FORECASTING AVOCADO PRICES USING FACEBOOK PROPHET

- Prophet is open source software released by Facebook’s Core Data Science team.
- Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects.
- Prophet works best with time series that have strong seasonal effects and several seasons of historical data.

STEP #0: PROBLEM STATEMENT

###To forecast Avocado prices using Prophet Time Series

- Data represents weekly 2018 retail scan data for National retail volume (units) and price.
- Retail scan data comes directly from retailers’ cash registers based on actual retail sales of Hass avocados.
- The Average Price (of avocados) in the table reflects a per unit (per avocado) cost, even when multiple units (avocados) are sold in bags.
- The Product Lookup codes (PLU’s) in the table are only for Hass avocados. Other varieties of avocados (e.g. greenskins) are not included in this table.

Some relevant columns in the dataset:

- Date - The date of the observation
- AveragePrice - the average price of a single avocado
- type - conventional or organic
- year - the year
- Region - the city or region of the observation
- Total Volume - Total number of avocados sold
- 4046 - Total number of avocados with PLU 4046 sold
- 4225 - Total number of avocados with PLU 4225 sold
- 4770 - Total number of avocados with PLU 4770 sold

STEP #1: IMPORTING DATA

```
# import libraries
import pandas as pd # Import Pandas for data manipulation using
dataframes
import numpy as np # Import Numpy for data statistical analysis
import matplotlib.pyplot as plt # Import matplotlib for data
visualisation
import random
import seaborn as sns
from prophet import Prophet
from sklearn.metrics import mean_absolute_error

# dataframes creation for both training and testing datasets
avocado_df = pd.read_csv('avocado.csv')
```

STEP #2: EXPLORING THE DATASET

```
# Let's view the head of the training dataset
avocado_df.head()
```

```
{"summary":"{\n  \"name\": \"avocado_df\",\n  \"rows\": 18249,\n
\"fields\": [\n    {\n      \"column\": \"   \",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
15,\n        \"min\": 0,\n        \"max\": 52,\n
\"num_unique_values\": 53,\n        \"samples\": [\n          19,\n
41,\n          47\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Date\",\n      \"properties\": {\n        \"dtype\": \"object\",\n
\"num_unique_values\": 169,\n        \"samples\": [\n
\"07/05/2017\",\n          \"31/05/2015\",\n          \"17/09/2017\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"AveragePrice\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.40267655549555065,\n        \"min\": 0.44,\n        \"max\": 3.25,\n
\"num_unique_values\": 259,\n        \"samples\": [\n          0.88,\n
0.91,\n          1.07\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Total Volume \",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 3453545.3553994712,\n        \"min\":
84.56,\n        \"max\": 62505646.52,\n        \"num_unique_values\":
18237,\n        \"samples\": [\n          9783.93,\n
2604991.25,\n          277267.97\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"4046\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1264989.0817627772,\n
\"min\": 0.0,\n        \"max\": 22743616.17,\n
\"num_unique_values\": 17702,\n        \"samples\": [\n
97.29,\n          97139.95,\n          6339.49\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"4225\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1204120.4011350507,\n
\"min\": 0.0,\n        \"max\": 20470572.61,\n
\"num_unique_values\": 18103,\n        \"samples\": [\n
11096.86,\n          22648.55,\n          87497.9\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"4770\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 107464.06843537073,\n
\"min\": 0.0,\n        \"max\": 2546439.11,\n
\"num_unique_values\": 12071,\n        \"samples\": [\n
74.0,\n          1875.51,\n          202.18\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Total Bags\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
986242.3992164118,\n        \"min\": 0.0,\n        \"max\":
19373134.37,\n        \"num_unique_values\": 18097,\n
\"samples\": [\n          63800.09,\n          12938.12,\n
912891.54\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Small Bags\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 746178.5149617889,\n        \"min\":
0.0,\n        \"max\": 13384586.8,\n        \"num_unique_values\":
17321,\n        \"samples\": [\n          41688.75,\n
118293.26,\n          93488.63\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Large Bags\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 243965.96454740883,\n
\"min\": 0.0,\n        \"max\": 5719096.61,\n
\"num_unique_values\": 15082,\n        \"samples\": [\n
5155.3,\n          43900.93,\n          163807.71\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"XLarge Bags\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
17692.894651916486,\n        \"min\": 0.0,\n        \"max\":
551693.65,\n        \"num_unique_values\": 5588,\n        \"samples\":
[\n          527.2,\n          1290.39,\n          7249.07\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"type       \",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"organic      \",\n          \"conventional \"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"year\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 2015,\n        \"max\": 2018,\n
\"num_unique_values\": 4,\n        \"samples\": [\n          2016,\n
2018\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"region\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 54,\n
\"samples\": [\n          \"Indianapolis\",\n          \"Syracuse\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe","variable_name":"avocado_df"}
```

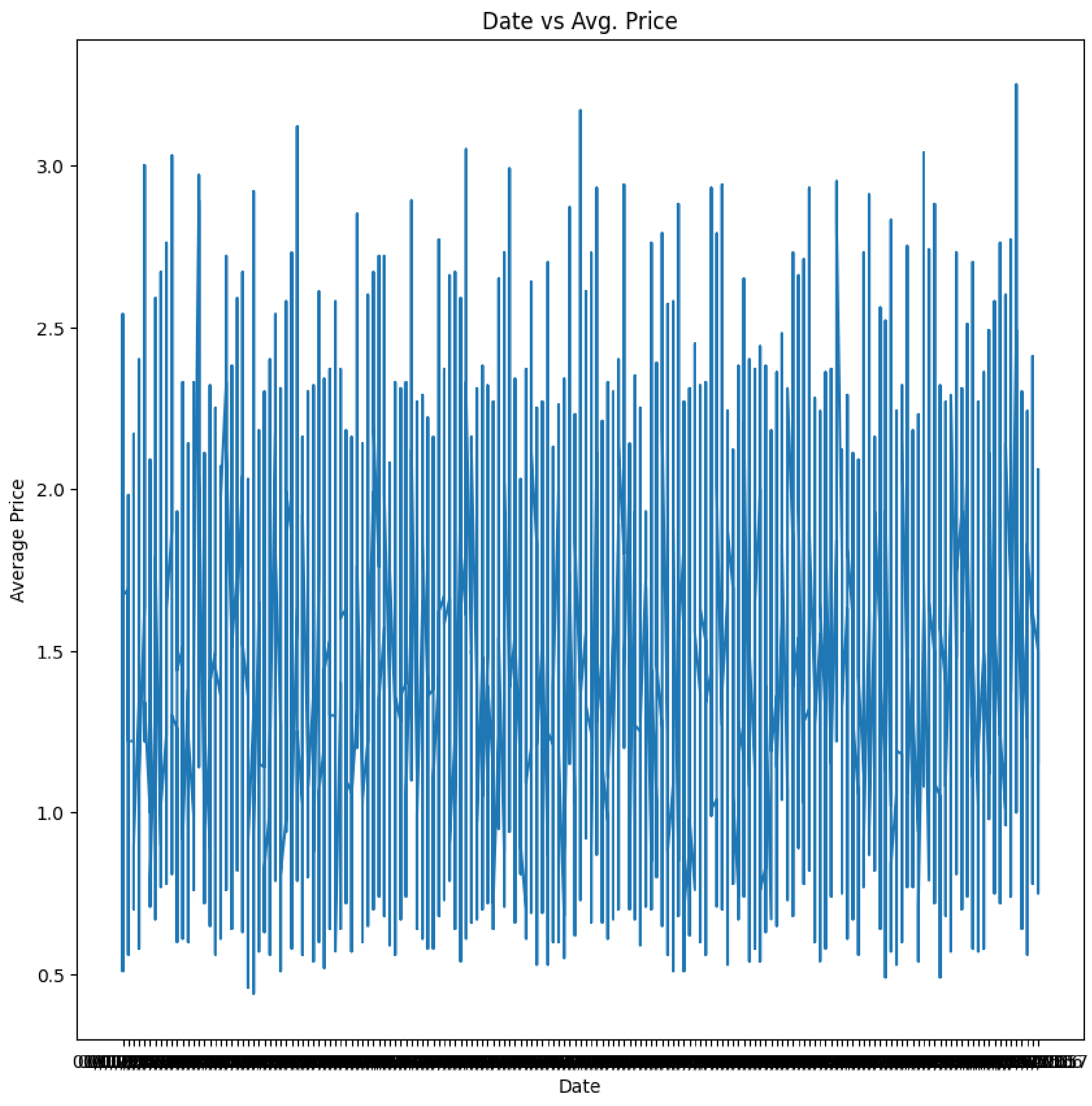
```python
# Let's view the last elements in the training dataset
avocado_df.tail()
```

{"repr_error":"0","type":"dataframe"}

```python
avocado_df = avocado_df.sort_values("Date")
```

```python
plt.figure(figsize=(10,10))
plt.xlabel("Date")
plt.ylabel("Average Price")
plt.title("Date vs Avg. Price")
plt.plot(avocado_df['Date'], avocado_df['AveragePrice'])
```

```
[<matplotlib.lines.Line2D at 0x7b79a186d5a0>]
```



Inference:

- The average price exhibits significant volatility and fluctuations over time, with numerous peaks and troughs observed throughout the plotted period.
- The overall trend seems to indicate cyclical or seasonal patterns, with the average price rising and falling within a certain range repeatedly.
- There are periods where the average price remains relatively stable, with minimal fluctuations, interspersed with periods of higher volatility and larger price swings.

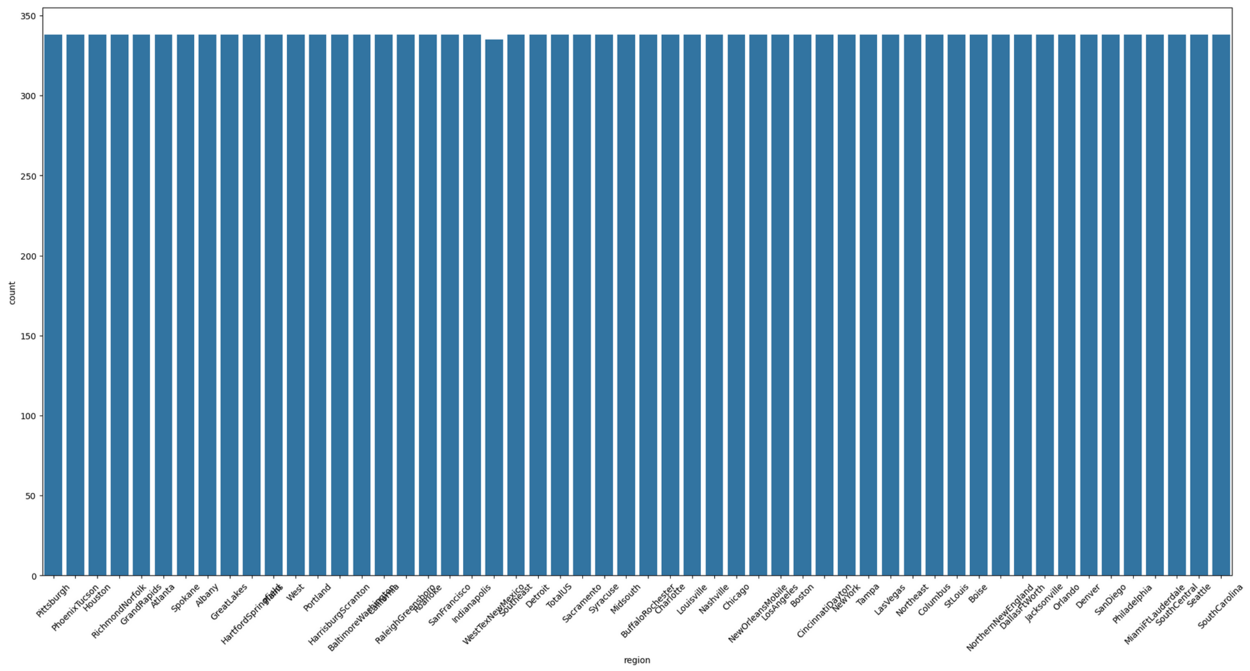
```
avocado_df
```

```
{"summary":"{\n  \"name\": \"avocado_df\",\n  \"rows\": 18249,\n
\"fields\": [\n    {\n      \"column\": \"   \",\n
```

```
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
15,\n        \"min\": 0,\n        \"max\": 52,\n
\"num_unique_values\": 53,\n        \"samples\": [\n          25,\n
40,\n          35\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Date\",\n      \"properties\": {\n        \"dtype\": \"object\",\n
\"num_unique_values\": 169,\n        \"samples\": [\n
\"25/10/2015\",\n          \"06/08/2017\",\n          \"22/03/2015\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"AveragePrice\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.40267655549555065,\n        \"min\": 0.44,\n        \"max\": 3.25,\n
\"num_unique_values\": 259,\n        \"samples\": [\n          1.58,\n
2.0,\n          0.88\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Total Volume \",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 3453545.3553994712,\n        \"min\":
84.56,\n        \"max\": 62505646.52,\n        \"num_unique_values\":
18237,\n        \"samples\": [\n          10909.04,\n
192774.07,\n          1201.07\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"4046\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1264989.0817627772,\n        \"min\":
0.0,\n        \"max\": 22743616.17,\n        \"num_unique_values\":
17702,\n        \"samples\": [\n          15584.56,\n
113516.0,\n          48624.65\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"4225\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1204120.4011350507,\n        \"min\":
0.0,\n        \"max\": 20470572.61,\n        \"num_unique_values\":
18103,\n        \"samples\": [\n          40377.82,\n          70.66,\
n          60.1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"4770\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 107464.06843537072,\n        \"min\": 0.0,\n        \"max\":
2546439.11,\n        \"num_unique_values\": 12071,\n
\"samples\": [\n          499.98,\n          2526.06,\n
143202.95\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Total Bags\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 986242.3992164118,\n        \"min\":
0.0,\n        \"max\": 19373134.37,\n        \"num_unique_values\":
18097,\n        \"samples\": [\n          277850.42,\n
54443.04,\n          567.68\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Small Bags\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 746178.5149617889,\n
\"min\": 0.0,\n        \"max\": 13384586.8,\n
\"num_unique_values\": 17321,\n        \"samples\": [\n
8086.79,\n          4433.33,\n          1226559.09\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Large Bags\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
243965.96454740883,\n        \"min\": 0.0,\n        \"max\":
5719096.61,\n        \"num_unique_values\": 15082,\n
\"samples\": [\n          689.64,\n          63758.11,\n
55171.08\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"XLarge Bags\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 17692.894651916486,\n        \"min\":
0.0,\n        \"max\": 551693.65,\n        \"num_unique_values\":
5588,\n        \"samples\": [\n          7328.13,\n          655.99,\n
33.55\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"type       \",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"organic      \",\n          \"conventional \"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"year\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 2015,\n        \"max\": 2018,\n
\"num_unique_values\": 4,\n        \"samples\": [\n          2015,\n
2018\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"region\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 54,\n
\"samples\": [\n          \"Indianapolis\",\n
\"Philadelphia\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"avocado_df"}
```

```
# Bar Chart to indicate the number of regions
plt.figure(figsize=[25,12])
sns.countplot(x = 'region', data = avocado_df)
plt.xticks(rotation = 45)
```

```
([0,
  1,
  2,
  3,
  4,
  5,
  6,
  7,
  8,
  9,
  10,
  11,
  12,
  13,
  14,
  15,
  16,
  17,
  18,
  19,
  20,
  21,
  22,
  23,
  24,
  25,
  26,
  27,
  28,
  29,
  30,
  31,
  32,
  33,
  34,
  35,
  36,
  37,
  38,
  39,
  40,
  41,
  42,
  43,
  44,
  45,
  46,
  47,
  48,
  49,
  50,
  51,
  52,
  53],
 [Text(0, 0, 'Pittsburgh'),
  Text(1, 0, 'PhoenixTucson'),
  Text(2, 0, 'Houston'),
  Text(3, 0, 'RichmondNorfolk'),
  Text(4, 0, 'GrandRapids'),
  Text(5, 0, 'Atlanta'),
  Text(6, 0, 'Spokane'),
  Text(7, 0, 'Albany'),
  Text(8, 0, 'GreatLakes'),
  Text(9, 0, 'HartfordSpringfield'),
  Text(10, 0, 'Plains'),
  Text(11, 0, 'West'),
  Text(12, 0, 'Portland'),
  Text(13, 0, 'HarrisburgScranton'),
  Text(14, 0, 'BaltimoreWashington'),
  Text(15, 0, 'California'),
  Text(16, 0, 'RaleighGreensboro'),
  Text(17, 0, 'Roanoke'),
  Text(18, 0, 'SanFrancisco'),
  Text(19, 0, 'Indianapolis'),
  Text(20, 0, 'WestTexNewMexico'),
  Text(21, 0, 'Southeast'),
  Text(22, 0, 'Detroit'),
  Text(23, 0, 'TotalUS'),
  Text(24, 0, 'Sacramento'),
  Text(25, 0, 'Syracuse'),
  Text(26, 0, 'Midsouth'),
  Text(27, 0, 'BuffaloRochester'),
  Text(28, 0, 'Charlotte'),
  Text(29, 0, 'Louisville'),
  Text(30, 0, 'Nashville'),
  Text(31, 0, 'Chicago'),
  Text(32, 0, 'NewOrleansMobile'),
  Text(33, 0, 'LosAngeles'),
  Text(34, 0, 'Boston'),
  Text(35, 0, 'CincinnatiDayton'),
  Text(36, 0, 'NewYork'),
  Text(37, 0, 'Tampa'),
  Text(38, 0, 'LasVegas'),
  Text(39, 0, 'Northeast'),
  Text(40, 0, 'Columbus'),
  Text(41, 0, 'StLouis'),
  Text(42, 0, 'Boise'),
  Text(43, 0, 'NorthernNewEngland'),
  Text(44, 0, 'DallasFtWorth'),
  Text(45, 0, 'Jacksonville'),
  Text(46, 0, 'Orlando'),
  Text(47, 0, 'Denver'),
  Text(48, 0, 'SanDiego'),
  Text(49, 0, 'Philadelphia'),
  Text(50, 0, 'MiamiFtLauderdale'),
  Text(51, 0, 'SouthCentral'),
  Text(52, 0, 'Seattle'),
  Text(53, 0, 'SouthCarolina')])
```

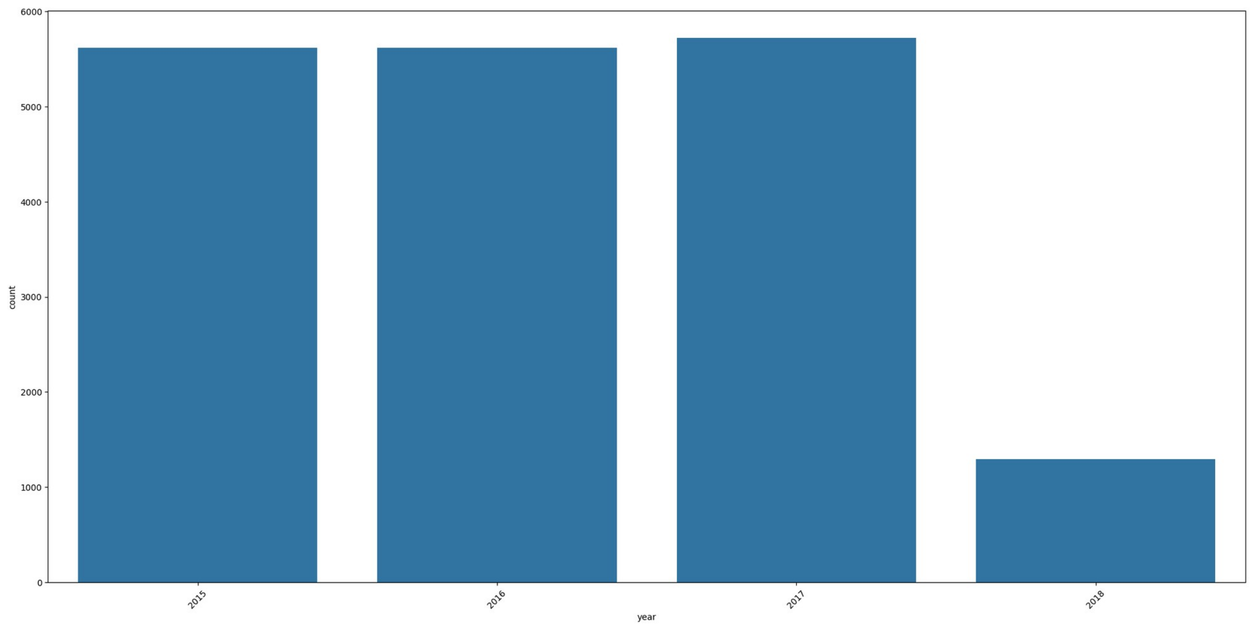


Inference:

- The graph shows constant values across all regions, as indicated by the uniform height of the bars.
- With only a single constant value shown, it is difficult to draw meaningful insights or comparisons between regions based solely on this graph.

```
# Bar Chart to indicate the year
plt.figure(figsize=[25,12])
sns.countplot(x = 'year', data = avocado_df)
plt.xticks(rotation = 45)
```

```
([0, 1, 2, 3],
 [Text(0, 0, '2015'),
  Text(1, 0, '2016'),
  Text(2, 0, '2017'),
  Text(3, 0, '2018')])
```



Inference:

- The height of the bars increases progressively from 2015 to 2017, suggesting an upward trend or growth in the values over those years.
- Potential Peak and Decline: While the value for 2017 is the highest, the bar for 2018 is noticeably shorter, implying a potential peak in 2017 followed by a decline or decrease in the value for 2018.

```
avocado_prophet_df = avocado_df[['Date', 'AveragePrice']]

avocado_prophet_df
```

```
{"summary":"{\n  \"name\": \"avocado_prophet_df\",\n  \"rows\":
18249,\n  \"fields\": [\n    {\n      \"column\": \"Date\",\n
\"properties\": {\n        \"dtype\": \"object\",\n
\"num_unique_values\": 169,\n        \"samples\": [\n
\"25/10/2015\",\n          \"06/08/2017\",\n          \"22/03/2015\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"AveragePrice\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.40267655549555065,\n        \"min\": 0.44,\n        \"max\": 3.25,\n
\"num_unique_values\": 259,\n        \"samples\": [\n          1.58,\n
2.0,\n          0.88\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"avocado_prophet_df"}
```

Here, we've created a new database with only relevant features for our database, i.e., Date and Price

STEP 3: MAKE PREDICTIONS

```
avocado_prophet_df = avocado_prophet_df.rename(columns={'Date':'ds',
'AveragePrice':'y'})

avocado_prophet_df
```

```
{"summary":"{\n  \"name\": \"avocado_prophet_df\",\n  \"rows\":
18249,\n  \"fields\": [\n    {\n      \"column\": \"ds\",\n
\"properties\": {\n        \"dtype\": \"object\",\n
\"num_unique_values\": 169,\n        \"samples\": [\n
\"25/10/2015\",\n          \"06/08/2017\",\n          \"22/03/2015\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"y\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.40267655549555065,\n
\"min\": 0.44,\n        \"max\": 3.25,\n        \"num_unique_values\":
259,\n        \"samples\": [\n          1.58,\n          2.0,\n
0.88\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"avocado_prophet_df"}
```

In Prophet Time Series, it is necessary to only have two attritbutes 'ds' and 'y'. 'ds' represents the date and 'y' represents the target variable or dependent variable. No other column names are accepted other than these.

```
# Creating an object and fitting the model
m = Prophet()
m.fit(avocado_prophet_df)
```

```
/usr/local/lib/python3.10/dist-packages/prophet/forecaster.py:1133:
UserWarning: Parsing dates in DD/MM/YYYY format when dayfirst=False
(the default) was specified. This may lead to inconsistently parsed
dates! Specify a format to ensure consistent parsing.
  self.history_dates =
pd.to_datetime(pd.Series(history['ds'].unique(),
name='ds')).sort_values()
/usr/local/lib/python3.10/dist-packages/prophet/forecaster.py:287:
UserWarning: Parsing dates in DD/MM/YYYY format when dayfirst=False
(the default) was specified. This may lead to inconsistently parsed
dates! Specify a format to ensure consistent parsing.
  df['ds'] = pd.to_datetime(df['ds'])
INFO:prophet:Disabling daily seasonality. Run prophet with
daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmpp_5m5kt4/42jxu4lg.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpp_5m5kt4/1vqwmm26.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-
packages/prophet/stan_model/prophet_model.bin', 'random',
'seed=17391', 'data', 'file=/tmp/tmpp_5m5kt4/42jxu4lg.json',
'init=/tmp/tmpp_5m5kt4/1vqwmm26.json', 'output',
'file=/tmp/tmpp_5m5kt4/prophet_model4546utd0/prophet_model-
20240403202922.csv', 'method=optimize', 'algorithm=lbfgs',
'iter=10000']
20:29:22 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
20:29:25 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing

<prophet.forecaster.Prophet at 0x7b79a0ce8430>

# Forcasting into the future
future = m.make_future_dataframe(periods=365)
forecast = m.predict(future)
```

- The make_future_dataframe function lets you specify the frequency and number of periods you would like to forecast into the future. By default, the frequency is set to days. Since we are using daily periodicity data in this example, we will leave freq at it's default and set the periods argument to 365, indicating that we would like to forecast 365 days into the future.
- We have then used the predict method to make predictions for each row in the future dataframe.

```
forecast
```

```
{"summary":"{\n  \"name\": \"forecast\",\n  \"rows\": 534,\n
\"fields\": [\n    {\n      \"column\": \"ds\",\n      \"properties\":
{\n        \"dtype\": \"date\",\n        \"min\": \"2015-01-02
00:00:00\",\n        \"max\": \"2019-11-03 00:00:00\",\n
\"num_unique_values\": 534,\n        \"samples\": [\n          \"2018-
12-27 00:00:00\",\n          \"2017-07-16 00:00:00\",\n
\"2017-11-19 00:00:00\"\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"trend\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.16279622324166915,\n        \"min\":
0.9598357992270589,\n        \"max\": 1.5479182612776778,\n
\"num_unique_values\": 534,\n        \"samples\": [\n
1.1842451804512826,\n          1.5438476255865397,\n
1.4750393303977853\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"yhat_lower\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.17248502476385477,\n        \"min\":
0.4214658677462716,\n        \"max\": 1.2246321779677691,\n
\"num_unique_values\": 534,\n        \"samples\": [\n
0.6843519821408584,\n          1.10898654214329,\n
1.0150223331998163\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"yhat_upper\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.1696430524233491,\n        \"min\":
1.3978154922794817,\n        \"max\": 2.1870202583596092,\n
```

```
\"num_unique_values\": 534,\n        \"samples\": [\n
1.7080618288867755,\n          2.0812413498301305,\n
1.9841543967033137\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"trend_lower\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.18435655319363412,\n        \"min\":
0.8757819669622361,\n        \"max\": 1.5479182612776778,\n
\"num_unique_values\": 534,\n        \"samples\": [\n
1.1809169287550167,\n          1.5438476255865397,\n
1.4750393303977853\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"trend_upper\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.14199371706951722,\n        \"min\":
1.045568691789347,\n        \"max\": 1.5479182612776778,\n
\"num_unique_values\": 534,\n        \"samples\": [\n
1.188197091298994,\n          1.5438476255865397,\n
1.4750393303977853\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"additive_terms\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.07564691084256846,\n        \"min\": -
0.157252965775505,\n        \"max\": 0.223058233978633,\n
\"num_unique_values\": 534,\n        \"samples\": [\n
0.006849130057972991,\n          0.041955306737958005,\n
0.02343892879835191\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"additive_terms_lower\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.07564691084256846,\n        \"min\": -
0.157252965775505,\n        \"max\": 0.223058233978633,\n
\"num_unique_values\": 534,\n        \"samples\": [\n
0.006849130057972991,\n          0.041955306737958005,\n
0.02343892879835191\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"additive_terms_upper\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.07564691084256846,\n        \"min\": -
0.157252965775505,\n        \"max\": 0.223058233978633,\n
\"num_unique_values\": 534,\n        \"samples\": [\n
0.006849130057972991,\n          0.041955306737958005,\n
0.02343892879835191\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"weekly\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.04263195168131799,\n        \"min\": -0.04338291345920475,\
n        \"max\": 0.09500097750001972,\n        \"num_unique_values\":
534,\n        \"samples\": [\n          0.09500097749999672,\n
0.014704700716266885,\n          0.014704700716597258\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"weekly_lower\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.04263195168131799,\n        \"min\": -0.04338291345920475,\n
\"max\": 0.09500097750001972,\n        \"num_unique_values\": 534,\n
```

```
\"samples\": [\n          0.09500097749999672,\n
0.014704700716266885,\n          0.014704700716597258\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"weekly_upper\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.04263195168131799,\n        \"min\": -0.04338291345920475,\n
\"max\": 0.09500097750001972,\n        \"num_unique_values\": 534,\n
\"samples\": [\n          0.09500097749999672,\n
0.014704700716266885,\n          0.014704700716597258\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"yearly\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
0.0634249368034316,\n        \"min\": -0.11397304602783717,\n
\"max\": 0.13071388254509142,\n        \"num_unique_values\": 534,\n
\"samples\": [\n          -0.08815184744202374,\n
0.027250606021691125,\n          0.008734228081754654\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"yearly_lower\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.0634249368034316,\n        \"min\": -0.11397304602783717,\n
\"max\": 0.13071388254509142,\n        \"num_unique_values\": 534,\n
\"samples\": [\n          -0.08815184744202374,\n
0.027250606021691125,\n          0.008734228081754654\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"yearly_upper\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.0634249368034316,\n        \"min\": -0.11397304602783717,\n
\"max\": 0.13071388254509142,\n        \"num_unique_values\": 534,\n
\"samples\": [\n          -0.08815184744202374,\n
0.027250606021691125,\n          0.008734228081754654\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"multiplicative_terms\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.0,\n        \"min\": 0.0,\n        \"max\": 0.0,\n
\"num_unique_values\": 1,\n        \"samples\": [\n          0.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"multiplicative_terms_lower\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.0,\n        \"min\": 0.0,\n        \"max\": 0.0,\n
\"num_unique_values\": 1,\n        \"samples\": [\n          0.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"multiplicative_terms_upper\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.0,\n        \"min\": 0.0,\n        \"max\": 0.0,\n
\"num_unique_values\": 1,\n        \"samples\": [\n          0.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"yhat\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
0.1705517544734876,\n        \"min\": 0.9295797823115649,\n
```

```
\"max\": 1.7186621834471192,\n        \"num_unique_values\": 534,\n
\"samples\": [\n          1.1910943105092555\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"forecast"}
```

Prophet is an additive model consisting of four main components:

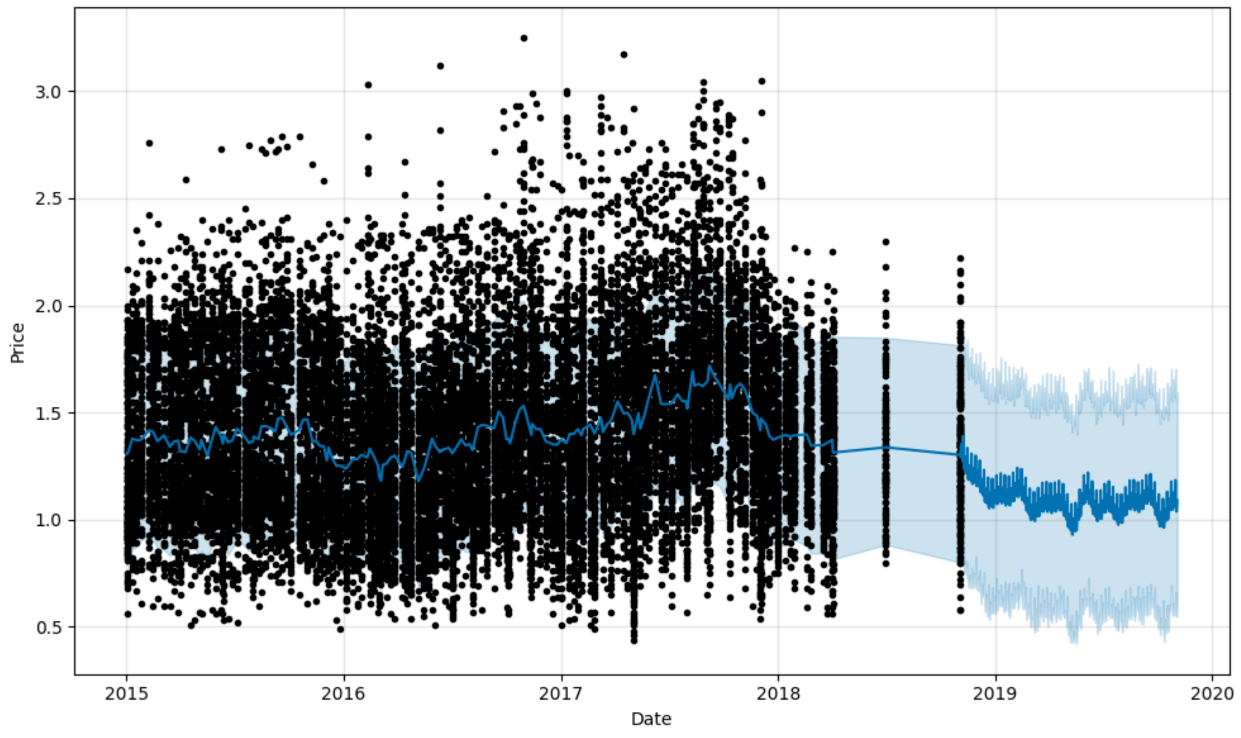
Here,

- g(t) refers to trend (changes over a long period of time)
- s(t) refers to seasonality (periodic or short-term changes)
- h(t) refers to effects of holidays to the forecast
- e(t) refers to the unconditional changes that is specific to a business or a person or a circumstance. It is also called the error term.
- y(t) is the forecast.

This new dataframe contains yhat the predicted values of future dates as well as uncertainty intervals and components for the forecast.

Using time as a regressor, Prophet is trying to fit several linear and non linear functions of time as components. Modeling seasonality as an additive component is the same approach taken by exponential smoothing in Holt-Winters technique.

```
figure = m.plot(forecast, xlabel='Date', ylabel='Price')
```



Inferences:

1. Two Data Patterns: There are two distinct patterns visible in the data. The first pattern, represented by the dense black scatter points, seems to be the historical data used for training the forecasting model. The second pattern, shown in blue, appears to be the forecast generated by the Prophet model.

2. High Volatility in Historical Data: The historical data (black scatter points) exhibits a high degree of volatility and fluctuations, suggesting that the underlying time series being modeled has significant variability.

3. Seasonal Pattern: The forecast (blue line) shows a distinct seasonal or cyclical pattern, with regular peaks and troughs repeating over time. This implies that the Prophet model has captured and projected the seasonal components present in the historical data.

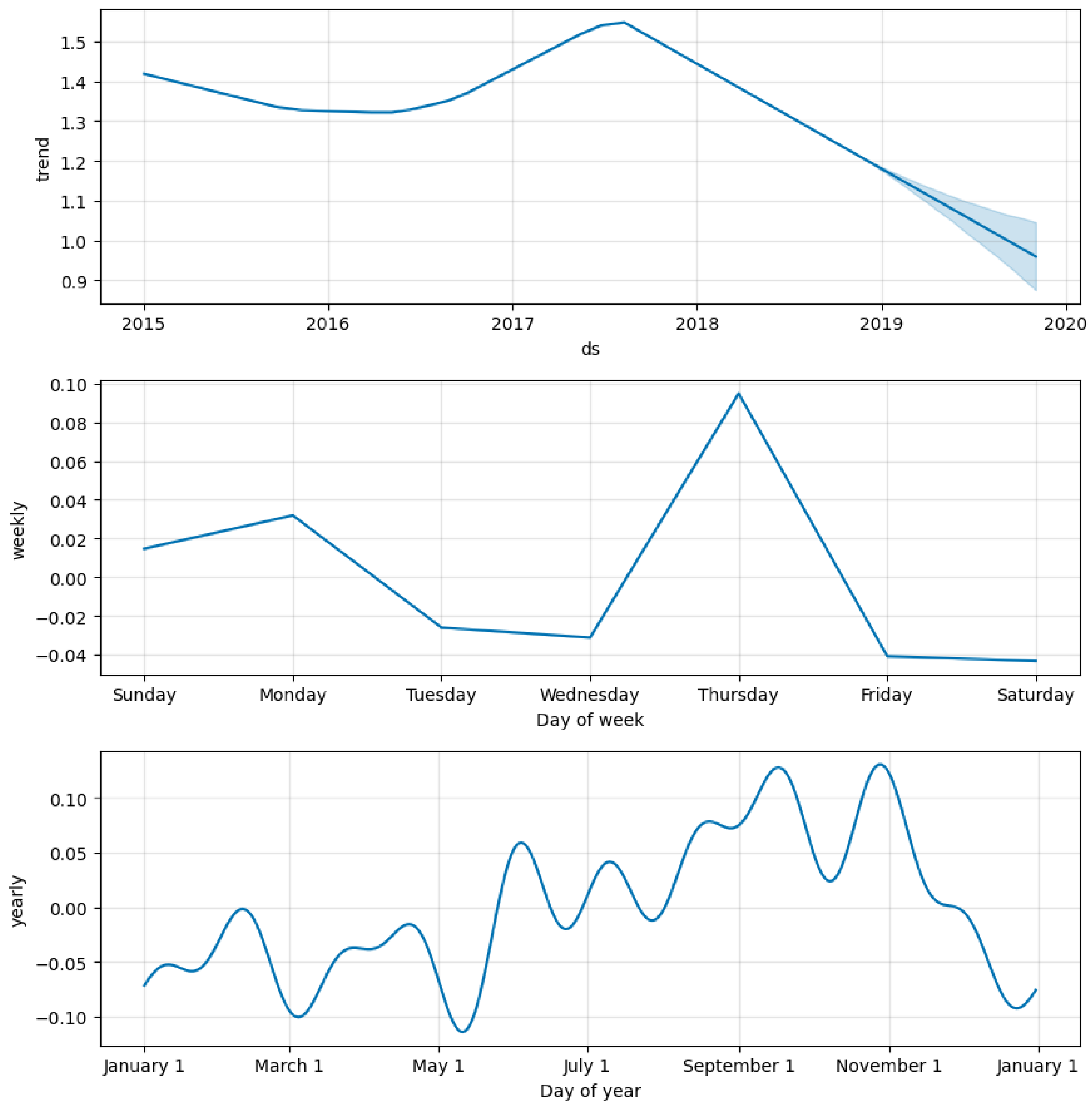
4. Increasing Trend: While the forecast fluctuates seasonally, there seems to be an overall increasing trend in the predicted values, suggesting that the time series being modeled has an underlying upward trajectory.

5. Uncertainty Intervals: The vertical black lines accompanying the forecast represent the uncertainty intervals or confidence intervals associated with the predictions. These intervals appear to be relatively narrow, indicating a higher level of confidence in the forecast.

6. Outliers in Forecast: There are a few isolated spikes or outliers in the forecast, which could represent exceptional events or anomalies that the Prophet model has identified and incorporated into the predictions.

Overall, the Prophet forecast graph suggests that the underlying time series exhibits significant volatility and seasonality, with an increasing trend over time. The Prophet model has captured these patterns and generated a forecast that incorporates the seasonal fluctuations while projecting an overall upward trajectory, accounting for potential outliers or exceptional events.

```
figure3 = m.plot_components(forecast)
```



Inferences:

Trend Inference:

1. Overall Trend: The upper plot shows the overall trend of the time series data. It exhibits a distinct yearly cycle with a peak around mid-year (likely summer months) and a trough towards the end of the year (likely winter months). This pattern suggests a seasonal or cyclical behavior in the data.

2. Decreasing Trend: While the seasonal pattern is evident, there is also an apparent decreasing trend in the overall level of the time series from 2015 to 2020. This

indicates that the underlying phenomenon being modeled is experiencing a gradual decline over the years.

Weekly Inference:

1. Weekly Seasonality: The middle plot displays the weekly seasonal pattern in the data. It shows a clear weekly cycle, with a peak around mid-week (Wednesday/Thursday) and lower values on weekends. This pattern suggests that the time series is influenced by weekly patterns, potentially related to human activities or business cycles.

2. Weekend Dip: The plot shows a distinct dip in values on Sundays, indicating a consistent lower level of activity or demand on that day of the week compared to other days.

Yearly Inference:

1. Yearly Seasonality: The lower plot reveals the yearly seasonal pattern in the data. It shows a distinct yearly cycle, with peaks around mid-year (likely summer months) and troughs towards the end of the year (likely winter months). This pattern aligns with the overall trend observed in the upper plot, confirming the presence of a strong yearly seasonality.

2. Irregular Patterns: While the yearly seasonal pattern is evident, there are also some irregular fluctuations and deviations from the smooth curve, suggesting the presence of additional factors or events influencing the time series beyond the regular seasonal patterns.

Overall, the Prophet forecast graph indicates that the time series being modeled exhibits strong yearly and weekly seasonal patterns, with peaks in summer and mid-week, respectively. Additionally, there is an overall decreasing trend in the data from 2015 to 2020, suggesting a gradual decline in the underlying phenomenon. The presence of irregular fluctuations in the lower plot suggests the influence of additional factors beyond the regular seasonal patterns.

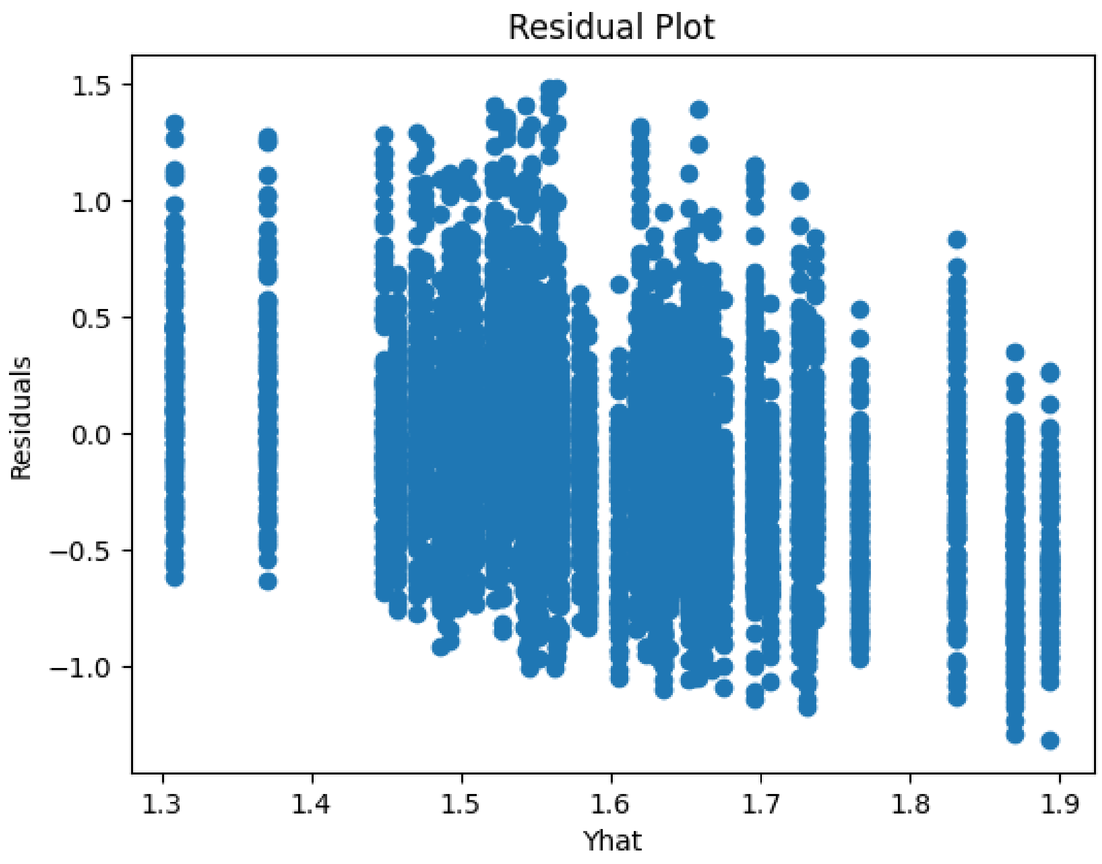
PERFORMANCE EVALUATION

```
from prophet.diagnostics import cross_validation
df_cv = cross_validation(m, initial='730 days', period='180 days',
horizon = '365 days')

df_cv.head()


INFO:prophet:Making 2 forecasts with cutoffs between 2017-05-07
00:00:00 and 2017-11-03 00:00:00
```

{"model_id":"481224b59dfe4472bdb04ab4af19d2fc","version_major":2,"version_minor":0}

```
DEBUG:cmdstanpy:input tempfile: /tmp/tmpp_5m5kt4/67akasqq.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpp_5m5kt4/fn82_wjx.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-
packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=6038',
'data', 'file=/tmp/tmpp_5m5kt4/67akasqq.json',
'init=/tmp/tmpp_5m5kt4/fn82_wjx.json', 'output',
'file=/tmp/tmpp_5m5kt4/prophet_model_s_da3j3/prophet_model-
20240403202929.csv', 'method=optimize', 'algorithm=lbfgs',
'iter=10000']
20:29:29 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
20:29:31 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmpp_5m5kt4/35m0gnp5.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpp_5m5kt4/jagul9zc.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-
packages/prophet/stan_model/prophet_model.bin', 'random',
'seed=89799', 'data', 'file=/tmp/tmpp_5m5kt4/35m0gnp5.json',
'init=/tmp/tmpp_5m5kt4/jagul9zc.json', 'output',
'file=/tmp/tmpp_5m5kt4/prophet_modell363ggy2/prophet_model-
20240403202935.csv', 'method=optimize', 'algorithm=lbfgs',
'iter=10000']
20:29:35 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
20:29:40 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
```

```
{"summary":"{\n  \"name\": \"df_cv\",\n  \"rows\": 6910,\n
\"fields\": [\n    {\n      \"column\": \"ds\",\n      \"properties\":
{\n        \"dtype\": \"date\",\n        \"min\": \"2017-05-11
00:00:00\",\n        \"max\": \"2018-11-03 00:00:00\",\n
\"num_unique_values\": 46,\n        \"samples\": [\n          \"2018-
03-18 00:00:00\",\n          \"2017-11-06 00:00:00\",\n
\"2017-11-19 00:00:00\"\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"yhat\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.10693636703093781,\n        \"min\":
1.308780838620803,\n        \"max\": 1.893856901846651,\n
\"num_unique_values\": 66,\n        \"samples\": [\n
1.6284284182816453,\n          1.6553877754970514,\n
1.5064427877263997\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"yhat_lower\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.10757282197502192,\n        \"min\":
0.7718797419811577,\n        \"max\": 1.435419052810479,\n
\"num_unique_values\": 6910,\n        \"samples\": [\n
1.1469964505534878,\n          1.0288074788415649,\n
1.155493901636949\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"yhat_upper\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.1105352037192861,\n        \"min\":
1.7398123410886093,\n        \"max\": 2.4570802510257903,\n
\"num_unique_values\": 6910,\n        \"samples\": [\n
2.1358737794796667,\n          2.0081549647009242,\n
2.114472195094222\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"y\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.39021082815529884,\n        \"min\": 0.54,\n
\"max\": 3.05,\n        \"num_unique_values\": 237,\n
\"samples\": [\n          0.74,\n          1.49,\n          2.92\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"cutoff\",\n      \"properties\":
{\n        \"dtype\": \"date\",\n        \"min\": \"2017-05-07
00:00:00\",\n        \"max\": \"2017-11-03 00:00:00\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n          \"2017-
11-03 00:00:00\",\n          \"2017-05-07 00:00:00\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"df_cv"}

# Residual Graph
residuals = df_cv['y'] - df_cv['yhat']
plt.scatter(df_cv['yhat'],residuals)
plt.xlabel("Yhat")
plt.ylabel("Residuals")
plt.title("Residual Plot")
plt.show()
```



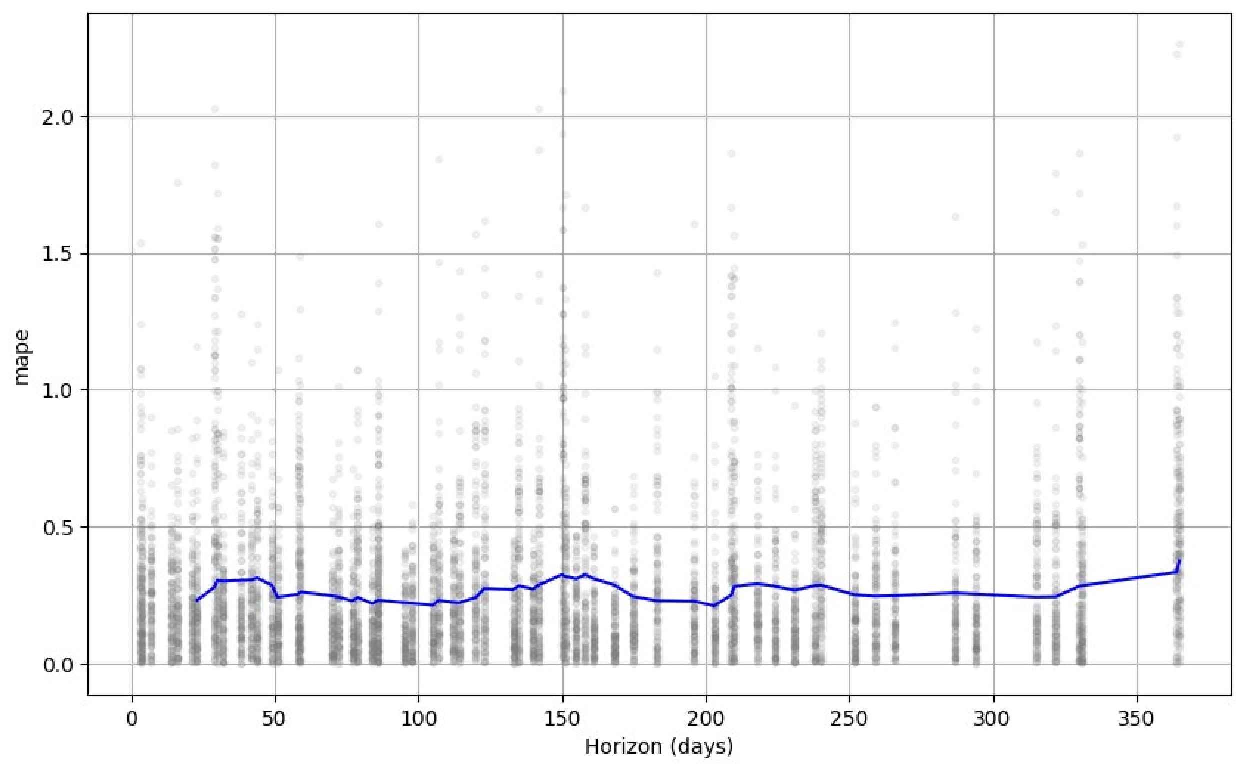
Inference:

From this graph, we can see that our residual values and randomly scatter around 0. This shows that it is a good model.

```
from prophet.diagnostics import performance_metrics
df_p = performance_metrics(df_cv)
df_p.head()
```

```
{"summary":"{\n  \"name\": \"df_p\",\n  \"rows\": 57,\n  \"fields\":
[\n    {\n      \"column\": \"horizon\",\n      \"properties\": {\n
\"dtype\": \"timedelta64[ns]\",\n        \"num_unique_values\": 57,\n
\"samples\": [\n          \"23 days 00:00:00\",\n          \"42 days
00:00:00\",\n          \"151 days 00:00:00\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"mse\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.03577866363613648,\n
\"min\": 0.11551674559992195,\n        \"max\": 0.24631443290258861,\n
\"num_unique_values\": 57,\n        \"samples\": [\n
0.17498239016652517,\n          0.20176034251066194,\n
0.2334805785885652\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"rmse\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.042492638310020155,\n        \"min\": 0.3398775450069068,\n
\"max\": 0.49630074844048805,\n        \"num_unique_values\": 57,\n
\"samples\": [\n          0.4183089649607395,\n
0.4491774065006631,\n          0.483198280821202\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"mae\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.0339823370045862,\n
\"min\": 0.27925321671684566,\n        \"max\": 0.3998748092867944,\n
\"num_unique_values\": 57,\n        \"samples\": [\n
0.3319680477699488,\n          0.3667641318564474,\n
0.38625959418152794\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"mape\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.035551631481986604,\n        \"min\": 0.20889066807563414,\
n        \"max\": 0.37288563626323457,\n        \"num_unique_values\":
57,\n        \"samples\": [\n          0.22985292396639867,\n
0.3046745156205977,\n          0.3171906867359756\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"mdape\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\":
0.023038148699319222,\n        \"min\": 0.14678057625467986,\n
\"max\": 0.28463802101304925,\n        \"num_unique_values\": 54,\n
\"samples\": [\n          0.16467113851912402,\n
0.1664375369903184,\n          0.1632265927734714\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"smape\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\":
0.02143545554004058,\n        \"min\": 0.189982536571891,\n
\"max\": 0.2841756167768809,\n        \"num_unique_values\": 57,\n
\"samples\": [\n          0.21808655478606162,\n
0.25031562757408565,\n          0.2574390712103908\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"coverage\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
0.050945894144592246,\n        \"min\": 0.6600203676904111,\n
\"max\": 0.8636972718014686,\n        \"num_unique_values\": 57,\n
\"samples\": [\n          0.7954789087205874,\n
0.746207857640564,\n          0.6759527255185721\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"df_p"}
```

```python
from prophet.plot import plot_cross_validation_metric
fig = plot_cross_validation_metric(df_cv, metric='mape')
```



Inference:

- From this graph we can see that our model has a MAPE ranging from 0.1 to 0.3 over a span of 365 days which makes this a fairly decent model given the resources used to make this model.

REGION SPECIFIC FORECASTING

```
# dataframes creation for both training and testing datasets
avocado_df = pd.read_csv('avocado.csv')

avocado_df
```

{"summary":"{\n \"name\": \"avocado_df\",\n \"rows\": 18249,\n \"fields\": [\n   {\n   \"column\": \"   \",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 15,\n   \"min\": 0,\n   \"max\": 52,\n   \"num_unique_values\": 53,\n   \"samples\": [\n   19,\n   41,\n   47\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Date\",\n   \"properties\": {\n   \"dtype\": \"object\",\n   \"num_unique_values\": 169,\n   \"samples\": [\n   \"07/05/2017\",\n   \"31/05/2015\",\n   \"17/09/2017\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"AveragePrice\",\n

```
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.40267655549555065,\n        \"min\": 0.44,\n        \"max\": 3.25,\n
\"num_unique_values\": 259,\n        \"samples\": [\n          0.88,\n
0.91,\n          1.07\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Total Volume \",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 3453545.3553994712,\n        \"min\":
84.56,\n        \"max\": 62505646.52,\n        \"num_unique_values\":
18237,\n        \"samples\": [\n          9783.93,\n
2604991.25,\n          277267.97\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"4046\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1264989.0817627772,\n
\"min\": 0.0,\n        \"max\": 22743616.17,\n
\"num_unique_values\": 17702,\n        \"samples\": [\n
97.29,\n          97139.95,\n          6339.49\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"4225\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1204120.4011350507,\n
\"min\": 0.0,\n        \"max\": 20470572.61,\n
\"num_unique_values\": 18103,\n        \"samples\": [\n
11096.86,\n          22648.55,\n          87497.9\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"4770\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 107464.06843537073,\n
\"min\": 0.0,\n        \"max\": 2546439.11,\n
\"num_unique_values\": 12071,\n        \"samples\": [\n
74.0,\n          1875.51,\n          202.18\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Total Bags\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
986242.3992164118,\n        \"min\": 0.0,\n        \"max\":
19373134.37,\n        \"num_unique_values\": 18097,\n
\"samples\": [\n          63800.09,\n          12938.12,\n
912891.54\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Small Bags\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 746178.5149617889,\n        \"min\":
0.0,\n        \"max\": 13384586.8,\n        \"num_unique_values\":
17321,\n        \"samples\": [\n          41688.75,\n
118293.26,\n          93488.63\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Large Bags\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 243965.96454740883,\n
\"min\": 0.0,\n        \"max\": 5719096.61,\n
\"num_unique_values\": 15082,\n        \"samples\": [\n
5155.3,\n          43900.93,\n          163807.71\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"XLarge Bags\",\n
```

```
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
17692.894651916486,\n        \"min\": 0.0,\n        \"max\":
551693.65,\n        \"num_unique_values\": 5588,\n        \"samples\":
[\n          527.2,\n          1290.39,\n          7249.07\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"type       \",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"organic      \",\n          \"conventional \"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"year\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 2015,\n        \"max\": 2018,\n
\"num_unique_values\": 4,\n        \"samples\": [\n          2016,\n
2018\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"region\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 54,\n
\"samples\": [\n          \"Indianapolis\",\n          \"Syracuse\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe","variable_name":"avocado_df"}
```

```
avocado_df_sample = avocado_df[avocado_df['region']=='West']

avocado_df_sample
```

```
{"repr_error":"0","type":"dataframe","variable_name":"avocado_df_sampl
e"}
```

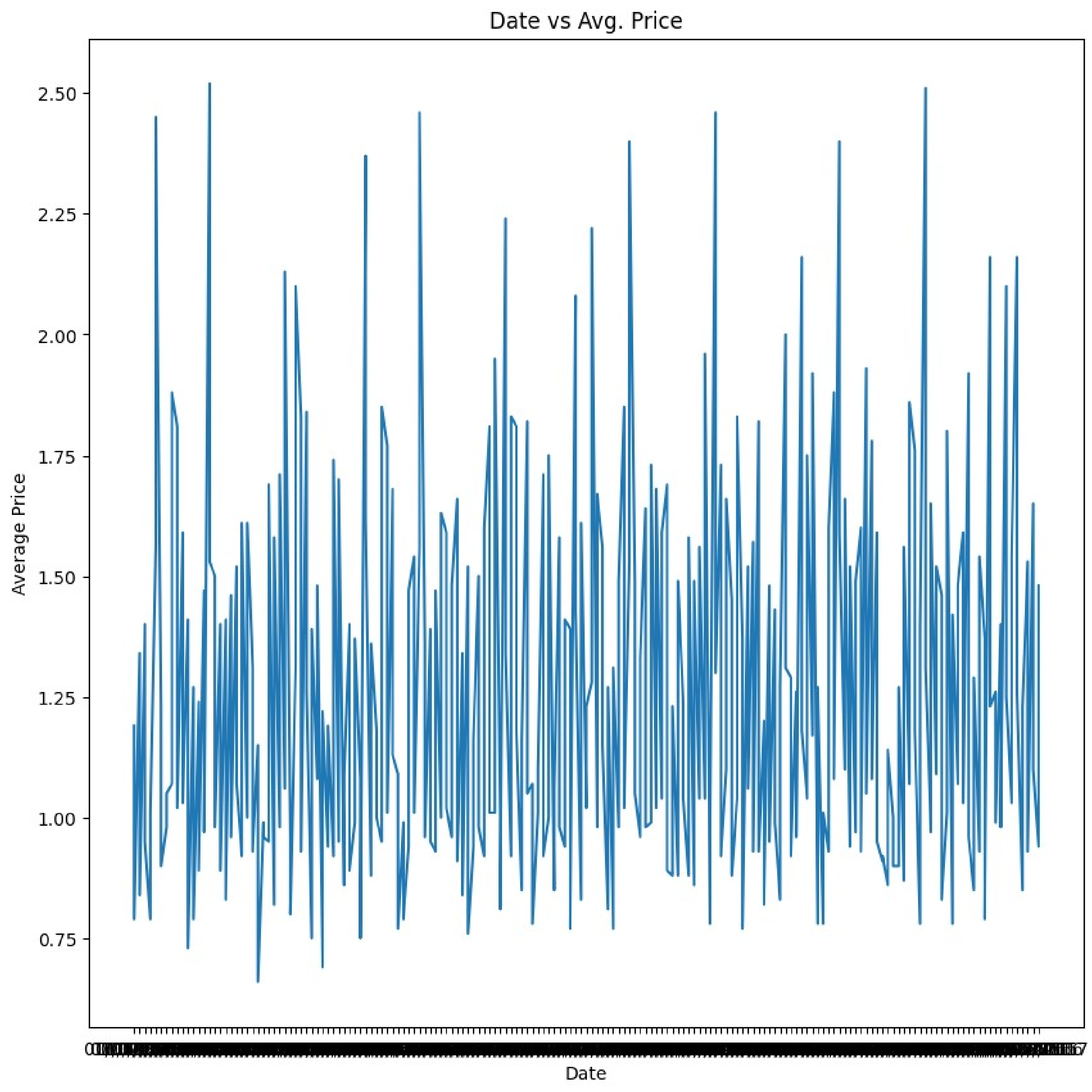
```
avocado_df_sample
```

```
{"repr_error":"0","type":"dataframe","variable_name":"avocado_df_sampl
e"}
```

```
avocado_df_sample = avocado_df_sample.sort_values("Date")

avocado_df_sample
```

```
{"repr_error":"0","type":"dataframe","variable_name":"avocado_df_sampl
e"}
```

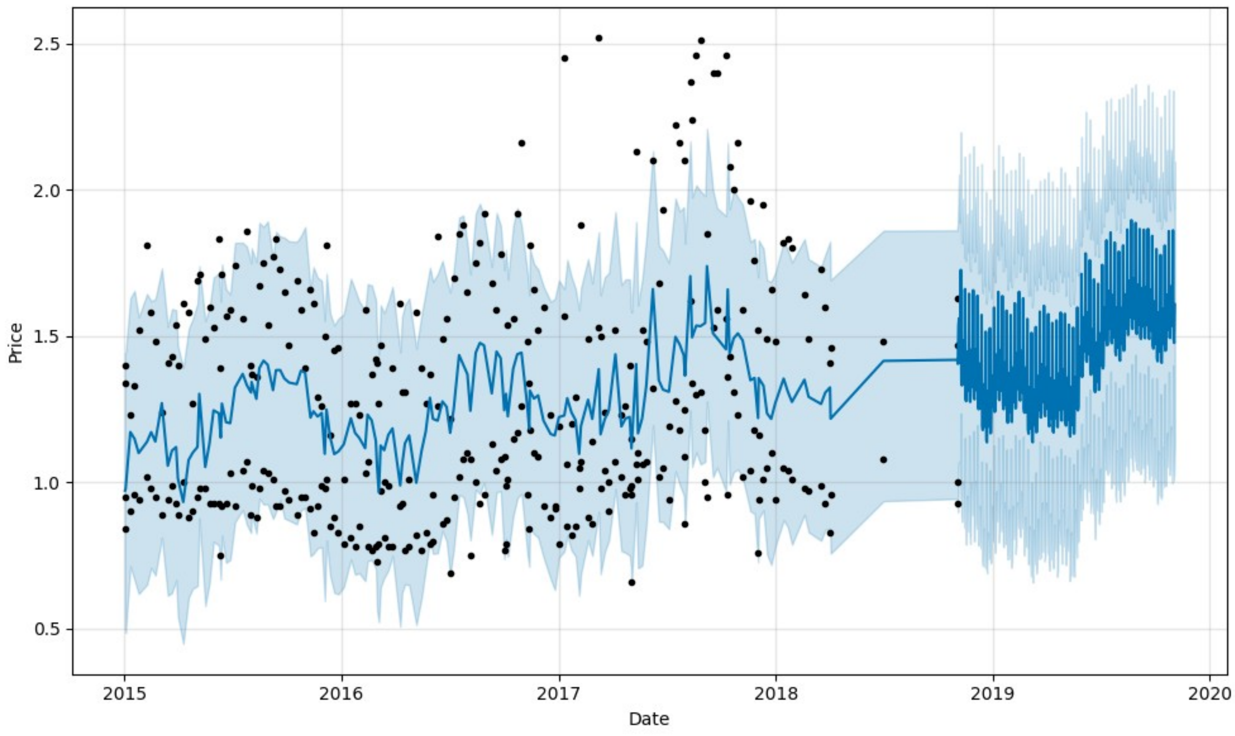
```
plt.figure(figsize=(10,10))
plt.xlabel("Date")
plt.ylabel("Average Price")
plt.title("Date vs Avg. Price")
plt.plot(avocado_df_sample['Date'], avocado_df_sample['AveragePrice'])

[<matplotlib.lines.Line2D at 0x7b79a37faec0>]
```



```
avocado_df_sample = avocado_df_sample.rename(columns={'Date':'ds',
'AveragePrice':'y'})

m = Prophet()
m.fit(avocado_df_sample)
# Forcasting into the future
future = m.make_future_dataframe(periods=365)
forecast = m.predict(future)
```

```
/usr/local/lib/python3.10/dist-packages/prophet/forecaster.py:1133:
UserWarning: Parsing dates in DD/MM/YYYY format when dayfirst=False
(the default) was specified. This may lead to inconsistently parsed
dates! Specify a format to ensure consistent parsing.
```

```
  self.history_dates =
pd.to_datetime(pd.Series(history['ds'].unique(),
name='ds')).sort_values()
/usr/local/lib/python3.10/dist-packages/prophet/forecaster.py:287:
UserWarning: Parsing dates in DD/MM/YYYY format when dayfirst=False
(the default) was specified. This may lead to inconsistently parsed
dates! Specify a format to ensure consistent parsing.
  df['ds'] = pd.to_datetime(df['ds'])
INFO:prophet:Disabling daily seasonality. Run prophet with
daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmpp_5m5kt4/onptdmvm.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpp_5m5kt4/gdrywqu8.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-
packages/prophet/stan_model/prophet_model.bin', 'random',
'seed=75495', 'data', 'file=/tmp/tmpp_5m5kt4/onptdmvm.json',
'init=/tmp/tmpp_5m5kt4/gdrywqu8.json', 'output',
'file=/tmp/tmpp_5m5kt4/prophet_modeliobnhn90/prophet_model-
20240403202948.csv', 'method=optimize', 'algorithm=lbfgs',
'iter=10000']
20:29:48 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
20:29:48 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing

figure = m.plot(forecast, xlabel='Date', ylabel='Price')
```



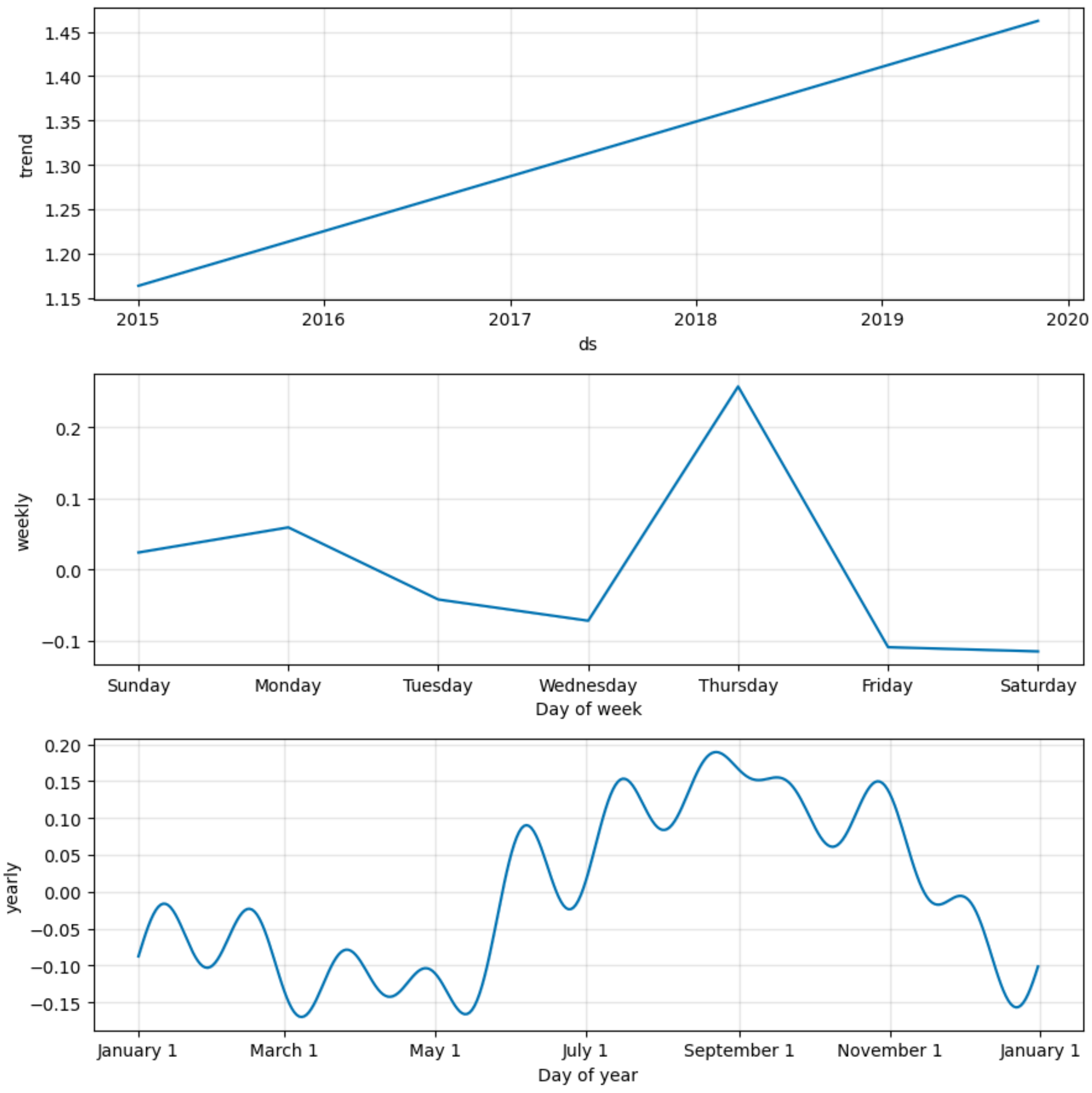
Inferences:

1. There is significant volatility in the prices, with frequent fluctuations and spikes observed throughout the time period.

2. The overall price trend shows an upward trajectory, with prices generally increasing from 2015 to 2020, despite the volatility.

3. The volatility and price spikes seem to be more pronounced towards the latter part of the timeframe, especially in 2019 and 2020 (covid), suggesting increased market uncertainty or turbulence during that period.

```
figure3 = m.plot_components(forecast)
```



COMPARISION WITH A LINEAR REGRESSION MODEL

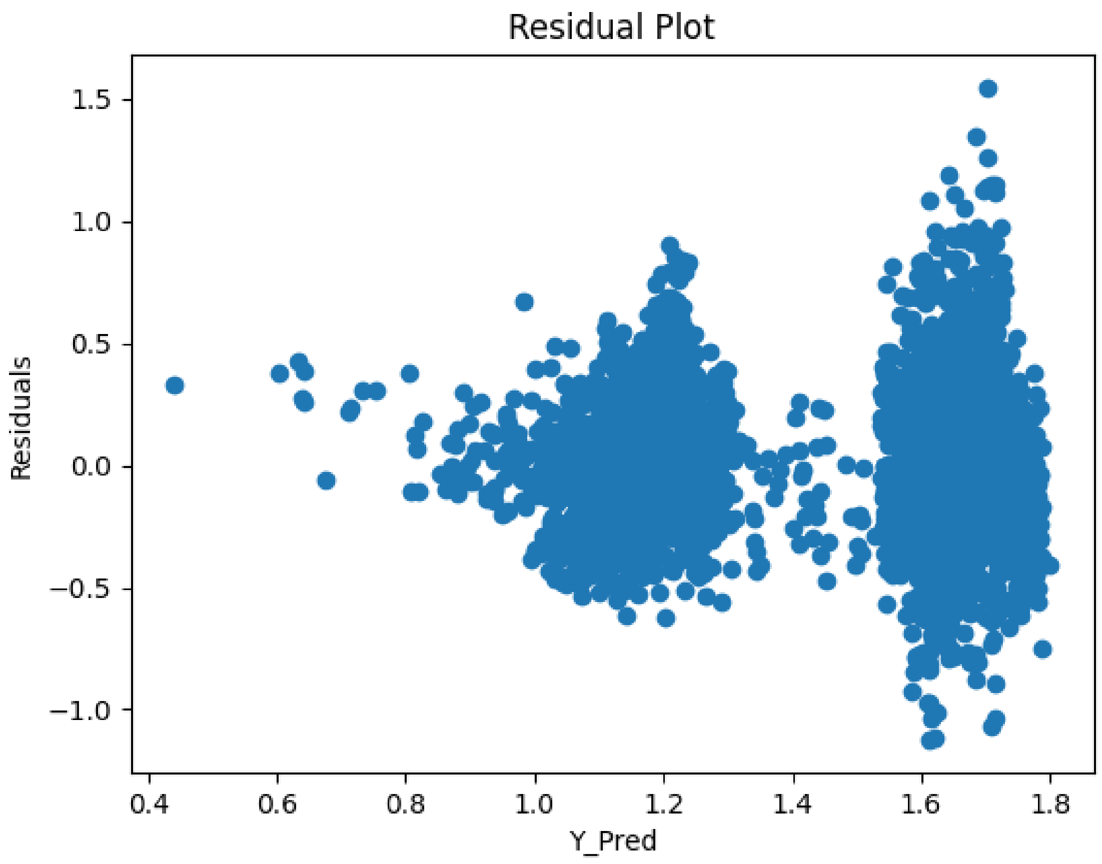
Now, let's comapare the same dataset on a Linear Regression model. Even though it might not make sense to make a Linear Model on such data because of such strong seasonality and nonlinear trends. Also because, we will be training the model on different features such as bags sold, total volumne, type of avocados which cannot be pre-determined for a date. The only true feature known is Date, hence it is always advisable to use Time-Series models on such datasets.

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn. linear_model import LinearRegression
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import r2_score

avocado_lr = pd.read_csv("/content/avocado.csv")
avocado_lr.head()
```

```
{"summary":"{\n  \"name\": \"avocado_lr\",\n  \"rows\": 18249,\n
\"fields\": [\n    {\n      \"column\": \"   \",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
15,\n        \"min\": 0,\n        \"max\": 52,\n
\"num_unique_values\": 53,\n        \"samples\": [\n          19,\n
41,\n          47\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Date\",\n      \"properties\": {\n        \"dtype\": \"object\",\n
\"num_unique_values\": 169,\n        \"samples\": [\n
\"07/05/2017\",\n          \"31/05/2015\",\n          \"17/09/2017\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"AveragePrice\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.40267655549555065,\n        \"min\": 0.44,\n        \"max\": 3.25,\n
\"num_unique_values\": 259,\n        \"samples\": [\n          0.88,\n
0.91,\n          1.07\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Total Volume \",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 3453545.3553994712,\n        \"min\":
84.56,\n        \"max\": 62505646.52,\n        \"num_unique_values\":
18237,\n        \"samples\": [\n          9783.93,\n
2604991.25,\n          277267.97\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"4046\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1264989.0817627772,\n
\"min\": 0.0,\n        \"max\": 22743616.17,\n
\"num_unique_values\": 17702,\n        \"samples\": [\n
97.29,\n          97139.95,\n          6339.49\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"4225\",\n      \"properties\": {\n
```

```
\"dtype\": \"number\",\n        \"std\": 1204120.4011350507,\n
\"min\": 0.0,\n        \"max\": 20470572.61,\n
\"num_unique_values\": 18103,\n        \"samples\": [\n
11096.86,\n          22648.55,\n          87497.9\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"4770\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 107464.06843537073,\n
\"min\": 0.0,\n        \"max\": 2546439.11,\n
\"num_unique_values\": 12071,\n        \"samples\": [\n
74.0,\n          1875.51,\n          202.18\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Total Bags\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
986242.3992164118,\n        \"min\": 0.0,\n        \"max\":
19373134.37,\n        \"num_unique_values\": 18097,\n
\"samples\": [\n          63800.09,\n          12938.12,\n
912891.54\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Small Bags\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 746178.5149617889,\n        \"min\":
0.0,\n        \"max\": 13384586.8,\n        \"num_unique_values\":
17321,\n        \"samples\": [\n          41688.75,\n
118293.26,\n          93488.63\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Large Bags\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 243965.96454740883,\n
\"min\": 0.0,\n        \"max\": 5719096.61,\n
\"num_unique_values\": 15082,\n        \"samples\": [\n
5155.3,\n          43900.93,\n          163807.71\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"XLarge Bags\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
17692.894651916486,\n        \"min\": 0.0,\n        \"max\":
551693.65,\n        \"num_unique_values\": 5588,\n        \"samples\":
[\n          527.2,\n          1290.39,\n          7249.07\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"type       \",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"organic      \",\n          \"conventional \"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"year\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 2015,\n        \"max\": 2018,\n
\"num_unique_values\": 4,\n        \"samples\": [\n          2016,\n
2018\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"region\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 54,\n
```

```
\"samples\": [\n          \"Indianapolis\",\n          \"Syracuse\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe","variable_name":"avocado_lr"}

#Changing Categorical data to numerical data
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()

avocado_lr['type       ']= label_encoder.
fit_transform(avocado_lr['type       '])
avocado_lr['region']= label_encoder.
fit_transform(avocado_lr['region'])

# Setting input and output features
X = avocado_lr.drop(["Date","AveragePrice"],axis=1)
y = avocado_lr[["AveragePrice"]].copy()

scaler = StandardScaler()
scaler.fit(X)
X=scaler.transform(X)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.20, random_state = 42)

lr_model = LinearRegression()
lr_model.fit(X_train,y_train)

y_pred = lr_model.predict(X_test)
r2 = r2_score(y_test, y_pred)

# Residual Graph
residuals = y_test-y_pred
plt.scatter(y_pred,residuals)
plt.xlabel("Y_Pred")
plt.ylabel("Residuals")
plt.title("Residual Plot")
plt.show()
```



```
print(r2)

0.40925667566177626
```

Inference:

- Here we can see that we achieved an accuracy of 0.40 which is fairly poor for a model.