Milano-Bicocca University
AA 2021/2022
JANUARY EXAM SESSION

STREAMING DATA MANAGEMENT & TIME SERIES ANALYSIS FINAL PROJECT

TITLE:

Prediction of a univariate time series of hourly measurements of carbon monoxide (CO)

AUTHOR:

Emanuele Marnati

UNIVERSITY ID:

812503

ABSTRACT

Carbon monoxide is undoubtedly a dangerous pollutant. The ability to predict its behaviour could be an essential factor for politicians or decision makers. The aim of the following project is to study a CO measurement time series, with an hourly frequency ranging from 2004-03-10 (hour=18) to 2005-02-28 (hour=23) in order to predict its CO value over the period ranging from 2005-03-01 (hour=0) to 2005-03-31 (hour=23).
To achieve this objective, after an initial preprocessing phase which consists in the integration of null values, three different methodologies were used: ARIMA, UCM (linear models) and non-linear models Recurrent Neural Network based (LSTM and GRU).
The models built were evaluated in terms of MAPE (Mean Absolute Error) on the validation set and then the best models for each category were selected. Lastly the forecasts for the following 31 days (of this series) were produced. Because of technical issues, for a greater computational flexibility and to exploit the strengths of both approaches, it was decided to use both R with the local Rstudio IDE (for Arima and UCM) and Python in the Colab environment (for ML models).

PREPROCESSING

FILL IN NULL VALUES

An analysis of the data shows that there are 365 zero hour values over 8526, 4.28% of the dataset. Furthermore, the annual series is not complete as the first 10 days of March are missing, but this last one is only an inconvenience, not a problem (moreover the series starts at 18.00 of the day not at 00.00).
To integrate the 365 values, their position is first assessed: april 2004 presents 3 days with missing values, may 2 days, june 3 days, july only 1 day, august 4 days, september 2 days, october 1 day, december 5 day, january 2005 presents 6 and february 4 days respectively. To integrate the series there are many ways that vary from basic to complex imputation. In Figure 1 there's an overview of the hourly missing values.

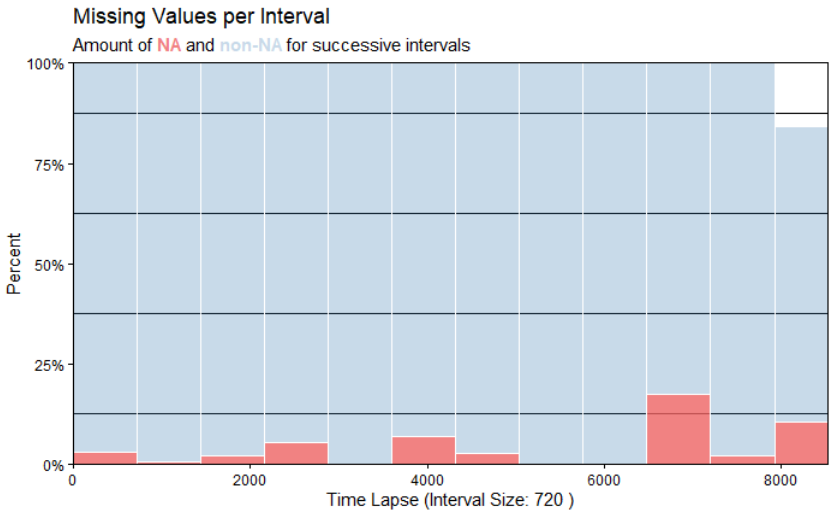

Figure 1

A classic approach of comparing different methodologies was used to make the missing data imputation: linear interpolation, exponential moving average and Seasonally Decomposed Missing Value Imputation were assessed. In order to understand which method was the best for imputation, I divided the data into train and test and validated the models for only one week (the last week of February, which has no missing values). The method chosen is at the end the last one (na.seadec). This method removes the seasonal component from the time series, performs imputation on the deseasonalized series and afterwards adds the seasonal component again. The table 1 below shows the results of the comparison based on different metrics such as RMSE and MAPE. The RMSE is chosen because it is recognised in the literature as a robust comparison measure while the MAPE is chosen because it is directly interpretable and describable. It is then decided to proceed with the third method as a 12.6% error is acceptable and the best MAPE I have.

MODEL/MAPE %	linear	exponential MA	seadec
RMSE	163.083	164.801	153.373
MAPE	13.8	14.0	12.6

Table 1

EXPLORATIVE ANALYSIS

It is now very useful to have an initial view of the data and though to do this it is necessary to plot the complete series to analyse the overall trend (Figure 2).

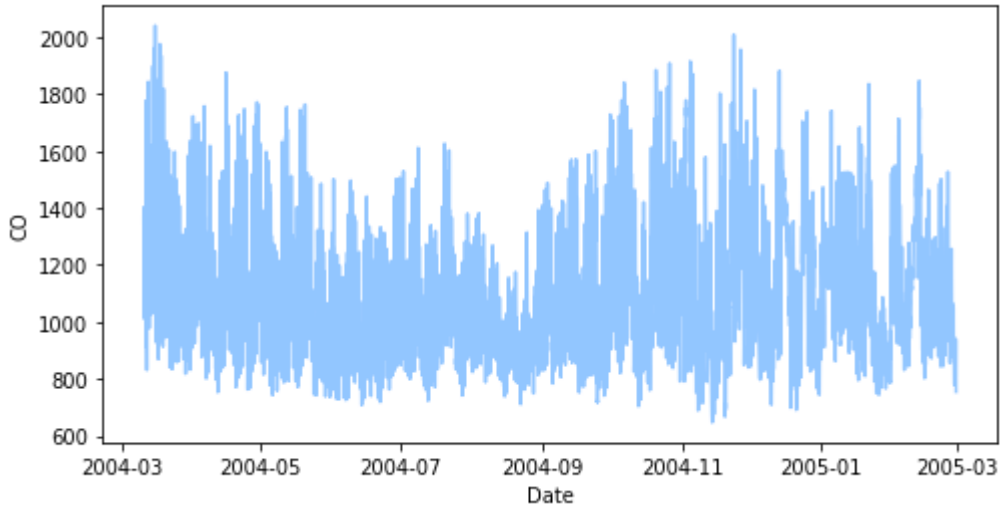

Figure 2

At a glance, there aren't any notable peaks. But the correlation between the daily mean value and the daily standard deviation is high (> 70%), therefore I have decided to use the box cox transformation to handle it. This transformation is based on a mathematical normalization on a lambda value: it allows to have more accurate models and therefore all the insights related should also gain better performance (later showed).

Analysing the patterns decomposed on different granularities as in the Figure below shows that there is a mixed general trend, on a daily basis the shape is an “up and down” line. What surprises me the most is that the weekly trend is quite regular. The seasonal decomposition in Figure 3 gives important information about the possible seasonality. It is clear that the 2 seasonality to take into account are the daily and the weakly one.

From now on, I proceed to divide the data into a training set (80% of the observations) and a validation set (20% of the observations).The validation set is used to evaluate the generalization capacity of each predictive model, and it was not taken into account for the lambda computation.

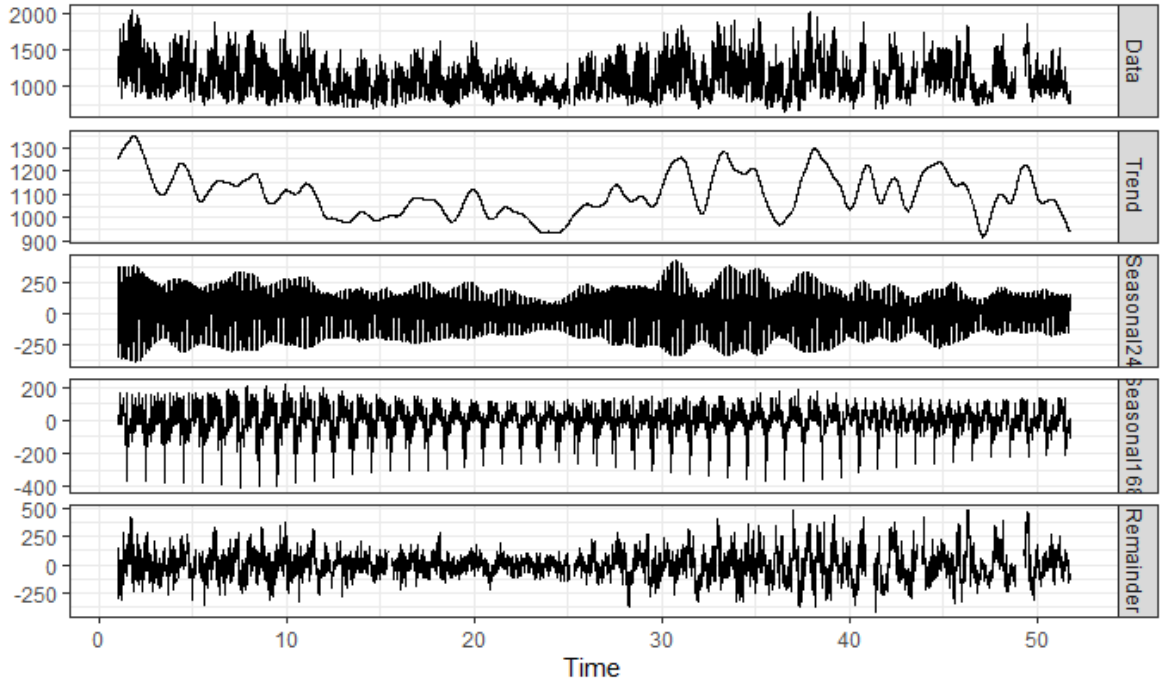

Figure 3

LINEAR MODELS

The first step consists in assigning a frequency to the historical series, looking at ACF and PACF it becomes immediately clear that the frequency is 24 (Figure 4). Although it would have been appropriate to also analyse the weekly seasonality as well, which is however more complex to interpret and manage.

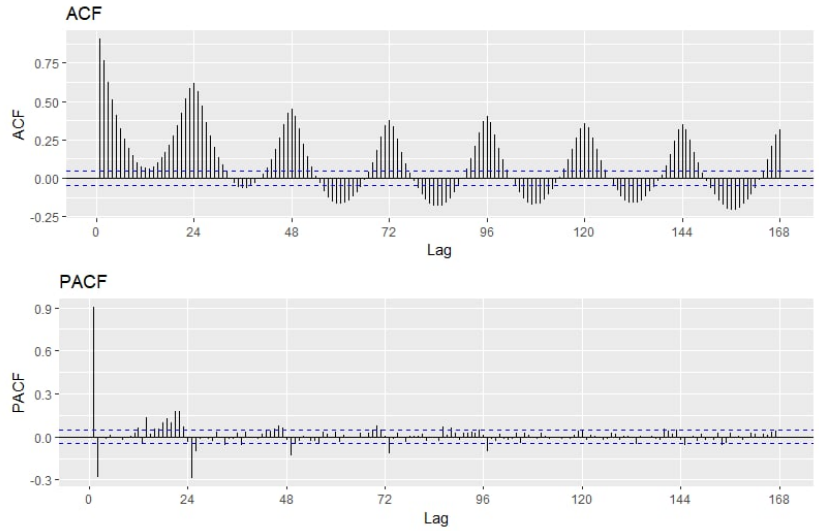

Figure 4

The second step is to make the series stationary in variance. I use the box and cox method as said in the explanatory phase. I first computed the lambda value that is very close to -0.999. I therefore decided to apply a seasonal difference based on lag 24 and a first degree simple difference: after the first seasonal difference the Ljung-Box test Null Hypothesis is rejected, after the second simple difference, the test is accepted, now the series seems to be without seasonality. It was also performed a Dikcey-Fuller test, which led to the rejection of the null hypothesis. The graph in Figure 5 proves stationarity although with this real world value it remains some background noise.

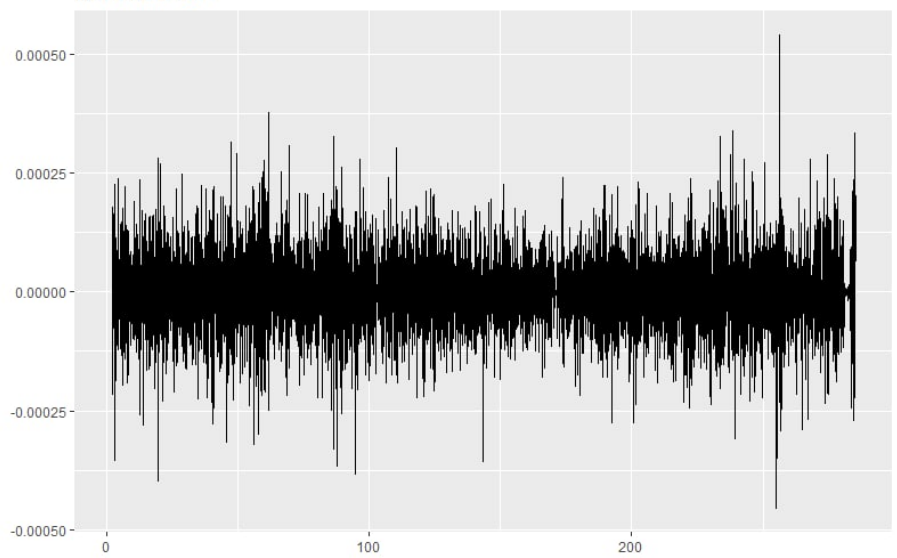

Figure 5

ARIMA

The first family of estimated models are the ARIMAs. The estimation of the ARIMA models was carried out according to the Box-Jenkins procedure, starting from this ACF- PACF plot in Figure 4 .

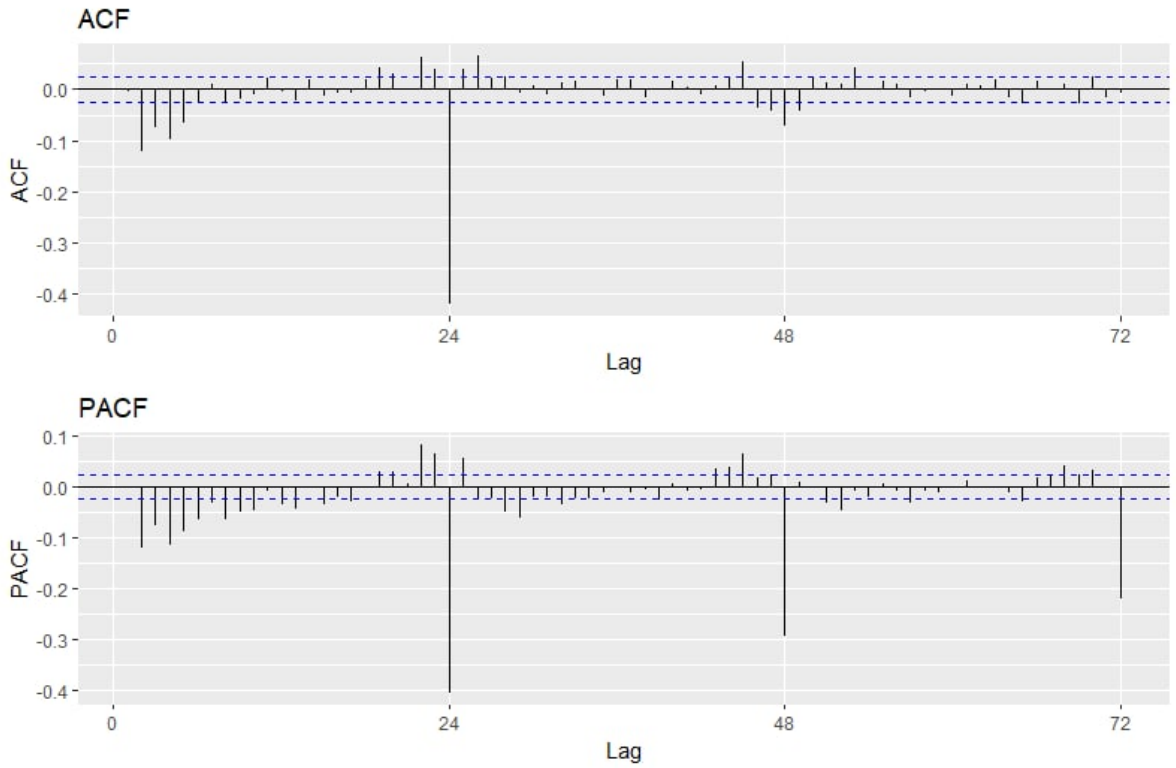

Figure 6

FIRST MODEL ARIMA

It is clear from Figure 6 that there's a moving average component from the pacf as well as different auto regressive components. After a lot of improving implementations (iterating on parameters and acf/pacf plots) with different parameters, the model with these parameters: **c(4,1,1), c(0,1,2)[24]** is the best I can estimate, with a MAPE of 0.1482. Unfortunately the Ljung-Box test is not so reliable on data that still has background noise (rejection of null hypothesis), so I can maybe consider the p-value to evaluate the implementation.
In Figure 7 a graphical representation of the model estimated on the validation subset, it seems like some patterns are caught even though not really precisely.

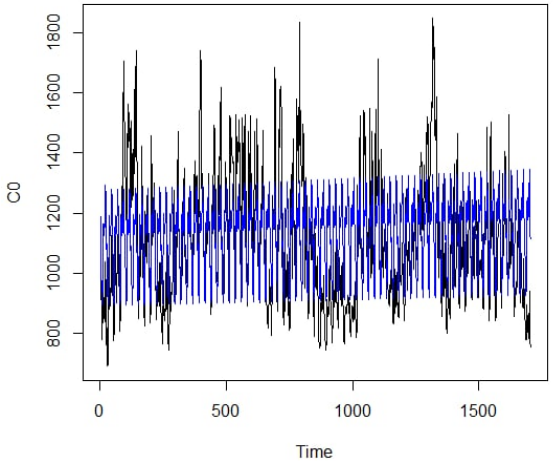

Figure 7

SECOND MODEL ARIMA

I now implement a model using sine functions as regressors. These functions should be able to interpolate certain trends that a single arima model cannot capture. Indeed, this implementation allows for multi-seasonality setted both at 24 and 168 hours (a day and a week). I chose seasonality to use 6 sinusoidal regressors (obtaining 6 sines and 6 cosines), because choosing more than that had worsened the performance when tried. The MAPE metric is 0.1422 on the validation. So the patterns are better catched as shown in Figure 8; the MAPE has improved.

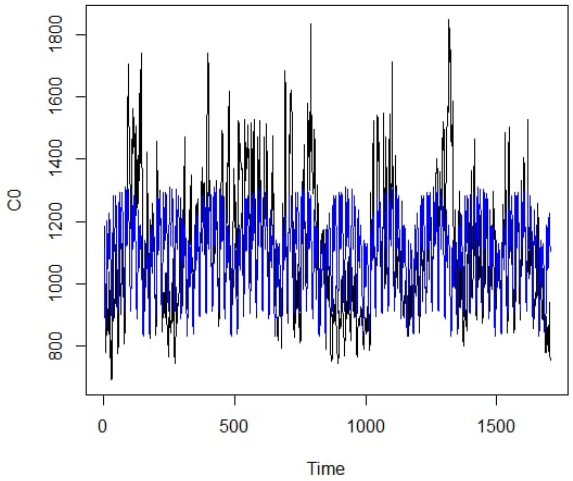

Figure 8

THIRD MODEL ARIMA

In addition to adding sinusoidal regressors, dummies were also added to include the most important holidays such as Easter, Easter Monday, Christmas, Boxing Day, the first of January and the Assumption of Holy Mary.

With this attempt to add dummies, however, a problem arose with 'optim'. In fact, R was unable to achieve a result with all seven dummies inserted, but was only able to converge after the following were removed: Christmas, Boxing Day and 1st January. This is due (after searching on stack overflow for an explanation) to the fact that the three deleted dummies were highly correlated.

The ARIMA model estimated in this way achieves very similar performances to the previous one, and the amount of information added is not so impressive due to the optim problem. The MAPE in this case is 0.1393, really not such an improvement as expected. And the graph of fitted values (Figure 9) shows how really it's quite the same as without dummies (three dummies are not a consistent number though). This evidence suggests to me that the carbon monoxide values indicated are not, at least directly, related to the typical human breaks of the holidays. This model will be the first taken into account for submission.

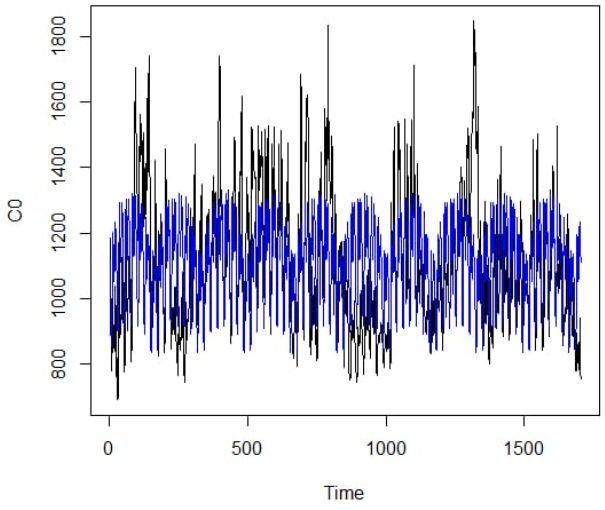

Figure 9

UNOBSERVED COMPONENT MODELS

On the basis of the considerations made for ARIMA, it was decided to estimate UCM models with these 3 different trend modellization: Local Linear Trends, Random Walks, integrated random walk. The trend is then added to other modellization approaches to handle the seasonality (with dummies and cycles) present in the series. The dummies implemented are stochastic (for daily seasonality) while the cycle is weekly setted.
I decided not to use the holiday dummies used in previous ARIMA models because they were of low explanatory power. The models generally fit a bit better than the simple Arima model, but there are no significant differences.

LOCAL LINEAR TREND

The first factor to be analysed is the LLT, which is basically an interpolating straight line, therefore I do not present the graph with the fitted values, but simply report the MAPE which is around **0.1818**.

LOCAL LINEAR TREND WITH SEASONAL DUMMIES

Using seasonal dummies also improves the fit of the model a little which leads to a MAPE of the **0.1381** in Figure 9 the fitted values on validation

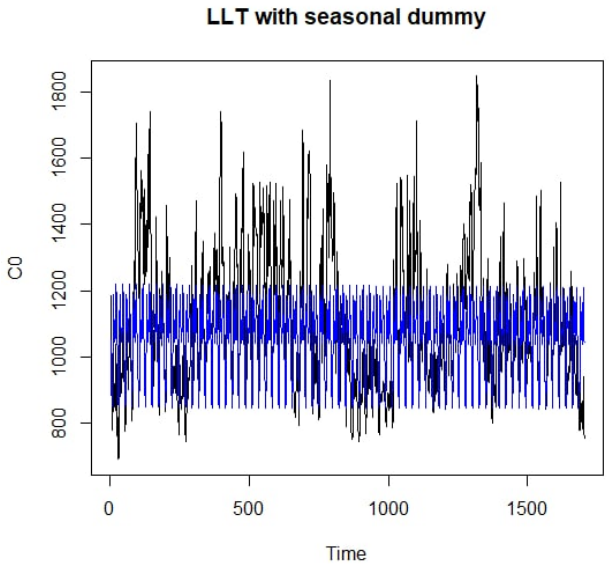

Figure 10

LOCAL LINEAR TREND WITH SEASONAL DUMMIES WITH WEEKLY CYCLE

Adding the weekly cycle to the previous model I get a MAPE of **0.1380** in Figure 10 the fitted values on validation. A slight improvement that leads to the best UCM model chosen for the submission.

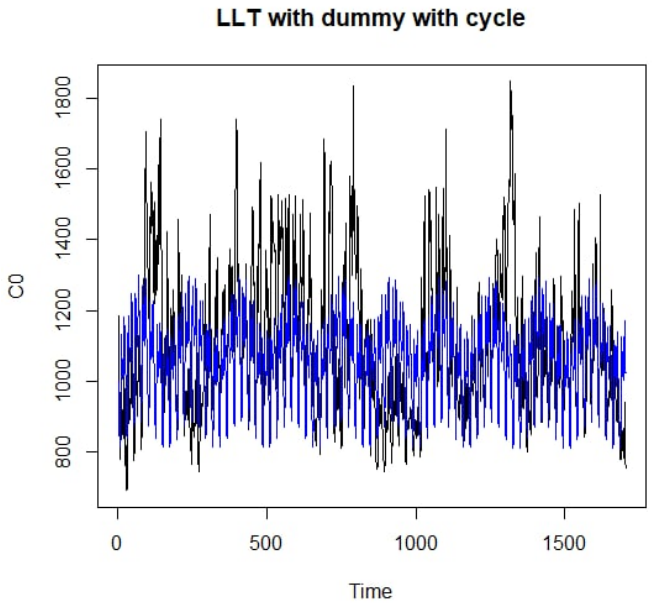

Figure 11

RANDOM WALK WITH DUMMIES AND CYCLE AND INTEGRATED RANDOM WALK

The last 2 attempts were with random walk with dummies and cycle and integrated random walk. These gained a MAPE respectively of **0.1392** and **0.1384** here in Figure 12 and 13 the plots.

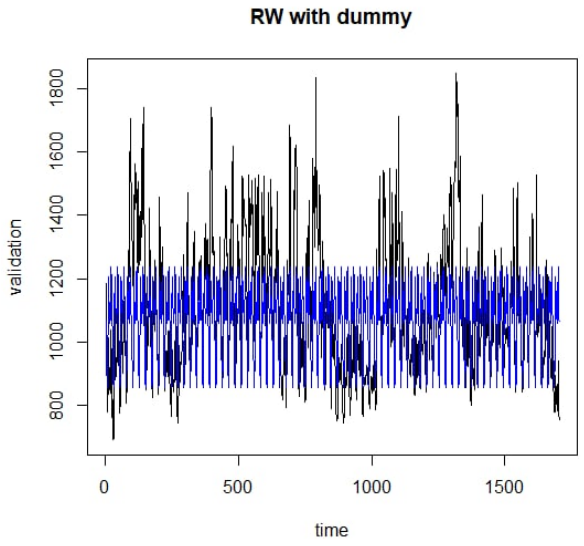

Figure 12

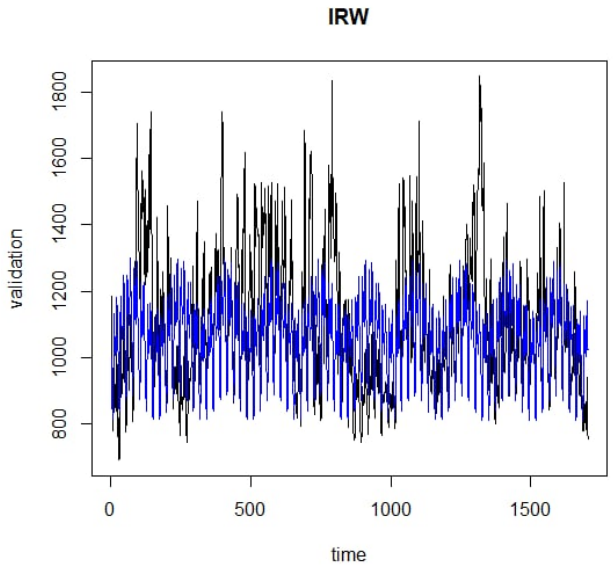

Figure 13

I noticed that the best UCM model I found is the one with a Local linear trend with seasonal dummies and the weekly cycle, the model that catches a MAPE of **0.1380**. As expected the UCM models tend to be a little better than the ARIMA models.

NON LINEAR MODELS-RNN

The basic idea behind RNNs is to make use of sequential information, time is indeed a sequential phenomenon. Another way to think about RNNs is that they have a “memory system” which captures information about what has been calculated so far, as a matter of fact the RNN training algorithm is called Backpropagation Through Time (BPTT). The most effective sequence models used in practical applications are called gated RNNs. They include LSTM (Long-Short Term Memory) and GRU (Gated Recurrent Unit) which are the models implemented.

About the general architecture implemented: the “look back” variable used for MAPE comparison will always be fixed at 14 day (336 hours, that is the information the model takes into account) and the number of neurons will always be 4, both for LSTM and GRU, to have a better comparison between results and a model that remains “simple”. The loss function chosen will be the Mean Squared Error and the optimizer will be Adam, adaptive moment estimation, that is an extension of a stochastic gradient descent. Concerning the dense output layer of the following models it will be implemented with a sigmoid activation function.

LSTM (Long-Short Term Memory)

The LSTM model is organized in cells which include several operations. LSTM has an internal state variable, which is passed from one cell to another and modified by the following Operation Gates: Forget, Input and Output gate (from which the name gated RNN). The three gates have independent weights and biases, hence the network will learn: how much of the past output to keep, how much of the current input to keep and how much of the internal state to send out to the output. In short, the model learns sequentiality as explained before.

FIRST IMPLEMENTATION (rectified linear unit)

ReLu is a faster activation function to train because its gradient is efficient to compute with respect to Sigmoid and Tanh (this activation function takes on the derivative of 1 in certain intervals and 0 in others). However, the typical vanishing gradient problem occurs less often because most of these units usually operate within the intervals where the gradient is. Although this linear variant has become far more popular than their smooth counterparts, they introduce the new problem of dead neurons that will be addressed in the next implementation. With this first implementation I obtained a test score MAPE of **5.45 %.** That is a really nice result. Here in Figure 14 is the resulting Plot concerning all the data I have.

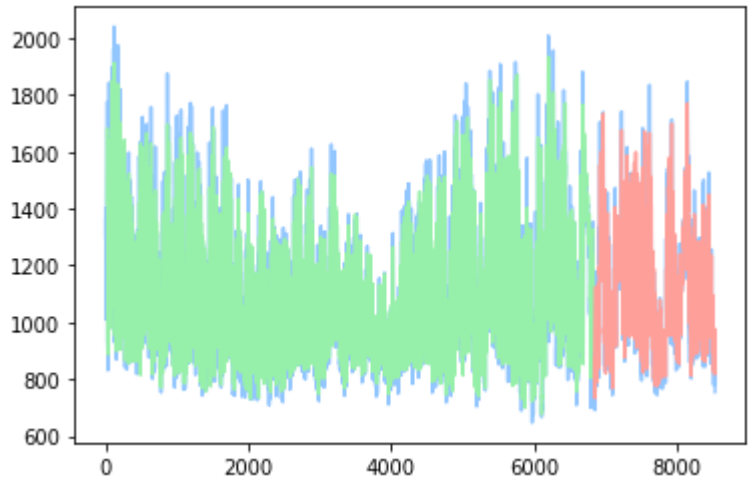

Figure 14

SECOND IMPLEMENTATION (Dropout layer implementation)

Dropout is a regularization method where input and recurrent connections to LSTM units are probabilistically excluded from activation and weight updates while training a network. This has the effect of reducing overfitting and improving model performance. Test score: **10.94 %** (MAPE) shows that probably given the paucity of data, the dropout does not improve performance: on the contrary, it reduces it. But it will be further investigated later.

THIRD IMPLEMENTATION AND DROPOUT (Leaky ReLu Implementation)

In literature it is explained that with ReLu, since neuron's output will not vary across different inputs, that will not play a role in discriminating between different instances. Such a neuron can be considered dead ("brain damage"). A partial fix to the problem of dying neurons is using learning rates that are reduced using the leaky ReLU. The test gives a 11.32 % MAPE score. Eliminating the dropout the test score is **9.93% MAPE**, confirming my previous intuition of inefficient dropout.

GRU (Gated Recurrent Unit)

In this model, the basic idea of using a gating mechanism to learn long-term dependencies is the same as in a LSTM, but there are a few key differences: A GRU has two gates (instead of three): a Reset Gate and an Update Gate. GRUs don't possess an internal memory (like LSTM) thus, the responsibility of the reset gate in a LSTM is really split up into both reset and update gates.

With respect to LSTM, Gated Recurrent Units is a slightly more simple variant that provides comparable performance and considerably faster computation (less memory too). Like LSTMs, they also capture long-term dependencies, but they do so by using reset and update gates without any cell state: while the update gate determines how much of the past information needs to be kept, the reset gate decides how much of the past information to forget. Doing fewer tensor operations as a matter of fact.

FIRST IMPLEMENTATION (rectified linear unit)

ReLu implementation gives a performance of test score: **5.38 %** MAPE, that is till now the best MAPE on validation. The plot results are pretty much the same as the first implementation of LSTM .

SECOND IMPLEMENTATION (dropout layer implementation)

As before , the test score with dropout is: **11.74%** MAPE

THIRD IMPLEMENTATION (Leaky ReLu implementation)

Leaky ReLu implementation worsened with respect to the first GRU prevision but outperformed the second, the test score: 10.97 % MAPE. The elimination of Dropout gives an improvement and the test score is **9.68 %** MAPE.

IMPLEMENTATION ISSUE WITH STOCHASTICITY

The neural network optimisation method always contains stochastic components, starting from the initialization of the weights. As far as the model architecture is concerned, optimizers also give stochasticity to the model (adam) as well as the dropout. This can definitely become inconvenient when comparing the performance of some models as well as when trying to derive predictions from this. In a matter of fact, the prediction that seems particularly good may be the result of the randomness that is present in the model. To reduce this problem I decided to set a seed for the initialization but this only solves one of the random components. Moreover the backend of colab's GPUs are also susceptible to randomness in computation. Therefore one cannot be sure of the reproducibility of all experiments with the exact same result. But on average the results of MAPE on the ML model should be consistent and should not vary more than some percentage point with respect to the results obtained; and GRU remains indeed the best model to be chosen for the submission. Afterwards, the forecast selected will in fact be chosen with a criterion that takes into account consistency with the series preceding the forecast

BEST MODEL

Selecting the best ARIMA and UCM forecasts was an immediate process, while selecting the GRU model forecasts (the best of the ML models) required an iterative process due to the randomness as explained before.

It also appears that ML models perform generally better than Arima and UCM models, which may be due to the fact that there is much granularity and poor annual general patterning that a GRU or LSTM models could understand better than an Arima model. If two years were provided or the granularity were daily and not hourly, this would probably have strengthened Arima (Nevertheless these are speculation because *there are no free meals in statistics!*).

To choose the forecast to be delivered, I confronted the values in table 2.

MODEL/ MAPE %	#1 IMP	#2 IMP	#3 IMP	#4 IMP	#5 IMP	#6 IMP
ARIMA	14.82	14.22	13.93			
UCM	18.18	13.81	13.80	13.92	13.84	
ML	5.45	10.94	9.93	5.38	11.74	9.68

Table 2

The forecasts are drawn together with the time series to see if they make sense. Observing the Figures below, it can be noted that the forecasts of the models seem to make sense as they are consistent with the trend of the historical series. In particular, the forecasts of the UCM and ARIMA models seem to capture a slight upward trend in the historical series, while the GRU forecasts seem to capture a larger upward trend.
Summing up, the following prediction plots are: prevision made with ARIMA seasonal with regressor and dummies (with weekly granularity on x-axis) (Figure 15); UCM with LLT with dummies and cycle (with daily granulary on x-axis) (taking the last 100 days) (Figure 16); GRU with ReLu (with daily granularity on x-axis) (Figure 17). ARIMA has a strong downward trend, UCM seems to be moving with an armonich trend and GRU seems to have a tendency to take on less fluctuation over time. It should be noted that using a bit more than 50 weeks for predicting 4/5 weeks is not the best of the starting scenarios. Arima indeed is really poor on such a long estimation.

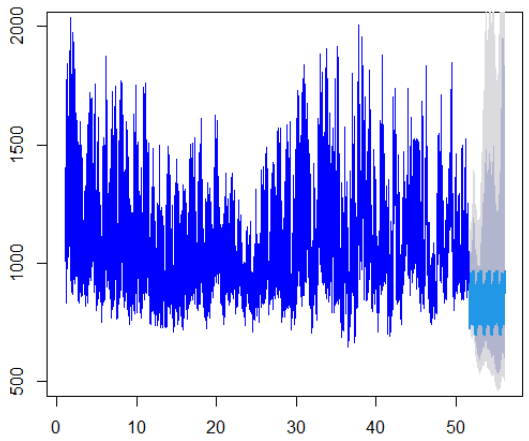

Figure 15

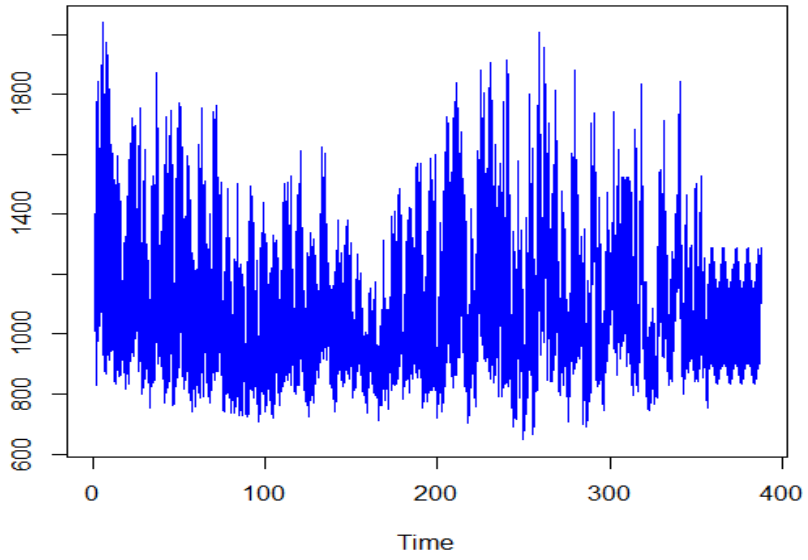

Figure 16

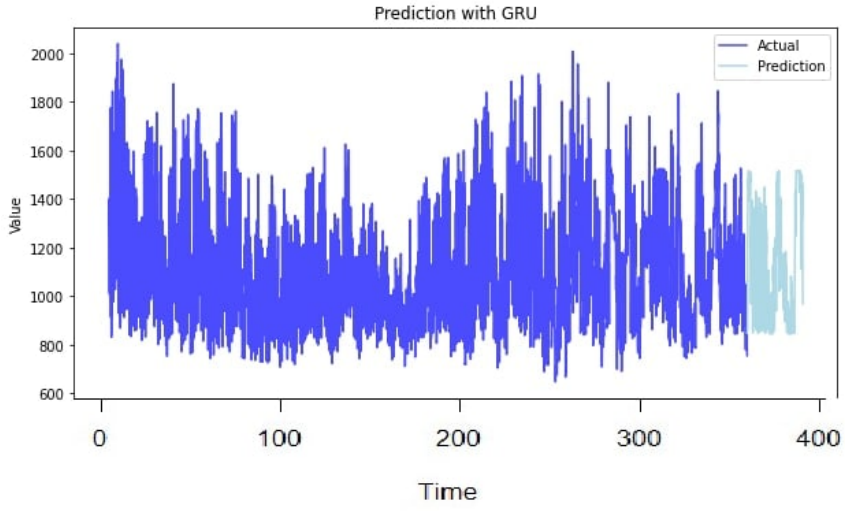

Figure 17

In the end, for the ML model in Figure 17, iteration with quite a downward trend results in a better forecast because the data closer to March also tend to be low. There is not any information on what this CO actually measures so no further assumptions can be made about it.

FURTHER IMPROVEMENTS & CONCLUSIONS

As far as the arima model is concerned, given the dynamics with which the forecasts are calculated a better modellization could be performed to improve the performance of the model. Additional dummies or regressors could have been added too. In contrast, UCM

models are more versatile and therefore there is even more potential for improvement. The cycle or seasonality or stochastic component and new trigonometric dummies could have been better investigated too.

The ML models, on the other hand, are such an evolving universe that there are so many improvements to be made. Apart from improving the models presented here (like fine tuning the learning rate), one could have used pre-set models such as AutoML to perform better (maybe). Use KNN prediction models, implement stateful LSTM, use Facebook Prophet or other state-of-the-art models are also further improvements.

REFERENCES & BIBLIOGRAPHY

Pelagatti (2015) Time Series Modelling With Unobserved Components

Rob J Hyndman and George Athanasopoulos Forecasting: Principles and Practice

LESSONS SLIDES AND NOTES (streaming data management & time series analysis 21/22 Pelagatti-Candelieri)

https://machinelearningmastery.com/reproducible-results-neural-networks-keras/

https://machinelearningmastery.com/use-dropout-lstm-networks-time-series-forecasting/