How did the pandemic impact Water Usage in Long Beach, CA?

By: Ensieh Bahrami

Data Science Intensive Capstone Project

In March 2020, a Safer at Home Order was declared in Long Beach in response to the COVID-19 pandemic. This led to the closing of many businesses and a large portion of the population staying at home. Long Beach Water seeks to understand the impacts of the COVID-19 pandemic on water use.

1. Data Summary

The water usage data for approximately 80,000 water customers for three consecutive years from July 2017 to July 2020 are available. Records in the data set correspond with water usage for individual customer water accounts that is recorded approximately each month. Each sheet in the Excel file corresponds with 12 "monthly" water usage records of data for each customer account. Data has been acquired from Long Beach Water Department.

1.1. Data Column Definitions

Columns in the data set for three years include:

- **LOC_ID** – Unique customer account number
- **WTR_RATE** – Identifies the types of customer accounts

  W-RSFD – Residential Single-Family Dwelling

  W-RDUPLX – Residential Duplex

  W-RMF – Residential Multi-family

  W-COM – Commercial

  W-IND – Industrial

  W-IRR – Irrigation

- **WTR_DWEL_UNIT** – Number of dwelling units associated with account. (W-COM "commercial" accounts with *non-zero dwelling units* are mixed-use properties that have both commercial and residential units using water from the same water account.)
- **WTR_USE1 - WTR_USE12** – How much water use was recorded since the last meter read in units of hundred cubic feet (hcf). 1 hcf = 748 gallons.
- **WTR_READ_DT1 - WTR_READ_DT12** – Date that water meter was read to record water used since previous read date. (E.g. if WTR_READ_DT1 is 7/19/2018, and WTR_READ_DT2 is 6/19/2018, and that means the value in WTR_USE1 for the record is reflective of how much water was used from 6/20/2018 through 7/19/2020.)
- **WTR_READ_DAYS1 - WTR_READ_DAYS12** – Number of days that have elapsed between the associated read date and the previous read date.

2. Data Cleaning

Rate

There were a few accounts with null value for Rate, the majority of their attributes for all 3 years had null values so they seemed not to be informative. Those accounts were dropped.

Dwelling Units

The number of dwelling units in each Rate (customer group) was checked. Irrigation and Industrial customers had none and zero dwelling units respectively, which are replaced with value of one. So, now all industrial and irrigation customers have one dwelling unit.

The number of dwelling units for Single Family and duplex customers should be one and two respectively. It seemed that there were a few errors happened during entering data, that were corrected.

Commercial customers had zero, one or higher number of dwelling units. Significant number of customers were pure commercial. Since, the pandemic caused shout

down of businesses for some months, it was worth to separate the pure commercial and explore more. So, Rate of Commercial was kept for those with zero dwelling and Rate of those with one or more dwelling units was changed to 'COM&RES'.

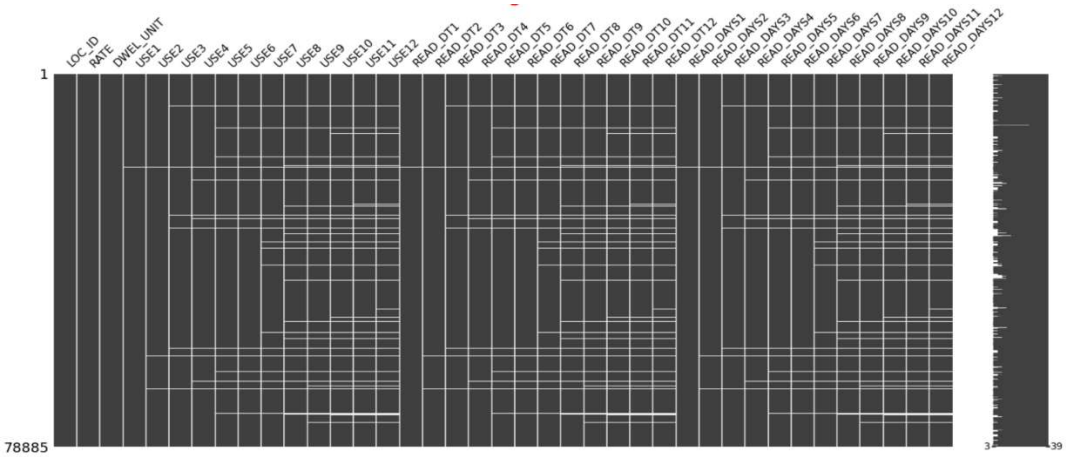
Rate or type of customer accounts includes:

- Single-Family
- Duplex
- Multi-family
- Pure Commercial
- Commercial and Residential
- Industrial
- Irrigation

USE, READ DAYS and REAT_DT

There were plenty missing values in Read Date columns particularly in first year (2017), the main reason was that reading the water meter records had not been performed monthly and regularly. But the promising data was that amount of usage recorded by water meter between two consecutive Read Dates. With the given use for each period (mostly longer than one month), and the number of Days elapsed between two consecutive Read Dates, it was possible to calculate monthly usage for all 3 years.

Figure 1. Matrix of Missing Values for Year 2017



I used wide_to_long method from pandas library to create a long data frame. Each row of the long data frame had a unique LOC_ID and RATE. It also had a group of columns including USE, READ_DT and READ_DAYS which contained all values of related columns in main data sets. Remaining column (DWEL_UNIT) of data sets was intact in long data frame.

Figure 2. New Data Frame

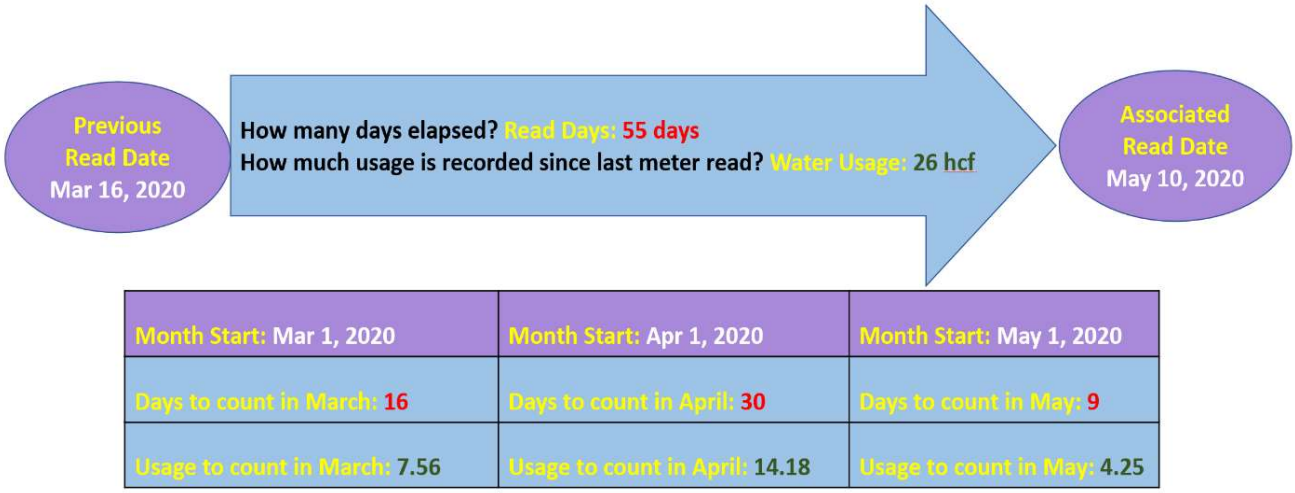
	LOC_ID	RATE	period	DWEL_UNIT	USE	READ_DT	READ_DAYS
0	1000000000	RSFD	1	1.0	22.0	2018-07-11	30.0
1	1000000000	RSFD	2	1.0	22.0	2018-06-11	32.0
2	1000000000	RSFD	3	1.0	15.0	2018-05-10	29.0
3	1000000000	RSFD	4	1.0	9.0	2018-04-11	29.0
4	1000000000	RSFD	5	1.0	10.0	2018-03-13	32.0

3. Feature Engineering

In the long data frame, I did some feature engineering to calculate monthly usage for each Rate. I took these steps:

1) Calculate Previous Read Date from subtracting Read Days from Read Date
2) For each row, find related month(s) between associated Read Date and its Previous Read Date
3) Set related month(s) as the index of each row
4) Find the Days in each of the related month(s)
5) Calculate days to count for each month
6) Calculate usage to count for each month
7) Group by Rate and month to count total number of dwelling and total usage

Figure 3. Find Days to count and usage to count in each month



And finally, the cleaned data set demonstrate for each Rate and month how much water used by how many dwellings.

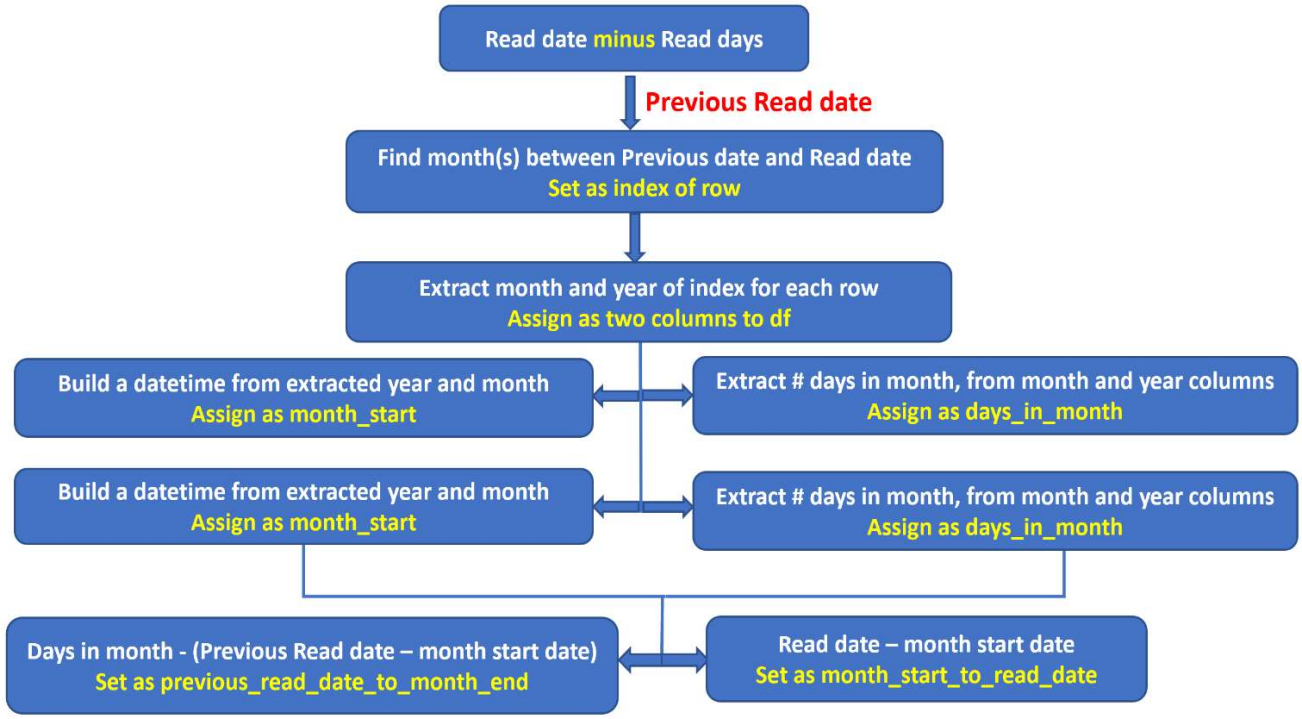
Figure 4. Cleaned data features

	RATE	month_start	DWEL_UNIT	usage
0	COM	2016-04-01	2	6.1
1	COM	2016-05-01	1	0.5
2	COM	2016-11-01	3	5.7
3	COM	2016-12-01	4	6.7
4	COM	2017-01-01	5	8.4

3.1. Algorithm

The algorithm that performed the steps mentioned above, created some new features to help calculate the monthly usage for each row:

Figure 5. Algorithm of feature engineering



Then, I set some conditional statements to calculate 'days to count' for each month using new features:

IF	Then 'days_to_count_in_period' will be
previous_read_date_to_month_end <= days in month	previous_read_date_to_month_end
previous_read_date_to_month_end > days in month & read_date_to_month_start >= days in month	Days in month
previous_read_date_to_month_end > days in month & read_date_to_month_start < days in month	read_date_to_month_start

Example 1

Days in March = 31			
Month Start Mar 1,2020	Previous Read Date March 16, 2020	Month End	Read Date May 10, 2020
	previous_read_date_to_month_end 16 days		
read_date_to_month_start 70			
previous_read_date_to_month_end <= days in month (16 < = 31)			

Month Start	days_to_count_in_period
2020-03-01	16

Days in April = 30			
Month Start April 1, 2020	Previous Read Date March 16, 2020	Month End	Read Date May 10, 2020
	previous_read_date_to_month_end 30-(-16) = 46 days		
read_date_to_month_start 39			
previous_read_date_to_month_end > days in month (46> 30) & read_date_to_month_start >= days in month (39 >= 30)			

Month Start	days_to_count_in_period
2020-04-01	30

Days in May = 31			
Month Start May 1, 2020	Previous Read Date March 16, 2020	Month End	Read Date May 10, 2020
	previous_read_date_to_month_end 31-(-46) = 77 days		
Read_date_to_month_start 9 days			
previous_read_date_to_month_end < days in month (77 > 31) & read_date_to_month_start < days in month (9 < 30)			

Month Start	days_to_count_in_period
2020-05-01	9

Example2

Days in March = 31			
Month Start Mar 1,2020	Previous Read Date March 16, 2020	Month End	Read Date May 1, 2020
	previous_read_date_to_month_end 16 days		
read_date_to_month_start 56			
previous_read_date_to_month_end <= days in month then (16 < = 56)			

Month Start	days_to_count_in_period
2020-03-01	16

Days in April = 30			
Month Start April 1, 2020	Previous Read Date March 16, 2020	Month End	Read Date May 1, 2020
	previous_read_date_to_month_end 30-(-16) = 46 days		
read_date_to_month_start 30			
previous_read_date_to_month_end > days in month (46> 30) & read_date_to_month_start >= days in month (30 >= 30)			

Month Start	days_to_count_in_period
2020-04-01	30

Days in May = 31			
Month Start May 1, 2020	Previous Read Date March 16, 2020	Month End	Read Date May 1, 2020
	previous_read_date_to_month_end 31-(-46) = 77 days		
Read_date_to_month_start 0 days			
previous_read_date_to_month_end < days in month (77 > 31) & read_date_to_month_start < days in month (9 < 30)			

Month Start	days_to_count_in_period
2020-05-01	0

After calculating the monthly usage for each account, monthly usage for each customer group was obtained.

Based on the number of days in each month, average daily usage for each segmented was calculated. Exploring average daily usage of all customers and each segment were more effective than exploring the usage.

4. EDA

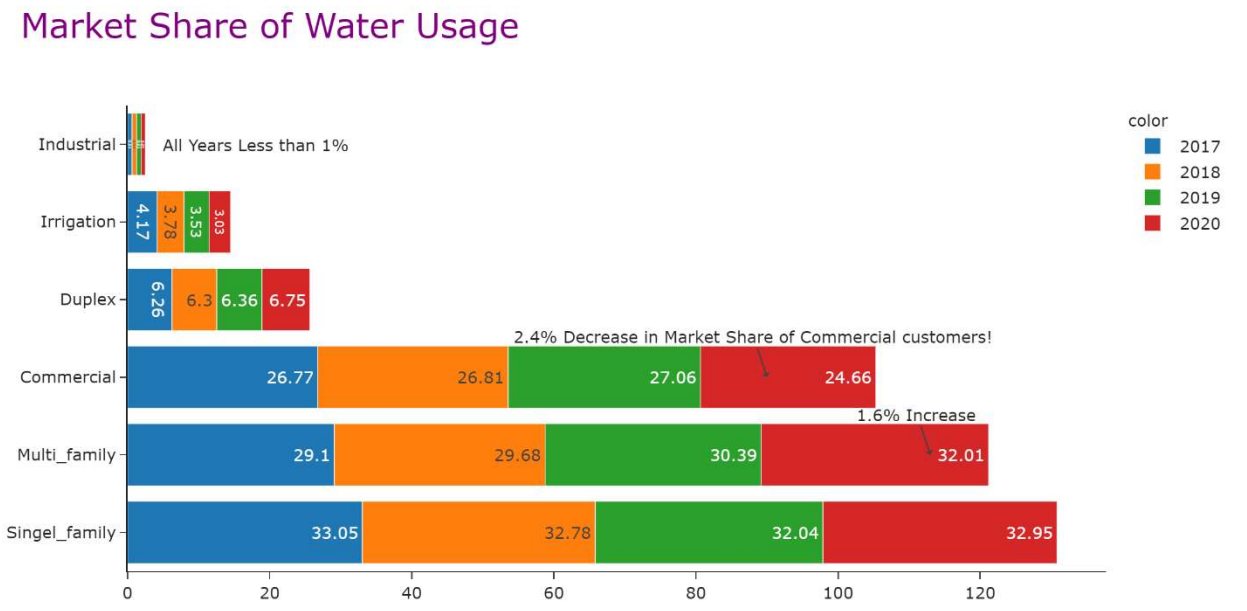
During data wrangling, I separated Pure Commercial from Commercial and Residential customers to correct the number of dwelling and know the percentage of pure commercial customers in commercial group.

High proportion of commercial customers are pure commercial. Now for exploration and modeling, I merge these subgroups as commercial group.

4.1. Market Share

Market share of Commercial segment decreased by 2.4% in the first 7 months of 2020. After shutdown, some businesses couldn't sustain, and they had to close. But there is 1% increase in Multifamily segment.

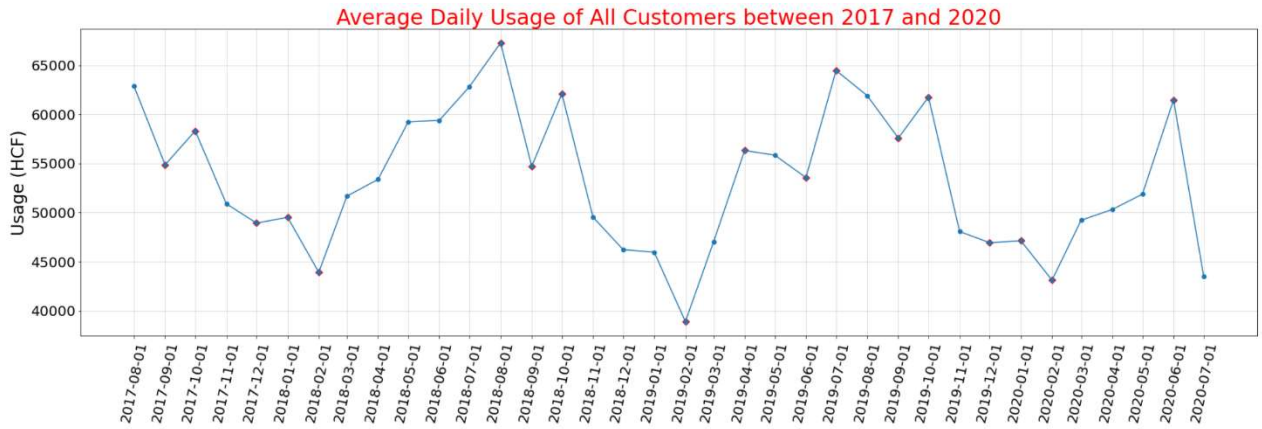
Figure 6. Market Share of water usage



4.2. Explore Changes in Usage

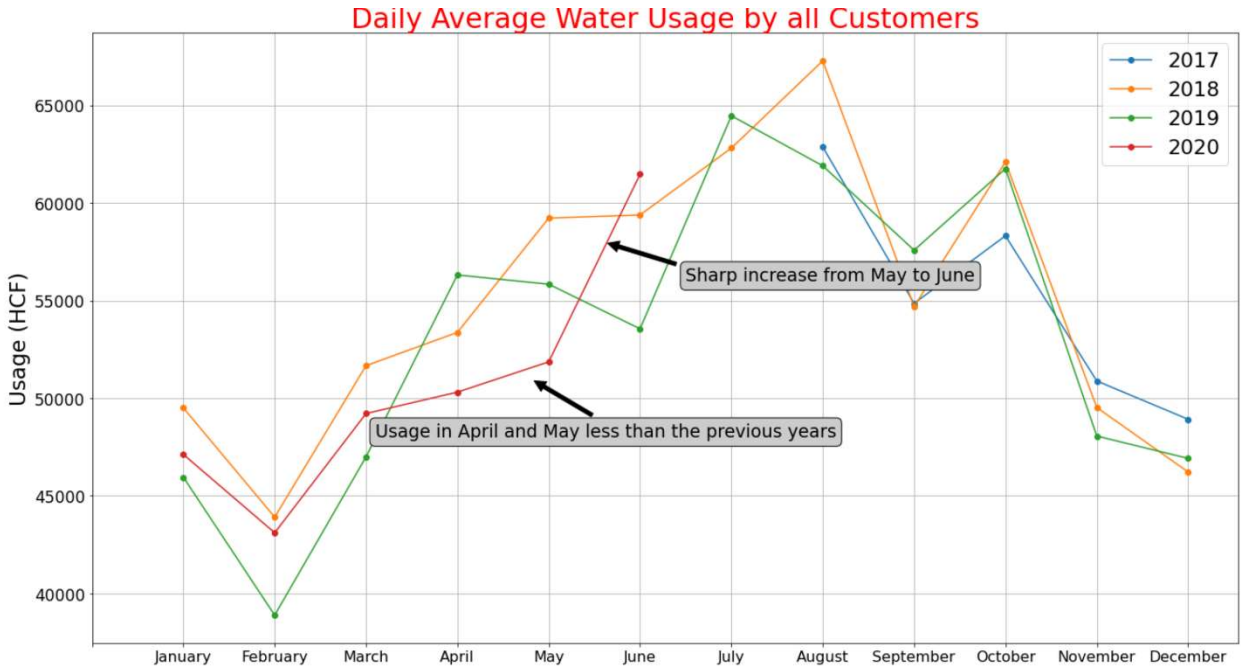
Average daily usage shows a trend and seasonality which has not been kept from March 2020.

Figure 7. Average daily usage by all groups of customers



Looking at daily average usage separated by year, the difference from March to June 2020 is more explicit. It shows usage drop in April and May 2020, compared with the same period in previous years and then a sharp increase from May to June 2020.

Figure 8. Average daily usage of all customers (2017-2020)

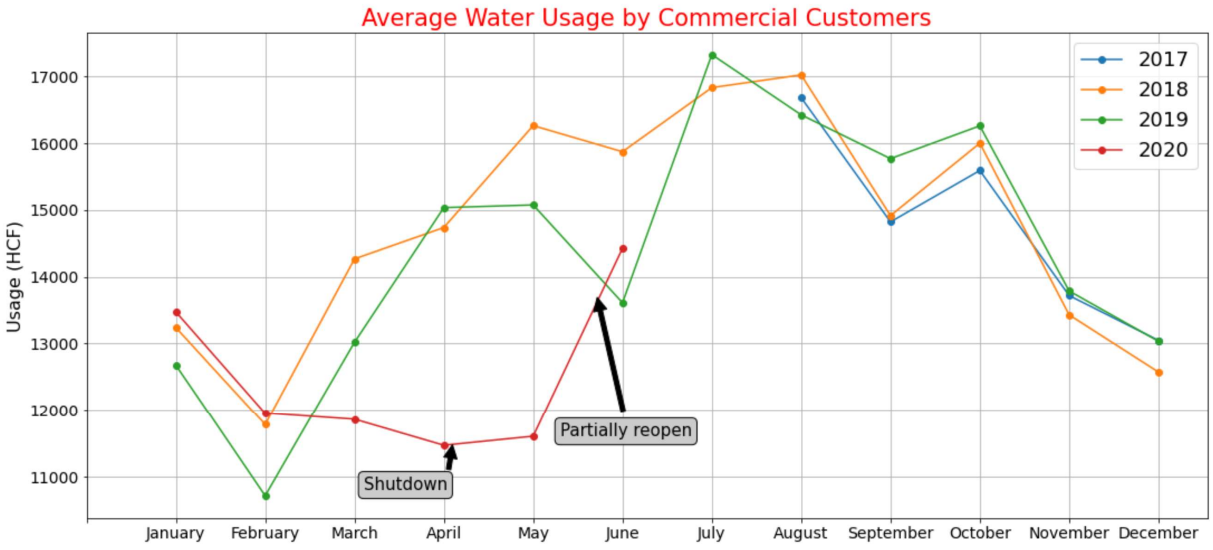


4.2.1. Changes in non-Residential Usage

During the shutdown in April and May, usage by commercial and Industrial customers significantly decreased.

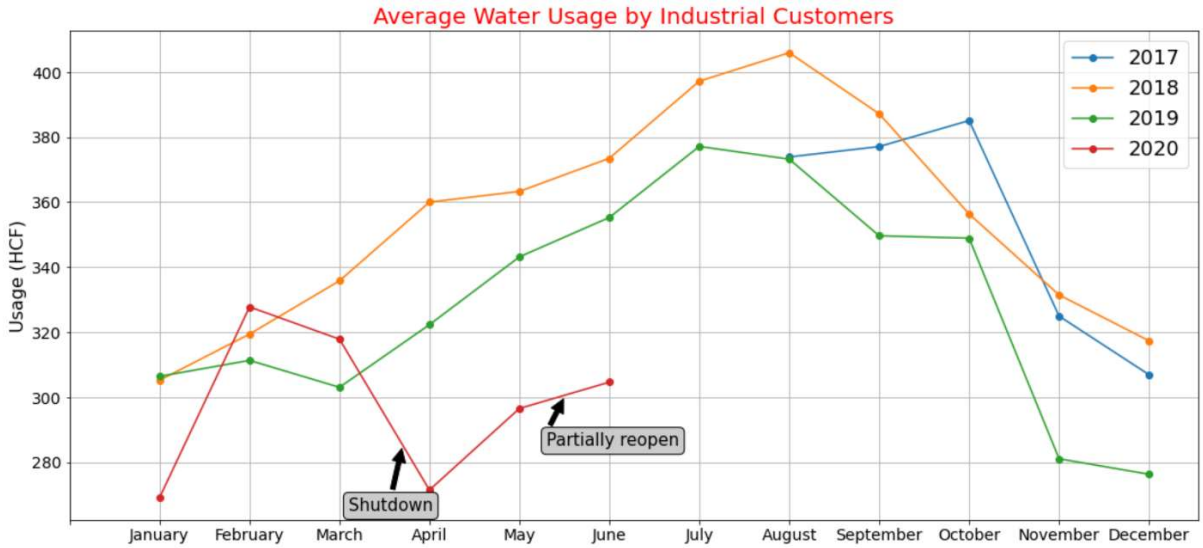
By partially reopening, commercial segment seemed to supply the delayed and aggregated demands so that usage jumped and passed the usage of June 2019.

Figure 9. Average daily water usage by Commercial customers



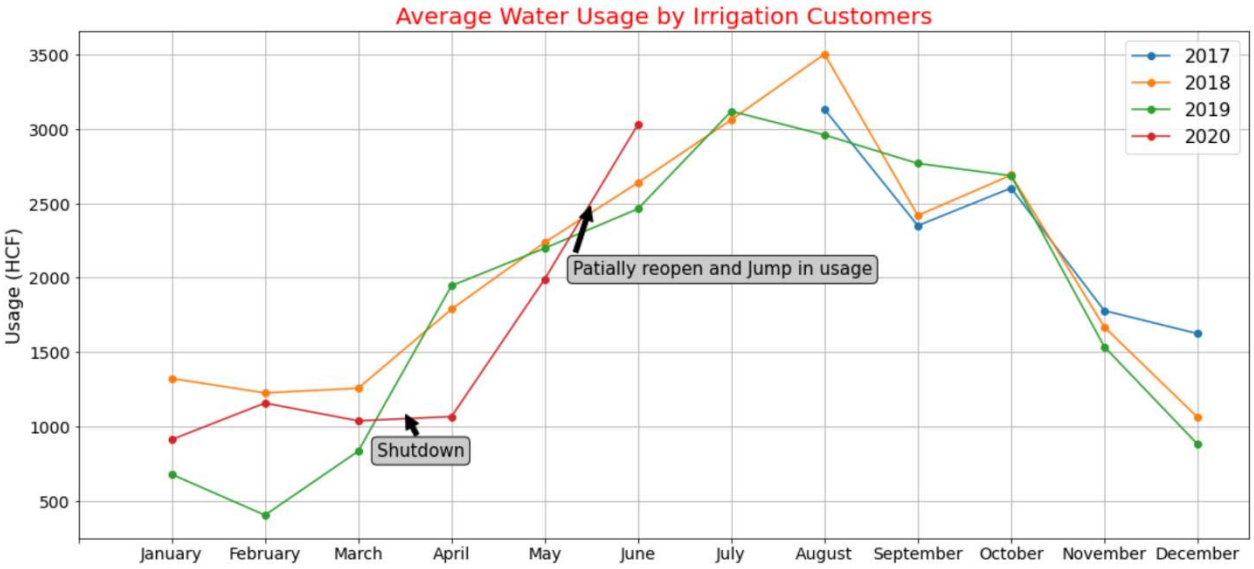
Partially reopening, slightly increased the usage in June 2020 but not even close to usage of similar period in previous years.

Figure 10. Average daily water usage by Industrial customers



In irrigation segment, the usage hadn't increase in April 2020 like previous years. But, usage in May and June had jumped and passed the usage of previous years.

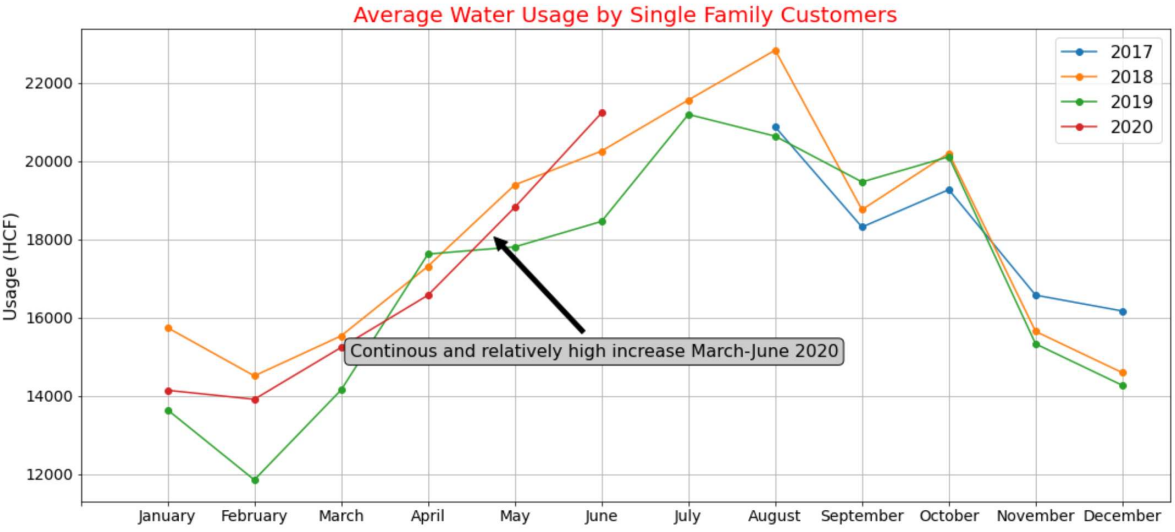
Figure 11. Average daily water usage by Irrigation customers



4.2.2. Changes in non-Residential Usage

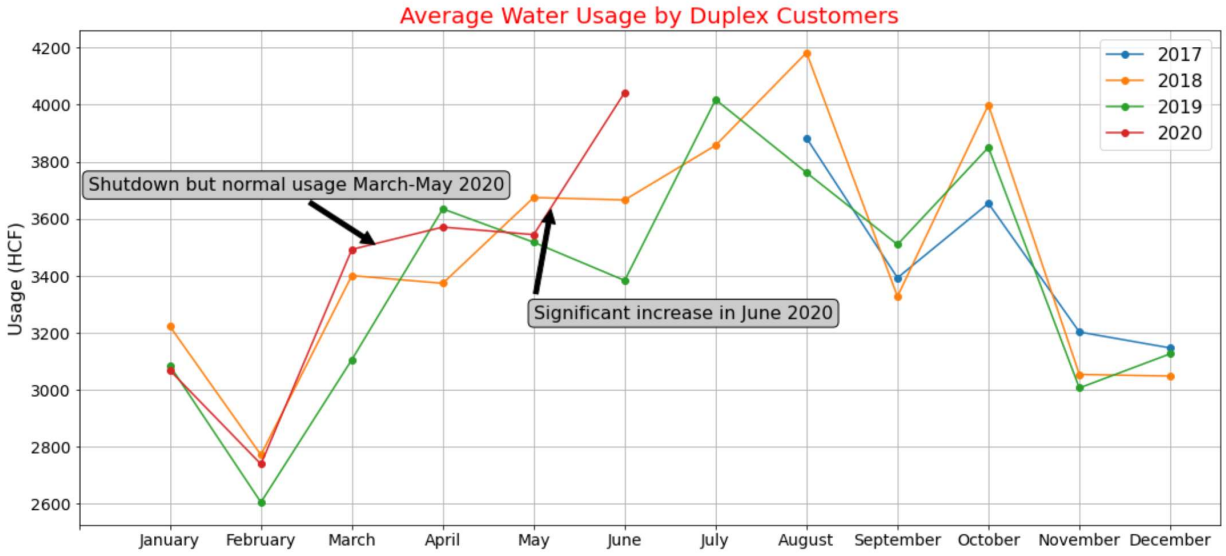
Single family usage kept its trend and seasonality.

Figure 12. Average daily water usage by Single Family customers



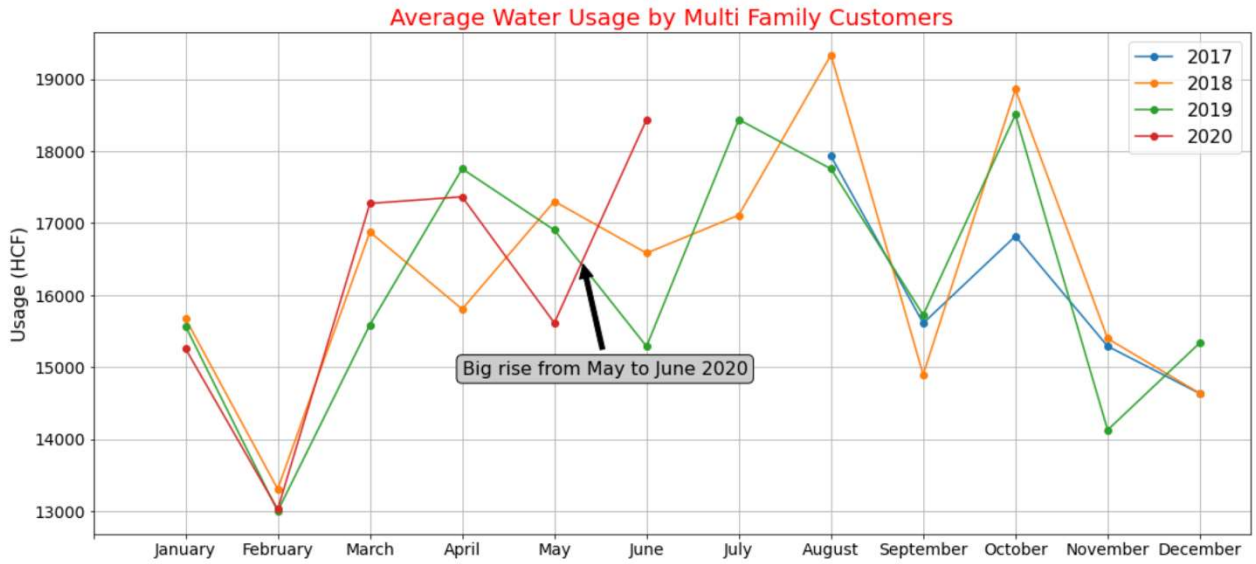
Duplex usage in April and May don't show significant difference, but it experienced significant rise in April 2020.

Figure 13. Average daily water usage by Duplex customers



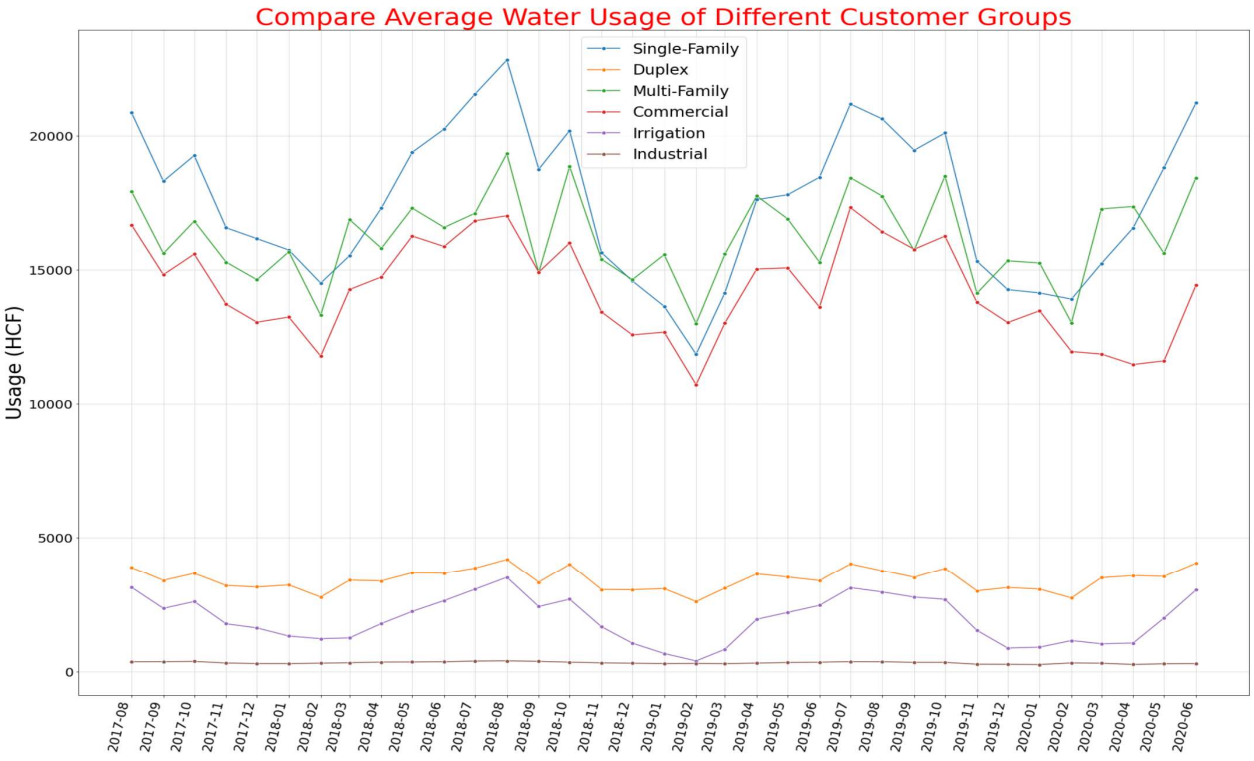
Multifamily usage dropped in May and then grew in June unlike the same period in the previous years.

Figure 14. Average daily water usage by Multi Family customers



It seems that Single Family and Multi Family usage tries to offset the decrease in Commercial usage from March 2020.

Figure 15. Compare average daily water usage by different customer groups



5. Time Series Analysis

In the cleaned data set, average daily usage for each month and each of RATE has calculated. Then, average daily usage by all customer groups were summed up for each month. Then amount of average usage was divided by number of dwelling units. The time series called ts_usage, demonstrated average daily usage per dwelling.

5.1 Analysis and Transforms

Each time series contains level and noise (random probabilistic process), but it might have seasonality and trend. The four components usually combine either additively or multiplicatively.

Level: When trend, seasonality and noise are removed from the time series, the remaining values are the level. They are the true values used for forecasting.

Trend: Trend is an increase or a decrease in the mean of daily usage over the time. It is a smooth, general, and long-term average of tendency.

Seasonality: When there is a specific pattern repeated in intervals regularly because of seasonal factors, we can say that there is a seasonality. In time series of usage, amount of usage in summer is much higher than winter.

White noise: if values are independent say that no correlation with other values, as well as having mean of zero and same variance, they have white noise.

Three types of time series modeling that I mostly focused on them are as follows:

- **Additive Model**

Additive model is a linear model of the 4 components, and its time series has a constant change over time.

Y(t) = Level + Noise + Trend + Seasonality

While trend is linear, and seasonality has constant frequency and amplitude.

- **Multiplicative Model**

Multiplicative model is not linear, its time series doesn't have a constant change over time.

Y(t) = Level * Noise * Trend * Seasonality

While trend is curve, seasonality is not constant.

- **Combination of an additive model and a multiplicate model**

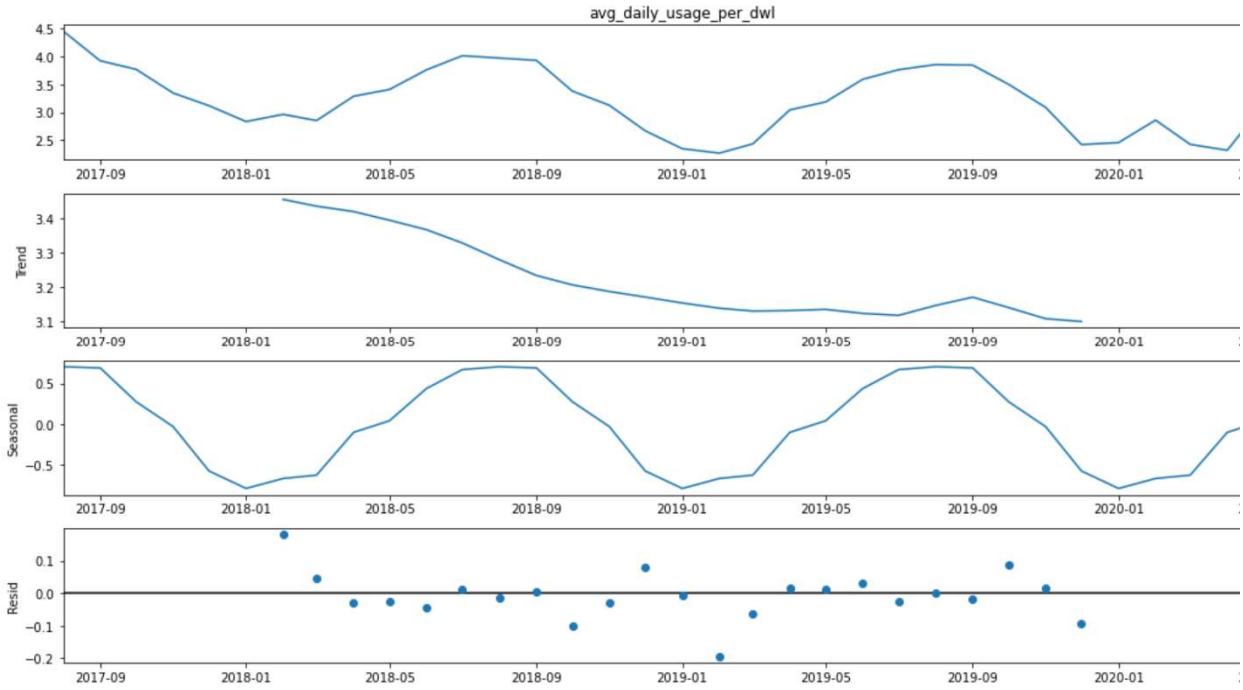
Most real-world time series are a combination of two models.

5.1.1 Time series decomposition

a) Decompose time series usage

Decomposition helped to identify each of 4 different parts of the time series and its behavior. As the below graphs show the usage time series had trend and yearly seasonality.

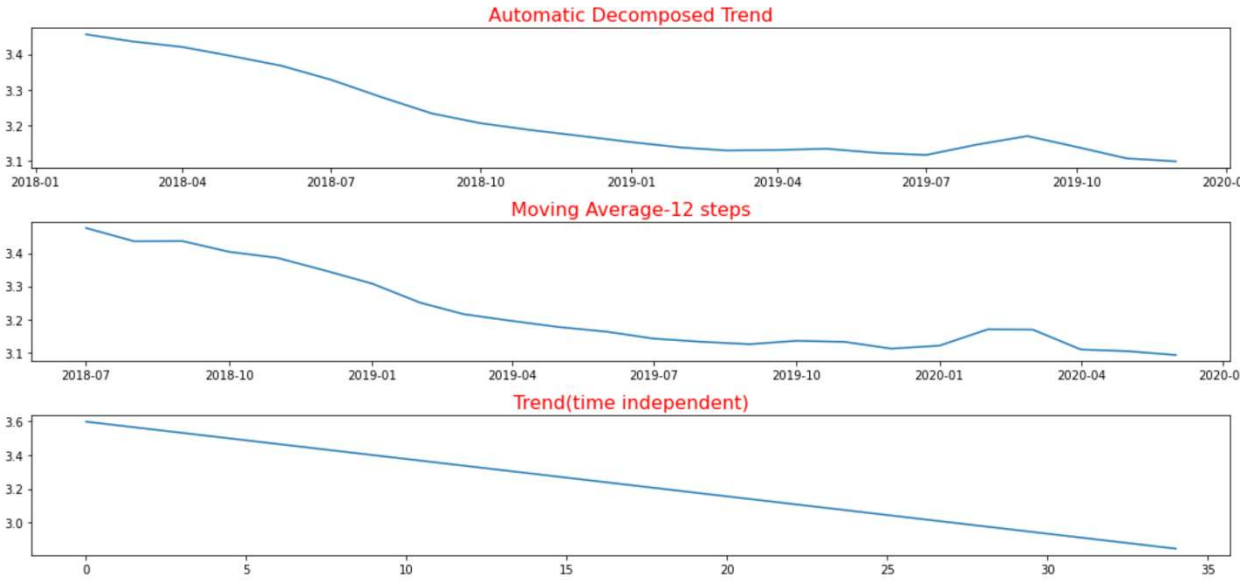
Figure 16. Time Series Decomposition



b) check trend

When I fitted linear regression on usage values (independent from time), the result of prediction is relatively close to the automatic decomposed trend. Time series usage seemed to have a trend which is dependent to time.

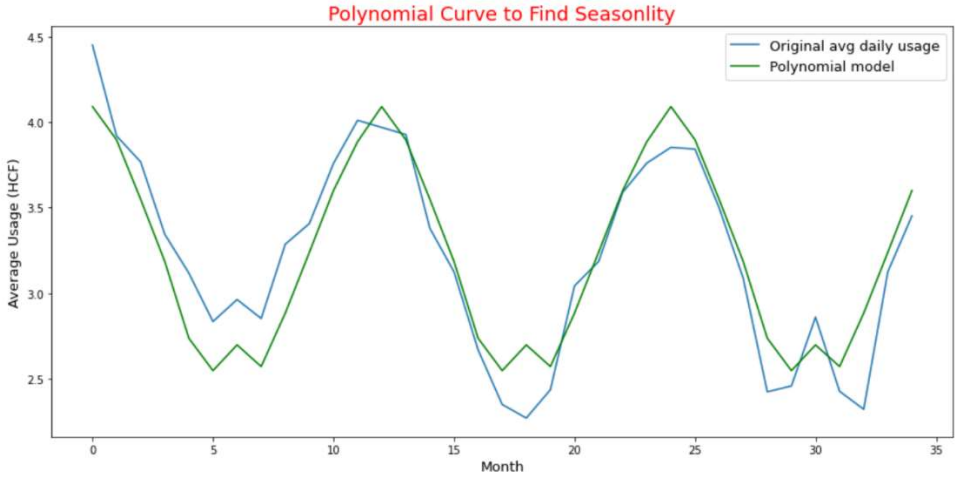
Figure 17. Automatic Decomposed Trend vs Time Independent Trend



c) Check the seasonality

In each year, usage has raised continuously from February to September, then from October to January usage has decreased. Fitting a polynomial curve helped to find a yearly seasonality in the time series.

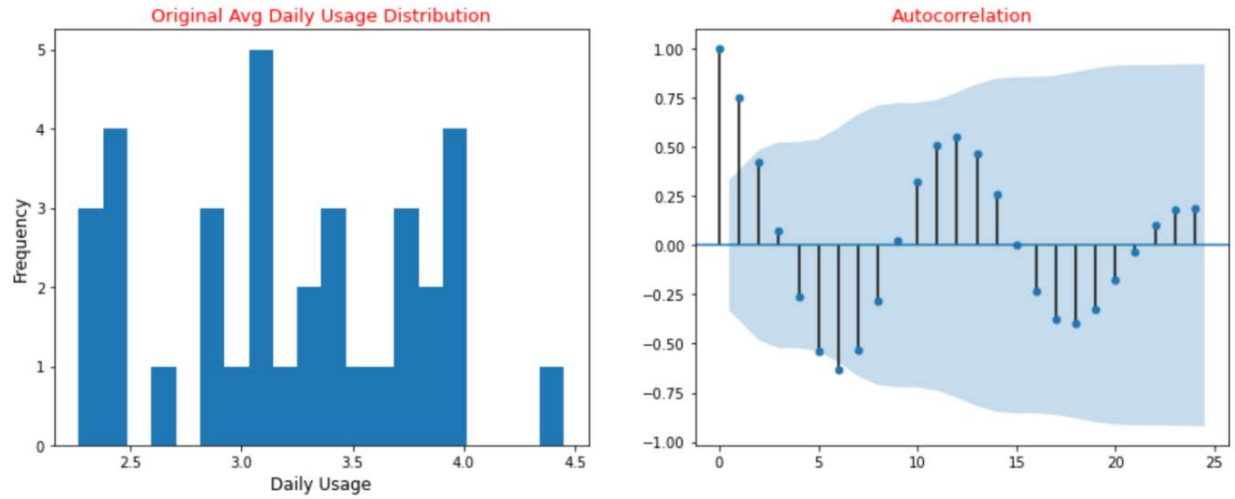
Figure 18. Automatic Decomposed Seasonality vs Time Independent Seasonality



d) Check the noise

Left plot shows that daily usage doesn't have a Gaussian distribution.
Autocorrelation measures correlation between a time series and itself, at different 'lags'(shift in time). Right plot demonstrates that daily usage values are not independent, they are correlated with values at different lags. So, time series of usage is not white noise.

Figure 19. Correlated Average Usage Values with not Normal Distribution



Time series is not white noise for three reasons:

1) Mean is not constant with time,
2) Variance is not constant with time, and
3) Autocorrelation at all lags are not zero. Usage values are correlated even with values in distant points.

So, the past will help me forecast the future.

5.1.2 Stationary

A time series is stationarity if it has constant mean and variance over time. Most models work only with stationary data. Not all time series are stationary, but we can transform them into stationary series in different ways.

Is the time series of daily usage stationary or random walk?

Checked stationary of time series using Augmented Dicky-Fuller test (ADF). ADF test is a type of statistical test called a unit root test. Unit root test determines how strongly a time series is defined by a trend. It uses an autoregressive model and optimizes an information criterion across multiple different lag values.

Null hypothesis: the time series follows a random walk. It has a unit root which means that it is non-stationary. It has some time dependent structure.

Alternative hypothesis: The time series is stationary. It does not have time-dependent structure.

How average daily usage were correlated?

Autocorrelation plot shows that usage values are correlated with distant points (lags) in time. Autocorrelation values are not quickly dopped over time, which means information is carried over time and the series should not be constant over time (showing a clear seasonality shape). These are sings of non-stationary time series.

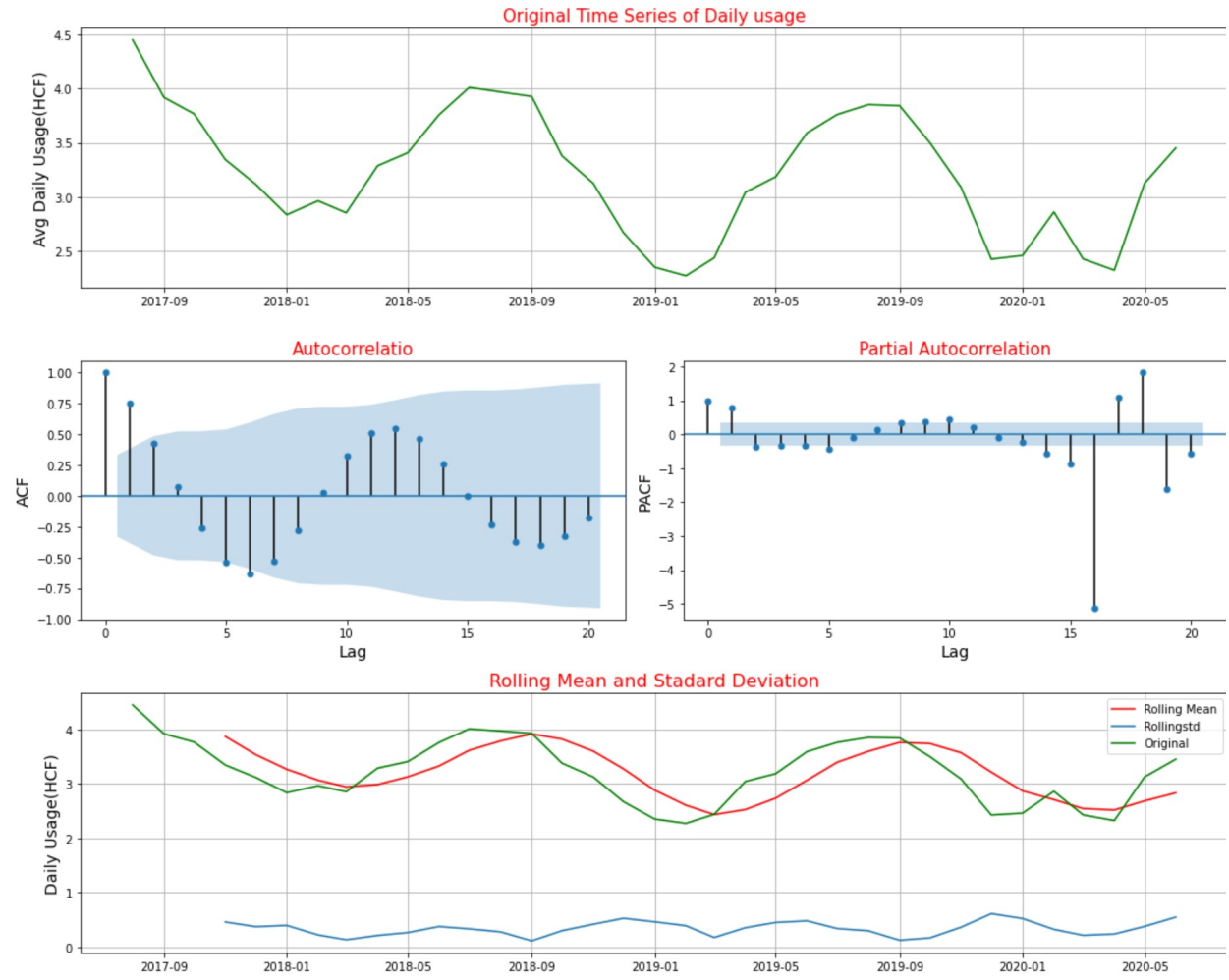
Partial Autocorrelation function measures the incremental benefit of adding another lag. Moving average with 6 steps didn't have constant mean and standard deviation over the time. So, it is not also stationary.

Autocorrelation function can be used for two purposes:

1) to detect non-randomness in data

2) to identify an appropriate time series model if the data are not random

Figure 20. Non-Stationary Time Series



Original values

Results of Dickey-Fuller Test:

test statistic	**-1.592271**
p-value	**0.487475**
# Lags Used	8.000000
# Observation Used	26.000000
Critical Value 1%	**-3.711212**
Critical Value 5%	**-2.981247**
Critical Value 10%	**-2.630095**

dtype: float64

Result of ADF test on original values showed (test statistic > Critical value at all significant levels) that **the original time series usage is not stationary.** It makes sense because I already explored that time series usage has trend and seasonality.

5.1.3 Make the daily usage time series stationary

Trend (change in mean over time) and seasonality (a specific pattern repeated over time) made the daily usage time series non-stationary.

In fact, trend and seasonality made the time series dependent structure. I need to remove trend and seasonality components to make it stationary.

Difference transform

Lag difference is subtracting the previous observation from the current observation. It helps to remove the seasonality when the lag is the period of seasonality.

difference(t) = observation(t) - observation(t-1)

Order difference is repeating the process of differencing more than once until all temporal dependency is removed. Number of times that differencing operation is repeated is called the difference order. Order difference is a method to remove trend.

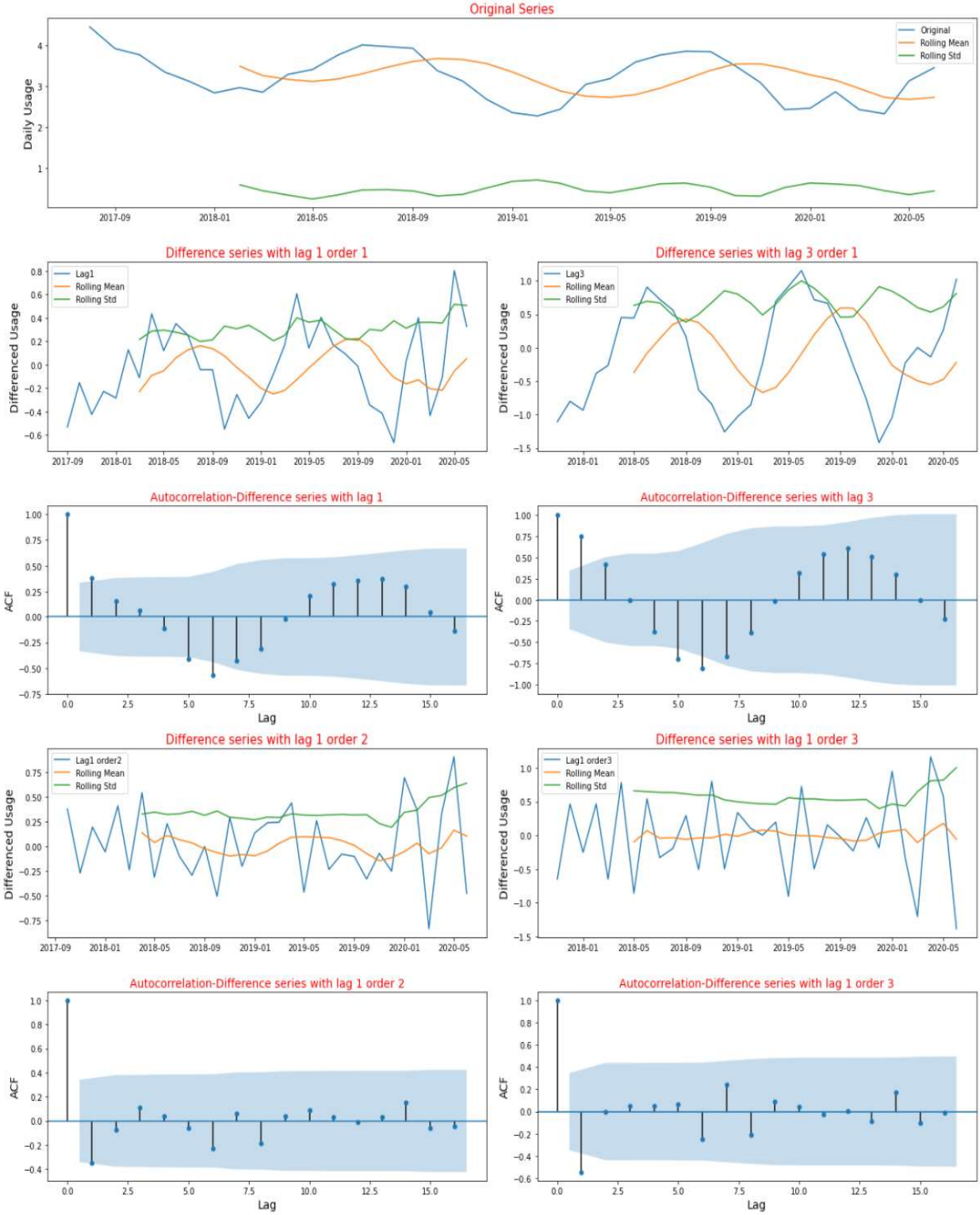
Performing lag and order difference makes the daily usage values independent from time and stationery.

After modeling and forecasting based on stationary time series, it is required to invert the difference operation for a single forecast.

Inverted value(t) = forecasted value (t) + difference(t)

Only lag differencing didn't make the series stationary but adding order differencing made it stationary. Plotting series of difference, autocorrelation plot and ADF test demonstrated that **difference series with lag 1 order 2 was stationary.**

Figure 21. Make Time Series Stationary using Difference Transforming



Lag1 Order2 Series

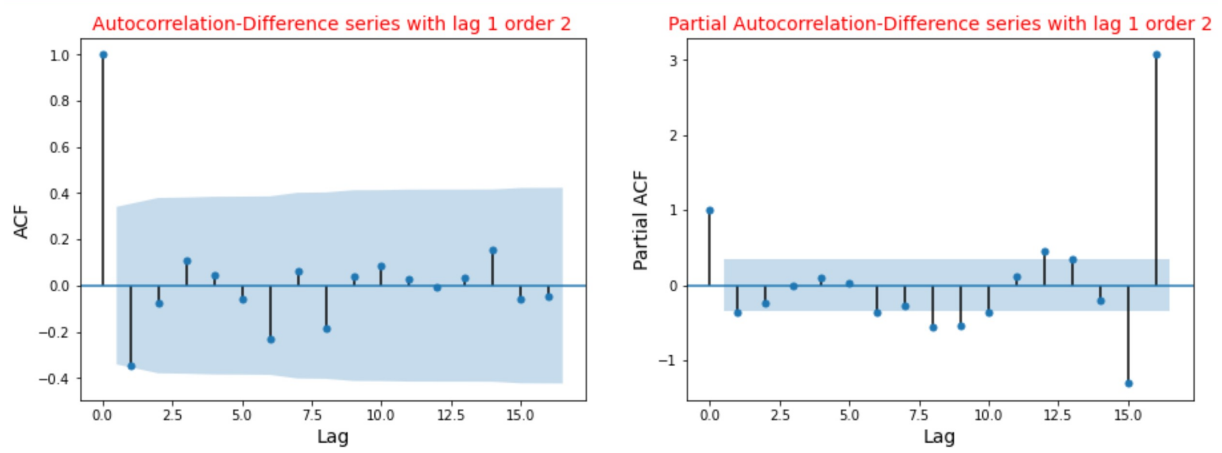
```
Results of Dickey-Fuller Test:
test statistic          -7.337845e+00
p-value                  1.084672e-10
# Lags Used              8.000000e+00
# Observation Used       2.400000e+01
Critical Value 1%       -3.737709e+00
Critical Value 5%       -2.992216e+00
Critical Value 10%      -2.635747e+00
dtype: float64
```

Test statistic is smaller than critical values at all significant levels, in addition p-value is quite small. Lag1 Order2 Series is stationary.

Lag1 Order3 Series

```
Results of Dickey-Fuller Test:
test statistic          -4.850687
p-value                  0.000043
# Lags Used              9.000000
# Observation Used       22.000000
Critical Value 1%       -3.769733
Critical Value 5%       -3.005426
Critical Value 10%      -2.642501
dtype: float64
```

Figure 22. ACF and PACF of Stationarized Time Series



Autocorrelation function of lag-one order-2 difference is pattern less, because seasonality adjustment eliminated the pattern. At first lag, ACF passed or is on the edge of confidence interval (p=1) and also at first lag PACF passed or is on the edge of confidence interval (q=1).

6. Algorithms and Machine Learning

Average daily usage per dwelling unit set as the target value. Then, data was spitted into train and test sets (70/30). I trained and tuned models on train set.

6.1. Univariant time series forecasting Models

Simple Exponential Smoothing Model (SES)

Model forecast only the next time step (one-month head) as an exponentially weighted linear function of observations at prior time steps. This model supports the non-stationary time.

Holt Winter's Exponential Smoothing (HWES)

Triple Exponential Smoothing (HWES) supports both seasonality and trend.

- Nature of the trend can be set as additive, multiplicative, or none,
- Nature of the seasonality can be set additive, multiplicative, or none
- Dampening of the trend or not

Autoregressive Model (AR)

Model forecasts the next step in the sequence as a linear function of the observations at prior time steps. Parameter of the model are:

- Number of AR terms(p) is the number of lags of dependent variable(past values). For instance if p is 3, the predictors for x(3) are x(1) and x(2).
- The trend to include in the model:
    - 'n' - No trend.
    - 'c' - Constant only.
    - 't' - Time trend only.
    - 'ct' - Constant and time trend.

ARMA(0,q) - Moving Average Model

The Moving Average model forecasts the next step in the sequence as the average of a window of observations at prior time steps. Parameters of the model are:

Number of MA (Moving Average) terms (q) is size of the moving average part window of the model i.e. lagged forecast errors in prediction equation.

Autoregressive Moving Average ARMA(p,q) Model

Model forecasts next step in the sequence by joining AR and MA.

Parameter of the model: The (p,q) order of the model for the number of AR parameters, and MA parameters to use.

Seasonal Autoregressive Integrated Moving Average (SARIMAX)

SARIMA supports seasonal time series. Model parameters are:

- Non-seasonal parameters of model (p,d,q):
  - Number of AR (Auto-Regressive) terms (p) is number of lags of dependent variable (past values).
  - Number of Differences (d) is associated with the integrated part of the mod el, which effects the amount of Differencing to apply to a time series. Non s tationarity series which has trend and seasonality requires positive level of differencing (d>0)
  - Number of MA (Moving Average) terms (q)is size of the moving average par t window of the model.
- seasonal parameters of model (P,D,Q)**:**

These parameters are liked (p,d,q) that applied to seasonal component

- m: The number of time steps for a single seasonal period

Summary of model has an attribute of BIC.

Bayesian Information Criterion

The information criteria adjust the goodness-of-fit of a model by imposing a penalty based on the number of parameters used.

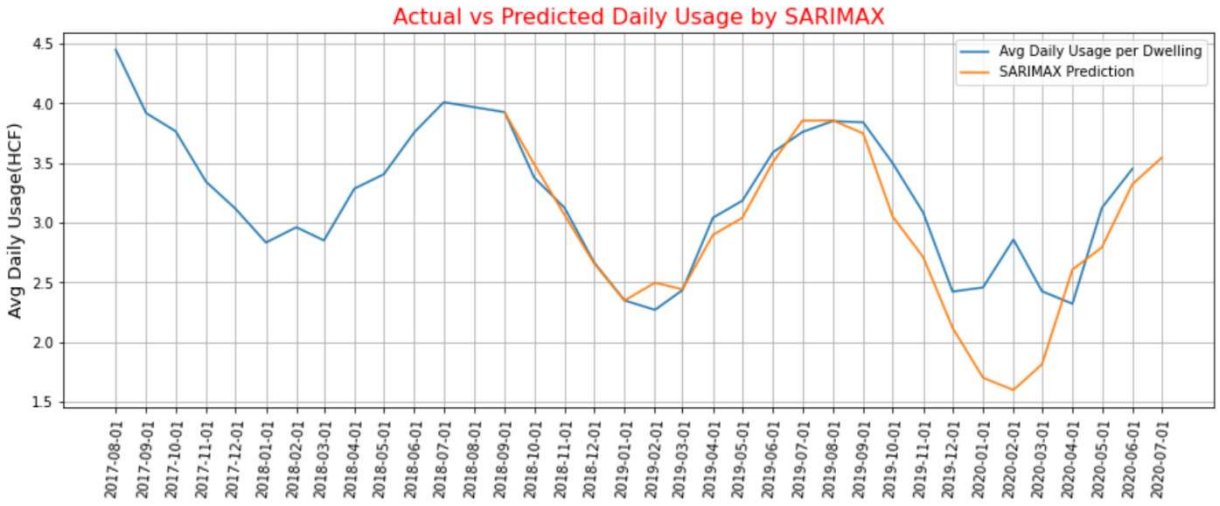
The way to use the information criteria is to fit several models, each with a different number of parameters, and choose the one with the lowest Bayesian Information Criterion. It deals with the trade-off between the goodness of fit of the model and the complexity of the model itself.

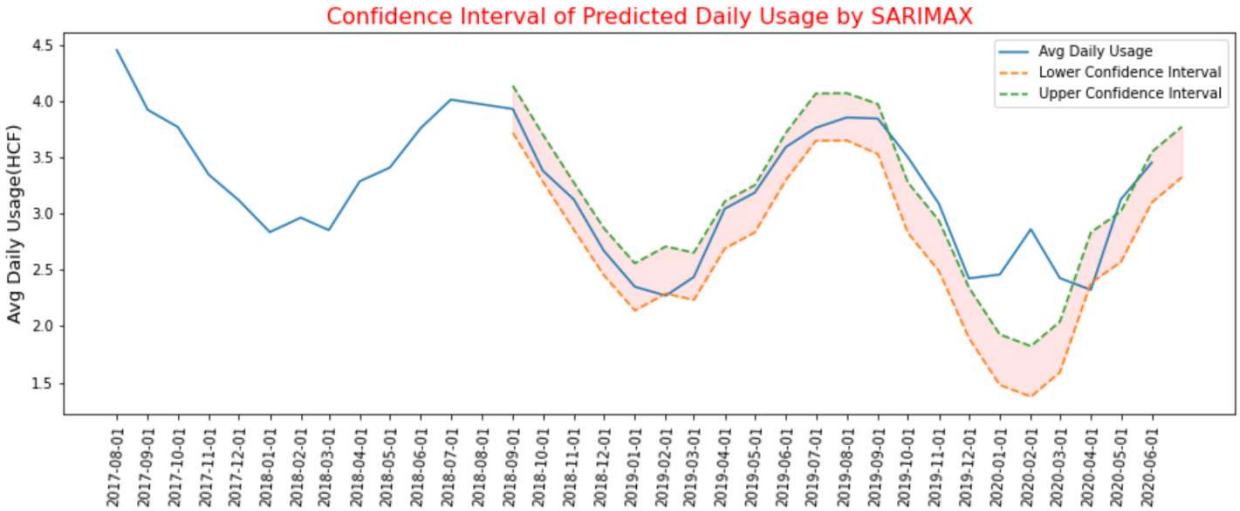
Univariant time series forecasting had very good performance since they had parameters to focus on trend, seasonality and mostly supports the non-stationary time series.

```
Params Grid for SARIMAX
```

	trend	param	seasonal_param	AIC
0	n	(1, 1, 2)	(1, 0, 0, 12)	-7.571470
1	t	(1, 1, 2)	(1, 0, 0, 12)	-5.628948
2	c	(1, 0, 0)	(1, 0, 0, 12)	-9.992228
3	ct	(1, 0, 0)	(1, 0, 0, 12)	-8.676631

I performed a grid parameters and model with constant trend had the lowest AIC. After fitting model with the best parameters, the test data was predicted.





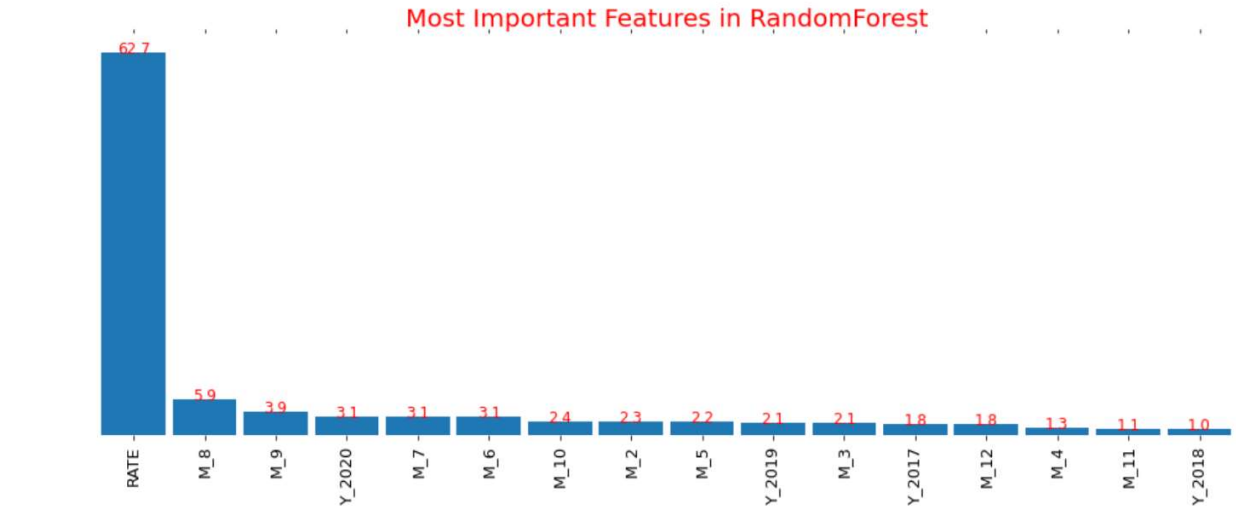
6.2 Performance of Univariant Time Forecasting Models

	Model	RMSE	MAE	MAPE	R2
0	SimpleExpSmoothing	0.409886	0.329360	11.462578	0.738515
1	ExponentialSmoothing	0.409886	0.329360	11.462578	0.738515
2	Autoregressive	0.313047	0.266939	9.541546	0.830775
3	ARMA(0,2)	0.178451	0.139238	5.016632	0.950049
4	ARMA(1,2)	0.178451	0.139238	5.016632	0.950049
5	SARIMA (1, 0, 0)x(1, 0, 0, 12)	0.537952	0.417924	15.147803	0.862873

6.3 Linear Models and Tree Based Models

In general, linear models and ensemble tree-based models even the tunned ones didn't have a good performance on time series prediction. It is mainly because they don't consider the trend or seasonality.

I build RandomForest Regressor and tunned by Randomized Search CV and Bayesian Optimization. The Tunned model estimated the importance of features as follow:



1) Customer Group
2) Month of Aguest
3) Month of September and year 2020
4) Month of June and July

6.4 Performance of Forecasting Models

Model	RMSE	MAE	MAPE	R2
SimpleExpSmoothing	0.409886	0.329360	11.462578	0.738515
ExponentialSmoothing	0.409886	0.329360	11.462578	0.738515
Autoregressive	0.313047	0.266939	9.541546	0.830775
ARMA(0,2)	0.178451	0.139238	5.016632	0.950049
ARMA(1,2)	0.178451	0.139238	5.016632	0.950049
SARIMA (1, 0, 0)x(1, 0, 0, 12)	0.537952	0.417924	15.147803	0.862873
BayesianRidge	0.274558	0.225684	200.045553	-0.273716
Lasso	0.262176	0.214882	196.283369	-0.434866
ElasticNet	0.263902	0.216201	197.374565	-0.422470
RandomizedSearchCV RandomForest Regressor	0.034560	0.028109	21.610100	0.194310
XGB Regressor	0.034560	0.028109	21.610100	0.194310
Bayesian Process-RandomForest Regressor	0.034721	0.028313	21.765036	0.189995