Case Study: Fiat Chrysler Automobiles Time Series Prediction in Welding Process Control

Team 36: Sanja Stanišić, n. 800409[1]

Abstract

Industry 4.0 enabled huge amount of data to be recorded and used in improving the entire industrial value chain. This project was aimed at forecasting the voltage level of welding guns at a FCA production plant using time series analysis. The analysis was performed using both traditional and Machine Learning models. The forecasted value of voltage could be used for anomaly detection, i.e. for real-time monitoring of welding enabling higher efficiency and better quality of product.

Keywords

welding, time series analysis, voltage, forecasting

[1] Università degli Studi di Milano Bicocca, CdLM Data Science

Contents

Introduction

Industry 4.0 represents a new phase in the organization and control of the industrial value chain, due to the integration of advanced technologies such as the Internet of Things (IoT), artificial intelligence (AI), cloud computing and big data analytics into the manufacturing process, as well as throughout the entire value chain. Products and means of production get networked and 'communicate', enabling new ways of production, value creation, and real-time optimization.

In automotive industry increased automation and robotics in the manufacturing process, have already resulted in increased efficiency and precision. In addition, the use of IoT and big data analytics allow real-time monitoring and optimization of production.

This case study comes from a production line of a Fiat Chrysler Automobiles (FCA) production plant, more precisely from a welding station n. 005 of this plant, where shells of cars come to be assembled. The heart of the system is the Open Gate station, a high-density cell where robots surround each car body and weld simultaneously [1]. It is a state-of-the-art car body welding line, as this robotic welding system is equipped with sensors and other devices that collect data relating to welding processes. These data are collected by Weld Quality System (WHS), whilst Weld Management System (WMS) generates welding curves of a specific process. The data are subsequently collected and stored in a file system enabling their analysis aimed at identifying patterns and trends that can be used to optimize the process, reduce waste, and improve product quality.

This paper aims at predicting the value of the voltage in different moments of time using time series. Namely, voltage values bellow/above the acceptable range suggest a possible anomaly. Being able to predict the voltage therefore is of high importance in anomaly detection, which contribute significantly to real-time control of the welding process.

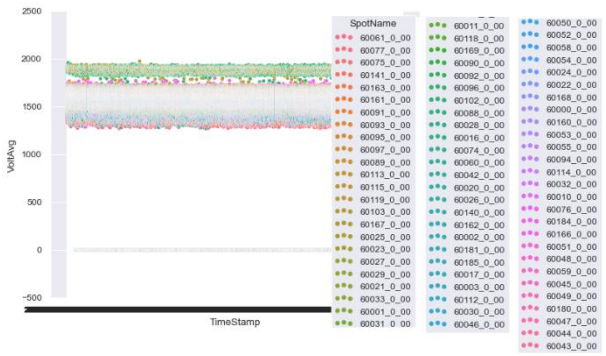
Data Exploration and Preprocessing

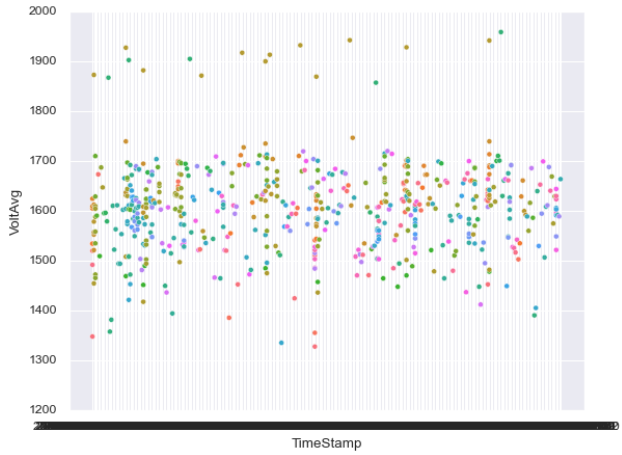
The data provided are *.json* files extracted from the above-mentioned WMS over 2019 and 2020, more precisely from September, November and December of 2019, and

January, February, March and May of 2020, totaling 222866 *.json* files.
Every extracted *.json* file contains 4 features: Time Stamp (date and time at the beginning of welding), Spot Name (welding spot), Current Curve (list of current values in A) and Voltage Curve (list of voltage values in V) as shown in the table below.

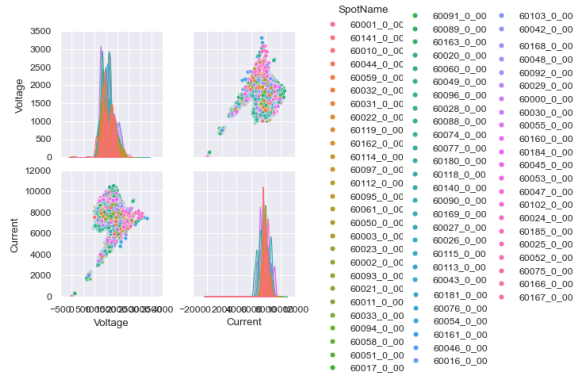
NAME	TYPE	DESCRIPTION
Time Stamp	string	"2019-09-17 17:33:48"
Spot Name	string	"60061_0_00"
Current curve	array (int)	[312, 2067, 3862, 5267, 6398, 7217, 7959, 8700...]
Voltage Curve	array (int)	[134, 756, 1292, 1623, 1812, 1876, 1946, 2010,...]

At the station n.005 WMS generates data every millisecond after the initial welding time at 76 welding spots.
Upon uploading the original files, they were collated in one data frame with the same structure as single files, and then saved as *.csv*. file. This was the **first step** of data preprocessing, which was, for me, the most challenging and time-consuming part of the project.
Due to a large amount of data, a special data set was prepared for the develop mode – i.e. for the trials, prior to running the code on the entire dataset. The appropriate dataset was made after every preprocessing step, and an underscore was added at the end of the file name in order to differentiate it from the complete dataset (e.g. the complete dataset after the first step of the preprocessing was named *data_structured.csv*, while the dataset for develop mode was name *data_structured_.csv*). This dataset has been furtherly analyzed: the first data records have been registered on September 17, 2019, while the last ones have been registered on May 28. However, over the entire period of eight months only the data registered in 60 days were provided. It was checked whether the data have been registered on Sundays and, as well, at what time the registration of data had begun and ended. Just for the sake of the brief preliminary analysis of this dataset, days with less than 500 data records were excluded since they didn't hold enough information (13 days were excluded, representing less than 10% of data). Subsequently correlation between the welding spots and the date was checked, and it was insignificant (0,000055).
Aiming at smoother the preprocessing, as well as training and testing, several custom functions were created and saved as python files: *columns.py, dataframes.py, infer.py, utils.py, utils_display.py, utils_train_test.py.*
The **second step** of data preprocessing was aimed at exploratory analysis of data and therefore a specific data frame for this analysis was made with summarized values of voltage and current curves, i.e. for every pair Spot Name – Time Stamp, the voltage curve and current curve have been summarized and for each one a minimum, maximum and average value as well as count of list's elements was calculated. This data frame was saved as *data_structured_sum.csv*, and *data_structured_sum_.csv*
The calculations have shown that there is no significant correlation between the voltage and the date, nor between the voltage and the spot name. Two following figures show the distribution of average voltage per Time Stamp, at different Spots.



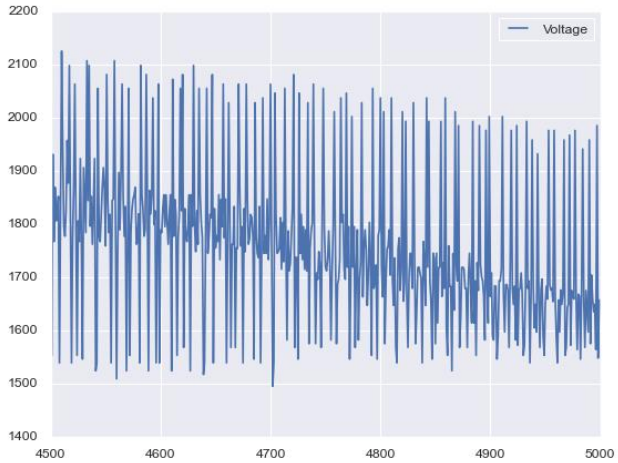


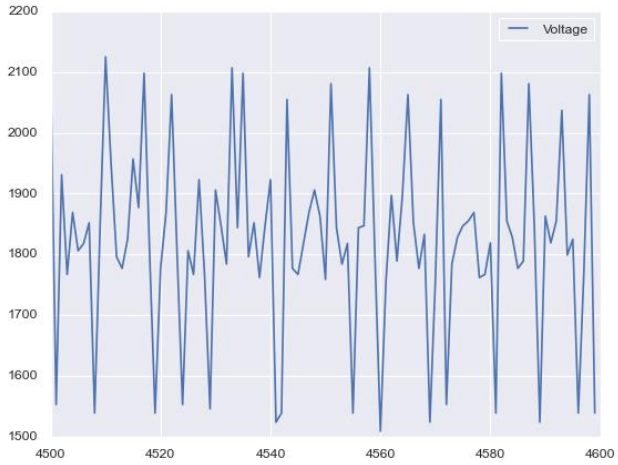
Third step in preprocessing was aimed at creating a data frame of regular, tabular structure to serve in further analyses and transformation (all values from voltage and current curves that were in a form of array were taken out and put in a table while welding time was incremented by 1ms for each value of the array; *data_plain.csv* and *data_plain_.csv*). Visualization of the entire dataset was impossible, due to large amount of data. Therefore, data were sampled, and figure bellow shows the distribution of the sampled values (1% of all data) of voltages and current, as well as their correlation per welding spot.



The figure shows that there is correlation between voltage and current levels.

Even the sampled dataset was too big to obtain a meaningful visualization of the values of voltage fluctuation. Therefore, a filter was applied, and voltage values form record indexed as 4500 to record indexed as 5000, as well as voltages between the records 4500 and 4600 are shown in the following figures.





The voltage fluctuation in the sampled and filtered data indicated that it makes sense to use time series analysis for forecasting voltage values. This data (voltage in manufacturing) by its nature should not have trends, and if there is seasonality, it can be only at daily level since the data provided were from different months over two years.

Due to high density and huge amount of data (and the non-disclosure agreement), it was decided to perform time series analysis separately for each welding spot.

The **fourth step** in data preprocessing was aimed at creating a dataset with possible exogenous variables that might be needed subsequently (Current, Day of the Week, Hour of the Day, Time Stamp).

In the **fifth step** of preprocessing, the data were divided by welding spot and therefore 76 *.csv* files were created, containing, in addition to voltage, other variables that might be needed.

In the **sixth step** of the preprocessing data registered in 60 days over eight months were transformed in 60 consecutive days at each spot.

These datasets were resampled subsequently at the level of 1ms enabling them to be subjected to time series analysis.

Here is the directory structure of the original and prepared data:

```
└──Data
    ├──Original
    │   ├──2019
    │   │   ├──11_Novembre
    │   │   ├──12_Dicembre
    │   │   └──9_Settembre
    │   └──2020
    │       ├──1_Gennaio
    │       ├──2_Febbraio
    │       ├──3_Marzo
    │       └──5_Maggio
    └──Prepared
        ├──by_spot
        ├──by_spot_
        ├──by_spot_succesive
        ├──by_spot_succesive_
        ├──by_spot_succesive_resampled
        ├──by_spot_succesive_resampled_
        ├──by_spot_succesive_resampled_1ms
        └──by_spot_succesive_resampled_1ms_
```

Models

Since data provided were extracts covering just part of a working day of welding guns, and covering only 60 nonconsecutive days spanned over eight months, it seemed logical to proceed to direct application of the models, without checking first for the eventual seasonality,

stationarity or autocorrelation. In addition, with this enormous amount of high-density data, it would have taken too much time or even would have been impossible to check these characteristics working only on my computer, despite being a powerful machine (as I understood, the non-disclosure agreement didn't allow me to use publicly open resources).

Therefore, I've chosen diverse models, ones more appropriate for stationary data and others more appropriate for data with seasonality or trends. The application of the models was made easier using the **sktime** [2] Phyton library, which features a unified interface for multiple time series learning tasks. At the time being, this open-source framework support forecasting, time series classification, time series regression and time series clustering.

Forecasting approaches used in this project can be classified in two groups, one encompassing traditional models (naïve, ARIMA, exponential smoothing), and one encompassing models based on machine learning (ML) techniques (Linear regression and Random Forest).

Naïve model is one of the simplest models for forecasting and here it will serve as a baseline model. According to this model, one step ahead is equal to the most recent actual value:

$$\hat{y}_t = y_{t-1}$$

ARIMA (p,d,q) i.e. **Autoregressive Integrated Moving Average** model is a linear (i.e., regression-type) model in which the predictors consist of lags of the dependent variable and/or lags of the forecast errors [3]:

$$\hat{y}_t = \mu + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} - \theta_1 e_{t-1} - \cdots - \theta_q e_{t-q}$$

where:

- The parameter p is the number of autoregressive terms or the number of "lag observations", also called the "lag order," and it determines the outcome of the model by providing lagged data points.
- The parameter d is known as the degree of differencing, and it indicates the number of times the lagged indicators have been subtracted to make the data stationary.
- The parameter q is the number of forecast errors in the model and is also referred to as the size of the moving average window.

In this project the **Auto ARIMA** model from the **sktime** [2] Phyton library was used. The auto-ARIMA algorithm seeks to identify the most optimal parameters for an ARIMA model, settling on a single fitted ARIMA model. Auto-ARIMA works by conducting differencing tests (i.e., Kwiatkowski–Phillips–Schmidt–Shin, Augmented Dickey-Fuller or Phillips–Perron) to determine the order of differencing, d, and then fitting models within predefined ranges. If the seasonal optional is enabled, auto-ARIMA also seeks to identify the optimal p and q hyper-parameters after conducting the Canova-Hansen to determine the optimal order of seasonal differencing, d [4].

Exponential Smoothing (ETS) encompasses a family of forecasting models each having the property that forecasts are weighted combinations of past observations, with recent observations being given relatively more weight than older observations. [5] The weights are exponentially decreasing over time, rather than the constant weights in simple moving average methods. The weights are dependent on a constant parameter, known as the smoothing parameter. [6] In this project **Auto ETS Model (AETS)** and **State Space Model** have been used.

When it comes to ML models available within the sktime framework, they can be used for forecasting thanks to reduction.

Reduction [4] is the concept of using an algorithm to solve a learning task that it was not designed for. It is the process of going from a complex learning task to a simpler one. Reduction can be used to transform a forecasting task into a tabular regression problem. This means that a forecasting task can be solved using scikit-learn's [7] estimators.

The key steps that take place in the reduction process are:

- Using a sliding window approach to split the training set into fixed-length windows. Example: if the window length is equal to 11, the process looks as follows: the 1st window contains data from time points 0–10 (where time points 0–9 become feature variables and time point 10 becomes the target variable). The 2nd window contains data from time points 1–11 (where 1–10 become feature variables and 11 becomes the target variable), etc.

- Arranging those windows on top of each other. As a result, data are given in tabular form, with clear distinction between feature and target variables.

- Subsequently one of the following strategies is used: recursive, direct, or multi-output, for generating forecasts.

In this project two ML models were used to forecast the voltage values: Linear Regression and Random Forest.

Linear Regression model assumes that the relationship between the dependent variable and regressors is linear. This relationship is modelled through a disturbance term or error variable ε — an unobserved random variable that adds "noise" to the linear relationship between the dependent variable and regressors.

The model takes the following form:

$$y_i = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p + \varepsilon_i \qquad i = 1, \ldots, n$$

The model is explained in detail in [8].

Random Forest, as explained in [9], operates by constructing a multitude of decision trees at training time and outputting the value that is mean/average prediction of the individual trees.

Training and Testing

Since data preprocessing was very time-consuming, and in order to avoid the situation of not being able to produce any results, I decided to proceed with the project by having two workflows: one **basic workflow** and the other **cross-validation workflow.**

Here is the directory structure produced after both workflows were executed.

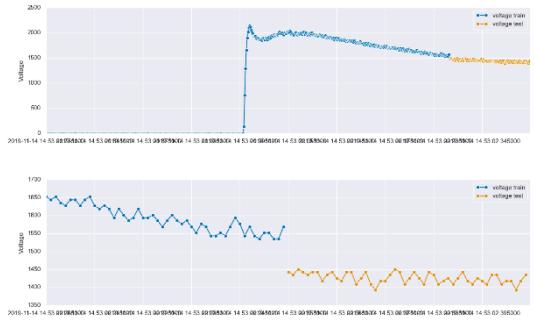
```
├──Data
│   └──Inference
│       ├──BasicWorkflow
│       │   ├──model_predict
│       │   ├──model_predict_
│       │   ├──train_test
│       │   ├──train_test_
│       │   ├──visual
│       │   └──visual_
│       └──CrossValidationWorkflow
│           ├──model_predict
│           ├──model_predict_
│           ├──train_test
│           ├──train_test_
│           ├──visual
│           └──visual_
├──Models
│   ├──BasicWorkflowBest
│   │   ├──models
│   │   ├──models_
│   │   ├──visual
│   │   └──visual_
│   ├──BasicWorkflowTest
│   │   ├──models
│   │   ├──models_
│   │   ├──visual
│   │   └──visual_
│   ├──CrossValidationWorkflowBest
│   │   ├──models
│   │   ├──models_
│   │   ├──visual
│   │   └──visual_
│   └──CrossValidationWorkflowTest
│       ├──models
│       ├──models_
│       ├──visual
│       └──visual_
└──Notebooks
```

Basic workflow was performed without cross-validation. The last dataset from data preprocessing, resampled at the level of 1 ms was split into training and testing set, using the appropriate temporal function. Since there is a delay of 50ms from the welding time till data registration, the last 50ms of the training set were excluded and then training and testing were performed (however these 50 data will be used afterwards for validation of quality of the best models).

It was impossible to make predictions from all available data at each welding spot, so I had to cut the training dataset and it was set to be the last 200.000 records (i.e. records corresponding to last 200s). Forecasting horizon was set to be 100 steps, corresponding therefore to future 100 ms. The dimensions of training set and forecasting window were set after trying to obtain predictions with bigger training set and forecasting window, and it was either impossible to obtain results or time for calculations was unacceptably long.

Training and testing sets for each welding spot were saved as *.csv* files (in Inference – BasicWorkFlow – train_test_ directory). For each spot, visualization of the last 500 records of the training set and the entire testing set, and last 50 records of both training and testing set were saved in the same directory.

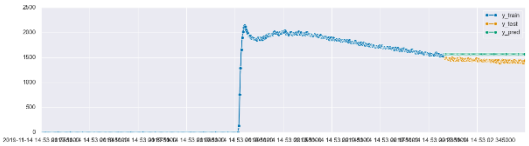
Following figures show respectfully these two visualizations at the welding spot named 60000_0_00.



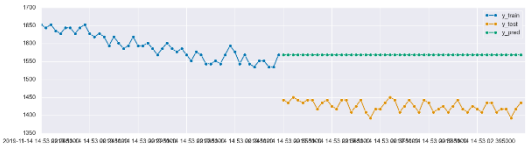
Six previously explained models were then applied to data from all welding spots. The parameters were fitted using *forecaster.fit* function, and then predictions were calculated using *forecaster.predict* function. It was impossible to train the data from 76 spots parallelly, so only few at a time were trained.

For each prediction the Mean Absolute Prediction Error (MAPE) was calculated as a measure of prediction accuracy. For each model, at each welding spot, model, prediction and MAPE have been saved as *.pkl* files.

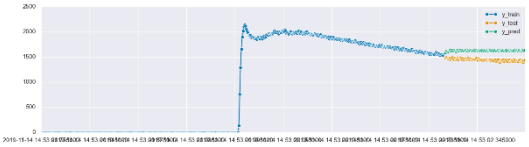
As an illustration of this process, here are the visualization of training-testing of all six models applied to spot named 60000_0_00 (as in previous figures, first figure relating to a model represent the last 500 records of training set and entire testing set, and the second one is relating to 50 last records of both training and testing sets). In all figures training set is in blue, testing set in yellow, while predicted values are green.



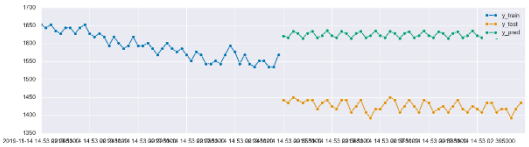
BWF_NAIVE_500



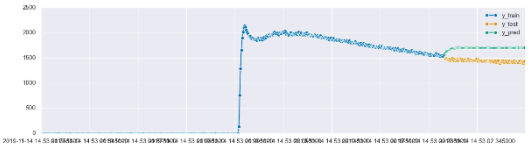
BWF_NAIVE_50



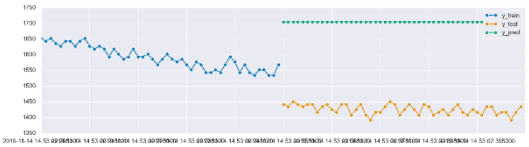
BWF_AARIMA_500



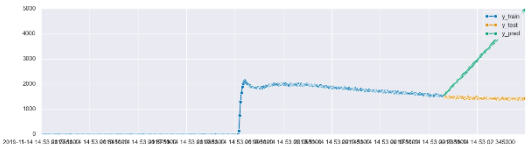
BWF_AARIMA_50



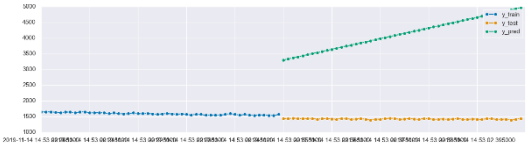
BWF_AETS_500



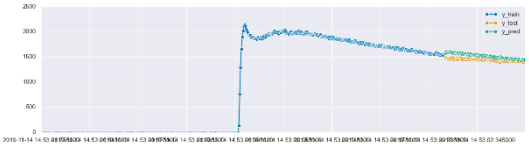
BWF_AETS_50



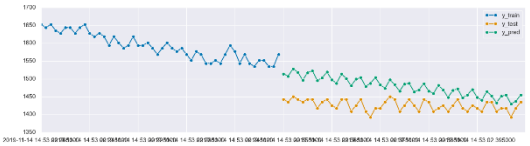
BWF_SpaceState_500



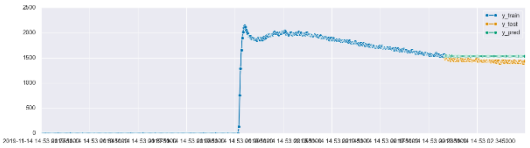
BWF_SpaceSate_50



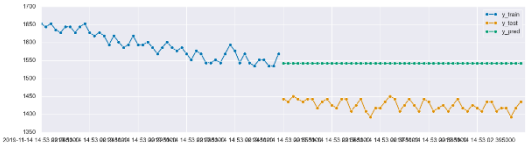
BWF_LinearRegression_500



BWF_LinearRegression_50



BWF_RandomForest_500



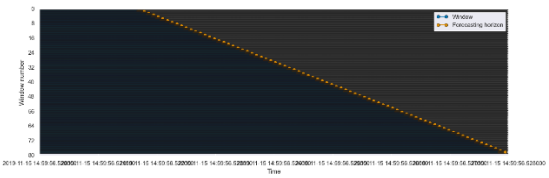
BWF_RandomForest_50

Subsequently, all *.pkl* files were unpacked and the best model (with lowest value of MAPE) was chosen for each welding spot, and then saved as a *.csv* files in Models – BasicWorkflowBest – models_ directory, as well as the graphical representation (…visual_ directory).

The next stage of the project was **testing**. Testing was performed with the best model chosen for each spot. Now the testing set was modified: the 50 last records excluded from the training set in the previous stage were added, but in order to respect the delay between welding time and data recording, testing set i.e. forecast window was set to 50: 50 last records corresponding to last 50 ms of the previous testing set (which was set to 100). Predictions have been saved as *.csv* in the Models-BasicWorkflowTest-models_ directory, while visualization in the …visual_ directory.

In the basic workflow the application of AARIMA model was applied to only one third of the spots, as it was extremely time-consuming.

As far as the **cross-validation workflow** is concerned, in general, it follows the logic of the basic workflow. However, here, instead of the temporal split function which divided the data in training and testing set, expanding window technique was used for cross validation. Namely, training set was set at 8000 records (unfortunately, again due to large amount od data that had to be processed I had to reduce the number of records until I reached a number my computer was able to process), with initial window consisting of 1600 records (1/5 of the overall length of the training set), and step length being 80 (1/100 of the overall length of the training set). The following figure is an illustration of the cross validation process at the spot 60001_0_00:



Light-blue lines on the left hand side represent each of the 80 expanding windows, and orange dots represent forecasting horizon for each of the 80 steps.

Training-testing process was performed for each model at each welding stop 80 times in order to complete the cross-validation and MAPE was calculated.

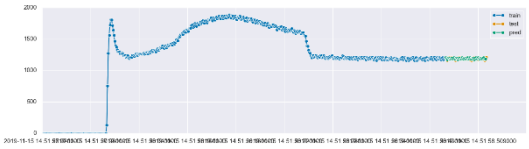
Therefore, at each spot, 80 MAPE were calculated for every model. An average value of MAPE was calculated subsequently per model, and, as in basic workflow, a pickle file comprising of forecaster, cv and average MAPE was saved in Inference-CrossValidationWorkflow-model_predict_ directory.

Unfortunately, the lack of processing power didn't allow the execution of the AARIMA model, as it had to automatically fit parameters 80 times per each of the 76 spots.

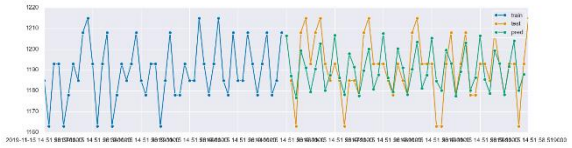
The best model at each spot was chosen among the other five models executed, and the prediction, as well as visualization, were saved in Models-CrossValidationWorkflowBest-models_ and …visual_ directory.

When it comes to **testing**, the logic of the basic workflow was followed and the dataset from the basic workflow was used (the one with 200.000 - 50 last records). Forecast horizon was reduced to 50 (the last 50 records) in order to resemble the delay in registering the data. However, it was impossible to perform the testing as in basic workflow – here the best model had to be fitted again when the 50 last records were added to the training set. After the fitting, testing of the best model was executed for each spot and the predictions and visualizations saved in the Models-CrossValidationWorkflowTest-models_ and …visual_ directory.

The following two figures illustrate the results of the testing at the spot name 60076_0_00, where the best prediction was obtained by random forest model. As before, training set is blue, testing set is yellow and prediction is green.



CVW_RandomForest_500



CVW_RandomForest_50

Conclusion

Available results have shown that, in the basic workflow, at 40% of spots the best predictions were obtained by linear regression model, at 30% of spots by naïve model, at 18% by AETS, at 7% by AARIMA and at 5% of spots by random forest model. State Space model was proven to be inadequate as it didn't perform best at any spot. It must be taken into consideration that AARIMA was applied only to one third of the spots, and therefore the percentage of the spots where AARIMA was the best model would've been higher than the one obtained.

In cross-validation workflow, the results are different: at 54% of spots the best results were obtained by naïve model, at 30% by linear regression model and at 16% by random forest.

It was expected to have different results in this workflow: the training set in the cross-validation model was 25 times smaller, and here the AARIMA model couldn't be performed due to lack of processing power.

Apart from many limitations that lack of processing power imposed on the execution of the project, there are several other important issues that have to be taken into consideration.

I believe the data available aren't the best choice for time-series analysis, especially without relevant information about the production process (working hours/days…) – we had days with records covering only 2.5s, and days covering almost an hour, and there were only 60 days over a period of 8 months.

The biggest issue, I believe, was how to create a regular time series. I have chosen to collate days (saving the information about the original day of the weak), maintaining the original working time. Since working time ranged from 2,5s to over 50 minutes per day, resampling at the level of 1ms, not only made the dataset enormous and hard for processing, but it made prevailing the periods without data (i.e. with 0s). This was the step that influenced the most the rest of the project. Firstly, because of the size of the data, I wasn't able to proceed with the analysis of the potential exogenous variables. Secondly, not having the exogenous variables, restricted the choice of the adequate model, as some of them (e.g. Extreme Gradient Boosting, Prophet…) need exogenous variables for the forecasting. Having the 0s prevailing in the dataset, made the MAPE less relevant.

Every idea I might've had to improve the presented forecasting process, was not applicable in practice, as I didn't have enough time/processing power after the data transformation was completed.
Should I have the possibility of redoing the project, I certainly would not make the same choice regarding the time series creation. Most probably, I would collate data directly based on welding time, and not the day as it was done. Surely, a bit of information would have been lost, but that very information I saved, I haven't been able to use in this project.

References

[1] D. Porteous, "Chrysler and the 700-Robot Butterfly System," 07 2021. [Online]. Available: https://emag.directindustry.com/2015/12/19/chrysler-and-the-700-robot-butterfly-system/.

[2] M. Löning, "sktime/sktime: v0.13.4," Zenodo, 2022. [Online]. Available: https://zenodo.org/record/7117735#.Y8sF5XbMK5d.

[3] R. Nau, "Statistical forecasting: notes on regression and time series analysis," [Online]. Available: https://people.duke.edu/~rnau/411arim.htm.

[4] M. Löning, A. Bagnall, S. Ganesh, V. Kazakov, J. Lines and F. J. Király, "sktime: A Unified Interface for Machine Learning with Time Series," in *Thirty-third Conference on Neural Information Processing Systems - NeurIPS 2019*, 2019.

[5] R. J. Hyndman and G. Athanasopoulos, Forecasting: Principles and Practice, OTexts, 2018.

[6] "Amazon Forecast Developer Guide," [Online]. Available: https://docs.aws.amazon.com/forecast/latest/dg/aws-forecast-recipe-ets.html.

[7] "Scikit-Learn," [Online]. Available: https://scikit-learn.org/stable/.

[8] K. P. Murphy, "Linear Regression," in *Machine Learning - A Probabilistic Perspective*, Cambridge, Massachusetts, MIT Press, 2012, pp. 217-243.

[9] K. P. Murphy, "Classification and Regression Trees CART," in *Machine Learning - A Probabilistic Perspective*, Cambridge, Massachusetts, MIT Press, 2012, pp. 544-552.