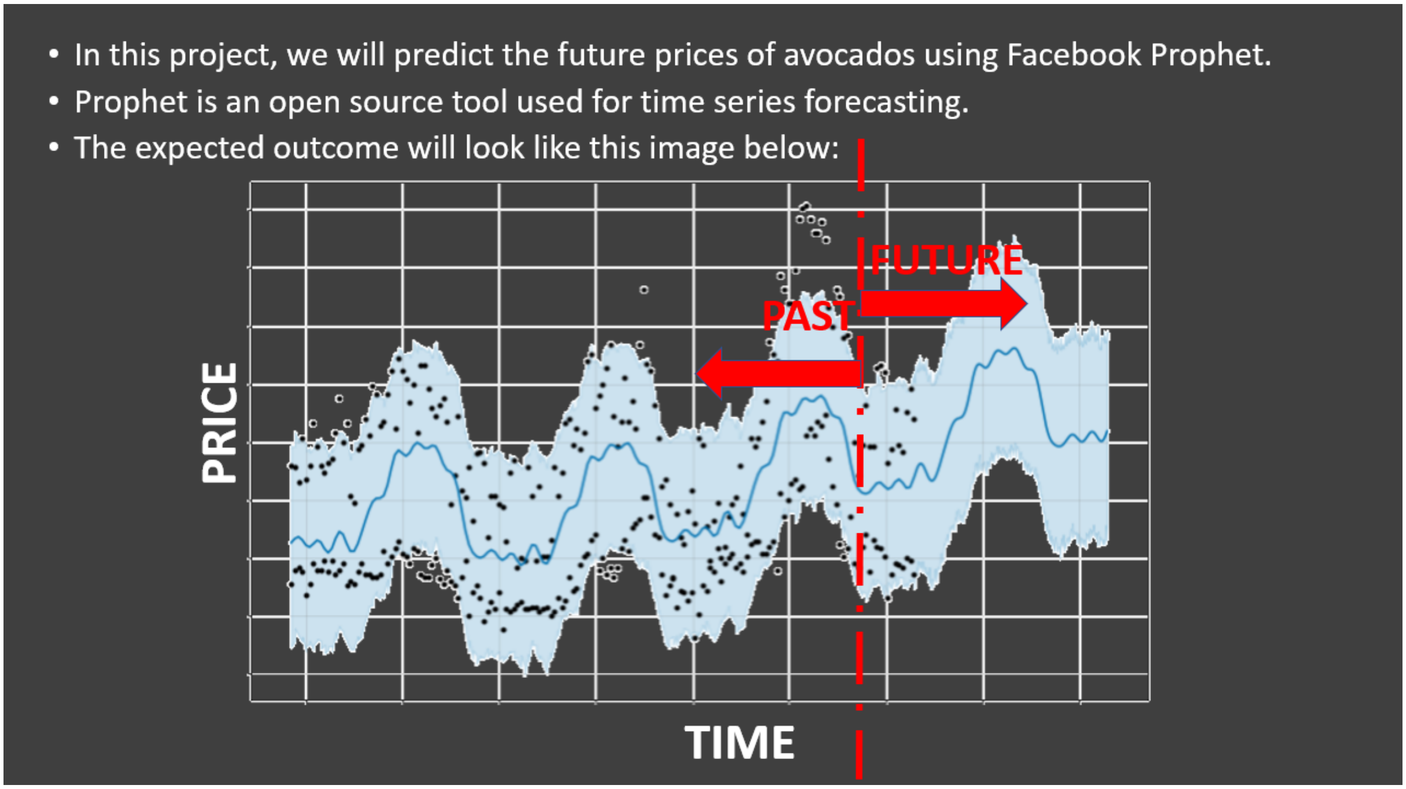
PREDICTING FUTURE PRODUCT PRICES USING FACEBOOK PROPHET

Professor Ahmed

Submitted by Donald Ghazi

TASK #1: PROJECT OVERVIEW



- In this project, we will predict the future prices of avocados using Facebook Prophet.
- Prophet is an open source tool used for time series forecasting.
- The expected outcome will look like this image below:

TASK #2: IMPORT LIBRARIES AND DATASET

- You must install fbprophet package as follows: pip install fbprophet
- If you encounter an error, try: conda install -c conda-forge fbprophet

In [1]:

```
# 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 fbprophet import Prophet
```

```
Importing plotly failed. Interactive plots will not work.
```

In [2]:

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

- 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

In [3]:

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

Out[3]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Lar Ba
0	0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.
1	1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.
2	2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.
3	3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.
4	4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.

In [4]:

```
# Let's view the last elements in the training dataset
avocado_df.tail(10)
```

Out[4]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags
18239	2	2018-03-11	1.56	22128.42	2162.67	3194.25	8.93	16762.57	16510.32
18240	3	2018-03-04	1.54	17393.30	1832.24	1905.57	0.00	13655.49	13401.93
18241	4	2018-02-25	1.57	18421.24	1974.26	2482.65	0.00	13964.33	13698.27
18242	5	2018-02-18	1.56	17597.12	1892.05	1928.36	0.00	13776.71	13553.53
18243	6	2018-02-11	1.57	15986.17	1924.28	1368.32	0.00	12693.57	12437.35
18244	7	2018-02-04	1.63	17074.83	2046.96	1529.20	0.00	13498.67	13066.82
18245	8	2018-01-28	1.71	13888.04	1191.70	3431.50	0.00	9264.84	8940.04
18246	9	2018-01-21	1.87	13766.76	1191.92	2452.79	727.94	9394.11	9351.80
18247	10	2018-01-14	1.93	16205.22	1527.63	2981.04	727.01	10969.54	10919.54
18248	11	2018-01-07	1.62	17489.58	2894.77	2356.13	224.53	12014.15	11988.14

In [5]:

```
avocado_df.describe()
```

Out[5]:

	Unnamed: 0	AveragePrice	Total Volume	4046	4225	4770	
count	18249.000000	18249.000000	1.824900e+04	1.824900e+04	1.824900e+04	1.824900e+04	1.8
mean	24.232232	1.405978	8.506440e+05	2.930084e+05	2.951546e+05	2.283974e+04	2.3
std	15.481045	0.402677	3.453545e+06	1.264989e+06	1.204120e+06	1.074641e+05	9.8
min	0.000000	0.440000	8.456000e+01	0.000000e+00	0.000000e+00	0.000000e+00	0.0
25%	10.000000	1.100000	1.083858e+04	8.540700e+02	3.008780e+03	0.000000e+00	5.0
50%	24.000000	1.370000	1.073768e+05	8.645300e+03	2.906102e+04	1.849900e+02	3.9
75%	38.000000	1.660000	4.329623e+05	1.110202e+05	1.502069e+05	6.243420e+03	1.1
max	52.000000	3.250000	6.250565e+07	2.274362e+07	2.047057e+07	2.546439e+06	1.9

In [6]:

```
avocado_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18249 entries, 0 to 18248
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Unnamed: 0    18249 non-null  int64  
 1   Date          18249 non-null  object 
 2   AveragePrice  18249 non-null  float64
 3   Total Volume  18249 non-null  float64
 4   4046          18249 non-null  float64
 5   4225          18249 non-null  float64
 6   4770          18249 non-null  float64
 7   Total Bags    18249 non-null  float64
 8   Small Bags    18249 non-null  float64
 9   Large Bags    18249 non-null  float64
 10  XLarge Bags   18249 non-null  float64
 11  type          18249 non-null  object 
 12  year          18249 non-null  int64  
 13  region        18249 non-null  object 
dtypes: float64(9), int64(2), object(3)
memory usage: 1.9+ MB
```

In [7]:

```
avocado_df.isnull().sum()
```

Out[7]:

```
Unnamed: 0      0
Date            0
AveragePrice    0
Total Volume    0
4046            0
4225            0
4770            0
Total Bags      0
Small Bags      0
Large Bags      0
XLarge Bags     0
type            0
year            0
region          0
dtype: int64
```

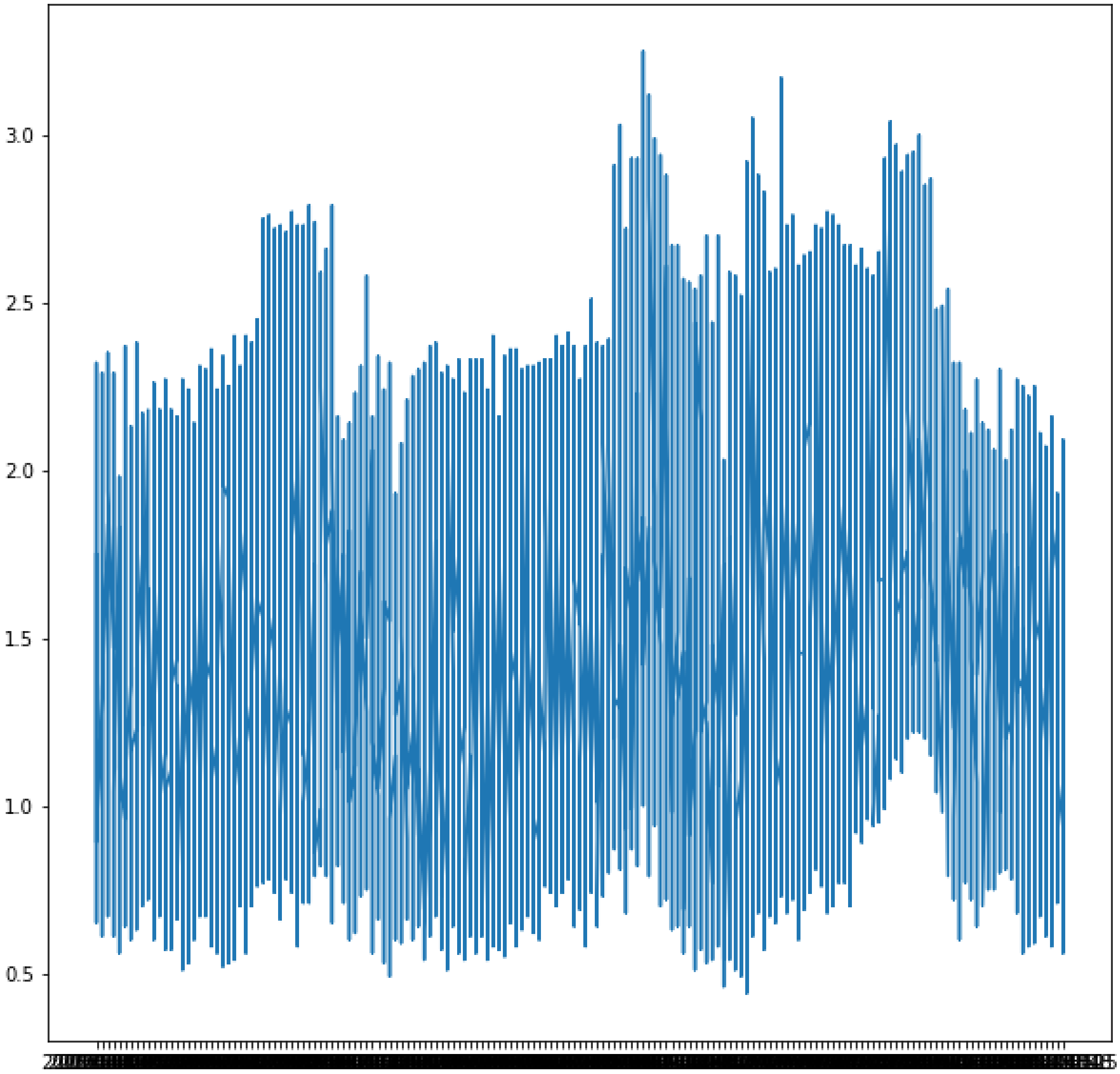
TASK #3: EXPLORE DATASET

In [8]:

```
avocado_df = avocado_df.sort_values('Date')
```

In [9]:
```
# Plot date and average price
plt.figure(figsize = (10,10))
plt.plot(avocado_df['Date'], avocado_df['AveragePrice'])
```

Out[9]:
```
[<matplotlib.lines.Line2D at 0x24c68d943c8>]
```

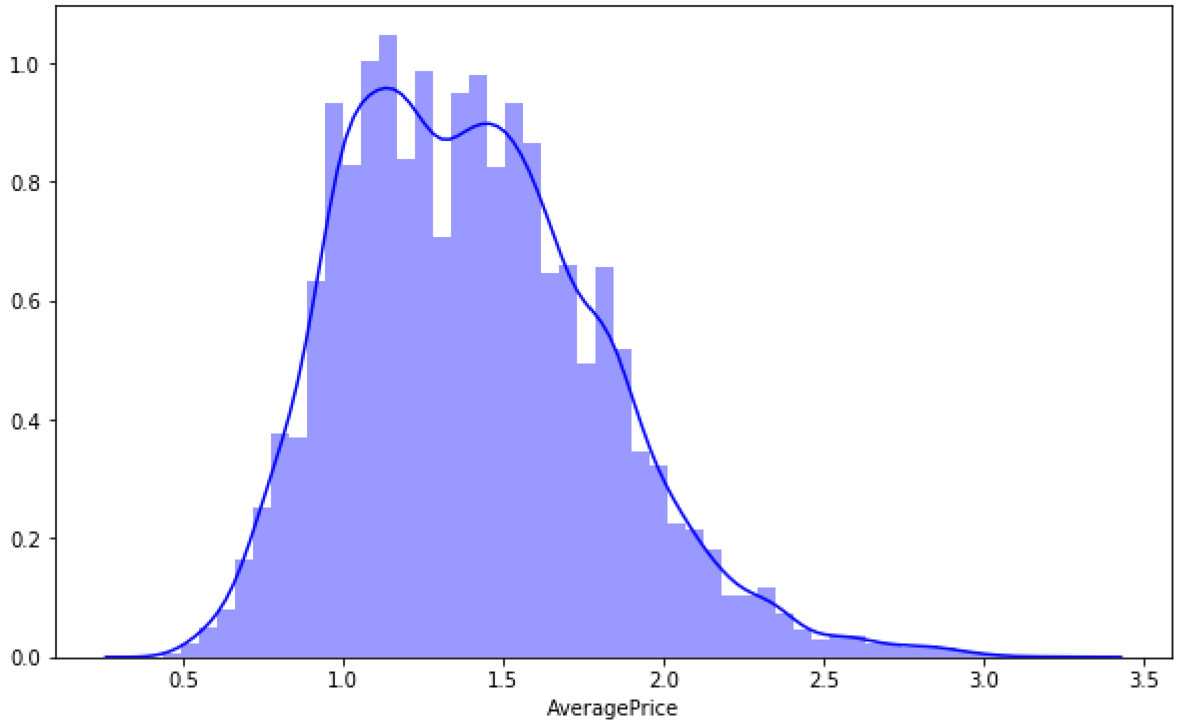


In [10]:

```
# Plot distribution of the average price
plt.figure(figsize = (10, 6))
sns.distplot(avocado_df['AveragePrice'], color = 'b')
```

Out[10]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24c69995508>
```



In [ ]:

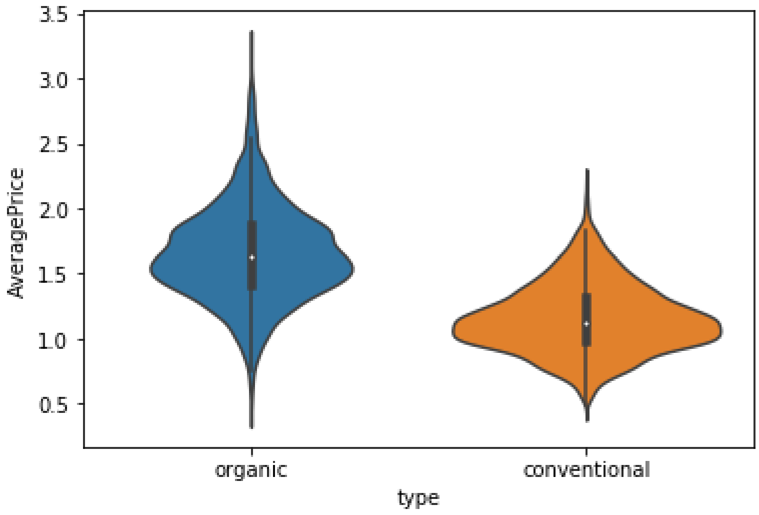
```

```

In [11]:

```
# Plot a violin plot of the average price vs. avocado type
sns.violinplot(y = 'AveragePrice', x ='type', data =avocado_df)
```

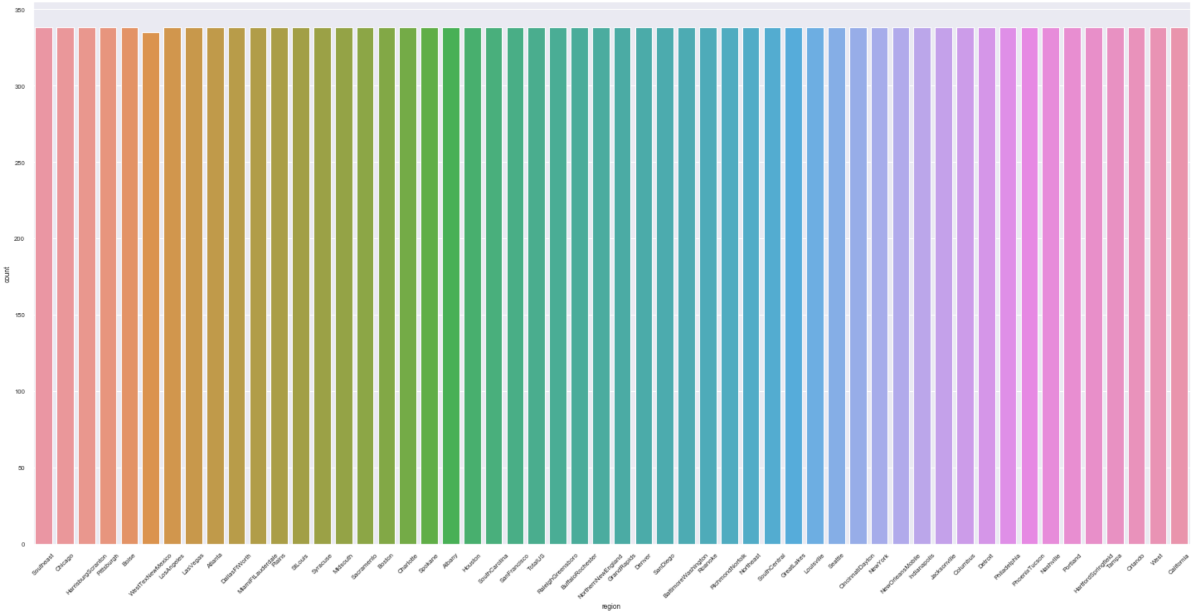
Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24c699e1d48>
```



In [12]:
```
# Bar Chart to indicate the number of regions

sns.set(font_scale=0.7)
plt.figure(figsize=[25,12])
sns.countplot(x = 'region', data = avocado_df)
plt.xticks(rotation = 45)
```

Out[12]:
```
(array([ 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]),
 <a list of 54 Text xticklabel objects>)
```

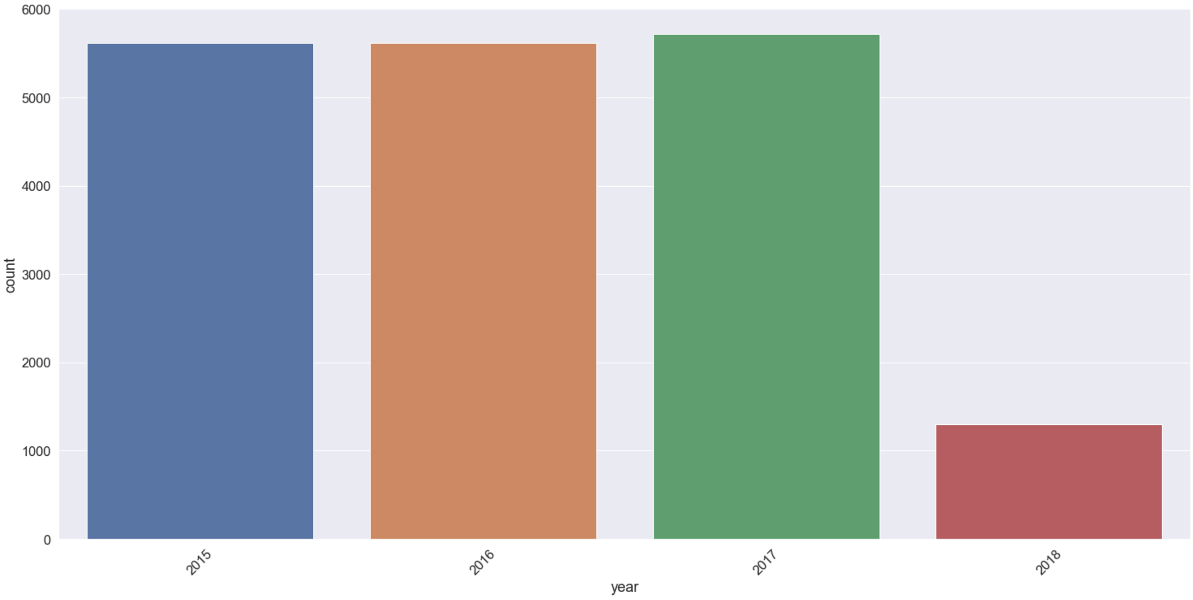


In [13]:

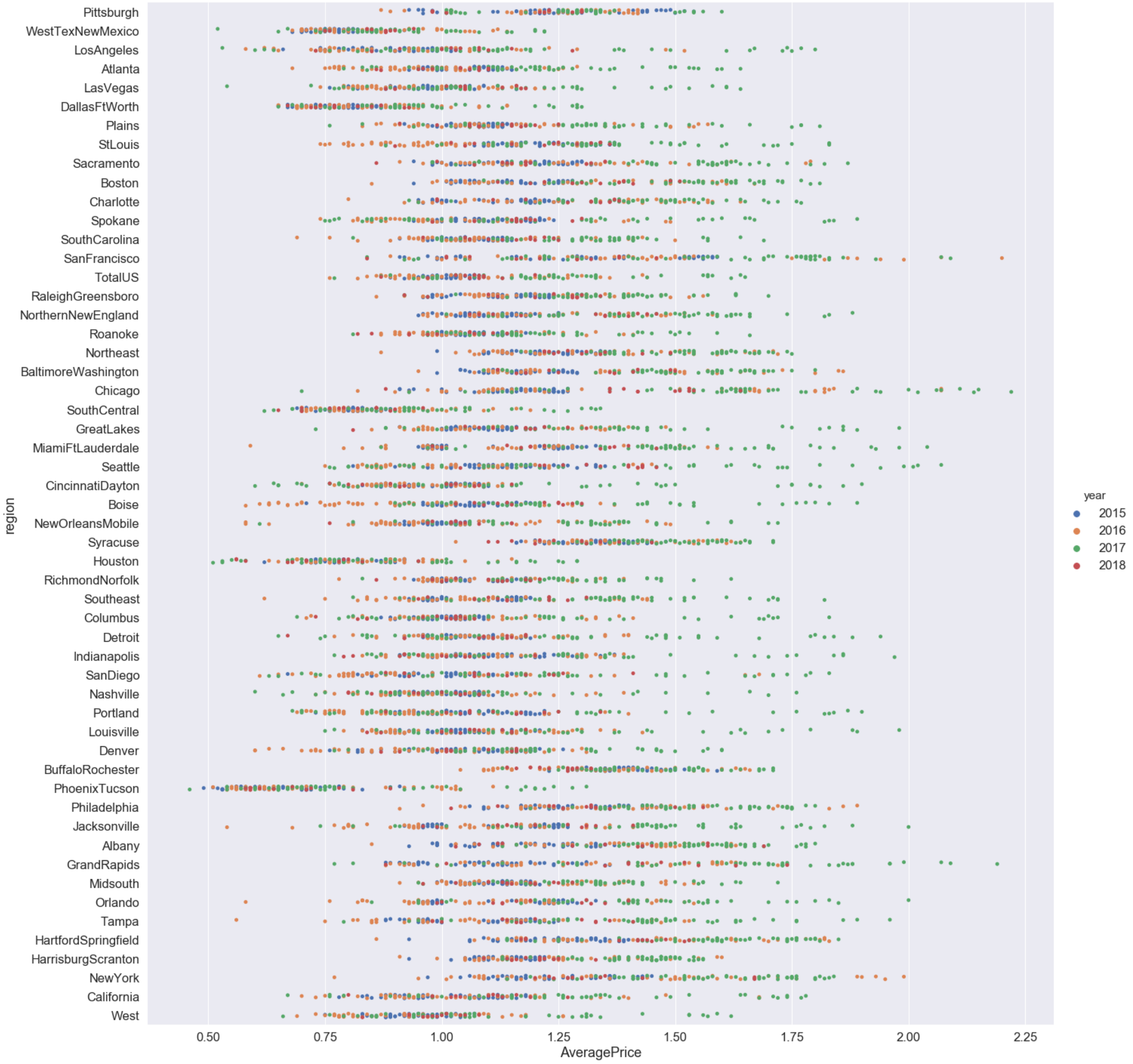
```
# Bar Chart to indicate the count in every year
sns.set(font_scale=1.5)
plt.figure(figsize=[25,12])
sns.countplot(x = 'year', data = avocado_df)
plt.xticks(rotation = 45)
```

Out[13]:

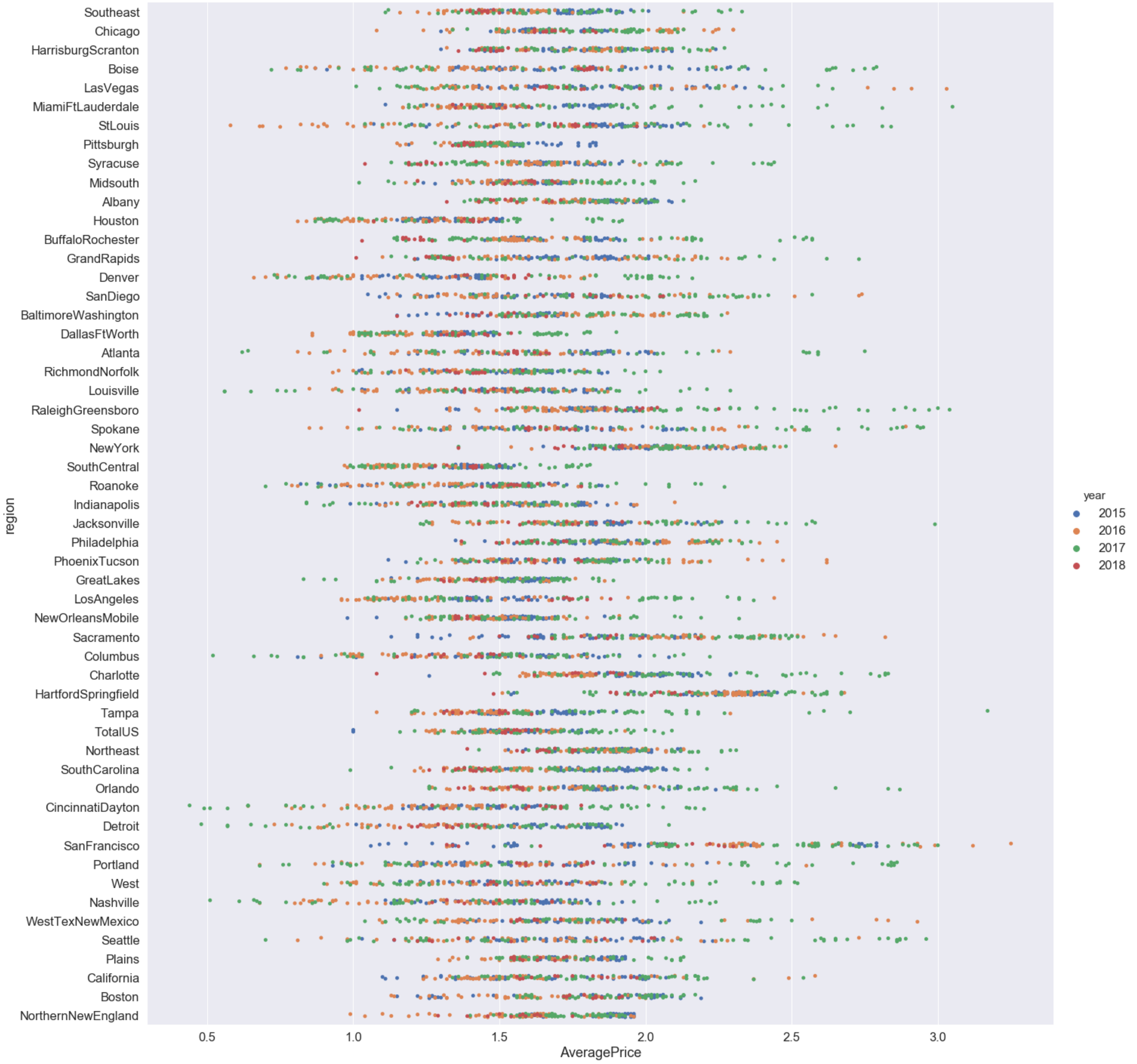
```
(array([0, 1, 2, 3]), <a list of 4 Text xticklabel objects>)
```



In [14]:
```
# plot the avocado prices vs. regions for conventional avocados
conventional = sns.catplot('AveragePrice', 'region', data = avocado_df[avocado_df['type']=='conventional'],hue = 'year', height = 20)
```



In [15]:
```
  # plot the avocado prices vs. regions for organic avocados
conventional = sns.catplot('AveragePrice', 'region', data = avocado_df[avocado_df['type']=='organic'],hue = 'year', height = 20)
```



TASK 4: PREPARE THE DATA BEFORE APPLYING FACEBOOK PROPHET TOOL

In [16]:
```
avocado_df
```

Out[16]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bag
11569	51	2015-01-04	1.75	27365.89	9307.34	3844.81	615.28	13598.4
9593	51	2015-01-04	1.49	17723.17	1189.35	15628.27	0.00	905.5
10009	51	2015-01-04	1.68	2896.72	161.68	206.96	0.00	2528.0
1819	51	2015-01-04	1.52	54956.80	3013.04	35456.88	1561.70	14925.1
9333	51	2015-01-04	1.64	1505.12	1.27	1129.50	0.00	374.3
...	...	...	...	...	...	...	...	..
8574	0	2018-03-25	1.36	908202.13	142681.06	463136.28	174975.75	127409.0
9018	0	2018-03-25	0.70	9010588.32	3999735.71	966589.50	30130.82	4014132.2
18141	0	2018-03-25	1.42	163496.70	29253.30	5080.04	0.00	129163.3
17673	0	2018-03-25	1.70	190257.38	29644.09	70982.10	0.00	89631.1
8814	0	2018-03-25	1.34	1774776.77	63905.98	908653.71	843.45	801373.6

18249 rows × 14 columns

In [17]:
```
avocado_prophet_df = avocado_df[['Date','AveragePrice']]
```

In [18]:

```
avocado_prophet_df
```

Out[18]:

	Date	AveragePrice
11569	2015-01-04	1.75
9593	2015-01-04	1.49
10009	2015-01-04	1.68
1819	2015-01-04	1.52
9333	2015-01-04	1.64
...	...	...
8574	2018-03-25	1.36
9018	2018-03-25	0.70
18141	2018-03-25	1.42
17673	2018-03-25	1.70
8814	2018-03-25	1.34

18249 rows × 2 columns

In [19]:

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

In [20]:

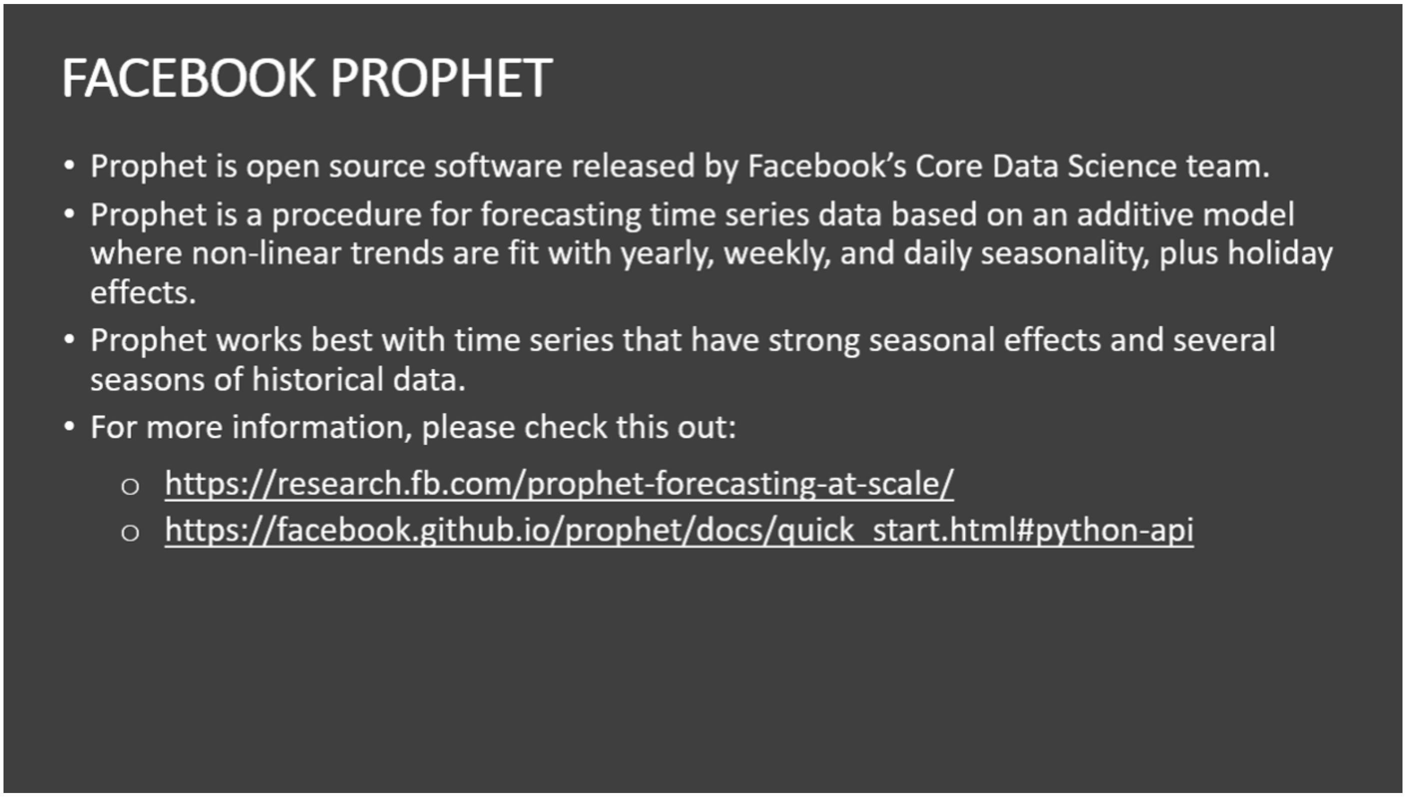
```
avocado_prophet_df
```

Out[20]:

	ds	y
11569	2015-01-04	1.75
9593	2015-01-04	1.49
10009	2015-01-04	1.68
1819	2015-01-04	1.52
9333	2015-01-04	1.64
...	...	...
8574	2018-03-25	1.36
9018	2018-03-25	0.70
18141	2018-03-25	1.42
17673	2018-03-25	1.70
8814	2018-03-25	1.34

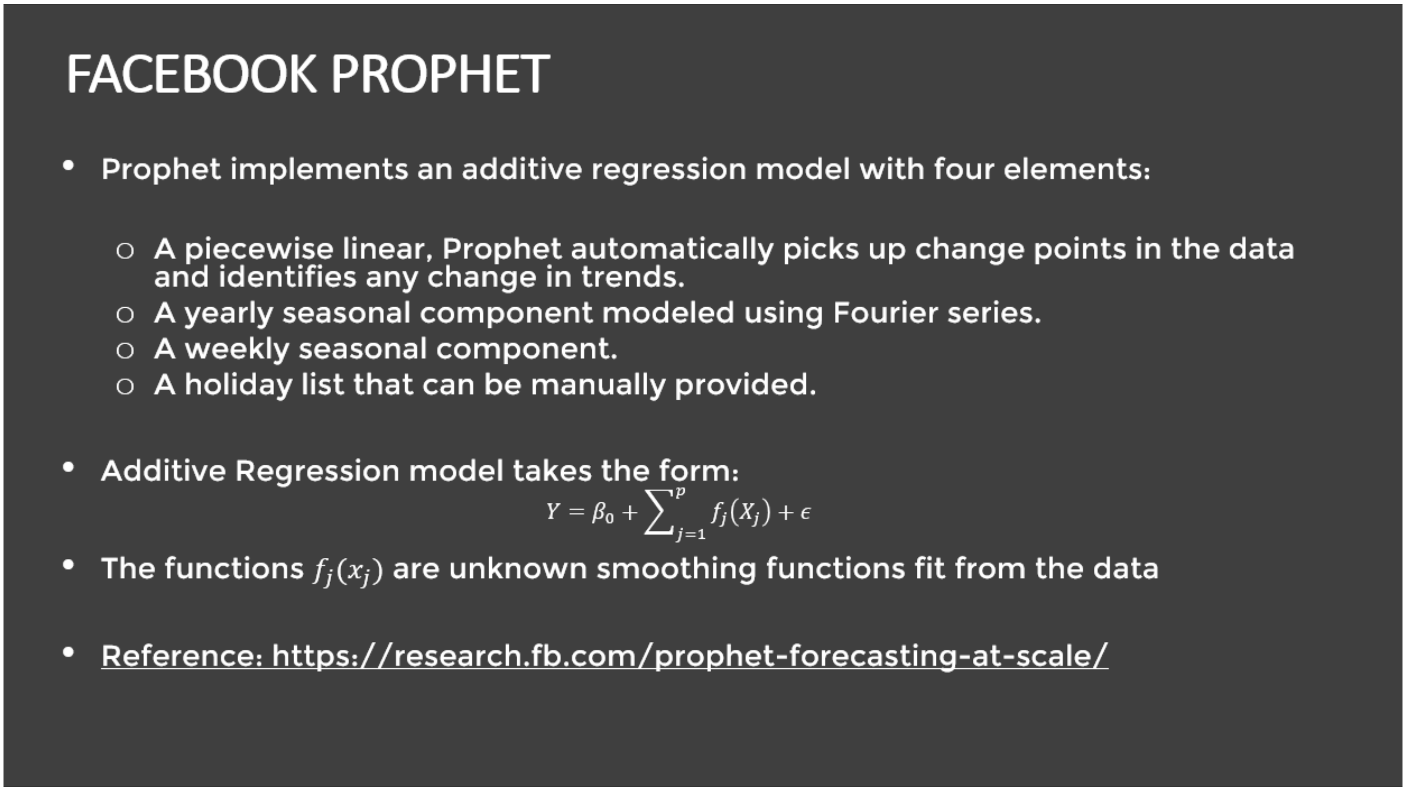
18249 rows × 2 columns

TASK 5: UNDERSTAND INTUITION BEHIND FACEBOOK PROPHET



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.
- For more information, please check this out:
  - https://research.fb.com/prophet-forecasting-at-scale/
  - https://facebook.github.io/prophet/docs/quick_start.html#python-api

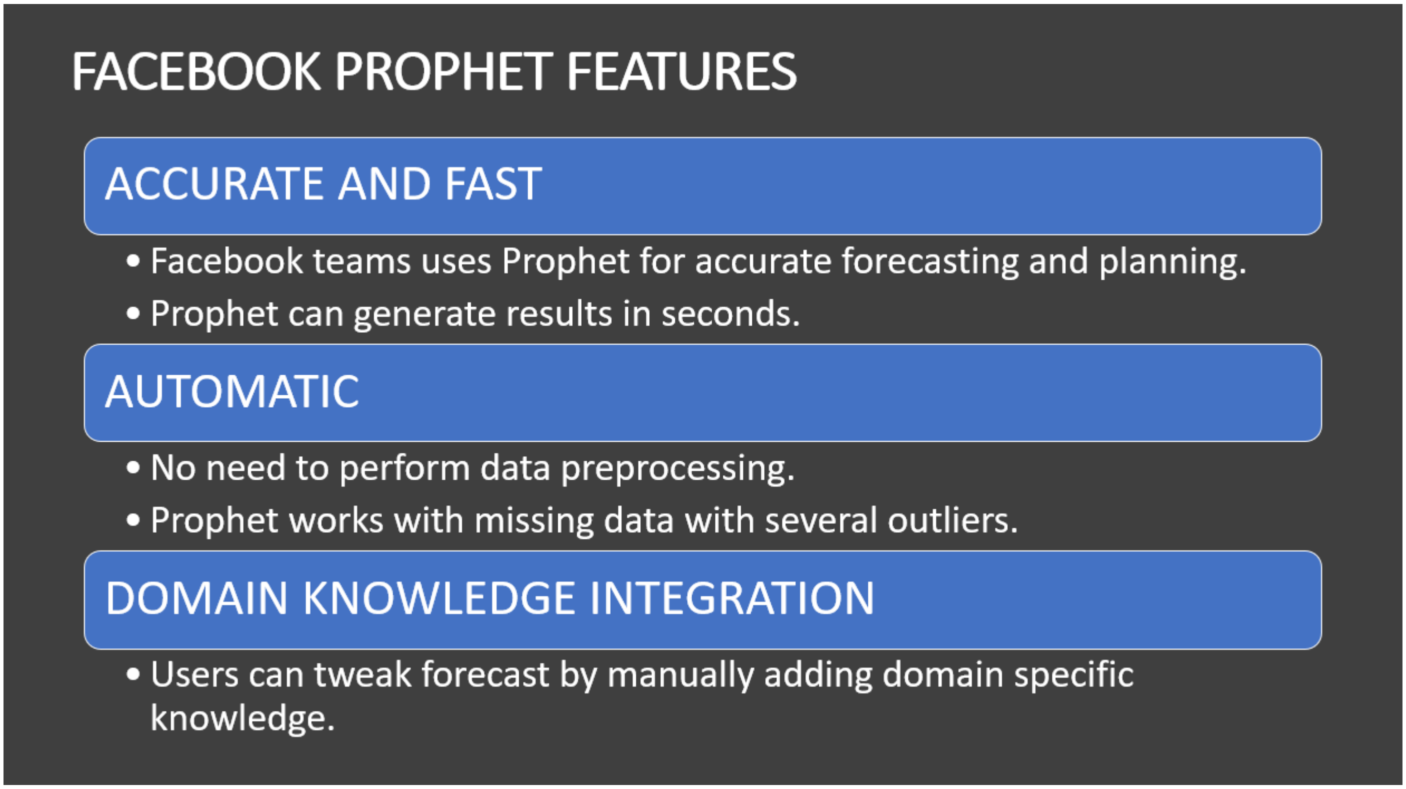


FACEBOOK PROPHET

- Prophet implements an additive regression model with four elements:
  - A piecewise linear, Prophet automatically picks up change points in the data and identifies any change in trends.
  - A yearly seasonal component modeled using Fourier series.
  - A weekly seasonal component.
  - A holiday list that can be manually provided.
- Additive Regression model takes the form:

$$Y = \beta_0 + \sum_{j=1}^{p} f_j(X_j) + \epsilon$$

- The functions $f_j(x_j)$ are unknown smoothing functions fit from the data
- Reference: https://research.fb.com/prophet-forecasting-at-scale/



FACEBOOK PROPHET FEATURES

ACCURATE AND FAST

- Facebook teams uses Prophet for accurate forecasting and planning.
- Prophet can generate results in seconds.

AUTOMATIC

- No need to perform data preprocessing.
- Prophet works with missing data with several outliers.

DOMAIN KNOWLEDGE INTEGRATION

- Users can tweak forecast by manually adding domain specific knowledge.

TASK 6: DEVELOP MODEL AND MAKE PREDICTIONS - PART A

In [21]:

```
m = Prophet()
m.fit(avocado_prophet_df)
```

```
INFO:numexpr.utils:NumExpr defaulting to 2 threads.
INFO:fbprophet:Disabling weekly seasonality. Run prophet with weekly_seasonal
ity=True to override this.
INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonalit
y=True to override this.
```

Out[21]:

```
<fbprophet.forecaster.Prophet at 0x24c6a276d88>
```

In [22]:

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

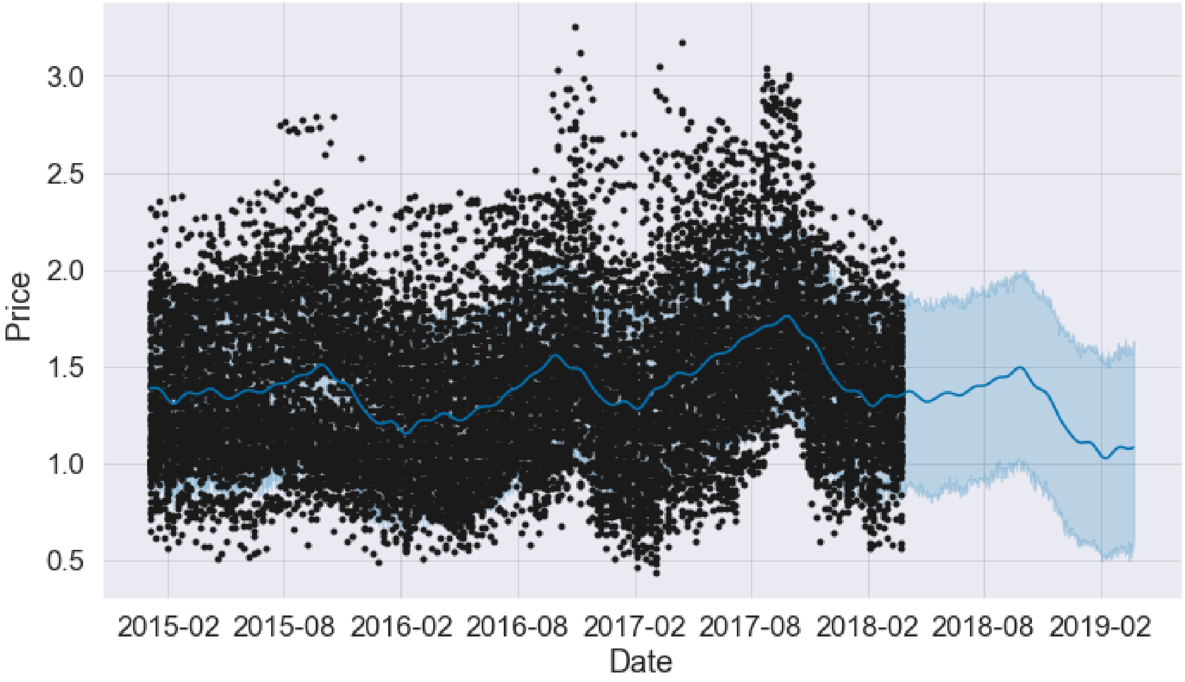
In [23]:
```
forecast
```

Out[23]:

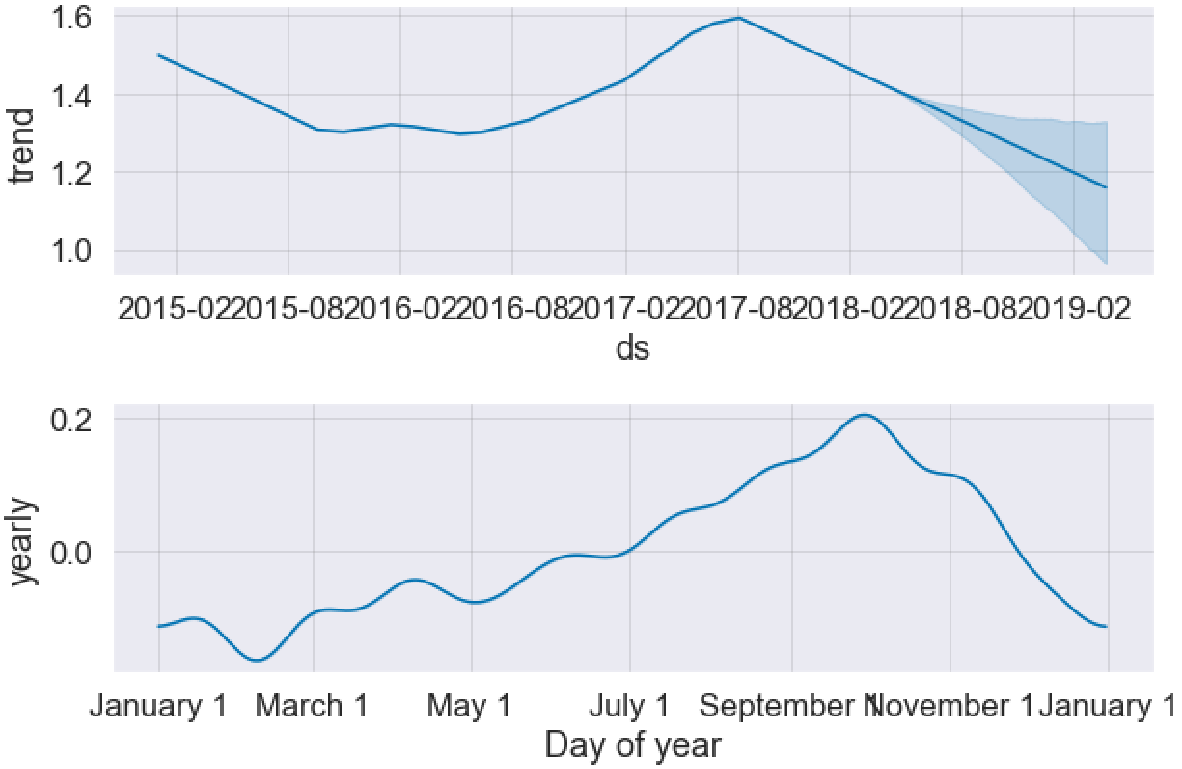
	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_terms	additi
0	2015-01-04	1.497917	0.878093	1.886680	1.497917	1.497917	-0.113109	
1	2015-01-04	1.497917	0.911452	1.889893	1.497917	1.497917	-0.113109	
2	2015-01-04	1.497917	0.881470	1.877826	1.497917	1.497917	-0.113109	
3	2015-01-04	1.497917	0.934261	1.871219	1.497917	1.497917	-0.113109	
4	2015-01-04	1.497917	0.911267	1.879417	1.497917	1.497917	-0.113109	
...	...	...	...	...	...	...	...	
18609	2019-03-21	1.161737	0.516425	1.567147	0.969928	1.328226	-0.086289	
18610	2019-03-22	1.161007	0.586514	1.569276	0.968600	1.328034	-0.084622	
18611	2019-03-23	1.160276	0.543222	1.561078	0.967258	1.328283	-0.082682	
18612	2019-03-24	1.159545	0.570143	1.599243	0.965863	1.328531	-0.080489	
18613	2019-03-25	1.158814	0.615067	1.631980	0.963728	1.328704	-0.078070	

18614 rows × 16 columns

In [24]:
```
figure = m.plot(forecast, xlabel = 'Date', ylabel = 'Price ')
```



```
In [25]: figure2 = m.plot_components(forecast)
```



TASK 7: DEVELOP MODEL AND MAKE PREDICTIONS (REGION SPECIFIC) - PART B

```
In [26]: # dataframes creation for both training and testing datasets
         avocado_df = pd.read_csv('avocado.csv')
```

```
In [27]: # Select specific region
         avocado_df_sample = avocado_df[avocado_df['region']=='West']
```

```
In [28]: avocado_df_sample = avocado_df_sample.sort_values('Date')
```

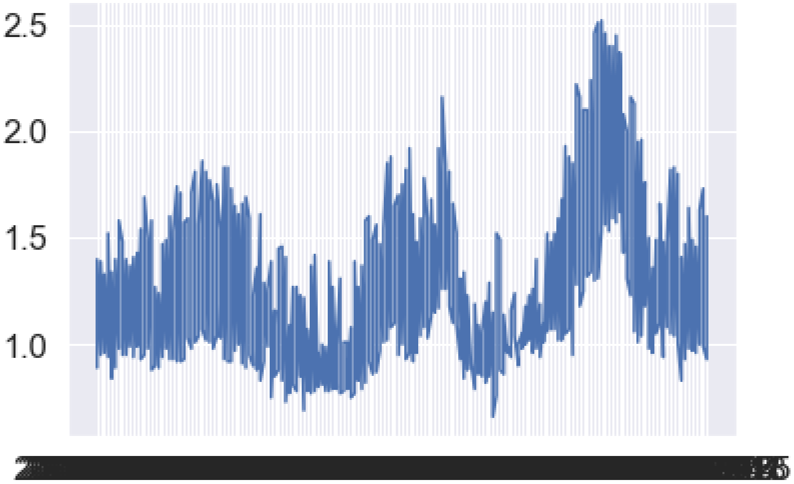
In [29]:

```
plt.plot(avocado_df_sample['Date'],avocado_df_sample['AveragePrice'])
```

```
INFO:matplotlib.category:Using categorical units to plot a list of strings th
at are all parsable as floats or dates. If these strings should be plotted as
numbers, cast to the appropriate data type before plotting.
INFO:matplotlib.category:Using categorical units to plot a list of strings th
at are all parsable as floats or dates. If these strings should be plotted as
numbers, cast to the appropriate data type before plotting.
```

Out[29]:

```
[<matplotlib.lines.Line2D at 0x24c6c59b388>]
```



In [30]:

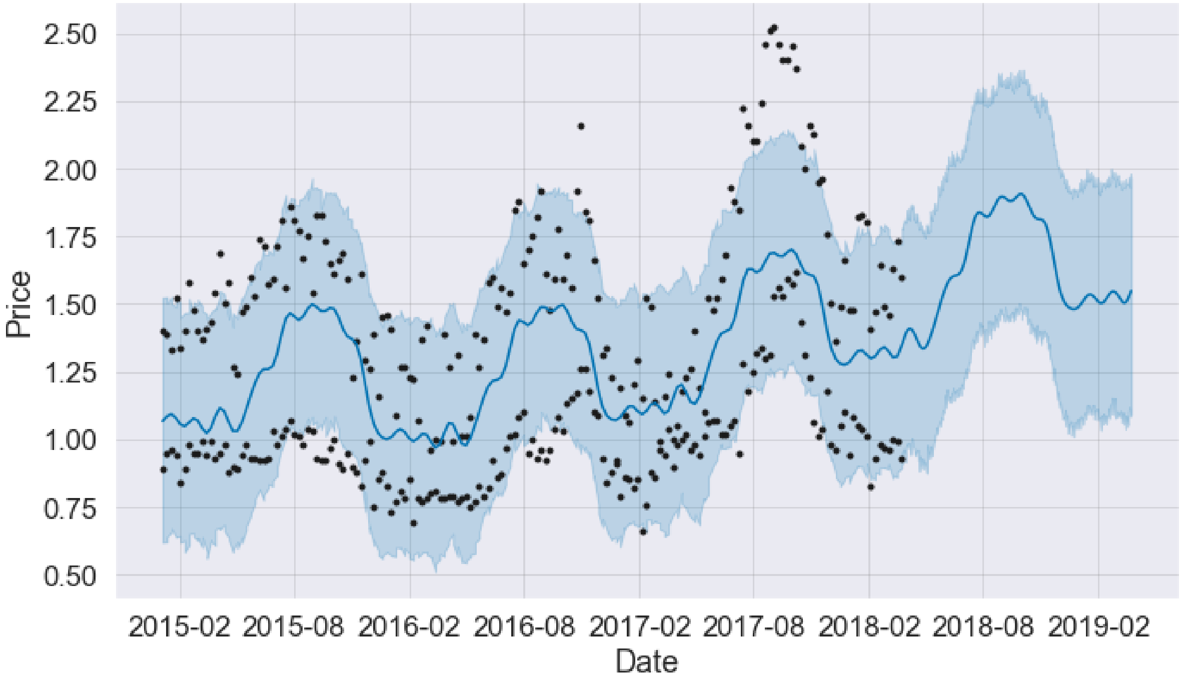
```
avocado_df_sample = avocado_df_sample.rename(columns = {'Date':'ds','AveragePr
ice':'y'})
```

In [31]:

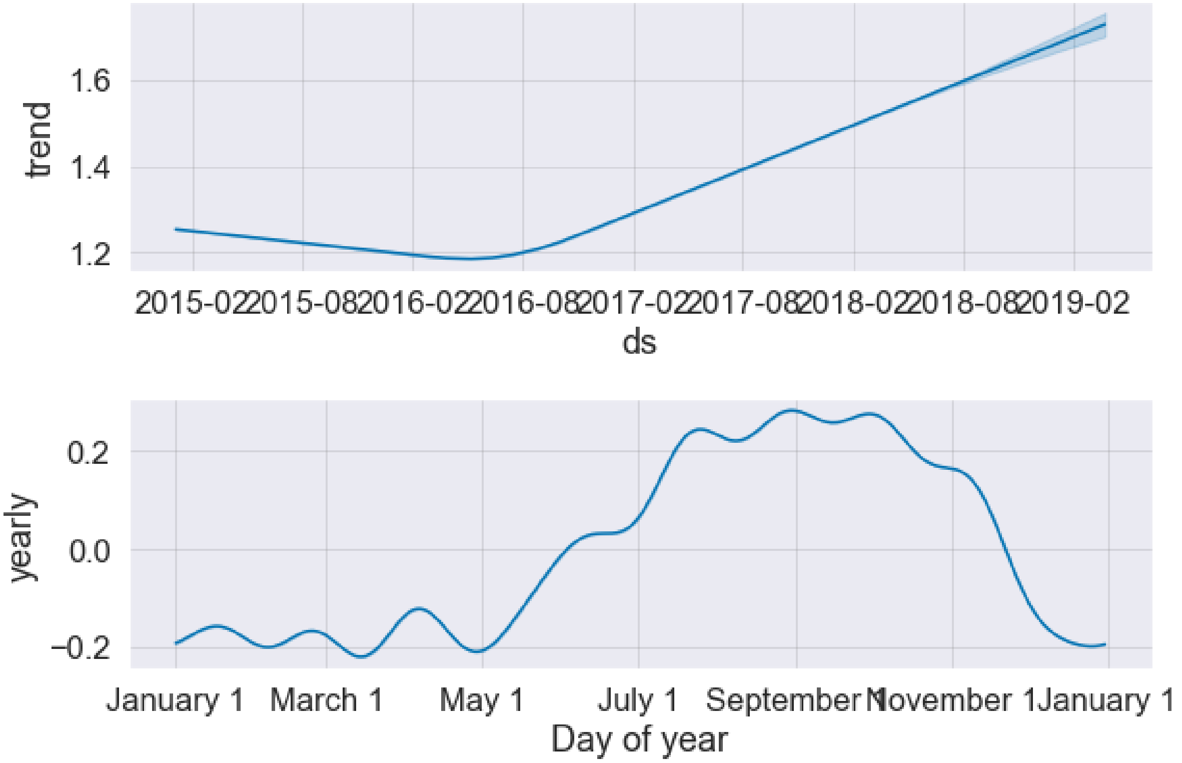
```
m = Prophet()
m.fit(avocado_df_sample)
# Forcasting into the future
future = m.make_future_dataframe(periods=365)
forecast = m.predict(future)
```

```
INFO:fbprophet:Disabling weekly seasonality. Run prophet with weekly_seasonal
ity=True to override this.
INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonalit
y=True to override this.
```

```
In [32]: figure = m.plot(forecast, xlabel='Date', ylabel='Price')
```



```
In [33]: figure3 = m.plot_components(forecast)
```


trend
ds
yearly
January 1
March 1
May 1
July 1
September 1
November 1
January 1
Day of year