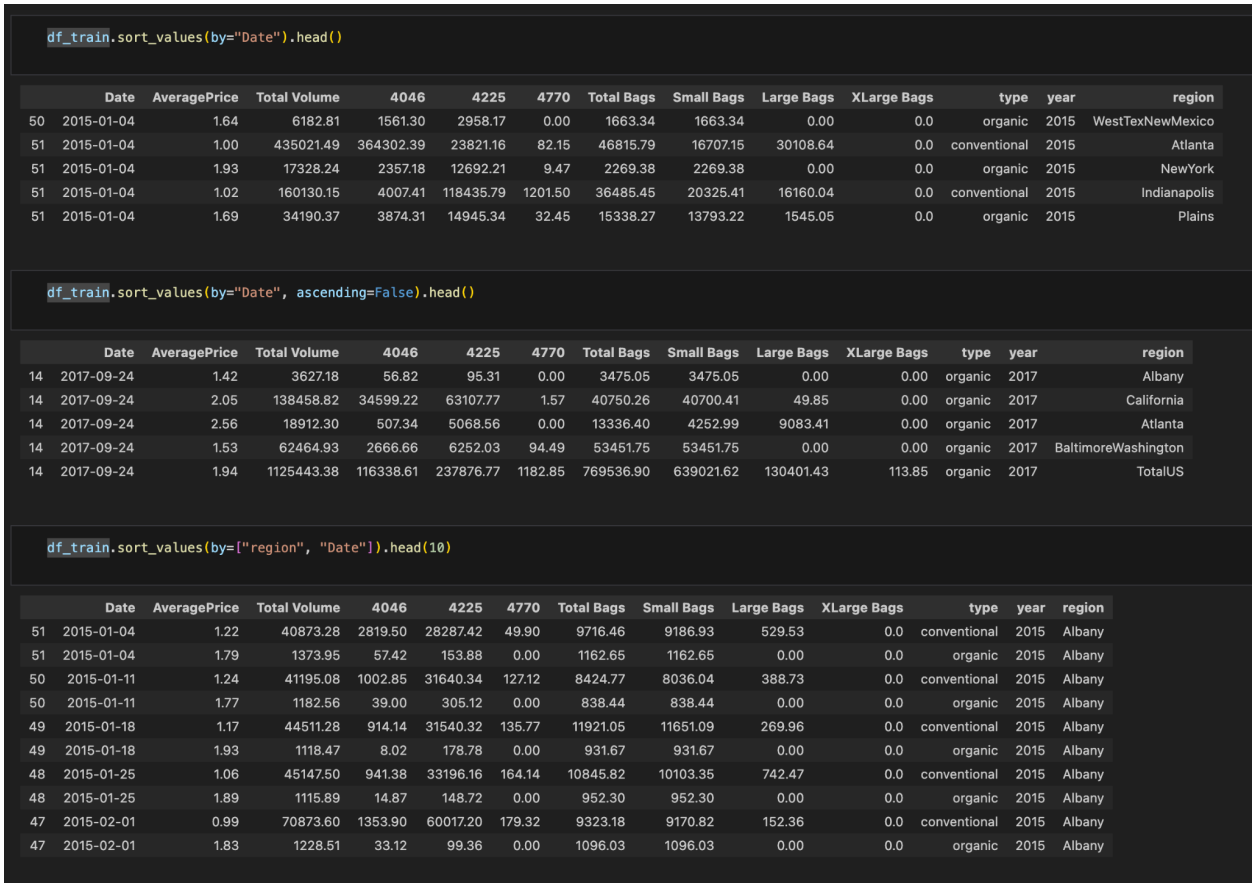
Unraveling the Mystery of Avocado Prices: A Time Series Detective Story

by Chloe Van

Posted on Dec 5, 2023

Imagine you're a detective in a digital world where every tick of the clock leaves a breadcrumb of data. Your mission is to solve the mystery hidden within these clues, predicting future events based on past patterns. This is the essence of time series, the focus of my recent machine learning assignment.

In this digital detective story, this case revolves around avocados, a seasonal fruit known for their versatility, health benefits, and tie to nature, affecting their availability and price. The first step in my investigation was examining the nature of the avocado dataset to understand its structure, analogous to a detective investigating various leads.



```
df_train.sort_values(by="Date").head()
```

	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	type	year	region
50	2015-01-04	1.64	6182.81	1561.30	2958.17	0.00	1663.34	1663.34	0.00	0.0	organic	2015	WestTexNewMexico
51	2015-01-04	1.00	435021.49	364302.39	23821.16	82.15	46815.79	16707.15	30108.64	0.0	conventional	2015	Atlanta
51	2015-01-04	1.93	17328.24	2357.18	12692.21	9.47	2269.38	2269.38	0.00	0.0	organic	2015	NewYork
51	2015-01-04	1.02	160130.15	4007.41	118435.79	1201.50	36485.45	20325.41	16160.04	0.0	conventional	2015	Indianapolis
51	2015-01-04	1.69	34190.37	3874.31	14945.34	32.45	15338.27	13793.22	1545.05	0.0	organic	2015	Plains

```
df_train.sort_values(by="Date", ascending=False).head()
```

	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	type	year	region
14	2017-09-24	1.42	3627.18	56.82	95.31	0.00	3475.05	3475.05	0.00	0.00	organic	2017	Albany
14	2017-09-24	2.05	138458.82	34599.22	63107.77	1.57	40750.26	40700.41	49.85	0.00	organic	2017	California
14	2017-09-24	2.56	18912.30	507.34	5068.56	0.00	13336.40	4252.99	9083.41	0.00	organic	2017	Atlanta
14	2017-09-24	1.53	62464.93	2666.66	6252.03	94.49	53451.75	53451.75	0.00	0.00	organic	2017	BaltimoreWashington
14	2017-09-24	1.94	1125443.38	116338.61	237876.77	1182.85	769536.90	639021.62	130401.43	113.85	organic	2017	TotalUS

```
df_train.sort_values(by=["region", "Date"]).head(10)
```

	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	type	year	region
51	2015-01-04	1.22	40873.28	2819.50	28287.42	49.90	9716.46	9186.93	529.53	0.0	conventional	2015	Albany
51	2015-01-04	1.79	1373.95	57.42	153.88	0.00	1162.65	1162.65	0.00	0.0	organic	2015	Albany
50	2015-01-11	1.24	41195.08	1002.85	31640.34	127.12	8424.77	8036.04	388.73	0.0	conventional	2015	Albany
50	2015-01-11	1.77	1182.56	39.00	305.12	0.00	838.44	838.44	0.00	0.0	organic	2015	Albany
49	2015-01-18	1.17	44511.28	914.14	31540.32	135.77	11921.05	11651.09	269.96	0.0	conventional	2015	Albany
49	2015-01-18	1.93	1118.47	8.02	178.78	0.00	931.67	931.67	0.00	0.0	organic	2015	Albany
48	2015-01-25	1.06	45147.50	941.38	33196.16	164.14	10845.82	10103.35	742.47	0.0	conventional	2015	Albany
48	2015-01-25	1.89	1115.89	14.87	148.72	0.00	952.30	952.30	0.00	0.0	organic	2015	Albany
47	2015-02-01	0.99	70873.60	1353.90	60017.20	179.32	9323.18	9170.82	152.36	0.0	conventional	2015	Albany
47	2015-02-01	1.83	1228.51	33.12	99.36	0.00	1096.03	1096.03	0.00	0.0	organic	2015	Albany

Figure 1: A snippet of the avocado dataset sorted by: Date ascending (top), Date descending (middle), Date & Region ascending (bottom)

My exploratory data analysis (Figure 1) revealed two distinct time series representing unique sequences of avocado prices over time. This was identified by observing measurements across regions and types on the same day. It was crucial to confirm that these time series had equally spaced measurements, similar to verifying the reliability of witnesses in an investigation. This consistency was vital to my analysis, as irregular data intervals could lead to misleading conclusions.

```
df_hastarget = create_lag_feature(df_sort, "AveragePrice", +1, ["region", "type"], "AveragePriceNextWeek", clip=True)
df_hastarget
```

	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	type	year	region	AveragePriceNextWeek
0	2015-01-04	1.22	40873.28	2819.50	28287.42	49.90	9716.46	9186.93	529.53	0.0	conventional	2015	Albany	1.24
1	2015-01-11	1.24	41195.08	1002.85	31640.34	127.12	8424.77	8036.04	388.73	0.0	conventional	2015	Albany	1.17
2	2015-01-18	1.17	44511.28	914.14	31540.32	135.77	11921.05	11651.09	269.96	0.0	conventional	2015	Albany	1.06
3	2015-01-25	1.06	45147.50	941.38	33196.16	164.14	10845.82	10103.35	742.47	0.0	conventional	2015	Albany	0.99
4	2015-02-01	0.99	70873.60	1353.90	60017.20	179.32	9323.18	9170.82	152.36	0.0	conventional	2015	Albany	0.99
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
18243	2018-02-18	1.56	17597.12	1892.05	1928.36	0.00	13776.71	13553.53	223.18	0.0	organic	2018	WestTexNewMexico	1.57
18244	2018-02-25	1.57	18421.24	1974.26	2482.65	0.00	13964.33	13698.27	266.06	0.0	organic	2018	WestTexNewMexico	1.54
18245	2018-03-04	1.54	17393.30	1832.24	1905.57	0.00	13655.49	13401.93	253.56	0.0	organic	2018	WestTexNewMexico	1.56
18246	2018-03-11	1.56	22128.42	2162.67	3194.25	8.93	16762.57	16510.32	252.25	0.0	organic	2018	WestTexNewMexico	1.56
18247	2018-03-18	1.56	15896.38	2055.35	1499.55	0.00	12341.48	12114.81	226.67	0.0	organic	2018	WestTexNewMexico	1.62

18141 rows × 14 columns

Figure 2: A snippet of creating a lag feature named AveragePriceNextWeek

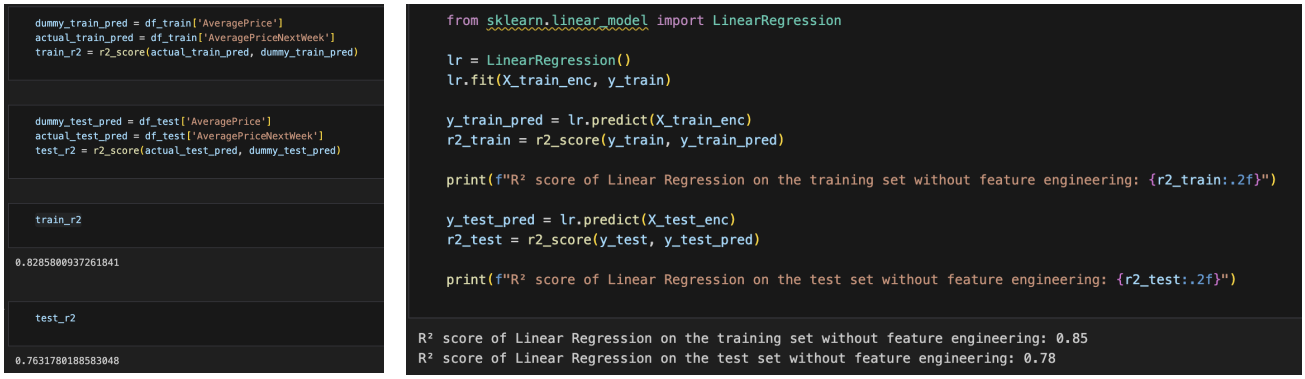
I created an 'AveragePriceNextWeek' feature, a lagged version of the 'AveragePrice', to uncover patterns for future insights (Figure 2). By providing a retrospective lens, this lagged feature helped me identify how past prices influenced avocado prices the following week. To ensure the strength of my investigation, I excluded rows where the 'target' data was missing, thereby preserving the integrity and accuracy of my findings akin to a detective ensuring that only complete and reliable evidence is considered in a case.

I then developed models to forecast next week's average avocado prices. As a foundational step, I established a baseline model, assuming that 'if today's price is X, then tomorrow's price will likely be the same.' This model's R^2 scores serve as a reference point to compare more complex models. As a benchmark, if a complex model couldn't outperform this baseline model, it must not be a very good model.

An R^2 score measures how well my "detective's hypothesis" (in this case, my baseline and linear regression models) matches the actual "facts of the case" (the real-world target of avocado prices). An R^2 score ranges from 0 to 1, where scores closer to 1 indicate that this model does a great job of predicting avocado prices. On the other hand, a score closer to 0 would mean that this model doesn't do a good job capturing the data trends and makes poor predictions.

Next, I experimented with a few approaches with Linear Regression, which helps me draw a straight line through a scatter plot of data points - each representing an avocado's price on a specific day. This line attempts to capture the general trend of data, showing me the direction of avocado prices moving over time. I chose Linear Regression for its simplicity and effectiveness,

providing a clear and straightforward way to understand trends with continuous values as targets.



```
dummy_train_pred = df_train['AveragePrice']
actual_train_pred = df_train['AveragePriceNextWeek']
train_r2 = r2_score(actual_train_pred, dummy_train_pred)

dummy_test_pred = df_test['AveragePrice']
actual_test_pred = df_test['AveragePriceNextWeek']
test_r2 = r2_score(actual_test_pred, dummy_test_pred)

train_r2
```

```
0.8285800937261841
```

```
test_r2
```

```
0.7631780188583048
```

```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(X_train_enc, y_train)

y_train_pred = lr.predict(X_train_enc)
r2_train = r2_score(y_train, y_train_pred)

print(f"R² score of Linear Regression on the training set without feature engineering: {r2_train:.2f}")

y_test_pred = lr.predict(X_test_enc)
r2_test = r2_score(y_test, y_test_pred)

print(f"R² score of Linear Regression on the test set without feature engineering: {r2_test:.2f}")
```

```
R² score of Linear Regression on the training set without feature engineering: 0.85
R² score of Linear Regression on the test set without feature engineering: 0.78
```

Figure 3: Left: snippet of baseline model R² scores, Right: snippet of linear regression model R² scores

As shown in Figure 3, the R² scores from Linear Regression only slightly improved compared to the baseline, suggesting that this basic approach (without feature engineering) was too straightforward. This model couldn't grasp the periodic nature of the data, which lay in how I encoded Date as integers. This approach limited the model's effectiveness, which could not flex around the more complex patterns of time.

To dive deeper, I experimented with various methods of encoding time. The first method involved extracting the "month" from the Date to capture the fluctuations in avocado prices throughout the year. Given that avocados are seasonal fruits, the relevance of the months seemed significant in understanding the relationship between price and availability, especially under the influence of supply and demand laws. The improvement in my results confirmed that this approach was more insightful, as seen in the improved R² scores below (Figure 4).

```
# Linear Regression with feature engineering extracting Month from Date

X_train_enc, y_train, X_test_enc, y_test, preprocessor = preprocess_features(
    df_month_train,
    df_month_test,
    numeric_features,
    categorical_features + ["Month"],
    drop_features,
    target
)

lr_month = LinearRegression()
lr_month.fit(X_train_enc, y_train)

y_train_pred = lr_month.predict(X_train_enc)
r2_train = r2_score(y_train, y_train_pred)

print(f"R² score of Linear Regression on the training set with Month: {r2_train:.2f}")

y_test_pred = lr_month.predict(X_test_enc)
r2_test = r2_score(y_test, y_test_pred)

print(f"R² score of Linear Regression on the test set with Month: {r2_test:.2f}")
```

```
Number of OHE features: 68
Total columns in new_columns: 78
Transformed data shape: (15441, 78)
R² score of Linear Regression on the training set with Month: 0.85
R² score of Linear Regression on the test set with Month: 0.80
```

```
# Linear Regression with feature engineering extracting Season from Month and Date

X_train_enc, y_train, X_test_enc, y_test, preprocessor = preprocess_features(
    df_season_train,
    df_season_test,
    numeric_features,
    categorical_features + ["Season"],
    drop_features,
    target
)

lr_season = LinearRegression()
lr_season.fit(X_train_enc, y_train)

y_train_pred = lr_season.predict(X_train_enc)
r2_train = r2_score(y_train, y_train_pred)

print(f"R² score of Linear Regression on the training set with Season: {r2_train:.2f}")

y_test_pred = lr_season.predict(X_test_enc)
r2_test = r2_score(y_test, y_test_pred)

print(f"R² score of Linear Regression on the test set with Season: {r2_test:.2f}")
```

```
Number of OHE features: 60
Total columns in new_columns: 70
Transformed data shape: (15441, 70)
R² score of Linear Regression on the training set with Season: 0.85
R² score of Linear Regression on the test set with Season: 0.79
```

Figure 4: Left: snippet of linear regression extracting month R² scores, Right: snippet of linear regression extracting seasons R² scores

The second method involved extracting "seasons" to categorize the data more broadly. However, this was like a detective making general assumptions. Seasons can have significant variability among themselves, where conditions in early summer versus late summer can differ significantly. The overgeneralization slightly reduced the model's effectiveness, as shown in the test set R^2 scores (Figure 4).

```python
# Linear Regression with feature engineering extracting days_since

X_train_enc, y_train, X_test_enc, y_test, preprocessor = preprocess_features(
    df_days_since_train,
    df_days_since_test,
    numeric_features + ["Days_since"],
    categorical_features,
    drop_features,
    target
)

lr_days_since = LinearRegression()
lr_days_since.fit(X_train_enc, y_train)

y_train_pred = lr_days_since.predict(X_train_enc)
r2_train = r2_score(y_train, y_train_pred)

print(f"R² score of Linear Regression on the training set with days_since: {r2_train:.2f}")

y_test_pred = lr_days_since.predict(X_test_enc)
r2_test = r2_score(y_test, y_test_pred)

print(f"R² score of Linear Regression on the test set with days_since: {r2_test:.2f}")
```

```
Number of OHE features: 56
Total columns in new_columns: 67
Transformed data shape: (15441, 67)
R² score of Linear Regression on the training set with days_since: 0.85
R² score of Linear Regression on the test set with days_since: 0.84
```

Figure 5: snippet of linear regression extracting days_since R^2 scores

The most effective method was extracting the "days since" feature, as shown by the best R^2 scores (Figure 5). Using the first Date of the data set as a reference date, this approach was like a detective piecing together an entire history, recognizing that patterns and behaviors evolve over long periods. This could potentially capture influences such as changing agricultural practices, climate change and increased market demand. I reduced seasonal bias by preventing the model from pigeonholing into expectations based on the time of the year only.

While the results are promising, there are several reasons why my model may not be as robust as they appear. One primary concern is the risk of overfitting, which means the model has become too tailored to the nuances of the patterns in this data set. Suppose the model is too finely tuned to this dataset with features like 'days since' and 'month.' In that case, it may not perform well with new data or under different avocado market conditions.

```
unique_regions = df_train['region'].unique()
unique_regions
```

```
array(['Albany', 'Atlanta', 'BaltimoreWashington', 'Boise', 'Boston',
       'BuffaloRochester', 'California', 'Charlotte', 'Chicago',
       'CincinnatiDayton', 'Columbus', 'DallasFtWorth', 'Denver',
       'Detroit', 'GrandRapids', 'GreatLakes', 'HarrisburgScranton',
       'HartfordSpringfield', 'Houston', 'Indianapolis', 'Jacksonville',
       'LasVegas', 'LosAngeles', 'Louisville', 'MiamiFtLauderdale',
       'Midsouth', 'Nashville', 'NewOrleansMobile', 'NewYork',
       'Northeast', 'NorthernNewEngland', 'Orlando', 'Philadelphia',
       'PhoenixTucson', 'Pittsburgh', 'Plains', 'Portland',
       'RaleighGreensboro', 'RichmondNorfolk', 'Roanoke', 'Sacramento',
       'SanDiego', 'SanFrancisco', 'Seattle', 'SouthCarolina',
       'SouthCentral', 'Southeast', 'Spokane', 'StLouis', 'Syracuse',
       'Tampa', 'TotalUS', 'West', 'WestTexNewMexico'], dtype=object)
```

Figure 6: all unique regions in the data set

During my exploratory data analysis, I uncovered location overlaps in 'region' such as TotalUS and individual US states (Figure 6). This overlap could skew my results as the same data might be counted more than once. This could lead to overemphasizing certain information and result in misleading conclusions about avocado prices.

This analysis and features available in the data set (Figure 1) do not explicitly include factors such as economic trends and consumer preferences. By not incorporating these economic indicators, this model may miss key drivers of price changes, additionally causing misleading predictions.

In summary, experimenting with lag features and encoding Date in my time series data was crucial. It enabled me to uncover the intricate patterns of how avocado prices change over time, analogous to a detective using various tools to discover the deeper layers of a complex case.