# TIME SERIES FORECASTING

Alessandro Fossati, *University Bicocca of Milan, Faculty of Data Science*

July 1, 2021

## 1 Introduction

In this report will be given details about the development and validation of some predictive models aimed to forecast 2 months of data for a given hourly time series. In this way have been trained models for each of these categories: *ARIMA*, *UCM* and *Machine Learning*. After a first explorative phase of the time series, all the models used in order to obtain predictions will be presented in details, and they will be finally compared in order to find a winner in this particular race. This project have been developed exploiting Python tools, particulary using *statsmodels* and *keras* for modeling, and *matplotlib* for plots.

## 2 Data Preprocessing and Exploration

In this first section will be shown the results of an *exploratory analysis* of time series and *preprocessing* operations on the starting dataset.
First of all is good to make the reader available of time series temporal horizon. We have values of an hourly time series starting from 2018-09-01 to 2020-08-31, and we want to forecast values starting from 2020-09-01 to 2020-10-31 (2 months).
As first preprocessing operation we want to set a DateTime value as index. Afterwards we find that there are missing values in starting time series, that we have to fill. For days in which time switches from solar to legal (2019-03-31 and 2020-03-29) values for 02:00 AM are missing, and we choose to fill them with the value of the hour before (01:00 AM). Moreover, values for the day of 2020-05-31 are missing (all 24 hourly values) and so they have been

filled with the average value of the same hour of the three May's weekday before (10th May, 17th May, 24th May), given that we can observe a weekly seasonality in original time series as will be shown later. In *Fig2.1* we can see the head of the dataset preprocessed and some statistics aimed to understand the distribution of time series' values:

| DateTime | Value |
|---|---|
| 2018-09-01 00:00:00 | 3646742.0 |
| 2018-09-01 01:00:00 | 3273110.0 |
| 2018-09-01 02:00:00 | 3069245.0 |
| 2018-09-01 03:00:00 | 2969621.0 |
| 2018-09-01 04:00:00 | 2944116.0 |

| | Value |
|---|---|
| count | 17544.000 |
| mean | 4233665.195 |
| std | 1258631.054 |
| min | 1796490.000 |
| 25% | 3155179.500 |
| 50% | 4182403.500 |
| 75% | 5146105.500 |
| max | 9099482.000 |

Fig2.1

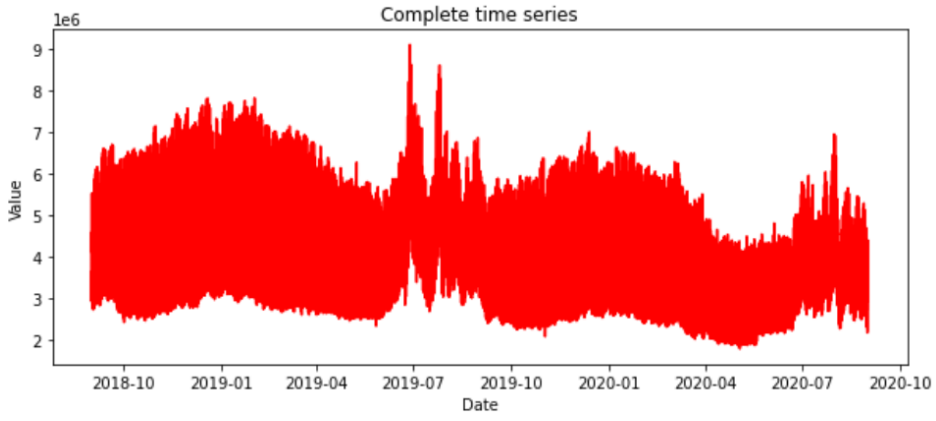


Fig2.2

As is visible in *Fig2.2* there are 20 outliers (less than a day of data) during 2019 summer period, so they have been brought back to the BoxPlot's wishkers for completeness. They won't have changed significantly results, but is always better to treat outliers before applying modeling strategies.
We can now analyze some plots related to the original time series in order to understand its shape and its seasonalities. In *Fig2.2* we can have a look on the shape of the starting time series, and we can notice a sort of descending trend going on, with some peak during summer periods. We can already notice seasonality in our series, but before analyzing it, we want to show trend during summer periods (*Fig2.3*). We can notice similar trends (excluding July 19) with different mean value at a first look:


Fig2.3

In *Fig2.4*, in the following page, are shown trends of value, on different years, for month and for week.

Analysis is not totally reliable because of the lack of data for 2018 and 2020, but similar trends between years are not questionable, especially for Month-Year plot. So, we can say that we are in presence of an *annual seasonality* for our time series. We can finally confirm that there's a *descendent*

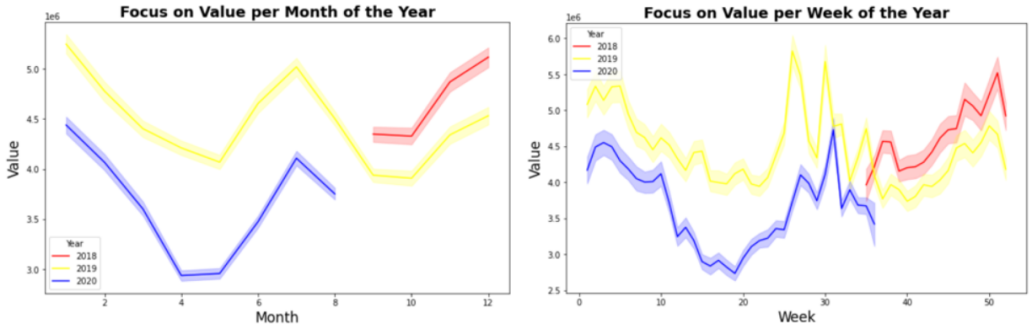


Fig2.4

*trend* year by year.
In *Fig2.5* we want to go deeper in time series seasonal analysis, showing that during workweek time series' average values grow up, going down in the weekend in the left side plot. In the right side plot we can observe a daily trend, and we can confirm the fact that there is not a statistic significative difference between values during workweek's days, differently from weekend's days. MatplotLib provides a confidence interval for the average value:

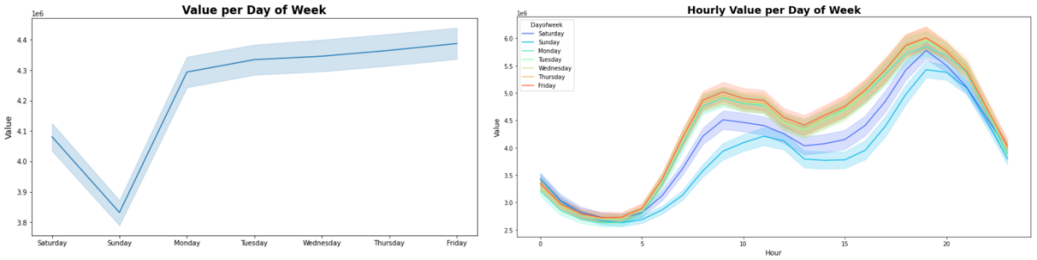


Fig2.5

Given that time series data are also relative to 2020, we have to consider *Covid-19 variable* in our analysis, but this variable is such impactful? Analyzing if series' values change significantly starting from Covid-19 discovery, have been found that mean percentage time series' value decreases passing from the period 2018-09-01 – 2018-12-31 to the period 2019-09-01 – 2019-12-31 and passing from the period 2019-01-01 2019-08-31 to the period 2020-01-01 2020-08-31. We obtain a degrowth of *10.41 %* for the first period and a degrowth of *20.43%* for the second period. As we have seen before

(*Fig2.4* for example), time series has a generalized decreasing trend passing from 2018 periods to 2020 periods. Hence, also due to the fact that we don't know to which sector this time series belong (could belong to a sector not so influenced or weakened by Covid-19) has been decided not to include a Covid-19 variable in modeling strategies.
We conclude this first section of the report explaining the choice of the proportion of *Training* and *Test* (Validation) Set. In order to validate forecasting models that will be shown in following sections, has been decided to have a portion of Test Set about equal to 9% of the time series given. This portion corresponds to 2 months of data, equivalent to the quantity of values that models have to forecast. The remaining 91% of the given time series will be our Training Set, finalized to train models. This division on Train and Test Set will be the same for all the models implemented.

# 3 Time Series Modeling

In this second section of the report will be presented all the modeling strategies developed in order to forecast time series' new values. In this way will be shown details and results concerning ARIMA, UCM and Machine Learning models, obtained exploiting statsmodels and keras Python tools as anticipated.

## 3.1 Training ARIMA Models

In order to make an accurate analysis using ARIMA models, time series used has to be *stationary*. In this way have been computed a couple of tests for stationarity as *Dickey Fuller* (D-F) test and *KPSS* test, and *correlograms* have been analyzed in order to understand if time series has to be differentiated in order to obtain stationarity. In *Fig3.1.1* (following page) we can see the results of D-F and KPSS test for not differentiated time series (left side) and for differentiated time series (right side).

Analyzing left side table's pvalue we can see that tests do not agree, in fact D-F test tells us time series is definitely stationary, but KPSS tells us that it is not stationary. In this case, applying a *seasonal difference* we obtain right table's results, which tell us that applying seasonal difference of order 24 (daily), stationarity is got. We can understand which order of seasonal difference to apply from correlograms (*Fig3.1.2*). In Fig3.1.2 we can see how

| Test | Statistic | p-value |
|---|---|---|
| D-F | -5.304 | 0.000 |
| KPSS | 14.503 | 0.010 |

| Test | Statistic | p-value |
|---|---|---|
| D-F | -24.347 | 0.000 |
| KPSS | 0.010 | 0.100 |

Fig3.1.1

seasonal difference improves the quality of the correlograms, helping us to understand which ARIMA model to select as we will see later.

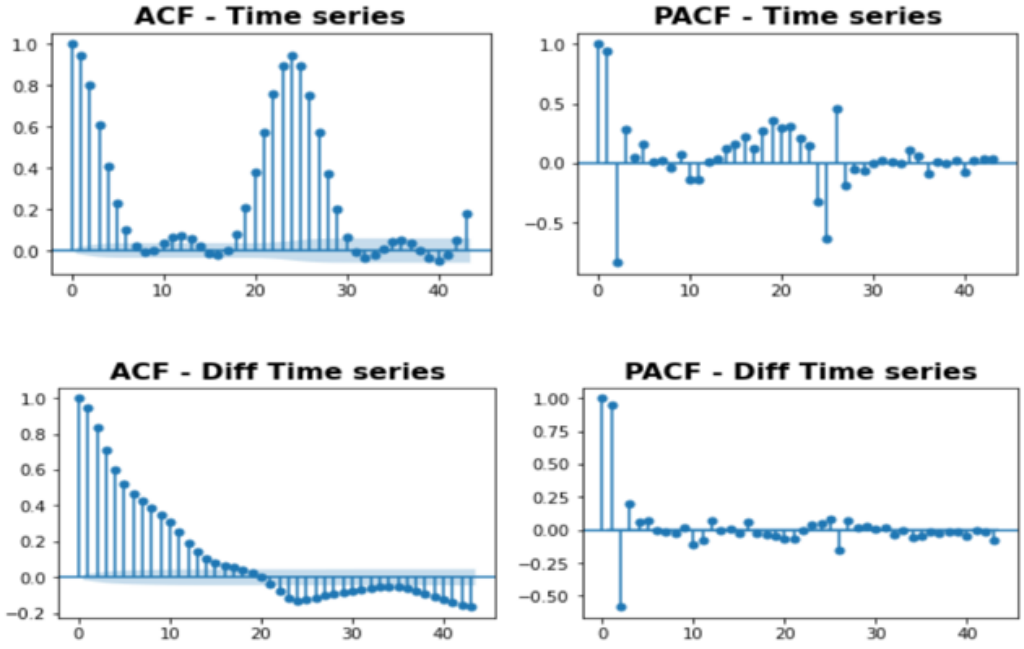


Fig3.1.2

In *Fig3.1.3* we can observe the improvement in terms of mean and variance stationarity (no need for log-transformation) of the time series.

Studying correlograms we can narrow it down selecting one of the ARIMA models presented in *Fig3.1.4* according to the values of *AIC*, *Mean Absolute Error on Training Set* and *Mean Absolute Error on TestSet* (MAE = $\frac{1}{n}\sum_{t=1}^{n}|e_t|$), exploiting a *Grid Search* method to select the best SARIMA

model:

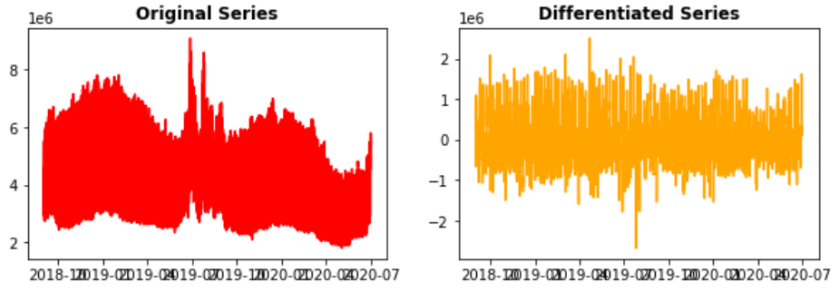


Fig3.1.3

| | Model | AIC | Train_MAE | Test_MAE |
|---|---|---|---|---|
| **0** | SARIMA(1,0,2)(1,1,1)24 | 415145.0 | 67459.3 | 465020.5 |
| **1** | SARIMA(2,0,1)(1,1,1)24 | 414935.0 | 66562.3 | 451173.0 |
| **2** | SARIMA(2,0,2)(1,1,1)24 | 414906.0 | 66453.4 | 452477.8 |
| **3** | SARIMA(3,0,1)(1,1,1)24 | 414604.0 | 65686.3 | 450422.6 |

Fig3.1.4

We can understand that best is SARIMA(3,0,1)(1,1,1)24, because of its lowest value of AIC and given its best values for Train and Test MAE (lowest as well). In *Fig3.1.5* we can see graphically performances of this model on Train and Test:

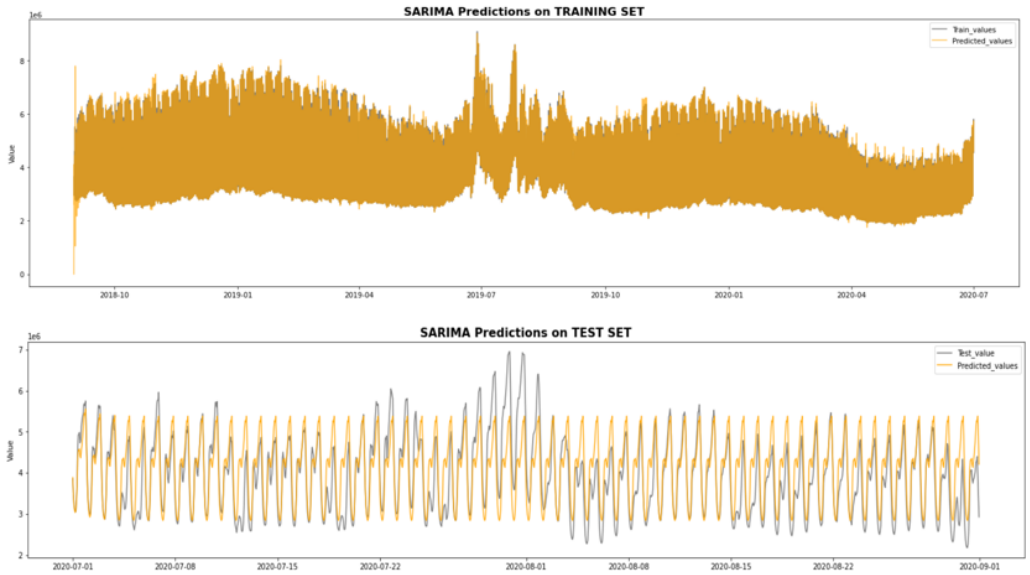


Fig3.1.5

While fitting on Train is good, it is not a great prediction on the Test Set, so we can improve the models with some external regressors. Simple Seasonal ARIMA does not catch all the seasonalities. In this way we can exploit *harmonics*. So we need Fourier series in order to catch weekly and annual seasonalities, and so we have to select the correct number of harmonics for the model. Time series is hourly and can be considered quite smooth, so we don't need too many harmonics. We put the limit to 6 harmonics for the week and to 8 harmonics for the year. Even in this case has been used a Grid Search approach finalized to minimize AIC value and MAE on Train and Test. Results have been reported in *Fig3.1.6*, on the following page. We can see that best pair of weekly-yearly harmonics number is 6-8, obtaining best performance on Test Set compared with the other pairs:

| Week_Harmonics | Year_Harmonics | AIC | Train_MAE | Test_MAE |
|---|---|---|---|---|
| 2 | 2 | 414288.0 | 65540.1 | 453625.5 |
| 2 | 4 | 414343.0 | 65657.7 | 414419.9 |
| 2 | 6 | 414357.0 | 65679.8 | 397688.0 |
| 2 | 8 | 414357.0 | 65668.5 | 393040.1 |
| 4 | 2 | 414138.0 | 65699.7 | 475626.3 |
| 4 | 4 | 414193.0 | 65825.8 | 431829.7 |
| 4 | 6 | 414209.0 | 65857.5 | 397133.4 |
| 4 | 8 | 414207.0 | 65842.2 | 391336.9 |
| 6 | 2 | 413753.0 | 65107.0 | 461477.5 |
| 6 | 4 | 413806.0 | 65238.9 | 417495.1 |
| 6 | 6 | 413821.0 | 65258.0 | 386858.9 |
| 6 | 8 | 413820.0 | 65240.7 | 379506.4 |

Fig3.1.6

In *Fig3.1.7* we can see best model's performance. We can observe a good fit on the Training set as expected, and a better fit on Test Set with respect to simple SARIMA, as anticipated by MAE values. We can observe a difficulty in catching July-August's peak, but trend and values in general are not bad on Test Set for SARIMAX.
Will be finally shown residuals diagnostic's results for the best ARIMA model (*Fig3.1.8*). Standardized residuals and correlogram seems to be quite good, but we don't obtain good results in *Normal QQ-plot* and in *Histogram of residuals*. We cannot assume Gaussian distribution of residuals, and this could be a problem if analysis would have been deeper, or if they would have been carried on in real life.
We can conclude this section saying that in forcasting phase will be used SARIMAX(3,0,1)(1,1,1)24 with 6 harmonics for the week and 8 harmonics for the year.

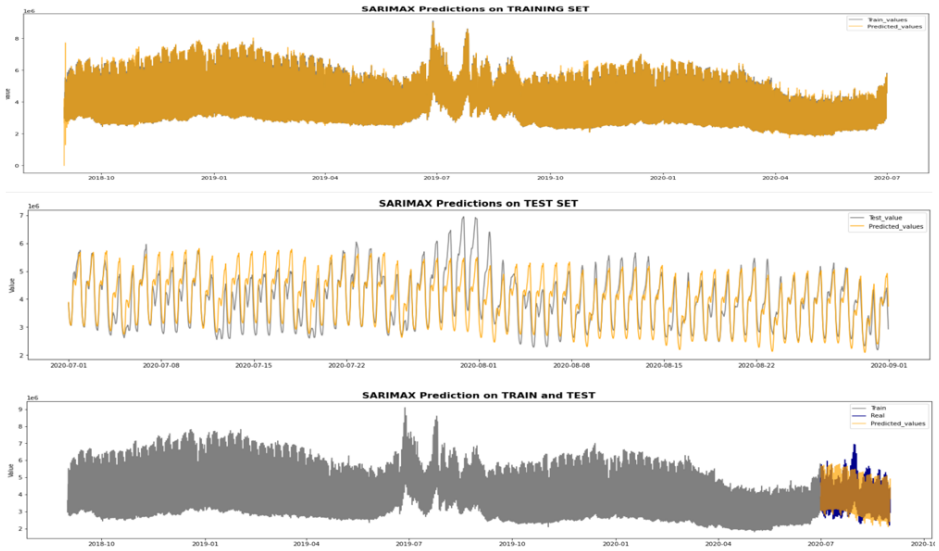


Fig3.1.7

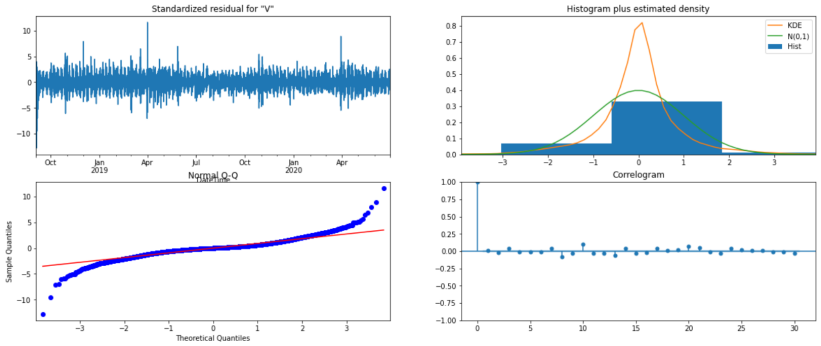


Fig3.1.8

## 3.2 Training UCM Models

In this subsection will be shown results and performances coming from the training of the best UCM model (Unobserved Component Models) selected. Has been decided to include *seasonal* and *trend* components. For seasonal component we confirm number of weekly (6) and yearly (8) harmonics selected for best ARIMA model adopting *trigonometric seasonalities*, while in order to select best kind of trend component has been applied a Grid Search method, which results are reported in *Fig3.2.1*:

| Trend | AIC | Train_MAE | Test_MAE |
|---|---|---|---|
| rwalk | 525651.0 | 134025.6 | 379506.4 |
| rwdrift | 525619.0 | 134025.5 | 379506.4 |
| lltrend | 528466.0 | 141779.1 | 379506.4 |
| lldtrend | 526925.0 | 137929.3 | 379506.4 |

Fig3.2.1

We search for the best trend between *random walk, random walk with drift, local linear trend and local linear deterministic trend*. Fig3.2.1 tells us that the best trend we can select is *random walk with drift* because the model with this kind of trend presents lowest AIC value and lowest MAE on train, for the same MAE on Test. Random walk with drift can be expressed in this way:

$$y_t = \mu_t \tag{1}$$

$$\mu_t = \mu_{t-1} + \beta + \eta_t \tag{2}$$

We can now analyze plots about performance reached from the best selected UCM model, focusing on performance of fitting of Training and Test Set (*Fig3.2.2*).

As for SARIMAX we can observe a good fit on Training Set, but in this case can be observed a generalized overestimation of the values on Test Set, particularly for the first half of the Test Set. This UCM model seems to be

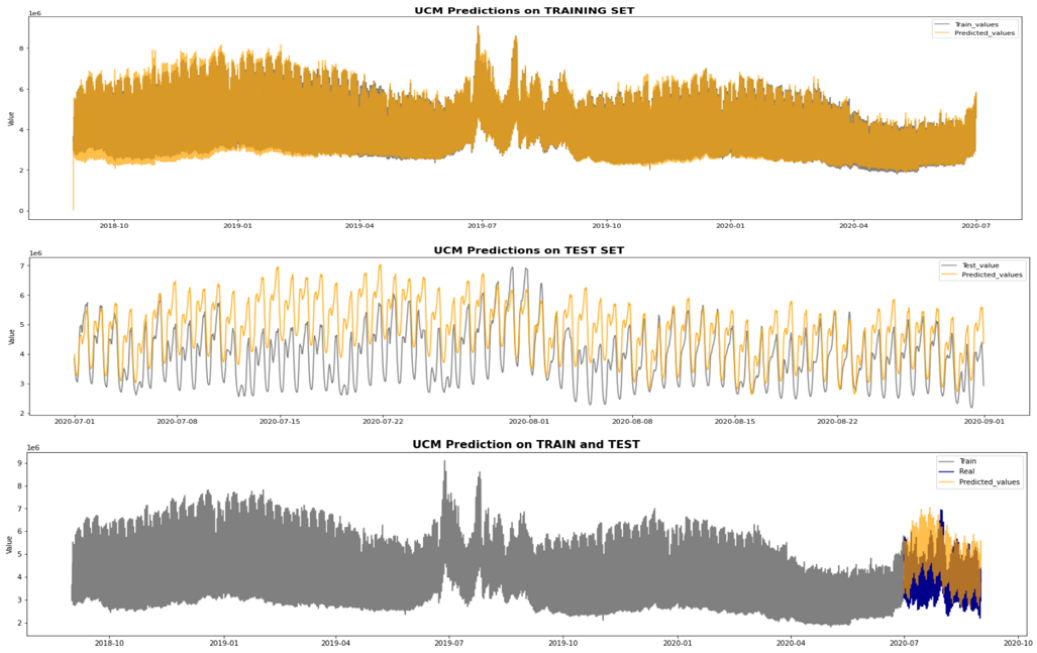


Fig3.2.2

better than best ARIMA in catching July-August's peak (maybe because of the generalized overestimation), but not good at all. Finally, we can observe that on the second half of Test Set, model seems to perform in a better way than the first half.

## 3.3 Training Machine Learning Models

As last group of models will be trained Machine Learning (ML) models, particularly have been used recurrent neural network as *LSTM* (Long-Short Term Memory) and *GRU* (Gated Recurrent Units). At this point the approach used becomes totally data driven, and we aim to select the best recurrent neural network for the problem we are trying to solve. The strategy is to tune network's parameters for both LSTM and GRU and finally compare final results in order to find the best model to get direct output vector predictions (one shot forecast). For both kind of network is first necessary to normalize data between a range of values (0,1) and to reshape the dataset.

Following parameters and characteristics are chosen for both neural networks after some attemps:

- *Stateless* networks, without memory between batches
- Training *lookback window* of a month (24x30=720 observations)
- *Single layer* of *300 neurons* for input
- *Dense* output layer , activation function *tanh*
- *Dropout Rate* equal to 0.2
- *EarlyStopping* method to avoid overfitting monitoring model's loss
- Loss function selected is *MAE* with optimizer *adam*

Fixed this parameters we aim to search for the correct number of *epochs* and for the correct *size of batch* for both networks.
Starting from epochs, we narrow the search in order to limit overfitting and computational time expenditure. We search for the correct number of epochs between 5, 10, 15, 20, 25, 30 epochs repeating the operation for 5 times and taking the average results of MAE obtained on Training and on Test Set. Once obtained best number of epochs for both RNNs, we apply the same strategy to get the best size of batch, searching between values 5, 10, 15, 20, 25, 35, 50 to obtain average values of Training and Test MAE on 5 iterations. Results for LSTM are reported in *Fig3.3.1*, while results for GRU are reported in *Fig3.3.2* in the following page.
In order to understand which are the best values for parameters tuned, is necessary to search for the lowest MAE on Test, which corresponds to 5 epochs for LSTM, subsequently, retraining the network with best number of epochs is possible to get best size of batch, which is 25 for LSTM.
For GRU we have that best selection corresponds to 15 epochs and batch size equals to 15 as well. Finally, we have to choose the best model that will be used in forecasting phase. *Fig3.3.3* shows final comparison between LSTM and GRU:

| Epochs_LSTM | Train_MAE_mean | Test_MAE_mean |
|---|---|---|
| 5 | 434272.72 | 324341.80 |
| 10 | 536036.58 | 427411.06 |
| 15 | 504690.42 | 386830.10 |
| 20 | 545160.38 | 439957.34 |
| 25 | 465673.08 | 369065.80 |
| 35 | 538625.54 | 428948.30 |

| Batch_size_LSTM | Train_MAE_mean | Test_MAE_mean |
|---|---|---|
| 5 | 486863.98 | 353616.20 |
| 10 | 493425.04 | 365133.40 |
| 15 | 523541.60 | 400761.30 |
| 20 | 480407.18 | 368558.02 |
| 25 | 419381.28 | 323647.68 |
| 35 | 527053.26 | 421167.04 |
| 50 | 468375.16 | 372926.80 |

Fig3.3.1

| Epochs_gru | Train_MAE_mean | Test_MAE_mean |
|---|---|---|
| 5 | 503064.12 | 389474.84 |
| 10 | 463881.22 | 355604.32 |
| 15 | 369072.20 | 258016.40 |
| 20 | 426500.58 | 325433.56 |
| 25 | 514857.84 | 414335.56 |
| 35 | 441229.74 | 345118.38 |

| Batch_size_gru | Train_MAE_mean | Test_MAE_mean |
|---|---|---|
| 5 | 453259.70 | 348430.64 |
| 10 | 500276.16 | 394071.06 |
| 15 | 427206.98 | 329279.52 |
| 20 | 517455.44 | 419225.92 |
| 25 | 484120.10 | 376954.02 |
| 35 | 481551.28 | 378244.66 |
| 50 | 468343.00 | 368505.02 |

Fig3.3.2

| RNN | Epochs | Neurons | Batch_size | Train_MAE_mean | Test_MAE_mean |
|---|---|---|---|---|---|
| LSTM | 5 | 300 | 25 | 472001.04 | 368665.78 |
| GRU | 15 | 300 | 15 | 469644.22 | 355799.66 |

Fig3.3.3

The winner is GRU, so *Fig3.3.4* shows graphically its performance on Train and Test Set:

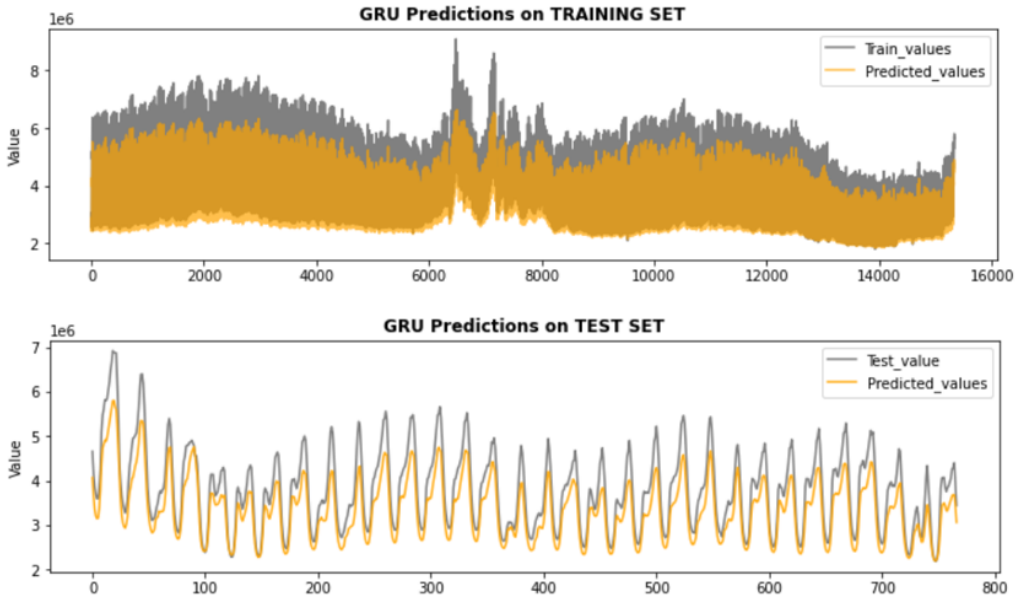


Fig3.3.4

Can be observed a generalized underestimation both on Training and Test Set, even if trend is definitely caught. Is good to remember that performance of ML models could be improved growing up number of layers, neurons, epochs and batch size, but due to limit of computational resources, results have been accepted. In a real life application this model could be improved.

# 4 Final Comparisons and Conclusions

In this final section will be compared performance on Train and Test Set of the models used and will be presented plots about their forecast. In real life we would have to choose the best model, the most reliable model in order to obtain accurate predictions. In this way, we could exploit MAE measure to have a clearer idea on which could be the best model (*Fig4.1*):

| Model | Train_MAE | Test_MAE |
|---|---|---|
| ARIMA | 65240.7 | 379506.4 |
| UCM | 134025.6 | 379506.4 |
| ML | 469644.2 | 355799.7 |

Fig4.1

We can notice that the best performer on Test Set is ML model, which however is definitely the worst performer on Training Set. ARIMA and UCM best models are a little bit worse on Test MAE value than ML model, but they are definitely better than ML on Training Set performance. ARIMA so could be selected as the best model. In fact, ARIMA model is the best in Train fitting and is the best in catching seasonalities of original time series as we can see in previous section's plots.

In *Fig4.2* in the following page we can graphically analyze results for the 3 best models selected, and we can observe the low reliability of ML model compared to the reliability of linear models. Their forecasts are more similar to original time series' September-October's trend and values than ML model. ML gives us very regular predictions, in the sense that if you consider a period of few days, you get very low standard deviation for time series' values differently from ARIMA and UCM. However looking at original time series we can see how it values for a brief period are definitely not similar, in this way linear models can better represent the series and its forecast.
We can finally observe that for September (first half of forecast) month predictions' values and trend are quite similar between the models (ML model carries around underestimation problem as said previously), heavily diverging for October (second half of forecast) month instead:

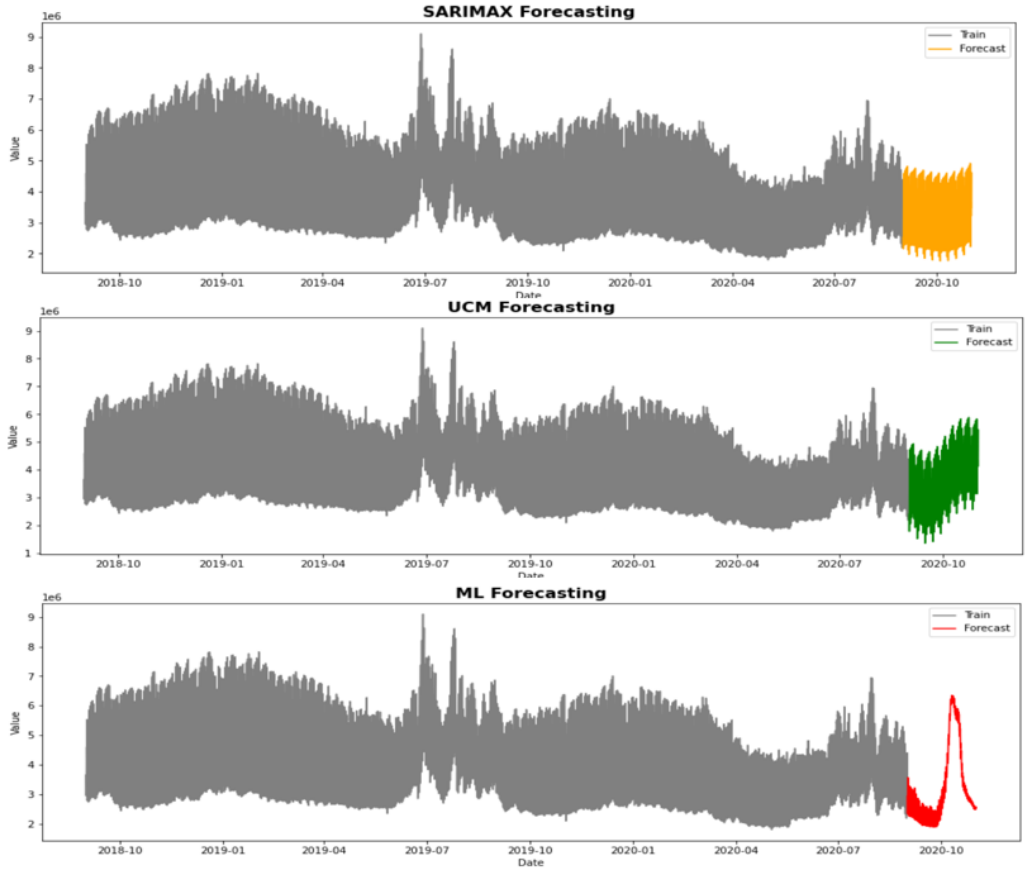
SARIMAX Forecasting
UCM Forecasting
ML Forecasting
Train
Forecast
Value
Date
1e6
2018-10
2019-01
2019-04
2019-07
2019-10
2020-01
2020-04
2020-07
2020-10

Fig4.2

# 5 Sitography


- https://machinelearningmastery.com/tune-lstm-hyperparameters-keras-time-series-forecasting/
- https://machinelearningmastery.com/multi-step-time-series-forecasting-long-short-term-memory-networks-python/
- https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/
- https://otexts.com/fpp2/
- https://www-taylorfrancis-com.proxy.unimib.it/books/mono/10.1201/b18766/time-series-modelling-unobserved-components-matteo-pelagatti
- https://elearning.unimib.it/course/view.php?id=31237