Analysis of Future Lego Set Attribute Prioritization

Identifying and Predicting Trends for Lego Brick Colors

Christina DaSilva
Syracuse University iSchool
Course IST 718 Big Data Analytics
Submission Date: 12/21/2022

Summary: *Lego brick building toys have grown in popularity since the company's inception earlier in the twentieth century. Lego set attributes with prior success and sales may be an indicator for future sales success. This analysis aimed at identifying attributes of lego sets to prioritize for upcoming production to aid future sales. The scope of this analysis encompassed brick color attributes only. Based on trend predictions, increased quantities of all brick color groups were found to be recommended, focusing on higher quantities of bricks that fall into blue, red, and neutral categories. Incorporation of additional data, as well as models, are recommended to support these findings.*

Specification

Problem

The Lego Group has been in operation since 1932[1] providing toys to children and adults alike. The business question to be addressed in this analysis is: *What lego set should be made next?* This aims to solve the problem of identifying which attributes would make for successful sales in the next year. This problem is important to the company on many levels. Increased popularity of lego sets being produced likely directly impacts sales, customer satisfaction, brand notoriety, and the company's bottom line.

In an effort to scope down the analysis to fit within the timeframe of this course, the analysis will focus on identifying what color bricks should be prioritized for future lego sets. Due to time and data availability constraints, a successful measure of previous success will be based on prior inventory.

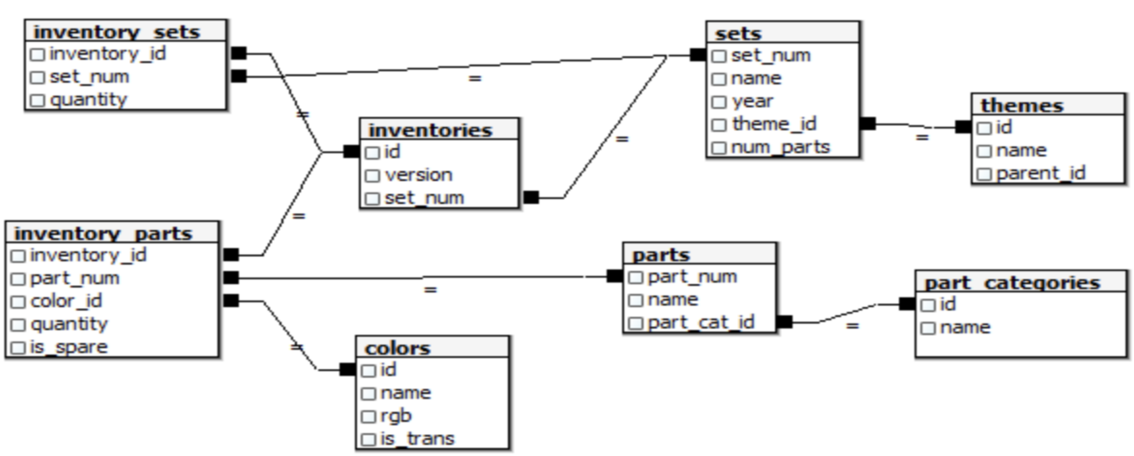
Hypothesis

Predicting which brick colors the next Lego sets should prioritize will aid future popularity and sales.

Data

Obtaining Initial Data Sets

Data from the Rebrickable Lego database, which is available on the Kaggle website[2,3], will be used for this analysis. The data set contains inventory data, including parts, sets, colors, and themes, as seen in the relationship diagram below[4]:



The initial data set includes a .csv file for each table in the diagram above, which are imported into data frames with the pandas python package. Each of the resulting data frames contains the following number of columns and rows:

Data Frame	# Rows	# Columns
colors_df	135	4
inventories_df	11681	3
inventory_parts_df	580251	5
inventory_sets_df	2846	3
part_categories_df	57	2
parts_df	25993	3
sets_df	11673	5
themes_df	614	3

Approach

The analysis is conducted in a Python notebook in Google Colab, following the OSEMN framework.

Data Scrubbing and Integration

In order to focus analysis around the brick colors for different time periods, data from several of the initial data frames requires integration. The colors, inventory parts, inventories, and sets are merged in the following, multi-step process.

```
# Merge Step 1: Merge part colors and inventory parts
color_invpart_df = pd.merge(colors_df, inventory_parts_df, left_on='id',
right_on='color_id').groupby(['name','quantity','inventory_id'],
as_index=False)['quantity'].sum()

# Merge Step 2: Merge with inventories
color_invpart_inventories_df = pd.merge(color_invpart_df, inventories_df,
left_on='inventory_id',
right_on='id').groupby(['name','quantity','inventory_id','id','set_num'],
as_index=False)['quantity'].sum()
```

```
# Merge Step 3: Merge with sets
color_invpart_inventories_sets_df = pd.merge(color_invpart_inventories_df, sets_df,
left_on='set_num', right_on='set_num')

###### Set aside new df to use for prophet analysis
color_prophet_df = color_invpart_inventories_sets_df
color_prophet_df.rename(columns={'year':'Year', 'name_x':'Color', 'quantity':'Quantity'})

# Continue merging color qty by year dataframe
color_invpart_inventories_sets_df =
color_invpart_inventories_sets_df.groupby(['year','name_x'],
as_index=False).agg({'quantity':sum})
color_qty_year_df = color_invpart_inventories_sets_df
color_qty_year_df=color_qty_year_df.rename(columns={'year':'Year', 'name_x':
'Color','quantity':'Quantity'})
```

The resulting merged data frame contains three attributes: brick *color*, *quantity,* and *year*.

	Year	Color	Quantity
0	1950	Blue	6
1	1950	Bright Green	4
2	1950	Green	6
3	1950	Light Green	2
4	1950	Medium Blue	2
...	...	...	...
2078	2017	Unknown	41
2079	2017	White	8830
2080	2017	Yellow	1956
2081	2017	Yellowish Green	131
2082	2017	[No Color]	30

2083 rows × 3 columns

Feature Generation

Time Frame Groupings

Two different features are generated to allow for different time frames views of the brick color and quantity data. First, the data are broken down by decade, from the 1950s through the 2010s.

```
# create a new dataframe with colors quantities by decade
color_qty_decade_df=color_qty_year_df

# create a list of our conditions
conditions = [
    (color_qty_year_df['Year'] <= 1959),
    (color_qty_year_df['Year'] > 1959) & (color_qty_year_df['Year'] < 1970),
    (color_qty_year_df['Year'] > 1969) & (color_qty_year_df['Year'] < 1980),
    (color_qty_year_df['Year'] > 1979) & (color_qty_year_df['Year'] < 1990),
    (color_qty_year_df['Year'] > 1989) & (color_qty_year_df['Year'] < 2000),
    (color_qty_year_df['Year'] > 1999) & (color_qty_year_df['Year'] < 2010),
    (color_qty_year_df['Year'] > 2009)
    ]
```

```
# create a list of the values we want to assign for each condition
values = ['1950s', '1960s', '1970s', '1980s','1990s','2000s','2010s']
#values = [1950, 1960, 1970, 1980, 1990, 2000, 2010]

# create a new column and use np.select to assign values to it using our lists as arguments
color_qty_decade_df['Decade'] = np.select(conditions, values)

color_qty_decade_df = color_qty_decade_df.filter(['Color','Quantity','Decade'])

color_qty_decade_df =
color_qty_decade_df.groupby(['Decade','Color'])['Quantity'].sum().to_frame().reset_index()
```

Similar methods are used to break down the same brick color and quantity data by three larger eras, including *Early Years* (1950-1979), *Middle Years* (1980-1999), and *Present Years* (2000-2017).

Generic Color Groupings

The initial brick color data contains 130 unique brick colors that have existed in inventory from the 1950s to the 2010s. In order to reduce some level of noise in the data, a mapping crosswalk was created to group brick colors into broader color categories, including *Blues, Greens, Metallics, Neutrals, Oranges, Purples, Reds,* and *Yellows*. These broader color groups were then incorporated into the following data frames to use for analysis:

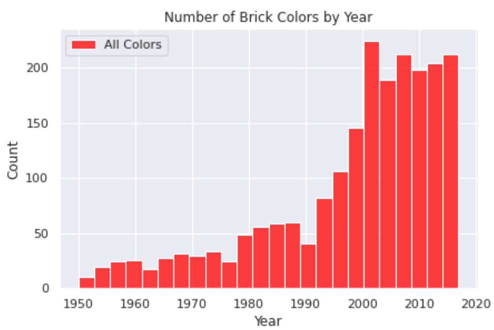
colorgroup_qty_year_df - contains the quantity of each broader color group by year
colorgroup_qty_decade_df - contains the quantity of each broader color group by decade
colorgroup_qty_eras_df - contains the quantity of each broader color group by each era

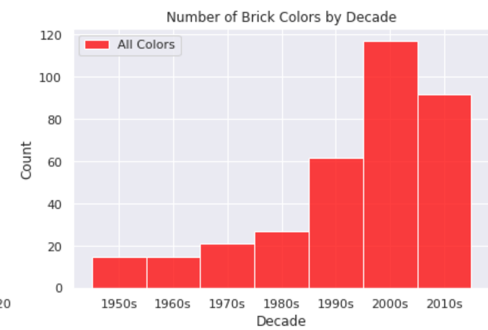
early_era_qty_df - contains the quantity of each color group in the early era (1950-1979)
mid_era_qty_df - contains the quantity of each color group in the mid era (1980-1999)
present_era_qty_df - contains the quantity of each color group in the present era (2000-2017)

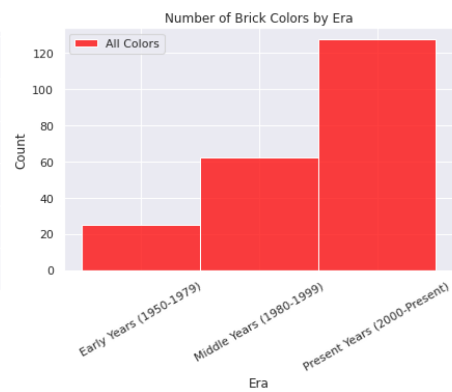
Observation

Explore

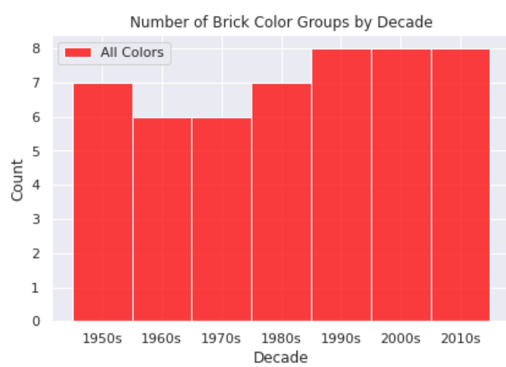
In exploring the data, it was evident that the number of brick colors increased from the 1950s to the 2010s, with the most brick color options available in the present era (2000-2017). Interestingly, the highest number of brick colors was found in the 2000s decade, and decreased in the 2010s. This may be partially due to the fact that the 2010 data is not a full decade, only going through July 2017.



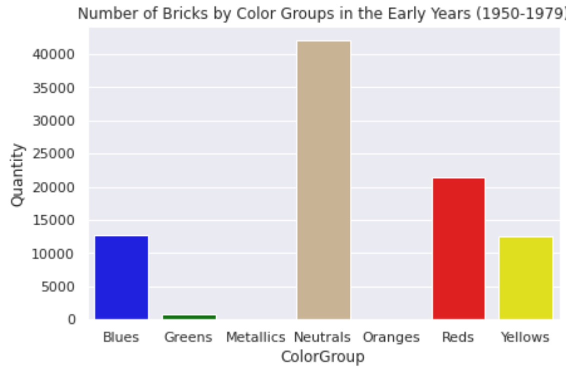


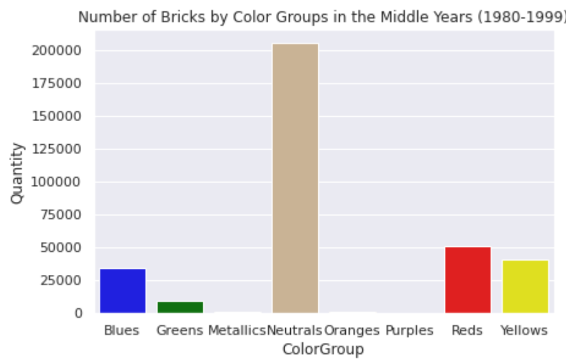


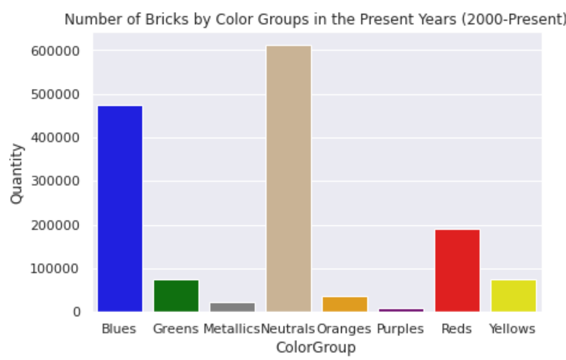
Looking at the broader color groups over time, however, there is not a large difference over time.



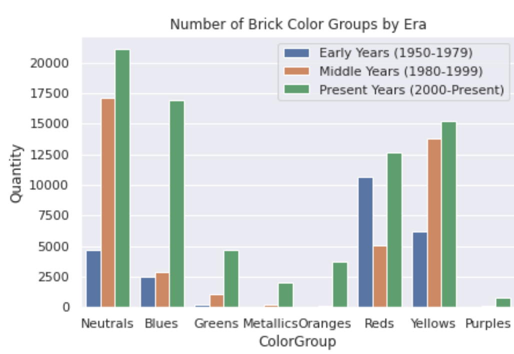
Quantities of each brick color group over each era can be seen as follows:







Blues, neutrals, and reds had the highest quantities overall. Purples and oranges only became notable in the 2000s and beyond. Another view of the changes in color group by era can be found below:



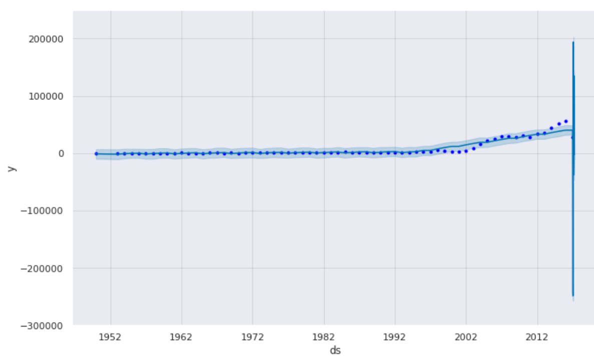
Analysis

Model: Time Series with Prophet Models

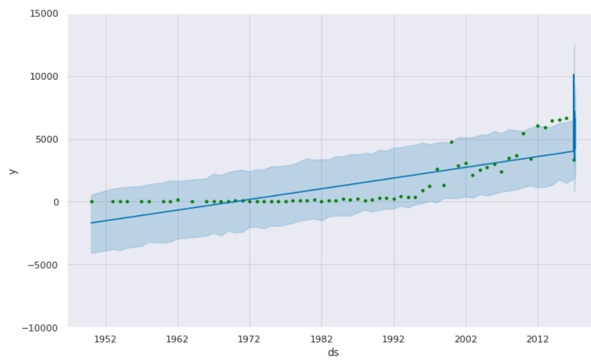
Prophet models were run on brick color quantities per year, for each of the higher level color groupings.

Predictions for Quantities of Each Brick Color Grouping

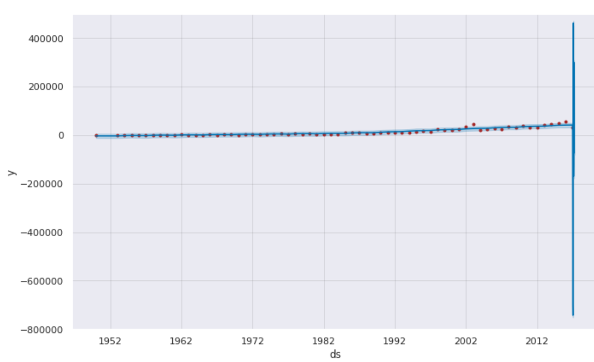
Blue



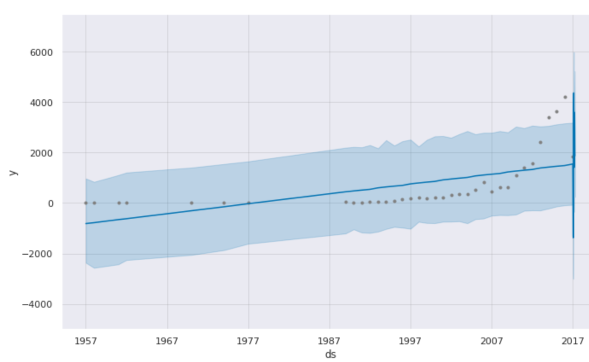
Green



Neutral

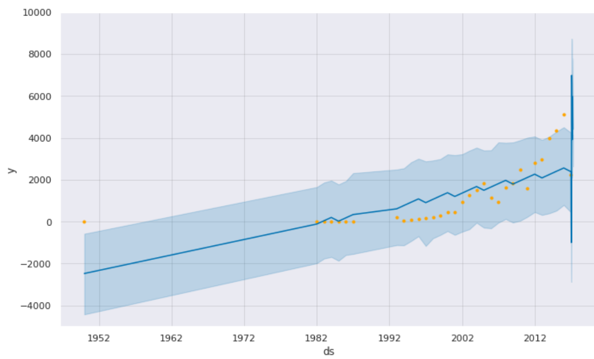


Metallic

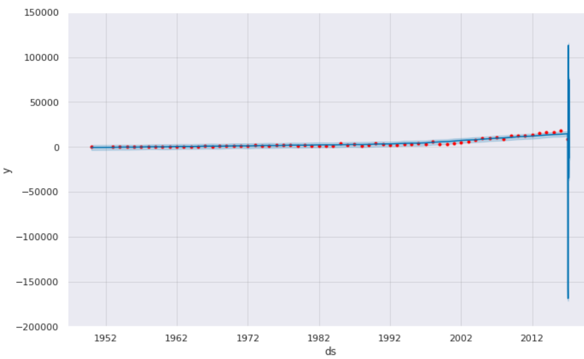


Prediction for Quantities of Each Brick Color Grouping (Continued)

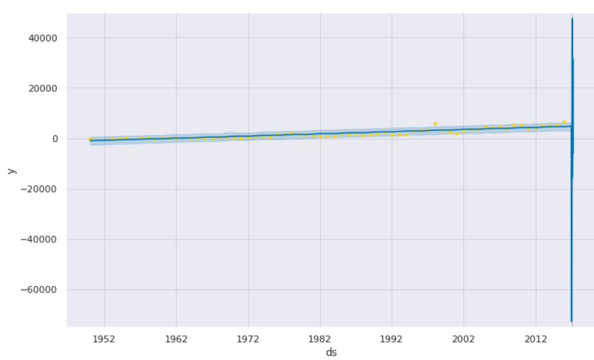
Orange



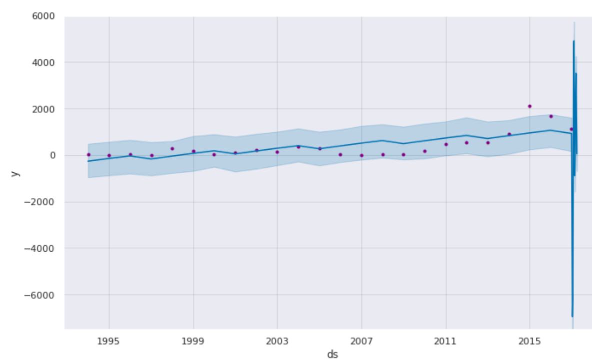
Red



Yellow

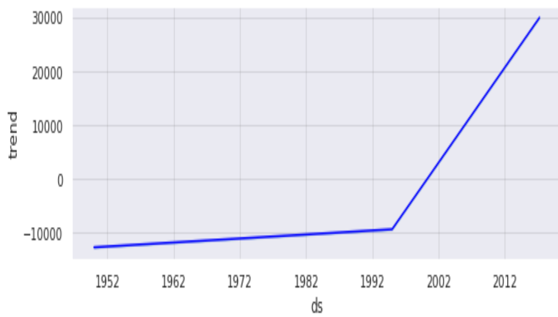


Purple

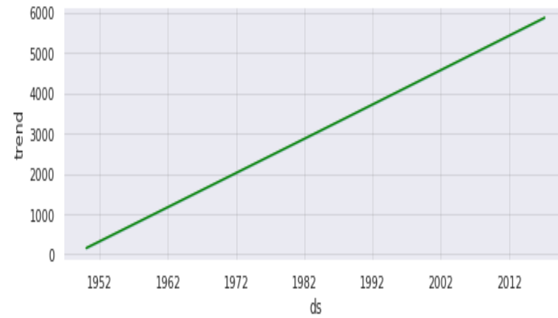


Trends for Brick Color Quantities

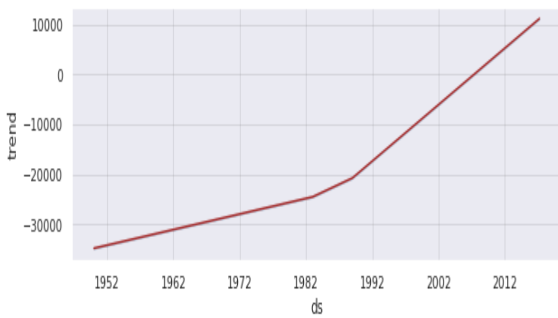
Blue



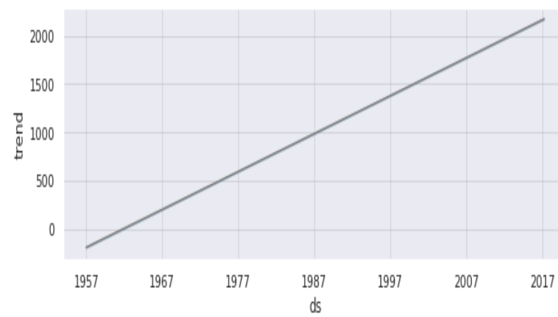
Green



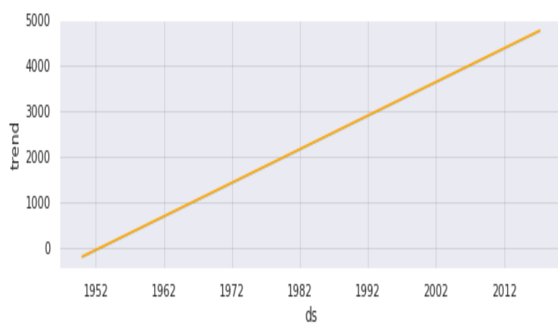
Neutral



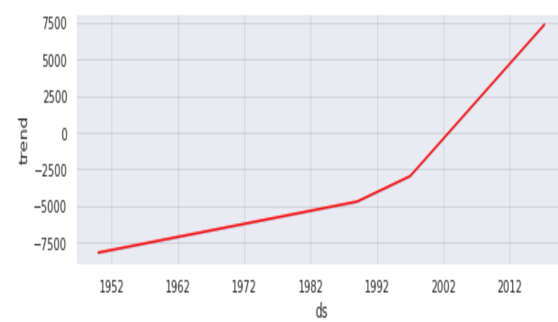
Metallic



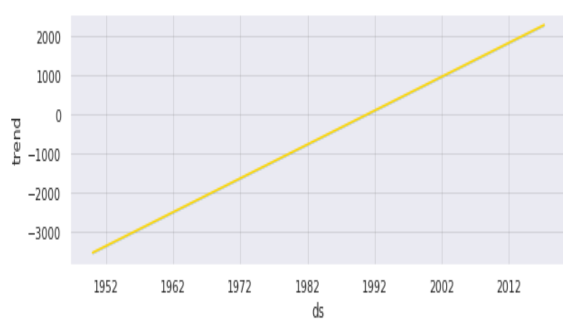
Orange



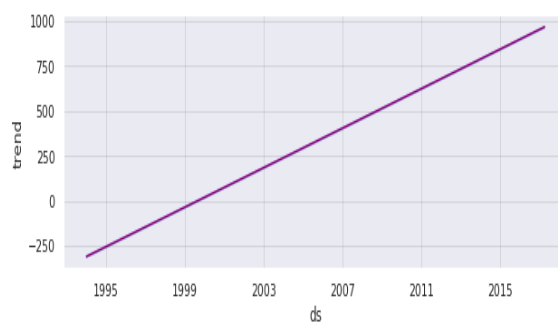
Red



Yellow



Purple



Interpretation: Prophet Models

Likely due to the yearly nature of the data, prophet model plots had high levels of variation for future prediction values for each color group. However, trends for each of these

Model: Support Vector Machine Models

SVM models were attempted to be run from the following python packages:

```
from sklearn import svm, datasets
import sklearn.model_selection as model_selection
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
```

The following accuracy and F1 score results were obtained from the SVM analysis for both polynomial kernel and RBF kernel.

```
[39] poly_accuracy = accuracy_score(y_test, poly_pred)
     poly_f1 = f1_score(y_test, poly_pred, average='weighted')
     print('Accuracy (Polynomial Kernel): ', "%.2f" % (poly_accuracy*100))
     print('F1 (Polynomial Kernel): ', "%.2f" % (poly_f1*100))

     Accuracy (Polynomial Kernel):  6.95
     F1 (Polynomial Kernel):  3.43
```

```
rbf_accuracy = accuracy_score(y_test, rbf_pred)
rbf_f1 = f1_score(y_test, rbf_pred, average='weighted')
print('Accuracy (RBF Kernel): ', "%.2f" % (rbf_accuracy*100))
print('F1 (RBF Kernel): ', "%.2f" % (rbf_f1*100))

Accuracy (RBF Kernel):  2.16
F1 (RBF Kernel):  0.09
```

Interpretation: Support Vector Machine Models

The support vector machine models unfortunately had very low accuracy and F1 score values, with unactionable results.

Recommendation

Time series trends indicate an increase in all brick color groupings over time. A higher increase in quantity was identified for blue, neutral, and red brick color groups. Based on the results obtained through this analysis, the next lego sets to be produced by the company should include all colors from previous sets, yet focus on increased quantities of blue, neutral, and red color groups.

Further analysis was desired, yet not achieved, due to time constraints of the project. There are additional analytical points to consider in future analysis. First, this analysis was based on prior inventory data. Future predictions would likely yield more specific and actionable results if sales specific data was available for integration, preferably with a more frequent cadence than yearly data, ideally daily. Other models could also be considered, including correlations to identify other factors than brick color that may have a relationship with quantities and sales. Additionally, other models may be considered to get a better understanding of clustering and groupings of potential attributes, such as random forest. Overall, this analysis was a strong effort in data science principles around integrating project management, data selection and cleansing, with business considerations and interpretations of the data and analytic results.

References

[1] https://www.lego.com/en-us/aboutus/lego-group/the-lego-group-history

[2] https://www.kaggle.com/datasets/rtatman/lego-database?select=colors.csv

[3] https://rebrickable.com/about/

[4] https://www.kaggle.com/datasets/rtatman/lego-database?select=downloads_schema.png

Appendices

Python code for this analysis is included in the following pages.

Lego Product Analysis

Authors: Christina DaSilva, Bourama Sidibe

Course: IST-718

Section: Sunday 7:30pm

Submission Date: 12-21-2022

OBTAIN

File Preparation

```
# THIS IS ONLY REQUIRED WHEN LOADING IN COLAB
# Mount Google Drive for file access
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Package Imports

```
# Package Imports
import pandas as pd # dataframes
import numpy as np  # arrays and math functions
import seaborn as sea # visuals
import matplotlib.pyplot as plt # plots
plt.style.use('fivethirtyeight')

# Other packages to consider:
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from scipy.stats import uniform  # for training-and-test split
import statsmodels.api as sm  # statistical models (including regression)
import statsmodels.formula.api as smf  # R-like model specification


# Prophet time series model
!pip install prophet
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: prophet in /usr/local/lib/python3.8/dist-packages (1.1.1)
Requirement already satisfied: wheel>=0.37.0 in /usr/local/lib/python3.8/dist-packages (from prophet) (0.38.4)
Requirement already satisfied: setuptools-git>=1.2 in /usr/local/lib/python3.8/dist-packages (from prophet) (1.2)
Requirement already satisfied: convertdate>=2.1.2 in /usr/local/lib/python3.8/dist-packages (from prophet) (2.4.0)
Requirement already satisfied: matplotlib>=2.0.0 in /usr/local/lib/python3.8/dist-packages (from prophet) (3.2.2)
Requirement already satisfied: cmdstanpy>=1.0.4 in /usr/local/lib/python3.8/dist-packages (from prophet) (1.0.8)
Requirement already satisfied: pandas>=1.0.4 in /usr/local/lib/python3.8/dist-packages (from prophet) (1.3.5)
Requirement already satisfied: setuptools>=42 in /usr/local/lib/python3.8/dist-packages (from prophet) (57.4.0)
Requirement already satisfied: tqdm>=4.36.1 in /usr/local/lib/python3.8/dist-packages (from prophet) (4.64.1)
Requirement already satisfied: LunarCalendar>=0.0.9 in /usr/local/lib/python3.8/dist-packages (from prophet) (0.0.9)
Requirement already satisfied: holidays>=0.14.2 in /usr/local/lib/python3.8/dist-packages (from prophet) (0.17.2)
Requirement already satisfied: python-dateutil>=2.8.0 in /usr/local/lib/python3.8/dist-packages (from prophet) (2.8.2)
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.8/dist-packages (from prophet) (1.21.6)
Requirement already satisfied: pymeeus<=1,>=0.3.13 in /usr/local/lib/python3.8/dist-packages (from convertdate>=2.1.2->proph
Requirement already satisfied: korean-lunar-calendar in /usr/local/lib/python3.8/dist-packages (from holidays>=0.14.2->proph
Requirement already satisfied: hijri-converter in /usr/local/lib/python3.8/dist-packages (from holidays>=0.14.2->prophet) (2
Requirement already satisfied: ephem>=3.7.5.3 in /usr/local/lib/python3.8/dist-packages (from LunarCalendar>=0.0.9->prophet)
Requirement already satisfied: pytz in /usr/local/lib/python3.8/dist-packages (from LunarCalendar>=0.0.9->prophet) (2022.6)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.8/dist-packages (from matp
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib>=2.0.0->prophet)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.8/dist-packages (from matplotlib>=2.0.0->prophet) (0.1
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.8/dist-packages (from python-dateutil>=2.8.0->prophet) (1.
```

```
# Prophet time series model
import timeit
from prophet import Prophet
```

Reading in Source Files

```
# Read in Lego data from csv files on Google Drive
colors_filename = 'drive/MyDrive/IST718/FinalProject/lego_files/colors.csv'
inventories_filename = 'drive/MyDrive/IST718/FinalProject/lego_files/inventories.csv'
inventory_parts_filename = 'drive/MyDrive/IST718/FinalProject/lego_files/inventory_parts.csv'
inventory_sets_filename = 'drive/MyDrive/IST718/FinalProject/lego_files/inventory_sets.csv'
part_categories_filename = 'drive/MyDrive/IST718/FinalProject/lego_files/part_categories.csv'
parts_filename = 'drive/MyDrive/IST718/FinalProject/lego_files/parts.csv'
sets_filename = 'drive/MyDrive/IST718/FinalProject/lego_files/sets.csv'
themes_filename = 'drive/MyDrive/IST718/FinalProject/lego_files/themes.csv'

# Create a dataframe for each .CSV file
colors_df = pd.read_csv(colors_filename)
inventories_df = pd.read_csv(inventories_filename)
inventory_parts_df = pd.read_csv(inventory_parts_filename)
inventory_sets_df = pd.read_csv(inventory_sets_filename)
part_categories_df = pd.read_csv(part_categories_filename)
parts_df = pd.read_csv(parts_filename)
sets_df = pd.read_csv(sets_filename)
themes_df = pd.read_csv(themes_filename)
```

About the Source Data

Initial Data Frames:

> *colors_df*
> *inventories_df*
> *inventory_parts_df*
> *inventory_sets_df*
> *part_categories_df*
> *parts_df*
> *sets_df*
> *themes_df*

```
# Print some information about the colors dataframe
# colors_df

colors_df.info()
print('Successfully read data into Colors Dataframe. Shape: {row} rows and {col} columns'.format(row = colors_df.shape[0], col = 
colors_df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 135 entries, 0 to 134
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   id        135 non-null    int64 
 1   name      135 non-null    object
 2   rgb       135 non-null    object
 3   is_trans  135 non-null    object
dtypes: int64(1), object(3)
memory usage: 4.3+ KB
Successfully read data into Colors Dataframe. Shape: 135 rows and 4 columns
```

	id	name	rgb	is_trans
0	-1	Unknown	0033B2	f
1	0	Black	05131D	f
2	1	Blue	0055BF	f
3	2	Green	237841	f
4	3	Dark Turquoise	008F9B	f

```
# Print some information about the inventories dataframe
# inventories_df
```

```
inventories_df.info()
print('Successfully read data into Inventories Dataframe. Shape: {row} rows and {col} columns'.format(row = inventories_df.shape[
inventories_df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11681 entries, 0 to 11680
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   id       11681 non-null  int64
 1   version  11681 non-null  int64
 2   set_num  11681 non-null  object
dtypes: int64(2), object(1)
memory usage: 273.9+ KB
Successfully read data into Inventories Dataframe. Shape: 11681 rows and 3 columns
```

	id	version	set_num
0	1	1	7922-1
1	3	1	3931-1
2	4	1	6942-1
3	15	1	5158-1
4	16	1	903-1

```
# Print some information about the inventory parts dataframe
# inventory_parts_df

inventory_parts_df.info()
print('Successfully read data into Inventory Parts Dataframe. Shape: {row} rows and {col} columns'.format(row = inventory_parts_d
inventory_parts_df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 580251 entries, 0 to 580250
Data columns (total 5 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   inventory_id  580251 non-null  int64
 1   part_num      580251 non-null  object
 2   color_id      580251 non-null  int64
 3   quantity      580251 non-null  int64
 4   is_spare      580251 non-null  object
dtypes: int64(3), object(2)
memory usage: 22.1+ MB
Successfully read data into Inventory Parts Dataframe. Shape: 580251 rows and 5 columns
```

	inventory_id	part_num	color_id	quantity	is_spare
0	1	48379c01	72	1	f
1	1	48395	7	1	f
2	1	mcsport6	25	1	f
3	1	paddle	0	1	f
4	3	11816pr0005	78	1	f

```
# Print some information about the inventory sets dataframe
# inventory_sets_df

inventory_sets_df.info()
print('Successfully read data into Inventory Sets Dataframe. Shape: {row} rows and {col} columns'.format(row = inventory_sets_df.
inventory_sets_df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2846 entries, 0 to 2845
Data columns (total 3 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   inventory_id  2846 non-null   int64
 1   set_num       2846 non-null   object
 2   quantity      2846 non-null   int64
dtypes: int64(2), object(1)
memory usage: 66.8+ KB
Successfully read data into Inventory Sets Dataframe. Shape: 2846 rows and 3 columns
```

	inventory_id	set_num	quantity
0	35	75911-1	1

```
# Print some information about the part categories dataframe
# part_categories_df

part_categories_df.info()
print('Successfully read data into Part Categories Dataframe. Shape: {row} rows and {col} columns'.format(row = part_categories_d
part_categories_df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 57 entries, 0 to 56
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   id      57 non-null     int64
 1   name    57 non-null     object
dtypes: int64(1), object(1)
memory usage: 1.0+ KB
Successfully read data into Part Categories Dataframe. Shape: 57 rows and 2 columns
```

	id	name
0	1	Baseplates
1	2	Bricks Printed
2	3	Bricks Sloped
3	4	Duplo, Quatro and Primo
4	5	Bricks Special

```
# Print some information about the parts dataframe
# parts_df

parts_df.info()
print('Successfully read data into Parts Dataframe. Shape: {row} rows and {col} columns'.format(row = parts_df.shape[0], col = pa
parts_df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25993 entries, 0 to 25992
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   part_num     25993 non-null  object
 1   name         25993 non-null  object
 2   part_cat_id  25993 non-null  int64
dtypes: int64(1), object(2)
memory usage: 609.3+ KB
Successfully read data into Parts Dataframe. Shape: 25993 rows and 3 columns
```

	part_num	name	part_cat_id
0	0687b1	Set 0687 Activity Booklet 1	17
1	0901	Baseplate 16 x 30 with Set 080 Yellow House Print	1
2	0902	Baseplate 16 x 24 with Set 080 Small White Hou...	1
3	0903	Baseplate 16 x 24 with Set 080 Red House Print	1
4	0904	Baseplate 16 x 24 with Set 080 Large White Hou...	1

```
# Print some information about the sets dataframe
# sets_df
```

```
sets_df.info()
print('Successfully read data into Sets Dataframe. Shape: {row} rows and {col} columns'.format(row = sets_df.shape[0], col = sets_
sets_df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11673 entries, 0 to 11672
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   set_num    11673 non-null  object
 1   name       11673 non-null  object
 2   year       11673 non-null  int64 
 3   theme_id   11673 non-null  int64 
 4   num_parts  11673 non-null  int64 
dtypes: int64(3), object(2)
memory usage: 456.1+ KB
Successfully read data into Sets Dataframe. Shape: 11673 rows and 5 columns
```

	set_num	name	year	theme_id	num_parts
0	00-1	Weetabix Castle	1970	414	471
1	0011-2	Town Mini-Figures	1978	84	12
2	0011-3	Castle 2 for 1 Bonus Offer	1987	199	2
3	0012-1	Space Mini-Figures	1979	143	12
4	0013-1	Space Mini-Figures	1979	143	12

```
# Print some information about the themes dataframe
# themes_df

themes_df.info()
print('Successfully read data into Themes Dataframe. Shape: {row} rows and {col} columns'.format(row = themes_df.shape[0], col = 
themes_df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   id         614 non-null    int64  
 1   name       614 non-null    object 
 2   parent_id  503 non-null    float64
dtypes: float64(1), int64(1), object(1)
memory usage: 14.5+ KB
Successfully read data into Themes Dataframe. Shape: 614 rows and 3 columns
```

	id	name	parent_id
0	1	Technic	NaN
1	2	Arctic Technic	1.0
2	3	Competition	1.0
3	4	Expert Builder	1.0
4	5	Model	1.0

SCRUB

Merging Data Frames

```
# Merge dataframes to match up part color data with inventory by year
# colors ->  inventory parts -> inventories -> sets
#
# Use the following IDs to merge:
# Colors (colors.id to inventory_parts.color_id) ->
# Inventory Parts (inventory_parts.inventory_id to inventories.id) ->
# Inventories (inventories.set_num to sets.set_num) ->
# Sets
```

```
# Merge Step 1: Merge part colors and inventory parts
color_invpart_df = pd.merge(colors_df, inventory_parts_df, left_on='id', right_on='color_id').groupby(['name','quantity','invento
color_invpart_df.head()
```

	name	inventory_id	quantity
0	Aqua	747	2
1	Aqua	1286	1
2	Aqua	1307	1
3	Aqua	1853	1
4	Aqua	2688	1

```
# Merge Step 2: Merge with inventories
color_invpart_inventories_df = pd.merge(color_invpart_df, inventories_df, left_on='inventory_id', right_on='id').groupby(['name',
color_invpart_inventories_df#.head()
```

	name	inventory_id	id	set_num	quantity
0	Aqua	1286	1286	7524-1	1
1	Aqua	1307	1307	10829-1	1
2	Aqua	1853	1853	5836-1	1
3	Aqua	2688	2688	7549-1	1
4	Aqua	3643	3643	1385-1	1
...	...	...	...	...	...
213960	[No Color]	216	216	1089-1	24
213961	[No Color]	6455	6455	7417-1	24
213962	[No Color]	12078	12078	9631-1	24
213963	[No Color]	3168	3168	7418-1	25
213964	[No Color]	6917	6917	7419-1	39

213965 rows × 5 columns

```
# Merge Step 3: Merge with sets

color_invpart_inventories_sets_df = pd.merge(color_invpart_inventories_df, sets_df, left_on='set_num', right_on='set_num')

###### Set aside new df to use for prophet analysis
color_prophet_df = color_invpart_inventories_sets_df
color_prophet_df.rename(columns={'year':'Year', 'name_x':'Color', 'quantity':'Quantity'})

# Continue merging color qty by year dataframe
color_invpart_inventories_sets_df = color_invpart_inventories_sets_df.groupby(['year','name_x'], as_index=False).agg({'quantity':
color_qty_year_df = color_invpart_inventories_sets_df
color_qty_year_df=color_qty_year_df.rename(columns={'year':'Year', 'name_x': 'Color','quantity':'Quantity'})

color_qty_year_df
```

	Year	Color	Quantity
2	1950	Green	6

Feature Generation

Timeframe groupings (Decades and Eras)

```
# Feature creation: Decade attribute
# Adapted from: https://www.dataquest.io/blog/tutorial-add-column-pandas-dataframe-based-on-if-else-condition/

# create a new dataframe with colors quantities by decade
color_qty_decade_df=color_qty_year_df

# create a list of our conditions
conditions = [
    (color_qty_year_df['Year'] <= 1959),
    (color_qty_year_df['Year'] > 1959) & (color_qty_year_df['Year'] < 1970),
    (color_qty_year_df['Year'] > 1969) & (color_qty_year_df['Year'] < 1980),
    (color_qty_year_df['Year'] > 1979) & (color_qty_year_df['Year'] < 1990),
    (color_qty_year_df['Year'] > 1989) & (color_qty_year_df['Year'] < 2000),
    (color_qty_year_df['Year'] > 1999) & (color_qty_year_df['Year'] < 2010),
    (color_qty_year_df['Year'] > 2009)
    ]

# create a list of the values we want to assign for each condition
values = ['1950s', '1960s', '1970s', '1980s','1990s','2000s','2010s']
#values = [1950, 1960, 1970, 1980, 1990, 2000, 2010]

# create a new column and use np.select to assign values to it using our lists as arguments
color_qty_decade_df['Decade'] = np.select(conditions, values)

color_qty_decade_df = color_qty_decade_df.filter(['Color','Quantity','Decade'])

color_qty_decade_df = color_qty_decade_df.groupby(['Decade','Color'])['Quantity'].sum().to_frame().reset_index()

# display updated DataFrame
#color_qty_decade_df.head(10)
color_qty_decade_df.tail(50)
```

	Decade	Color	Quantity
299	2010s	Light Gray	48
300	2010s	Light Green	3
301	2010s	Light Lime	5
302	2010s	Light Purple	2
303	2010s	Light Yellow	3
304	2010s	Lime	10445
305	2010s	Maersk Blue	404
306	2010s	Magenta	2677
307	2010s	Medium Azure	3617
308	2010s	Medium Blue	5096
309	2010s	Medium Dark Flesh	3866
310	2010s	Medium Lavender	1972
311	2010s	Metallic Gold	180
312	2010s	Metallic Silver	1028
313	2010s	Milky White	1
314	2010s	Olive Green	2031
315	2010s	Orange	10762
316	2010s	Pearl Dark Gray	1914
317	2010s	Pearl Gold	10278
318	2010s	Pearl Light Gray	345
319	2010s	Purple	2
320	2010s	Red	49755
321	2010s	Reddish Brown	35965
322	2010s	Royal Blue	8
323	2010s	Sand Blue	1124
324	2010s	Sand Green	2198
325	2010s	Speckle Black-Gold	5
326	2010s	Speckle Black-Silver	28
327	2010s	Tan	37882
328	2010s	Trans-Black	3769

```
# Feature creation: Era attribute
# Adapted from: https://www.dataquest.io/blog/tutorial-add-column-pandas-dataframe-based-on-if-else-condition/

# create a new dataframe with colors quantities by era (three groupings)
color_qty_eras_df=color_qty_year_df

# create a list of our conditions
conditions = [
    (color_qty_year_df['Year'] <= 1979),
    (color_qty_year_df['Year'] > 1979) & (color_qty_year_df['Year'] < 2000),
    (color_qty_year_df['Year'] > 1999)
    ]

# create a list of the values we want to assign for each condition
values = ['Early Years (1950-1979)', 'Middle Years (1980-1999)', 'Present Years (2000-Present)']

# create a new column and use np.select to assign values to it using our lists as arguments
color_qty_eras_df['Era'] = np.select(conditions, values)

color_qty_eras_df = color_qty_eras_df.filter(['Color','Quantity','Era'])

color_qty_eras_df = color_qty_eras_df.groupby(['Era','Color'])['Quantity'].sum().to_frame().reset_index()

# display updated DataFrame
#color_qty_eras_df.head(10)
```

```
color_qty_eras_df.tail(50)
```

	Era	Color	Quantity
164	Present Years (2000-Present)	Pearl Dark Gray	2325
165	Present Years (2000-Present)	Pearl Gold	11002
166	Present Years (2000-Present)	Pearl Light Gold	29
167	Present Years (2000-Present)	Pearl Light Gray	3507
168	Present Years (2000-Present)	Pearl Very Light Gray	5

Generic Color Mapping and Groupings

```
# Get all unique colors to create color group mapping
unique_colors = color_qty_year_df.Color.unique()

unique_colors_df = pd.DataFrame(unique_colors, columns = ['UniqueColors'])

unique_colors_df.to_string()
```

```
'                UniqueColors\n0                       Blue\n1              Bright Green\n2                      Green\
reen\n4                Medium Blue\n5              Medium Orange\n6                        Red\n7                 Trans-C
White\n9                  Yellow\n10             Light Gray\n11             [No Color]\n12                    Royal
Black\n14           Metallic Silver\n15            Milky White\n16            Trans-Green\n17                     Tran
ans-Yellow\n19                   Brown\n20             Maersk Blue\n21             Chrome Silver\n22
Trans-Dark Blue\n24                 Unknown\n25             Earth Orange\n26             Fabuland Brown\n27
Tan\n29           Trans-Light Blue\n30          Trans-Neon Green\n31      ...'
```

182	Present Years (2000-Present)	Speckle Black-Gold	5

```
# Map each unique color to a more general color group to reduce the number of individual colors
# Color groupings include:
#     Blues, Greens, Metallics, Neutrals, Oranges, Purples, Reds, Yellows

color_mapping = {'Blue':'Blues', 'Bright Green':'Greens', 'Green':'Greens', 'Light Green':'Greens', 'Medium Blue':'Blues', 'Mediu
color_mapping
```

```
{'Blue': 'Blues',
 'Bright Green': 'Greens',
 'Green': 'Greens',
 'Light Green': 'Greens',
 'Medium Blue': 'Blues',
 'Medium Orange': 'Oranges',
 'Red': 'Reds',
 'Trans-Clear': 'Neutrals',
 'White': 'Neutrals',
 'Yellow': 'Yellows',
 'Light Gray': 'Neutrals',
 '[No Color]': 'Neutrals',
 'Royal Blue': 'Blues',
 'Black': 'Neutrals',
 'Metallic Silver': 'Metallics',
 'Milky White': 'Neutrals',
 'Trans-Green': 'Greens',
 'Trans-Red': 'Reds',
 'Trans-Yellow': 'Yellows',
 'Brown': 'Neutrals',
 'Maersk Blue': 'Blues',
 'Chrome Silver': 'Metallics',
 'Dark Gray': 'Neutrals',
 'Trans-Dark Blue': 'Blues',
 'Unknown': 'Neutrals',
 'Earth Orange': 'Oranges',
 'Fabuland Brown': 'Neutrals',
 'Lime': 'Greens',
 'Tan': 'Neutrals',
 'Trans-Light Blue': 'Blues',
 'Trans-Neon Green': 'Greens',
 'Chrome Gold': 'Metallics',
 'Glow In Dark Opaque': 'Metallics',
 'Pink': 'Reds',
 'Bright Light Blue': 'Blues',
 'Medium Dark Pink': 'Reds',
 'Orange': 'Oranges',
 'Reddish Brown': 'Reds',
 'Trans-Neon Orange': 'Oranges',
 'Dark Pink': 'Reds',
```

```
        'Light Pink': 'Reds',
        'Light Violet': 'Purples',
        'Light Yellow': 'Yellows',
        'Medium Green': 'Greens',
        'Light Blue': 'Blues',
        'Purple': 'Purples',
        'Light Salmon': 'Oranges',
        'Metallic Gold': 'Metallics',
        'Rust': 'Reds',
        'Salmon': 'Oranges',
        'Chrome Blue': 'Blues',
        'Dark Orange': 'Oranges',
        'Dark Turquoise': 'Blues',
        'Light Bluish Gray': 'Blues',
        'Chrome Green': 'Greens',
        'Glitter Trans-Dark Pink': 'Metallics',
        'Light Turquoise': 'Blues',
        'Medium Lime': 'Greens',

# Function for assignment of broader color groupings
# Adapted from: https://stackoverflow.com/questions/62567406/pandas-check-if-a-substring-exists-in-another-column-then-create-a-n

def check_color(x):
    for key in color_mapping:
        if key.lower() in x.lower():
            return color_mapping[key]
    return ''


# Assign color group to color breakdown by Era
color_qty_eras_df['ColorGroup']  = color_qty_eras_df['Color'] .map(lambda x: check_color(x))
colorgroup_qty_eras_df = color_qty_eras_df.groupby(['Era','ColorGroup'], as_index=False).agg({'Quantity':sum})
colorgroup_qty_eras_df
```

	Era	ColorGroup	Quantity
0	Early Years (1950-1979)	Blues	12649
1	Early Years (1950-1979)	Greens	894
2	Early Years (1950-1979)	Metallics	34
3	Early Years (1950-1979)	Neutrals	42039
4	Early Years (1950-1979)	Oranges	2
5	Early Years (1950-1979)	Reds	21393
6	Early Years (1950-1979)	Yellows	12472
7	Middle Years (1980-1999)	Blues	34314
8	Middle Years (1980-1999)	Greens	9489
9	Middle Years (1980-1999)	Metallics	1113
10	Middle Years (1980-1999)	Neutrals	205438
11	Middle Years (1980-1999)	Oranges	1152
12	Middle Years (1980-1999)	Purples	567
13	Middle Years (1980-1999)	Reds	50977
14	Middle Years (1980-1999)	Yellows	41390
15	Present Years (2000-Present)	Blues	473471
16	Present Years (2000-Present)	Greens	74492
17	Present Years (2000-Present)	Metallics	24107
18	Present Years (2000-Present)	Neutrals	611543
19	Present Years (2000-Present)	Oranges	37455
20	Present Years (2000-Present)	Purples	8934
21	Present Years (2000-Present)	Reds	189396
22	Present Years (2000-Present)	Yellows	75857

```
# Assign color group to color breakdown by Era
color_qty_eras_df['ColorGroup']  = color_qty_eras_df['Color'].map(lambda x: check_color(x))
colorgroup_qty_eras_df = color_qty_eras_df.groupby(['Era','ColorGroup'], as_index=False).agg({'Quantity':sum})
colorgroup_qty_eras_df
```

	Era	ColorGroup	Quantity
0	Early Years (1950-1979)	Blues	12649
1	Early Years (1950-1979)	Greens	894
2	Early Years (1950-1979)	Metallics	34
3	Early Years (1950-1979)	Neutrals	42039
4	Early Years (1950-1979)	Oranges	2
5	Early Years (1950-1979)	Reds	21393
6	Early Years (1950-1979)	Yellows	12472
7	Middle Years (1980-1999)	Blues	34314
8	Middle Years (1980-1999)	Greens	9489
9	Middle Years (1980-1999)	Metallics	1113
10	Middle Years (1980-1999)	Neutrals	205438
11	Middle Years (1980-1999)	Oranges	1152
12	Middle Years (1980-1999)	Purples	567
13	Middle Years (1980-1999)	Reds	50977
14	Middle Years (1980-1999)	Yellows	41390
15	Present Years (2000-Present)	Blues	473471
16	Present Years (2000-Present)	Greens	74492
17	Present Years (2000-Present)	Metallics	24107
18	Present Years (2000-Present)	Neutrals	611543
19	Present Years (2000-Present)	Oranges	37455
20	Present Years (2000-Present)	Purples	8934
21	Present Years (2000-Present)	Reds	189396
22	Present Years (2000-Present)	Yellows	75857

```
# Assign color group to color breakdown by Decade
color_qty_decade_df['ColorGroup']  = color_qty_decade_df['Color'].map(lambda x: check_color(x))
colorgroup_qty_decade_df = color_qty_decade_df.groupby(['Decade','ColorGroup'], as_index=False).agg({'Quantity':sum})
colorgroup_qty_decade_df
```

	Decade	ColorGroup	Quantity
0	1950s	Blues	560
1	1950s	Greens	47
2	1950s	Metallics	12
3	1950s	Neutrals	2368
4	1950s	Oranges	2
5	1950s	Reds	982
6	1950s	Yellows	431
7	1960s	Blues	3204
8	1960s	Greens	262
9	1960s	Metallics	2
10	1960s	Neutrals	10387
11	1960s	Reds	5398
12	1960s	Yellows	1748
13	1970s	Blues	8885
14	1970s	Greens	585
15	1970s	Metallics	20
16	1970s	Neutrals	29284
17	1970s	Reds	15013
18	1970s	Yellows	10293
19	1980s	Blues	11697
20	1980s	Greens	1329
21	1980s	Metallics	64
22	1980s	Neutrals	60346
23	1980s	Oranges	22
24	1980s	Reds	17779
25	1980s	Yellows	13997
26	1990s	Blues	22617
27	1990s	Greens	8160
28	1990s	Metallics	1049
29	1990s	Neutrals	145092
30	1990s	Oranges	1130
31	1990s	Purples	567
32	1990s	Reds	33198
33	1990s	Yellows	27393
34	2000s	Blues	167183

```
# Assign color group to color breakdown by Year
color_qty_year_df['ColorGroup']  = color_qty_year_df['Color'].map(lambda x: check_color(x))
colorgroup_qty_year_df = color_qty_year_df.groupby(['Year','ColorGroup'], as_index=False).agg({'Quantity':sum})
colorgroup_qty_year_df
```

	Year	ColorGroup	Quantity
0	1950	Blues	8
1	1950	Greens	12
2	1950	Neutrals	25
3	1950	Oranges	2
4	1950	Reds	12

```
# Assign color group to prophet model dataframe: color breakdown by Year
color_prophet_df['ColorGroup']  = color_qty_year_df['Color'].map(lambda x: check_color(x))
color_prophet_df = color_qty_year_df.groupby(['Year','ColorGroup'], as_index=False).agg({'Quantity':sum})
color_prophet_df
```

	Year	ColorGroup	Quantity
0	1950	Blues	8
1	1950	Greens	12
2	1950	Neutrals	25
3	1950	Oranges	2
4	1950	Reds	12
...	...	...	...
411	2017	Neutrals	30766
412	2017	Oranges	2226
413	2017	Purples	1127
414	2017	Reds	8799
415	2017	Yellows	2478

416 rows × 3 columns

```
early_era_qty_df = colorgroup_qty_eras_df.loc[colorgroup_qty_eras_df['Era'] == 'Early Years (1950-1979)'].drop(columns=['Era'])
early_era_qty_df
```

	ColorGroup	Quantity
0	Blues	12649
1	Greens	894
2	Metallics	34
3	Neutrals	42039
4	Oranges	2
5	Reds	21393
6	Yellows	12472

```
mid_era_qty_df = colorgroup_qty_eras_df.loc[colorgroup_qty_eras_df['Era'] == 'Middle Years (1980-1999)']
mid_era_qty_df
```

	Era	ColorGroup	Quantity
7	Middle Years (1980-1999)	Blues	34314
8	Middle Years (1980-1999)	Greens	9489
9	Middle Years (1980-1999)	Metallics	1113
10	Middle Years (1980-1999)	Neutrals	205438
11	Middle Years (1980-1999)	Oranges	1152
12	Middle Years (1980-1999)	Purples	567
13	Middle Years (1980-1999)	Reds	50977
14	Middle Years (1980-1999)	Yellows	41390

```
present_era_qty_df = colorgroup_qty_eras_df.loc[colorgroup_qty_eras_df['Era'] == 'Present Years (2000-Present)']
present_era_qty_df
```

	Era	ColorGroup	Quantity
15	Present Years (2000-Present)	Blues	473471
16	Present Years (2000-Present)	Greens	74492
17	Present Years (2000-Present)	Metallics	24107
18	Present Years (2000-Present)	Neutrals	611543
19	Present Years (2000-Present)	Oranges	37455
20	Present Years (2000-Present)	Purples	8934
21	Present Years (2000-Present)	Reds	189396
22	Present Years (2000-Present)	Yellows	75857

EXPLORE

Initial Data Frames

> *colors_df*
> *inventories_df*
> *inventory_parts_df*
> *inventory_sets_df*
> *part_categories_df*
> *parts_df*
> *sets_df*
> *themes_df*

Scrubbed Dataframes for Analysis

> *color_qty_year_df* - contains the quantity of each color by each year
> *color_qty_decade_df* - contains the quantity of each color by each decade
> *color_qty_eras_df* - contains the quantity of each color by three eras:
>
> - Early Years (1950-1979)
> - Middle Years (1980-1999)
> - Present Years (2000-2017)

> *colorgroup_qty_year_df* - contains the quantity of each broader color group by year
> *colorgroup_qty_decade_df* - contains the quantity of each broader color group by decade
> *colorgroup_qty_eras_df* - contains the quantity of each broader color group by each era

> *early_era_qty_df* - contains the quantity of each color group in the early era (1950-1979)
> *mid_era_qty_df* - contains the quantity of each color group in the mid era (1980-1999)
> *present_era_qty_df* - contains the quantity of each color group in the present era (2000-2017)

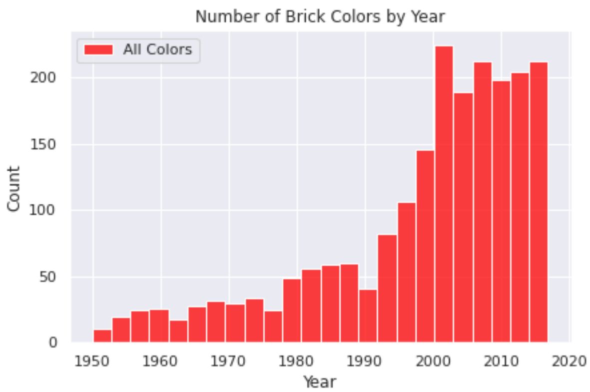
Exploratory Data Analysis

```
# Info on color_qty_eras_df
color_qty_eras_df.info()
```

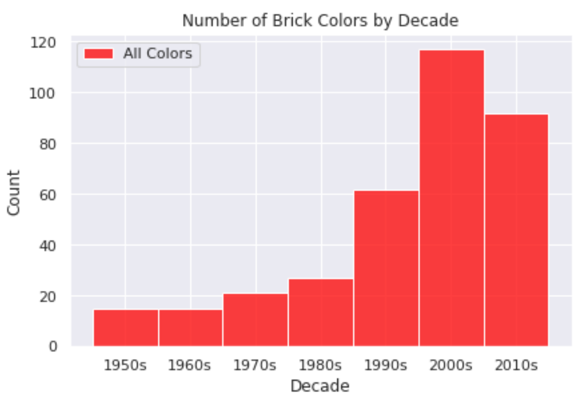
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 214 entries, 0 to 213
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Era         214 non-null    object
 1   Color       214 non-null    object
 2   Quantity    214 non-null    int64
```

```
 3   ColorGroup  214 non-null    object
dtypes: int64(1), object(3)
memory usage: 6.8+ KB
```

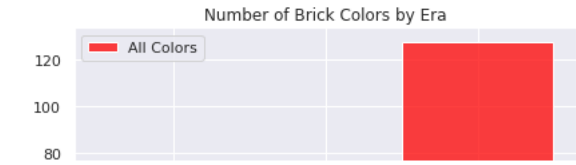
```
# Visualize the total number of brick colors by year

# Set a gray background
sea.set(style="darkgrid")

sea.histplot(data=color_qty_year_df, x="Year", color="red", label="All Colors")#, kde=True)

plt.title('Number of Brick Colors by Year')
plt.legend()
plt.show()
```



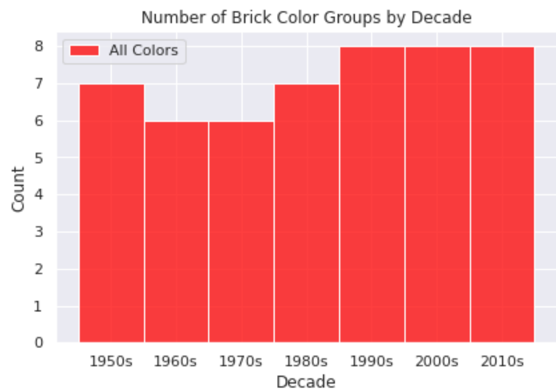

```
# Visualize the total number of brick colors by decade
sea.histplot(data=color_qty_decade_df, x="Decade", color="red", label="All Colors")#, kde=True)
plt.title('Number of Brick Colors by Decade')
plt.legend()
plt.show()
```



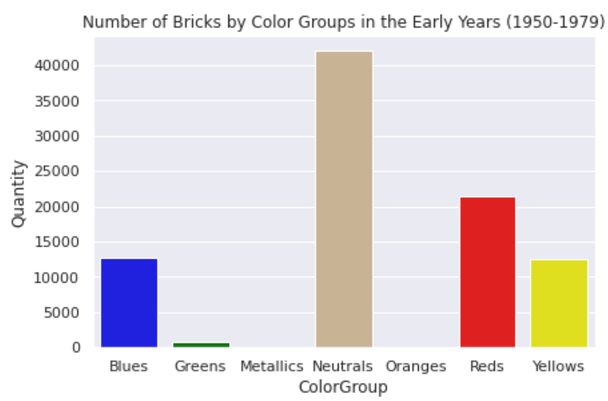

```
# Visualize the total number of brick colors by era
sea.histplot(data=color_qty_eras_df, x="Era", color="red", label="All Colors")#, kde=True)
plt.title('Number of Brick Colors by Era')
plt.xticks(rotation=30)
plt.legend()
plt.show()
```



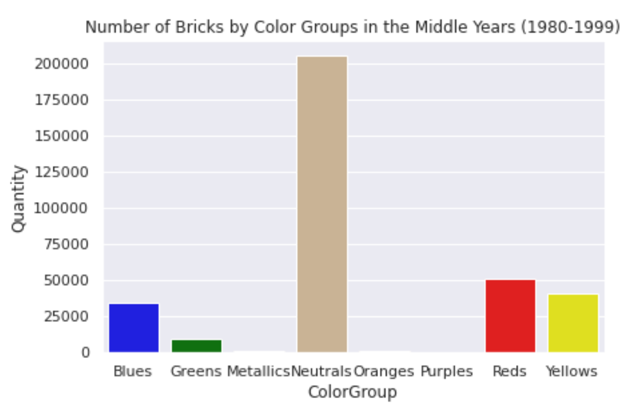

```
# Visualize the total number of brick color groups by decade
sea.histplot(data=colorgroup_qty_decade_df, x="Decade", color="red", label="All Colors")#, kde=True)
plt.title('Number of Brick Color Groups by Decade')
plt.legend()
plt.show()
```



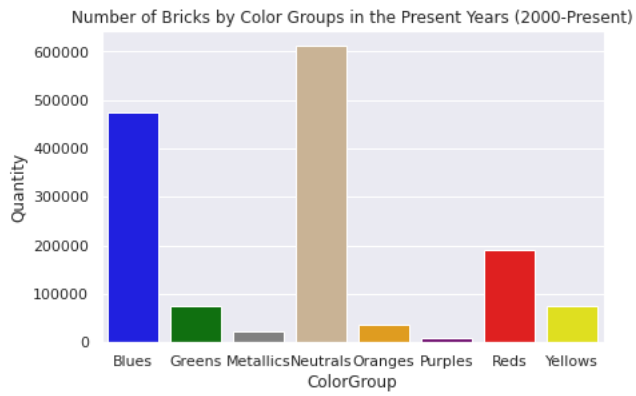

```
# Visualize the number of bricks per color groups in the early era
clrs = ['Blue','Green', 'Grey', 'Tan', 'Orange','Red','Yellow']
sea.barplot(data=early_era_qty_df, x='ColorGroup', y='Quantity',palette=clrs)
plt.title('Number of Bricks by Color Groups in the Early Years (1950-1979)')
plt.show()
```



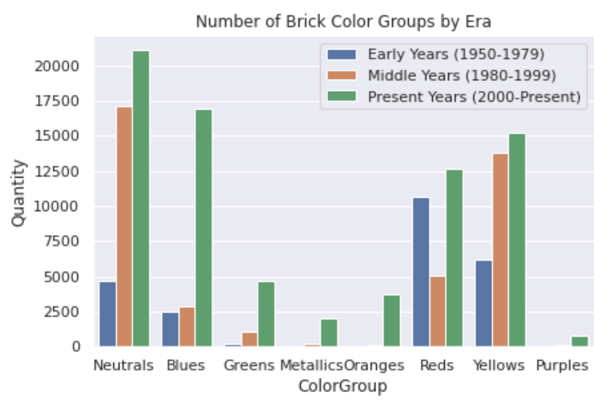

```
# Visualize the number of bricks per color groups in the mid era
clrs = ['Blue','Green', 'Grey', 'Tan', 'Orange','Purple','Red','Yellow']
sea.barplot(data=mid_era_qty_df, x='ColorGroup', y='Quantity', palette=clrs)
plt.title('Number of Bricks by Color Groups in the Middle Years (1980-1999)')
plt.show()
```



```
# Visualize the number of bricks per color groups in the present era
clrs = ['Blue','Green', 'Grey', 'Tan', 'Orange','Purple','Red','Yellow']
sea.barplot(data=present_era_qty_df, x='ColorGroup', y='Quantity',palette=clrs)
plt.title('Number of Bricks by Color Groups in the Present Years (2000-Present)')
plt.show()
```




```
# Color group quantities broken down by Era
sea.barplot(data=color_qty_eras_df, x="ColorGroup", y="Quantity", hue='Era', ci=None)
plt.title('Number of Brick Color Groups by Era')
plt.legend()
plt.show()
```



MODEL

Time Series Model

```
# Convert year data to date format for model
color_prophet_df['Date']=pd.to_datetime(color_prophet_df.Year, format='%Y')
color_prophet_df
```

	Year	ColorGroup	Quantity	Date
0	1950	Blues	8	1950-01-01
3	1950	Oranges	2	1950-01-01

Time Series Model for Quantities of Blue Legos

SCRUB

```
# Rename columns for prophet
prophet_blue_df = color_prophet_df.rename(index=str, columns={"Quantity": "y", "Date": "ds"})
prophet_blue_df = prophet_blue_df[prophet_blue_df['ColorGroup']=='Blues']
prophet_blue_df = prophet_blue_df.drop(columns = ['ColorGroup','Year'])
prophet_blue_df
```

	y	ds
0	8	1950-01-01
6	1	1953-01-01
11	13	1954-01-01
16	185	1955-01-01
21	6	1956-01-01
...	...	...
376	35375	2013-01-01
384	44146	2014-01-01
392	51010	2015-01-01
400	55609	2016-01-01
408	28110	2017-01-01

66 rows × 2 columns

MODEL

```
# Set the uncertainty interval to 95% (the Prophet default is 80%)
blue_prophet_model = Prophet(interval_width=0.95)
blue_prophet_model.fit(prophet_blue_df)
```

```
INFO:prophet:Disabling weekly seasonality. Run prophet with weekly_seasonality=True to override this.
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmpy7zff7ui/keubhxqw.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpy7zff7ui/pomv0qsy.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.8/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'see
07:20:59 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
07:20:59 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
<prophet.forecaster.Prophet at 0x7f58ca8153d0>
```

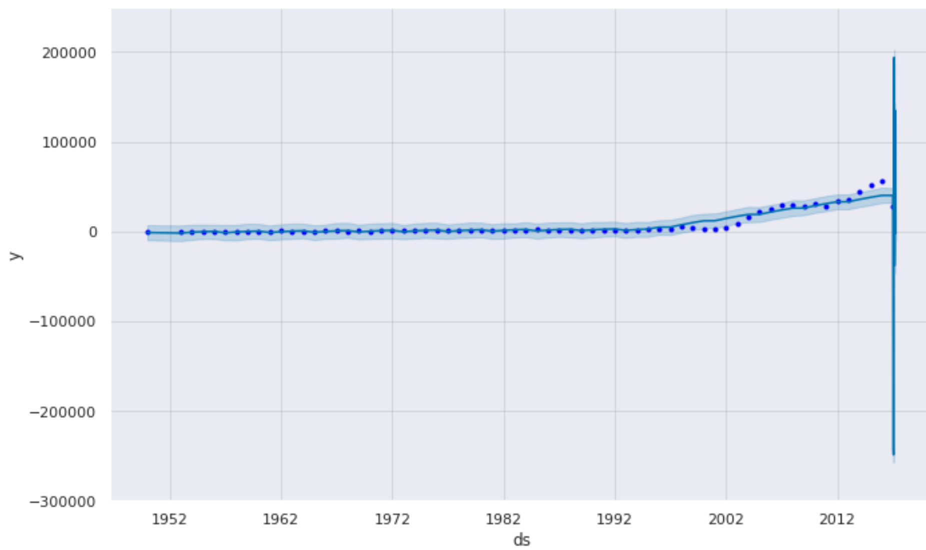
```
test_blue_future_dates = blue_prophet_model.make_future_dataframe(periods=13, freq='W')
test_blue_future_dates.head()
```

	ds
0	1950-01-01
1	1953-01-01
2	1954-01-01
3	1955-01-01
4	1956-01-01

```
forecast_blue_prophet = blue_prophet_model.predict(test_blue_future_dates)
forecast_blue_prophet[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

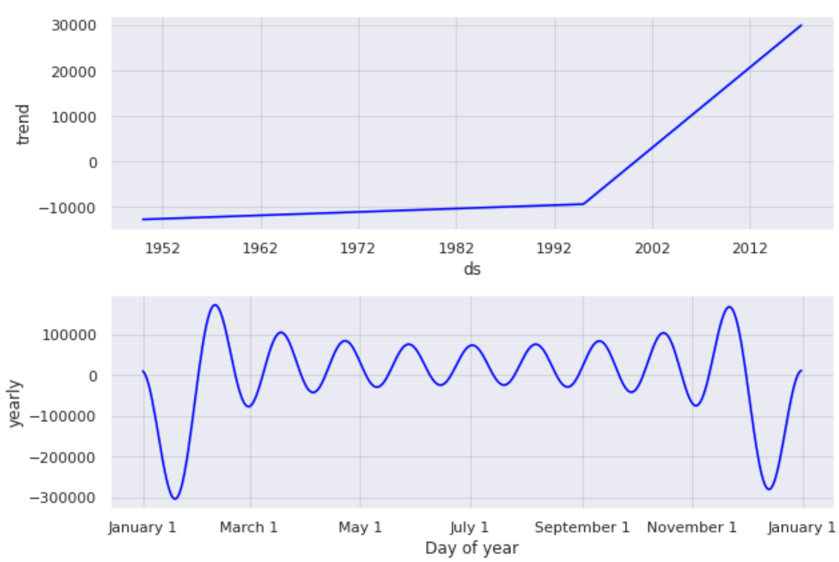
	ds	yhat	yhat_lower	yhat_upper
74	2017-03-05	-16677.007200	-25553.523332	-8941.610759
75	2017-03-12	87694.458301	79636.128499	95625.243422
76	2017-03-19	134877.964173	126766.085580	142883.468347
77	2017-03-26	75354.945661	66922.187156	84412.396249
78	2017-04-02	-1973.621150	-10109.098583	5978.665920

```
blue_prophet_model.plot(forecast_blue_prophet, uncertainty=True)
plt.ylim(-300000, 250000)
plt.gca().get_lines()[0].set_color("blue")
```



INTERPRET

```
blue_prophet_model.plot_components(forecast_blue_prophet)
for ax in plt.gcf().axes:
    ax.get_lines()[0].set_color("blue")
```



Time Series Model for Quantities of Green Colored Legos

SCRUB

```
# Rename columns for prophet
prophet_green_df = color_prophet_df.rename(index=str, columns={"Quantity": "y", "Date": "ds"})
prophet_green_df = prophet_green_df[prophet_green_df['ColorGroup']=='Greens']
prophet_green_df = prophet_green_df.drop(columns = ['ColorGroup','Year'])
prophet_green_df
```

	y	ds
1	12	1950-01-01
7	13	1953-01-01
12	4	1954-01-01
17	6	1955-01-01
25	4	1957-01-01
...	...	...
377	5896	2013-01-01
385	6426	2014-01-01
393	6553	2015-01-01
401	6637	2016-01-01
409	3335	2017-01-01

62 rows × 2 columns

MODEL

```
# Set the uncertainty interval to 95% (the Prophet default is 80%)
green_prophet_model = Prophet(interval_width=0.95)
green_prophet_model.fit(prophet_green_df)
```

```
INFO:prophet:Disabling weekly seasonality. Run prophet with weekly_seasonality=True to override this.
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmpy7zff7ui/zbhg_2ds.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpy7zff7ui/8x1vbzmm.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.8/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'see
07:21:03 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
07:21:03 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
<prophet.forecaster.Prophet at 0x7f58c86eb6a0>
```

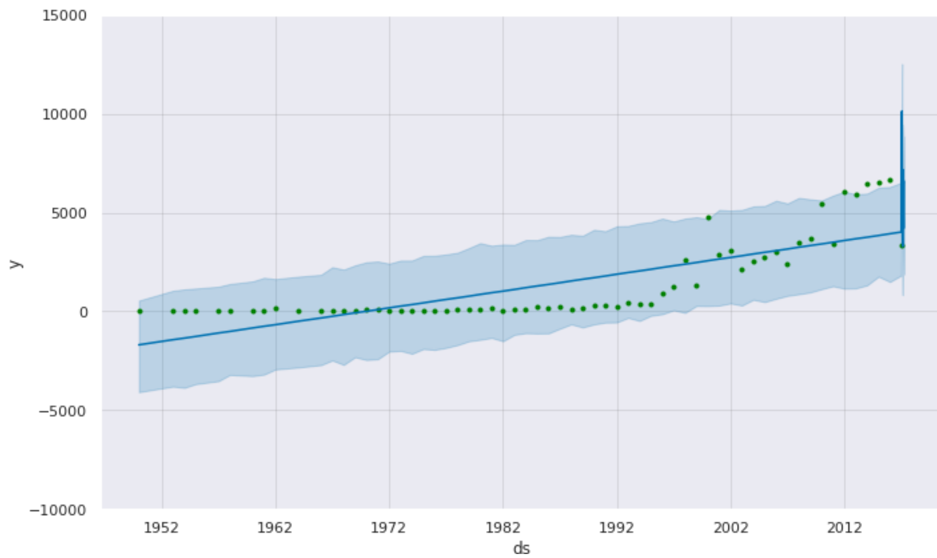
```
test_green_future_dates = green_prophet_model.make_future_dataframe(periods=13, freq='W')
test_green_future_dates.head()
```

	ds
0	1950-01-01
1	1953-01-01
2	1954-01-01
3	1955-01-01
4	1957-01-01

```
forecast_green_prophet = green_prophet_model.predict(test_green_future_dates)
forecast_green_prophet[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

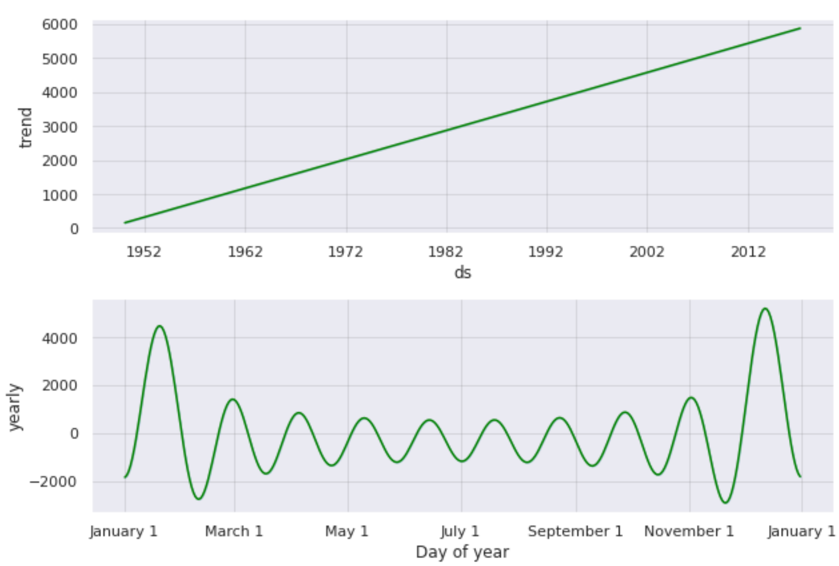
	ds	yhat	yhat_lower	yhat_upper
70	2017-03-05	6713.555469	4506.388518	9029.164986

```
green_prophet_model.plot(forecast_green_prophet, uncertainty=True)
plt.ylim(-10000, 15000)
plt.gca().get_lines()[0].set_color("green")
```



INTERPRET

```
green_prophet_model.plot_components(forecast_green_prophet)
for ax in plt.gcf().axes:
    ax.get_lines()[0].set_color("green")
```



Time Series Model for Quantities of Neutral Colored Legos

SCRUB

```
# Rename columns for prophet
prophet_neutral_df = color_prophet_df.rename(index=str, columns={"Quantity": "y", "Date": "ds"})
prophet_neutral_df = prophet_neutral_df[prophet_neutral_df['ColorGroup']=='Neutrals']
prophet_neutral_df = prophet_neutral_df.drop(columns = ['ColorGroup','Year'])
prophet_neutral_df
```

	y	ds
2	25	1950-01-01
8	23	1953-01-01
13	71	1954-01-01
18	412	1955-01-01
22	153	1956-01-01
...	...	...
379	43347	2013-01-01
387	44450	2014-01-01
395	48833	2015-01-01
403	57396	2016-01-01
411	30766	2017-01-01

MODEL

```
# Set the uncertainty interval to 95% (the Prophet default is 80%)
neutral_prophet_model = Prophet(interval_width=0.95)
neutral_prophet_model.fit(prophet_neutral_df)
```

```
INFO:prophet:Disabling weekly seasonality. Run prophet with weekly_seasonality=True to override this.
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmpy7zff7ui/fwezw465.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpy7zff7ui/5chs1d38.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.8/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'see
07:21:06 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
07:21:06 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
<prophet.forecaster.Prophet at 0x7f58efca7df0>
```

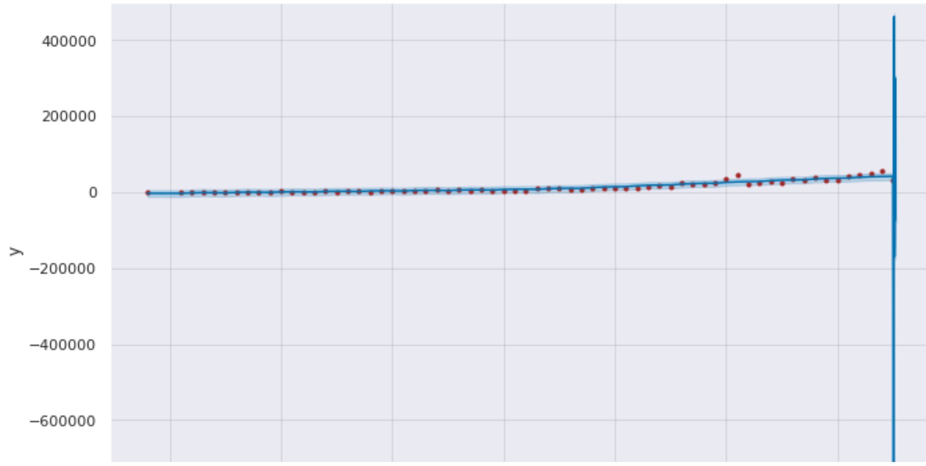
```
test_neutral_future_dates = neutral_prophet_model.make_future_dataframe(periods=13, freq='W')
test_neutral_future_dates.head()
```

	ds
0	1950-01-01
1	1953-01-01
2	1954-01-01
3	1955-01-01
4	1956-01-01

```
forecast_neutral_prophet = neutral_prophet_model.predict(test_neutral_future_dates)
forecast_neutral_prophet[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

	ds	yhat	yhat_lower	yhat_upper
74	2017-03-05	-120904.519169	-130122.988095	-111856.079058
75	2017-03-12	163303.904208	153988.802536	172094.835069
76	2017-03-19	299501.398576	290063.983639	308809.327853
77	2017-03-26	141893.270574	133371.664534	150415.777090
78	2017-04-02	-72860.474167	-81725.209949	-64275.073044

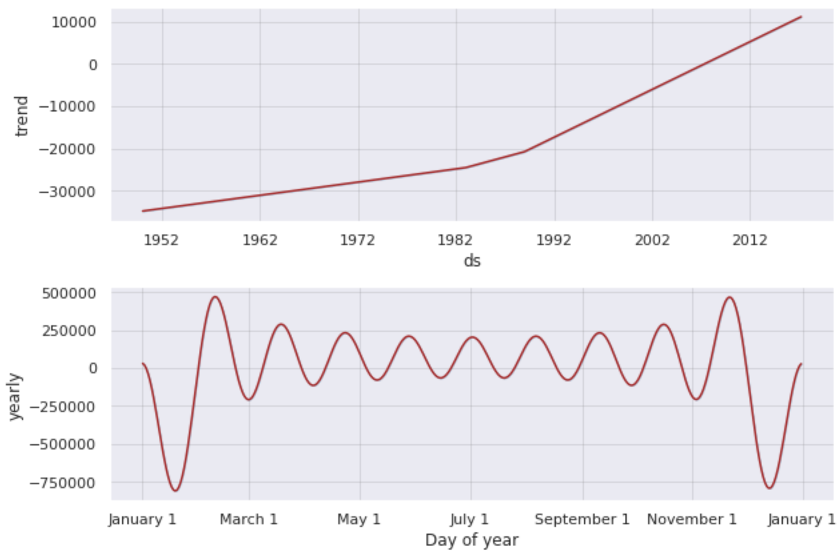
```
neutral_prophet_model.plot(forecast_neutral_prophet, uncertainty=True)
plt.ylim(-800000, 500000)
plt.gca().get_lines()[0].set_color("brown")
```



INTERPRET

ds

```
neutral_prophet_model.plot_components(forecast_neutral_prophet)
for ax in plt.gcf().axes:
    ax.get_lines()[0].set_color("brown")
```



Time Series Model for Quantities of Metallic Legos

SCRUB

```
# Rename columns for prophet
prophet_metallic_df = color_prophet_df.rename(index=str, columns={"Quantity": "y", "Date": "ds"})
prophet_metallic_df = prophet_metallic_df[prophet_metallic_df['ColorGroup']=='Metallics']
prophet_metallic_df = prophet_metallic_df.drop(columns = ['ColorGroup','Year'])
prophet_metallic_df
```

	y	ds
26	2	1957-01-01
32	10	1958-01-01
47	1	1961-01-01
53	1	1962-01-01
91	6	1970-01-01
112	12	1974-01-01
128	2	1977-01-01
195	64	1989-01-01
201	10	1990-01-01
207	12	1991-01-01
213	39	1992-01-01
219	56	1993-01-01
226	59	1994-01-01
234	97	1995-01-01
242	155	1996-01-01
250	201	1997-01-01
258	231	1998-01-01
266	189	1999-01-01
274	233	2000-01-01
282	210	2001-01-01
290	321	2002-01-01
298	346	2003-01-01
306	358	2004-01-01
314	519	2005-01-01
322	811	2006-01-01
330	466	2007-01-01
338	627	2008-01-01
346	629	2009-01-01

MODEL

362 1416 2011-01-01

```
# Set the uncertainty interval to 95% (the Prophet default is 80%)
metallic_prophet_model = Prophet(interval_width=0.95)
metallic_prophet_model.fit(prophet_metallic_df)
```

```
INFO:prophet:Disabling weekly seasonality. Run prophet with weekly_seasonality=True to override this.
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmpy7zff7ui/bk018xtd.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpy7zff7ui/v8ga48pl.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.8/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'see
07:21:09 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
07:21:09 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
<prophet.forecaster.Prophet at 0x7f58ca852820>
```

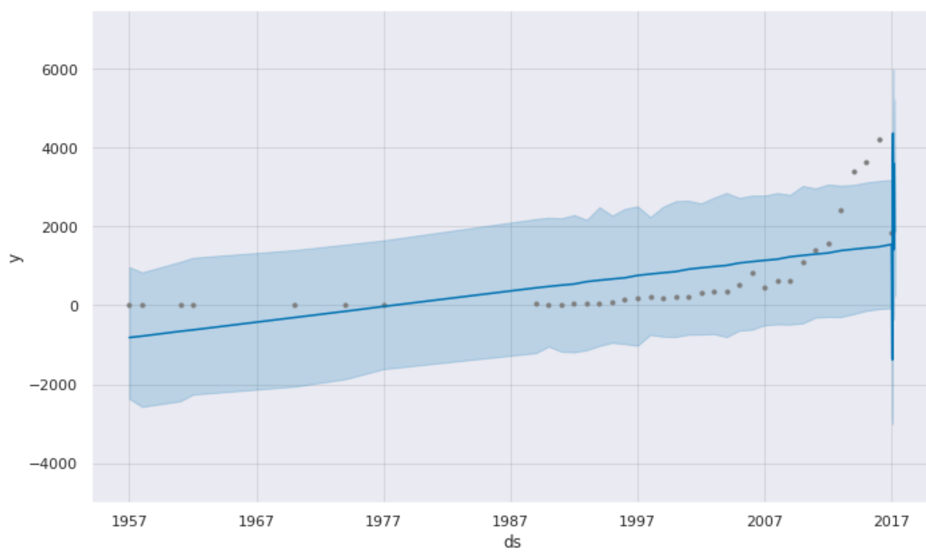
```
test_metallic_future_dates = metallic_prophet_model.make_future_dataframe(periods=13, freq='W')
test_metallic_future_dates.head()
```

	ds
0	1957-01-01
1	1958-01-01

```
forecast_metallic_prophet = metallic_prophet_model.predict(test_metallic_future_dates)
forecast_metallic_prophet[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

	ds	yhat	yhat_lower	yhat_upper
44	2017-03-05	1579.663917	-84.417218	3187.350677
45	2017-03-12	2904.052545	1347.840315	4556.250215
46	2017-03-19	3591.279631	1954.681913	5221.661521
47	2017-03-26	2891.769824	1351.417080	4564.860225
48	2017-04-02	1865.530967	242.823375	3661.594069

```
metallic_prophet_model.plot(forecast_metallic_prophet, uncertainty=True)
plt.ylim(-5000, 7500)
for ax in plt.gcf().axes:
    ax.get_lines()[0].set_color("gray")
```



INTERPRET

```
metallic_prophet_model.plot_components(forecast_metallic_prophet)
for ax in plt.gcf().axes:
    ax.get_lines()[0].set_color("gray")
#    plt.title('Time Series for Metallic Brick Quantities')
```

Time Series Model for Quantities of Orange Legos

SCRUB

```
# Rename columns for prophet
prophet_orange_df = color_prophet_df.rename(index=str, columns={"Quantity": "y", "Date": "ds"})
prophet_orange_df = prophet_orange_df[prophet_orange_df['ColorGroup']=='Oranges']
prophet_orange_df = prophet_orange_df.drop(columns = ['ColorGroup','Year'])
prophet_orange_df
```

	y	ds
3	2	1950-01-01
155	12	1982-01-01
161	2	1983-01-01
167	1	1984-01-01
173	3	1985-01-01
179	1	1986-01-01
185	3	1987-01-01
221	214	1993-01-01
228	25	1994-01-01
236	78	1995-01-01
244	131	1996-01-01
252	156	1997-01-01
260	223	1998-01-01
268	303	1999-01-01
276	443	2000-01-01
284	461	2001-01-01
292	929	2002-01-01
300	1279	2003-01-01
308	1522	2004-01-01
316	1844	2005-01-01
324	1159	2006-01-01
332	928	2007-01-01
340	1623	2008-01-01
348	1811	2009-01-01
356	2460	2010-01-01
364	1574	2011-01-01
372	2812	2012-01-01
380	2954	2013-01-01
388	3980	2014-01-01
396	4328	2015-01-01
404	5122	2016-01-01
412	2226	2017-01-01

MODEL

```
# Set the uncertainty interval to 95% (the Prophet default is 80%)
orange_prophet_model = Prophet(interval_width=0.95)
```

```
orange_prophet_model.fit(prophet_orange_df)
```

```
INFO:prophet:Disabling weekly seasonality. Run prophet with weekly_seasonality=True to override this.
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
INFO:prophet:n_changepoints greater than number of observations. Using 24.
DEBUG:cmdstanpy:input tempfile: /tmp/tmpy7zff7ui/wpwl37n4.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpy7zff7ui/7z1mo03s.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.8/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'see
07:21:12 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
07:21:12 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
<prophet.forecaster.Prophet at 0x7f58ca9b0550>
```

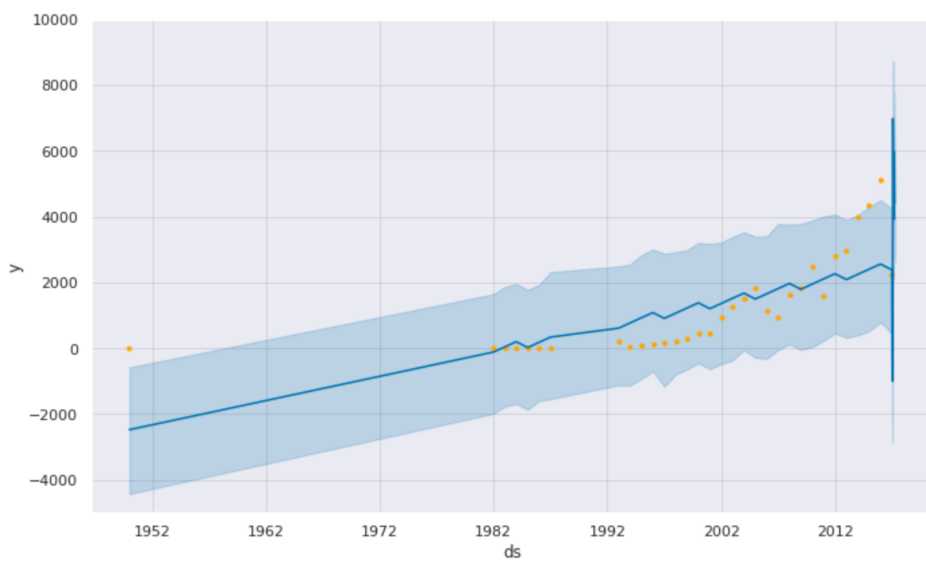
```
test_orange_future_dates = orange_prophet_model.make_future_dataframe(periods=13, freq='W')
test_orange_future_dates.head()
```

	ds
0	1950-01-01
1	1982-01-01
2	1983-01-01
3	1984-01-01
4	1985-01-01

```
forecast_orange_prophet = orange_prophet_model.predict(test_orange_future_dates)
forecast_orange_prophet[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

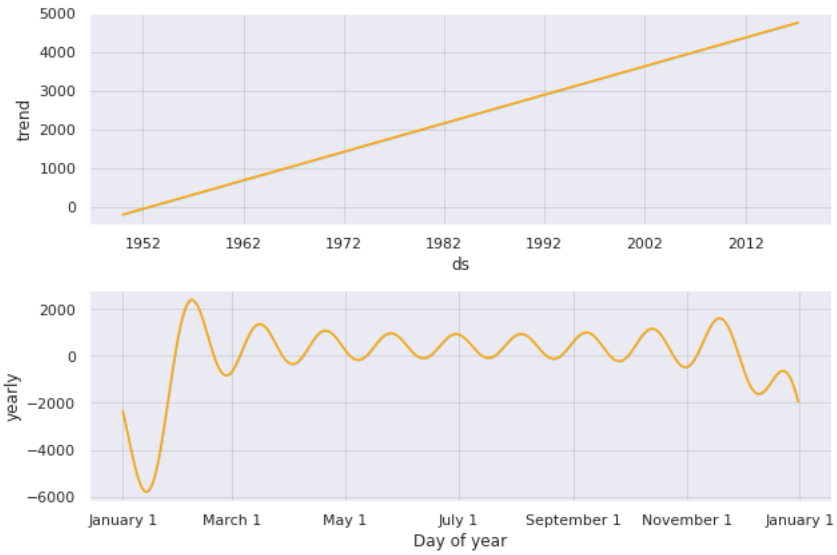
	ds	yhat	yhat_lower	yhat_upper
40	2017-03-05	4715.949213	2791.157145	6621.861064
41	2017-03-12	5891.477624	4039.037583	7794.819562
42	2017-03-19	5964.259447	4144.737616	7639.975681
43	2017-03-26	5033.797590	3300.736429	6837.170178
44	2017-04-02	4408.234206	2635.606081	6128.383640

```
orange_prophet_model.plot(forecast_orange_prophet, uncertainty=True)
plt.ylim(-5000, 10000)
for ax in plt.gcf().axes:
    ax.get_lines()[0].set_color("orange")
```



INTERPRET

```
orange_prophet_model.plot_components(forecast_orange_prophet)
for ax in plt.gcf().axes:
    ax.get_lines()[0].set_color("orange")
#    plt.title('Time Series for Orange Brick Quantities')
```



Time Series Model for Quantities of Red Legos

SCRUB

```
# Rename columns for prophet
prophet_red_df = color_prophet_df.rename(index=str, columns={"Quantity": "y", "Date": "ds"})
prophet_red_df = prophet_red_df[prophet_red_df['ColorGroup']=='Reds']
prophet_red_df = prophet_red_df.drop(columns = ['ColorGroup','Year'])
prophet_red_df
```

	y	ds
4	12	1950-01-01
9	16	1953-01-01
14	69	1954-01-01
19	250	1955-01-01
23	63	1956-01-01
...	...	...
382	15184	2013-01-01
390	16610	2014-01-01
398	15979	2015-01-01
406	17958	2016-01-01
414	8799	2017-01-01

66 rows × 2 columns

MODEL

```
# Set the uncertainty interval to 95% (the Prophet default is 80%)
red_prophet_model = Prophet(interval_width=0.95)
red_prophet_model.fit(prophet_red_df)
```

```
INFO:prophet:Disabling weekly seasonality. Run prophet with weekly_seasonality=True to override this.
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmpy7zff7ui/gdkq9tix.json
```

```
DEBUG:cmdstanpy:input tempfile: /tmp/tmpy7zff7ui/p9046hig.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.8/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'see
07:21:14 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
07:21:15 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
<prophet.forecaster.Prophet at 0x7f58ca959b20>
```

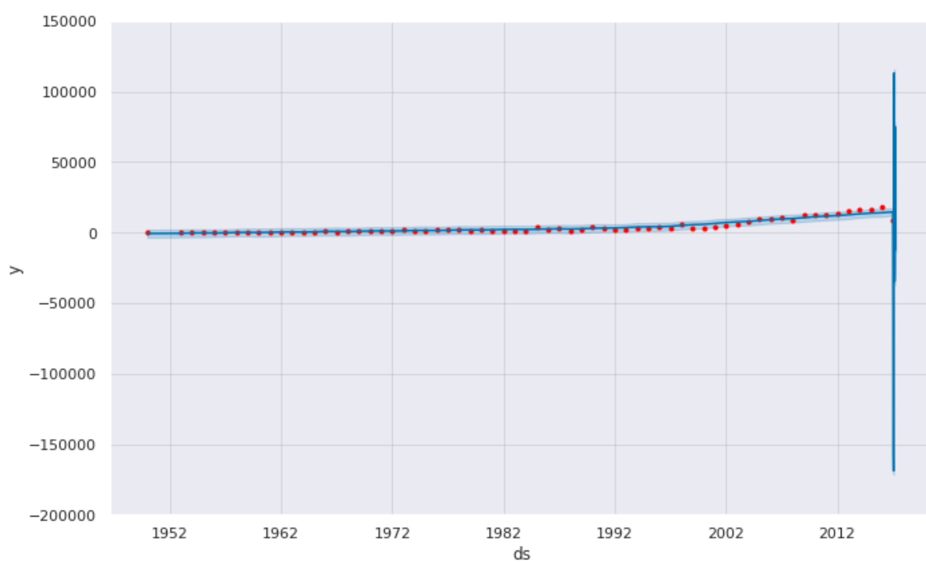
```
test_red_future_dates = red_prophet_model.make_future_dataframe(periods=13, freq='W')
test_red_future_dates.head()
```

	ds
0	1950-01-01
1	1953-01-01
2	1954-01-01
3	1955-01-01
4	1956-01-01

```
forecast_red_prophet = red_prophet_model.predict(test_red_future_dates)
forecast_red_prophet[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

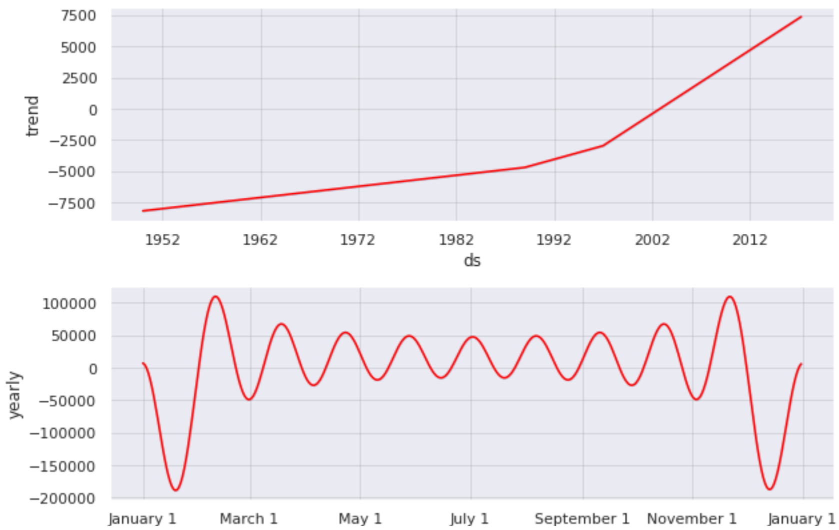
	ds	yhat	yhat_lower	yhat_upper
74	2017-03-05	-24047.478715	-27128.360790	-21230.076352
75	2017-03-12	42365.536009	39714.503828	45053.632080
76	2017-03-19	74937.001106	72219.959614	77709.506117
77	2017-03-26	38568.225689	35667.157800	41263.800635
78	2017-04-02	-11978.683876	-14665.105111	-9027.112506

```
red_prophet_model.plot(forecast_red_prophet, uncertainty=True)
plt.ylim(-200000, 150000)
for ax in plt.gcf().axes:
    ax.get_lines()[0].set_color("red")
```



INTERPRET

```
red_prophet_model.plot_components(forecast_red_prophet)
for ax in plt.gcf().axes:
    ax.get_lines()[0].set_color("red")
#    plt.title('Time Series for Red Brick Quantities')
```



Time Series Model for Quantities of Yellow Legos

SCRUB

```
# Rename columns for prophet
prophet_yellow_df = color_prophet_df.rename(index=str, columns={"Quantity": "y", "Date": "ds"})
prophet_yellow_df = prophet_yellow_df[prophet_yellow_df['ColorGroup']=='Yellows']
prophet_yellow_df = prophet_yellow_df.drop(columns = ['ColorGroup','Year'])
prophet_yellow_df
```

	y	ds
5	12	1950-01-01
10	13	1953-01-01
15	16	1954-01-01
20	185	1955-01-01
29	20	1957-01-01
...	...	...
383	5025	2013-01-01
391	5081	2014-01-01
399	5410	2015-01-01
407	6565	2016-01-01
415	2478	2017-01-01

64 rows × 2 columns

MODEL

```
# Set the uncertainty interval to 95% (the Prophet default is 80%)
yellow_prophet_model = Prophet(interval_width=0.95)
yellow_prophet_model.fit(prophet_yellow_df)
```

```
INFO:prophet:Disabling weekly seasonality. Run prophet with weekly_seasonality=True to override this.
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmpy7zff7ui/npfxb9m7.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpy7zff7ui/p51cko60.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.8/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'see
07:21:18 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
07:21:18 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
<prophet.forecaster.Prophet at 0x7f58cac09640>
```

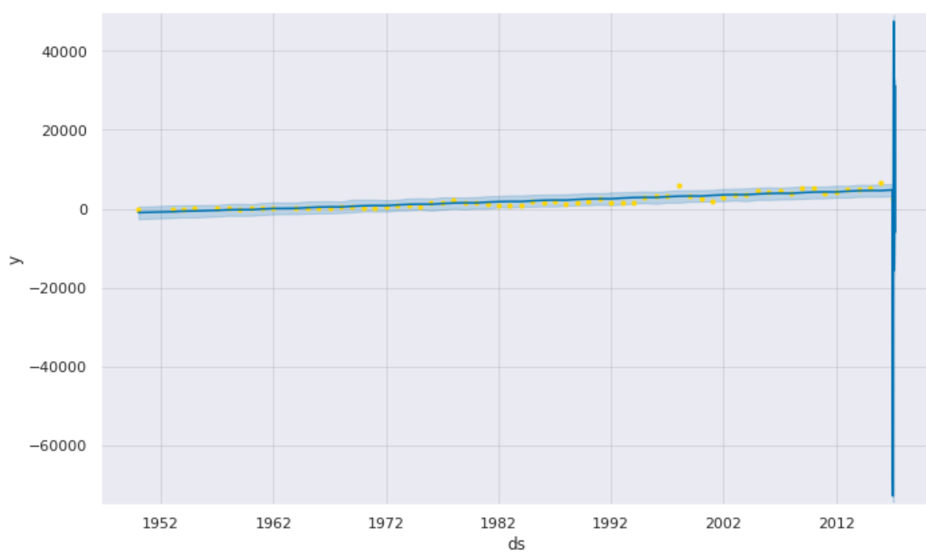
```
test_yellow_future_dates = yellow_prophet_model.make_future_dataframe(periods=13, freq='W')
test_yellow_future_dates.head()
```

	ds
0	1950-01-01
1	1953-01-01
2	1954-01-01
3	1955-01-01
4	1957-01-01

```
forecast_yellow_prophet = yellow_prophet_model.predict(test_yellow_future_dates)
forecast_yellow_prophet[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

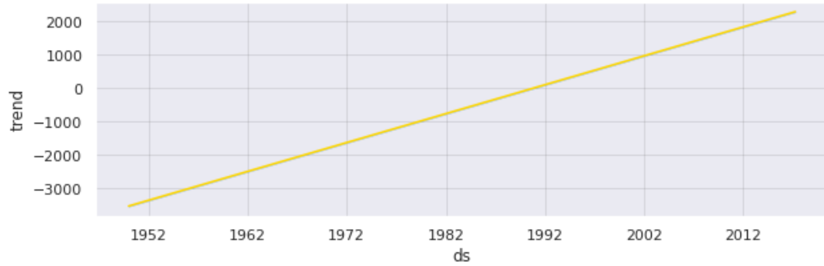
	ds	yhat	yhat_lower	yhat_upper
72	2017-03-05	-11076.908625	-12741.055530	-9477.511953
73	2017-03-12	17339.013930	15830.548368	18788.650952
74	2017-03-19	31253.840476	29774.246390	32870.919426
75	2017-03-26	15670.129308	14065.349316	17196.615734
76	2017-04-02	-5961.786162	-7622.735317	-4431.632181

```
yellow_prophet_model.plot(forecast_yellow_prophet, uncertainty=True)
plt.ylim(-75000, 50000)
for ax in plt.gcf().axes:
    ax.get_lines()[0].set_color("gold")
```



INTERPRET

```
yellow_prophet_model.plot_components(forecast_yellow_prophet)
for ax in plt.gcf().axes:
    ax.get_lines()[0].set_color("gold")
#    plt.title('Time Series for Yellow Brick Quantities')
```



Time Series Model for Quantities of Purple Legos

SCRUB

```
# Rename columns for prophet
prophet_purple_df = color_prophet_df.rename(index=str, columns={"Quantity": "y", "Date": "ds"})
prophet_purple_df = prophet_purple_df[prophet_purple_df['ColorGroup']=='Purples']
prophet_purple_df = prophet_purple_df.drop(columns = ['ColorGroup','Year'])
prophet_purple_df
```

	y	ds
229	30	1994-01-01
237	14	1995-01-01
245	20	1996-01-01
253	8	1997-01-01
261	302	1998-01-01
269	193	1999-01-01
277	53	2000-01-01
285	124	2001-01-01
293	231	2002-01-01
301	156	2003-01-01
309	367	2004-01-01
317	274	2005-01-01
325	50	2006-01-01
333	15	2007-01-01
341	43	2008-01-01
349	25	2009-01-01
357	184	2010-01-01
365	488	2011-01-01
373	565	2012-01-01
381	539	2013-01-01
389	917	2014-01-01
397	2100	2015-01-01
405	1676	2016-01-01
413	1127	2017-01-01

MODEL

```
# Set the uncertainty interval to 95% (the Prophet default is 80%)
purple_prophet_model = Prophet(interval_width=0.95)
purple_prophet_model.fit(prophet_purple_df)
```

```
INFO:prophet:Disabling weekly seasonality. Run prophet with weekly_seasonality=True to override this.
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
INFO:prophet:n_changepoints greater than number of observations. Using 18.
DEBUG:cmdstanpy:input tempfile: /tmp/tmpy7zff7ui/4wqamjbr.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpy7zff7ui/j5nkuzcq.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.8/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'see
07:21:21 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
07:21:21 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
<prophet.forecaster.Prophet at 0x7f58ca66da60>
```

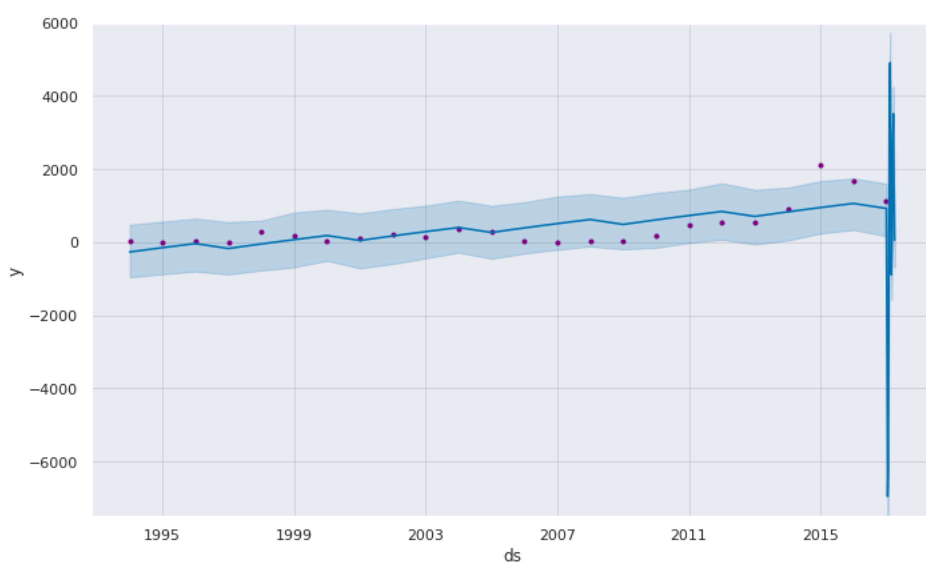
```
test_purple_future_dates = purple_prophet_model.make_future_dataframe(periods=13, freq='W')
test_purple_future_dates.head()
```

	ds
0	1994-01-01
1	1995-01-01
2	1996-01-01
3	1997-01-01
4	1998-01-01

```
forecast_purple_prophet = purple_prophet_model.predict(test_purple_future_dates)
forecast_purple_prophet[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

	ds	yhat	yhat_lower	yhat_upper
32	2017-03-05	38.876717	-650.069998	744.193767
33	2017-03-12	2657.104095	1920.436965	3411.354630
34	2017-03-19	3514.936159	2794.245730	4260.920544
35	2017-03-26	1828.101999	1092.490945	2594.372716
36	2017-04-02	57.101864	-686.661987	789.956064

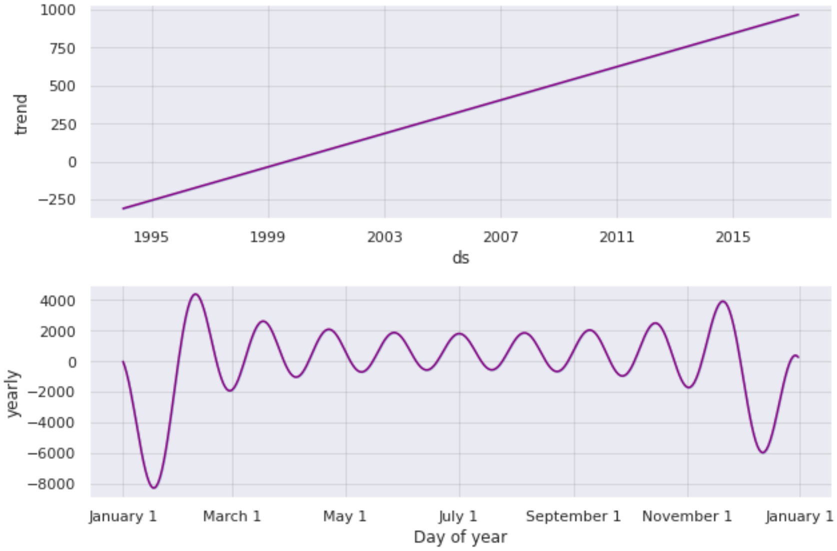
```
purple_prophet_model.plot(forecast_purple_prophet, uncertainty=True)
plt.ylim(-7500, 6000)
for ax in plt.gcf().axes:
    ax.get_lines()[0].set_color("purple")
```



INTERPRET

```
purple_prophet_model.plot_components(forecast_purple_prophet)
for ax in plt.gcf().axes:
```

```
    ax.get_lines()[0].set_color("purple")
#    plt.title('Time Series for Purple Brick Quantities')
```



SVM Attempt

```
from sklearn import svm, datasets
import sklearn.model_selection as model_selection
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
```

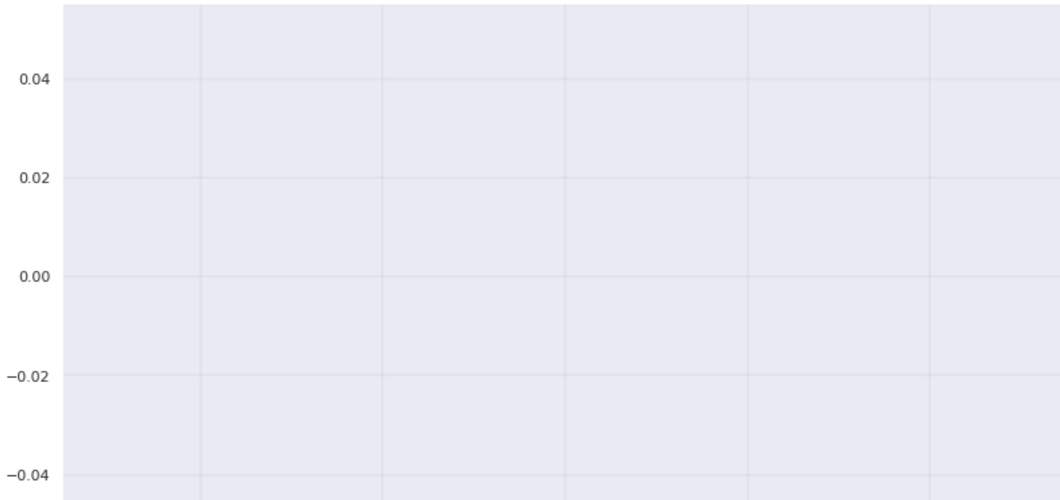
```
color_qty_year_df
```

	Year	Color	Quantity	Decade	Era	ColorGroup
0	1950	Blue	6	1950s	Early Years (1950-1979)	Blues
1	1950	Bright Green	4	1950s	Early Years (1950-1979)	Greens
2	1950	Green	6	1950s	Early Years (1950-1979)	Greens
3	1950	Light Green	2	1950s	Early Years (1950-1979)	Greens
4	1950	Medium Blue	2	1950s	Early Years (1950-1979)	Blues
...	...	...	...	...	...	...
2078	2017	Unknown	41	2010s	Present Years (2000-Present)	Neutrals
2079	2017	White	8830	2010s	Present Years (2000-Present)	Neutrals
2080	2017	Yellow	1956	2010s	Present Years (2000-Present)	Yellows
2081	2017	Yellowish Green	131	2010s	Present Years (2000-Present)	Greens
2082	2017	[No Color]	30	2010s	Present Years (2000-Present)	Neutrals

2083 rows × 6 columns

```
# Adapted from: https://towardsdatascience.com/support-vector-machines-explained
# Hold out 20% of the dataset for training
size = color_qty_year_df.size
test_size = int(np.round(size * 0.2, 0))
features = color_qty_year_df.filter(['Year','Quantity'],axis=1)
label = color_qty_year_df.filter(['Color'],axis=1)

# Split dataset into training and testing sets
x_train = features[:-test_size].values
y_train = label[:-test_size].values
x_test = features[-test_size:].values
y_test = label[-test_size:].values

# Plotting the training set
fig, ax = plt.subplots(figsize=(12, 7))

# removing to and right border
```

```
ax.spines['top'].set_visible(False)
ax.spines['left'].set_visible(False)
ax.spines['right'].set_visible(False)

# adding major gridlines
ax.grid(color='grey', linestyle='-', linewidth=0.25, alpha=0.5)
ax.scatter(features[:-test_size]['Year'], features[:-test_size]['Quantity'], col
plt.show()
```




```
# Adapted from: https://www.baeldung.com/cs/svm-multiclass-classification
X = color_qty_year_df.filter(['Year','Quantity'],axis=1)
y = color_qty_year_df.filter(['Color'],axis=1)
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, train_size=0.80, test_size=0.20, random_state=101)

rbf = svm.SVC(kernel='rbf', gamma=0.5, C=0.1).fit(X_train, y_train)
poly = svm.SVC(kernel='poly', degree=3, C=1).fit(X_train, y_train)

poly_pred = poly.predict(X_test)
rbf_pred = rbf.predict(X_test)
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/utils/validation.py:993: DataConversionWarning: A column-vector y was passed
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.8/dist-packages/sklearn/utils/validation.py:993: DataConversionWarning: A column-vector y was passed
  y = column_or_1d(y, warn=True)
```

```
poly_accuracy = accuracy_score(y_test, poly_pred)
poly_f1 = f1_score(y_test, poly_pred, average='weighted')
print('Accuracy (Polynomial Kernel): ', "%.2f" % (poly_accuracy*100))
print('F1 (Polynomial Kernel): ', "%.2f" % (poly_f1*100))
```

```
Accuracy (Polynomial Kernel):  6.95
F1 (Polynomial Kernel):  3.43
```

```
rbf_accuracy = accuracy_score(y_test, rbf_pred)
rbf_f1 = f1_score(y_test, rbf_pred, average='weighted')
print('Accuracy (RBF Kernel): ', "%.2f" % (rbf_accuracy*100))
print('F1 (RBF Kernel): ', "%.2f" % (rbf_f1*100))
```

```
Accuracy (RBF Kernel):  2.16
F1 (RBF Kernel):  0.09
```

Incomplete or inconclusive code to revisit at a future time

SCRUB [FIRST ROUND]

[ ] ↳ *15 cells hidden*

EXPLORE [FIRST ROUND]

↳ 42 cells hidden