Springboard - Data Science Career Track

Capstone Project 2
Predicting Water Availability

By Ellen Savoye - June 12, 2021

Source: https://images.app.goo.gl/t2BoGqDTi4aakNrX8

Introduction

Water is an important part of everyday life. It's easy to take the seemingly endless supply for granted. However, the water supply isn't endless. To sustain our usage, water supply companies need to be able to forecast the amount of water in a given body of water under their purview to handle user consumption. These bodies of water could be a water spring, a lake, a river, or an aquifer. Each is affected by the changing of the seasons. Typically, bodies of water are replenished in the cooler seasons (autumn, winter) while warmer months (spring, summer) have a decrease in water level. Regardless of the season, it is important to be able to predict water availability, in terms of level or flow. Can a model be developed to predict water availability?

1

1. https://www.kaggle.com/c/acea-water-prediction/overview
2. https://towardsdatascience.com/stationarity-in-time-series-analysis-90c94f27322#:~:text=In%20t%20he%20most%20intuitive,not%20itself%20change%20over%20time
3. https://www.statsmodels.org/stable/generated/statsmodels.tsa.seasonal.seasonal_decompose.html
4. https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/
5. https://research.fb.com/blog/2017/02/prophet-forecasting-at-scale/

Acea Group, an Italian multi-utility operator, is invested in finding an answer. The company manages and develops water and electricity networks and environmental services. *Acea* is the foremost Italian operator in the water services sector supplying 9 million inhabitants in Lazio, Tuscany, Umbria, Molise, and Campania[1].

Data

The data used for this project is provided by *Acea*. It is publicly available at the following address: https://www.kaggle.com/c/acea-water-prediction/data. The data comes from *Acea*'s records. Each dataset, a CSV file, is independent of each other as the different water sources are all independent. My focus was on aquifers, of which there are 4.

When delving into the data, I noticed only 2 variables overlapped between the 4 datasets: date and rainfall. In wanting to create a model for aquifers in general, I decided to focus on one Aquifer in particular; Petrignano.

Data Munging

Within this aquifer, the date is captured daily. However, I noticed that data coverage before 2009 was spotty at best. Given the intent to do a time series analysis, I chose to remove all records before January 1, 2009; keeping years 2009 to 2020.

Year	Date	Rainfall_Bastia_Umbra	Depth_to_Groundwater_P24	Depth_to_Groundwater_P25	Temperature_Bastia_Umbra	Temperature_Petrignano	Volume_C10_Petrignano	Hydrometry_Fiume_Chiascio_Petrignano
2006	0	293	3	0	293	293	197	293
2007	0	365	2	2	365	365	0	365
2008	0	366	11	10	366	366	0	366
2009	0	0	0	0	0	0	0	0
2010	0	0	0	0	0	0	0	0
2011	0	0	0	0	0	0	0	0
2012	0	0	7	7	0	0	0	0
2013	0	0	21	0	0	0	0	0
2014	0	0	0	9	0	0	0	0
2015	0	0	0	0	0	0	0	0
2016	0	0	0	0	0	0	0	0
2017	0	0	0	0	0	0	0	0
2018	0	0	0	0	0	0	0	0
2019	0	0	7	7	0	0	0	0
2020	0	0	4	4	0	0	1	0

Exhibit 1

1. https://www.kaggle.com/c/acea-water-prediction/overview
2. https://towardsdatascience.com/stationarity-in-time-series-analysis-90c94f27322#:~:text=In%20t%20he%20most%20intuitive,not%20itself%20change%20over%20time
3. https://www.statsmodels.org/stable/generated/statsmodels.tsa.seasonal.seasonal_decompose.html
4. https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/
5. https://research.fb.com/blog/2017/02/prophet-forecasting-at-scale/

In looking at the missing values in exhibit 1, we can see missing values in depth_to_groundwater_P24 (P24) and depth_to_groundwater_P25 (P25); two separate groundwater levels calculated by a piezometer. Because the depth-to-groundwater variables are the target variables, the missing values present an issue as the time series analysis requires consecutive values. This presented me with two options. I could either aggregate to a less granular level of data, week, month or year, or I could try to fill in the missing values. As shown in the results below in exhibit 2, there are 39 and 27 missing values for P24 and P25, respectively, over 11 years. I made a copy of my dataset to apply forward linear interpolation to approximate the missing values. A quick check showed no more NULL values and confirmed my data was in chronological order with a time interval of 1 day.

```
1 Aq_Pet_df.isnull().sum()
```

```
Date                                  0
Rainfall_Bastia_Umbra                 0
Depth_to_Groundwater_P24             39
Depth_to_Groundwater_P25             27
Temperature_Bastia_Umbra              0
Temperature_Petrignano                0
Volume_C10_Petrignano                 1
Hydrometry_Fiume_Chiascio_Petrignano  0
Year                                  0
Month                                 0
Day                                   0
Quarter                               0
Day_of_year                           0
dtype: int64
```

Exhibit 2

Thinking I had covered all possible aspects of missing data, I graphed P24, P25 hydrometry, temperature, volume, and rainfall; exhibit 3. Immediately, the dip in 2015 temperature Petrignano values stands out as it overlays temperature Bastia Umbra. The distribution between the two temperature fields is nearly identical except for the span of 2015. A similar dip can be seen in Hydrometry. Given that these values did not show up as NA previously, I looked for zero values in all columns. There are a few in Volume and Temperature_Bastia_Umbra. However, both Temperature_Petrignano and Hydrometry show a significant amount of missing data, approximately 5 months worth of

```
1 (Aq_Pet_df_daily == 0).sum()
```

```
Date            0
Rainfall     1105
DG_P24          0
DG_P25          0
Temp_BU         4
Temp_P          0
Volume         22
Hydrometry      0
Year            0
Month           0
Day             0
Quarter         0
Day_of_year     0
dtype: int64
```

Exhibit 4

1. https://www.kaggle.com/c/acea-water-prediction/overview
2. https://towardsdatascience.com/stationarity-in-time-series-analysis-90c94f27322#:~:text=In%20t%20he%20most%20intuitive,not%20itself%20change%20over%20time
3. https://www.statsmodels.org/stable/generated/statsmodels.tsa.seasonal.seasonal_decompose.html
4. https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/
5. https://research.fb.com/blog/2017/02/prophet-forecasting-at-scale/

consecutive data. As there are over 10 years of data and the intent is to find the next consecutive day (or month) value, I'm comfortable in slimming the data down to 2016 and onward instead of trying to approximate 5 months worth of data. I believe this data set will still be robust enough to provide an excellent train/test set.

After dropping data before 2016, I checked for NULL, NAN, or 0 values. I found some missing values in temperature Bastia Umbra and volume. The zero values were converted to NAN before applying forward linear interpolation. Now that the data has been cleaned, it's time to explore stationarity and seasonality.

Aquifer Petrignano

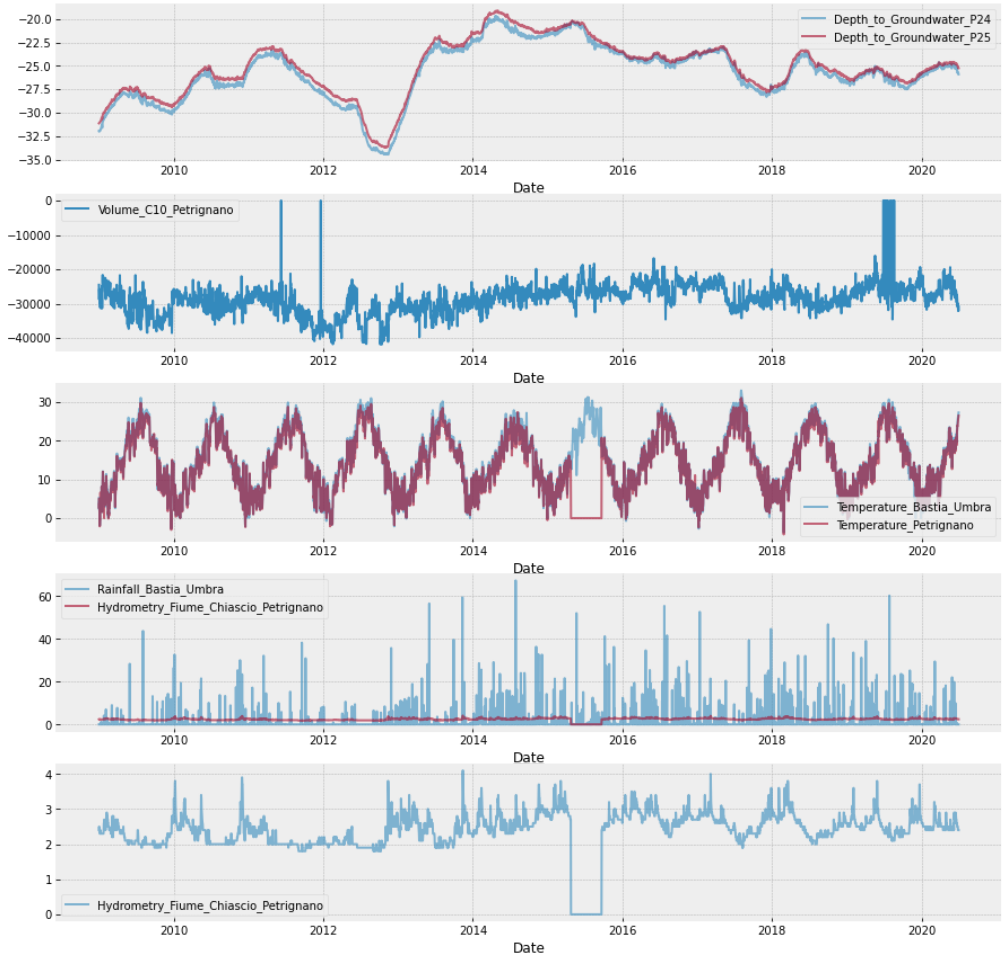


Exhibit 3

1. https://www.kaggle.com/c/acea-water-prediction/overview
2. https://towardsdatascience.com/stationarity-in-time-series-analysis-90c94f27322#:~:text=In%20t%20he%20most%20intuitive,not%20itself%20change%20over%20time
3. https://www.statsmodels.org/stable/generated/statsmodels.tsa.seasonal.seasonal_decompose.html
4. https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/
5. https://research.fb.com/blog/2017/02/prophet-forecasting-at-scale/

Exploring Stationarity and Seasonality

Some time-series models, such as ARIMA, assume the underlying data is stationary. Therefore, before applying any time-series models, stationarity needs to be checked. "[S]tationarity means that the statistical properties of a process generating a time series do not change over time. It does not mean that the series does not change over time, just that the way it changes does not itself change over time"[2]. Noticeable trends, seasonality, and changing levels are a few of the basic properties of non-stationary data. Looking at weekly temperature, we can see prominent seasonality. Volume clearly shows a noticeable trend and changing levels in exhibit 5 below.

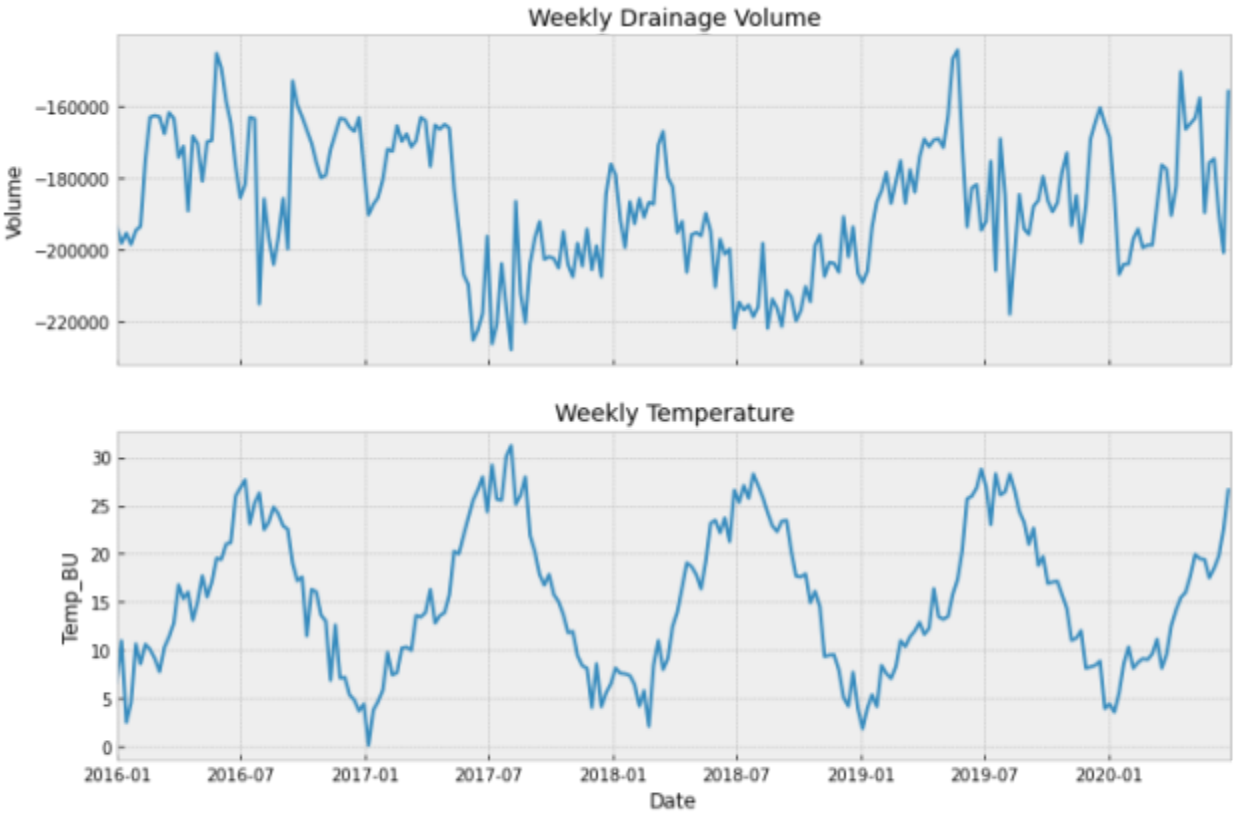


Exhibit 5

Based on my observations, my initial take is the data is non-stationary. However, I used the

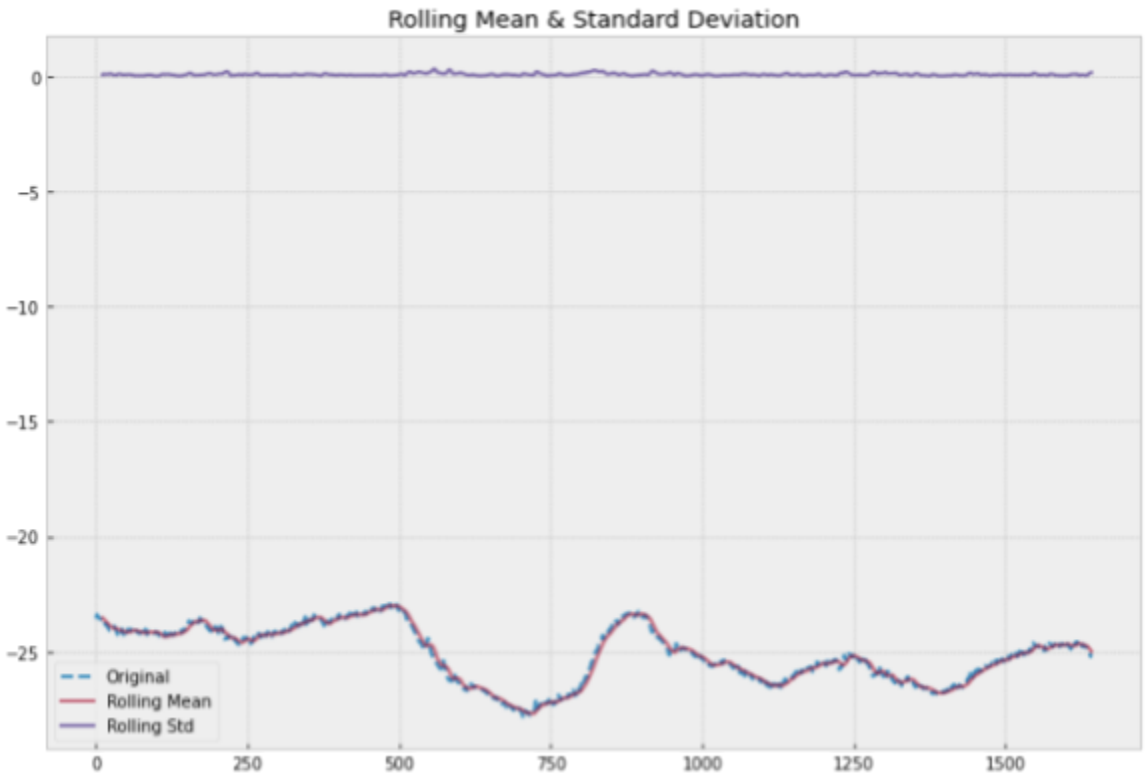
1. https://www.kaggle.com/c/acea-water-prediction/overview
2. https://towardsdatascience.com/stationarity-in-time-series-analysis-90c94f27322#:~:text=In%20t%20he%20most%20intuitive,not%20itself%20change%20over%20time
3. https://www.statsmodels.org/stable/generated/statsmodels.tsa.seasonal.seasonal_decompose.html
4. https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/
5. https://research.fb.com/blog/2017/02/prophet-forecasting-at-scale/

Augmented Dickey-Fuller (ADF) test to try to confirm my observation. The test determines if the unit root is present in the series, which will determine if the series is stationary or not.

Null and Alternate hypothesis for ADF Test:

- **Null**: the series has a unit root (value of a=1)
- **Alternate**: the series has no unit root

If we fail to reject the null hypothesis, we can say the series is non-stationary.




```
Results of Dickey-Fuller Test:
Test Statistic                  -2.447341
p-value                          0.128818
#Lags Used                      24.000000
Number of Observations Used    1618.000000
Critical Value (1%)              -3.434398
Critical Value (5%)              -2.863328
Critical Value (10%)             -2.567722
dtype: float64
```

Exhibit 6

1. https://www.kaggle.com/c/acea-water-prediction/overview
2. https://towardsdatascience.com/stationarity-in-time-series-analysis-90c94f27322#:~:text=In%20t%20he%20most%20intuitive,not%20itself%20change%20over%20time
3. https://www.statsmodels.org/stable/generated/statsmodels.tsa.seasonal.seasonal_decompose.html
4. https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/
5. https://research.fb.com/blog/2017/02/prophet-forecasting-at-scale/

If the test statistic is greater than the critical value, we reject the null hypothesis which means the series is not stationary. If the test statistic is less than the critical value, we fail to reject the null hypothesis therefore the series is stationary. Because our test statistic is greater than the critical values, as shown in exhibit 6, we reject the null hypothesis. Therefore, the series is not stationary.

The baseline model that I worked towards is the ARIMA model which requires the data to be stationary. As the current data wasn't stationary, the data needed to be transformed using either differencing or decomposition.

- **Decomposition** – modeling both trend and seasonality and removing them from the model
- **Differencing** – taking the difference with a particular time lag; i.e. 1 day

Decomposition

I looked to split the time series into pieces using decomposition. The pieces are as follows:

- Seasonal - periodic up-and-downs in the data
- Trend - pattern in the data spanning across seasonal periods
- Cyclical - phenomena that happen across seasonal periods
- Noise - random variation in the series

Seasonal, trend and cyclical components can either be added or multiplied together.

- Additive: $y(t) = seasonal + trend + cyclical + noise$
- Multiplicative: $y(t) = seasonal * trend * cyclical * noise$

Using the statsmodel library, I used the function seasonal_decompose()[3] to piece apart my time series. I left the default seasonal component set to 'Additive' for the decomposition of 'Temp_BU' and 'DG_P25' (P25). The decomposition of the two fields can be found in exhibit 7.

1. https://www.kaggle.com/c/acea-water-prediction/overview
2. https://towardsdatascience.com/stationarity-in-time-series-analysis-90c94f27322#:~:text=In%20t%20he%20most%20intuitive,not%20itself%20change%20over%20time
3. https://www.statsmodels.org/stable/generated/statsmodels.tsa.seasonal.seasonal_decompose.html
4. https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/
5. https://research.fb.com/blog/2017/02/prophet-forecasting-at-scale/

To check stationarity on the decomposed data, I used the ADF test again. Looking at the ADF test results on the residuals, I can see that decomposing my time series, by removing trend and seasonality, did not create a stationary series; exhibit 8.

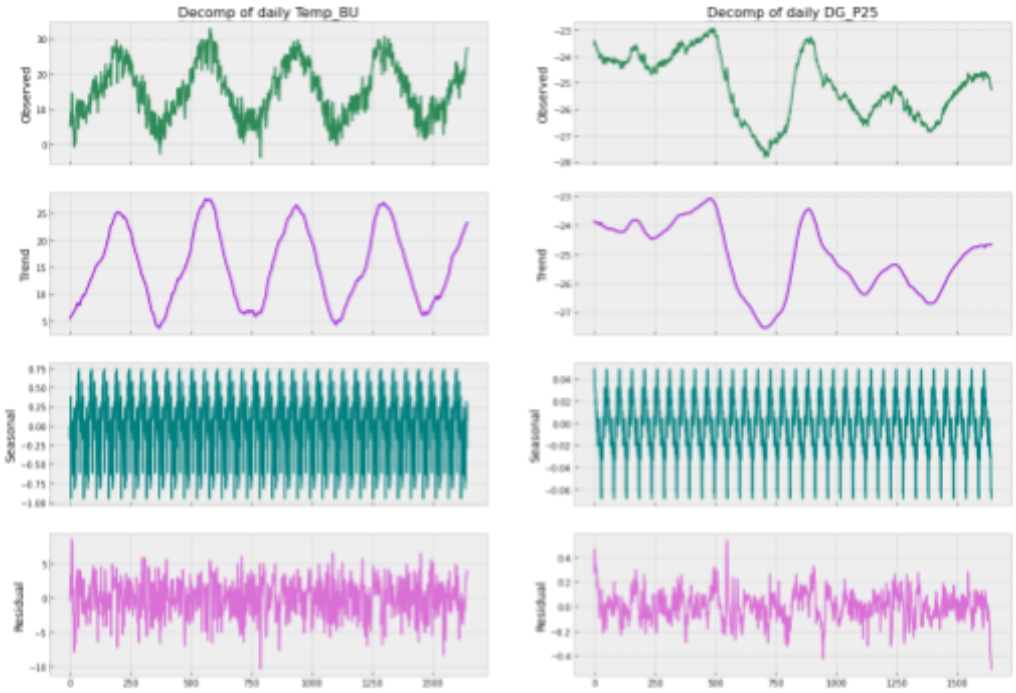
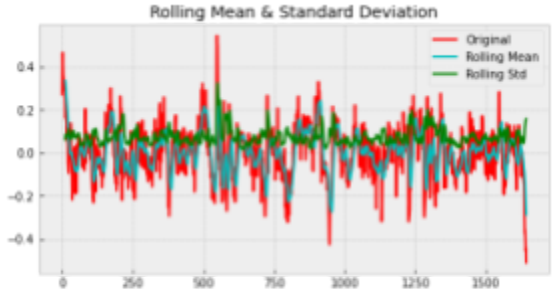


Exhibit 7



Results of Dickey-Fuller Test:

Test Statistic	-1.082606e+01
p-value	1.755758e-19
#Lags Used	2.000000e+00
Number of Observations Used	1.640000e+03
Critical Value (1%)	-3.434344e+00
Critical Value (5%)	-2.863304e+00
Critical Value (10%)	-2.567709e+00

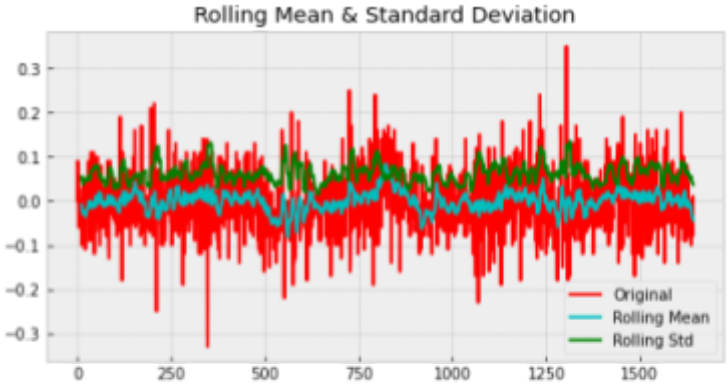
dtype: float64

Exhibit 8

1. https://www.kaggle.com/c/acea-water-prediction/overview
2. https://towardsdatascience.com/stationarity-in-time-series-analysis-90c94f27322#:~:text=In%20t%20he%20most%20intuitive,not%20itself%20change%20over%20time
3. https://www.statsmodels.org/stable/generated/statsmodels.tsa.seasonal.seasonal_decompose.html
4. https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/
5. https://research.fb.com/blog/2017/02/prophet-forecasting-at-scale/

Differencing

Because we didn't achieve a stationary dataset using decomposition, differencing was the next attempt. In differencing, we take the difference of the observation at a particular record with that at the previous record based on x-number of days. To start, I tried first-order differencing, a single-day increment, to P25. With the application of differencing, it was necessary to apply the ADF test to determine if we achieved stationarity. The results of the test, seen in exhibit 9, show the test statistic is less than the critical values. I failed to reject the null hypothesis. Therefore, the series is now stationary.



Results of Dickey-Fuller Test:

Test Statistic	-4.787604
p-value	0.000057
#Lags Used	23.000000
Number of Observations Used	1619.000000
Critical Value (1%)	-3.434396
Critical Value (5%)	-2.863327
Critical Value (10%)	-2.567721

dtype: float64

Exhibit 9

1. https://www.kaggle.com/c/acea-water-prediction/overview
2. https://towardsdatascience.com/stationarity-in-time-series-analysis-90c94f27322#:~:text=In%20t%20he%20most%20intuitive,not%20itself%20change%20over%20time
3. https://www.statsmodels.org/stable/generated/statsmodels.tsa.seasonal.seasonal_decompose.html
4. https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/
5. https://research.fb.com/blog/2017/02/prophet-forecasting-at-scale/

Baseline Model - ARIMA(p, d, q)

The ARIMA model has 3 components: AR (p), I (d), and MA (q). ARIMA stands for AutoRegressive Integrated Moving Average. The definition of each component is:

- AR: *Autoregression*. A model that uses the dependent relationship between an observation and some number of lagged observations[4]
- I: *Integrated*. The use of differencing of raw observations (e.g. subtracting an observation from an observation at the previous time step) to make the time series stationary[4]
- MA: *Moving Average*. A model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations[4]

I already have the I component with d=1 day. I needed to determine isR (p) and MA (q) for my base model. By looking at autocorrelation and partial autocorrelation plots, ACF and PACF respectively, on my differenced series, I was able to identify AR and MR.

- q - The lag value where the ACF chart crosses the upper confidence interval for the first time. As seen in exhibit 10, it is around q=1

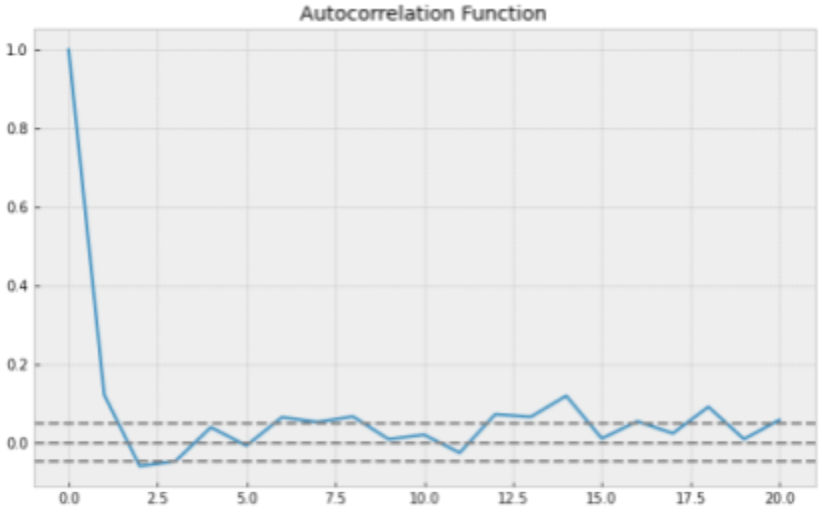


Exhibit 10

1. https://www.kaggle.com/c/acea-water-prediction/overview
2. https://towardsdatascience.com/stationarity-in-time-series-analysis-90c94f27322#:~:text=In%20t%20he%20most%20intuitive,not%20itself%20change%20over%20time
3. https://www.statsmodels.org/stable/generated/statsmodels.tsa.seasonal.seasonal_decompose.html
4. https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/
5. https://research.fb.com/blog/2017/02/prophet-forecasting-at-scale/

- p - The lag value where the PACF chart crosses the upper confidence interval for the first time. As seen in exhibit 11, it is around p=1

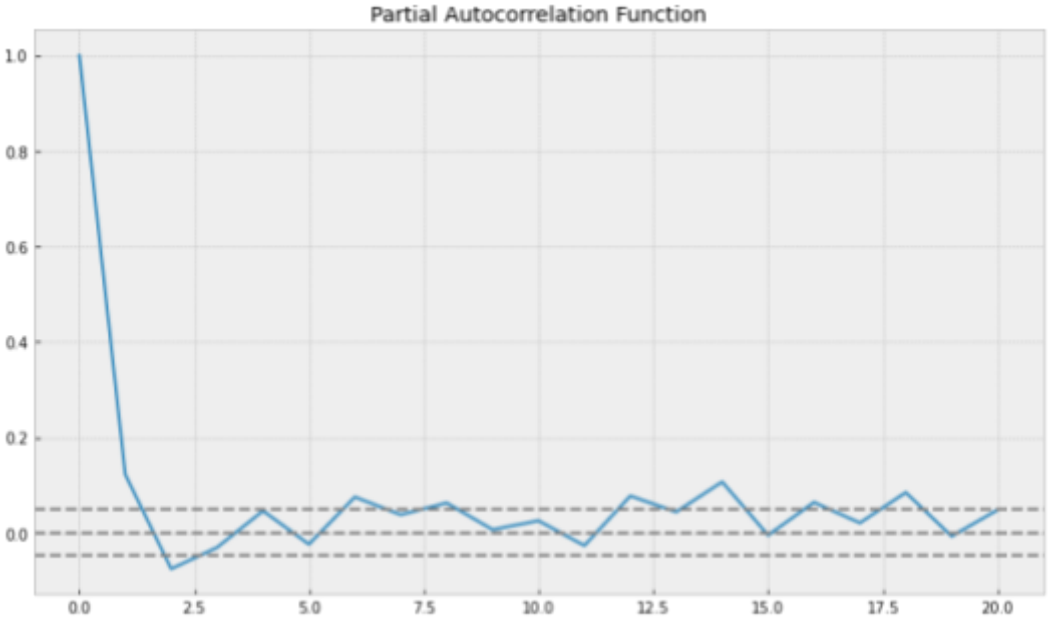


Exhibit 11

With my ARIMA constants determined, I set up my train/valid datasets. These datasets were used for my baseline model (ARIMA), second model (Auto-ARIMA), and third model (Prophet). Of the 1,643 records in my aquifer dataset, I set aside the last 90 days for validation. In doing so, I kept 1,553 days for model training. For all 3 models, I used a combination of actual vs. predicted, mean absolute percentage error (MAPE), and a distribution plot to determine the accuracy of each model.

ARIMA Model

The ARIMA model has ~0.4% MAPE which implies about 99.6% accuracy in predicting the next 90 observations. When looking at the actual vs. expected, the graph in exhibit 12 supports this result. While this is an excellent outcome, we can see the forecasted values in a straight line in blue while the actual values are more varied in yellow.

1. https://www.kaggle.com/c/acea-water-prediction/overview
2. https://towardsdatascience.com/stationarity-in-time-series-analysis-90c94f27322#:~:text=In%20t%20he%20most%20intuitive,not%20itself%20change%20over%20time
3. https://www.statsmodels.org/stable/generated/statsmodels.tsa.seasonal.seasonal_decompose.html
4. https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/
5. https://research.fb.com/blog/2017/02/prophet-forecasting-at-scale/

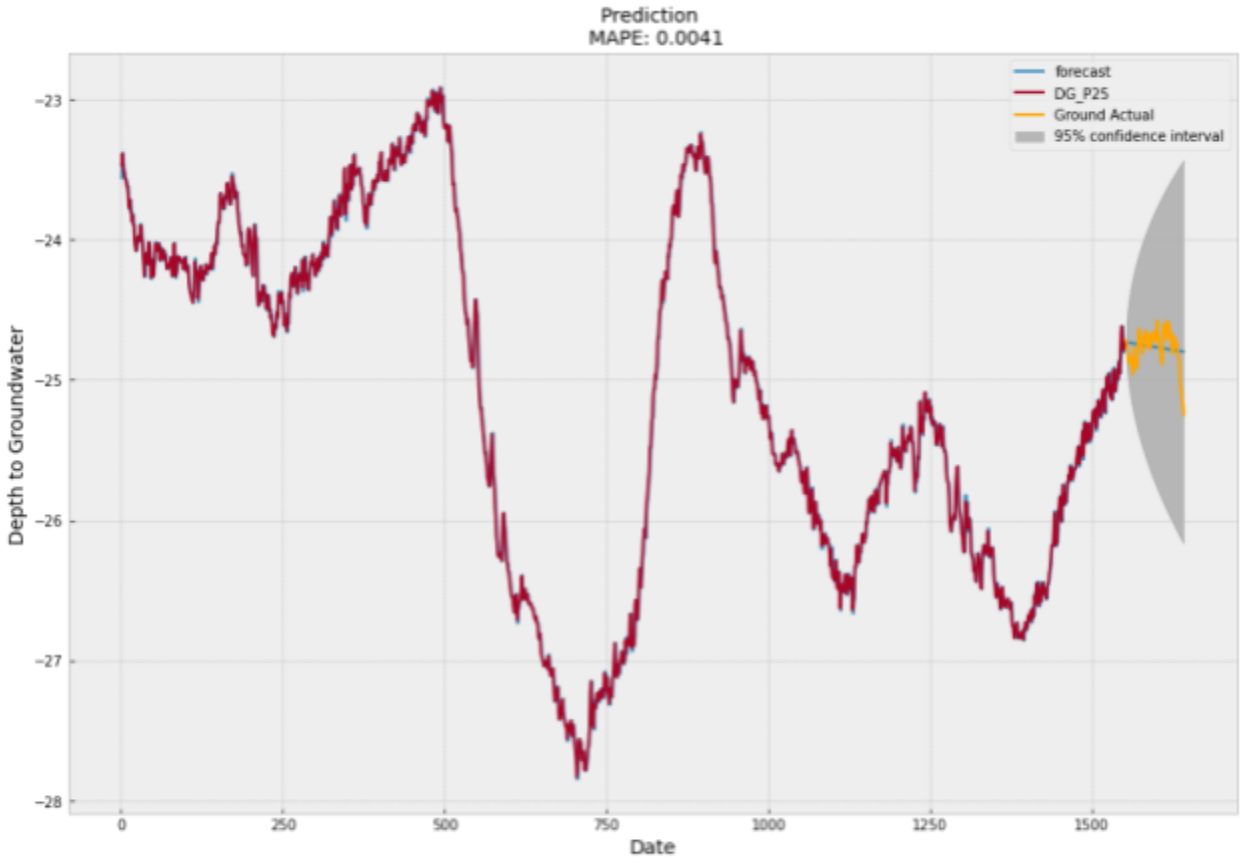


Exhibit 12

Auto-ARIMA Model

Where I had to provide the p, d, and q values for ARIMA, auto-ARIMA itself will generate the optimal p, d, and q values which are most suited for better forecasting. The MAPE of auto-ARIMA is higher at ~1.54% which indicates a slightly lower accuracy. When comparing the predicted values to actual, we can see the predicted values arch over the actual values in contrast to ARIMA where it's more level.

1. https://www.kaggle.com/c/acea-water-prediction/overview
2. https://towardsdatascience.com/stationarity-in-time-series-analysis-90c94f27322#:~:text=In%20t%20he%20most%20intuitive,not%20itself%20change%20over%20time
3. https://www.statsmodels.org/stable/generated/statsmodels.tsa.seasonal.seasonal_decompose.html
4. https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/
5. https://research.fb.com/blog/2017/02/prophet-forecasting-at-scale/

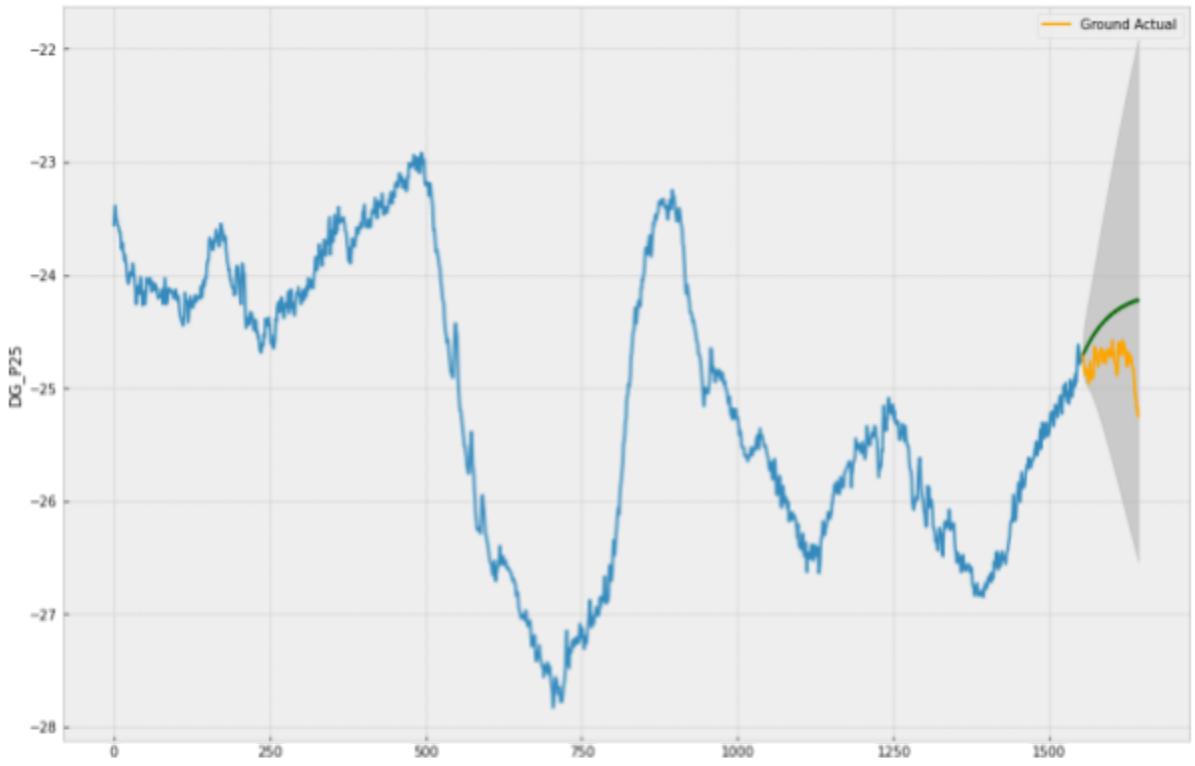


Exhibit 13

Prophet Model

At its core, the Prophet procedure is an additive regression model with four main components:

- A piecewise linear or logistic growth curve trend. Prophet automatically detects changes in trends by selecting changepoints from the data.
- A yearly seasonal component modeled using Fourier series.
- A weekly seasonal component using dummy variables.
- A user-provided list of important holidays.[5]

The prophet model generated a MAPE of ~0.43% which is only a touch worse than ARIMA at 0.4%. However, in comparing the actual vs predicted, we see, in exhibit 14, the predicted values curve are similar to the actual values, just slightly higher.

1. https://www.kaggle.com/c/acea-water-prediction/overview
2. https://towardsdatascience.com/stationarity-in-time-series-analysis-90c94f27322#:~:text=In%20t%20he%20most%20intuitive,not%20itself%20change%20over%20time
3. https://www.statsmodels.org/stable/generated/statsmodels.tsa.seasonal.seasonal_decompose.html
4. https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/
5. https://research.fb.com/blog/2017/02/prophet-forecasting-at-scale/

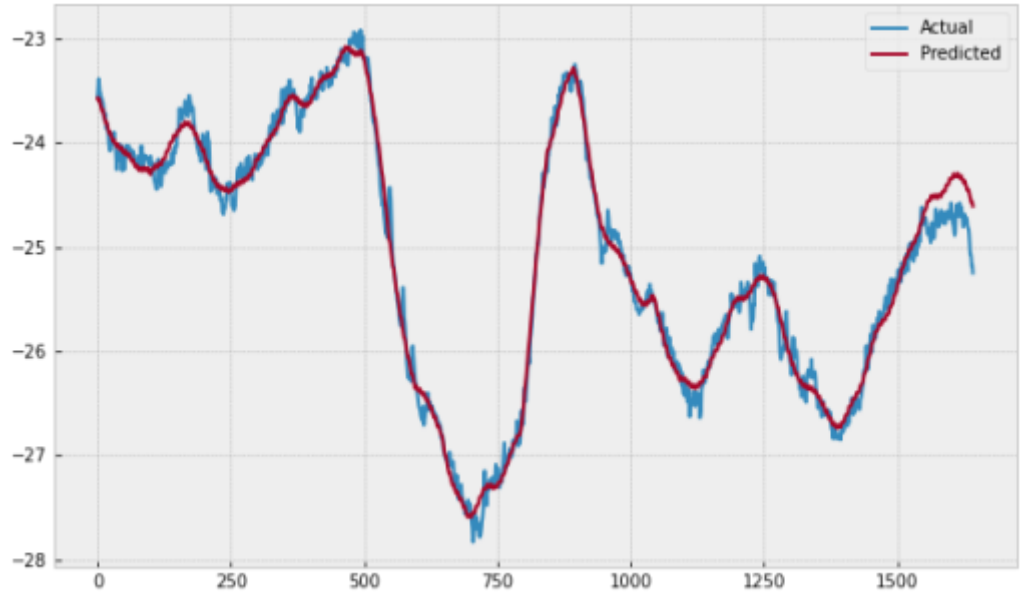


Exhibit 14

Comparing All 3

For a more accurate comparison, I graphed all 3 models' predicted values against actual on the same graph (Exhibit 15).

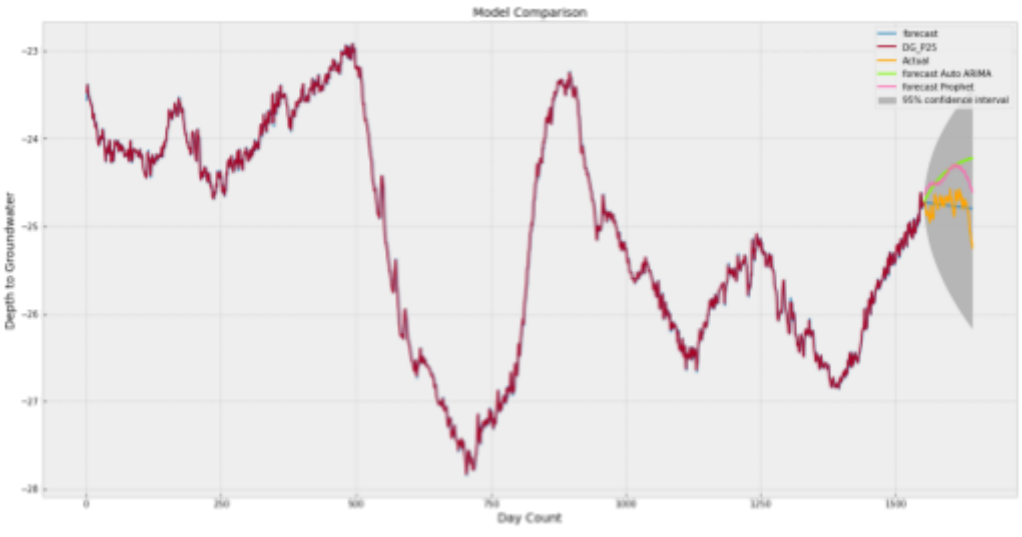


Exhibit 15

1. https://www.kaggle.com/c/acea-water-prediction/overview
2. https://towardsdatascience.com/stationarity-in-time-series-analysis-90c94f27322#:~:text=In%20t%20he%20most%20intuitive,not%20itself%20change%20over%20time
3. https://www.statsmodels.org/stable/generated/statsmodels.tsa.seasonal.seasonal_decompose.html
4. https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/
5. https://research.fb.com/blog/2017/02/prophet-forecasting-at-scale/

From the exhibit, we can see that ARIMA follows the actual results the closest. For confirmation, we can look at the distribution of all 3 models' residuals in exhibit 16.

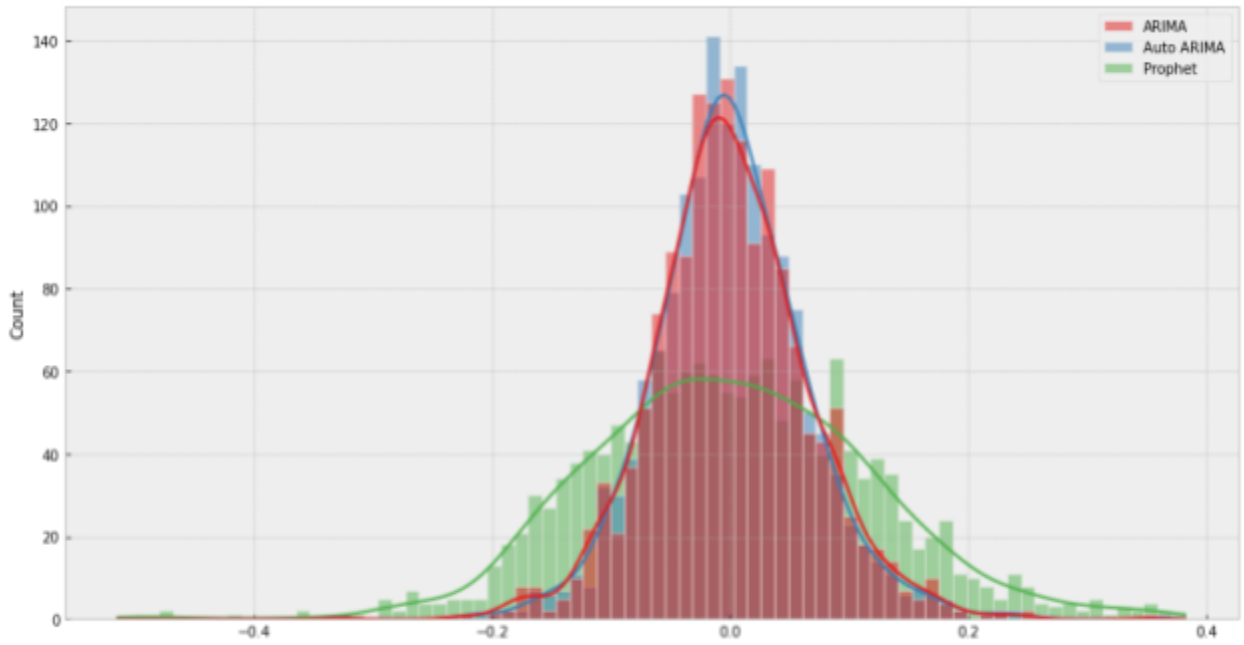


Exhibit 16

The rule of thumb for residual distribution is tall and slim are better than short and squat. Tall and slim indicates the residuals are clustered around 0.0. Short and squat indicate the residuals are more spread out.

We can see auto-ARIMA is taller and slightly slimmer than the ARIMA distribution. However, the MAPE of auto-ARIMA was almost 1 full point more than ARIMA, 1.5% and 0.4% respectively.

1. https://www.kaggle.com/c/acea-water-prediction/overview
2. https://towardsdatascience.com/stationarity-in-time-series-analysis-90c94f27322#:~:text=In%20t%20he%20most%20intuitive,not%20itself%20change%20over%20time
3. https://www.statsmodels.org/stable/generated/statsmodels.tsa.seasonal.seasonal_decompose.html
4. https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/
5. https://research.fb.com/blog/2017/02/prophet-forecasting-at-scale/

Wrap-up

Model Summary

Model	RMSE	MAE	MAPE	Lower/Upper Bound (meters)
ARIMA	0.133285	0.101149	0.004071	-0.04047/0.03853
Auto-ARIMA	0.429095	0.383468	0.015447	-0.039098/0.037956
Prophet	0.141993	0.107153	0.004275	-0.26365/0.29985

To compare the 3 models, I put together the model summary to compare Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and the lower/upper bounds (~2 standard deviations) from the mean error. We can see RMSE, MAE, and MAPE are all higher for auto-AIMA except the error bounds. The higher RSME indicates the residuals are more spread out than ARIMA or Prophet. Similarly, MAPE is higher which indicates a less accurate model than the other two.

Comparing ARIMA and Prophet, ARIMA performs better across RMSE, MAE, MAPE, and has a smaller distribution between the two error bounds. Of the three models, I chose ARIMA.

Recommendations for the Client

I recommend using the ARIMA model for estimating groundwater depth. Of the 3 models, the ARIMA model has the smallest distribution of residual errors and the highest accuracy. To compensate for model errors, the client can use the lower and upper error bounds to adjust where they feel it's needed.

1. https://www.kaggle.com/c/acea-water-prediction/overview
2. https://towardsdatascience.com/stationarity-in-time-series-analysis-90c94f27322#:~:text=In%20t%20he%20most%20intuitive,not%20itself%20change%20over%20time
3. https://www.statsmodels.org/stable/generated/statsmodels.tsa.seasonal.seasonal_decompose.html
4. https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/
5. https://research.fb.com/blog/2017/02/prophet-forecasting-at-scale/

Future Work

- Multivariate ARIMA

  An interesting avenue to pursue further would be comparing the univariate model to a multivariate version. I would include ‘rainfall', 'temperature', 'volume', and ‘hydrometry’ as additional variables. It is possible that the addition of one or many of these variables can lead to a more accurate model

- Additional Model: Univariate LSTM

  Another model to consider is a long short-term memory (LSTM) recurrent neural network (RNN). This type of RNN can learn and remember over long sequences. Essentially, the memory-capacity of LSTM is extremely useful for time-series or sequential data.

1. https://www.kaggle.com/c/acea-water-prediction/overview
2. https://towardsdatascience.com/stationarity-in-time-series-analysis-90c94f27322#:~:text=In%20t%20he%20most%20intuitive,not%20itself%20change%20over%20time
3. https://www.statsmodels.org/stable/generated/statsmodels.tsa.seasonal.seasonal_decompose.html
4. https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/
5. https://research.fb.com/blog/2017/02/prophet-forecasting-at-scale/