Acea Smart Water Prediction Review

* My Method- Biggest Challenge to me: very dirty data;
* Marco Marchetti- [Acea Smart Water Model](https://www.kaggle.com/marcomarchetti/acea-smart-water-model)
* Alberto Lopez- [Water supplies forecasting with neural networks](https://www.kaggle.com/albertoj6/water-supplies-forecasting-with-neural-networks)

很短model也很漂亮, 可以細看他怎麼樣把fourier series公式放寬利用neuralnetwork去學習參數, 並讓基因演算法去學習hyperparameters做prediciton. (def geneticAlgorithm() in the code)

Transform data into **weekly** frequency via mean (code: df.resample(rule=’w’).mean()). Next, split data into train/val半年/test半年, then use geneticAlgorithm() to repeatedly fine tune the hyperparameters. (In each round 50 models were kept and paired with replacement to generate 25 children, then 50 out of 50+25=75 models will be kept for the next round via val\_loss on val set.) Finally, the best model will be returned via final evaluation of last 50 models on the score E = train\_score\*.25 + val\_score\*.25 + test\_score\*.5.

表現不錯, 比我們的VAR MAE&RMSE還要好。

缺點: 1. 仍應該要先make dataset stationary.

* Ankur Chaudhari- [Fork of Acea\_all](https://www.kaggle.com/ace526/fork-of-acea-all)

EDA section只是看圖說故事.

Data Preprocessing: 先切掉大量涵蓋NA的過去資料, 接下來仍然含超過20%NA的變量直接踢除, 其餘則用[LightGMB](https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/) 透過新創造的”時間\_sin/cos”來補值，補完後用ADF檢測stationary 但並未做stationary…而是直接砸LSTM.

FE: for all features, create its 7D rolling mean feature correspondingly.

看起來是1-step-ahead forecasting task with LSTM (多期input, 單一值output).

優點1:

def gradient\_importance(seq, model): #model can be LSTM

seq = tf.Variable(seq, dtype=tf.float32)

with tf.GradientTape() as tape:

predictions = model(seq)

grads = tape.gradient(predictions, seq)

grads = tf.reduce\_mean(grads, axis=1).numpy()[0]

return grads

優點2:

LSTM train的很好, 雖然1-step-ahead 我們VAR更好, 但相信他的LSTM在連續多期預測上會表現優異(雖然沒實作)

缺點1:

僅使用LSTM model 單一形式, 並未測試其他更多其他形式。且data也未做stationary, 雖然可能LSTM沒對此這麼要求, 但仍應做個實驗才對。

缺點2:

補值前並未先將testing data hold out, 導致future data leakage 🡪 作弊。

* Yana Malysheva- [Modeling the physical properties of a water system](https://www.kaggle.com/yanamal/modeling-the-physical-properties-of-a-water-system)

建Causal Graph, 強調並假設rainfall effect on water body on day t = .也提供詳細物理解釋(4參數: fraction\_retained, first\_day\_flow, funnel\_start\_width, time\_gap)與為何選該模型: The fraction of water lost each day creates the exponential decay effect, and the funnel/watershed model creates the linear ramp-up. The funnel shape affects how steep the exponential decay is: the effect is more drawn out with a wider base. The time lag, of course, delays the entire effect.

1. No imputation of data needed, except in some cases it may be helpful to impute rain - otherwise, the method is robust to missing data. (rolling後直接dropna)
2. Hydrometry = river flow rate
3. 二階段ML模型:   
   Rainfall + Temperature 🡪 River Hydrometry;   
   River Hydrometry + R + T 🡪 (1st diff) Depth2Groundwater
4. Q:為何二階段?   
   A: 因為預測時看不到Rainfall, Temperature, and River H; 所以用7D\_rollingMean歷史資料來取代R, and T(每新看到一筆test datapoint就移動視窗更新資料計算移動平均). 至於H, 則用Stage 1 model來預測. 以上做完後就有資料可以做Stage2預測.  
   A2: 二階段模型的另一個好處是可解釋性很強. 甚至當ML學得的 fraction\_retained參數很小時, 整個二階模型可以直接化簡為LinearReg.
5. Model結果雖不可能預測瞬時變化(eg., 1D,7D) , 但預測趨勢線倒是很準確.



* Aspiring Kaggler- [Water Prediction w/ LSTM, XGB, Attention](https://www.kaggle.com/aspiringkaggler/water-prediction-w-lstm-xgb-attention/)

預測連續30天. Test set=365D.

公式:

: rainfall 但會延遲或蒸發或其他用途而減少

flow\_rate(flux) 和湖泊面積等(以估計蒸發速度), 但還有non-linear情況如觀光客,節慶,水壩開關, 故不可能寫下公式做理論模型, 而改用ML.

用過KNN, XGB, FNN, LSTM, SVM, 當中XGB和LSTM最好故下文著重這兩者:

1. 補值用interpolation linear/quadratic看哪個在圖上更合適. 看到Groundwater從100跳到120覺得不合理就改NA再補值.
2. Dummy: 月份; Temperature>25?
3. 一樣用1st difference 做預測

Pro: 1.投機-有些要預測的Variable, 因為跟其他的歷史很像所以就取其中一個預測並做為其他的代表就好. 2. Temp>25 dummy. 3. 每一個補值細節步驟都解釋. 4.看一些paper有做功課

Con: 有些認為不合理就改nan再插值, 問題是這就是該變量唯一有的variance, 拿掉的話你預測當然準啊; LSTM表現差; Code很糟; 視覺畫圖表很亂

English Comment: Explains everything in detail. Read some papers and cite them. Use Temperature>25 dummy. Smart to predict only one target variable instead of many if they possess the same historic pattern. Shitty code, shitty graph, shitty LSTM performance. Shitty imputing reasoning esp. on jump values of a feature (what if that’s the only variance of the feature but you impute it?!)