Sequential Analysis Forecasting

Yevhen Horban

April 2021

1 Introduction

This project is focused on identifying the patterns and predicting the number of roses produced by a specific greenhouse. Plants require nutritious soil, water, sunlight, and carbon dioxide to grow. While the greenhouse controls the constant supply of nutrition and water, environmental conditions vary and potentially affect the flowers' growth. The amount of roses produced appears to have a sinusoidal pattern with a variable vertical shift, amplitude, and period. This project will attempt to determine how this sinusoidal pattern is affected by environmental conditions to predict future production accurately.

The nature of this particular business is such that the flowers are often sold before they are produced. Knowing the available quantity ahead of time can advise the amount the greenhouse will be able to sell and adjust the price accordingly to maximize profit. Additionally, each rose brand has a distinct sinusoidal production pattern, and when their peaks align, the greenhouse runs out of cold storage space and has a shorter window to sell the produce. Making accurate predictions of the amount produced can benefit the business financially and reduce the waste of resources required to produce roses and roses themselves.

This project assumes that if the environmental factors were controlled, then the number of flowers produced would follow a sinusoidal pattern with no noise. We will note the importance of national holidays that correlate with the increased demand for flowers like March 8th and February 14th. A predictive model for the production of one brand of flowers should be generalizable for the other brands the greenhouse produces. Additionally, the created model has to be equal or better in performance of the standard time series prediction techniques like ARIMA [3], LSTM [4], and Prophet [5].

2 Data

The time-series data that will be used for this project is a combination of two datasets. First, the data from the company that owns the greenhouse recorded by the sensors: the lux of the sun `lm/m`^{2}, the temperature inside `°C`, the concentration of CO_2 in `ppm`, number of `minutes` the lamps were turned on, and the `number` of Red Eagle brand of roses produced. Second, the historical weather data from the scanners at a nearby airport available at Weather Underground containing average values of temperature `°C`, dew point `°C`, humidity `%`, pressure `mmHg`.

The data set is already cleaned and includes 1037 entries (almost three years of daily entries). The last project [1] emphasized feature engineering, but it will not be the primary focus for this project. For example, it explored the accumulative effect of several days of high temperatures on roses production. In this project, both features and the target variable will be smoothed with a mean value of the last seven days and include an additional boolean feature representing that an important date will occur in seven days or less. Below are the summary statistics for some of the variables in the dataset:

	rsum	lamps	t_inside	co2	t_avg	dp_avg	h_avg	p_avg	target	imp_day
mean	1114.30	753.01	21.12	740.60	8.95	4.15	76.53	29.56	7251.73	0.06
std	831.66	436.47	1.67	184.32	9.43	7.84	15.96	0.24	3075.43	0.24
min	36.00	0.00	16.50	388.00	-17.10	-19.70	30.30	28.60	716.00	0.00
max	2994.00	1440.00	27.60	1217.00	29.70	26.20	100.00	30.30	32588.00	1.00

Figure 1: Summary Statistics for original data

	rsum	lamps	t_inside	co2	t_avg	dp_avg	h_avg	p_avg	target	imp_day
mean	1118.83	753.01	21.14	740.46	8.98	4.16	76.46	29.56	7226.71	0.06
std	752.29	436.47	1.51	172.08	9.09	7.29	13.78	0.17	2559.52	0.24
min	77.62	0.00	16.79	435.12	-11.82	-13.71	41.50	29.14	758.00	0.00
max	2884.88	1440.00	26.64	1095.38	25.62	19.49	99.70	30.05	16054.88	1.00

Figure 2: Summary Statistics for smoothed data

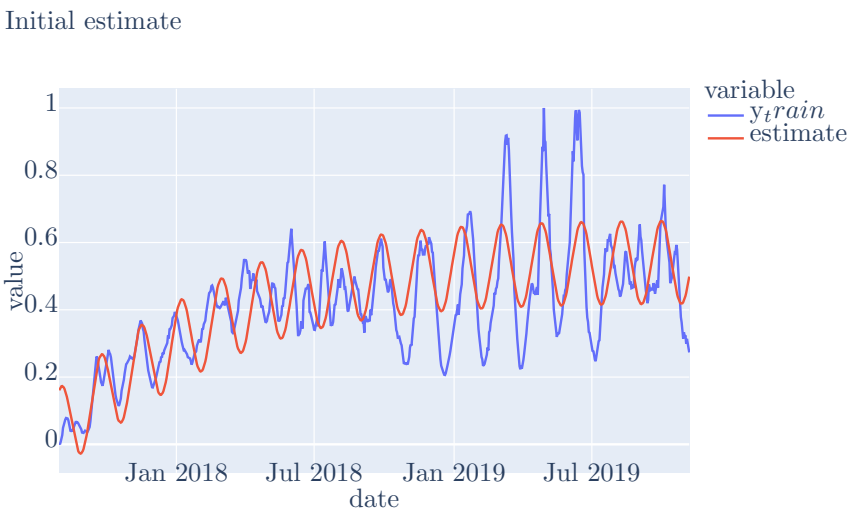
The two tables above show that smoothing the data reduces the variance and ignores abnormal peaks or dips. There might be shortcomings to doing smoothing, but hopefully, the project will get a chance to investigate whether it will significantly affect the model's performance. Note: lamps and imp_day were not smoothed. After smoothing, parameters and target are split into training and testing, all except `imp_day` are fit to the training data and transformed with MinMaxScaler.

3 Method

The previous project on this topic used the common train-test split approach. However, this might not be suitable for the desired model's goal because the evaluation should happen on the ability to predict the future rather than to connect the gaps in the data. Ideally, the model would be daily supplied with new data, compare the accuracy of the prediction is made, adjust its parameters and make new predictions. The current approach splits the first 80% and last 20% of the data points into training and testing.

The function chosen to approximate the target variable is the sum of general forms of the logistic regression and a sinusoid:

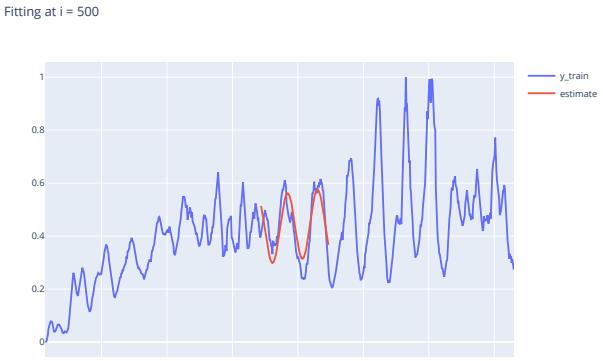
$$a(x) = \frac{A}{1 + e^{Bx}} + D$$
$$b(x) = Asin(Bx + C) + D$$
$$f(x) = a(x) + b(x)$$

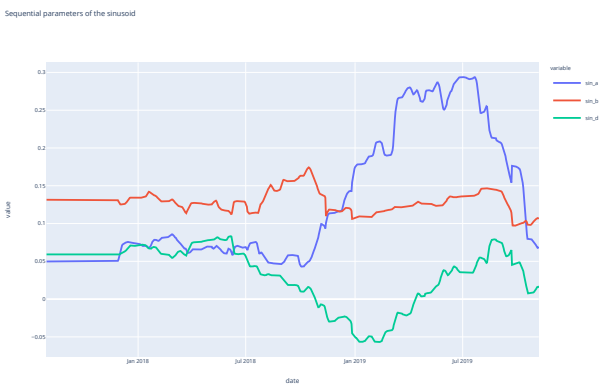


The model makes an initial estimate over the whole range of the training data to get a starting point for sequential fitting. The project assumes that the training data is in the range between zero and one and hence will have a mean around 0.5. Additionally, when a brand of roses is planted, the initial production is close to zero, which means that we can use the parameters $A = 1$, $D = -0.5$, and B is the only parameter that needs fitting. One parameter of logistic regression and all sinusoid parameters are fit by minimizing the sum squared error compared to the given target variable. The R2 value for the initial fit is around .5 and will serve as a baseline in future analysis.

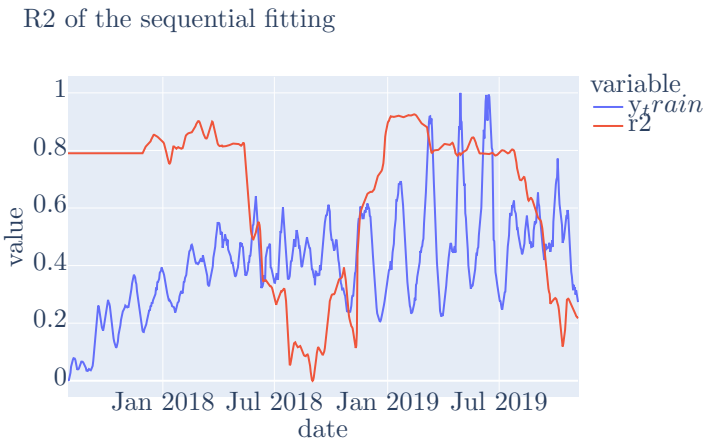
After the initial fit, the logistic regression parameter is saved and remains fixed for further fitting. Then, the model sequentially fits and records the sinusoidal parameters for M number of days from left to right. It is possible to adjust M when initializing the model, but $M = 120$ seemed to be the best value during the experimentation. Since the model uses the sinusoidal parameters to predict, the first M entries in the data frame will be the same.

Bellow, you can see the visualization of the saved sinusoid parameters. Note that the C parameter is not shown because it has a different scale, but it is used in the prediction. It might be in the interest of this project to leave the curve's vertical shift to the logistic regression and ignore the D parameter of the sinusoid. However, this attempt has not improved the results, likely due to a more significant amount of variance possible to explain by keeping an extra parameter.





After the sequential scanning is complete, the model uses a given regressor as the base in a MultiOutputRegressor and fits the training data to recorded sinusoidal parameters. Making a prediction is a two-step process: first, given X_train, the MultiOutputRegressor predicts the sinusoidal parameters, then the sinusoidal parameters are used to make the prediction. The caveat lies in an additional parameter required to calculate $f(x)$. During fitting, x ranges from zero to the training size and assumes that the x range for the testing data starts from the training size.



The graph above demonstrates that some parts of the target variable resemble a sinusoid more than others. It seemed like a good idea to record both the sinusoidal parameters and the r2 of the fit to emphasize more

stable periods during training by using r2 values as weights. Unfortunately, this approach did not improve the results.

4 Results

So far, the project was focused on evaluating the model visually and based on the $r2$ score of the sinusoidal parameters, training, and testing data. An expectation of the project that Linear Regression would work well as the base estimator of the MultiOutputRegressor was not met. Neither did most of the regressors the project experimented with. Out of Linear, Lasso, Ridge, Lars, SVR, ElasticNet, SGDRegressor, GradientBoostingRegressor, AdaBoostRegressor, DecisionTreeRegressor, and RandomForestRegressor, only the last two were able to achieve acceptable results.

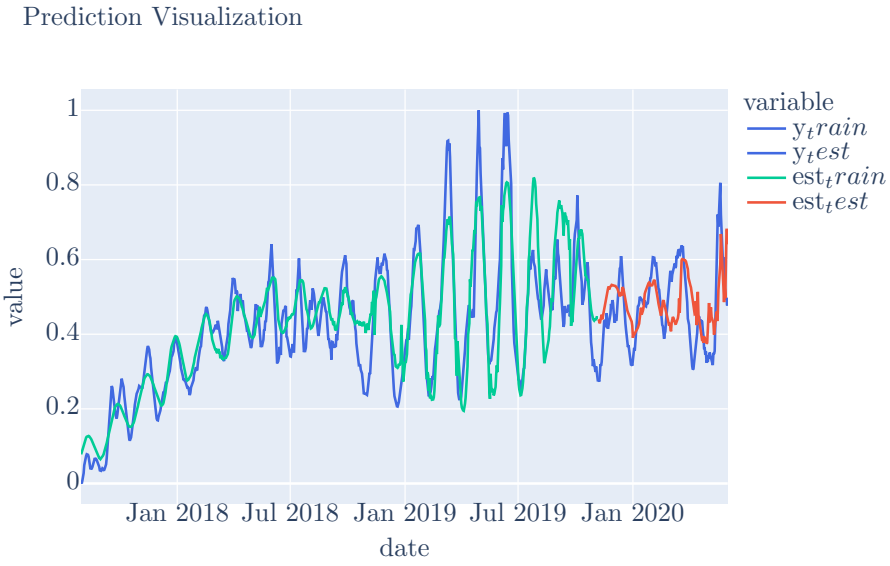
	e	M	sin_params	train	test
RandomForestRegressor(max_depth=15, random_sta...	120		0.999	0.753	0.263
DecisionTreeRegressor(max_depth=10, random_sta...	120		1.000	0.752	0.191
GradientBoostingRegressor(random_state=0)	120		0.996	0.715	-0.316
AdaBoostRegressor(random_state=0)	120		0.939	0.048	-0.318
SVR()	90		0.138	0.050	-0.344

Figure 3: Regressor Search

The project initially assumed that the model's performance on the testing data would depend on how well it could predict sinusoidal parameters and the training subset of the target. As one can see in the table above, AdaBoostRegressor and SVR prove both assumptions wrong. On the other hand, DecisionTreeRegressor, RandomForestRegressor, and some other regressors satisfy the goal of increasing the amount of variance explained on the training data. Generally, poor performance is likely due to the project's evaluation choice. If the project was instead focused on predicting a short-term future, the results could be very different.

bootstrap	criterion	max_depth	max_features	n_estimators	time_avg	sin_params_avg	train_avg	test_avg
True	mae	NaN	auto	100	11.155625	0.999000	0.760667	0.269333
True	mae	15.0	auto	100	11.210051	0.999000	0.752333	0.246667
True	mse	10.0	auto	50	1.715407	0.999000	0.745000	0.223333
True	mae	10.0	auto	100	15.165274	0.998667	0.747333	0.219333
True	mae	NaN	auto	50	6.312004	0.999000	0.750667	0.203333

Figure 4: Random Forest Regressor Grid Search



The graph above shows the prediction of the RandomForestRegressor on both training and testing data. The higher score on the training data is more apparent visually. The model seems to capture the change in the sinusoid's amplitude and the periods seem to align better than the initial estimate. As a sanity check, the r2 score for predicting the testing data using the initial estimate parameters is -1.4, which is significantly lower than the score of the best performing regressors. Unfortunately, the methodology of this project does not allow for cross-validation, so instead, the tuning was done without specifying random states and recording mean scores over three runs. An average test score of the RandomForestRegressor of .269 compared to -1.4 could be considered pretty good.

x	0.825823	x	0.557168	x	0.661575	x	0.382621
rsum	0.093517	dp_avg	0.113795	dp_avg	0.093778	t_inside	0.239520
t_inside	0.021872	t_inside	0.093796	lamps	0.069618	dp_avg	0.156746
t_avg	0.013874	rsum	0.053461	t_avg	0.045685	t_avg	0.070604
co2	0.012823	lamps	0.049447	t_inside	0.033265	lamps	0.051609
p_avg	0.008436	t_avg	0.043309	p_avg	0.032589	rsum	0.033741
h_avg	0.008203	p_avg	0.035040	rsum	0.025335	co2	0.025472
dp_avg	0.007998	h_avg	0.025857	co2	0.019675	h_avg	0.021611
lamps	0.007111	co2	0.025846	h_avg	0.018242	p_avg	0.017700
imp_day	0.000343	imp_day	0.002282	imp_day	0.000237	imp_day	0.000376
Name: sin_a, dtype:		Name: sin_b, dtype:		Name: sin_c, dtype:		Name: sin_d, dtype:	

Figure 5: Feature Importance

Surprisingly, only after adding the index column to the training and the testing data improved the estimators' performance became acceptable. The lack of explanation of the data provided by the features could have contributed to the poor general performance of the regressors. Only the performance of the regressors that are prone to overfit was improved after adding this additional feature. From the table above, x has the most significant importance for predicting all of the sinusoidal parameters. Only in the vertical shift x is less than half responsible for the prediction. Maybe that is why not recording the D

parameter of the sinusoid did not improve the results.

5 Conclusion

After completing the project, I realized that the approach and evaluation do not make a lot of sense practically. I mention in the beginning that ideally, the model would be making predictions each day, get evaluated and adjusted. This approach sounded a lot like reinforcement learning, and my lack of experience and the instructors' suggestion steered me away from this method. However, I would be interested in framing this problem in this way in the future if I get a chance.

Unfortunately, one of the goals of comparing the performance of the engineered model to the common time series analysis techniques was not satisfied. This project focused too much on one particular implementation, and other methods should have been in my mind from the very beginning if they were to be implemented.

Despite the limitations of this project, the results are quite astonishing considering how far into the future the prediction is made. If the model were used in real life, short-term weather predictions would not be an issue because prediction would require smoothed data represents a general trend rather than sharp volatility. I wish I had more time to investigate the importance of x for predicting the sinusoidal parameters. Additionally, I wonder if any of the steps in the method could have compromised the performance, for example:

- Smoothing the training and testing data
- Using x as a feature
- Predicting too far into the future
- Not making cross-validation possible
- Recording M data points at the beginning of the sinusoidal parameters data frame

Code can be found here https://github.com/h0rban/sequential-analysis-forecasting

References

[1] Yevhen Horban, and Pranav Ahluwalia. *Flower Harvest Prediction*. Boston, June 2020. https://github.com/h0rban/flower-harvest-prediction

[2] Neelam Tyagi *Introduction to Time Series Analysis in Machine learning*. Jan 05, 2020. https://www.analyticssteps.com/blogs/introduction-time-series-analysis-time-series-forecasting-machine-learning-methods-models

[3] Jason Brownlee. *How to Create an ARIMA Model for Time Series Forecasting in Python*. January 9, 2017. https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/

[4] Jason Brownlee. *Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras*. July 21, 2016. https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/

[5] Facebook Open Source. *Prophet*. https://facebook.github.io/prophet/docs/quick$_s$*tart.html*