```
pip install pandas matplotlib seaborn statsmodels scikit-learn prophet openpyxl joblib
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.11/dist-packages (0.13.2)
Requirement already satisfied: statsmodels in /usr/local/lib/python3.11/dist-packages (0.14.4)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: prophet in /usr/local/lib/python3.11/dist-packages (1.1.6)
Requirement already satisfied: openpyxl in /usr/local/lib/python3.11/dist-packages (3.1.5)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages (1.5.0)
Requirement already satisfied: numpy>=1.23.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.0.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (4.58.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (11.2.1)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.3)
Requirement already satisfied: scipy!=1.9.2,>=1.8 in /usr/local/lib/python3.11/dist-packages (from statsmodels) (1.15.3)
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.11/dist-packages (from statsmodels) (1.0.1)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: cmdstanpy>=1.0.4 in /usr/local/lib/python3.11/dist-packages (from prophet) (1.2.5)
Requirement already satisfied: holidays<1,>=0.25 in /usr/local/lib/python3.11/dist-packages (from prophet) (0.72)
Requirement already satisfied: tqdm>=4.36.1 in /usr/local/lib/python3.11/dist-packages (from prophet) (4.67.1)
Requirement already satisfied: importlib-resources in /usr/local/lib/python3.11/dist-packages (from prophet) (6.5.2)
Requirement already satisfied: et-xmlfile in /usr/local/lib/python3.11/dist-packages (from openpyxl) (2.0.0)
Requirement already satisfied: stanio<2.0.0,>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from cmdstanpy>=1.0.4->prophet) (0.5.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
```

```
from google.colab import files
uploaded = files.upload()
```

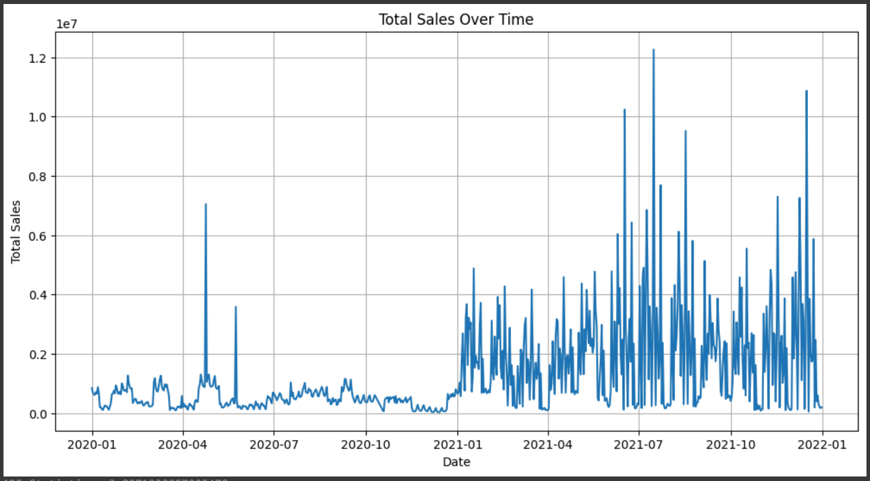
Choose Files Adidas US …atasets.xlsx

- **Adidas US Sales Datasets.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 765895 bytes, last modified: 3/5/2025 - 100% done

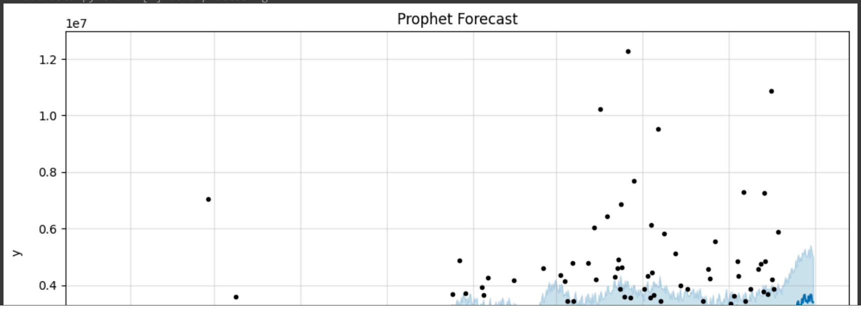
```
Saving Adidas US Sales Datasets.xlsx to Adidas US Sales Datasets.xlsx
```

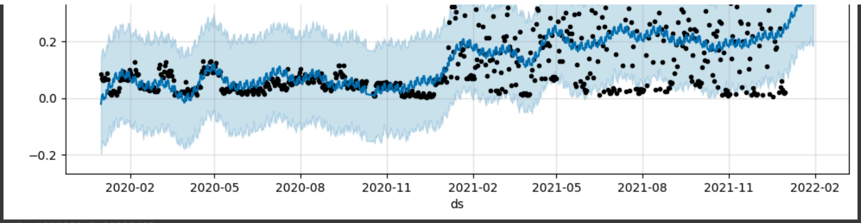
```
# 📦 1. Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.arima.model import ARIMA
from prophet import Prophet
from sklearn.metrics import mean_absolute_error, mean_squared_error
import joblib

# 📁 2. Load and Clean Data
df = pd.read_excel("Adidas US Sales Datasets.xlsx", sheet_name="Data Sales Adidas", skiprows=4)

# Convert Excel serial date to datetime
```

```
df["Invoice Date"] = pd.to_datetime("1899-12-30") + pd.to_timedelta(df["Invoice Date"], unit="D")

# 📊 3. Aggregate Daily Total Sales
daily_sales = df.groupby("Invoice Date")["Total Sales"].sum().reset_index()
daily_sales.columns = ["ds", "y"]  # Prophet requires columns to be named this way

# Plot the time series
plt.figure(figsize=(12, 6))
plt.plot(daily_sales["ds"], daily_sales["y"])
plt.title("Total Sales Over Time")
plt.xlabel("Date")
plt.ylabel("Total Sales")
plt.grid(True)
plt.show()

# 🔄 4. Check for Stationarity (ARIMA prep)
result = adfuller(daily_sales["y"])
print("ADF Statistic:", result[0])
print("p-value:", result[1])

# 📈 5. ARIMA Model (Basic)
model_arima = ARIMA(daily_sales["y"], order=(1, 1, 1))
model_arima_fit = model_arima.fit()
forecast_arima = model_arima_fit.forecast(steps=30)

# 🔮 6. Prophet Model
model_prophet = Prophet()
model_prophet.fit(daily_sales)
future = model_prophet.make_future_dataframe(periods=30)
forecast_prophet = model_prophet.predict(future)

# Plot Prophet Forecast
model_prophet.plot(forecast_prophet)
plt.title("Prophet Forecast")
plt.show()

# 📏 7. Evaluation
actual = daily_sales.set_index("ds").iloc[-30:]["y"]
predicted = forecast_prophet.set_index("ds").iloc[-60:-30]["yhat"]

print("MAE:", mean_absolute_error(actual, predicted))
print("RMSE:", np.sqrt(mean_squared_error(actual, predicted)))

# 💾 8. Save the Model (as .pkl in Colab)
import pickle
with open("prophet_sales_model.pkl", "wb") as f:
    pickle.dump(model_prophet, f)

# Optionally download it
from google.colab import files
files.download("prophet_sales_model.pkl")
```



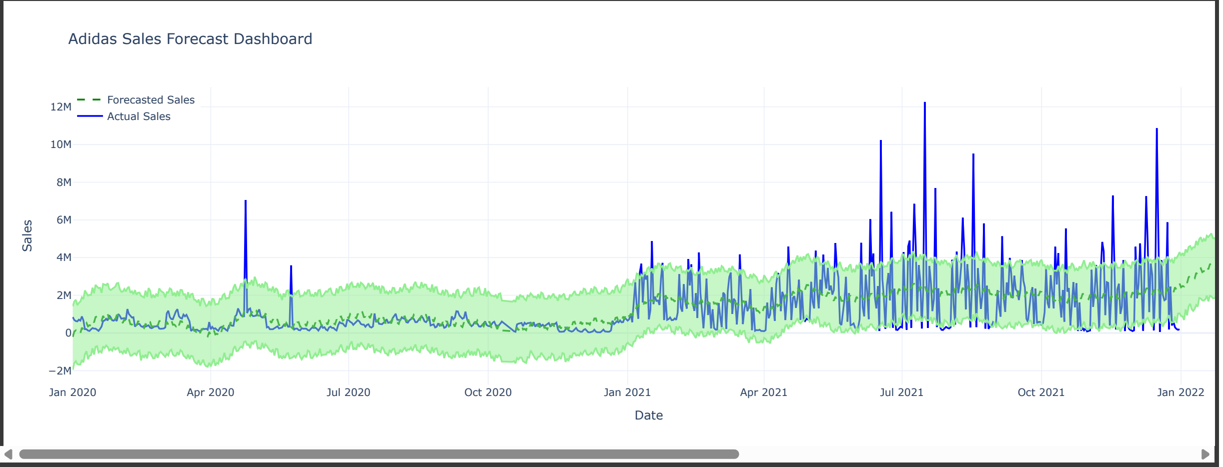
```
ADF Statistic: -3.8271930957005478
p-value: 0.002641031529006596
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmp42l0rw38/ao121kqu.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp42l0rw38/eiaj0vtf.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.11/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=21670', 'data', 'file=/tmp/tmp42l0rw38/ao121kqu.json', '
13:55:44 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
13:55:44 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
```





```
MAE: 1905829.7100933683
RMSE: 2495273.829104833
```

```
!pip install plotly --quiet

import plotly.graph_objects as go

# Plot actual vs forecast
fig = go.Figure()

# Actual sales
fig.add_trace(go.Scatter(
    x=daily_sales["ds"],
    y=daily_sales["y"],
    mode='lines',
    name='Actual Sales',
    line=dict(color='blue')
))

# Forecasted sales
fig.add_trace(go.Scatter(
    x=forecast_prophet["ds"],
    y=forecast_prophet["yhat"],
    mode='lines',
    name='Forecasted Sales',
    line=dict(color='green', dash='dash')
))

# Upper and Lower bounds
fig.add_trace(go.Scatter(
    x=forecast_prophet["ds"],
    y=forecast_prophet["yhat_upper"],
    mode='lines',
    name='Upper Bound',
    line=dict(color='lightgreen'),
    showlegend=False
))
fig.add_trace(go.Scatter(
    x=forecast_prophet["ds"],
    y=forecast_prophet["yhat_lower"],
    fill='tonexty',
    mode='lines',
    name='Lower Bound',
    line=dict(color='lightgreen'),
    showlegend=False
))

# Layout
fig.update_layout(
    title="Adidas Sales Forecast Dashboard",
    xaxis_title="Date",
    yaxis_title="Sales",
    template="plotly_white",
    legend=dict(x=0, y=1)
)

fig.show()
```


Adidas Sales Forecast Dashboard
Forecasted Sales
Actual Sales
12M
10M
8M
6M
4M
2M
−2M
Sales
Jan 2020
Apr 2020
Jul 2020
Oct 2020
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Date