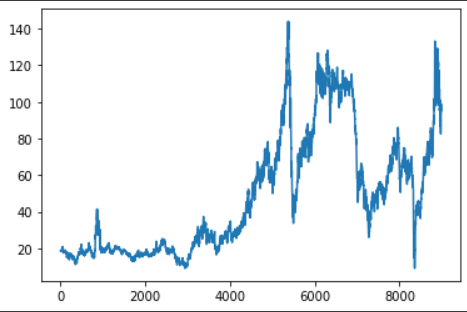
Data Cleaning:

1. After analysing the data we spotted some missing values, these represented 2.7% of the total data so we decided to remove them. We could not impute them with (mean/mode) because the time series is very (volatile/seasonal). We could not develop a ML algorithm for each cell because that would be extremely difficult and time consuming. Finally doing a forward fill in python could have worked but the outcome is almost the same as just deleting the missing cells.
2. After running some forecasts we decided that the data before 2005 was skewing the prediction since it was much less volatile and homogenous. We cut that part off and started from 2005 onwards.



1. Later on, we will explain why we didn’t include more variables into the prediction model.
2. Matrix Profile:

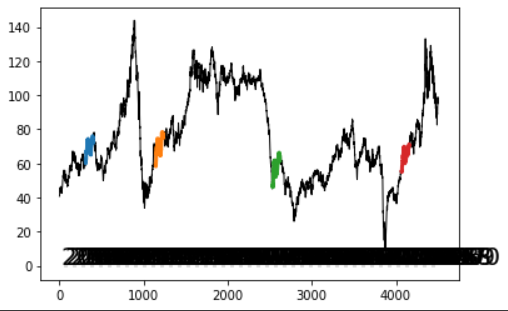
We decided to conduct a matrix profile analysis to see if we could find motifs or anomalies within the time-series that could aid us in the forecasting step. If repeating patterns were evidenced then we would include these in either the forecast or the recommender strategy.

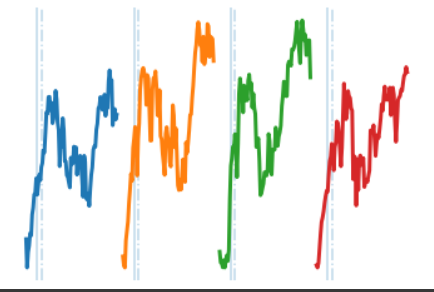
The advantages of using the Matrix Profile (over hashing, indexing, brute forcing a dimensionality reduced representation etc.) for most time series data mining tasks include:

1. It is **exact**: For motif discovery, discord discovery, time series joins etc., the Matrix Profile based methods provide no false positives or false dismissals.
2. It is **simple** and **parameter-free**: In contrast, the more general spatial access method algorithms typically require building and tuning spatial access methods and/or hash function.
3. It is **space efficient**: Matrix Profile construction algorithms requires an inconsequential space overhead, just linear in the time series length with a small constant factor, allowing massive datasets to be processed in main memory.
4. It allows **anytime algorithms**: While our exact algorithms are extremely scalable, for extremely large datasets we can compute the Matrix Profile in an anytime fashion, allowing ultra-fast approximate solutions.
5. It is **incrementally maintainable**: Having computed the Matrix Profile for a dataset, we can incrementally update it very efficiently. In many domains this means we can effectively maintain exact joins/motifs/discords on streaming data forever.
6. It **does not require the user to set similarity/distance threshold**s: For time series joins, the Matrix Profile provides *full* joins, eliminating the need to specify a similarity threshold, which is an unintuitive task for time series.
7. It can **leverage hardware**: Matrix Profile construction is embarrassingly parallelizable, both on multicore processors and in distributed systems.
8. It has **time complexity that is constant in subsequence length**: This is a very unusual and desirable property; all known time series join/motif/discord algorithms scale poorly as the subsequence length grows. In contrast, we have shown time series joins/motifs with subsequences lengths up to 100,000, at least two orders of magnitude longer than any other work we are aware of.
9. It can be constructed in **deterministic time**: All join/motif/discord algorithms we are aware of can radically different times to finish on two (even slightly) different datasets. In contrast, given only the length of the time series, we can precisely predict in advance how long it will take to compute the Matrix Profile
10. It can handle **missing data**: Even in the presence of missing data, we can provide answers which are guaranteed to have no false negatives.

Taken from (<https://www.cs.ucr.edu/~eamonn/MatrixProfile.html>)

After calculating the matrix for various time windows, (90 days/180 days/360 days) we did see repeating patterns/motifs within the time series. Unfortunately, after cross-checking possible relationships between these patterns we came to the conclusion that these patterns were not related to each other in any way. These were the possible events that triggered the similar patterns within the time series:





2006:

* The ongoing conflict in the Middle East and tensions between the US and Iran over the latter's nuclear program raised concerns over the stability of oil supplies and contributed to a rise in oil prices.

2009:

* The global financial crisis led to a sharp drop in oil demand, causing prices to fall.

2015:

* The global oil market was heavily oversupplied, leading to a sharp decline in oil prices. This was due to a combination of factors, including increased production from US shale oil, a slowdown in the global economy, and a decision by the Organization of the Petroleum Exporting Countries (OPEC) to maintain high levels of production.

2021:

* The COVID-19 pandemic led to a sharp decrease in global oil demand as lockdowns and travel restrictions reduced consumption. However, as the world began to recover from the pandemic, demand for oil increased, driving prices higher.

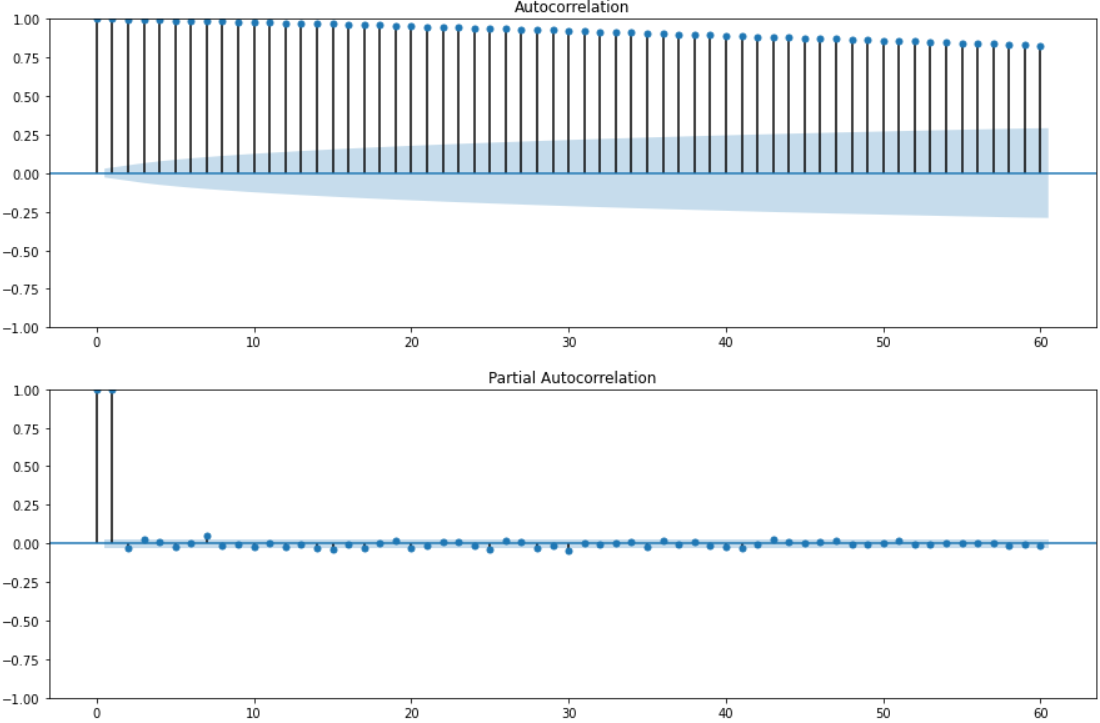
It is worth noting that these events were not necessarily the only factors affecting oil prices during these years, but they were among the most significant.

All the events that I mentioned had an impact on oil prices because they affected global oil demand or supply. In general, any event that changes the balance between supply and demand has the potential to influence oil prices. For example, geopolitical tensions that raise concerns about the stability of oil supplies will typically lead to higher prices, while economic slowdowns that decrease demand will tend to lower prices. Disruptions to the oil supply chain, such as natural disasters or infrastructure breakdowns, can also impact prices.

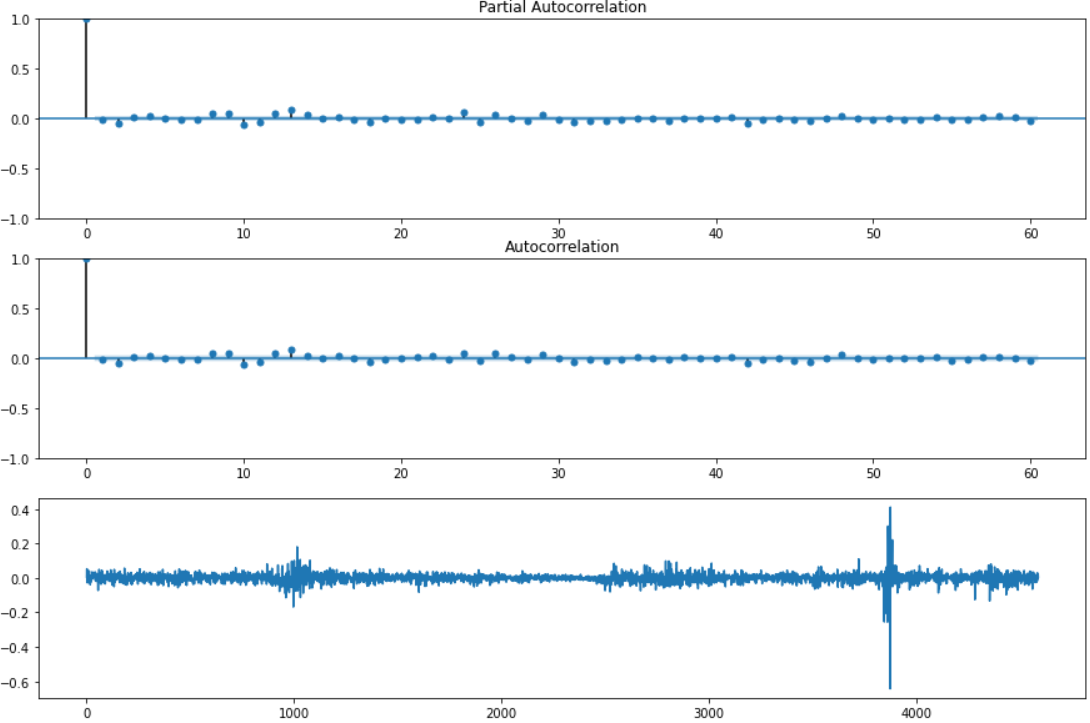
As it can be evidenced, the events don’t present a clear correlation or causality between them, **which resulted in discarding the matrix profile methodology as a means of forecasting. (all geopolitical events, too volatile and difficult to quantify numerically or categorically).Part of the process is abandoning approaches that don’t contribute.**

2. GARCH:

We decided to run a GARCH model to see if we could improve our prophet prediction.

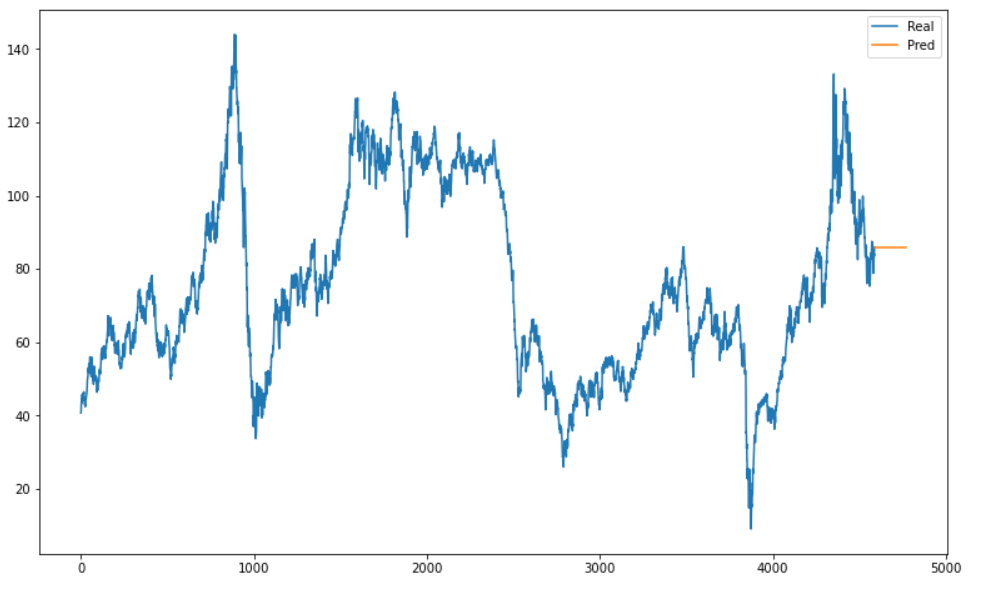


From here we can se 2 things. The ACF shows a gradual decrease to 0 which means there's no stationarity and the PACF shows both sinusoidal lags and a first lag which is almost 100% correlated with lag 0. This means there is seasonality but it is hidden by the high volatility of the time series and also means the best point prediction for the next period is the last period.



Here we got rid of the very correlated lag by applying one regular difference to the model, however, by applying a log transformation, we can see the clusters of volatity that usually indicate a GARCH time series.

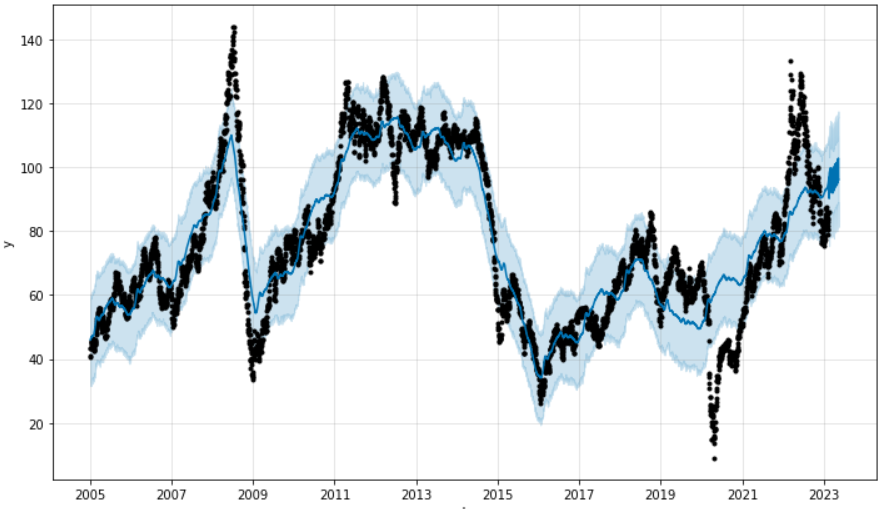
Due to the time series being of oil data which is extremely unpredictable and it being a string GARCH, the library couldn’t output an acceptable prediction, being just a straight horizontal line.

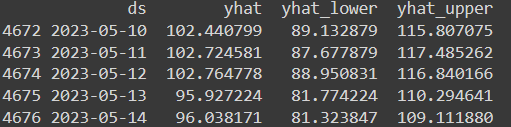


3. Prophet: Our final and best approach was using Prophet because of the following reasons:

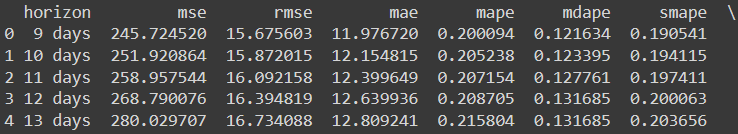
* Open source
* Designed for univariate time series: Only 1 variable is needed to train model
* Agnostic data friendly: The method or format of data transmission is irrelevant to the device or program’s function.
* Easy to use and designed to automatically find a good set of hyperparameters.
* Designed for data with trends and seasonality.

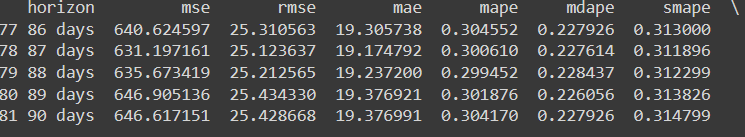
This code will first display the original series and then perform the prophet prediciton for a period of 90 days.

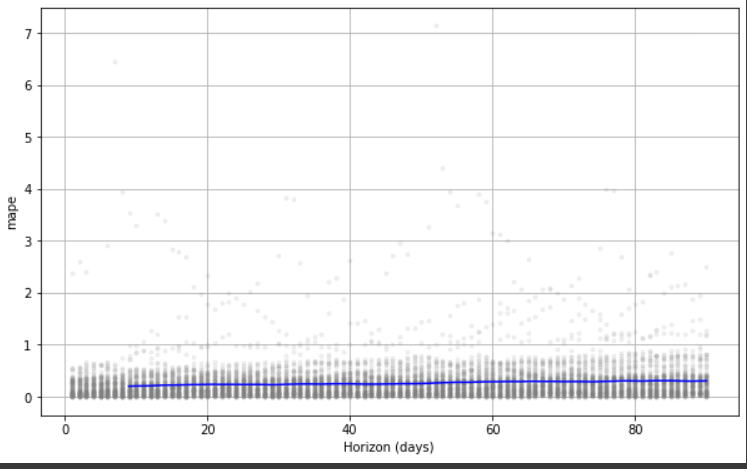




This piece of code will perform cross-validation on the regression to check and check the performance. It will split the data into proportional chunks and will iterate over a fold of 45 days. The prediction horizon will remain the same.







This part will show a chart with performance metrics such as MAPE and MSE. From the graph it can be inferred that as we increase our prediction horizon, the MAPE will be worse (increase) which is normal behaviour. Playing with the hyperparameters/number of iterations and k-folds didn’t decrease the MAPE.

**Conclusion:** Forecasting these types of commodities is extremely difficult no matter which algorithm/variables or data cleaning techniques are utilised. Hence we decided to focus on designing an optimal buying strategy that is going to minimise the costs for the company.