Forecasting Net Prophet

You're a growth analyst at MercadoLibre. With over 200 million users, MercadoLibre is the most popular e-commerce site in Latin America. You've been tasked with analyzing the company's financial and user data in clever ways to make the company grow. So, you want to find out if the ability to predict search traffic can translate into the ability to successfully trade the stock.

Instructions

This section divides the instructions for this Challenge into four steps and an optional fifth step, as follows:

- Step 1: Find unusual patterns in hourly Google search traffic
- Step 2: Mine the search traffic data for seasonality
- Step 3: Relate the search traffic to stock price patterns
- Step 4: Create a time series model with Prophet
- Step 5 (optional): Forecast revenue by using time series models

The following subsections detail these steps.

Step 1: Find Unusual Patterns in Hourly Google Search Traffic

The data science manager asks if the Google search traffic for the company links to any financial events at the company. Or, does the search traffic data just present random noise? To answer this question, pick out any unusual patterns in the Google search data for the company, and connect them to the corporate financial events.

To do so, complete the following steps:

1. Read the search data into a DataFrame, and then slice the data to just the month of May 2020. (During this month, MercadoLibre released its quarterly financial results.) Use hvPlot to visualize the results. Do any unusual patterns exist?
2. Calculate the total search traffic for the month, and then compare the value to the monthly median across all months. Did the Google search traffic increase during the month that MercadoLibre released its financial results?

Step 2: Mine the Search Traffic Data for Seasonality

Marketing realizes that they can use the hourly search data, too. If they can track and predict interest in the company and its platform for any time of day, they can focus their marketing efforts around the times that have the most traffic. This will get a greater return on investment (ROI) from their marketing budget.

To that end, you want to mine the search traffic data for predictable seasonal patterns of interest in the company. To do so, complete the following steps:

1. Group the hourly search data to plot the average traffic by the day of the week (for example, Monday vs. Friday).
2. Using hvPlot, visualize this traffic as a heatmap, referencing the `index.hour` as the x-axis and the `index.dayofweek` as the y-axis. Does any day-of-week effect that you observe concentrate in just a few hours of that day?
3. Group the search data by the week of the year. Does the search traffic tend to increase during the winter holiday period (weeks 40 through 52)?

Step 3: Relate the Search Traffic to Stock Price Patterns

You mention your work on the search traffic data during a meeting with people in the finance group at the company. They want to know if any relationship between the search data and the company stock price exists, and they ask if you can investigate.

To do so, complete the following steps:

1. Read in and plot the stock price data. Concatenate the stock price data to the search data in a single DataFrame.
2. Market events emerged during the year of 2020 that many companies found difficult. But, after the initial shock to global financial markets, new customers and revenue increased for e-commerce platforms. Slice the data to just the first half of 2020 (`2020-01` to `2020-06` in the DataFrame), and then use hvPlot to plot the data. Do both time series indicate a common trend that's consistent with this narrative?
3. Create a new column in the DataFrame named "Lagged Search Trends" that offsets, or shifts, the search traffic by one hour. Create two additional columns:
    - "Stock Volatility", which holds an exponentially weighted four-hour rolling average of the company's stock volatility
    - "Hourly Stock Return", which holds the percent change of the company's stock price on an hourly basis

4. Review the time series correlation, and then answer the following question: Does a predictable relationship exist between the lagged search traffic and the stock volatility or between the lagged search traffic and the stock price returns?

Step 4: Create a Time Series Model with Prophet

Now, you need to produce a time series model that analyzes and forecasts patterns in the hourly search data. To do so, complete the following steps:

1. Set up the Google search data for a Prophet forecasting model.
2. After estimating the model, plot the forecast. How's the near-term forecast for the popularity of MercadoLibre?
3. Plot the individual time series components of the model to answer the following questions:
    - What time of day exhibits the greatest popularity?
    - Which day of the week gets the most search traffic?
    - What's the lowest point for search traffic in the calendar year?

Step 5 (Optional): Forecast Revenue by Using Time Series Models

A few weeks after your initial analysis, the finance group follows up to find out if you can help them solve a different problem. Your fame as a growth analyst in the company continues to grow!

Specifically, the finance group wants a forecast of the total sales for the next quarter. This will dramatically increase their ability to plan budgets and to help guide expectations for the company investors.

To do so, complete the following steps:

1. Read in the daily historical sales (that is, revenue) figures, and then apply a Prophet model to the data.
2. Interpret the model output to identify any seasonal patterns in the company's revenue. For example, what are the peak revenue days? (Mondays? Fridays? Something else?)
3. Produce a sales forecast for the finance group. Give them a number for the expected total sales in the next quarter. Include the best- and worst-case scenarios to help them make better plans.

Install and import the required libraries and dependencies

```
# Install the required libraries
# !pip install pystan
# !pip install fbprophet
# !pip install hvplot
# !pip install holoviews
```

```
# Install the required libraries
from IPython.display import clear_output
try:
  !pip install pystan~=2.14
  !pip install fbprophet
  !pip install hvplot
  !pip install holoviews
except:
  print("Error installing libraries")
finally:
  clear_output()
  print('Libraries successfully installed')
```

```
    Libraries successfully installed
```

```
# Import the required libraries and dependencies
import pandas as pd
import holoviews as hv
import numpy as np
from prophet import Prophet
import hvplot.pandas
import datetime as dt
%matplotlib inline
```

Step 1: Find Unusual Patterns in Hourly Google Search Traffic

The data science manager asks if the Google search traffic for the company links to any financial events at the company. Or, does the search traffic data just present random noise? To answer this question, pick out any unusual patterns in the Google search data for the company, and connect them to the corporate financial events.

To do so, complete the following steps:

1. Read the search data into a DataFrame, and then slice the data to just the month of May 2020. (During this month, MercadoLibre released its quarterly financial results.) Use hvPlot to visualize the results. Do any unusual patterns exist?
2. Calculate the total search traffic for the month, and then compare the value to the monthly median across all months. Did the Google search traffic increase during the month that MercadoLibre released its financial results?

Step 1: Read the search data into a DataFrame, and then slice the data to just the month of May 2020. (During this month, MercadoLibre released its quarterly financial results.) Use hvPlot to visualize the results. Do any unusual patterns exist?

```
# Upload the "google_hourly_search_trends.csv" file into Colab, then store in a Pandas DataFrame
# Set the "Date" column as the Datetime Index.

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

df_mercado_trends = pd.read_csv(
    "google_hourly_search_trends.csv",
    index_col = "Date",
    parse_dates=True,
    infer_datetime_format=True
).dropna()


# Review the first and last five rows of the DataFrame
display(df_mercado_trends.head())
display(df_mercado_trends.tail())
```

Choose Files google_hour...trends.csv

- **google_hourly_search_trends.csv**(text/csv) - 645970 bytes, last modified: 9/17/2022 - 100% done

```
Saving google_hourly_search_trends.csv to google_hourly_search_trends.csv
```

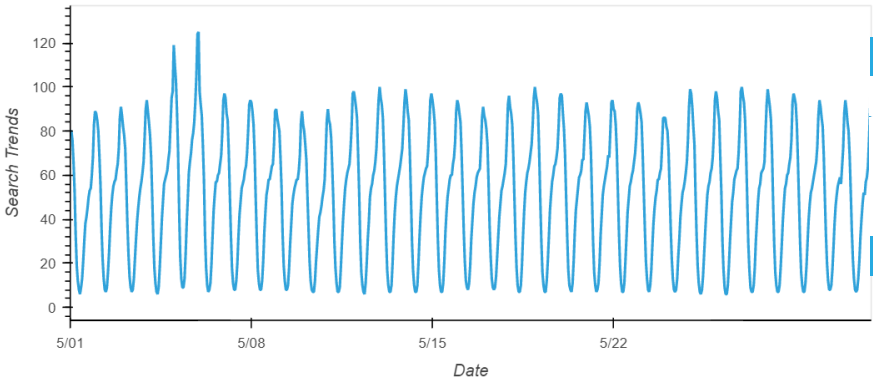
Date	Search Trends
2016-06-01 00:00:00	97
2016-06-01 01:00:00	92
2016-06-01 02:00:00	76
2016-06-01 03:00:00	60
2016-06-01 04:00:00	38

Date	Search Trends
2020-09-07 20:00:00	71
2020-09-07 21:00:00	83
2020-09-07 22:00:00	96
2020-09-07 23:00:00	97
2020-09-08 00:00:00	96

```
# Review the data types of the DataFrame using the info function
df_mercado_trends.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 37106 entries, 2016-06-01 00:00:00 to 2020-09-08 00:00:00
Data columns (total 1 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Search Trends  37106 non-null  int64
dtypes: int64(1)
memory usage: 579.8 KB
```

```
# Holoviews extension to render hvPlots in Colab
hv.extension('bokeh')

# Slice the DataFrame to just the month of May 2020
df_may_2020 = df_mercado_trends.loc["2020-05"]

# Use hvPlot to visualize the data for May 2020
df_may_2020.hvplot()
```



Step 2: Calculate the total search traffic for the month, and then compare the value to the monthly median across all months. Did the Google search traffic increase during the month that MercadoLibre released its financial results?

```
# Calculate the sum of the total search traffic for May 2020
traffic_may_2020 = df_may_2020.sum()

# View the traffic_may_2020 value
traffic_may_2020
```

```
Search Trends    38181
dtype: int64
```

```
# Calcluate the monhtly median search traffic across all months
# Group the DataFrame by index year and then index month, chain the sum and then the median functions
median_monthly_traffic = df_mercado_trends.groupby(by=[df_mercado_trends.index.year,df_mercado_trends.index.month] ).sum().median()

# View the median_monthly_traffic value
median_monthly_traffic
```

```
Search Trends    35172.5
dtype: float64
```

```
# Compare the seach traffic for the month of May 2020 to the overall monthly median value
traffic_may_2020/median_monthly_traffic
```

```
Search Trends    1.085536
dtype: float64
```

Answer the following question:

Question: Did the Google search traffic increase during the month that MercadoLibre released its financial results?

Answer: The Google search traffic for May was higher, approximately 8 percent over the median monthly MercadoLibre traffic.

Step 2: Mine the Search Traffic Data for Seasonality

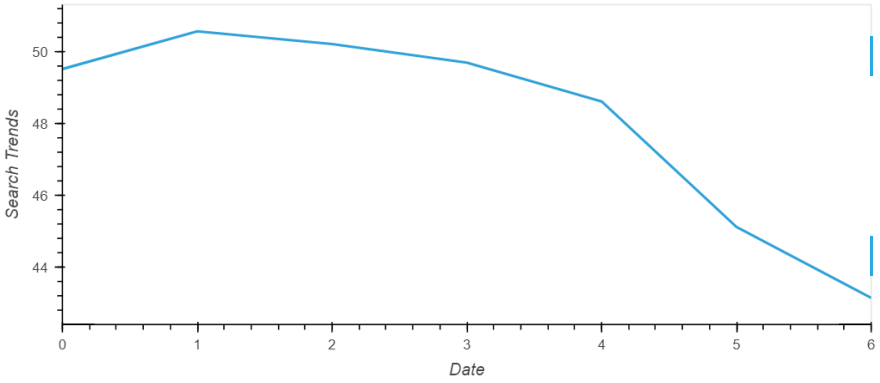
Marketing realizes that they can use the hourly search data, too. If they can track and predict interest in the company and its platform for any time of day, they can focus their marketing efforts around the times that have the most traffic. This will get a greater return on investment (ROI) from their marketing budget.

To that end, you want to mine the search traffic data for predictable seasonal patterns of interest in the company. To do so, complete the following steps:

1. Group the hourly search data to plot the average traffic by the day of the week (for example, Monday vs. Friday).
2. Using hvPlot, visualize this traffic as a heatmap, referencing the `index.hour` as the x-axis and the `index.dayofweek` as the y-axis. Does any day-of-week effect that you observe concentrate in just a few hours of that day?
3. Group the search data by the week of the year. Does the search traffic tend to increase during the winter holiday period (weeks 40 through 52)?

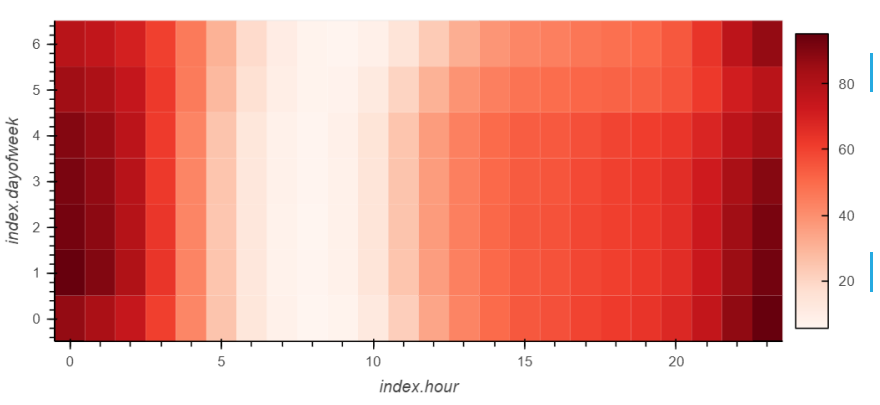
Step 1: Group the hourly search data to plot the average traffic by the day of the week (for example, Monday vs. Friday).

```
# Holoviews extension to render hvPlots in Colab
hv.extension('bokeh')

# Group the hourly search data to plot (use hvPlot) the average traffic by the day of week
df_mercado_trends.groupby(df_mercado_trends.index.dayofweek).mean().hvplot()
```



Step 2: Using hvPlot, visualize this traffic as a heatmap, referencing the `index.hour` as the x-axis and the `index.dayofweek` as the y-axis. Does any day-of-week effect that you observe concentrate in just a few hours of that day?

```
# Holoviews extension to render hvPlots in Colab
hv.extension('bokeh')

# Use hvPlot to visualize the hour of the day and day of week search traffic as a heatmap.
df_mercado_trends.hvplot.heatmap(x='index.hour',
                                 y='index.dayofweek',
                                 C='Search Trends',
                                 cmap='reds').aggregate(function=np.mean)
```



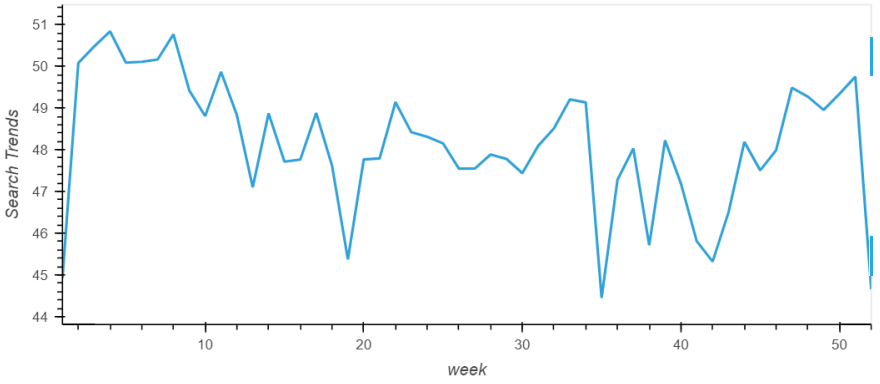
Answer the following question:

Question: Does any day-of-week effect that you observe concentrate in just a few hours of that day?

Answer: Answer: According to the heatmap graph it seems the effect is over the weekend (5 and 6). There is reduced google traffic shown by the lighter areas of the heatmap opposed to when shifted a few hours towards the weekdays.

Step 3: Group the search data by the week of the year. Does the search traffic tend to increase during the winter holiday period (weeks 40 through 52)?

```
# Holoviews extension to render hvPlots in Colab
hv.extension('bokeh')

# Group the hourly search data to plot (use hvPlot) the average traffic by the week of the year
df_mercado_trends.groupby(df_mercado_trends.index.isocalendar().week).mean().hvplot()
```



Answer the following question:

Question: Does the search traffic tend to increase during the winter holiday period (weeks 40 through 52)?

Answer: Yes, the search traffic does tend to increase during the winter holiday period specifically between weeks 41 through 51.

Step 3: Relate the Search Traffic to Stock Price Patterns

You mention your work on the search traffic data during a meeting with people in the finance group at the company. They want to know if any relationship between the search data and the company stock price exists, and they ask if you can investigate.

To do so, complete the following steps:

1. Read in and plot the stock price data. Concatenate the stock price data to the search data in a single DataFrame.
2. Market events emerged during the year of 2020 that many companies found difficult. But, after the initial shock to global financial markets, new customers and revenue increased for e-commerce platforms. Slice the data to just the first half of 2020 (`2020-01` to `2020-06` in the DataFrame), and then use hvPlot to plot the data. Do both time series indicate a common trend that's consistent with this narrative?
3. Create a new column in the DataFrame named "Lagged Search Trends" that offsets, or shifts, the search traffic by one hour. Create two additional columns:
    - "Stock Volatility", which holds an exponentially weighted four-hour rolling average of the company's stock volatility
    - "Hourly Stock Return", which holds the percent change of the company's stock price on an hourly basis
4. Review the time series correlation, and then answer the following question: Does a predictable relationship exist between the lagged search traffic and the stock volatility or between the lagged search traffic and the stock price returns?

Step 1: Read in and plot the stock price data. Concatenate the stock price data to the search data in a single DataFrame.

```
# Upload the "mercado_stock_price.csv" file into Colab, then store in a Pandas DataFrame
# Set the "date" column as the Datetime Index.
from google.colab import files #check these
uploaded = files.upload() #check these
```

```
df_mercado_stock = pd.read_csv(
    "mercado_stock_price.csv",
    index_col="date",
    parse_dates=True,
    infer_datetime_format=True
).dropna()

# View the first and last five rows of the DataFrame
display(df_mercado_stock.head())
display(df_mercado_stock.tail())
```

Choose Files mercado_stock_price.csv

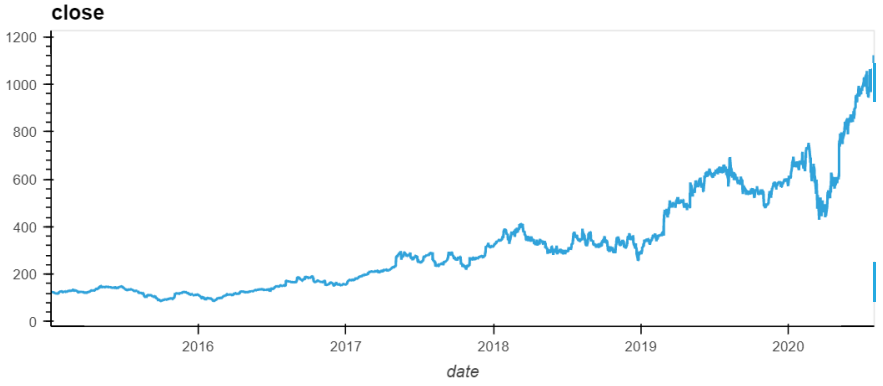
- **mercado_stock_price.csv**(text/csv) - 1082540 bytes, last modified: 9/17/2022 - 100% done

Saving mercado_stock_price.csv to mercado_stock_price.csv

date	close
2015-01-02 09:00:00	127.67
2015-01-02 10:00:00	125.44
2015-01-02 11:00:00	125.57
2015-01-02 12:00:00	125.40
2015-01-02 13:00:00	125.17

date	close
2020-07-31 11:00:00	1105.780
2020-07-31 12:00:00	1087.925
2020-07-31 13:00:00	1095.800
2020-07-31 14:00:00	1110.650
2020-07-31 15:00:00	1122.510

```
# Holoviews extension to render hvPlots in Colab
hv.extension('bokeh')

# Use hvPlot to visualize the closing price of the df_mercado_stock DataFrame
df_mercado_stock['close'].hvplot()
```




```
# Concatenate the df_mercado_stock DataFrame with the df_mercado_trends DataFrame
# Concatenate the DataFrame by columns (axis=1), and drop and rows with only one column of data
mercado_stock_trends_df = pd.concat([df_mercado_stock,df_mercado_trends],axis=1).dropna()

# View the first and last five rows of the DataFrame
display(mercado_stock_trends_df.head())
display(mercado_stock_trends_df.tail())
```

	close	Search Trends
2016-06-01 09:00:00	135.16	6.0
2016-06-01 10:00:00	136.63	12.0
2016-06-01 11:00:00	136.56	22.0
2016-06-01 12:00:00	136.42	33.0
2016-06-01 13:00:00	136.10	40.0

	close	Search Trends
2020-07-31 11:00:00	1105.780	20.0
2020-07-31 12:00:00	1087.925	32.0

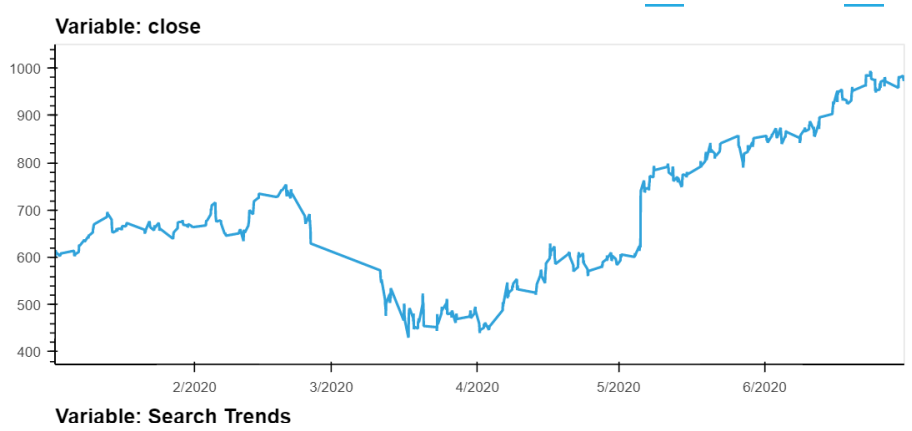
Step 2: Market events emerged during the year of 2020 that many companies found difficult. But, after the initial shock to global financial markets, new customers and revenue increased for e-commerce platforms. Slice the data to just the first half of 2020 (`2020-01` to `2020-06` in the DataFrame), and then use hvPlot to plot the data. Do both time series indicate a common trend that's consistent with this narrative?

```
# For the combined dataframe, slice to just the first half of 2020 (2020-01 through 2020-06)
first_half_2020 = mercado_stock_trends_df['2020-01':'2020-06']

# View the first and last five rows of first_half_2020 DataFrame
display(first_half_2020.head())
display(first_half_2020.tail())
```

	close	Search Trends
2020-01-02 09:00:00	601.085	9.0
2020-01-02 10:00:00	601.290	14.0
2020-01-02 11:00:00	615.410	25.0
2020-01-02 12:00:00	611.400	37.0
2020-01-02 13:00:00	611.830	50.0

	close	Search Trends
2020-06-30 11:00:00	976.17	17.0
2020-06-30 12:00:00	977.50	27.0
2020-06-30 13:00:00	973.23	37.0
2020-06-30 14:00:00	976.50	45.0
2020-06-30 15:00:00	984.93	51.0

```
# Holoviews extension to render hvPlots in Colab
hv.extension('bokeh')

# Use hvPlot to visualize the close and Search Trends data
# Plot each column on a separate axes using the following syntax
# `hvplot(shared_axes=False, subplots=True).cols(1)`
first_half_2020.hvplot(shared_axes=False,subplots=True).cols(1)
```



Answer the following question:

Question: Do both time series indicate a common trend that's consistent with this narrative?

Answer: Answer: Both time series do indicate a common trend that's consistent with this narrative. The low points are from March to early April. There is also a spike on May 5th for a few hours, this could indicate that additional research may be required to determine a final conclusion or analysis determination.

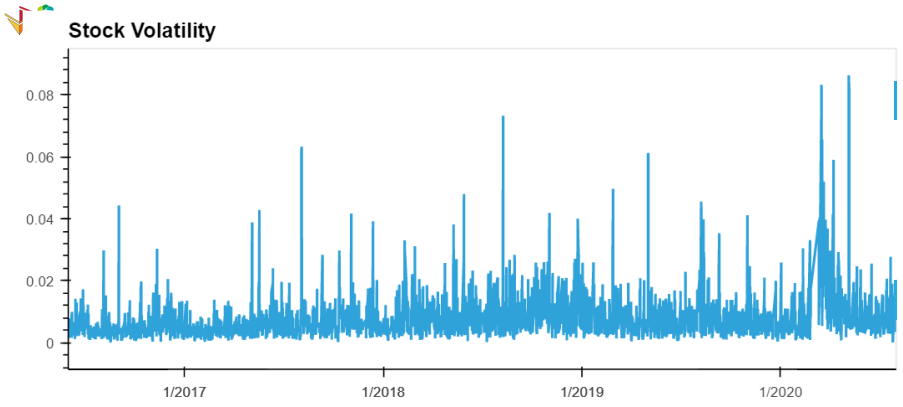
Step 3: Create a new column in the DataFrame named "Lagged Search Trends" that offsets, or shifts, the search traffic by one hour. Create two additional columns:

- "Stock Volatility", which holds an exponentially weighted four-hour rolling average of the company's stock volatility
- "Hourly Stock Return", which holds the percent change of the company's stock price on an hourly basis

```
# Create a new column in the mercado_stock_trends_df DataFrame called Lagged Search Trends
# This column should shift the Search Trends information by one hour
mercado_stock_trends_df['Lagged Search Trends'] = mercado_stock_trends_df['Search Trends'].shift(1)
```

```
# Create a new column in the mercado_stock_trends_df DataFrame called Stock Volatility
# This column should calculate the standard deviation of the closing stock price return data over a 4 period rolling window
mercado_stock_trends_df['Stock Volatility'] = mercado_stock_trends_df['close'].pct_change().rolling(window=4).std()
```

```
# Holoviews extension to render hvPlots in Colab
hv.extension('bokeh')

# Use hvPlot to visualize the stock volatility
mercado_stock_trends_df['Stock Volatility'].hvplot()
```



Solution Note: Note how volatility spiked, and tended to stay high, during the first half of 2020. This is a common characteristic of volatility in stock returns worldwide: high volatility days tend to be followed by yet more high volatility days. When it rains, it pours.

```
# Create a new column in the mercado_stock_trends_df DataFrame called Hourly Stock Return
# This column should calculate hourly return percentage of the closing price
mercado_stock_trends_df['Hourly Stock Return'] = mercado_stock_trends_df['close'].pct_change()
```

```
# View the first and last five rows of the mercado_stock_trends_df DataFrame
display(mercado_stock_trends_df.head())
display(mercado_stock_trends_df.tail())
```

	close	Search Trends	Lagged Search Trends	Stock Volatility	Hourly Stock Return
2016-06-01 09:00:00	135.16	6.0	NaN	NaN	NaN
2016-06-01 10:00:00	136.63	12.0	6.0	NaN	0.010876
2016-06-01 11:00:00	136.56	22.0	12.0	NaN	-0.000512
2016-06-01 12:00:00	136.42	33.0	22.0	NaN	-0.001025
2016-06-01 13:00:00	136.10	40.0	33.0	0.006134	-0.002346

	close	Search Trends	Lagged Search Trends	Stock Volatility	Hourly Stock Return
2020-07-31 11:00:00	1105.780	20.0	11.0	0.012837	0.006380
2020-07-31 12:00:00	1087.925	32.0	20.0	0.013549	-0.016147
2020-07-31 13:00:00	1095.800	41.0	32.0	0.013295	0.007239
2020-07-31 14:00:00	1110.650	47.0	41.0	0.013001	0.013552
2020-07-31 15:00:00	1122.510	53.0	47.0	0.013566	0.010678

Step 4: Review the time series correlation, and then answer the following question: Does a predictable relationship exist between the lagged search traffic and the stock volatility or between the lagged search traffic and the stock price returns?

```
# Construct correlation table of Stock Volatility, Lagged Search Trends, and Hourly Stock Return
mercado_stock_trends_df[['Stock Volatility','Lagged Search Trends','Hourly Stock Return']].corr()
```

	Stock Volatility	Lagged Search Trends	Hourly Stock Return
Stock Volatility	1.000000	-0.148938	0.061424
Lagged Search Trends	-0.148938	1.000000	0.017929
Hourly Stock Return	0.061424	0.017929	1.000000

Answer the following question:

Question: Does a predictable relationship exist between the lagged search traffic and the stock volatility or between the lagged search traffic and the stock price returns?

Answer: There appears to be a minor negative correlation between lagged searches of the firm and stock volitility. Due to the weak correlations the effect is small on one another. It is difficult to predict due to the low correlation across the board.

Step 4: Create a Time Series Model with Prophet

Now, you need to produce a time series model that analyzes and forecasts patterns in the hourly search data. To do so, complete the following steps:

1. Set up the Google search data for a Prophet forecasting model.
2. After estimating the model, plot the forecast. How's the near-term forecast for the popularity of MercadoLibre?
3. Plot the individual time series components of the model to answer the following questions:
    - What time of day exhibits the greatest popularity?
    - Which day of the week gets the most search traffic?
    - What's the lowest point for search traffic in the calendar year?

Step 1: Set up the Google search data for a Prophet forecasting model.

```
# Using the df_mercado_trends DataFrame, reset the index so the date information is no longer the index
mercado_prophet_df = df_mercado_trends.reset_index()

# Label the columns ds and y so that the syntax is recognized by Prophet
mercado_prophet_df.columns = ['ds','y']

# Drop an NaN values from the prophet_df DataFrame
mercado_prophet_df = mercado_prophet_df.dropna()

# View the first and last five rows of the mercado_prophet_df DataFrame
display(mercado_prophet_df.head())
display(mercado_prophet_df.tail())
```

	ds	y
0	2016-06-01 00:00:00	97
1	2016-06-01 01:00:00	92
2	2016-06-01 02:00:00	76
3	2016-06-01 03:00:00	60
4	2016-06-01 04:00:00	38

	ds	y
37101	2020-09-07 20:00:00	71
37102	2020-09-07 21:00:00	83
37103	2020-09-07 22:00:00	96
37104	2020-09-07 23:00:00	97
37105	2020-09-08 00:00:00	96

```
# Call the Prophet function, store as an object
model_mercado_trends = Prophet()
model_mercado_trends
```

```
<prophet.forecaster.Prophet at 0x7fad428a16d0>
```

```
# Fit the time-series model.
model_mercado_trends.fit(mercado_prophet_df)
```

```
<prophet.forecaster.Prophet at 0x7fad428a16d0>
```

```
# Create a future dataframe to hold predictions
# Make the prediction go out as far as 2000 hours (approx 80 days)
future_mercado_trends = model_mercado_trends.make_future_dataframe(periods=2000,freq='H')

# View the last five rows of the future_mercado_trends DataFrame
future_mercado_trends.tail()
```

	ds
39101	2020-11-30 04:00:00
39102	2020-11-30 05:00:00
39103	2020-11-30 06:00:00
39104	2020-11-30 07:00:00
39105	2020-11-30 08:00:00

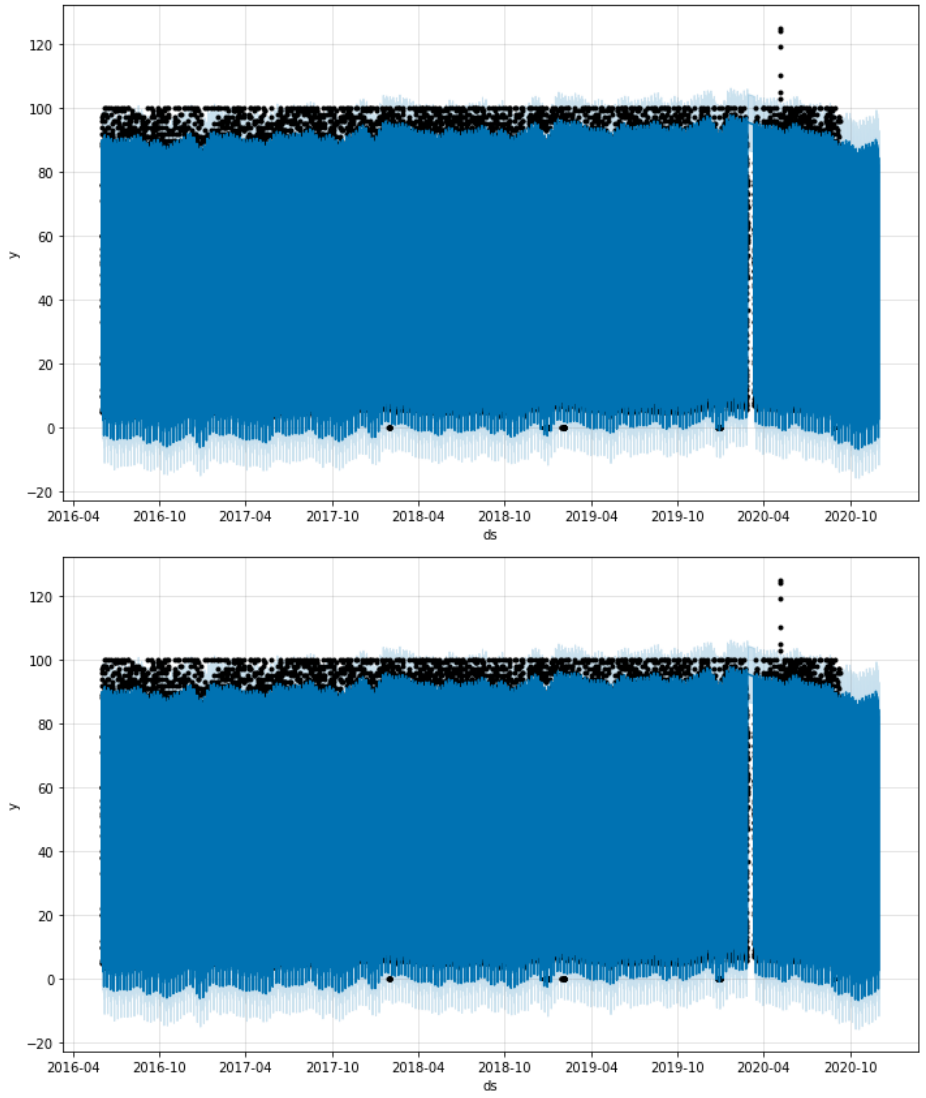
```
# Make the predictions for the trend data using the future_mercado_trends DataFrame
forecast_mercado_trends = model_mercado_trends.predict(future_mercado_trends)

# Display the first five rows of the forecast_mercado_trends DataFrame
forecast_mercado_trends.head()
```

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_terms	additive_terms_lower	additive_terms_upp
0	2016-06-01 00:00:00	44.390875	81.636808	98.582418	44.390875	44.390875	45.227910	45.227910	45.2279
1	2016-06-01 01:00:00	44.391810	77.680511	94.534398	44.391810	44.391810	41.673716	41.673716	41.6737
2	2016-06-01 02:00:00	44.392746	66.927913	84.713148	44.392746	44.392746	31.350247	31.350247	31.3502
3	2016-06-01 03:00:00	44.393681	52.134330	69.043691	44.393681	44.393681	16.083050	16.083050	16.0830
4	2016-06-01 04:00:00	44.394617	34.772767	51.938176	44.394617	44.394617	-1.031813	-1.031813	-1.0318

5 rows × 22 columns

- Step 2: After estimating the model, plot the forecast. How's the near-term forecast for the popularity of MercadoLibre?

```
# Plot the Prophet predictions for the Mercado trends data
model_mercado_trends.plot(forecast_mercado_trends)
```



- Answer the following question:

Question: How's the near-term forecast for the popularity of MercadoLibre?

Answer: The near-term forecast for the popularity of MercadoLibre is (85/90-1 = 5.6%)

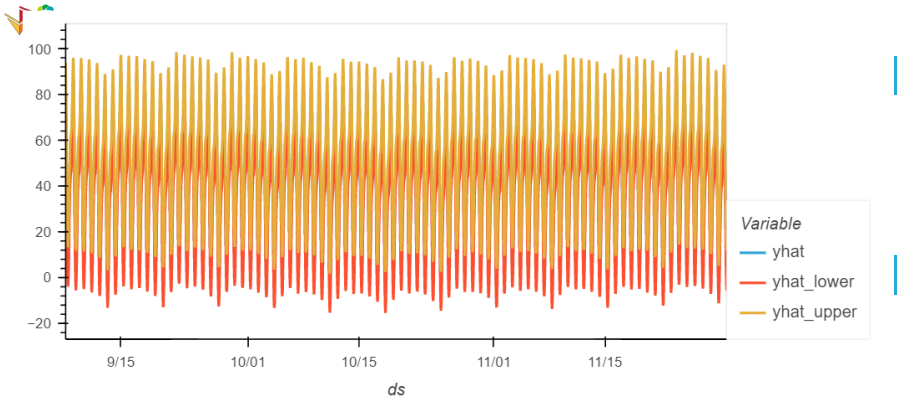
- Step 3: Plot the individual time series components of the model to answer the following questions:
  - What time of day exhibits the greatest popularity?
  - Which day of the week gets the most search traffic?
  - What's the lowest point for search traffic in the calendar year?

```
# Set the index in the forecast_mercado_trends DataFrame to the ds datetime column
forecast_mercado_trends = forecast_mercado_trends.set_index('ds')

# View the only the yhat,yhat_lower and yhat_upper columns from the DataFrame
forecast_mercado_trends[['yhat','yhat_lower','yhat_upper']].head()
```

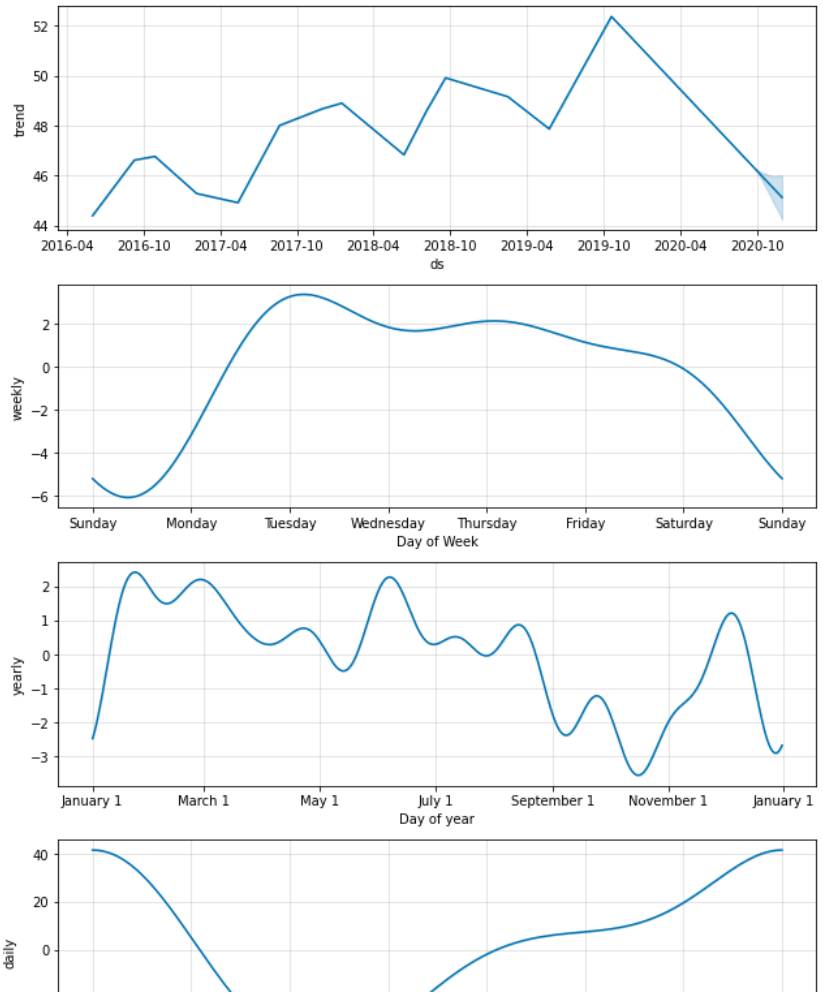
ds	yhat	yhat_lower	yhat_upper
2016-06-01 00:00:00	89.618785	81.636808	98.582418
2016-06-01 01:00:00	86.065527	77.680511	94.534398
2016-06-01 02:00:00	75.742993	66.927913	84.713148
2016-06-01 03:00:00	60.476731	52.134330	69.043691
2016-06-01 04:00:00	43.362804	34.772767	51.938176

Solutions Note: `yhat` represents the most likely (average) forecast, whereas `yhat_lower` and `yhat_upper` represents the worst and best case prediction (based on what are known as 95% confidence intervals).

```
# Holoviews extension to render hvPlots in Colab
hv.extension('bokeh')

# From the forecast_mercado_trends DataFrame, use hvPlot to visualize
#  the yhat, yhat_lower, and yhat_upper columns over the last 2000 hours
forecast_mercado_trends[['yhat','yhat_lower','yhat_upper']].iloc[-2000:,:].hvplot()
```




```
# Reset the index in the forecast_mercado_trends DataFrame
forecast_mercado_trends = forecast_mercado_trends.reset_index()

# Use the plot_components function to visualize the forecast results
# for the forecast_mercado_trends DataFrame
figures_mercado_trends = model_mercado_trends.plot_components(forecast_mercado_trends)
```



▾ Answer the following questions:

Question: What time of day exhibits the greatest popularity?

Answer: The time of day that exhibits the greatest popularity is from 21:00 to 24:00.

Question: Which day of week gets the most search traffic?

Answer: Tuesday gets the most search traffic.

Question: What's the lowest point for search traffic in the calendar year?

Answer: The lowest point for search traffic in the calendar year occurs the last week of the year, the first week of the year and the middle of October.

▾ Step 5 (Optional): Forecast Revenue by Using Time Series Models

A few weeks after your initial analysis, the finance group follows up to find out if you can help them solve a different problem. Your fame as a growth analyst in the company continues to grow!

Specifically, the finance group wants a forecast of the total sales for the next quarter. This will dramatically increase their ability to plan budgets and to help guide expectations for the company investors.

To do so, complete the following steps:

1. Read in the daily historical sales (that is, revenue) figures, and then apply a Prophet model to the data. The daily sales figures are quoted in millions of USD dollars.
2. Interpret the model output to identify any seasonal patterns in the company's revenue. For example, what are the peak revenue days? (Mondays? Fridays? Something else?)
3. Produce a sales forecast for the finance group. Give them a number for the expected total sales in the next quarter. Include the best- and worst-case scenarios to help them make better plans.

Step 1: Read in the daily historical sales (that is, revenue) figures, and then apply a Prophet model to the data.

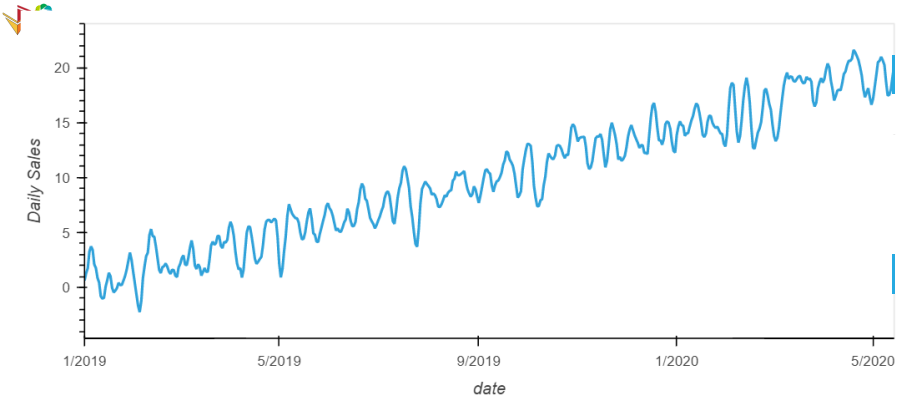
```
# Upload the "mercado_daily_revenue.csv" file into Colab, then store in a Pandas DataFrame
# Set the "date" column as the DatetimeIndex
# Sales are quoted in millions of US dollars
from google.colab import files
uploaded = files.upload()

df_mercado_sales = pd.read_csv(
    "mercado_daily_revenue.csv",
    index_col='date',
    parse_dates=True,
    infer_datetime_format=True
)
# Review the DataFrame
display(df_mercado_sales.head())
display(df_mercado_sales.tail())
```

```
Choose Files mercado_da...evenue.csv
• mercado_daily_revenue.csv(text/csv) - 14781 bytes, last modified: 9/17/2022 - 100% done
Saving mercado_daily_revenue.csv to mercado_daily_revenue.csv
```

date	Daily Sales
2019-01-01	0.626452
2019-01-02	1.301069
2019-01-03	1.751689
2019-01-04	3.256294
2019-01-05	3.732920

date	Daily Sales
2020-05-10	17.467814
2020-05-11	17.537152
2020-05-12	18.031773
2020-05-13	19.165315
2020-05-14	20.246570

```
# Holoviews extension to render hvPlots in Colab
hv.extension('bokeh')

# Use hvPlot to visualize the daily sales figures
df_mercado_sales.hvplot()
```




```
# Apply a Facebook Prophet model to the data.

# Set up the dataframe in the neccessary format:
```

```
# Reset the index so that date becomes a column in the DataFrame
mercado_sales_prophet_df = df_mercado_sales.reset_index()

# Adjust the columns names to the Prophet syntax
mercado_sales_prophet_df.columns = ['ds','y']

# Visualize the DataFrame
mercado_sales_prophet_df.head()
```

	ds	y
0	2019-01-01	0.626452
1	2019-01-02	1.301069
2	2019-01-03	1.751689
3	2019-01-04	3.256294
4	2019-01-05	3.732920

```
# Create the model
mercado_sales_prophet_model = Prophet(yearly_seasonality=True,daily_seasonality=True)

# Fit the model
mercado_sales_prophet_model.fit(mercado_sales_prophet_df)
```

```
<prophet.forecaster.Prophet at 0x7fad3f2e55d0>
```

```
# Predict sales for 90 days (1 quarter) out into the future.

# Start by making a future dataframe
mercado_sales_prophet_future = mercado_sales_prophet_model.make_future_dataframe(periods=90,freq="D")

# Display the last five rows of the future DataFrame
mercado_sales_prophet_future.tail()
```

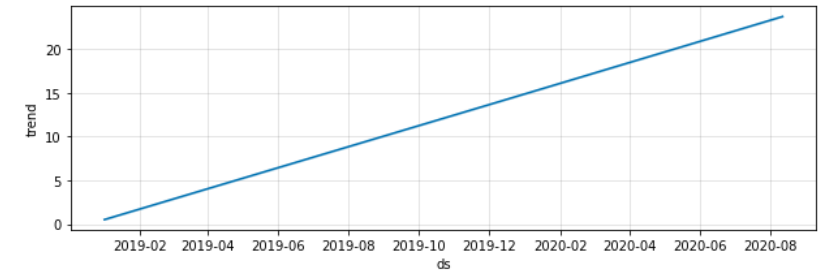
	ds
585	2020-08-08
586	2020-08-09
587	2020-08-10
588	2020-08-11
589	2020-08-12

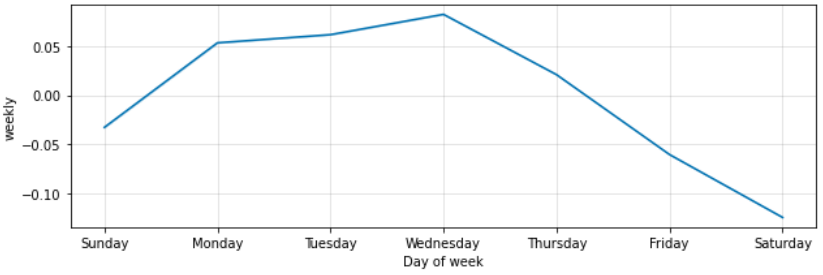
```
# Make predictions for the sales each day over the next quarter
mercado_sales_prophet_forecast = mercado_sales_prophet_model.predict(mercado_sales_prophet_future)

# Display the first 5 rows of the resulting DataFrame
mercado_sales_prophet_future.head()
```

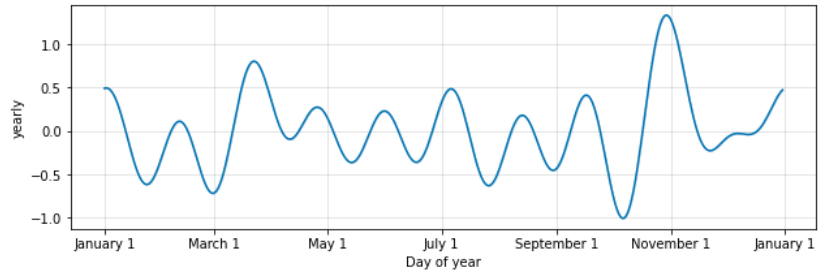
	ds
0	2019-01-01
1	2019-01-02
2	2019-01-03
3	2019-01-04
4	2019-01-05

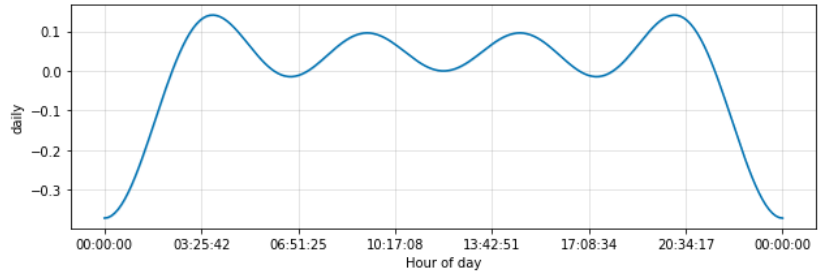
Step 2: Interpret the model output to identify any seasonal patterns in the company's revenue. For example, what are the peak revenue days? (Mondays? Fridays? Something else?)

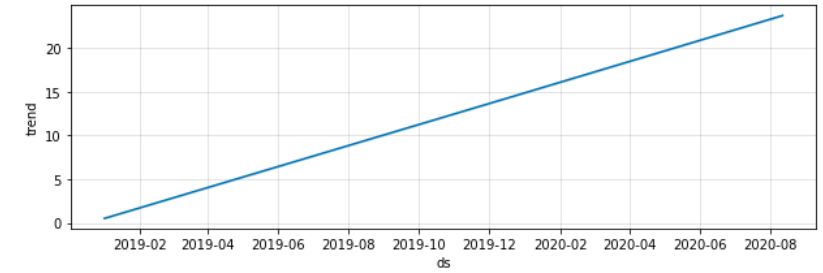
```
# Use the plot_components function to analyze seasonal patterns in the company's revenue
mercado_sales_prophet_model.plot_components(mercado_sales_prophet_forecast)
```

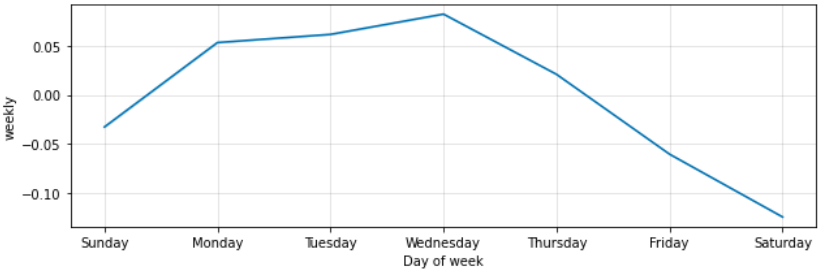

trend
20
15
10
5
0
2019-02
2019-04
2019-06
2019-08
2019-10
2019-12
2020-02
2020-04
2020-06
2020-08
ds

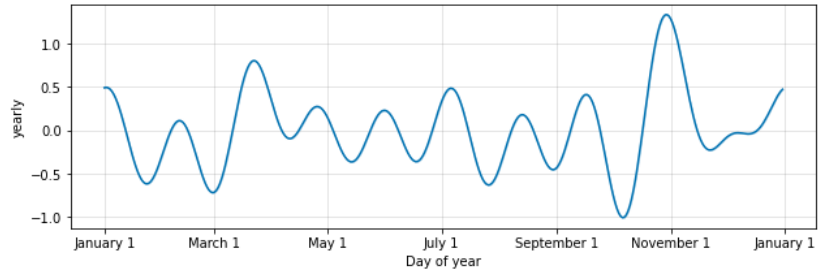

weekly
0.05
0.00
−0.05
−0.10
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Day of week

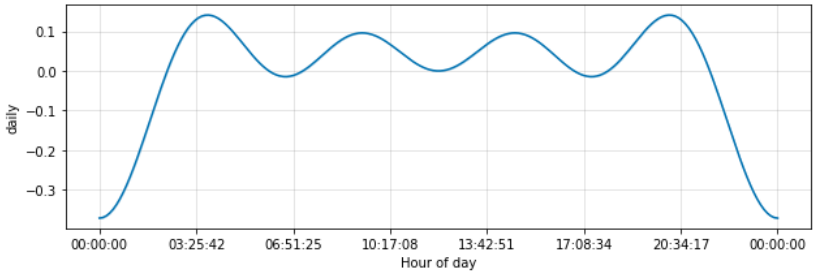

yearly
1.0
0.5
0.0
−0.5
−1.0
January 1
March 1
May 1
July 1
September 1
November 1
January 1
Day of year


daily
0.1
0.0
−0.1
−0.2
−0.3
00:00:00
03:25:42
06:51:25
10:17:08
13:42:51
17:08:34
20:34:17
00:00:00
Hour of day


trend
20
15
10
5
0
2019-02
2019-04
2019-06
2019-08
2019-10
2019-12
2020-02
2020-04
2020-06
2020-08
ds


weekly
0.05
0.00
−0.05
−0.10
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Day of week


yearly
1.0
0.5
0.0
−0.5
−1.0
January 1
March 1
May 1
July 1
September 1
November 1
January 1
Day of year



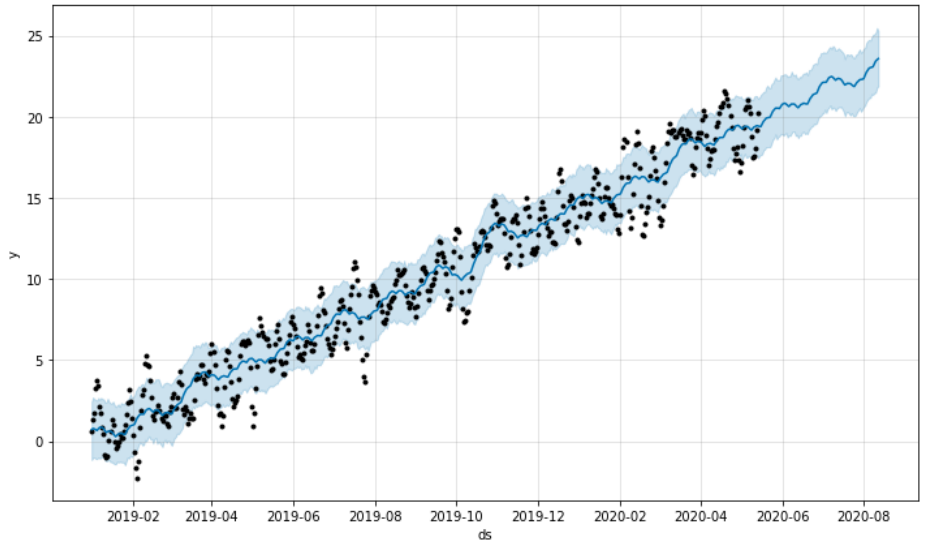
Answer the following question:

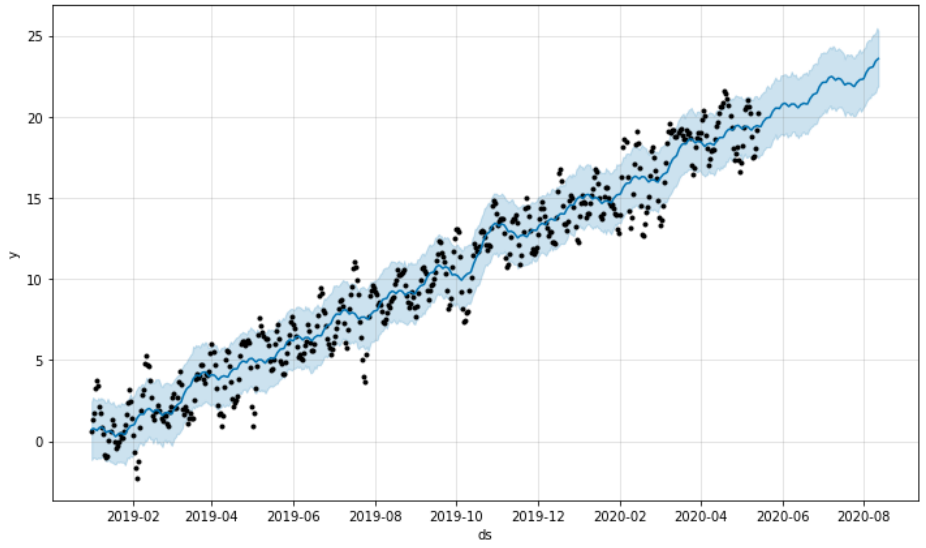
Question: For example, what are the peak revenue days? (Mondays? Fridays? Something else?)

Answer: The peak revenue days are Thursday and Friday.

Step 3: Produce a sales forecast for the finance group. Give them a number for the expected total sales in the next quarter. Include the best- and worst-case scenarios to help them make better plans.

```
# Plot the predictions for the Mercado sales
mercado_sales_prophet_model.plot(mercado_sales_prophet_forecast)
```






```
# For the mercado_sales_prophet_forecast DataFrame, set the ds column as the DataFrame Index
mercado_sales_prophet_forecast = mercado_sales_prophet_forecast.set_index('ds')
```

```
# Display the first and last five rows of the DataFrame
display(mercado_sales_prophet_forecast.head())
display(mercado_sales_prophet_forecast.tail())
```

ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_terms	additive_terms_lower	additive_terms_upper	
2019-01-01	0.533530	-1.157762	2.432782	0.533530	0.533530	0.177688	0.177688	0.177688	-0.
2019-01-02	0.572718	-1.030342	2.702659	0.572718	0.572718	0.207495	0.207495	0.207495	-0.
2019-01-03	0.611906	-1.001672	2.587905	0.611906	0.611906	0.146126	0.146126	0.146126	-0.
2019-01-04	0.651093	-1.081989	2.565389	0.651093	0.651093	0.055034	0.055034	0.055034	-0.
2019-01-05	0.690281	-1.095232	2.503922	0.690281	0.690281	-0.028828	-0.028828	-0.028828	-0.

5 rows × 21 columns

ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_terms	additive_terms_lower	additive_terms_upper	
2020-08-08	23.531399	21.341165	25.050956	23.529763	23.532916	-0.454351	-0.454351	-0.454351	-0
2020-08-09	23.570828	21.452205	25.084459	23.569169	23.572370	-0.314080	-0.314080	-0.314080	-0
2020-08-10	23.610258	21.478445	25.188372	23.608572	23.611822	-0.188816	-0.188816	-0.188816	-0
2020-08-11	23.649687	21.781006	25.473716	23.647975	23.651273	-0.151652	-0.151652	-0.151652	-0
2020-08-12	23.689117	21.880587	25.277804	23.687379	23.690729	-0.113345	-0.113345	-0.113345	-0

5 rows × 21 columns

```
# Produce a sales forecast for the finance division
# giving them a number for expected total sales next quarter.
# Provide best case (yhat_upper), worst case (yhat_lower), and most likely (yhat) scenarios.

# Create a forecast_quarter Dataframe for the period 2020-07-01 to 2020-09-30
# The DataFrame should include the columns yhat_upper, yhat_lower, and yhat
mercado_sales_forecast_quarter = mercado_sales_prophet_forecast['2020-07-01':'2020-09-30'][['yhat_upper','yhat_lower','yhat']]


# Update the column names for the forecast_quarter DataFrame
# to match what the finance division is looking for
mercado_sales_forecast_quarter = mercado_sales_forecast_quarter.rename(columns = {'yhat_upper':'Best Case','yhat_lower':'Worst Case','yhat':'Most

# Review the last five rows of the DataFrame
mercado_sales_forecast_quarter.tail()
```

ds	Best Case	Worst Case	Most Likely Case
2020-08-08	25.050956	21.341165	23.077048
2020-08-09	25.084459	21.452205	23.256749
2020-08-10	25.188372	21.478445	23.421442
2020-08-11	25.473716	21.781006	23.498035
2020-08-12	25.277804	21.880587	23.575772

```
# Displayed the summed values for all the rows in the forecast_quarter DataFrame
mercado_sales_forecast_quarter.sum()
```

```
Best Case           1041.531886
Worst Case           885.707484
Most Likely Case     963.917744
dtype: float64
```

Double-click (or enter) to edit