forecasting_net_prophet_vscode

May 10, 2022

1 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.

1.1 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?

1.2 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)?

1.3 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?

1.4 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?

1.5 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.

1.6 Install and import the required libraries and dependencies

[ ]:
```
''' # Install the required libraries
from IPython.display import clear_output
try:
  !pip install pystan
  !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
from fbprophet import Prophet
import hvplot.pandas
import datetime as dt
%matplotlib inline

# for importing from within the same system if not using google colab
from pathlib import Path
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
/Users/jalhussain/ASU_Fintech/07_TimeSeries/TimeSeries-Analysis-MercadoLibre/
 ↪forecasting_net_prophet_vscode.ipynb Cell 4' in <cell line: 4>()
      <a href='vscode-notebook-cell:/Users/jalhussain/ASU_Fintech/07_TimeSeries/
 ↪TimeSeries-Analysis-MercadoLibre/forecasting_net_prophet_vscode.
 ↪ipynb#ch0000003?line=1'>2</a> import pandas as pd
      <a href='vscode-notebook-cell:/Users/jalhussain/ASU_Fintech/07_TimeSeries/
 ↪TimeSeries-Analysis-MercadoLibre/forecasting_net_prophet_vscode.
 ↪ipynb#ch0000003?line=2'>3</a> import holoviews as hv
----> <a href='vscode-notebook-cell:/Users/jalhussain/ASU_Fintech/07_TimeSeries/
 ↪TimeSeries-Analysis-MercadoLibre/forecasting_net_prophet_vscode.
 ↪ipynb#ch0000003?line=3'>4</a> from fbprophet import Prophet
      <a href='vscode-notebook-cell:/Users/jalhussain/ASU_Fintech/07_TimeSeries/
 ↪TimeSeries-Analysis-MercadoLibre/forecasting_net_prophet_vscode.
 ↪ipynb#ch0000003?line=4'>5</a> import hvplot.pandas
      <a href='vscode-notebook-cell:/Users/jalhussain/ASU_Fintech/07_TimeSeries/
 ↪TimeSeries-Analysis-MercadoLibre/forecasting_net_prophet_vscode.
 ↪ipynb#ch0000003?line=5'>6</a> import datetime as dt

ModuleNotFoundError: No module named 'fbprophet'
```

1.7 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('../Resources/google_hourly_search_trends.csv')



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

```
<IPython.core.display.HTML object>

Saving google_hourly_search_trends.csv to google_hourly_search_trends.csv
          Date  Search Trends
0  6/1/16 0:00             97
1  6/1/16 1:00             92
2  6/1/16 2:00             76
3  6/1/16 3:00             60
4  6/1/16 4:00             38
               Date  Search Trends
37101  9/7/20 20:00             71
37102  9/7/20 21:00             83
37103  9/7/20 22:00             96
37104  9/7/20 23:00             97
37105   9/8/20 0:00             96
```

```
[ ]: # Review the data types of the DataFrame using the info function
     df_mercado_trends['Date'] = pd.to_datetime(df_mercado_trends['Date'] ,
     ↪infer_datetime_format=True , utc=True, )
     df_mercado_trends.set_index(df_mercado_trends['Date'] , inplace=True)
     print(df_mercado_trends.info(),
     df_mercado_trends.tail())
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 37106 entries, 2016-06-01 00:00:00+00:00 to 2020-09-08
00:00:00+00:00
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Date           37106 non-null  datetime64[ns, UTC]
 1   Search Trends  37106 non-null  int64
```

```
dtypes: datetime64[ns, UTC](1), int64(1)
memory usage: 869.7 KB
None                                                Date  Search Trends
Date
2020-09-07 20:00:00+00:00 2020-09-07 20:00:00+00:00             71
2020-09-07 21:00:00+00:00 2020-09-07 21:00:00+00:00             83
2020-09-07 22:00:00+00:00 2020-09-07 22:00:00+00:00             96
2020-09-07 23:00:00+00:00 2020-09-07 23:00:00+00:00             97
2020-09-08 00:00:00+00:00 2020-09-08 00:00:00+00:00             96
```

[ ]:
```
# 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["2020-05-01":"2020-05-30"]
print(df_may_2020.head())

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

```
                                               Date  Search Trends
Date
2020-05-01 00:00:00+00:00 2020-05-01 00:00:00+00:00             80
2020-05-01 01:00:00+00:00 2020-05-01 01:00:00+00:00             80
2020-05-01 02:00:00+00:00 2020-05-01 02:00:00+00:00             76
2020-05-01 03:00:00+00:00 2020-05-01 03:00:00+00:00             66
2020-05-01 04:00:00+00:00 2020-05-01 04:00:00+00:00             53
```

[ ]:
```
:Curve   [Date]   (Search Trends)
```

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['Search Trends'].sum()

# View the traffic_may_2020 value
traffic_may_2020
```

[ ]:
```
37014
```

[ ]:
```
# 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().mean()

# View the median_monthly_traffic value
```

```
median_monthly_traffic
```

[ ]:
```
Search Trends    34343.557692
dtype: float64
```

[ ]:
```
# Compare the seach traffic for the month of May 2020 to the overall monthly␣
↪median value
compare_search_pct = ((traffic_may_2020 - median_monthly_traffic)/
↪median_monthly_traffic )*100
compare_search_pct

search_diff = traffic_may_2020 - median_monthly_traffic
print(f" The Percent change of search trend is {compare_search_pct} and the␣
↪difference in search was {search_diff} for May 2020 compared to the mdeian ")
```

```
 The Percent change of search trend is Search Trends    7.775672
dtype: float64 and the difference in search was Search Trends    2670.442308
dtype: float64 for May 2020 compared to the mdeian
```

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

Answer: Based on the numbers calculated there was a 7.78% increase in searches for May 2020 and a difference of 2670 searchs compared to the median

1.8 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(by=df_mercado_trends.index.day_of_week).mean().
↪hvplot()
```

[ ]: :Curve [Date] (Search Trends)

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.day_of_week',
    C='Search Trends'
)
```

```
WARNING:param.HeatMapPlot02001: HeatMap element index is not unique,  ensure you
aggregate the data before displaying it, e.g. using
heatmap.aggregate(function=np.mean). Duplicate index values have been dropped.
```

[ ]: :HeatMap [index.hour,index.day_of_week] (Search Trends)

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

Answer: There is no clear distinction for any day of the week however only start and end of the day do both show more activity. depending on the time this was collected it could be an international effect perhaps but more studying is required to be sure.

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(by=df_mercado_trends.index.weekofyear).mean().hvplot()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: FutureWarning:
weekofyear and week have been deprecated, please use
DatetimeIndex.isocalendar().week instead, which returns a Series.  To exactly
reproduce the behavior of week and weekofyear and return an Index, you may call
```

```
pd.Int64Index(idx.isocalendar().week)
  """
```

[ ]:
```
:Curve   [Date]   (Search Trends)
```

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

Answer: we can observe a slight upward trend from week 40 onward

1.9 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.

[ ]:
```python
# 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
uploaded = files.upload()

df_mercado_stock = pd.read_csv('mercado_stock_price.csv')
```

```
df_mercado_stock['date'] = pd.to_datetime(df_mercado_stock['date'] ,␣
↪infer_datetime_format=True , utc=True, )
df_mercado_stock.set_index(df_mercado_stock['date'] , inplace=True)
print(df_mercado_stock.info())

# View the first and last five rows of the DataFrame
print(df_mercado_stock.head(5),df_mercado_stock.tail(5))
```

```
<IPython.core.display.HTML object>

Saving mercado_stock_price.csv to mercado_stock_price.csv
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 48895 entries, 2015-01-02 09:00:00+00:00 to 2020-07-31
15:00:00+00:00
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   date    48895 non-null  datetime64[ns, UTC]
 1   close   9336 non-null   float64
dtypes: datetime64[ns, UTC](1), float64(1)
memory usage: 1.1 MB
None
                                               date   close
date
2015-01-02 09:00:00+00:00 2015-01-02 09:00:00+00:00  127.67
2015-01-02 10:00:00+00:00 2015-01-02 10:00:00+00:00  125.44
2015-01-02 11:00:00+00:00 2015-01-02 11:00:00+00:00  125.57
2015-01-02 12:00:00+00:00 2015-01-02 12:00:00+00:00  125.40
2015-01-02 13:00:00+00:00 2015-01-02 13:00:00+00:00  125.17
date     close
date
2020-07-31 11:00:00+00:00 2020-07-31 11:00:00+00:00  1105.780
2020-07-31 12:00:00+00:00 2020-07-31 12:00:00+00:00  1087.925
2020-07-31 13:00:00+00:00 2020-07-31 13:00:00+00:00  1095.800
2020-07-31 14:00:00+00:00 2020-07-31 14:00:00+00:00  1110.650
2020-07-31 15:00:00+00:00 2020-07-31 15:00:00+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.hvplot()
```

[ ]: :Curve   [date]   (close)

[ ]:
```
# 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_trends,df_mercado_stock],␣
↪axis=1).dropna()
#drop the extra date col
mercado_stock_trends_df.drop(columns='date' ,inplace=True)
# View the first and last five rows of the DataFrame
mercado_stock_trends_df
```

[ ]:

```
                                         Date  Search Trends     close
2016-06-01 09:00:00+00:00 2016-06-01 09:00:00+00:00            6.0   135.160
2016-06-01 10:00:00+00:00 2016-06-01 10:00:00+00:00           12.0   136.630
2016-06-01 11:00:00+00:00 2016-06-01 11:00:00+00:00           22.0   136.560
2016-06-01 12:00:00+00:00 2016-06-01 12:00:00+00:00           33.0   136.420
2016-06-01 13:00:00+00:00 2016-06-01 13:00:00+00:00           40.0   136.100
…                                               …             …         …
2020-07-31 11:00:00+00:00 2020-07-31 11:00:00+00:00           20.0  1105.780
2020-07-31 12:00:00+00:00 2020-07-31 12:00:00+00:00           32.0  1087.925
2020-07-31 13:00:00+00:00 2020-07-31 13:00:00+00:00           41.0  1095.800
2020-07-31 14:00:00+00:00 2020-07-31 14:00:00+00:00           47.0  1110.650
2020-07-31 15:00:00+00:00 2020-07-31 15:00:00+00:00           53.0  1122.510

[7067 rows x 3 columns]
```

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.loc['2020-1':'2020-06']
# View the first and last five rows of first_half_2020 DataFrame
print(
    first_half_2020.head(),
    first_half_2020.tail()
)
```

```
                                         Date  Search Trends     close
2020-01-02 09:00:00+00:00 2020-01-02 09:00:00+00:00            9.0   601.085
2020-01-02 10:00:00+00:00 2020-01-02 10:00:00+00:00           14.0   601.290
2020-01-02 11:00:00+00:00 2020-01-02 11:00:00+00:00           25.0   615.410
2020-01-02 12:00:00+00:00 2020-01-02 12:00:00+00:00           37.0   611.400
2020-01-02 13:00:00+00:00 2020-01-02 13:00:00+00:00           50.0   611.830
Date  Search Trends   close
2020-06-30 11:00:00+00:00 2020-06-30 11:00:00+00:00           17.0   976.17
2020-06-30 12:00:00+00:00 2020-06-30 12:00:00+00:00           27.0   977.50
```

```
2020-06-30 13:00:00+00:00 2020-06-30 13:00:00+00:00           37.0  973.23
2020-06-30 14:00:00+00:00 2020-06-30 14:00:00+00:00           45.0  976.50
2020-06-30 15:00:00+00:00 2020-06-30 15:00:00+00:00           51.0  984.93
```

[ ]:
```
# 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)
```

[ ]:
```
:NdLayout   [Variable]
   :Curve   [index]   (value)
```

Answer the following question: **Question:** Do both time series indicate a common trend that's consistent with this narrative?

Answer: Only Stock Prices show an upward trend which agrees with the narrtive porbably but search trends dont seem to be effecte

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'].
↪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.hvplot(
    x='Date',
```

```
    y='Stock Volatility'
)
```

[ ]:
```
:Curve   [Date]   (Stock Volatility)
```

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()
mercado_stock_trends_df
```

[ ]:
```
                                               Date  Search Trends     close  \
2016-06-01 09:00:00+00:00 2016-06-01 09:00:00+00:00            6.0   135.160
2016-06-01 10:00:00+00:00 2016-06-01 10:00:00+00:00           12.0   136.630
2016-06-01 11:00:00+00:00 2016-06-01 11:00:00+00:00           22.0   136.560
2016-06-01 12:00:00+00:00 2016-06-01 12:00:00+00:00           33.0   136.420
2016-06-01 13:00:00+00:00 2016-06-01 13:00:00+00:00           40.0   136.100
…                                                 …            …         …
2020-07-31 11:00:00+00:00 2020-07-31 11:00:00+00:00           20.0  1105.780
2020-07-31 12:00:00+00:00 2020-07-31 12:00:00+00:00           32.0  1087.925
2020-07-31 13:00:00+00:00 2020-07-31 13:00:00+00:00           41.0  1095.800
2020-07-31 14:00:00+00:00 2020-07-31 14:00:00+00:00           47.0  1110.650
2020-07-31 15:00:00+00:00 2020-07-31 15:00:00+00:00           53.0  1122.510

                           Lagged Search Trends  Stock Volatility  \
2016-06-01 09:00:00+00:00                   NaN               NaN
2016-06-01 10:00:00+00:00                   6.0               NaN
2016-06-01 11:00:00+00:00                  12.0               NaN
2016-06-01 12:00:00+00:00                  22.0          0.693848
2016-06-01 13:00:00+00:00                  33.0          0.235142
…                                           …               …
2020-07-31 11:00:00+00:00                  11.0          7.495900
2020-07-31 12:00:00+00:00                  20.0         12.188462
2020-07-31 13:00:00+00:00                  32.0          7.393646
2020-07-31 14:00:00+00:00                  41.0         10.169735
2020-07-31 15:00:00+00:00                  47.0         15.408790

                           Hourly Stock Return
2016-06-01 09:00:00+00:00                  NaN
2016-06-01 10:00:00+00:00             0.010876
2016-06-01 11:00:00+00:00            -0.000512
2016-06-01 12:00:00+00:00            -0.001025
2016-06-01 13:00:00+00:00            -0.002346
```

```
…                                               …
2020-07-31 11:00:00+00:00            0.006380
2020-07-31 12:00:00+00:00           -0.016147
2020-07-31 13:00:00+00:00            0.007239
2020-07-31 14:00:00+00:00            0.013552
2020-07-31 15:00:00+00:00            0.010678

[7067 rows x 6 columns]
```

[ ]:
```
# View the first and last five rows of the mercado_stock_trends_df DataFrame
print(
    mercado_stock_trends_df.head(5),
    mercado_stock_trends_df.tail(5)
)
```

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

                           Lagged Search Trends  Stock Volatility  \
2016-06-01 09:00:00+00:00                   NaN               NaN
2016-06-01 10:00:00+00:00                   6.0               NaN
2016-06-01 11:00:00+00:00                  12.0               NaN
2016-06-01 12:00:00+00:00                  22.0          0.693848
2016-06-01 13:00:00+00:00                  33.0          0.235142

                           Hourly Stock Return
2016-06-01 09:00:00+00:00                  NaN
2016-06-01 10:00:00+00:00             0.010876
2016-06-01 11:00:00+00:00            -0.000512
2016-06-01 12:00:00+00:00            -0.001025
2016-06-01 13:00:00+00:00            -0.002346
Date  Search Trends     close  \
2020-07-31 11:00:00+00:00 2020-07-31 11:00:00+00:00           20.0  1105.780
2020-07-31 12:00:00+00:00 2020-07-31 12:00:00+00:00           32.0  1087.925
2020-07-31 13:00:00+00:00 2020-07-31 13:00:00+00:00           41.0  1095.800
2020-07-31 14:00:00+00:00 2020-07-31 14:00:00+00:00           47.0  1110.650
2020-07-31 15:00:00+00:00 2020-07-31 15:00:00+00:00           53.0  1122.510

                           Lagged Search Trends  Stock Volatility  \
2020-07-31 11:00:00+00:00                  11.0          7.495900
2020-07-31 12:00:00+00:00                  20.0         12.188462
2020-07-31 13:00:00+00:00                  32.0          7.393646
2020-07-31 14:00:00+00:00                  41.0         10.169735
2020-07-31 15:00:00+00:00                  47.0         15.408790
```

```
                           Hourly Stock Return
2020-07-31 11:00:00+00:00             0.006380
2020-07-31 12:00:00+00:00            -0.016147
2020-07-31 13:00:00+00:00             0.007239
2020-07-31 14:00:00+00:00             0.013552
2020-07-31 15:00:00+00:00             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  \
Stock Volatility              1.000000             -0.118945
Lagged Search Trends         -0.118945              1.000000
Hourly Stock Return           0.046723              0.017929

                      Hourly Stock Return
Stock Volatility                 0.046723
Lagged Search Trends             0.017929
Hourly Stock Return              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 is a negative correlation

1.10 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.

[ ]:
```
#dropping the extra date col from the DF
df_mercado_trends.drop(columns='Date' ,inplace=True)
```

[ ]:
```
import fbprophet

# 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()

#format the date for prophet input
mercado_prophet_df['ds']= pd.
↪to_datetime(mercado_prophet_df['ds'],format='%Y%m%d')
mercado_prophet_df['ds']=mercado_prophet_df['ds'].dt.tz_localize(None)
# View the first and last five rows of the mercado_prophet_df DataFrame
mercado_prophet_df
```

[ ]:

	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
…	…	..
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

[37106 rows x 2 columns]

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

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

[ ]: <fbprophet.forecaster.Prophet at 0x7fa413a68750>

[ ]:
```
# 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(5)
```

[ ]:
```
                       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
print(
    forecast_mercado_trends.head(),
    forecast_mercado_trends.tail()
)
```

```
                   ds      trend  yhat_lower  yhat_upper  trend_lower  \
0 2016-06-01 00:00:00  44.437254   81.469840   98.037735    44.437254
1 2016-06-01 01:00:00  44.438181   77.851726   94.892330    44.438181
2 2016-06-01 02:00:00  44.439108   67.843181   84.180436    44.439108
3 2016-06-01 03:00:00  44.440034   52.702161   68.789860    44.440034
4 2016-06-01 04:00:00  44.440961   35.021499   51.753917    44.440961

   trend_upper  additive_terms  additive_terms_lower  additive_terms_upper  \
0    44.437254       45.198724             45.198724             45.198724
1    44.438181       41.644504             41.644504             41.644504
2    44.439108       31.321010             31.321010             31.321010
3    44.440034       16.053788             16.053788             16.053788
4    44.440961       -1.061098             -1.061098             -1.061098

       daily  …    weekly  weekly_lower  weekly_upper    yearly  \
0  41.452626  …  1.860528      1.860528      1.860528  1.885569
1  37.943462  …  1.810432      1.810432      1.810432  1.890611
2  27.656545  …  1.768844      1.768844      1.768844  1.895621
3  12.417331  …  1.735858      1.735858      1.735858  1.900600
4  -4.678073  …  1.711428      1.711428      1.711428  1.905547

   yearly_lower  yearly_upper  multiplicative_terms  \
0      1.885569      1.885569                   0.0
```

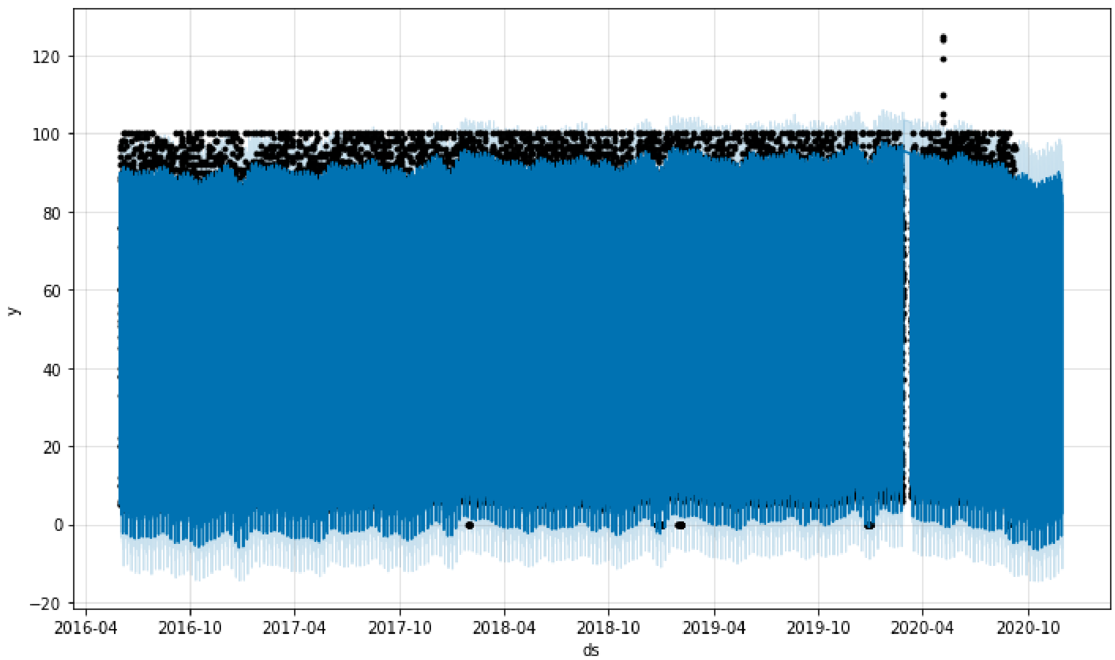
```
1      1.890611      1.890611                   0.0
2      1.895621      1.895621                   0.0
3      1.900600      1.900600                   0.0
4      1.905547      1.905547                   0.0

   multiplicative_terms_lower  multiplicative_terms_upper       yhat
0                         0.0                         0.0  89.635978
1                         0.0                         0.0  86.082685
2                         0.0                         0.0  75.760118
3                         0.0                         0.0  60.493823
4                         0.0                         0.0  43.379863

[5 rows x 22 columns]                        ds      trend  yhat_lower
yhat_upper  trend_lower  \
39101 2020-11-30 04:00:00  45.120891   31.443821   48.661232    44.251615
39102 2020-11-30 05:00:00  45.120148   15.986895   32.260925    44.249976
39103 2020-11-30 06:00:00  45.119405    3.642285   20.787234    44.248338
39104 2020-11-30 07:00:00  45.118662   -3.564709   13.621496    44.246699
39105 2020-11-30 08:00:00  45.117920   -5.794846   10.830538    44.245061

       trend_upper  additive_terms  additive_terms_lower  \
39101    46.106745       -5.390095             -5.390095
39102    46.106086      -20.860475            -20.860475
39103    46.105427      -32.825391            -32.825391
39104    46.104768      -40.096790            -40.096790
39105    46.105373      -42.290922            -42.290922

       additive_terms_upper      daily  …    weekly  weekly_lower  \
39101             -5.390095  -4.678073  … -1.746847     -1.746847
39102            -20.860475 -20.514514  … -1.384964     -1.384964
39103            -32.825391 -32.844594  … -1.023942     -1.023942
39104            -40.096790 -40.477998  … -0.666042     -0.666042
39105            -42.290922 -43.028770  … -0.313470     -0.313470

       weekly_upper    yearly  yearly_lower  yearly_upper  \
39101     -1.746847  1.034825      1.034825      1.034825
39102     -1.384964  1.039003      1.039003      1.039003
39103     -1.023942  1.043146      1.043146      1.043146
39104     -0.666042  1.047250      1.047250      1.047250
39105     -0.313470  1.051318      1.051318      1.051318

       multiplicative_terms  multiplicative_terms_lower  \
39101                   0.0                         0.0
39102                   0.0                         0.0
39103                   0.0                         0.0
39104                   0.0                         0.0
39105                   0.0                         0.0
```

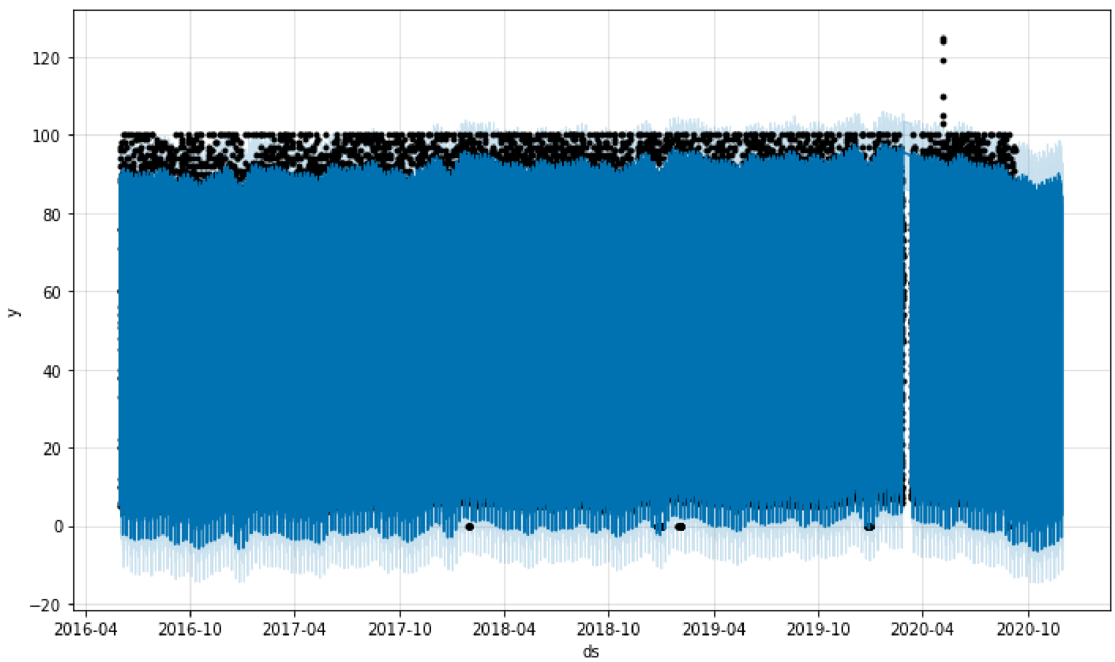
```
       multiplicative_terms_upper       yhat
39101                         0.0  39.730796
39102                         0.0  24.259673
39103                         0.0  12.294015
39104                         0.0   5.021873
39105                         0.0   2.826998

[5 rows x 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: # YOUR ANSWER HERE

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']]
```

[ ]:
```
                          yhat  yhat_lower  yhat_upper
ds
2016-06-01 00:00:00  89.635978   81.469840   98.037735
2016-06-01 01:00:00  86.082685   77.851726   94.892330
2016-06-01 02:00:00  75.760118   67.843181   84.180436
2016-06-01 03:00:00  60.493823   52.702161   68.789860
```

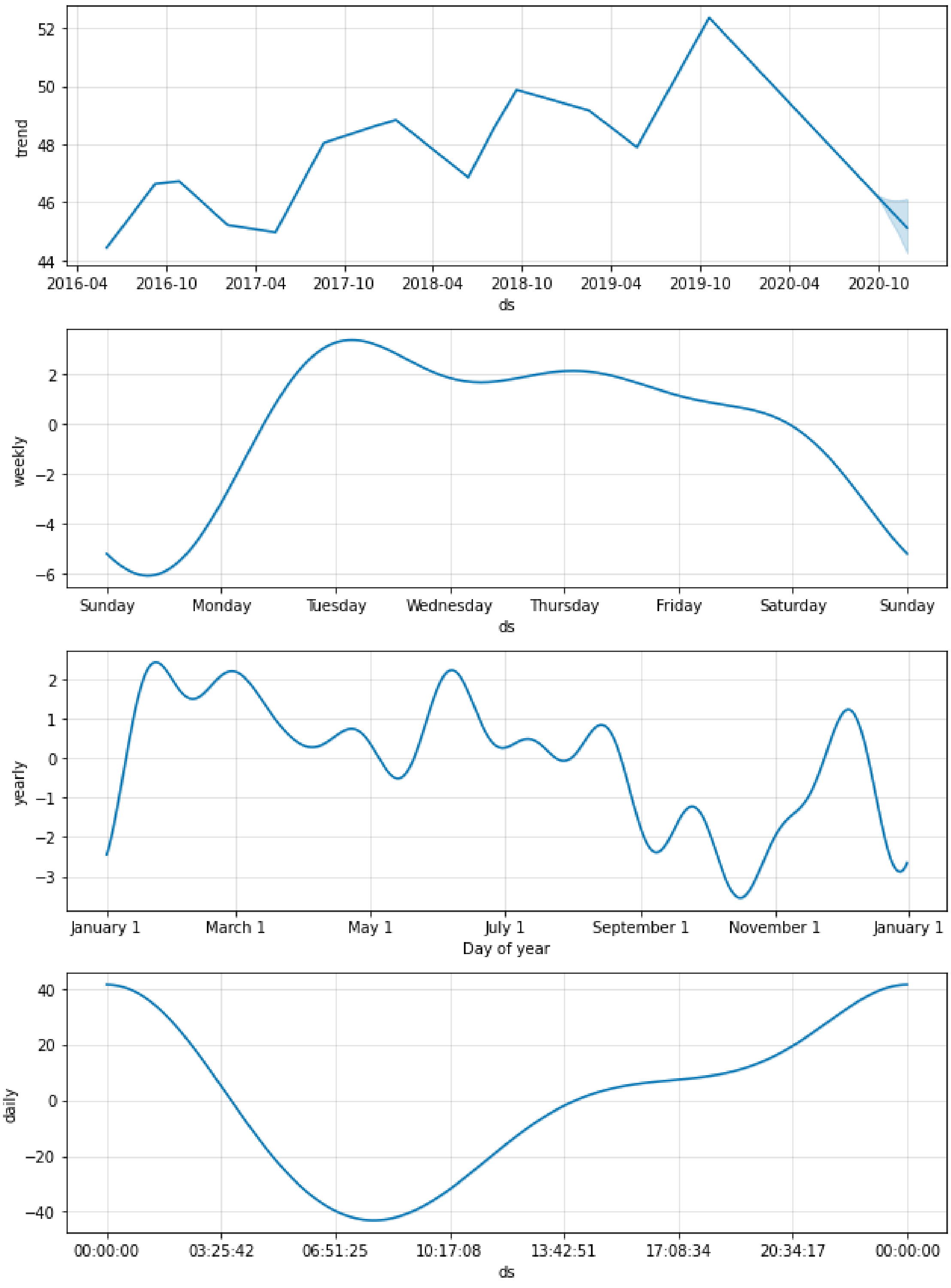
```
2016-06-01 04:00:00  43.379863   35.021499   51.753917
…                            …          …           …
2020-11-30 04:00:00  39.730796   31.443821   48.661232
2020-11-30 05:00:00  24.259673   15.986895   32.260925
2020-11-30 06:00:00  12.294015    3.642285   20.787234
2020-11-30 07:00:00   5.021873   -3.564709   13.621496
2020-11-30 08:00:00   2.826998   -5.794846   10.830538

[39106 rows x 3 columns]
```

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()
```

[ ]:
```
:NdOverlay   [Variable]
   :Curve   [ds]   (value)
```

[ ]:
```
# 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

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 end of the day

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

Answer: Tuesday

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

Answer: end of september

1.11 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')

# Review the DataFrame
df_mercado_sales.head()
```

```
<IPython.core.display.HTML object>
```

```
Saving mercado_daily_revenue.csv to mercado_daily_revenue (2).csv
```

[ ]:

	date	Daily Sales
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
```

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

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

[ ]:
```
:Curve   [index]   (Daily Sales)
```

[ ]:
```
# 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 ( this is not␣
 ↪needed)
mercado_sales_prophet_df = df_mercado_sales.copy()

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

# Visualize the DataFrame
mercado_sales_prophet_df
```

[ ]:
```
             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
..          …          …
495  2020-05-10  17.467814
496  2020-05-11  17.537152
497  2020-05-12  18.031773
498  2020-05-13  19.165315
499  2020-05-14  20.246570

[500 rows x 2 columns]
```

[ ]:
```
# Create the model
mercado_sales_prophet_model = Prophet()

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

```
INFO:fbprophet:Disabling yearly seasonality. Run prophet with
yearly_seasonality=True to override this.
INFO:fbprophet:Disabling daily seasonality. Run prophet with
```

```
daily_seasonality=True to override this.
```

[ ]:
```
<fbprophet.forecaster.Prophet at 0x7fa3fe935e50>
```

[ ]:
```python
# 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
```

[ ]:
```python
# 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_forecast.head()
```

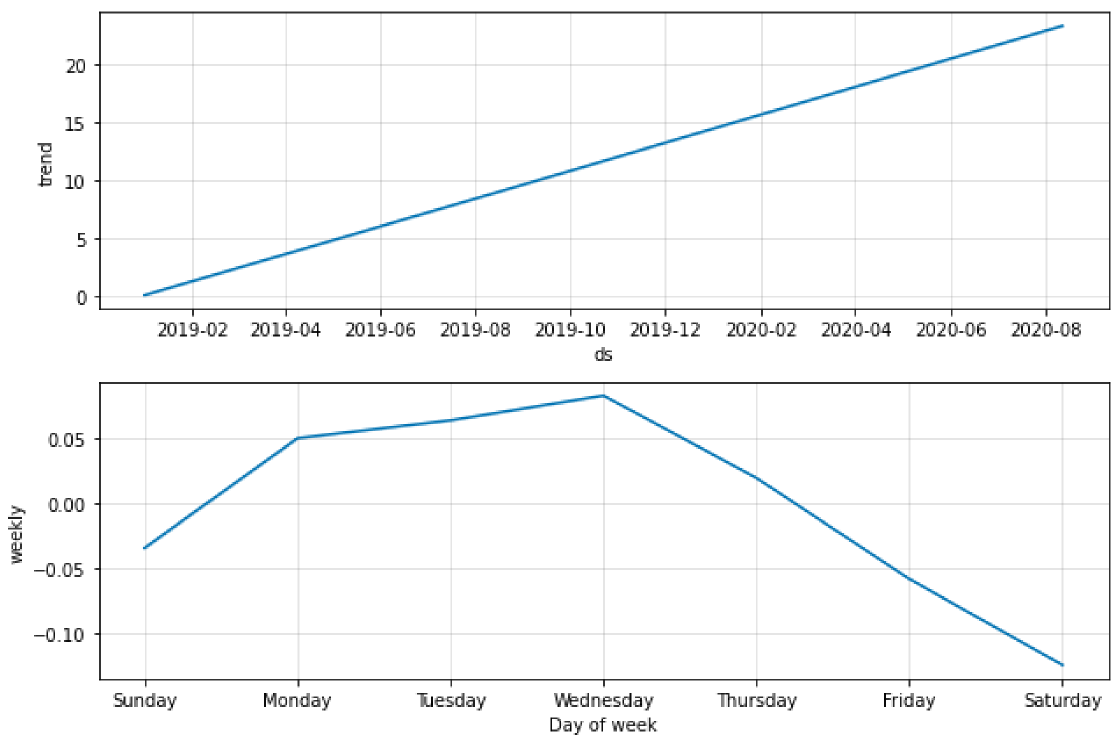
[ ]:
```
          ds     trend  yhat_lower  yhat_upper  trend_lower  trend_upper  \
0 2019-01-01  0.133067   -1.724678    2.164204     0.133067     0.133067
1 2019-01-02  0.172247   -1.681786    2.175038     0.172247     0.172247
2 2019-01-03  0.211428   -1.628050    2.087284     0.211428     0.211428
3 2019-01-04  0.250609   -1.781537    2.159062     0.250609     0.250609
4 2019-01-05  0.289789   -1.666849    2.173484     0.289789     0.289789

   additive_terms  additive_terms_lower  additive_terms_upper    weekly  \
0        0.063730              0.063730              0.063730  0.063730
1        0.082772              0.082772              0.082772  0.082772
2        0.019580              0.019580              0.019580  0.019580
3       -0.057997             -0.057997             -0.057997 -0.057997
4       -0.123972             -0.123972             -0.123972 -0.123972

   weekly_lower  weekly_upper  multiplicative_terms  \
0      0.063730      0.063730                   0.0
1      0.082772      0.082772                   0.0
2      0.019580      0.019580                   0.0
3     -0.057997     -0.057997                   0.0
4     -0.123972     -0.123972                   0.0
```

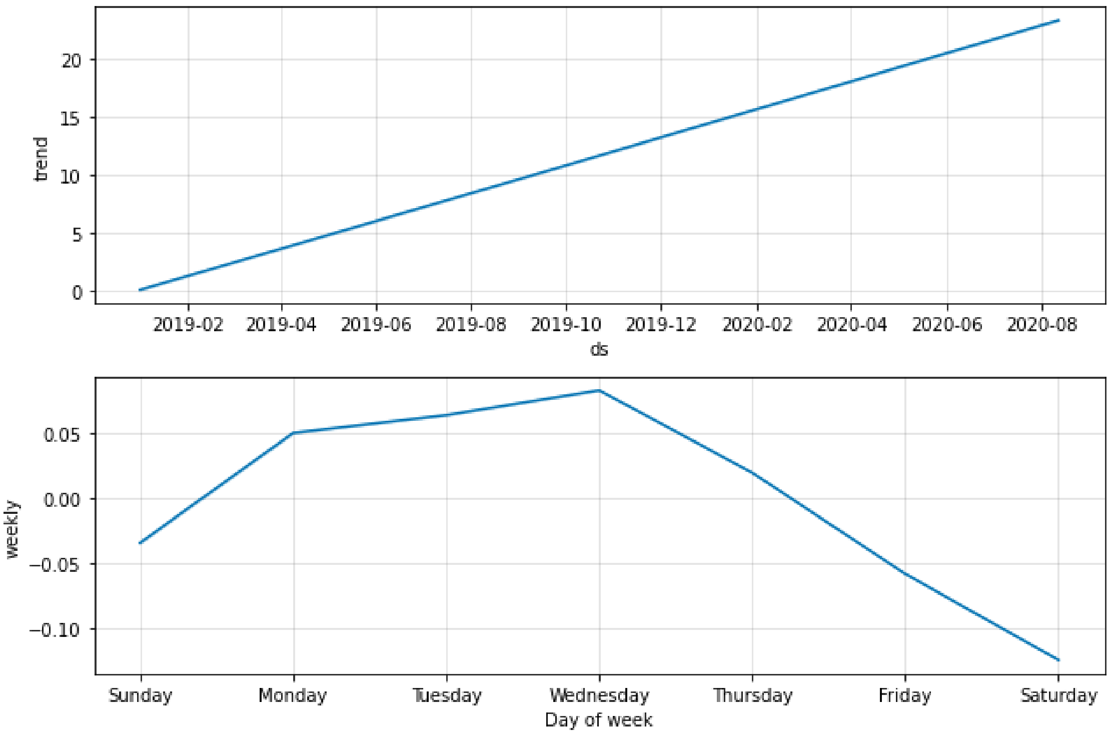
```
   multiplicative_terms_lower  multiplicative_terms_upper      yhat
0                         0.0                         0.0  0.196797
1                         0.0                         0.0  0.255019
2                         0.0                         0.0  0.231008
3                         0.0                         0.0  0.192611
4                         0.0                         0.0  0.165817
```

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)
```

[ ]:





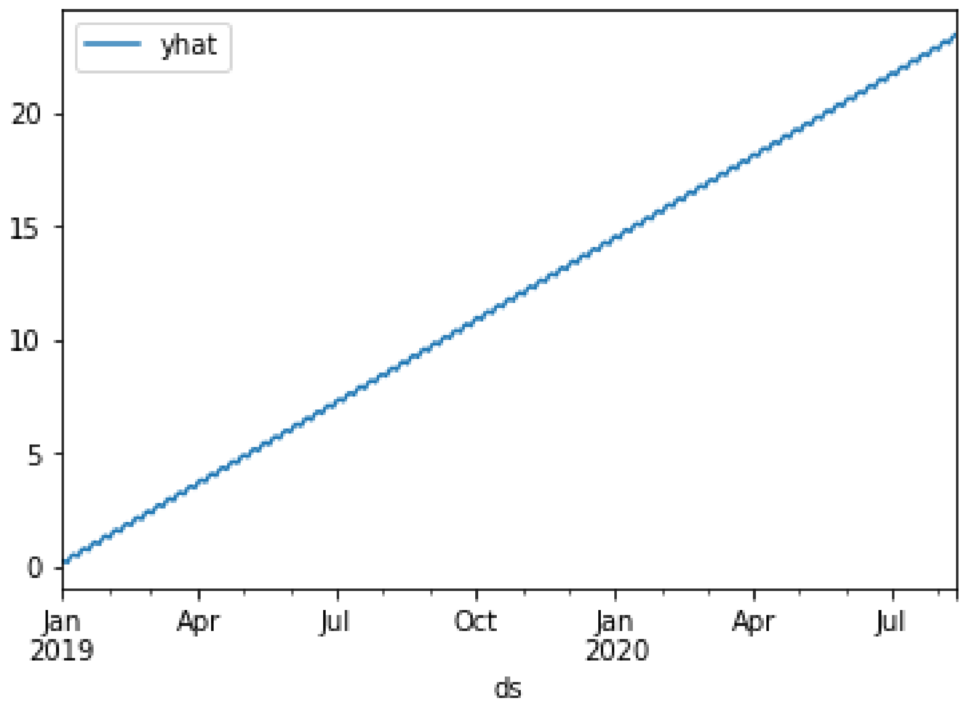
Answer the following question: Question: For example, what are the peak revenue days? (Mondays? Fridays? Something else?)

Answer: Wednesday is the peak revenue Day

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_forecast.plot(x='ds',y='yhat')
```

[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7fa3fe8ce250>



[ ]:
```
# 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
mercado_sales_prophet_forecast.head()
```

[ ]:
```
               trend  yhat_lower  yhat_upper  trend_lower  trend_upper  \
ds
2019-01-01  0.133067   -1.724678    2.164204     0.133067     0.133067
2019-01-02  0.172247   -1.681786    2.175038     0.172247     0.172247
2019-01-03  0.211428   -1.628050    2.087284     0.211428     0.211428
2019-01-04  0.250609   -1.781537    2.159062     0.250609     0.250609
2019-01-05  0.289789   -1.666849    2.173484     0.289789     0.289789

            additive_terms  additive_terms_lower  additive_terms_upper  \
ds
2019-01-01        0.063730              0.063730              0.063730
2019-01-02        0.082772              0.082772              0.082772
2019-01-03        0.019580              0.019580              0.019580
2019-01-04       -0.057997             -0.057997             -0.057997
2019-01-05       -0.123972             -0.123972             -0.123972
```

```
              weekly  weekly_lower  weekly_upper  multiplicative_terms  \
ds
2019-01-01  0.063730      0.063730      0.063730                   0.0
2019-01-02  0.082772      0.082772      0.082772                   0.0
2019-01-03  0.019580      0.019580      0.019580                   0.0
2019-01-04 -0.057997     -0.057997     -0.057997                   0.0
2019-01-05 -0.123972     -0.123972     -0.123972                   0.0

            multiplicative_terms_lower  multiplicative_terms_upper      yhat
ds
2019-01-01                         0.0                         0.0  0.196797
2019-01-02                         0.0                         0.0  0.255019
2019-01-03                         0.0                         0.0  0.231008
2019-01-04                         0.0                         0.0  0.192611
2019-01-05                         0.0                         0.0  0.165817
```

[ ]:
```
# 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.
↪loc['2020-07-01':'2020-09-30']

# Update the column names for the forecast_quarter DataFrame
# to match what the finance division is looking for
mercado_sales_forecast_quarter = mercado_sales_prophet_forecast.
↪rename(columns={'yhat_upper':'best case', 'yhat_lower':'worst case', 'yhat':
↪'most likley'})
mercado_sales_forecast_quarter = mercado_sales_forecast_quarter[['best␣
↪case','worst case','most likley']]
# Review the last five rows of the DataFrame
mercado_sales_forecast_quarter.head()
```

[ ]:
```
            best case  worst case  most likley
ds
2019-01-01   2.164204   -1.724678     0.196797
2019-01-02   2.175038   -1.681786     0.255019
2019-01-03   2.087284   -1.628050     0.231008
2019-01-04   2.159062   -1.781537     0.192611
2019-01-05   2.173484   -1.666849     0.165817
```

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

[ ]:
```
best case       8040.762168
worst case      5797.792379
most likley     6920.017460
dtype: float64
```

1.11.1 Based on the forecast information generated above, produce a sales forecast for the finance division, giving them a number for expected total sales next quarter. Include best and worst case scenarios, to better help the finance team plan.

Answer:
best case 8040.762168 , worst case 5797.792379 , most likley 6920.017460

[ ]:
```

```