# ARIMA and LSTM Univariate Modeling steps

## ARIMA Definitions

AR = a Auto Regression model

p = the order of the AR term

MA = a moving Average only model

q = the order of the MA term

d = the number of differencing required to make the time series stationary

1. Split the data into two dataframes; one for conventional avocados and one for Hass avocados. These along with the complete dataset (both types) will be modeled

2. ARIMA only - Attempt to determine if the data is stationary. Conduct the Dicky Fuller test. H0 = avg\_price is non-stationary. P value was 0.0 so H0 is rejected. Ha, avg\_price is stationary. Note - I still tried differencing to 1. learn and 2. I notice fluctuations in the data; possibly due to seasonality.

3. ARIMA only - Looked at the lags visually. Based on this believe p and q should equal 1.

4. Split the data. 75% for training and 25% for test. since the order of the data must be maintained, did not randomly split.

5. ARIMA only - Used auto arima which sequences through the different p, and q options to minimize Akaike Information Criteria (AIC). A statistical method that quantifies goodness of fit. This produces a model with p =3, d = 0, q = 3. Since "d" is 0, this also confirms the Dicky Fuller test that avg price is non-stationary.

6. Created a forecast from the test data for both ARIMA and LSTM.

7. The ARIMA model's have a number of scoring metrics but the LSTM models are scored on Root Mean Square Error (RMSE). The ARIMA RMSE score = 0.29 for the combined data set, RMSE = 0.21 for the conventional dataset and organic data set = 0.27, . This model is also based on using auto arima which uses the best model having p = 3, d = 0, q = 3.

8. LSTM models are scored on Root Mean Square Error (RMSE). By iterating through LSTM neurons and epochs, which did not change the results significantly, I found the results to be Train Score = 0.34 and Test Score = 0.34.

# LSTM Multivariate Modeling steps

1. Defined a function for creating lags, series\\_to\\_supervised. This transposes and shifts the parameter to the same row but next column. t, t-1, t-2, etc. Doing this introduces Nan's in the bottom rows equal to the number of lags so the function drops those rows.

2. Read in the cleaned data.

3. generate the categorical variables and encodes them.

4. Scale the data using MinMaxScaler.

5. for the 2nd model, create an extra variable to allow changing the number of lags used.

6. use series\_to\_supervised function to create the lags.

7. Drop the columns we do not want to predict. We want to predict avg\_price, with is t1. For multiple lags we do not drop the extra T1's.

8. Split the data for training and testing. Because we're looking at the previous value to determine the impact of the current value we split the data by time, meaning we will use the 2017 through 2019 data to predict 2020. 2020 data goes from Jan 6th through sept. 9th.

9. Reshape the data into a 3 dimensional dataset. samples, timesteps, features. LSTM requires this.

10. instantiate the model.

11. Fit the training data. Note I iterated through various combinations of neurons, epochs, batch sizes and activation functions.

12. Plot the train vs test loss,

13. predict the average prices.

14. Invert the scaling for both the forecast and the actual data.

15. Calculate the RMSE. The RMSE for the combined dataset is 0.22, For conventional RMSE = 0.162 and for organic the RMSE = 0.17.

By iterating through the number of lags. I found 3 to 4 lags, depending on which dataset was run, produced the best results. The sigmoid activation function produced the best results. By iterating through different features, pulling them out of the model, found the following features hurt the model results: 'phdi','zndx', 'pmdi', 'cdd', 'sp01', 'sp02', 'sp03', 'sp06', 'sp09', 'sp12', 'sp24','tmin', 'tmax'. Conversely the average temperature (tavg) and precipitation (pcp) positively contributed to the model.

16. Merge the predictions with original features to graph actual vs predictions.