ML(models_list[[name]], newdata = matrix(newdata_ML,1), horizon)
    prevision = ts(prevision, start = c(8,92), frequency = 7*96)
  }
  else # Use forecast function with ts models
  {
    prevision = forecast(models_list[[name]], h = horizon)
    prevision = prevision$mean
  }
  previsions_list[[name]] = prevision
  rmsep_list[[name]] = RMSE(y_weekly_test,prevision)

  cat(paste0("Test set RMSE: ", rmsep_list[[name]], "\n\n"))
}

# Plots
autoplot(y_weekly_test) +
  autolayer(previsions_list$SARIMA_512_010_672) +
  autolayer(previsions_list$NNetAR_weekly) +
  autolayer(previsions_list$RF_weekly) +
  autolayer(previsions_list$XGBoots_weekly)
```

```{r}
fit = readRDS("PLS_2weeks.rds")

cat(paste0("Forcasting model:", "PLS_2weeks", "\n"))
````

```
prevision = predict(fit, newdata = t(as.data.frame(tail(y_weekly_train, 2*7*96))), ncomp = 144)
prevision = ts(prevision[1,,1], start = c(8,92), frequency = 7*96)

RMSE_PLS_2weeks = RMSE(y_weekly_test,prevision)
cat(paste0("Test set RMSE: ", RMSE_PLS_2weeks, "\n\n"))

autoplot(y_weekly_test) +
  autolayer(prevision)
```

Modeling, with co-variates

Correlation power vs. temperature

```{r}
plot(data$`Temp (C°)`, data$`Power (kW)`)
abline(lm(data$`Power (kW)`~data$`Temp (C°)`), col="red")

data_impute = data
data_impute[1:length(ts_power_impute), 2] = ts_power_impute
plot(data_impute$`Temp (C°)`, data_impute$`Power (kW)`)
abline(lm(data_impute$`Power (kW)`~data_impute$`Temp (C°)`), col="red")

print(paste0("Correlation coef. Power vs. Temp = ", cor(data_impute$`Power (kW)`, data_impute$`Temp (C°)`, use = "complete.obs")))
```

Notes:

- A correlation exists between Power and Temperature (higher Power is observed when Temperature increases).