Ground Water Prediction Project

Intoduction

We are going to analyse and predict the depth to groundwater of an aquifer located in Petrignano, Italy.

The question we want to answer is:

What is the future depth to groundwater of a well belonging to the aquifier in Petrigrano over the next quarter?

The wells field of the alluvial plain between Ospedalicchio di Bastia Umbra and Petrignano is fed by three underground aquifers separated by low permeability septa. The aquifer can be considered a water table groundwater and is also fed by the Chiascio river. The groundwater levels are influenced by the following parameters: rainfall, depth to groundwater, temperatures and drainage volumes, level of the Chiascio river.

Indeed, both rainfall and temperature affect features like level, flow, depth to groundwater and hygrometry sometime after it fell down.

Data Collection

The data from the Acea Smart Water Analytics.

Link: (https://www.kaggle.com/competitions/acea-water-prediction/data)

Although the dataset contains multiple waterbodies, we will only be looking at the Aquifer_Petrignano.csv file.

Features:

- **Rainfall:** indicates the quantity of rain falling (mm)
- **Temperature:** indicates the temperature (°C)
- **Volume:** indicates the volume of water taken from the drinking water treatment plant (m 3)
- **Hydrometry:** indicates the groundwater level (m)

Target:

- **Depth to Groundwater:** indicates the groundwater level (m from the ground floor)

Contents

Section 1: Data Preprocessing

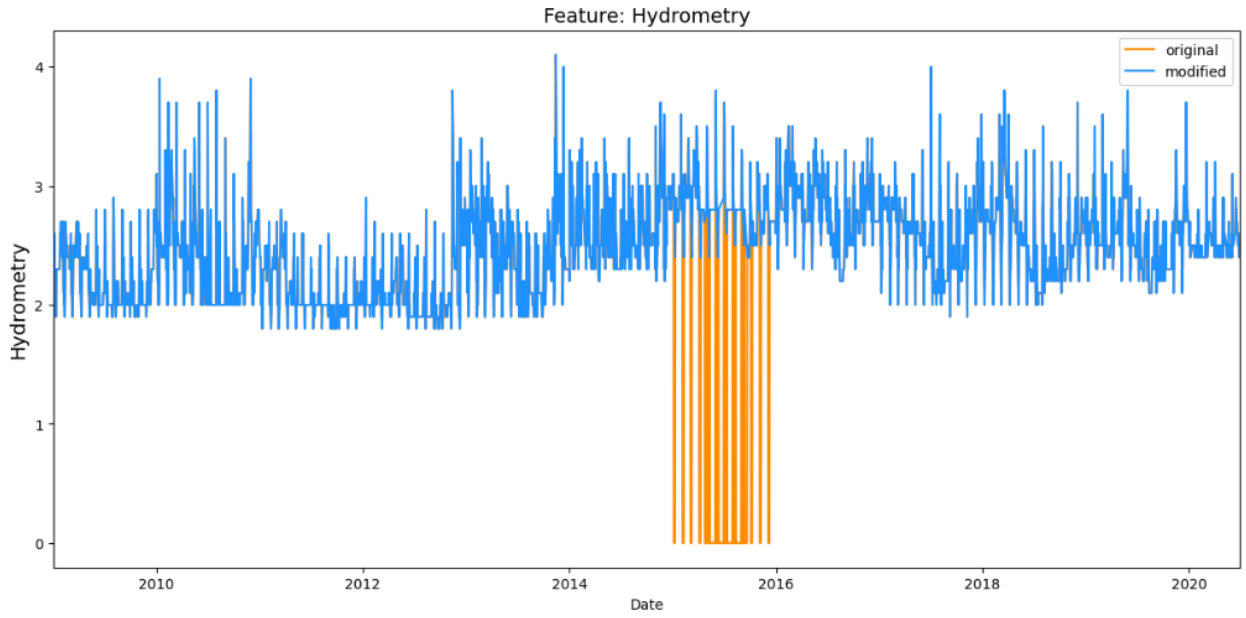
In the preprossing step we dropped the *'Temperature_Petrignano'* column to focus on only one target, therefore, we have 4199 rows and 6 columns in our data, which are: `'Date'`, `'Rainfall'`, `'Depth_to_Groundwater'`, `'Temperature'`, `'Drainage_Volume'`, `'River_Hydrometry'`.

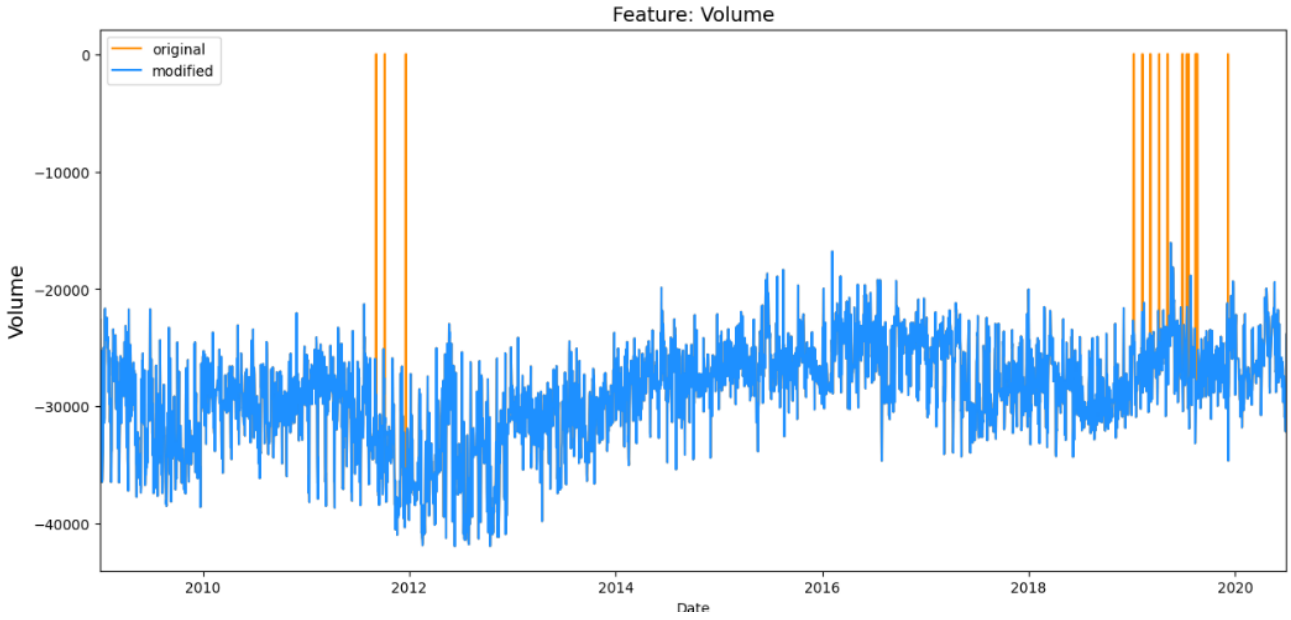
Section: 1.1 Missing Values

We saw that we had some missing values, as follows:

```
Date                    0
Rainfall                0
Depth_to_Groundwater    27
Temperature             0
Drainage_Volume         1
River_Hydrometry        0
dtype: int64
```

`'Depth_to_Groundwater'` – had 27 missing values and `'Drainage_Volume'` – had 1.



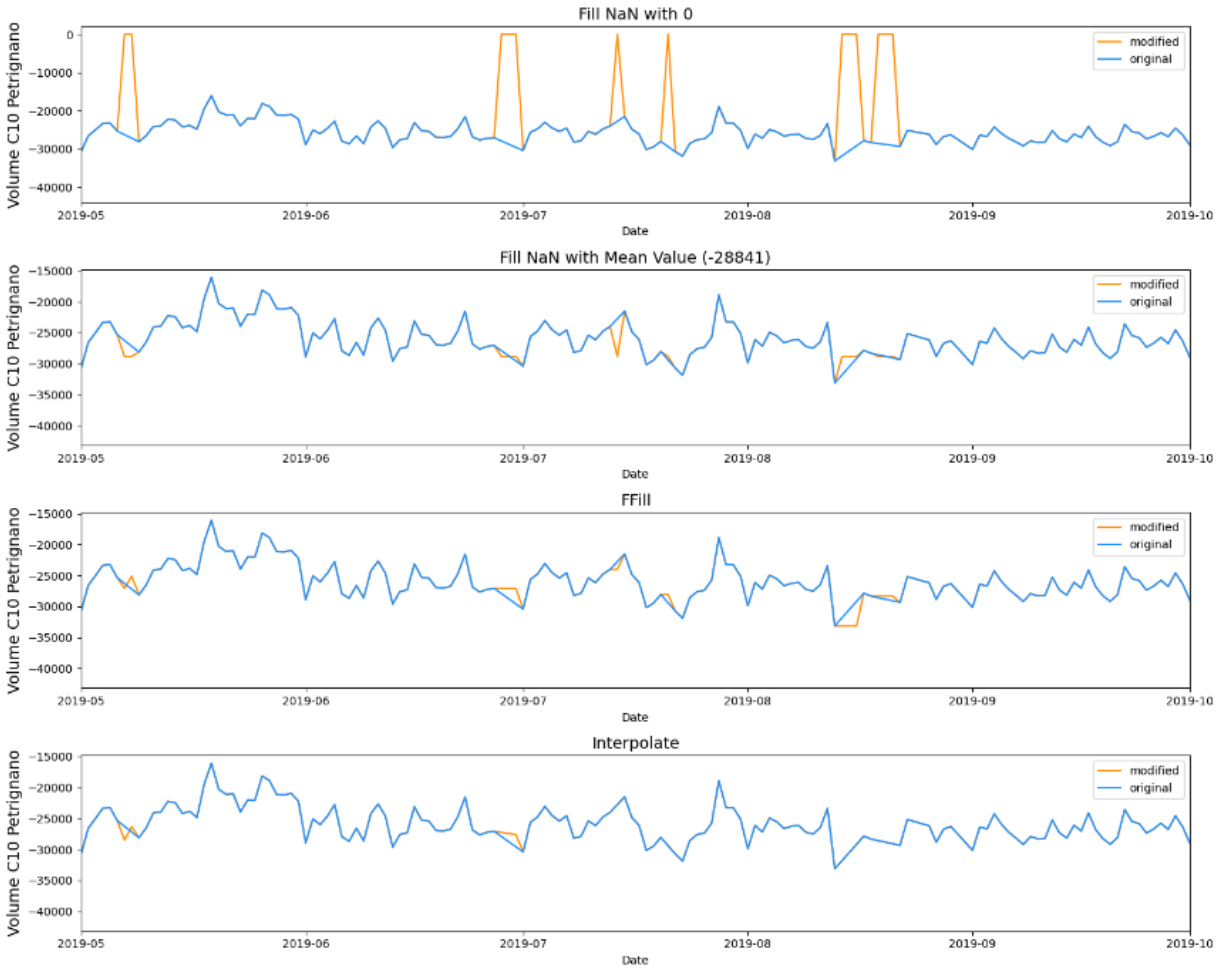


Furthermore, plotting the time series revealed that there were some implausible zero values for *Drainage_Volume*, and *River_Hydrometry*. We had to clean them by replacing them by nan values and filling them afterwards as in the above graphs(The orange colour represents the filled values).

Then, we handled the rest of the missing values as following:

1. **Fill NaN with Outlier or Zero**
   - Filling the missing value with an outlier value such as np.inf or 0 seems to be very naive. However, using values like -999, is sometimes a good idea.

2. **Fill NaN with Mean Value**
   - Filling NaNs with the mean value is also not sufficient and naive, and doesn't seems to be a good option.
3. **Fill NaN with Last Value with .ffill()**
   - Replacing NaNs with the most recent preceding value using forward fill (.ffill()) can be effective. This method maintains continuity in the data and can be suitable when the time sequence matters.
4. **Filling NaNs with the last value could be bit better.**
   - Fill NaN with Linearly Interpolated Value with .interpolate() Filling NaNs with the interpolated values is the best option in this small examlple but it requires knowledge of the neighouring value.



The orange colour shows the filled values. As we can see, the interpolate might be the best option.

Section 2: 1.2 Resampling

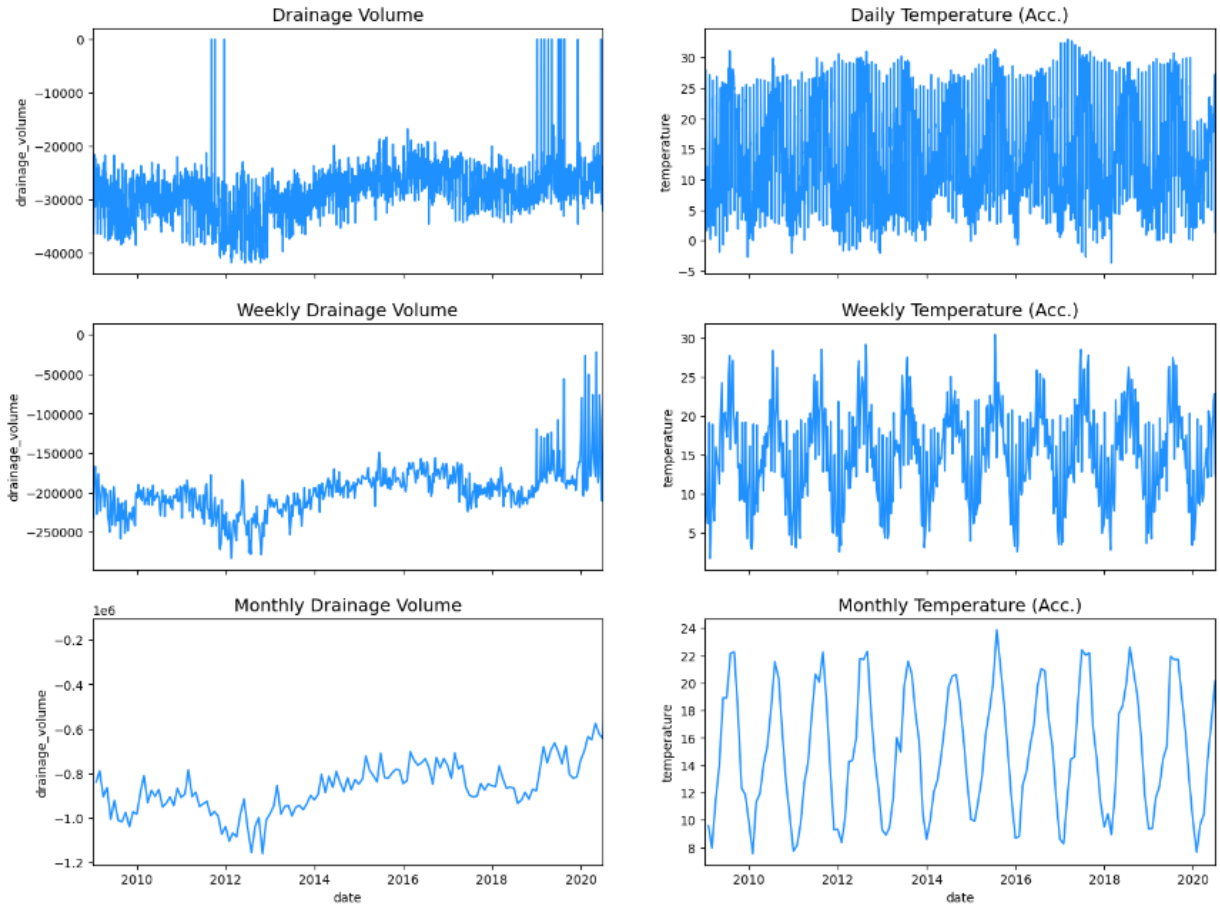
Resampling can provide additional information on the data. There are two types of resampling:

- Upsampling: is when the frequency of samples is increased (e.g. days to hours)
- Downsampling: is when the frequency of samples is decreased (e.g. days to weeks)

In our case, we will do some downsampling with the .resample() function (similar to groupby and aggregate as mean).

In this example, resampling would not be necessary. Considering weekly data seems to be sufficient as well. Therefore, we will downsample the data to a weekly basis.

These are how the columns look like after the resampling:



Section 3: 1.3 Stationarity

The check for stationarity can be done via three different approaches:

1. Visually: plot time series and check for trends or seasonality.
2. Basic statistics: split time series and compare the mean and variance of each partition.
3. Statistical test: Augmented Dickey Fuller test.

We are going to check by the statistical way, which by Augmented Dickey-Fuller (ADF)

Augmented Dickey-Fuller (ADF)

Augmented Dickey-Fuller (ADF) test is a type of statistical test called a unit root test. Unit roots are a cause for non-stationarity.

- Null Hypothesis (H0): Time series has a unit root. (Time series is not stationary).
- Alternate Hypothesis (H1): Time series has no unit root (Time series is stationary).

If the null hypothesis can be rejected, we can conclude that the time series is stationary.

There are two ways to rejects the null hypothesis:

On the one hand, the null hypothesis can be rejected if the p-value is below a set significance level. The defaults significance level is 5%.

1. **p-value > significance level (default: 0.05**): Fail to reject the null hypothesis (H0), the data has a unit root and is non-stationary.
2. **p-value <= significance level (default: 0.05):** Reject the null hypothesis (H0), the data does not have a unit root and is stationary. On the other hand, the null hypothesis can be rejects if the test statistic is less than the critical value.

- **ADF statistic > critical value:** Fail to reject the null hypothesis (H0), the data has a unit root and is non-stationary.
- **ADF statistic < critical value:** Reject the null hypothesis (H0), the data does not have a unit root and is stationary.

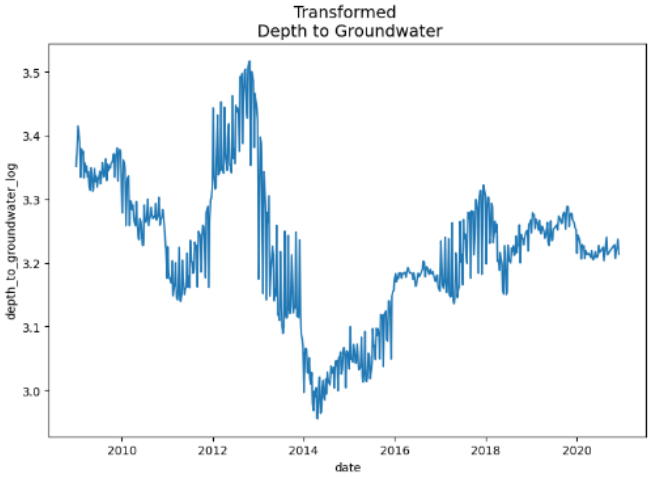
```
ADF Test Statistic : -2.406800339469522
p-value : 0.13982625511402186
#Lags Used : 18
Number of Observations Used : 604
Weak evidence against null hypothesis, time series has a unit root, indicating it is non-stationary
```

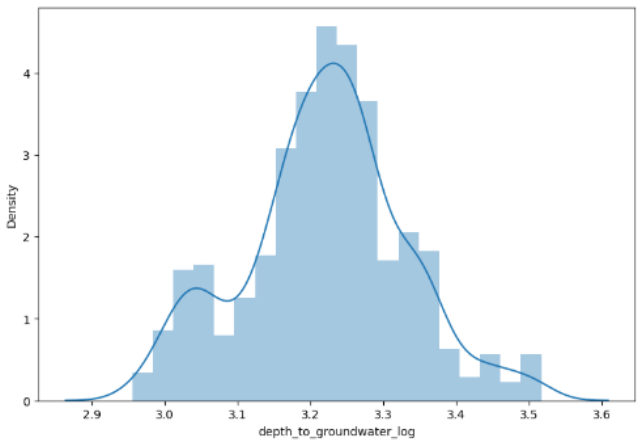
The above test indicated that our column is not stationary.

If the data is not stationary but we want to use a model such as ARIMA (that requires this characteristic), the data has to be transformed.

The two most common methods to transform series into stationarity ones are:

- Transformation: e.g. log or square root to stabilize non-constant variance
- Differencing: subtracts the current value from the previous





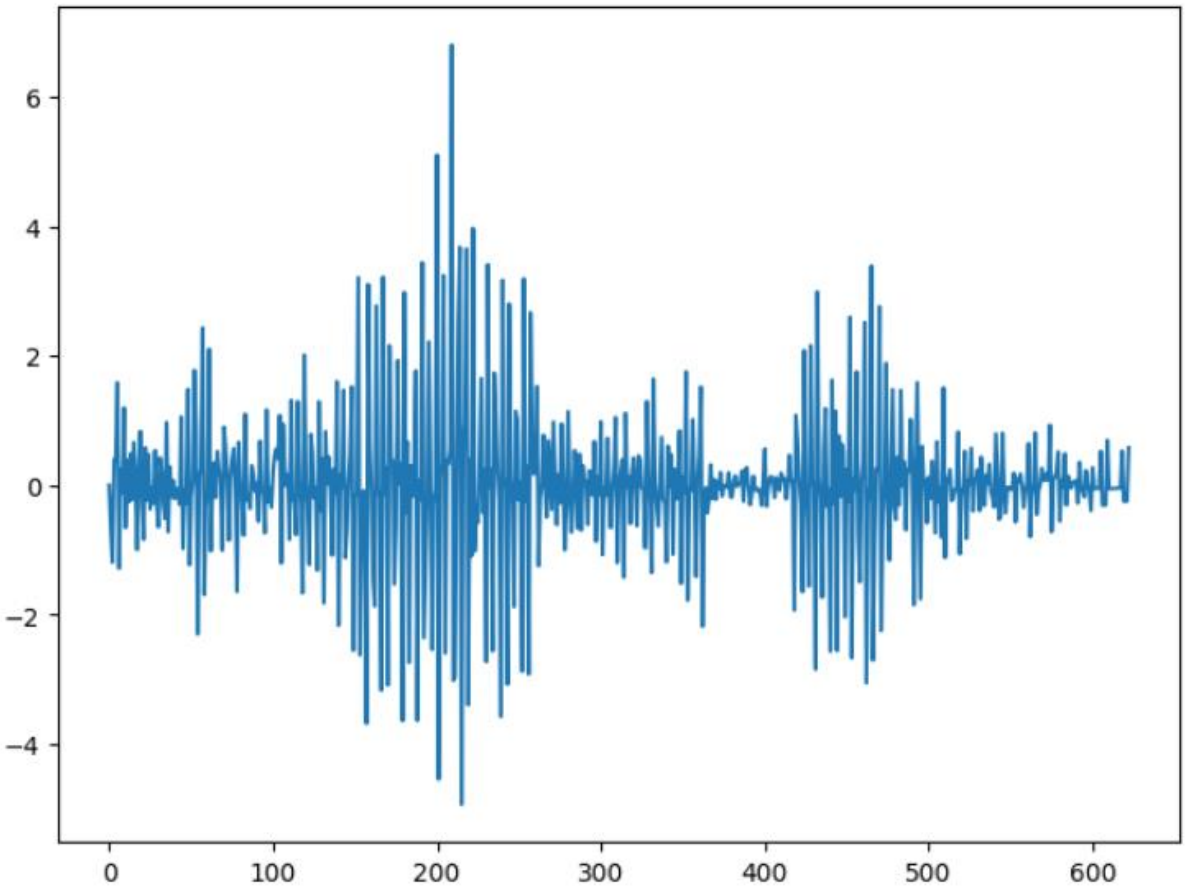
Section 4: 1.5 Transforming the data

After the Log Transform of absolute values we have the following chart:

Differencing can be done in different orders:

- First order differencing: linear trends with
- Second-order differencing: quadratic trends with and so on...

After differencing the first order we have the chart:



As we can see from the above chart our column is stationary now.

Section 5: Part 2. Feature Engineering

2.1 Decomposition

Time series decomposition involves thinking of a series as a combination of level, trend, seasonality, and noise components.

These components are defined as follows:

- Level: The average value in the series.
- Trend: The increasing or decreasing value in the series.

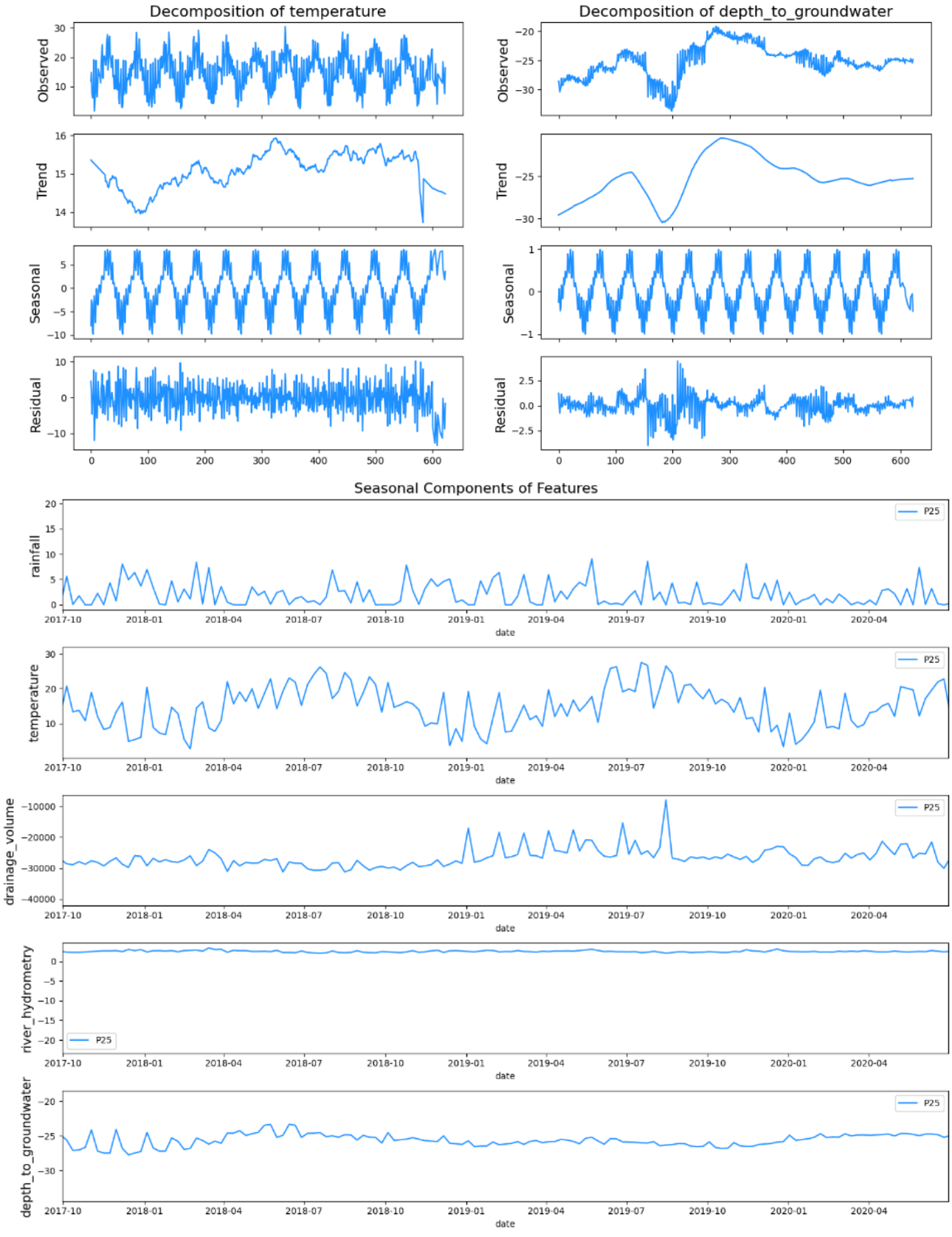
- Seasonality: The repeating short-term cycle in the series.
- Noise: The random variation in the series.

Decomposition provides a useful abstract model for thinking about time series generally and for better understanding problems during time series analysis and forecasting.

All series have a level and noise. The *trend* and *seasonality* components are optional.

Section 6: 2.2 Lag

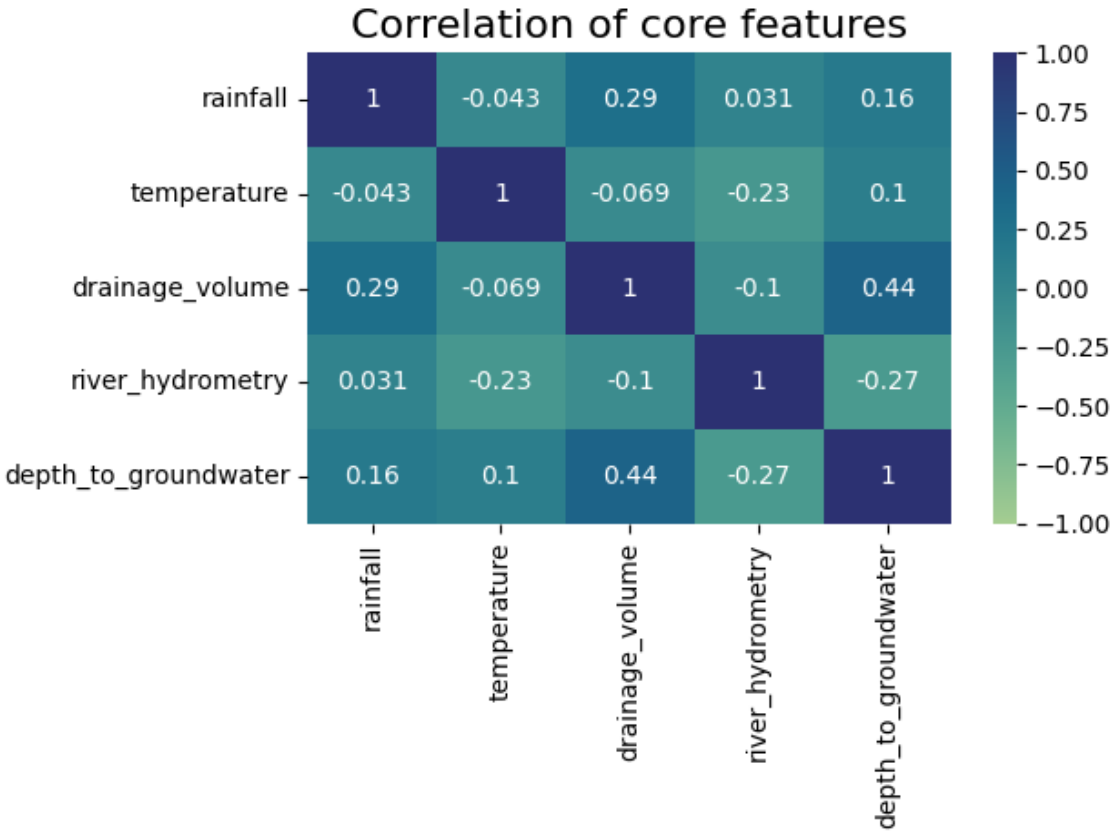
We want to calculate each variable with a shift() (lag) to compare the correlation with the other variables. Then, we are going to plot the data and try to extract some insights.



As we can see:

- **Depth_to_groundwater:** reaches its maximum around May/June and its minimum around November
- **Temperature:** reaches its maxmium around August and its minimum around January
- **Drainage_volume:** reaches its minimum around July.

Section 7: 3.1 Autocorrelation Analysis

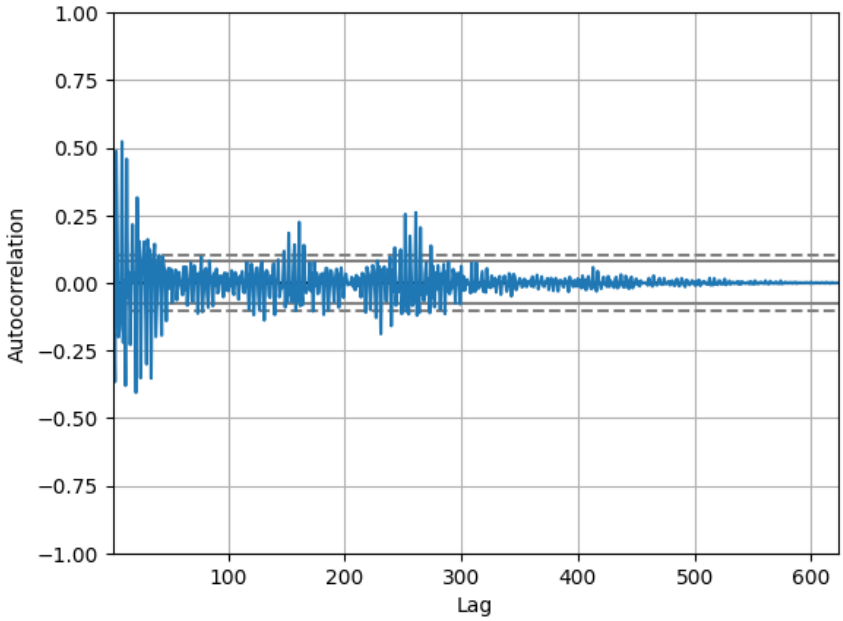


Autocorrelation, often denoted as ACF (Autocorrelation Function), and Partial Autocorrelation Function (PACF) are statistical tools used in time series analysis to understand and analyze the patterns and dependencies within a time series dataset. They help in identifying the presence of serial correlation, which is the correlation between a time series and its lagged versions (previous time points). Here's a brief explanation of each:

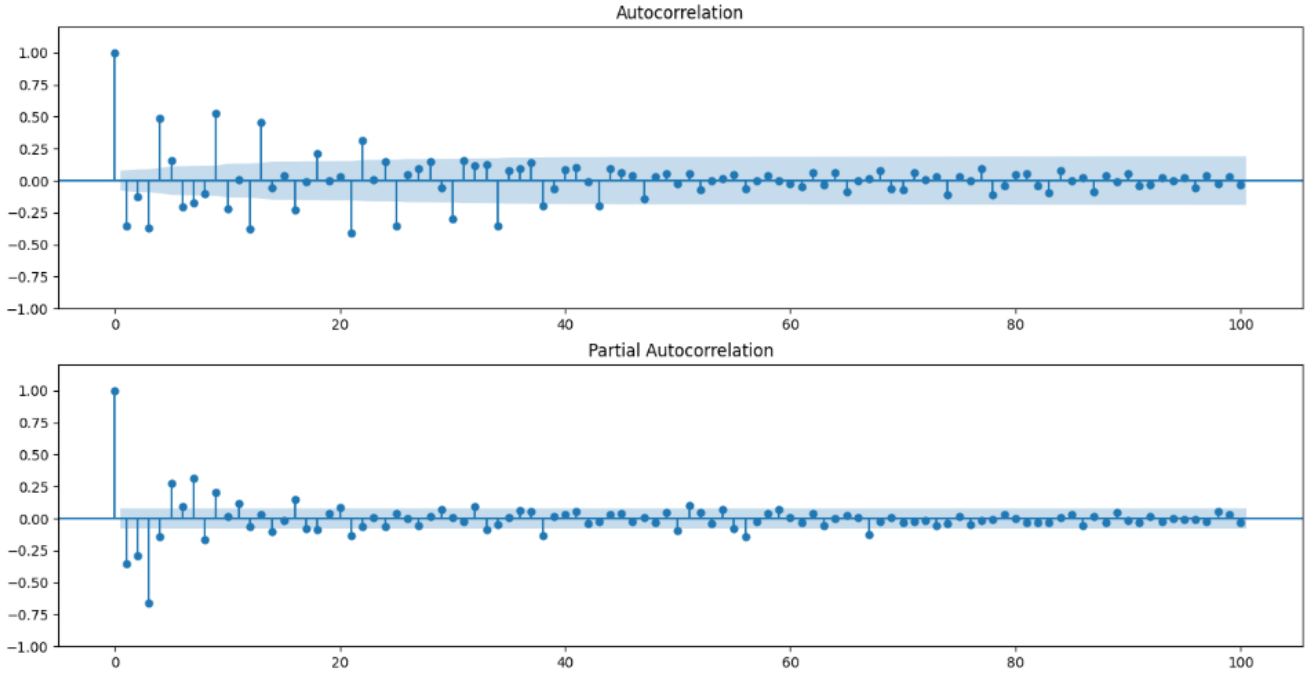
1. Autocorrelation Function (ACF):
    - Definition: ACF measures the correlation between a time series and its lagged values at different time intervals.
    - Purpose: It helps to identify the underlying patterns or trends in the time series. A strong autocorrelation at a particular lag indicates that the current value of the time series depends on its past values up to that lag.
    - Interpretation:
        - If ACF at lag 1 is high, it suggests a strong linear relationship between the current value and the previous value.
        - If ACF shows periodicity (e.g., every 12 lags), it might indicate a seasonal pattern in the data.
        - If ACF drops off quickly after a few lags, it suggests that most of the information in the series is contained within those lags.

2. Partial Autocorrelation Function (PACF):
    - Definition: PACF measures the correlation between a time series and its lagged values after removing the linear dependence of the series on the intervening lags.
    - Purpose: PACF helps in identifying the order of an autoregressive (AR) model. An AR model represents the current value as a linear combination of its past values. PACF helps to identify the number of past values that directly influence the current value, effectively showing the "pure" correlation.
- Interpretation:
    - A significant PACF value at lag k indicates that there's a direct relationship between the current value and the value k time units ago.
    - Non-significant PACF values at lags beyond the identified order suggest that those lags do not contribute significantly to predicting the current value.

Autocorrelation plot of ‘Depth_to_Groundwater’ of first difference looks like this:



Now, it is time to look at auto and partial correlation of the 'depth_to_groundwater_diff_1' column with 100 lags.



Section 8: 3.2 Train and Test Split

TimeSeriesSplit is a specific cross-validation technique in scikit-learn (sklearn) designed for time series data. It differs from traditional cross-validation methods because it takes into account the temporal order of data points, which is crucial in time series analysis.

The process of splitting:

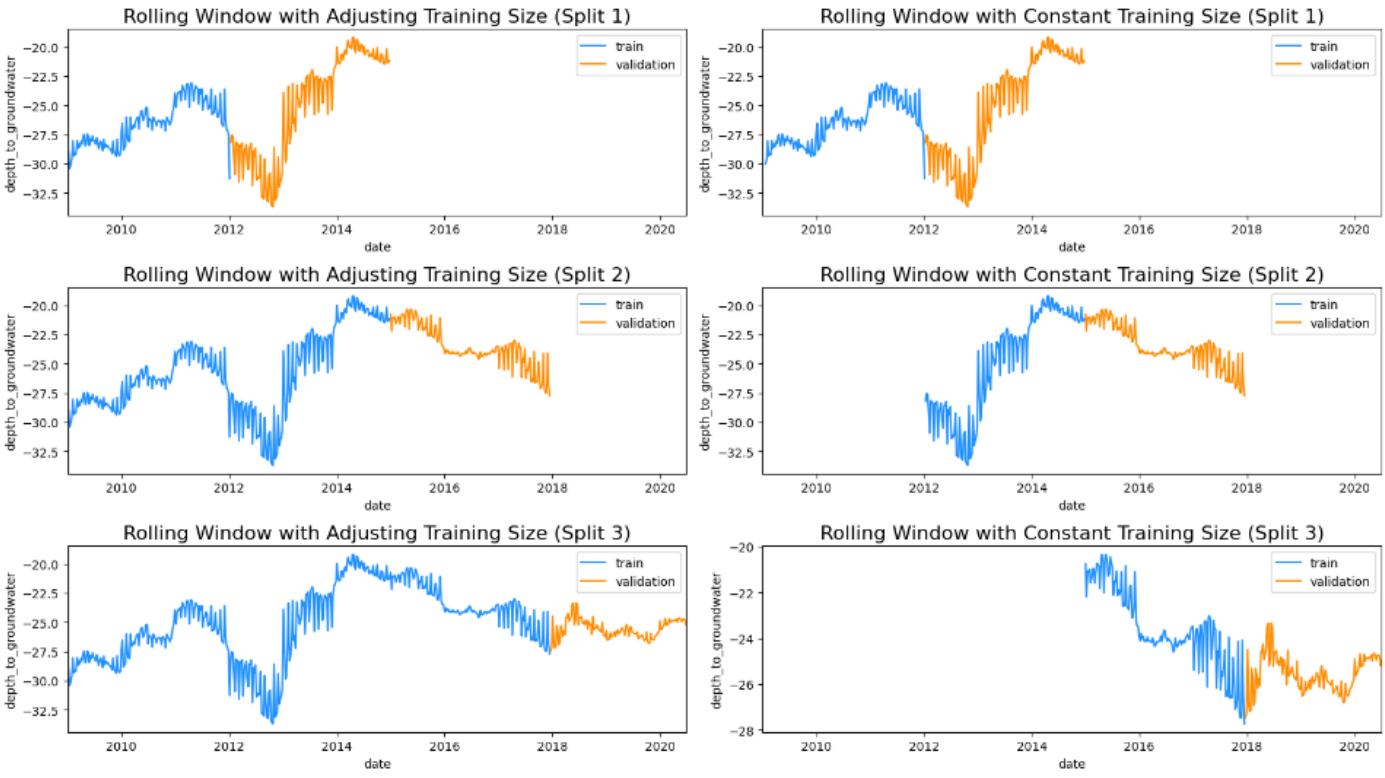
- **Data Preparation:** first, get your time series data ready, typically in the form of a pandas DataFrame or numpy array with a time-based index.
- **Initialization:** create an instance of the TimeSeriesSplit class, specifying the number of splits or folds you want. Each split represents a train-test pair where the data is split into consecutive time periods. The n_splits parameter determines how many of these pairs you want.
- **Splitting:** TimeSeriesSplit generates a set of indices for each fold. In our case, we will split in a 3-fold split.

Section 9: 3.3 Cross Validation

Model Training and Evaluation: You train and evaluate your model on each fold. Since the data in each fold is sequential, this helps simulate a more realistic scenario where you train on past data and test on future data.

Repeat: You can repeat this process for each fold, allowing you to assess how well your model generalizes to unseen future data. The performance metrics from each fold can be aggregated to get an overall measure of model performance.

After splitting and cross validation we can see the following chart:



Section 10: Part 4: Modelling

Time series can be either univariate or multivariate:

- Univariate time series only has a single time-dependent variable.
- Multivariate time series have a multiple time-dependent variable.

Our example originally is a multivariate time series because it has multiple features that are all time-dependent. However, by only looking at the target variable *Depth to Groundwater* we can convert it to a univariate time series.

We will evaluate the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE) of the models. For metrics are better the smaller they are.

Section 11: 4.1 Prophet

The first model (which also can handle multivariate problems) we are going to try is Facebook Prophet.

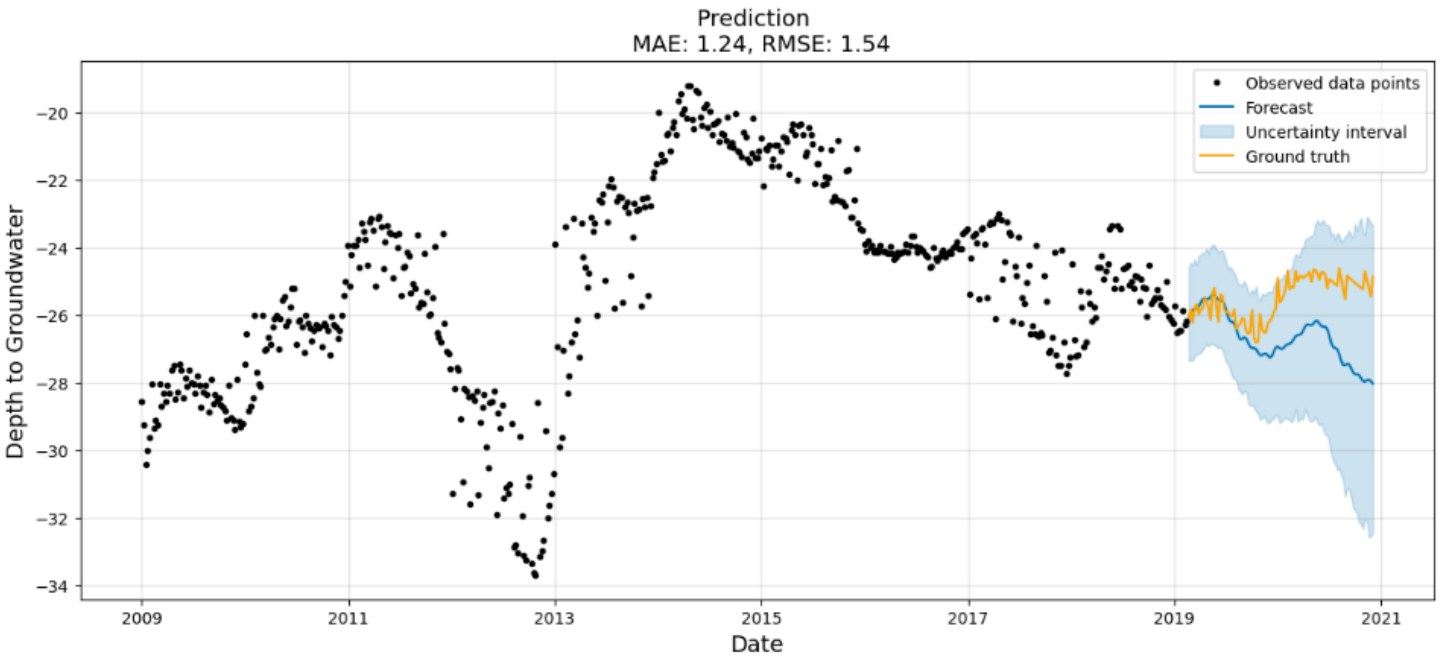
Prophet, or “Facebook Prophet,” is an open-source library for univariate (one variable) time series forecasting developed by Facebook.

Prophet implements what they refer to as an additive time series forecasting model, and the implementation supports trends, seasonality, and holidays.

The Prophet has:

RMSE: 1.5420012361144 89

MAE: 1.2373973862599126



Section 12: 4.2 ARIMA

The Auto-Regressive Integrated Moving Average (ARIMA) model describes the autocorrelations in the data. The model assumes that the time-series is stationary. It consists of three main parts:

1. p: Lag order (reference PACF in Autocorrelation Analysis)
2. d: Degree of differencing. (reference Differencing in Stationarity)
3. q: Order of moving average (check out ACF in Autocorrelation Analysis)

*Steps to analyze ARIMA *

1. Check stationarity: If a time series has a trend or seasonality component, it must be made stationary before we can use ARIMA to forecast.
2. Difference: If the time series is not stationary, it needs to be stationarized through differencing. Take the first difference, then check for stationarity. Take as many differences as it takes. Make sure you check seasonal differencing as well.
3. Filter out a validation sample: This will be used to validate how accurate our model is. Use train test validation split to achieve this
4. Select AR and MA terms: Use the ACF and PACF to decide whether to include an AR term(s), MA term(s), or both.
5. Build the model: Build the model and set the number of periods to forecast to N (depends on your needs).
6. Validate model: Compare the predicted values to the actuals in the validation sample.

The summary report looks like this:

Dep. Variable:	y	**No. Observations:**	529
Model:	ARIMA(1, 1, 1)	**Log Likelihood**	-769.650
Date:	Sat, 09 Mar 2024	**AIC**	1545.300
Time:	00:30:23	**BIC**	1558.107
Sample:	0 - 529	**HQIC**	1550.314

Covariance Type: opg

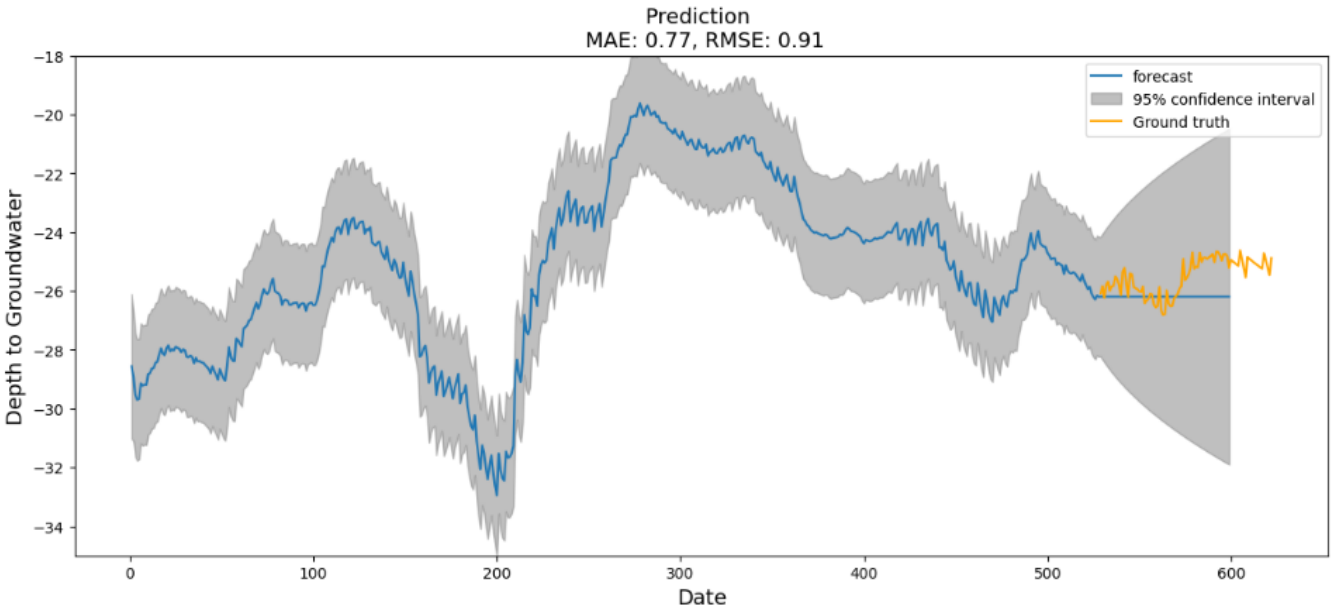
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.0157	0.062	0.254	0.799	-0.105	0.136
ma.L1	-0.6921	0.041	-16.723	0.000	-0.773	-0.611
sigma2	1.0792	0.032	33.784	0.000	1.017	1.142

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	1171.77
Prob(Q):	0.95	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.81	**Skew:**	0.83
Prob(H) (two-sided):	0.16	**Kurtosis:**	10.11

The ARIMA has:

RMSE: 0.9138188382174358

MAE: 0.7739767755202263



Section 13: 4.3 - Auto-ARIMA

AutoARIMA is a statistical modelling technique for time series forecasting that automates the process of selecting the order of an ARIMA (AutoRegressive Integrated Moving Average) model. The ARIMA model is a powerful and widely used method for forecasting time series data, but selecting the appropriate model order (p, d, q) can be a challenging and time-consuming task.

The summary of Auto-ARIMA

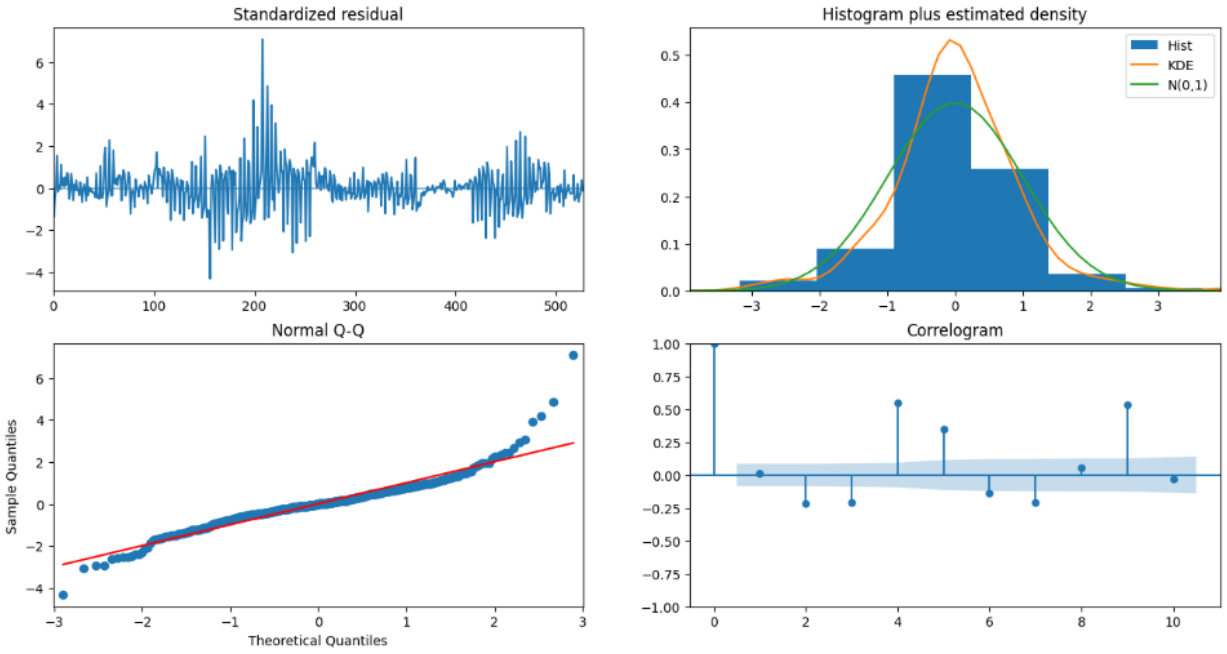
```
Performing stepwise search to minimize aic
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=1547.135, Time=0.88 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=1741.989, Time=0.04 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=1672.184, Time=0.09 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=1545.224, Time=0.14 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=1739.996, Time=0.03 sec
 ARIMA(0,1,2)(0,0,0)[0] intercept   : AIC=1547.010, Time=0.24 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=1547.904, Time=0.40 sec
 ARIMA(0,1,1)(0,0,0)[0]             : AIC=1543.390, Time=0.06 sec
 ARIMA(1,1,1)(0,0,0)[0]             : AIC=1545.300, Time=0.16 sec
 ARIMA(0,1,2)(0,0,0)[0]             : AIC=1545.175, Time=0.09 sec
 ARIMA(1,1,0)(0,0,0)[0]             : AIC=1670.200, Time=0.08 sec
 ARIMA(1,1,2)(0,0,0)[0]             : AIC=1546.069, Time=0.51 sec

Best model:  ARIMA(0,1,1)(0,0,0)[0]
Total fit time: 2.751 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  529
Model:               SARIMAX(0, 1, 1)   Log Likelihood                -769.695
Date:                Tue, 12 Mar 2024   AIC                           1543.390
Time:                        06:04:36   BIC                           1551.928
Sample:                             0   HQIC                          1546.733
                                  - 529
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -0.6873      0.024    -29.153      0.000      -0.734      -0.641
sigma2         1.0794      0.032     33.826      0.000       1.017       1.142
===================================================================================
Ljung-Box (L1) (Q):                   0.06   Jarque-Bera (JB):              1177.24
Prob(Q):                              0.81   Prob(JB):                         0.00
Heteroskedasticity (H):               0.81   Skew:                             0.83
Prob(H) (two-sided):                  0.16   Kurtosis:                        10.12
===================================================================================
```

The Auto – Arima has:

RMSE: 0.9146542146751206

MAE: 0.7747341325727144

For evaluating the mode we plotted diagnostic chart.



The function generates four key diagnostic plots for assessing an ARIMA model:

1. Time Series Plot with Residuals: This plot shows the residuals over time. Ideally, residuals should look like white noise, indicating that the model captures the underlying patterns.
2. Histogram Plus Estimated Density: This plot displays the distribution of residuals. It helps assess whether they approximately follow a normal distribution, a common assumption in statistical modelling.
3. Quantile-Quantile (Q-Q) Plot: This plot compares the quantiles of residuals to those of a theoretical normal distribution. If points align with the diagonal line, it suggests residuals are close to normal
4. Correlogram of Residuals: This plot shows autocorrelation in residuals. Lack of significant autocorrelation is desired; it indicates that the model captures time-dependent patterns.

Interpreting these plots helps evaluate if the ARIMA model is suitable for the time series data. Look for patterns, normality in residuals, and any remaining autocorrelation. Adjustments may be needed if issues are identified.

This proves that for our given data (0,1,1) is the best configuration for (p,q,d) of our ARIMA model.

Section 14: 4.4 LSTM

We are going to use a multi-layered LSTM recurrent neural network to predict the last value of a sequence of values.

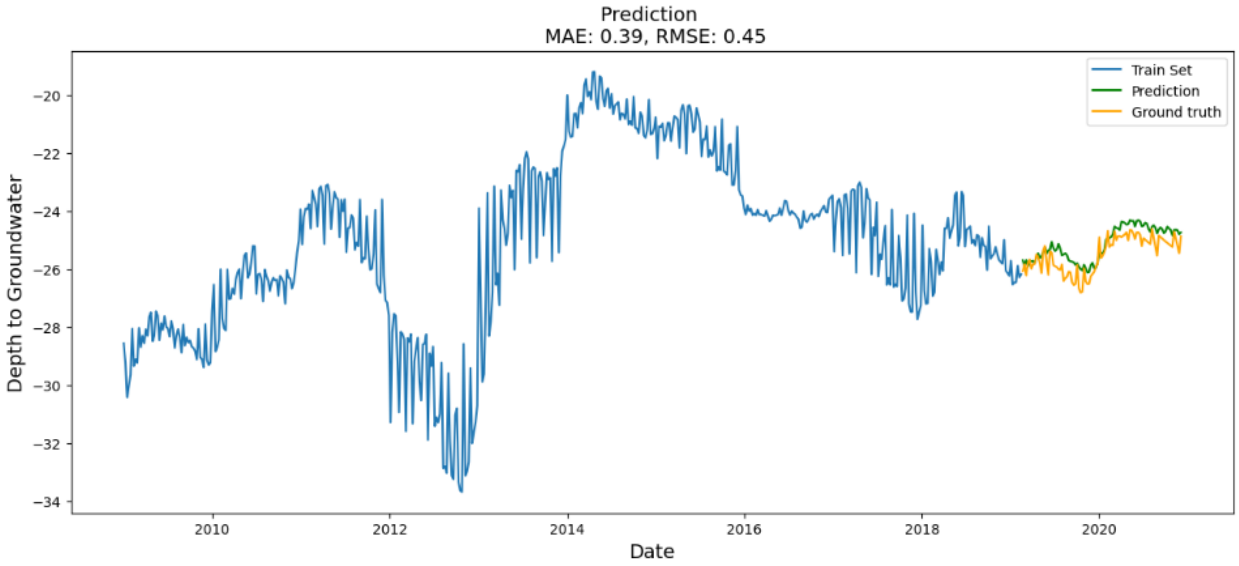
The following data pre-processing and feature engineering need to be done before construct the LSTM model:

1. Create the dataset, ensure all data is float.
2. Normalize the features.
3. Split into training and test sets.
4. Convert an array of values into a dataset matrix.
5. Reshape into X=t and Y=t+1
6. Reshape input to be 3D (num_samples, num_timesteps, num_features)

The LSTM has:

RMSE: 0.4548196970026224

MAE: 0.3908846915665366



Section 15: 5. Conclusion

*In conclusion, this comprehensive guide to time series analysis has covered key aspects from data visualization and preprocessing to advanced modelling techniques.

- We've explored methods for handling missing data
- Achieving stationarity
- Conducting critical tests like the ADF test.
- Transformations, feature engineering, and encoding cyclic features have been discussed to enhance model performance.
- Time series decomposition
- Lag analysis
- Partial Autocorrelation and Autocorrelation exploration have been instrumental in understanding data patterns.

We've also delved into both univariate and multivariate modeling approaches, including the use of Prophet, ARIMA, Auto – ARIMA, and LSTM models.

At the end we've discovered that the best models is LSTM (with MAE = 0.39 and RMSE = 0.45) for this analysis.