# The challenge: ACEA Smart Water Analytics Competition

The goal of this challenge is to predict water levels in a collection of different water bodies based in Italy. Specifically we have to predict based on a time series model, to accurately assess the water level of tomorrow, based on data of today. To shortly note the Kaggle rules, this is an analytics challenge, which means creating a compelling story & notebook is a very important part. My notebook is publicly available on Kaggle here, but I will work through some code excerpts and interesting highlights in these blogs.

So far we have discussed the challenge more generally, looked at some data wrangling and new features for modelling. Last time we overcame a classic issue with time series modelling concerning our cross-validation. This week we will finish off our modelling and how we finish off the challenge for handin. You can find the blogs below if you want to read back first:

- Challenge goals and initial discussion
- Data setup for modelling
- Data modelling
- Final modelling and discussion

# ACEA Smart Water Analytics Competition Final model

Last blog was a more indepth discussion of hindsight bias in the cross-validation stage of the modelling. By using specific a specific time series method we stabilized our model. In the week after I stumbled upon a work from Hanna Meyer. She is a machine learning expert in the field of environmental data. There is a great research paper she wrote available here (https://www.sciencedirect.com/science/article/abs/pii/S1364815217310976?via%3Dihub)

Studying her work made me realise that the Kaggle dataset has all the characteristics she discussed in her paper. She discusses indepth both the time series aspect covered by Hyndman, but also the spatial element of the dataset. In this case having multiple measurement points in the dataset that measure the water level. The main advantage of setting up the model in this way is that it is more sensitive to unknown locations within the given water body. On top of that it allows for simultaneous modelling of all given locations.

## Leave Location and Time Out (LLTO) modelling

So in the final model I introduce both a spatial element and a time series element in my model. Methodologically this is called LLTO (Leave Location and Time Out) Cross validation. Essentially the idea combines all earlier discussed steps.

In your training set you leave out the location that is in your validation dataset, this covers the spatial element. You then only include timepoints from before the time series in the validation dataset, this covers the time element. If you have 4 locations you have 4 folds per timeperiod. Each time one of the locations is placed in the validation dataset. This method is implemented in the CAST package in R. However I found that the time aspect is actually not handled properly in the relevant function (CreateSpaceTimeFolds). Hence I ended up making my own folds that respect both aspects.

## R-Code

In the code below there is an example function that handles all these steps. Specifically of interest may be lines 47-101, this is where the handmade folds in the cross-validation are

created. I have not perfected this code, but it shows the main steps accordingly. If you want to know more about this or discuss then dont hesitate to contact me. We might get back to that later in a different blog.

```
model_handmade_folds <- function(data, horizon, dataset, lag, features,
basefeatures){

#basefeatures <- 'Depth'

# Make lags:
features <- grep(features,names(data),value = T)
basefeatures <- grep(basefeatures,names(data),value = T)

for(i in 1:length(features)){
    for(j in 1:lag){
        data$temp <- Lag(data[,features[i],+j])
        names(data)[which(names(data)=='temp')] <- paste(features[i],j,
sep = '_')
    }
}


data <- data[,which(colMeans(!is.na(data))>.2)]

# Inlude seasonality:
data$year <- as.numeric(substr(data$Date,7,10))
data$year <- data$year - min(data$year) + 1
data$month <- as.numeric(substr(data$Date,4,5))
data$quarter <- ifelse(data$month <= 3,1,
                           ifelse(data$month >=4 & data$month <= 6,2,
                               ifelse(data$month >=7 & data$month <=
9,3,
                                   ifelse(data$month >9,4,NA))))

data_long <- tidyr::pivot_longer(data, basefeatures,names_to =
'location', values_to = 'depth_to_groundwater')

data_long <- data_long[complete.cases(data_long),]
data_long <- data_long[which(data_long$depth_to_groundwater != 0),]

#data_model <- data_long[,-grep('location|Date|name',names(data_long))]

temp <- data_long[,which(!names(data_long)%in%c('depth_to_
groundwater','Date','location'))]
nzv <- nearZeroVar(temp)
# excluding variables with very low frequencies
if(length(nzv)>0){temp <- temp[, -nzv]}
i <- findCorrelation(cor(temp))
# excluding variables that are highly correlated with others
if(length(i) > 0) temp <- temp[, -i]
i <- findLinearCombos(temp)
# excluding variables that are a linear combination of others
```

```
if(!is.null(i$remove)) temp <- temp[, -i$remove]
data_model <- data_long[,c('depth_to_groundwater','Date','location',
names(temp))]

data_model$Date <- as.Date(as.character(data_model$Date), format =
'%d/%m/%Y')

# Handmade indexes:
index_hand_design <- function(data,period, location, horizon,
location_one = NULL){
    horizon2 <- max(period)-horizon
    if(!is.null(location_one)){
    indexin <- which(data$Date >= min(period) & data$Date <= horizon2)
    indexout <- which(data$Date > horizon2 & data$Date <= max(period))

    } else {
    indexin <- which(data$Date >= min(period) & data$Date <= horizon2 &
data$location != location)
    indexout <- which(data$Date > horizon2 & data$Date <= max(period) &
data$location == location)
    }
    output <-c(list(indexin),list(indexout))
    output
}

periods <- round(length(seq.Date(from = min(data_model$Date),to =
max(data_model$Date), by = 'day'))/horizon,0)
dates   <- seq.Date(from = min(data_model$Date),to =
max(data_model$Date), by = 'day')
indices <- 1:periods*horizon

periods_final <- dates[indices]
periods_final <- periods_final[!is.na(periods_final)]

stopifnot(length(periods_final)>=4)

for(i in 3:length(periods_final)){
    output <- list(c(periods_final[i-2], periods_final[i]))
    if(i <= 3){
        output_final <- output
    } else {
        output_final <- c(output_final, output)
    }
}

locations <- unique(data_model$location)

for(i in 1:length(locations)){
    for(j in 1:length(output_final)){
        if(length(locations)==1){

        output_temp <- index_hand_design(data_model,output_final[[j]],
```

```
locations[i], horizon, location_one = 'yes')
        } else {
        output_temp <- index_hand_design(data_model,output_final[[j]],
locations[i], horizon)
        }
        if(j == 1){
            final_inner <- output_temp
        } else {
            final_inner <- c(final_inner, output_temp)
        }
    }
    if(i == 1){
        final <- final_inner
    } else {
        final <- c(final, final_inner)
    }
}

index_final <- list(index = final[seq(1, length.out =
length(locations)*length(output_final), by = 2)],
                    indexOut = final[seq(2, length.out
=length(locations)*length(output_final), by = 2)])


fitcontrol <- trainControl(verboseIter = T,
                           index = index_final$index,
                           indexOut = index_final$indexOut)

gbmGrid <-  expand.grid(interaction.depth = c(1,2,4),
                           n.trees = 1:4000,
                           shrinkage = c(0.01),
                           n.minobsinnode = c(2,5))

if(length(locations)>1){
for(i in 1:length(locations)){
    data_model$temp <- ifelse(data_model$location == locations[i],1,0)
    names(data_model)[which(names(data_model) == 'temp')] <-
paste(locations[i],'ind', sep = '_')
}
}

err <- try(load(paste(maindir, modeldir, paste('data_model =
',dataset,'horizon =',horizon,'.RData', sep = ''), sep = '/')))
if(err != 'train'){

    train <- train(depth_to_groundwater ~ . , method = 'gbm', trControl
= fitcontrol, tuneGrid = gbmGrid,
                       data = data_model[,-grep('Date|
location',names(data_model))])
    save(train, file = paste(maindir, modeldir, paste('data_model = ',
dataset,'horizon =', horizon,'.RData', sep = ''), sep = '/'))
}
```

```
return(train)
}

train_auser <- model_handmade_folds(data_auser, 365, 'auser',15,
                      'Rainfall|Temperature|Flow_
Rate|Volume|Hydrometry|Depth', 'Depth')

train_petrig <- model_handmade_folds(data_petrignano, 365,
'petrignano',15,
                      'Rainfall|Temperature|Flow_
Rate|Volume|Hydrometry|Depth', 'Depth')

train_amiata <- model_handmade_folds(data_amiata, 365, 'amiata',15,
                      'Rainfall|Temperature|Flow_
Rate|Volume|Hydrometry|Depth', 'Flow_Rate')

train_lupa <- model_handmade_folds(data_lupa, 365, 'lupa',15,
                      'Rainfall|Temperature|Flow_
Rate|Volume|Hydrometry|Depth', 'Flow_Rate')

train_arno <- model_handmade_folds(data_arno, 365, 'arno',15,
                      'Rainfall|Temperature|Flow_
Rate|Volume|Hydrometry|Depth', 'Hydrometry')

train_bila_flo <- model_handmade_folds(data_bilancino, 365,
'bilancino_flo',15,
                      'Rainfall|Temperature|Flow_
Rate|Volume|Hydrometry|Depth', 'Flow_Rate')

train_bila_hyd <- model_handmade_folds(data_bilancino, 365,
'bilancino_hyd',15,
                      'Rainfall|Temperature|Flow_
Rate|Volume|Hydrometry|Depth|Lake_Level', 'Lake_Level')
```

## ACEA Smart Water Analytics Competition; final thoughts

In doing this challenge I've ended up putting a lot of focus on the modelling stage. I did learn a lot going through all these steps on different data then I usually work on. When I look over the final model I am happy with the outcome of the project. The main reason is that it can generalize well to new locations of water level measurement points and is robustly designed for time series effects. Overall I feel that hints back to the original spirit of the challenge.

There are some improvements to be made, for example the model failed on some data sets due to lack of usable data. I could fix this by looking at imputation methods, an area I have skipped completely during the project. I might revisit that later as needed, as the dataset provided in the ACEA Smart Water Analytics Competition contains alot of missing data points.

Furthermore the true applicability of this model is still to be determined. I made some assumptions, namely that we want to predict next-day measurements, when ACEA might be interested in multiple steps ahead predictions. The model is also more generalized, this results in easier applicability on new and unknown datasets, as well as new locations. But from a business standpoint it can also be logical to focus on optimizing currently known locations more

thoroughly.  It definitely interests me how subtle changes in the data setup and modelling approach can lead to completely different use cases from a business perspective.