```
from prophet import Prophet
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from statsmodels.tsa.seasonal import seasonal_decompose
import random

from sklearn.metrics import mean_absolute_percentage_error
```

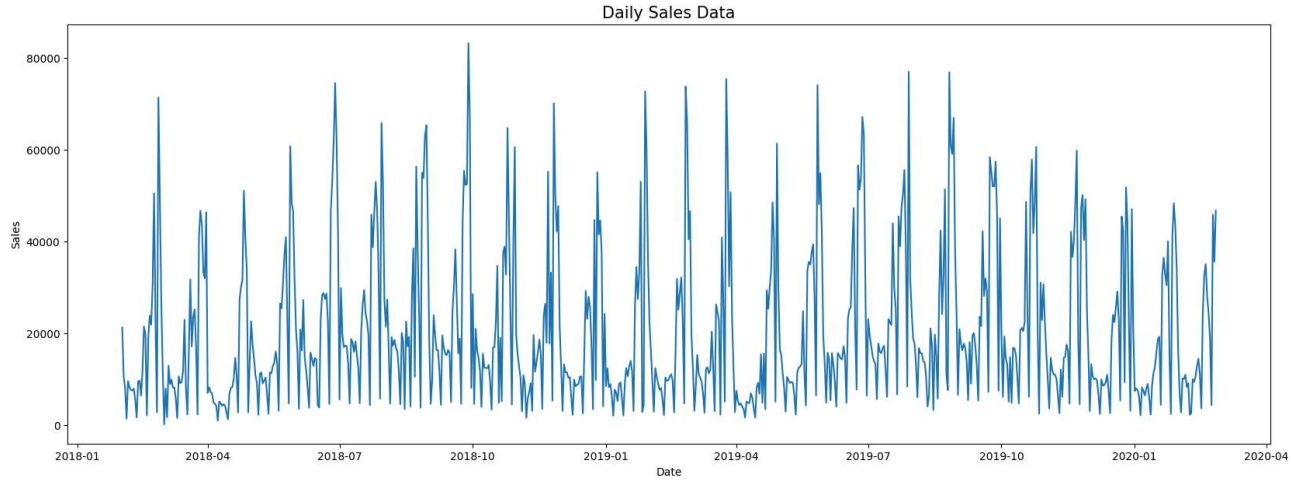
```
df = pd.read_excel(f"Groceries_Sales_data.xlsx",index_col=0)
```

```
df.head()
```

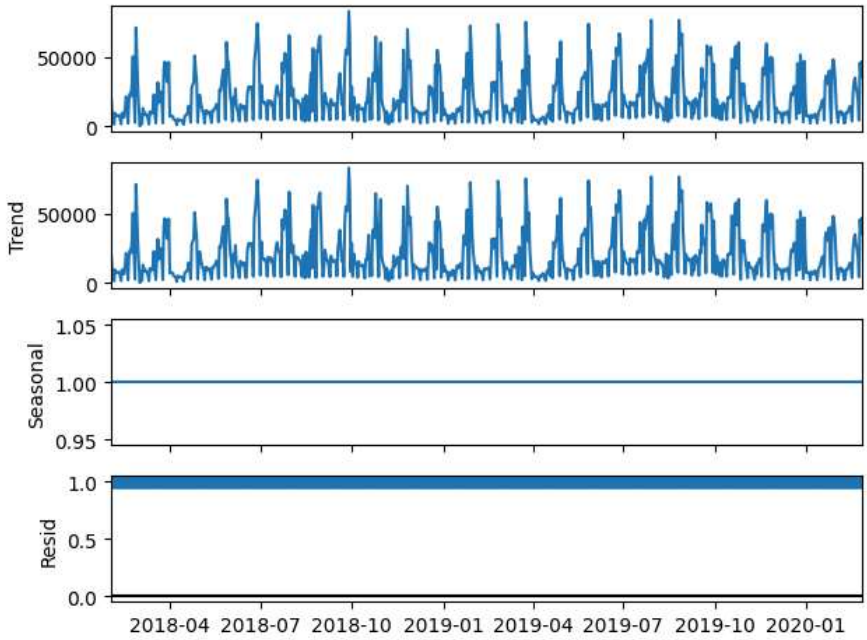
Date	Sales
2018-02-01	21199.0
2018-02-02	10634.0
2018-02-03	7966.0
2018-02-04	1353.0
2018-02-05	9497.0

Next steps: Generate code with df | View recommended plots | New interactive sheet

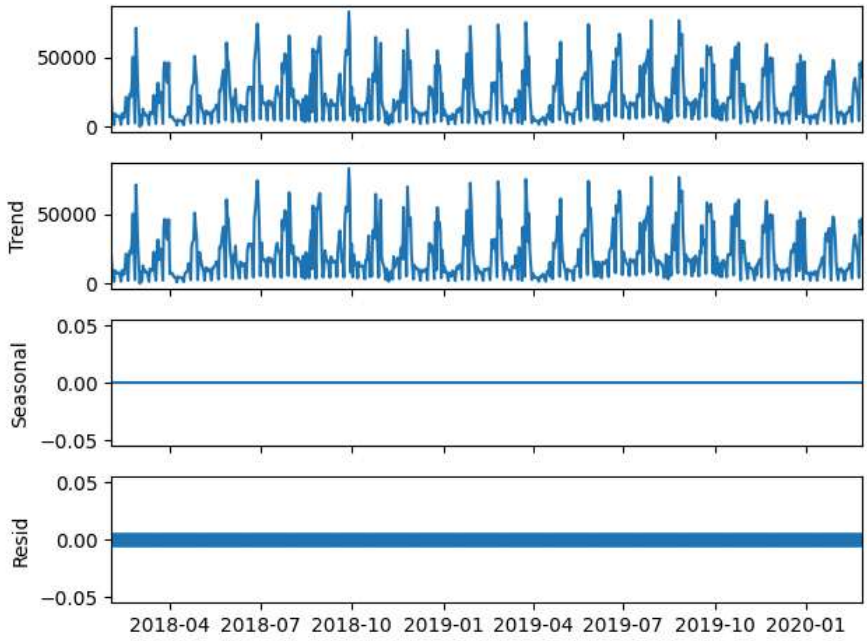
```
fig, ax = plt.subplots(figsize=(20,7))
a = sns.lineplot(x="Date", y="Sales", data=df)
a.set_title("Daily Sales Data",fontsize=15)
plt.show()
```




```
result_2 = seasonal_decompose(df, model='multiplicative', period=1)
result_2.plot()
plt.show()
```




```
result_3 = seasonal_decompose(df, model='additive', period=1)
result_3.plot()
plt.show()
```




```
Start coding or generate with AI.
```

```
df.index = pd.to_datetime(df.index)
```

```
df
```

Date	Sales
2018-02-01	21199.0
2018-02-02	10634.0
2018-02-03	7966.0
2018-02-04	1353.0
2018-02-05	9497.0
...	...
2020-02-22	18723.1
2020-02-23	4274.9
2020-02-24	45805.7
2020-02-25	35566.3
2020-02-26	46703.0

756 rows × 1 columns

Next steps: Generate code with df View recommended plots New interactive sheet

```
y = df["Sales"]
y.name = "Sales"
```

```
seasonal_df = y.to_frame()
seasonal_df
```

Date	Sales
2018-02-01	21199.0
2018-02-02	10634.0
2018-02-03	7966.0
2018-02-04	1353.0
2018-02-05	9497.0
...	...
2020-02-22	18723.1
2020-02-23	4274.9
2020-02-24	45805.7
2020-02-25	35566.3
2020-02-26	46703.0

756 rows × 1 columns

Next steps: Generate code with seasonal_df View recommended plots New interactive sheet

```
seasonal_df["trend"] = seasonal_df["Sales"].rolling(window=7, center=True).mean()
seasonal_df.head(10)
```

Date	Sales	trend
2018-02-01	21199.0	NaN
2018-02-02	10634.0	NaN
2018-02-03	7966.0	NaN
2018-02-04	1353.0	9491.000000
2018-02-05	9497.0	7529.857143
2018-02-06	8207.0	7136.142857
2018-02-07	7581.0	6757.285714
2018-02-08	7471.0	6793.285714
2018-02-09	7878.0	6782.142857
2018-02-10	5314.0	6982.571429

Next steps: Generate code with seasonal_df | View recommended plots | New interactive sheet

```
seasonal_df["detrended"] = seasonal_df["Sales"] - seasonal_df["trend"]
seasonal_df.head(10)
```

Date	Sales	trend	detrended
2018-02-01	21199.0	NaN	NaN
2018-02-02	10634.0	NaN	NaN
2018-02-03	7966.0	NaN	NaN
2018-02-04	1353.0	9491.000000	-8138.000000
2018-02-05	9497.0	7529.857143	1967.142857
2018-02-06	8207.0	7136.142857	1070.857143
2018-02-07	7581.0	6757.285714	823.714286
2018-02-08	7471.0	6793.285714	677.714286
2018-02-09	7878.0	6782.142857	1095.857143
2018-02-10	5314.0	6982.571429	-1668.571429

Next steps: Generate code with seasonal_df | View recommended plots | New interactive sheet

```
seasonal_df.index = pd.to_datetime(seasonal_df.index)
seasonal_df["month"] = seasonal_df.index.month
seasonal_df["seasonality"] = seasonal_df.groupby("month")["detrended"].transform("mean")
seasonal_df.head(15)
```

Date	Sales	trend	detrended	month	seasonality
2018-02-01	21199.0	NaN	NaN	2	-112.66203
2018-02-02	10634.0	NaN	NaN	2	-112.66203
2018-02-03	7966.0	NaN	NaN	2	-112.66203
2018-02-04	1353.0	9491.000000	-8138.000000	2	-112.66203

```
seasonal_df = y.to_frame()

# calculate the trend component
seasonal_df["trend"] = seasonal_df["Sales"].rolling(window=13, center=True).mean()

# detrend the series
seasonal_df["detrended"] = seasonal_df["Sales"] - seasonal_df["trend"]

# calculate the seasonal component
seasonal_df.index = pd.to_datetime(seasonal_df.index)
seasonal_df["month"] = seasonal_df.index.month
seasonal_df["seasonality"] = seasonal_df.groupby("month")["detrended"].transform("mean")

# get the residuals
seasonal_df["resid"] = seasonal_df["detrended"] - seasonal_df["seasonality"]

# display the DF
seasonal_df.head(15)
```

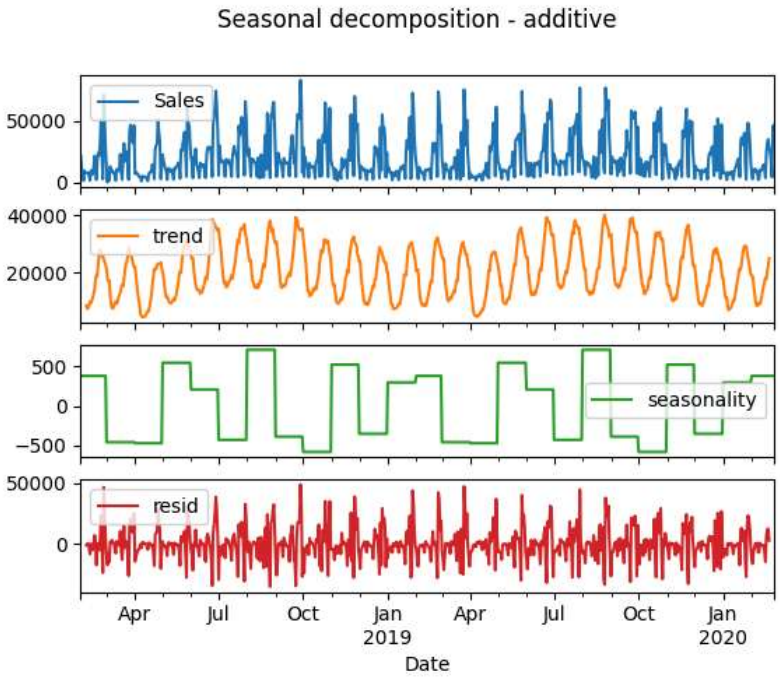
Next steps: Generate code with seasonal_df View recommended plots New interactive sheet

Date	Sales	trend	detrended	month	seasonality	resid
2018-02-01	21199.0	NaN	NaN	2	381.152198	NaN
2018-02-02	10634.0	NaN	NaN	2	381.152198	NaN
2018-02-03	7966.0	NaN	NaN	2	381.152198	NaN
2018-02-04	1353.0	NaN	NaN	2	381.152198	NaN
2018-02-05	9497.0	NaN	NaN	2	381.152198	NaN
2018-02-06	8207.0	NaN	NaN	2	381.152198	NaN
2018-02-07	7581.0	8287.230769	-706.230769	2	381.152198	-1087.382967
2018-02-08	7471.0	7147.930769	323.069231	2	381.152198	-58.082967
2018-02-09	7878.0	7237.546154	640.453846	2	381.152198	259.301648
2018-02-10	5314.0	8273.784615	-2959.784615	2	381.152198	-3340.936813
2018-02-11	1605.0	9693.061538	-8088.061538	2	381.152198	-8469.213736
2018-02-12	9419.0	9123.369231	295.630769	2	381.152198	-85.521429
2018-02-13	9610.0	9949.884615	-339.884615	2	381.152198	-721.036813
2018-02-14	6388.1	11195.930769	-4807.830769	2	381.152198	-5188.982967
2018-02-15	11799.0	12301.600000	-502.600000	2	381.152198	-883.752198

Next steps: Generate code with seasonal_df View recommended plots New interactive sheet

```
seasonal_df.loc[:, ["Sales", "trend", "seasonality", "resid"]].plot(subplots=True, title="Seasonal decomposition - addi
```

```
array([<Axes: xlabel='Date'>, <Axes: xlabel='Date'>,
       <Axes: xlabel='Date'>, <Axes: xlabel='Date'>], dtype=object)
```




```
df=df.reset_index(['Date'])
```

```
df
```

	Date	Sales
0	2018-02-01	21199.0
1	2018-02-02	10634.0
2	2018-02-03	7966.0
3	2018-02-04	1353.0
4	2018-02-05	9497.0
...	...	...
751	2020-02-22	18723.1
752	2020-02-23	4274.9
753	2020-02-24	45805.7
754	2020-02-25	35566.3
755	2020-02-26	46703.0

756 rows × 2 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# Extract features
df_copy = df.copy()

df_copy['date'] = df['Date']
df_copy['month'] = df_copy['date'].dt.strftime('%B')
df_copy['year'] = df_copy['date'].dt.strftime('%Y')
df_copy['dayofweek'] = df_copy['date'].dt.strftime('%A')
df_copy['quarter'] = df_copy['date'].dt.quarter
df_copy['dayofyear'] = df_copy['date'].dt.dayofyear
df_copy['dayofmonth'] = df_copy['date'].dt.day
df_copy['weekofyear'] = df_copy['date'].dt.isocalendar().week
```

```
df_copy.columns
```

```
Index(['Date', 'Sales', 'date', 'month', 'year', 'dayofweek', 'quarter',
       'dayofyear', 'dayofmonth', 'weekofyear'],
      dtype='object')
```

```
X = df_copy[['dayofweek','quarter','month','year',
           'dayofyear','dayofmonth','weekofyear']]
y = df['Sales']
```

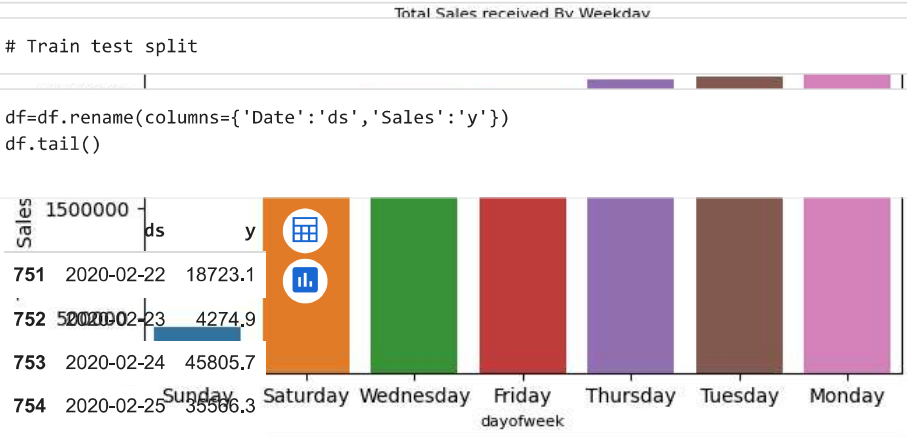
```
df_new = pd.concat([X, y], axis=1)
df_new.head()
```

	dayofweek	quarter	month	year	dayofyear	dayofmonth	weekofyear	Sales
0	Thursday	1	February	2018	32	1	5	21199.0
1	Friday	1	February	2018	33	2	5	10634.0
2	Saturday	1	February	2018	34	3	5	7966.0
3	Sunday	1	February	2018	35	4	5	1353.0
4	Monday	1	February	2018	36	5	6	9497.0

Next steps: Generate code with df_new | View recommended plots | New interactive sheet

```
# fig,(ax1,ax2,ax3,ax4)= plt.subplots(nrows=4)
fig,(ax1,ax2)= plt.subplots(nrows=2)
fig.set_size_inches(7,7)

week_day_Aggregated = pd.DataFrame(df_new.groupby("dayofweek")["Sales"].sum()).reset_index().sort_values('Sales')
sns.barplot(data=week_day_Aggregated,x="dayofweek",y="Sales",hue = 'dayofweek',ax=ax1,dodge=False)
ax1.set(xlabel='dayofweek', ylabel='Total Sales received')
ax1.xaxis.label.set_size(8)
ax1.set_title("Total Sales received By Weekday",fontsize=8)
ax1.ticklabel_format(style='plain',axis='y')
ax1.legend_.remove()


yearAggregated = pd.DataFrame(df_new.groupby("year")["Sales"].sum()).reset_index()
sns.barplot(data=yearAggregated,x="year",y="Sales",hue='year',ax=ax2)
ax2.set(xlabel='year', ylabel='Total Sales received')
ax2.xaxis.label.set_size(8)
ax2.set_title("Total Sales received By year",fontsize=8)
ax2.ticklabel_format(style='plain',axis='y')

fig.tight_layout()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/tmp/ipython-input-2172668525.py in <cell line: 0>()
      9 ax1.set_title("Total Sales received By Weekday",fontsize=8)
     10 ax1.ticklabel_format(style='plain',axis='y')
---> 11 ax1.legend_.remove()
     12
     13

AttributeError: 'NoneType' object has no attribute 'remove'
```



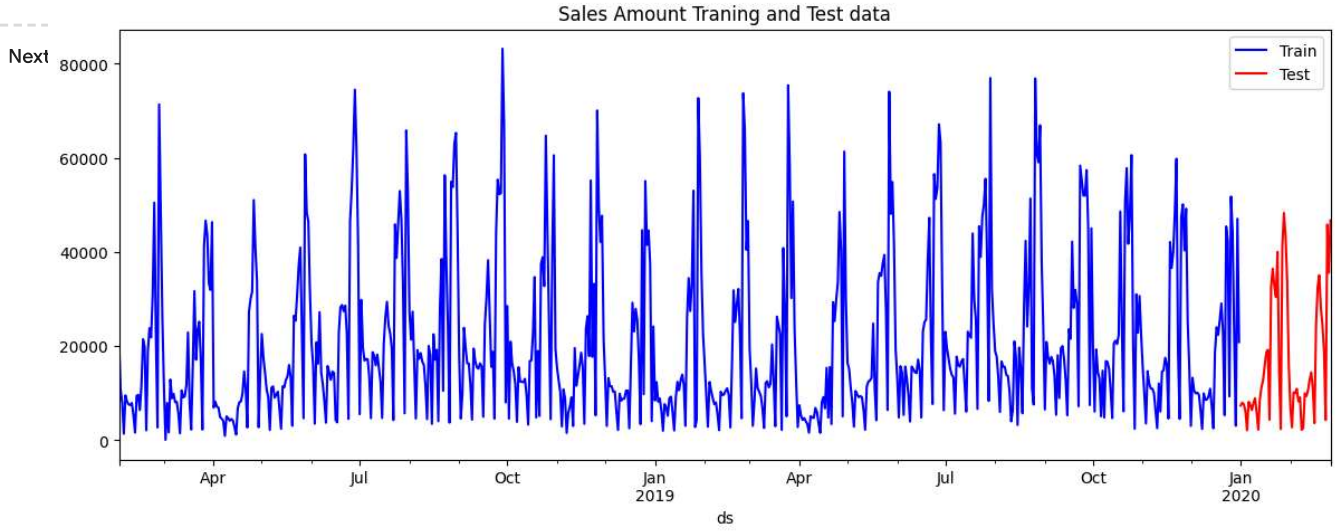

```
# Train test split
```

```
df=df.rename(columns={'Date':'ds','Sales':'y'})
df.tail()
```

	ds	y
751	2020-02-22	18723.1
752	2020-02-23	4274.9
753	2020-02-24	45805.7
754	2020-02-25	35586.3
755	2020-02-26	46703.0

```
end_date = '2019-12-31'
X_tr = df.loc[df['ds'] <= end_date]
X_tst = df.loc[df['ds'] > end_date]
```

```
pd.plotting.register_matplotlib_converters()
f, ax = plt.subplots(figsize=(14,5))
X_tr.plot(kind='line', x='ds', y='y', color='blue', label='Train', ax=ax)
X_tst.plot(kind='line', x='ds', y='y', color='red', label='Test', ax=ax)
plt.title('Sales Amount Traning and Test data')
plt.show()
```




```
# Modelling
```

```
model =Prophet()
model.fit(X_tr)
```

```
INFO:prophet:Disabling yearly seasonality. Run prophet with yearly_seasonality=True to override this.
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmp0z1ib4sr/qrokk3g9.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp0z1ib4sr/lvvd6e3j.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.12/dist-packages/prophet/stan_model/prophet_model.bin', 'random',
11:01:15 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
```

```
11:01:15 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
<prophet.forecaster.Prophet at 0x7b485a96dac0>
```

```
len(X_tst)
```

```
57
```

```
X_tst
```

Next steps: Generate code with X_tst View recommended plots New interactive sheet

	ds	y
699	2020-01-01	7353.6
700	2020-01-02	7959.7
701	2020-01-03	7559.3
702	2020-01-04	6162.3
703	2020-01-05	2085.9
704	2020-01-06	8120.0
705	2020-01-07	7464.4
706	2020-01-08	6385.6
707	2020-01-09	7873.0

```
future = model.make_future_dataframe(periods=57, freq='D')
forecast = model.predict(future)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].head()
```

	ds	yhat	yhat_lower	yhat_upper
0	2018-02-01	22918.473941	4309.902531	41382.122311
1	2018-02-02	21669.554668	2484.440740	39732.517145
2	2018-02-03	14076.387315	-5366.567568	31683.608667
3	2018-02-04	16690.017765	-17168.907855	20456.339101
4	2018-02-05	25535.083176	6342.620793	44469.766034

```
future_2 = model.make_future_dataframe(periods=60)
forecast_2 = model.predict(future_2)
forecast_2[['ds', 'yhat']].tail(5)
```

	ds	yhat
754	2020-02-25	27945.349608
755	2020-02-26	25546.363061
756	2020-02-27	28114.586689
757	2020-02-28	26805.667405
758	2020-02-29	19272.500042

```
X_tst_forecast = model.predict(X_tst)
X_tst_forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail(7)
```

	ds	yhat	yhat_lower	yhat_upper
50	2020-02-20	28066.474534	8629.602838	47123.794519
51	2020-02-21	26757.555250	7722.491668	45729.362031
52	2020-02-22	19224.387887	-292.238138	38104.672555
53	2020-02-23	6817.012325	-12434.189570	27572.742364
54	2020-02-24	30683.083725	11796.467416	49312.197799
55	2020-02-25	27945.349608	8769.784260	48096.094269
56	2020-02-26	25546.363061	8405.718771	44849.754932

```
X_tst_forecast
```

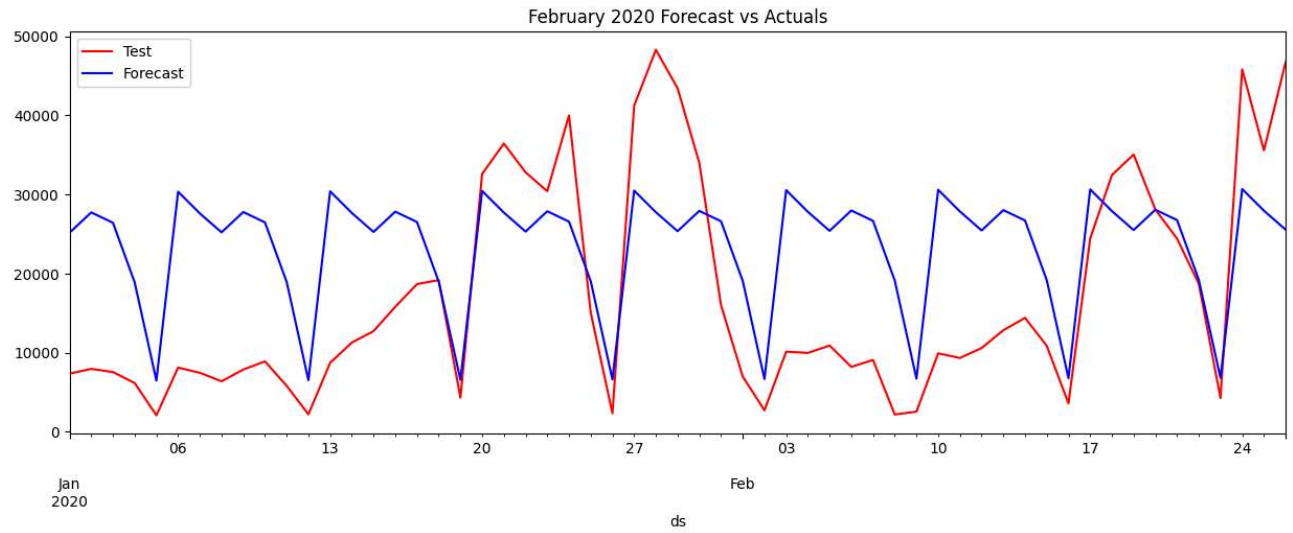
739	2020-02-10	9920.1
740	2020-02-11	9332.1
741	2020-02-12	10593.3
742	2020-02-13	12838.9
743	2020-02-14	14402.9
744	2020-02-15	10866.3
745	2020-02-16	3598.1

746	2020-02-17	24425.7
747	2020-02-18	32450.5
748	2020-02-19	35041.9
749	2020-02-20	28088.8
750	2020-02-21	24412.5
751	2020-02-22	18723.1
752	2020-02-23	4274.9
753	2020-02-24	45805.7
754	2020-02-25	35566.3
755	2020-02-26	46703.0

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_terms	additive_terms_lower	ad
0	2020-01-01	23212.897455	6102.533931	44322.953076	23212.897455	23212.897455	1948.568368	1948.568368	
1	2020-01-02	23219.770620	7036.115730	47539.945742	23219.770620	23219.770620	4509.918830	4509.918830	
2	2020-01-03	23226.643785	6584.008607	46084.543046	23226.643785	23226.643785	3194.126381	3194.126381	
3	2020-01-04	23233.516950	-1098.809896	38431.289825	23233.516950	23233.516950	-4345.914147	-4345.914147	
4	2020-01-05	23240.390115	-12187.457020	26151.027146	23240.390115	23240.390115	-16760.162873	-16760.162873	
5	2020-01-06	23247.263280	10076.575279	50069.484143	23247.263280	23247.263280	7099.035362	7099.035362	
6	2020-01-07	23254.136445	9588.131422	46839.763855	23254.136444	23254.136446	4354.428080	4354.428080	
7	2020-01-08	23261.009610	6584.920713	45081.213929	23261.009608	23261.009612	1948.568368	1948.568368	
8	2020-01-09	23267.882775	9112.225697	46488.264092	23267.882772	23267.882779	4509.918830	4509.918830	
9	2020-01-10	23274.755940	7273.982742	44451.122451	23274.755936	23274.755945	3194.126381	3194.126381	
10	2020-01-11	23281.629105	-173.099260	38916.091749	23281.629099	23281.629112	-4345.914147	-4345.914147	
11	2020-01-12	23288.502270	-12334.657257	25392.683041	23288.502262	23288.502278	-16760.162873	-16760.162873	
12	2020-01-13	23295.375435	10509.605542	48496.294042	23295.375425	23295.375445	7099.035362	7099.035362	
13	2020-01-14	23302.248600	8071.113464	47766.648732	23302.248588	23302.248613	4354.428079	4354.428079	
14	2020-01-15	23309.121765	6769.554647	44127.005801	23309.121751	23309.121780	1948.568368	1948.568368	
15	2020-01-16	23315.994930	9239.068835	46979.418659	23315.994914	23315.994948	4509.918830	4509.918830	
16	2020-01-17	23322.868095	7999.357471	46787.002188	23322.868077	23322.868115	3194.126381	3194.126381	
17	2020-01-18	23329.741260	-1649.610006	36647.016451	23329.741239	23329.741282	-4345.914147	-4345.914147	
18	2020-01-19	23336.614425	-12511.800838	26296.345828	23336.614401	23336.614449	-16760.162873	-16760.162873	
19	2020-01-20	23343.487590	10741.865824	50187.863658	23343.487562	23343.487618	7099.035362	7099.035362	
20	2020-01-21	23350.360755	8383.602316	46184.310592	23350.360725	23350.360786	4354.428079	4354.428079	
21	2020-01-22	23357.233920	5523.952192	44431.749944	23357.233888	23357.233953	1948.568368	1948.568368	
22	2020-01-23	23364.107085	8337.342894	46097.575391	23364.107051	23364.107123	4509.918830	4509.918830	
23	2020-01-24	23370.980250	6735.583421	46055.927504	23370.980214	23370.980290	3194.126381	3194.126381	
24	2020-01-25	23377.853415	-2200.756238	39447.460577	23377.853377	23377.853458	-4345.914147	-4345.914147	
25	2020-01-26	23384.726580	-12956.141437	25627.734338	23384.726538	23384.726627	-16760.162873	-16760.162873	
26	2020-01-27	23391.599745	11280.141735	50341.462514	23391.599699	23391.599798	7099.035362	7099.035362	

Next steps: Generate code with X_tst_forecast View recommended plots New interactive sheet

```
f, ax = plt.subplots(figsize=(14,5))
f.set_figheight(5)
f.set_figwidth(15)
X_tst.plot(kind='line',x='ds', y='y', color='red', label='Test', ax=ax)
X_tst_forecast.plot(kind='line',x='ds',y='yhat', color='blue',label='Forecast', ax=ax)
plt.title('February 2020 Forecast vs Actuals')
plt.show()
```



38	2020-02-08	23474.077724	-381.192282	39371.168743	23474.077634	23474.077833	-4345.914147	-4345.914147

```
mape = mean_absolute_percentage_error(X_tst['y'],X_tst_forecast['yhat'])
print("MAPE",mape)
```

```
MAPE 2.2952508075863325
```

40	2020-02-10	23487.824054	11742.113715	48480.836174	23487.823954	23487.824169	7099.035362	7099.035362

```
# Add Holidays import holidays which is a different libraries
# Handdling holidays
```

```
import holidays
```

```
india_holidays = holidays.India(years = 2018)

holiday_india_df = pd.DataFrame([])

for date, name in sorted(india_holidays.items()):
    print(date, name)
    holiday_india_df = pd.concat([holiday_india_df, pd.DataFrame({'ds': date, 'holiday': name}, index=[0])], ignore_ind
```

```
2018-01-26 Republic Day
2018-03-13 Maha Shivaratri
2018-03-29 Mahavir Jayanti
2018-03-30 Good Friday
2018-04-30 Buddha Purnima
2018-06-16 Eid al-Fitr
2018-08-15 Independence Day
2018-08-22 Eid al-Adha
2018-09-03 Janmashtami
2018-09-21 Ashura
2018-10-02 Gandhi Jayanti
2018-10-19 Dussehra
2018-11-07 Diwali
2018-11-21 Prophet's Birthday
2018-11-23 Guru Nanak Jayanti
2018-12-25 Christmas
```

47	2020-02-17	23535.936209	10955.268231	50916.557943	23535.936075	23535.936357	7099.035362	7099.035362
48	2020-02-18	23542.809374	7128.448738	46494.861470	23542.809236	23542.809525	4354.428079	4354.428079
49	2020-02-19	23549.682539	5802.627538	44672.529650	23549.682396	23549.682696	1948.568368	1948.568368
50	2020-02-20	23556.555704	8629.602838	47123.794519	23556.555558	23556.555867	4509.918830	4509.918830
51	2020-02-21	23563.428869	7722.491668	45729.362031	23563.428717	23563.429038	3194.126381	3194.126381
52	2020-02-22	23570.302034	-292.238138	38104.672555	23570.301877	23570.302207	-4345.914147	-4345.914147

```
holiday_india_df
```

54	2020-02-24	23584.048364	11796.467416	49312.197799	23584.048198	23584.048546	7099.035362	7099.035362
55	2020-02-25	23590.921529	8769.784260	48096.094269	23590.921356	23590.921716	4354.428079	4354.428079
56	2020-02-26	23597.794694	8405.718771	44849.754932	23597.794516	23597.794886	1948.568368	1948.568368

	ds	holiday
0	2018-01-26	Republic Day
1	2018-02-13	Maha Shivaratri
2	2018-03-29	Mahavir Jayanti
3	2018-03-30	Good Friday
4	2018-04-30	Buddha Purnima
5	2018-06-16	Eid al-Fitr
6	2018-08-15	Independence Day

```
india_holidays_mh = holidays.India(years = 2018,subdiv='MH')

holiday_india_mh = pd.DataFrame([])

for date, name in sorted(india_holidays_mh.items()):
    print(date, name)
    holiday_india_mh = pd.concat([holiday_india_mh, pd.DataFrame({'ds': date, 'holiday': name}, index=[0])], ignore_ind
```

```
2018-01-26 Republic Day
2018-02-13 Maha Shivaratri
2018-02-19 Chhatrapati Shivaji Maharaj Jayanti
2018-03-18 Gudi Padwa
2018-03-29 Mahavir Jayanti
2018-03-30 Good Friday
2018-04-14 Dr. B. R. Ambedkar's Jayanti
2018-04-30 Buddha Purnima
2018-05-01 Maharashtra Day
2018-06-16 Eid al-Fitr
2018-08-15 Independence Day
2018-08-22 Eid al-Adha
2018-09-03 Janmashtami
2018-09-21 Ashura
2018-10-02 Gandhi Jayanti
2018-10-19 Dussehra
2018-11-07 Diwali
2018-11-21 Prophet's Birthday
2018-11-23 Guru Nanak Jayanti
2018-12-25 Christmas
```

	ds	holiday
12	2018-11-07	Diwali
13	2018-11-21	Prophet's Birthday
14	2018-11-23	Guru Nanak Jayanti
15	2018-12-25	Christmas

Next steps: Generate code with holiday_india_df | View recommended plots | New interactive sheet

```
holiday_india_mh
```

	ds	holiday
0	2018-01-26	Republic Day
1	2018-02-13	Maha Shivaratri
2	2018-02-19	Chhatrapati Shivaji Maharaj Jayanti
3	2018-03-18	Gudi Padwa
4	2018-03-29	Mahavir Jayanti
5	2018-03-30	Good Friday
6	2018-04-14	Dr. B. R. Ambedkar's Jayanti
7	2018-04-30	Buddha Purnima
8	2018-05-01	Maharashtra Day
9	2018-06-16	Eid al-Fitr
10	2018-08-15	Independence Day
11	2018-08-22	Eid al-Adha
12	2018-09-03	Janmashtami
13	2018-09-21	Ashura
14	2018-10-02	Gandhi Jayanti
15	2018-10-19	Dussehra
16	2018-11-07	Diwali
17	2018-11-21	Prophet's Birthday
18	2018-11-23	Guru Nanak Jayanti
19	2018-12-25	Christmas

Next steps: Generate code with holiday_india_mh | View recommended plots | New interactive sheet

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

```
holiday = pd.DataFrame([])
for date, name in sorted(holidays.UnitedStates(years=[2018,2019,2020]).items()):
    holiday = pd.concat([holiday, pd.DataFrame({'ds': date, 'holiday': "US-Holidays"}, index=[0])], ignore_index=True)
holiday['ds'] = pd.to_datetime(holiday['ds'], format='%Y-%m-%d', errors='ignore')
```

```
/tmp/ipython-input-3775490065.py:4: FutureWarning: errors='ignore' is deprecated and will raise in a future version. Use
  holiday['ds'] = pd.to_datetime(holiday['ds'], format='%Y-%m-%d', errors='ignore')
```

Start coding or generate with AI.

```
holiday.tail(7)
```

	ds	holiday
25	2020-07-03	US-Holidays
26	2020-07-04	US-Holidays
27	2020-09-07	US-Holidays
28	2020-10-12	US-Holidays
29	2020-11-11	US-Holidays
30	2020-11-26	US-Holidays
31	2020-12-25	US-Holidays

```
holiday_2 = pd.DataFrame([])

for date, name in sorted(holidays.India(years=[2018,2019,2020]).items()):
```

```
    holiday_2 = pd.concat([holiday_2, pd.DataFrame({'ds': date, 'holiday': "India-Holidays"}, index=[0])], ignore_index=
holiday_2['ds'] = pd.to_datetime(holiday_2['ds'], format='%Y-%m-%d', errors='ignore')
```

```
/tmp/ipython-input-2575076591.py:5: FutureWarning: errors='ignore' is deprecated and will raise in a future version. Use
  holiday_2['ds'] = pd.to_datetime(holiday_2['ds'], format='%Y-%m-%d', errors='ignore')
```

```
m_2 = Prophet()
m_2.add_country_holidays(country_name='IN')
m_2.fit(df)
```

```
/usr/local/lib/python3.12/dist-packages/holidays/countries/india.py:182: Warning: Requested Holidays are available only f
  warnings.warn(warning_msg, Warning)
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmp0z1ib4sr/t7uz5p4s.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp0z1ib4sr/ssdwkhcl.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.12/dist-packages/prophet/stan_model/prophet_model.bin', 'random',
11:07:12 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
11:07:12 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
<prophet.forecaster.Prophet at 0x7b485824dd30>
```

```
m_2.train_holiday_names
```

	0
0	Republic Day
1	Independence Day
2	Gandhi Jayanti
3	Buddha Purnima
4	Diwali
5	Janmashtami
6	Dussehra
7	Mahavir Jayanti
8	Maha Shivaratri
9	Guru Nanak Jayanti
10	Ashura
11	Prophet's Birthday
12	Eid al-Fitr
13	Eid al-Adha
14	Good Friday
15	Christmas

dtype: object

```
# fit with holidays
```

```
model_with_holidays = Prophet(holidays=holiday)
model_with_holidays.fit(X_tr)
```

```
INFO:prophet:Disabling yearly seasonality. Run prophet with yearly_seasonality=True to override this.
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmp0z1ib4sr/u236qj7z.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp0z1ib4sr/6j0tlids.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.12/dist-packages/prophet/stan_model/prophet_model.bin', 'random',
11:07:20 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
11:07:20 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
<prophet.forecaster.Prophet at 0x7b4853d97950>
```

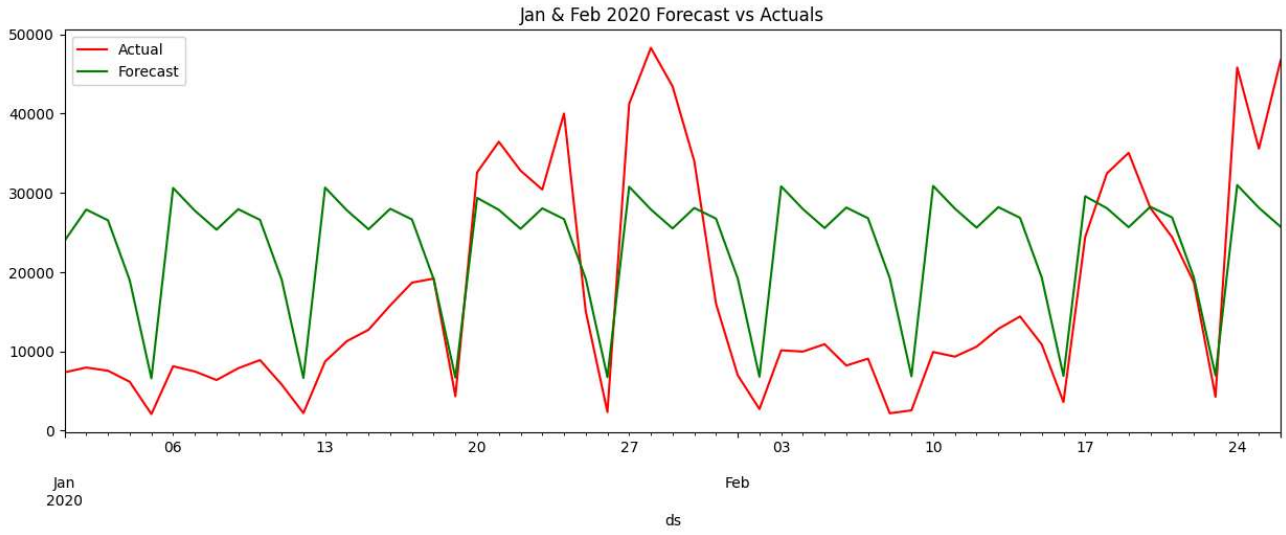
```
future_holiday = model_with_holidays.make_future_dataframe(periods=57, freq='D')
forecast_holiday = model_with_holidays.predict(future_holiday)
forecast_holiday[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail(7)
```

	ds	yhat	yhat_lower	yhat_upper
749	2020-02-20	28244.407248	8627.550979	47359.315458
750	2020-02-21	26893.953958	8577.026275	46454.202199
751	2020-02-22	19356.400851	748.003899	39839.221352
752	2020-02-23	6955.481209	-11189.434969	26049.579608
753	2020-02-24	30969.646063	11656.259083	50224.605022
754	2020-02-25	28108.442406	9062.089398	47660.871262
755	2020-02-26	25710.999870	7436.656987	45082.128230

```
X_tst_forecast_holiday = model_with_holidays.predict(X_tst)
X_tst_forecast_holiday[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail(7)
```

	ds	yhat	yhat_lower	yhat_upper
50	2020-02-20	28244.407248	9496.894129	47326.632143
51	2020-02-21	26893.953958	6486.930415	44407.484919
52	2020-02-22	19356.400851	665.627959	38331.919280
53	2020-02-23	6955.481209	-11036.942675	25689.891979
54	2020-02-24	30969.646063	12728.829587	48834.770698
55	2020-02-25	28108.442406	9935.542834	47212.249117
56	2020-02-26	25710.999870	6271.306900	44589.431292

```
f, ax = plt.subplots(figsize=(14,5))
f.set_figheight(5)
f.set_figwidth(15)
X_tst.plot(kind='line',x='ds', y='y', color='red', label='Actual', ax=ax)
X_tst_forecast_holiday.plot(kind='line',x='ds',y='yhat', color='green',label='Forecast', ax=ax)
plt.title('Jan & Feb 2020 Forecast vs Actuals')
plt.show()
```




```
mape_holiday = mean_absolute_percentage_error(X_tst['y'],X_tst_forecast_holiday['yhat'])
print("MAPE",round(mape_holiday,4))
```

```
MAPE 1.3068
```

```
# n_changepoints is the number of change happen in the data. Prophet model detects them by its own. By default, its val

# changepoint_prior_scale to indicate how flexible the changepoints are allowed to be. In other words, how much can the
```

```
# seasonality_mode There are 2 types model seasonality mode. Additive & multiplicaticative. By default Prophet fits add

# holiday_prior_scale just like changepoint_prior_scale, holiday_prior_scale is used to smoothning the effect of holida

# Seasonalities with fourier_order Prophet model, by default finds the seasonalities and adds the default parameters of
# seasonality_weekly / seasonality_weekly = True / Fl
```

```
from sklearn.model_selection import ParameterGrid
params_grid = {'seasonality_mode':('multiplicative','additive'),
               'changepoint_prior_scale':[0.1,0.2,0.3],
              'holidays_prior_scale':[0.1,0.2,0.3],
              'n_changepoints' : [100,150]}
grid = ParameterGrid(params_grid)
cnt = 0
for p in grid:
    cnt = cnt+1

print('Total Possible Models',cnt)
```

```
Total Possible Models 36
```

```
%%timeit
strt='2019-12-31'
end='2020-02-26'
model_parameters = pd.DataFrame(columns = ['MAPE','Parameters'])
for p in grid:
    test = pd.DataFrame()
    print(p)
    random.seed(0)
    train_model =Prophet(changepoint_prior_scale = p['changepoint_prior_scale'],
                         holidays_prior_scale = p['holidays_prior_scale'],
                         n_changepoints = p['n_changepoints'],
                         seasonality_mode = p['seasonality_mode'],
                         weekly_seasonality=True,
                         daily_seasonality = True,
                         yearly_seasonality = True,
                         holidays=holiday,
                         interval_width=0.95)
    train_model.add_country_holidays(country_name='US')
    train_model.fit(X_tr)
    train_forecast = train_model.make_future_dataframe(periods=57, freq='D',include_history = False)
    train_forecast = train_model.predict(train_forecast)
    test=train_forecast[['ds','yhat']]
    Actual = df[(df['ds']>strt) & (df['ds']<=end)]
    MAPE = mean_absolute_percentage_error(Actual['y'],abs(test['yhat']))
    print('Mean Absolute Percentage Error(MAPE)------------------------------------',MAPE)
    model_parameters = pd.concat([model_parameters, pd.DataFrame({'MAPE':MAPE,'Parameters':p},index=[0])],ignore_index=
```

Show hidden output

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
import pandas as pd
import numpy as np
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_absolute_percentage_error

# Example dataset (replace with your own)
data = [112,118,132,129,121,135,148,148,136,119,104,118,
        115,126,141,135,125,149,170,170,158,133,114,140]

# Convert to pandas series
series = pd.Series(data)

# Store results
model_parameters = pd.DataFrame(columns=['p','d','q','MAPE'])

# Try multiple (p,d,q)
p_values = [0,1,2]
d_values = [0,1]
q_values = [0,1,2]

for p in p_values:
```

```
    for d in d_values:
        for q in q_values:
            try:
                model = ARIMA(series, order=(p,d,q))
                model_fit = model.fit()

                predictions = model_fit.predict(start=0, end=len(series)-1)
                mape = mean_absolute_percentage_error(series, predictions)

                model_parameters.loc[len(model_parameters)] = [p,d,q,mape]
            except:
                continue
                import pandas as pd
import numpy as np
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_absolute_percentage_error

# Example dataset
data = [112,118,132,129,121,135,148,148,136,119,104,118,
        115,126,141,135,125,149,170,170,158,133,114,140]

series = pd.Series(data)

# DataFrame to store results
model_parameters = pd.DataFrame(columns=['p','d','q','MAPE'])

p_values = [0,1,2]
d_values = [0,1]
q_values = [0,1,2]

for p in p_values:
    for d in d_values:
        for q in q_values:
            try:
                model = ARIMA(series, order=(p,d,q))
                model_fit = model.fit()
                predictions = model_fit.predict(start=0, end=len(series)-1)
                mape = mean_absolute_percentage_error(series, predictions)
                model_parameters.loc[len(model_parameters)] = [p,d,q,mape]
            except:
                continue

# ✅ Create a new column "Parameters" from p,d,q
model_parameters['Parameters'] = model_parameters.apply(
    lambda row: f"({row.p},{row.d},{row.q})", axis=1
)

# ✅ Sort by MAPE
parameters = model_parameters.sort_values(by=['MAPE']).reset_index(drop=True)

print(parameters[['Parameters','MAPE']])


# Sort by MAPE
parameters = model_parameters.sort_values(by=['MAPE']).reset_index(drop=True)
print(parameters.head())
```

```
/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/statespace/sarimax.py:978: UserWarning: Non-invertible starting M
  warn('Non-invertible starting MA parameters found.'
/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/statespace/sarimax.py:978: UserWarning: Non-invertible starting M
  warn('Non-invertible starting MA parameters found.'
      Parameters      MAPE
0   (0.0,0.0,2.0)  0.071397
1   (2.0,0.0,2.0)  0.071478
2   (1.0,0.0,2.0)  0.071503
3   (2.0,0.0,1.0)  0.073555
4   (2.0,0.0,0.0)  0.075784
5   (1.0,0.0,1.0)  0.077804
6   (0.0,0.0,1.0)  0.078978
7   (1.0,0.0,0.0)  0.086338
8   (0.0,0.0,0.0)  0.103599
9   (2.0,1.0,2.0)  0.112170
10  (2.0,1.0,1.0)  0.112890
11  (1.0,1.0,2.0)  0.116109
12  (2.0,1.0,0.0)  0.116266
13  (0.0,1.0,2.0)  0.121708
14  (0.0,1.0,1.0)  0.123195
15  (1.0,1.0,1.0)  0.123999
16  (1.0,1.0,0.0)  0.131988
17  (0.0,1.0,0.0)  0.132371
```

```
     p    d    q      MAPE     Parameters
0  0.0  0.0  2.0  0.071397  (0.0,0.0,2.0)
1  2.0  0.0  2.0  0.071478  (2.0,0.0,2.0)
2  1.0  0.0  2.0  0.071503  (1.0,0.0,2.0)
3  2.0  0.0  1.0  0.073555  (2.0,0.0,1.0)
4  2.0  0.0  0.0  0.075784  (2.0,0.0,0.0)
```

```
model_parameters['Parameters'] = model_parameters.apply(lambda row: f"({row.p},{row.d},{row.q})", axis=1)
print(model_parameters[['Parameters','MAPE']])

parameters = parameters.reset_index(drop=True)
parameters.head()
```

```
       Parameters      MAPE
0   (0.0,0.0,0.0)  0.103599
1   (0.0,0.0,1.0)  0.078978
2   (0.0,0.0,2.0)  0.071397
3   (0.0,1.0,0.0)  0.132371
4   (0.0,1.0,1.0)  0.123195
5   (0.0,1.0,2.0)  0.121708
6   (1.0,0.0,0.0)  0.086338
7   (1.0,0.0,1.0)  0.077804
8   (1.0,0.0,2.0)  0.071503
9   (1.0,1.0,0.0)  0.131988
10  (1.0,1.0,1.0)  0.123999
11  (1.0,1.0,2.0)  0.116109
12  (2.0,0.0,0.0)  0.075784
13  (2.0,0.0,1.0)  0.073555
14  (2.0,0.0,2.0)  0.071478
15  (2.0,1.0,0.0)  0.116266
16  (2.0,1.0,1.0)  0.112890
17  (2.0,1.0,2.0)  0.112170
```

	p	d	q	MAPE	Parameters
0	0.0	0.0	2.0	0.071397	(0.0,0.0,2.0)
1	2.0	0.0	2.0	0.071478	(2.0,0.0,2.0)
2	1.0	0.0	2.0	0.071503	(1.0,0.0,2.0)
3	2.0	0.0	1.0	0.073555	(2.0,0.0,1.0)
4	2.0	0.0	0.0	0.075784	(2.0,0.0,0.0)

Next steps: Generate code with parameters | View recommended plots | New interactive sheet

```
model_parameters['Parameters'][0]
```

```
'(0.0,0.0,0.0)'
```

```
parameters['Parameters'][0]
```

```
'(0.0,0.0,2.0)'
```

```
Start coding or generate with AI.
```

```
final_model = Prophet(holidays=holiday,
                      changepoint_prior_scale= 0.1,
                      holidays_prior_scale = 0.2,
                      n_changepoints = 100,
                      seasonality_mode = 'multiplicative',
                      weekly_seasonality=True,
                      daily_seasonality = True,
                      yearly_seasonality = True,
                      interval_width=0.95)
final_model.add_country_holidays(country_name='US')
final_model.fit(X_tr)
```

```
DEBUG:cmdstanpy:input tempfile: /tmp/tmp0z1ib4sr/9dhho2cc.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp0z1ib4sr/pujcny7o.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.12/dist-packages/prophet/stan_model/prophet_model.bin', 'random',
11:27:16 - cmdstanpy - INFO - Chain [1] start processing
```

```
INFO:cmdstanpy:Chain [1] start processing
11:27:16 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
<prophet.forecaster.Prophet at 0x7b4853d97c50>
```

```
future = final_model.make_future_dataframe(periods=122, freq='D')
forecast = final_model.predict(future)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail(7)
```

	ds	yhat	yhat_lower	yhat_upper
814	2020-04-25	11727.404577	-14962.179358	39757.527056
815	2020-04-26	-1573.550753	-29578.158527	23562.729555
816	2020-04-27	25615.198006	-6192.753471	52199.542551
817	2020-04-28	22843.821639	-3557.400831	51028.325135
818	2020-04-29	20609.757573	-7701.741697	47250.524910
819	2020-04-30	23300.399235	-3746.720639	48027.254729
820	2020-05-01	22562.749546	-5559.326643	51505.668515

```
fig =final_model.plot_components(forecast)
```