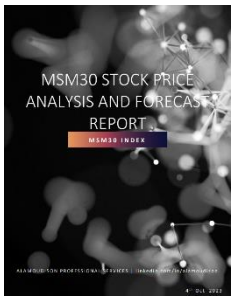
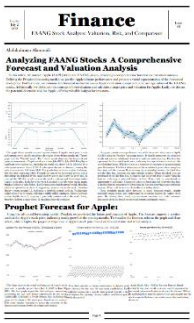
Guide to Analyzing Financial Data Using Facebook Prophet

MSM30 analysis powered by Facebook Prophet MSM30



Check out Finance Issue #2 for an in-depth analysis of FAANG stocks including insights from Facebook Prophet and other tools!



Introduction:

This guide will walk you through the process of analyzing financial data using Facebook Prophet in Google Colab. With Prophet, you can forecast future trends in stock prices, enabling informed decision-making in financial investments.

Prerequisites:

- Access to Google Colab
- Basic understanding of Python and data analysis concepts

Steps:

1. Open the Colab Notebook:

- Open Google Colab and upload the provided notebook file (.ipynb).
- Alternatively, create a new notebook and copy-paste the provided code.

2. Install Necessary Libraries:

- Run the code cell to install required libraries such as Prophet and Plotly.

3. Import Dataset from Yahoo Finance:

- Use the provided code to import the dataset from Yahoo Finance.
- The dataset should be in CSV format and contain columns like Date, Open, High, Low, Close, Adj Close, and Volume.

4. Data Visualization:

- Explore the dataset by visualizing it using Plotly Express.
- Generate line graphs, area plots, box plots, and bar charts to understand the trends in stock prices and trading volume.

5. Prepare Data for Facebook Prophet:

- Rename the columns to 'ds' (Date) and 'y' (Close) as required by Prophet.
- Create a new DataFrame with these renamed columns.

6. Create Facebook Prophet Model:

- Initialize a Prophet object and fit it with the prepared dataset.

- Optionally, you can customize the model by adding holidays or adjusting other parameters.

7. Forecast Future Trends:

- Generate future dates using the make_future_dataframe() method.
- Use the predict() method to forecast future stock prices.

8. Visualize Forecasted Data:

- Plot the forecasted data alongside the historical data using Plotly Express.
- Explore the trend, seasonality, and uncertainty in the forecast.

Using the Notebook:

- Upon opening the notebook, ensure that all code cells are executed sequentially.
- To replace the dataset:

1. Upload your new dataset in CSV format to Google Colab.
2. Replace the filename in the code cell where the dataset is imported (e.g., `df = pd.read_csv("your_dataset.csv")`).
3. Run all subsequent cells to analyze the new dataset and generate updated forecasts.

Conclusion:

By following this guide, you can effectively analyze financial data using Facebook Prophet in Google Colab. With the ability to forecast future trends, you can make informed decisions in the dynamic world of financial markets.

Quick and Easy Financial Analysis:

- With just a few simple steps, anyone can utilize this tool:
  1. Download the dataset from Yahoo Finance.
  2. Upload it to Google Colab.
  3. Replace the filename with the new file name.
  4. Run all cells.

  In under 45 seconds, you'll have your analysis ready to explore!"

Appendices

Appendix 1: code only

importing all the necessary libraries

In [ ]:
```
import pandas as pd
import plotly.express as px
```

In [ ]:
```
#initializing plotly
import plotly.io as pio
pio.renderers.default='colab'
```

In [ ]:
```
!pip install prophet
import prophet
```

In [ ]:
```
from prophet import Prophet
```

after instalaition , import dataset from yahoo.finance

In [ ]:
```
df= pd.read_csv("^TASI.SR.csv")
```

In [ ]:
```
df
```

In [ ]:
```
df.info()
```

In [ ]:
```
df.describe()
```

data visualization using plotly

In [ ]:
```
#Line graph, area garph , box plot (analyzing price and volume)
px.area(df, x="Date" , y="Close")
```

In [ ]:
```
px.line(df, x="Date" , y="Close")
```

In [ ]:
```
px.area(df, x="Date" , y="Volume")
```

In [ ]:
```
px.bar(df, y="Volume")
```

In [ ]:
```
px.box(df, y="Close")
```

Facebook Prophet

In [ ]:
```

```

Data Preperation

In [ ]:
```
df
```

In [ ]:
```
columns=['Date', "Close"]
ndf= pd.DataFrame(df, columns=columns)
```

In [ ]:
```
ndf
```

In [ ]:
```
prophet_df= ndf.rename(columns={'Date':'ds', 'Close':'y'})
```

In [ ]:
```
prophet_df
```

Creating Facebook Prophet Model

In [ ]:
```
m= Prophet()
m.fit(prophet_df)
```

Forecasting 30 days in the future

In [ ]:
```
future= m.make_future_dataframe(periods=30)
forecast=m.predict(future)
```

In [ ]:
```
forecast
```

In [ ]:
```
px.line(forecast, x='ds' , y='yhat')
```

In [ ]:
```
figure= m.plot(forecast, xlabel='ds' , ylabel='y')
```

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

In [ ]:
```
from google.colab import files
forecast.to_csv('forecast.csv')
files.download('forecast.csv')
```

Appendix 2: code and the output

importing all the necessary libraries

```
[36] import pandas as pd
     import plotly.express as px
```

Double-click (or enter) to edit

```
[37] #initializing plotly
     import plotly.io as pio
     pio.renderers.default='colab'
```

```
[38] !pip install prophet
     import prophet
```

```
Requirement already satisfied: prophet in /usr/local/lib/python3.10/dist-packages (1.1.5)
Requirement already satisfied: cmdstanpy>=1.0.4 in /usr/local/lib/python3.10/dist-packages (from prophet) (1.2.2)
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.10/dist-packages (from prophet) (1.25.2)
Requirement already satisfied: matplotlib>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from prophet) (3.7.1)
Requirement already satisfied: pandas>=1.0.4 in /usr/local/lib/python3.10/dist-packages (from prophet) (2.0.3)
Requirement already satisfied: holidays>=0.25 in /usr/local/lib/python3.10/dist-packages (from prophet) (0.46)
Requirement already satisfied: tqdm>=4.36.1 in /usr/local/lib/python3.10/dist-packages (from prophet) (4.66.2)
Requirement already satisfied: importlib-resources in /usr/local/lib/python3.10/dist-packages (from prophet) (6.4.0)
Requirement already satisfied: stanio<2.0.0,>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from cmdstanpy>=1.0.4->prophet) (0.5.0)
```

```
[39] from prophet import Prophet
```

after instalaition , import dataset from yahoo.finance

```
[40] df= pd.read_csv("^TASI.SR.csv")
```

```
[41] df
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2023-04-17	11029.070313	11163.570313	11028.000000	11163.570313	11163.570313	178160200.0
1	2023-04-25	11178.700195	11265.110352	11141.900391	11265.110352	11265.110352	214604500.0
2	2023-04-26	11264.049805	11330.660156	11216.919922	11307.219727	11307.219727	245937000.0
3	2023-04-27	11278.240234	11291.610352	11238.490234	11271.190430	11271.190430	208169700.0
4	2023-04-30	11301.379883	11350.250000	11300.969727	11307.769531	11307.769531	174623000.0
...	...	...	...	...	...	...	...
241	2024-04-04	12664.009766	12725.719727	12609.919922	12705.419922	12705.419922	271114700.0
242	2024-04-14	12465.719727	12727.519531	12458.980469	12666.900391	12666.900391	243170700.0

```
[42] df.info()
```

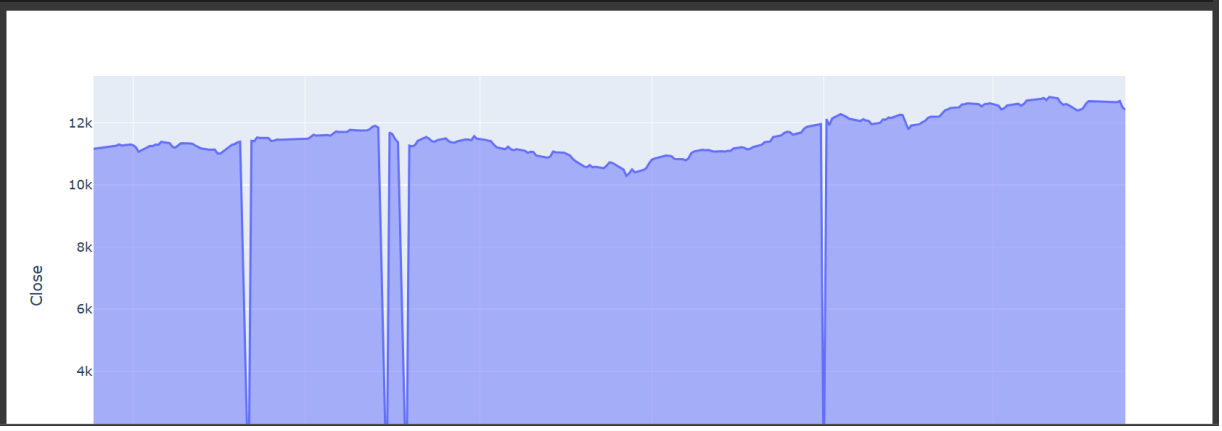
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 246 entries, 0 to 245
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       246 non-null    object
 1   Open       242 non-null    float64
 2   High       242 non-null    float64
 3   Low        242 non-null    float64
 4   Close      242 non-null    float64
 5   Adj Close  242 non-null    float64
 6   Volume     242 non-null    float64
dtypes: float64(6), object(1)
memory usage: 13.6+ KB
```

```
[43] df.describe()
```

	Open	High	Low	Close	Adj Close	Volume
count	242.000000	242.000000	242.000000	242.000000	242.000000	2.420000e+02
mean	11582.998584	11635.003910	11531.620787	11584.257962	11584.257962	2.655380e+08
std	616.030624	619.174637	615.659085	617.634652	617.634652	9.433284e+07

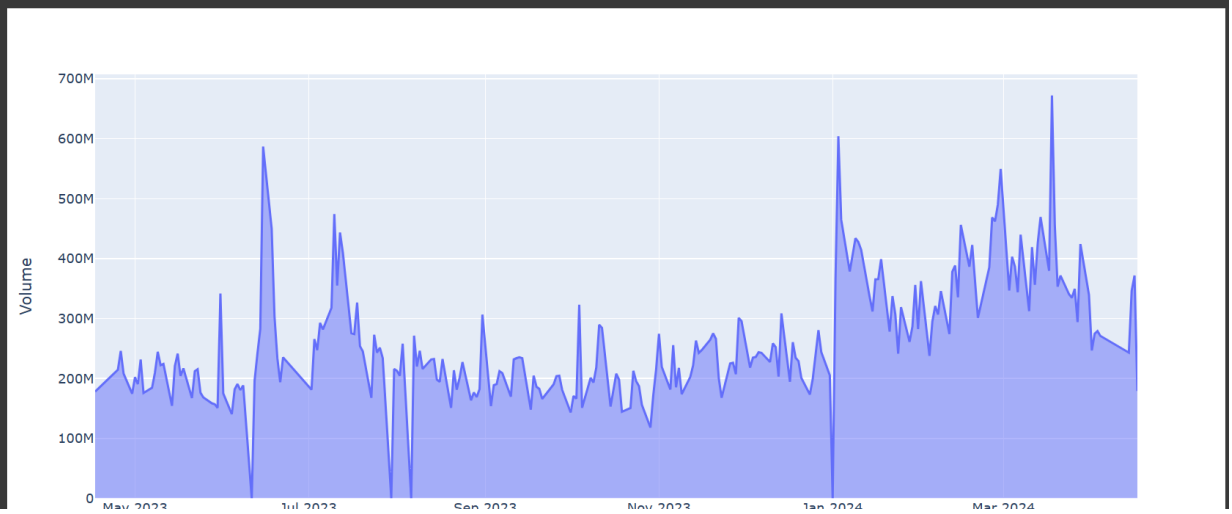
data visualization using plotly

```
[44] #line graph, area garph , box plot (analyzing price and volume)
     px.area(df, x="Date" , y="Close")
```

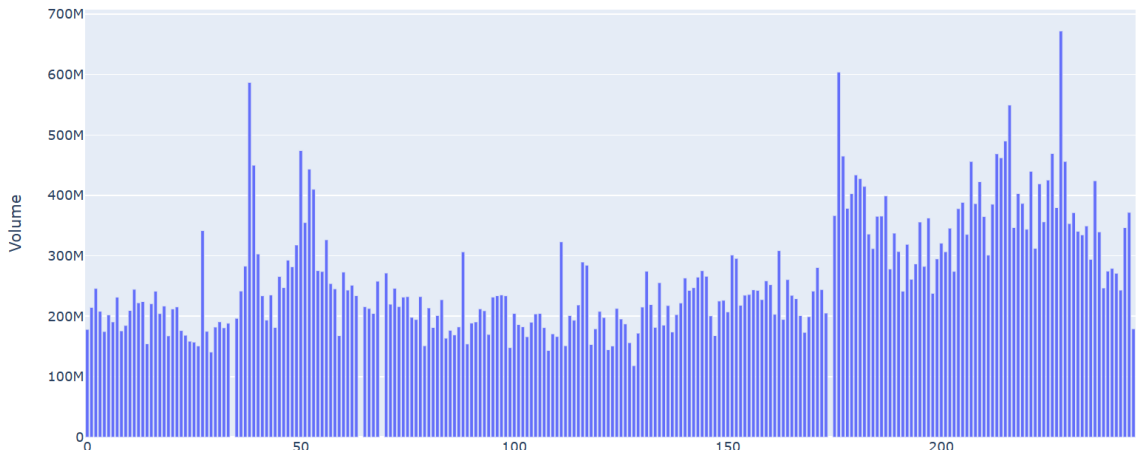



```
[45] px.line(df, x="Date" , y="Close")
```

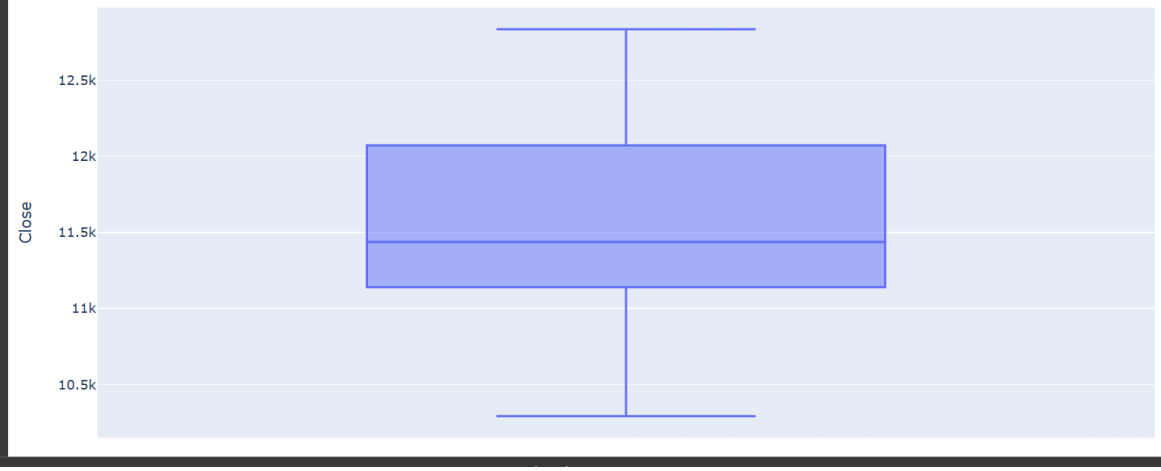

```
[46] px.area(df, x="Date" , y="Volume")
```



```
px.bar(df, y="Volume")
```




```
[48] px.box(df, y="Close")
```



Facebook Prophet

```
[48] Start coding or generate with AI.
```

Data Preperation

```
[49] df
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2023-04-17	11029.070313	11163.570313	11028.000000	11163.570313	11163.570313	178160200.0
1	2023-04-25	11178.700195	11265.110352	11141.900391	11265.110352	11265.110352	214604500.0
2	2023-04-26	11264.049805	11330.660156	11216.919922	11307.219727	11307.219727	245937000.0
3	2023-04-27	11278.240234	11291.610352	11238.490234	11271.190430	11271.190430	208169700.0
4	2023-04-30	11301.379883	11350.250000	11300.969727	11307.769531	11307.769531	174623000.0
...	...	...	...	...	...	...	...
241	2024-04-04	12664.009766	12725.719727	12609.919922	12705.419922	12705.419922	271114700.0
242	2024-04-14	12465.719727	12727.519531	12458.980469	12666.900391	12666.900391	243170700.0
243	2024-04-15	12665.889648	12812.129883	12640.440430	12708.339844	12708.339844	346583400.0

```
[50] columns=['Date', "Close"]
     ndf= pd.DataFrame(df, columns=columns)
```

```
[51] ndf
```

	Date	Close
0	2023-04-17	11163.570313
1	2023-04-25	11265.110352
2	2023-04-26	11307.219727
3	2023-04-27	11271.190430
4	2023-04-30	11307.769531
...	...	...
241	2024-04-04	12705.419922
242	2024-04-14	12666.900391
243	2024-04-15	12708.339844
244	2024-04-16	12500.429688
245	2024-04-17	12427.849609

```
[54] m= Prophet()
     m.fit(prophet_df)
```

```
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/tmp163w4yzg/2tri4siw.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp163w4yzg/3w3zrud6.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=61074', 'data', '
10:28:35 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
10:28:35 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
<prophet.forecaster.Prophet at 0x7afce9a897e0>
```

Forecasting 30 days in the futuer

```
[55] future= m.make_future_dataframe(periods=30)
     forecast=m.predict(future)
```

```
[56] forecast
```

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Colab AI

```
[52] prophet_df= ndf.rename(columns={'Date':'ds', 'Close':'y'})
```

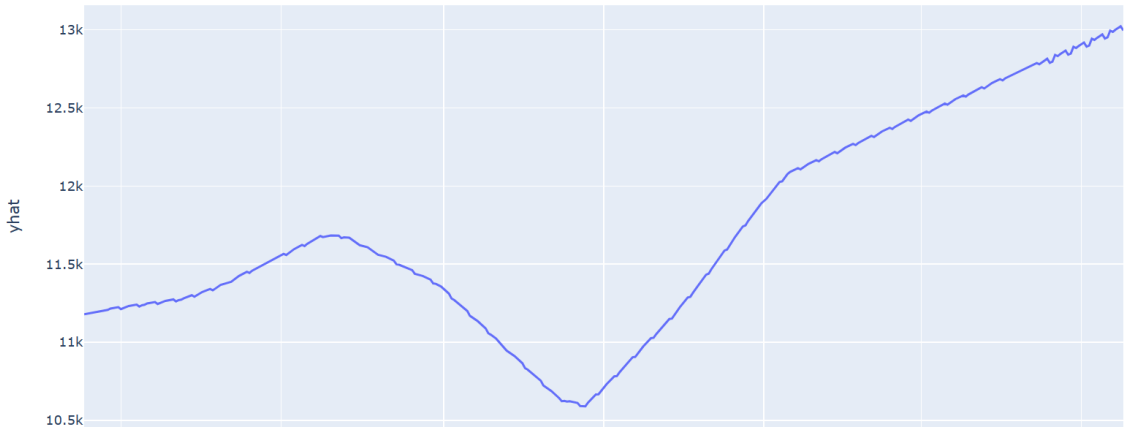
```
prophet_df
```

	ds	y
0	2023-04-17	11163.570313
1	2023-04-25	11265.110352
2	2023-04-26	11307.219727
3	2023-04-27	11271.190430
4	2023-04-30	11307.769531
...	...	...
241	2024-04-04	12705.419922
242	2024-04-14	12666.900391
243	2024-04-15	12708.339844
244	2024-04-16	12500.429688
245	2024-04-17	12427.849609

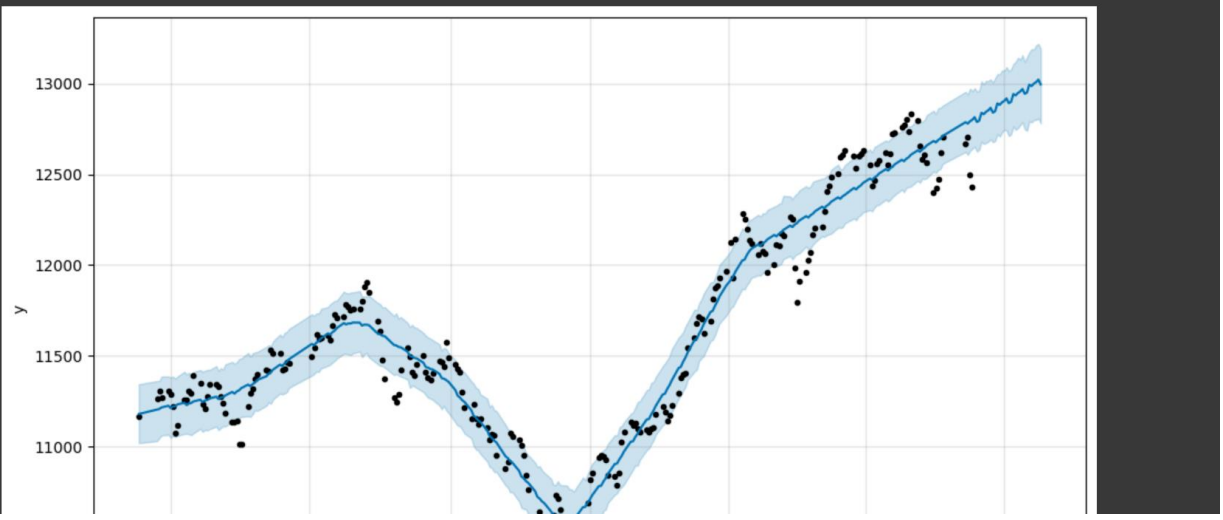
```
forecast
```

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_terms	additive_terms_lower	additive_terms_upper	w
0	2023-04-17	11178.978666	11018.817063	11343.485985	11178.978666	11178.978666	0.088104	0.088104	0.088104	0.0
1	2023-04-25	11197.791493	11030.186942	11361.111727	11197.791493	11197.791493	6.000810	6.000810	6.000810	6.0
2	2023-04-26	11200.143097	11046.777835	11361.622676	11200.143097	11200.143097	6.714660	6.714660	6.714660	6.7
3	2023-04-27	11202.494700	11061.050116	11385.786925	11202.494700	11202.494700	13.148738	13.148738	13.148738	13.1
4	2023-04-30	11209.549510	11065.254636	11386.180552	11209.549510	11209.549510	15.227331	15.227331	15.227331	15.2
...	...	...	...	...	...	...	...	...	...	
267	2024-05-13	12987.118192	12787.246552	13191.013528	12881.755410	13082.368367	0.088104	0.088104	0.088104	0.0
268	2024-05-14	12994.518178	12799.600354	13188.339260	12880.943729	13095.878773	6.000810	6.000810	6.000810	6.0
269	2024-05-15	13001.918163	12800.848318	13212.303186	12881.031640	13109.723836	6.714660	6.714660	6.714660	6.7

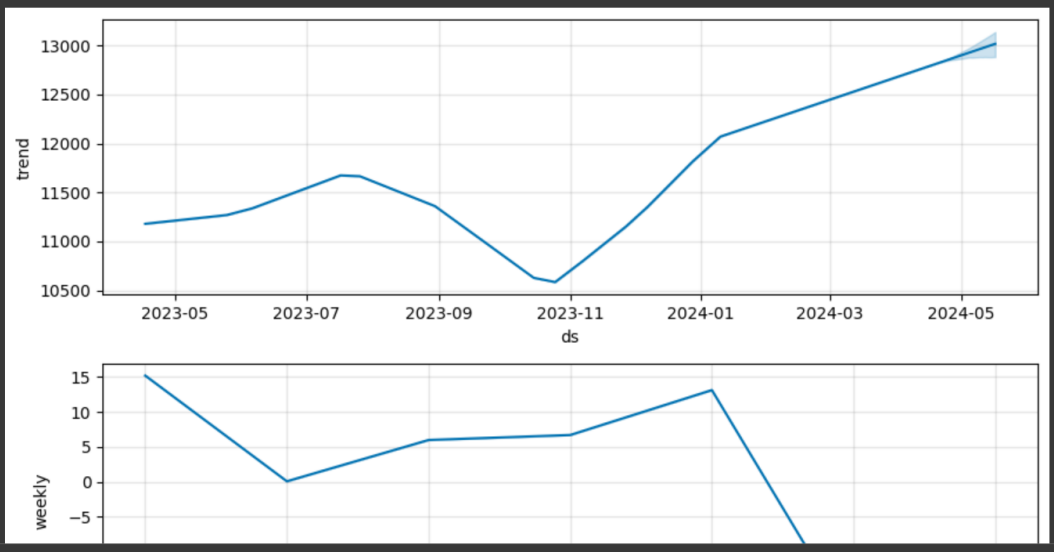
```
[57] px.line(forecast, x='ds' , y='yhat')
```




```
figure= m.plot(forecast, xlabel='ds' , ylabel='y')
```



```
figure2=m.plot_components(forecast)
```

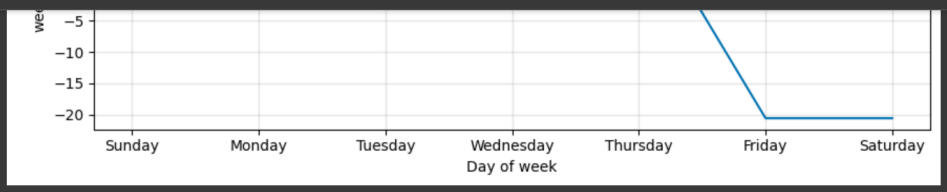


forecasting automation

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

+ Code + Text Reconnect Colab AI




```
[ ] from google.colab import files
    forecast.to_csv('forecast.csv')
    files.download('forecast.csv')
```

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

0s completed at 2:28 PM