Notebook cs.png

Time Series with Financial Data

Financial analysts use time series data such as stock price movements, or a company's sales over time, to analyze a company's performance see.

Investors can take advantage of new growth investing strategies in order to more precisely hone in on stocks or other investments offering above-average profit potential. When it comes to investing in the stock market, there are always a variety of approaches that can be taken. The goal, however, is generally always the same, regardless of the approach – grow your investments and increase your profits see

---

Source:

- Candle Stick Charts with Plotly
- Scatter Plot of Financial Data with Plotly
- Bar Race Charts
- Feature Engineering Techniques For Time Series Data
- Differencing Time Series

Data (from Yahoo Finance):

- Credit Suisse Stock Market Price (April 2009 - March 2023) -- **DATA-CS.csv**
- UBS Group Stock Market Price (April 2009 - March 2023) -- **DATA-UBS.csv**

Author:

- dr. daniel benninger

History:

- 2023-04-06 v2 dbe --- initial version for BINA FS23

Load Libraries and Check Environment

```
import pandas as pd
from datetime import datetime
import plotly.graph_objects as go
```

```
print(pd.__version__)
```

```
1.5.3
```

```
%ls
%cd sample_data
```

```
sample_data/
/content/sample_data
```

```
import warnings
warnings.filterwarnings("ignore")
```

Load Financial Data and Verify Structure/Format/Values

```
# load the financial dataset from the BINA FS23 github repositors
path = 'https://raw.githubusercontent.com/sawubona-gmbh/BINA-FS23-WORK/main/LB10-Regression%2BTimeSeries/Python/DATA-C
data = pd.read_csv(path)

# OPTION: load the financial dataset from a local file
# data = pd.read_csv('DATA-CS.csv')
```

```
data.head(5)
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2009-04-06	29.658203	30.937500	29.541016	30.703125	20.256773	3253043
1	2009-04-07	29.482422	29.960938	29.072266	29.482422	19.451399	1795584
2	2009-04-08	30.361328	30.908203	29.746094	30.644531	20.218113	1202688
3	2009-04-09	31.240234	33.027344	31.025391	32.841797	21.667789	2358579
4	2009-04-13	32.470703	34.423828	32.285156	33.974609	22.415174	1897062

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3524 entries, 0 to 3523
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       3524 non-null   object 
 1   Open       3524 non-null   float64
 2   High       3524 non-null   float64
 3   Low        3524 non-null   float64
 4   Close      3524 non-null   float64
 5   Adj Close  3524 non-null   float64
 6   Volume     3524 non-null   int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 192.8+ KB
```

```
data.describe()
```

	Open	High	Low	Close	Adj Close	Volume
count	3524.000000	3524.000000	3524.000000	3524.000000	3524.000000	3.524000e+03
mean	21.714721	21.902277	21.504521	21.708342	16.745795	4.267032e+06
std	12.494230	12.609161	12.356681	12.490288	7.632176	1.254198e+07
min	0.820000	0.860000	0.820000	0.850000	0.850000	1.301500e+05
25%	12.090000	12.190000	12.020000	12.097500	11.051464	1.283234e+06
50%	17.231445	17.440703	17.124454	17.302578	14.742901	2.185800e+06
75%	28.840000	28.992500	28.625000	28.825000	21.594921	3.759675e+06
max	58.300781	58.671875	57.509766	58.437500	38.651604	4.341040e+08

```
# convert date colume to "datetime" format
data[["Date"]] = data[["Date"]].apply(pd.to_datetime)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3524 entries, 0 to 3523
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype         
---  ------     --------------  -----         
 0   Date       3524 non-null   datetime64[ns]
```

```
 1   Open       3524 non-null   float64
 2   High       3524 non-null   float64
 3   Low        3524 non-null   float64
 4   Close      3524 non-null   float64
 5   Adj Close  3524 non-null   float64
 6   Volume     3524 non-null   int64
dtypes: datetime64[ns](1), float64(5), int64(1)
memory usage: 192.8 KB
```

Select time range and plot time series

Select a specific timeframe

```
df= data[(data['Date'] > "2019-01-01") & (data['Date'] < "2023-01-01")]
# df= data[(data['Date'] > "2018-01-01")]
```

and plot the financial time series OHLC as **candlesticks** using *plotly.graph_objects*

```
fig = go.Figure(data=[go.Candlestick(x=df['Date'],
                open=df['Open'],
                high=df['High'],
                low=df['Low'],
                close=df['Close'])])

fig.update_layout(
    title="Finance Institutes - Stock Market Price <br><sup>CREDIT SUISSE</sup>",
    yaxis_title='US$',
        width=1000, height=600,
    yaxis_range = (0,25))


fig.show()
```

Some Feature Engineering Techniques applied to Financial Time Series Data

Date-Related Features

Information about the day, month, year e.g. *day of the week*, *quarter*, *day/week of year* etc.

```
data['year']=data['Date'].dt.year
data['month']=data['Date'].dt.month
data['day']=data['Date'].dt.day

data['dayofweek_num']=data['Date'].dt.dayofweek
data['dayofyear_num']=data['Date'].dt.dayofyear
data['weekofyear_num']=data['Date'].dt.week
data['quarter_num']=data['Date'].dt.quarter
data['daysinmonth_num']=data['Date'].dt.days_in_month

data.head()
```

```
<ipython-input-12-d7df8bb37f5a>:7: FutureWarning:

Series.dt.weekofyear and Series.dt.week have been deprecated. Please use Series.dt.isocalendar().week i
```

	Date	Open	High	Low	Close	Adj Close	Volume	year	month	day	dayofweek_num
0	2009-04-06	29.658203	30.937500	29.541016	30.703125	20.256773	3253043	2009	4	6	0
1	2009-04-07	29.482422	29.960938	29.072266	29.482422	19.451399	1795584	2009	4	7	1
2	2009-04-08	30.361328	30.908203	29.746094	30.644531	20.218113	1202688	2009	4	8	2
3	2009-04-09	31.240234	33.027344	31.025391	32.841797	21.667789	2358579	2009	4	9	3
4	2009-04-13	32.470703	34.423828	32.285156	33.974609	22.415174	1897062	2009	4	13	0

Lag-Related Features

If we like predicting the stock price for a company. So, the previous day's stock price is important to make a prediction. In other words, the value at time t is greatly affected by the value at time t-1. The past values are known as lags, so t-1 is lag 1, t-2 is lag 2, and so on.

```
data['lag_1'] = data['Close'].shift(1)

dataX = data[['Date', 'lag_1', 'Close']]
dataX.head()
```

	Date	lag_1	Close
0	2009-04-06	NaN	30.703125

```
dataX['performance_1']=dataX['Close']-dataX['lag_1']

dataX.head()
```

```
<ipython-input-14-8f4c8b051d68>:1: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
```

	Date	lag_1	Close	performance_1
0	2009-04-06	NaN	30.703125	NaN
1	2009-04-07	30.703125	29.482422	-1.220703
2	2009-04-08	29.482422	30.644531	1.162109
3	2009-04-09	30.644531	32.841797	2.197266
4	2009-04-13	32.841797	33.974609	1.132812

If the series has a weekly trend, which means the value last Monday can be used to predict the value for this Monday, we should create lag features for seven days.

We can create multiple lag features as well! Let's say we want lag 1 to lag 7 – we can let the model decide which is the most valuable one. So, if we train a linear regression model, it will assign appropriate weights (or coefficients) to the lag features

```
data['lag_1'] = data['Close'].shift(1)
data['lag_2'] = data['Close'].shift(2)
data['lag_3'] = data['Close'].shift(3)
data['lag_4'] = data['Close'].shift(4)
data['lag_5'] = data['Close'].shift(5)
data['lag_6'] = data['Close'].shift(6)
data['lag_7'] = data['Close'].shift(7)

dataX = data[['Date', 'lag_1', 'lag_2', 'lag_3', 'lag_4', 'lag_5', 'lag_6', 'lag_7', 'Close']]
dataX.head(10)
```

	Date	lag_1	lag_2	lag_3	lag_4	lag_5	lag_6	lag_7	Close
0	2009-04-06	NaN	NaN	NaN	NaN	NaN	NaN	NaN	30.703125
1	2009-04-07	30.703125	NaN	NaN	NaN	NaN	NaN	NaN	29.482422
2	2009-04-08	29.482422	30.703125	NaN	NaN	NaN	NaN	NaN	30.644531
3	2009-04-09	30.644531	29.482422	30.703125	NaN	NaN	NaN	NaN	32.841797
4	2009-04-13	32.841797	30.644531	29.482422	30.703125	NaN	NaN	NaN	33.974609
5	2009-04-14	33.974609	32.841797	30.644531	29.482422	30.703125	NaN	NaN	31.689453
6	2009-04-15	31.689453	33.974609	32.841797	30.644531	29.482422	30.703125	NaN	32.910156
7	2009-04-16	32.910156	31.689453	33.974609	32.841797	30.644531	29.482422	30.703125	35.087891
8	2009-04-17	35.087891	32.910156	31.689453	33.974609	32.841797	30.644531	29.482422	33.300781
9	2009-04-20	33.300781	35.087891	32.910156	31.689453	33.974609	32.841797	30.644531	30.361328

Rolling Window Features

How about calculating some statistical values based on past values? This method is called the rolling window method because the window would be different for every data point.

We will select a window size, take the average of the values in the window, and use it as a feature.

```
data['rolling_mean7'] = data['Close'].rolling(window=7).mean()

dataX = data[['Date', 'rolling_mean7', 'Close']]
dataX.head(10)
```

	Date	rolling_mean7	Close
0	2009-04-06	NaN	30.703125
1	2009-04-07	NaN	29.482422
2	2009-04-08	NaN	30.644531
3	2009-04-09	NaN	32.841797
4	2009-04-13	NaN	33.974609
5	2009-04-14	NaN	31.689453
6	2009-04-15	31.749442	32.910156
7	2009-04-16	32.375837	35.087891
8	2009-04-17	32.921317	33.300781
9	2009-04-20	32.880859	30.361328

```
import plotly.express as px
df= dataX[(dataX['Date'] > "2019-01-01") & (dataX['Date'] < "2019-12-31")]
#df.info()

# Create Line plot
fig = px.line(df, x=df['Date'], y=['Close', 'rolling_mean7'])


# Setup Layout
fig.update_layout(
    title="Finance Institutes - Stock Market Price with Rolling Means <br><sup>CREDIT SUISSE</sup>",
    legend_title="Data Points",
    yaxis_title='US$',
    width=1000, height=600,
    yaxis_range = (10,15))

# Display the plot
fig.show()
```


```
data['rolling_mean20'] = data['Close'].rolling(window=20).mean()
data['rolling_mean60'] = data['Close'].rolling(window=60).mean()

dataY = data[['Date', 'Close','rolling_mean20','rolling_mean60',]]
dataY.head(25)
```

	Date	Close	rolling_mean20	rolling_mean60
0	2009-04-06	30.703125	NaN	NaN
1	2009-04-07	29.482422	NaN	NaN
2	2009-04-08	30.644531	NaN	NaN
3	2009-04-09	32.841797	NaN	NaN
4	2009-04-13	33.974609	NaN	NaN
5	2009-04-14	31.689453	NaN	NaN
6	2009-04-15	32.910156	NaN	NaN
7	2009-04-16	35.087891	NaN	NaN
8	2009-04-17	33.300781	NaN	NaN
9	2009-04-20	30.361328	NaN	NaN
10	2009-04-21	33.193359	NaN	NaN
11	2009-04-22	32.226563	NaN	NaN
12	2009-04-23	37.587891	NaN	NaN
13	2009-04-24	37.460938	NaN	NaN
14	2009-04-27	36.308594	NaN	NaN
15	2009-04-28	36.376953	NaN	NaN
16	2009-04-29	37.792969	NaN	NaN
17	2009-04-30	37.382813	NaN	NaN
18	2009-05-01	37.744141	NaN	NaN
19	2009-05-04	39.863281	34.346680	NaN
20	2009-05-05	38.369141	34.729981	NaN
21	2009-05-06	38.388672	35.175293	NaN
22	2009-05-07	36.708984	35.478516	NaN
23	2009-05-08	40.546875	35.863770	NaN
24	2009-05-11	40.097656	36.169922	NaN

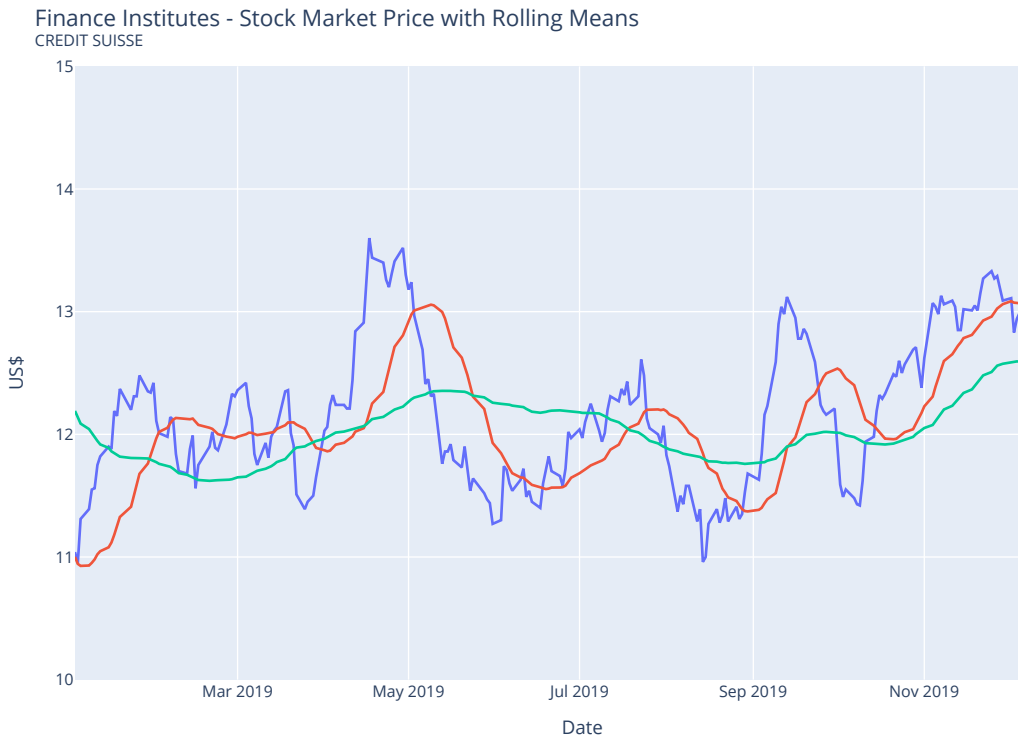
```
import plotly.express as px
df= dataY[(dataY['Date'] > "2019-01-01") & (dataY['Date'] < "2019-12-31")]
```

```
#df.info()

# Create Line plot
fig = px.line(df, x=df['Date'], y=['Close', 'rolling_mean20','rolling_mean60'])


# Setup Layout
fig.update_layout(
    title="Finance Institutes - Stock Market Price with Rolling Means <br><sup>CREDIT SUISSE</sup>",
    legend_title="Data Points",
    yaxis_title='US$',
    width=1000, height=600,
    yaxis_range = (10,15))

# Display the plot
fig.show()
```



Differencing Time Series

Differencing is a method of transforming a time series dataset. Differencing is performed by subtracting the previous observation from the current observation.

Differencing can help stabilize the mean of the time series by removing changes in the level of a time series, and so eliminating (or reducing) trend and seasonality.

```
dataZ = data[['Date', 'Close']]
dataZ['diff1'] = dataZ['Close'].diff(periods=1)

dataZ.head()
```

```
<ipython-input-20-55cf9bddf0dc>:2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
```

	Date	Close	diff1
0	2009-04-06	30.703125	NaN
1	2009-04-07	29.482422	-1.220703
2	2009-04-08	30.644531	1.162109
3	2009-04-09	32.841797	2.197266

```
dataZ['diff2'] = dataZ['Close'].diff(periods=2)
dataZ['diff5'] = dataZ['Close'].diff(periods=5)

dataZ.head(10)
```

```
<ipython-input-21-7e2bdcedfedc>:1: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing

<ipython-input-21-7e2bdcedfedc>:2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
```

	Date	Close	diff1	diff2	diff5
0	2009-04-06	30.703125	NaN	NaN	NaN
1	2009-04-07	29.482422	-1.220703	NaN	NaN
2	2009-04-08	30.644531	1.162109	-0.058594	NaN
3	2009-04-09	32.841797	2.197266	3.359375	NaN
4	2009-04-13	33.974609	1.132812	3.330078	NaN
5	2009-04-14	31.689453	-2.285156	-1.152344	0.986328
6	2009-04-15	32.910156	1.220703	-1.064453	3.427734
7	2009-04-16	35.087891	2.177735	3.398438	4.443360
8	2009-04-17	33.300781	-1.787110	0.390625	0.458984
9	2009-04-20	30.361328	-2.939453	-4.726563	-3.613281

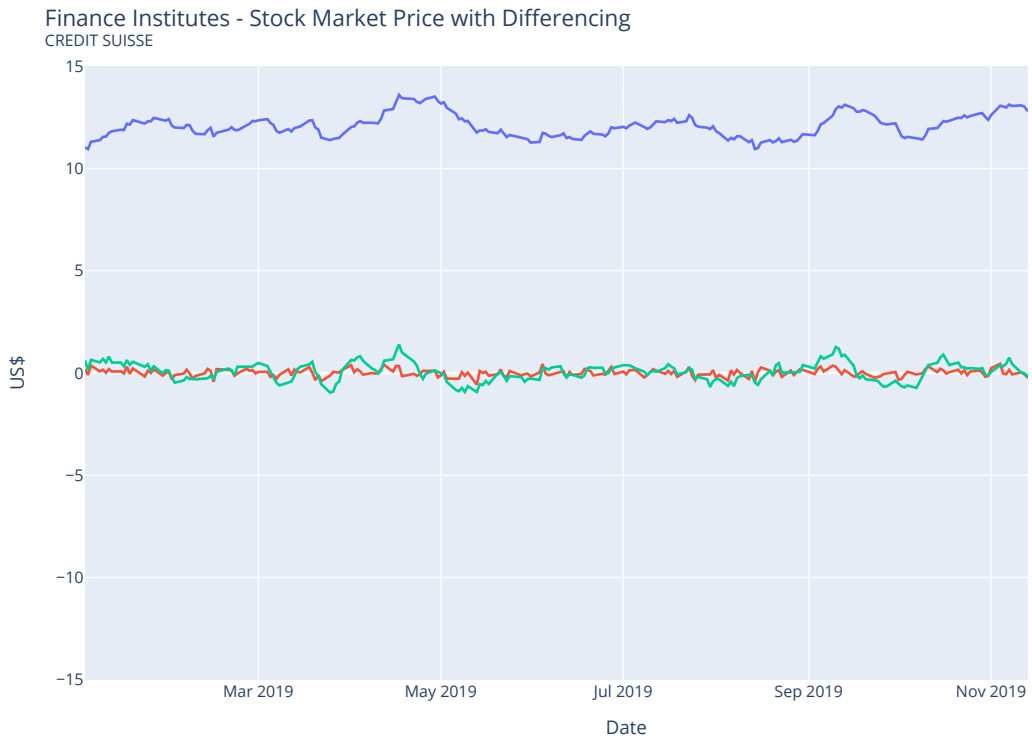
```
import plotly.express as px
df=·dataZ[(dataZ['Date']·>·"2019-01-01")·&·(dataZ['Date']·<·"2019-12-31")]

# Create Line plot
fig = px.line(df, x=df['Date'], y=['Close', 'diff1','diff5'])


# Setup Layout
fig.update_layout(
    title="Finance Institutes - Stock Market Price with Differencing <br><sup>CREDIT SUISSE</sup>",
    legend_title="Data Points",
    vaxis title='US$'.
```

```
    width=1000, height=600,
    yaxis_range = (-15,15))

# Display the plot
fig.show()
```



ADD ON: Line or Bar Charts for Time Series?

```
dataZ = data[['Date', 'Close']]
dataZ.info()
data.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3524 entries, 0 to 3523
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    3524 non-null   datetime64[ns]
 1   Close   3524 non-null   float64
dtypes: datetime64[ns](1), float64(1)
memory usage: 55.2 KB
```

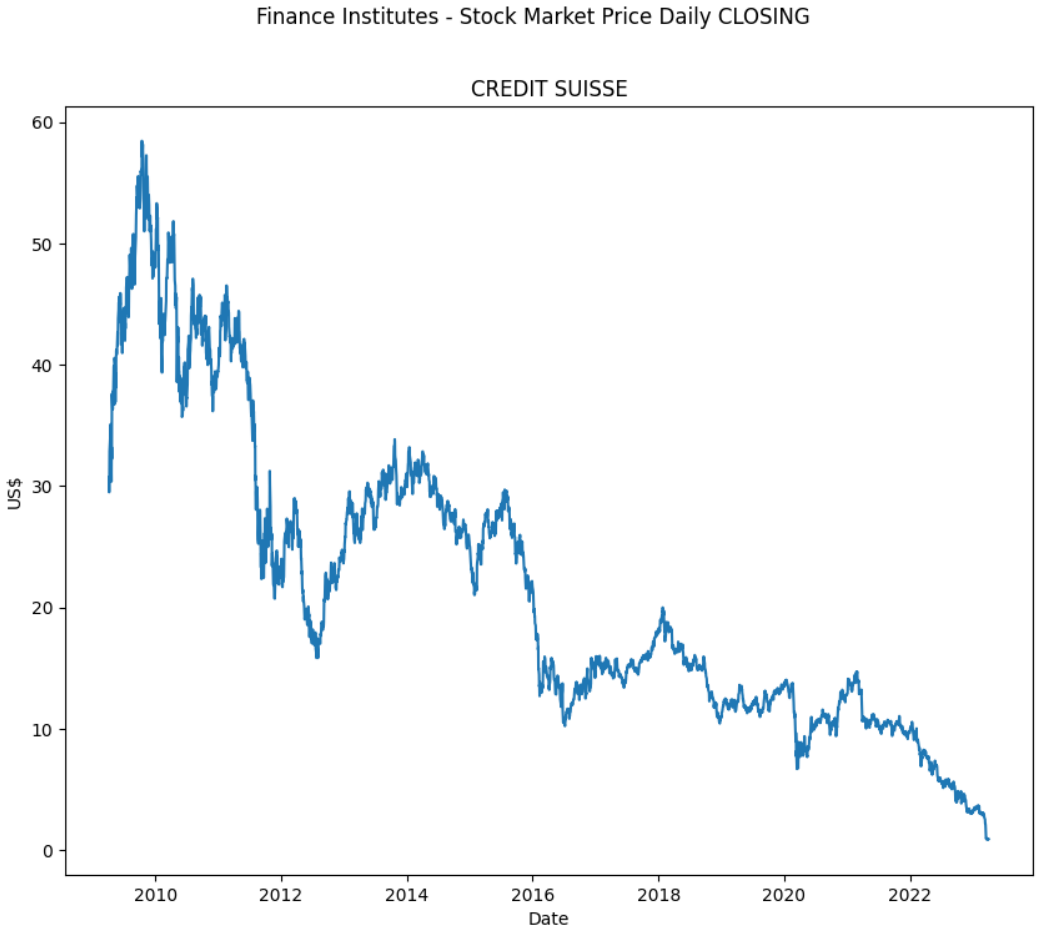
```
import matplotlib.pyplot as plt

dataZ = data[['Date', 'Close']]

plt.figure(figsize=(10, 8))
# as LINE chart
plt.plot(dataZ.Date, dataZ.Close)
# as BAR chart
#plt.bar(dataZ.Date, dataZ.Close)

plt.suptitle("Finance Institutes - Stock Market Price Daily CLOSING")
plt.title("CREDIT SUISSE")
plt.xlabel('Date')
plt.ylabel('US$')

plt.show()
```



ADD ON: Visualizing time series data in Heatmap form

```
!pip install calplot
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting calplot
  Downloading calplot-0.1.7.5.tar.gz (132 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 132.3/132.3 kB 3.4 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: matplotlib in /usr/local/lib/python3.9/dist-packages (from calplot) (3.7.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.9/dist-packages (from calplot) (1.22.4)
Requirement already satisfied: pandas>=1 in /usr/local/lib/python3.9/dist-packages (from calplot) (1.5.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.9/dist-packages (from pandas>=1-
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (from pandas>=1->calplot)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.9/dist-packages (from matplotlib->calplot)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.9/dist-packages (from matplotlib->calpl
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.9/dist-packages (from matplotlib->calpl
Requirement already satisfied: importlib-resources>=3.2.0 in /usr/local/lib/python3.9/dist-packages (from matplot
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib->calp
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.9/dist-packages (from matplotlib->calp
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib->calplo
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib->calplot)
Requirement already satisfied: zipp>=3.1.0 in /usr/local/lib/python3.9/dist-packages (from importlib-resources>=3
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from python-dateutil>=2.8.1->p
Building wheels for collected packages: calplot
  Building wheel for calplot (setup.py) ... done
  Created wheel for calplot: filename=calplot-0.1.7.5-py3-none-any.whl size=8119 sha256=2a8549155e8c0f744454f0771
  Stored in directory: /root/.cache/pip/wheels/a4/51/68/89dbd39aa6abbe8e34f410a810421335b157fb162b99841c30
Successfully built calplot
Installing collected packages: calplot
Successfully installed calplot-0.1.7.5
```

```
dataZ = data[['Date','Close']]
df= dataZ[(dataZ['Date'] > "2019-01-01") & (dataZ['Date'] < "2022-01-01")]
df.head()
```

	Date	Close
2452	2019-01-02	11.04
2453	2019-01-03	10.95
2454	2019-01-04	11.31
2455	2019-01-07	11.39
2456	2019-01-08	11.55

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 757 entries, 2452 to 3208
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype         
---  ------  --------------  -----         
 0   Date    757 non-null    datetime64[ns]
 1   Close   757 non-null    float64       
dtypes: datetime64[ns](1), float64(1)
memory usage: 17.7 KB
```

```
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace = True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 757 entries, 2019-01-02 to 2021-12-31
Data columns (total 1 columns):
```

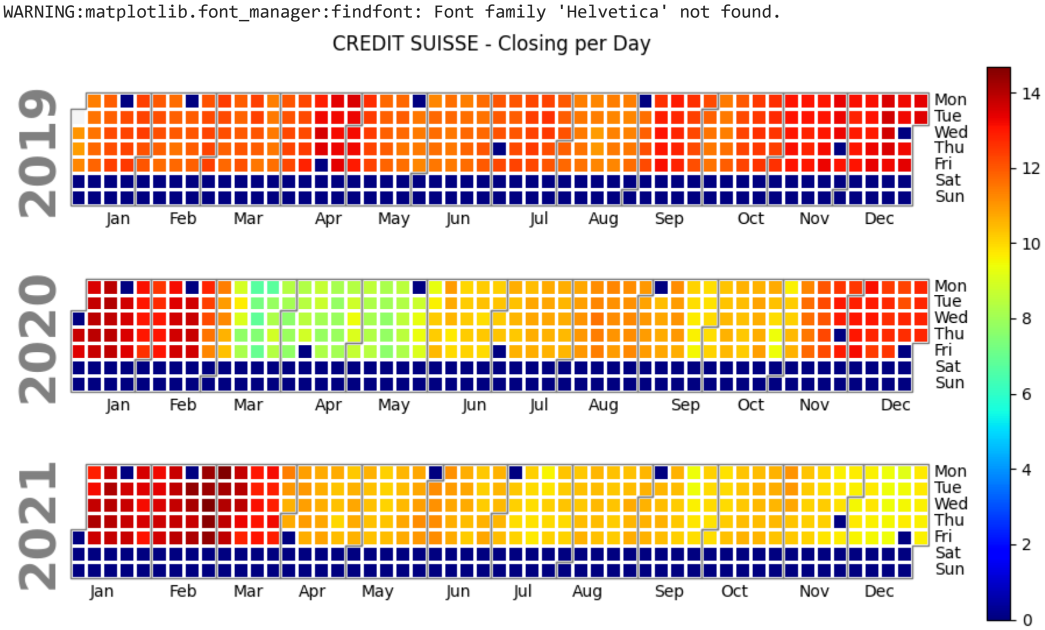
```
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Close   757 non-null    float64
dtypes: float64(1)
memory usage: 11.8 KB
<ipython-input-20-b9a73327b629>:1: SettingWithCopyWarning: 
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#retu
  df['Date'] = pd.to_datetime(df['Date'])
```

```
import calplot
fig1 = calplot.calplot(data = df['Close'],
                      cmap = 'jet',
                      figsize = (10, 5),
                      suptitle = "CREDIT SUISSE - Closing per Day",
                      )

import pylab
pylab.savefig('cs-heatmap.png')
```

```
WARNING:matplotlib.font_manager:findfont: Font family 'Helvetica' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Helvetica' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Helvetica' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Helvetica' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Helvetica' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Helvetica' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Helvetica' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Helvetica' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Helvetica' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Helvetica' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Helvetica' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Helvetica' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Helvetica' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Helvetica' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Helvetica' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Helvetica' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Helvetica' not found.
```



```
---
### **ADD ON:** Systematic Feature Engineering with [tsfres
tsfresh.readthedocs.io/en/latest/text/introduction.html)
**tsfresh** is used for systematic feature engineering fro
and other sequential data. These data have in common that
ordered by an independent variable. The most common indepe
variable is time (time series).
If we want to calculate different characteristics of time
as the maximum or minimum, the average or the number of te
peaks, without tsfresh, we have to calculate all those cha
manually.
tsfresh automates this process calculating and returning a
features automatically.
```

ADD ON: Systematic Feature Engineering with tsfresh

tsfresh is used for systematic feature engineering from time-series and other sequential data. These data have in common that they are ordered by an independent variable. The most common independent variable is time (time series).
If we want to calculate different characteristics of time series such as the maximum or minimum, the average or the number of temporary peaks, without tsfresh, we have to calculate all those characteristics manually.
tsfresh automates this process calculating and returning all those features automatically.

```
!pip install -U tsfresh
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting tsfresh
  Downloading tsfresh-0.20.0-py2.py3-none-any.whl (98 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 98.2/98.2 kB 2.8 MB/s eta 0:00:00
Requirement already satisfied: tqdm>=4.10.0 in /usr/local/lib/python3.9/dist-packages (from tsfresh) (4.65.0)
Requirement already satisfied: scikit-learn>=0.22.0 in /usr/local/lib/python3.9/dist-packages (from tsfresh) (1.2
Collecting stumpy>=1.7.2
  Downloading stumpy-1.11.1-py3-none-any.whl (136 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 136.2/136.2 kB 4.7 MB/s eta 0:00:00
Requirement already satisfied: statsmodels>=0.13 in /usr/local/lib/python3.9/dist-packages (from tsfresh) (0.13.5
Requirement already satisfied: distributed>=2.11.0 in /usr/local/lib/python3.9/dist-packages (from tsfresh) (202
Requirement already satisfied: numpy>=1.15.1 in /usr/local/lib/python3.9/dist-packages (from tsfresh) (1.22.4)
Requirement already satisfied: patsy>=0.4.1 in /usr/local/lib/python3.9/dist-packages (from tsfresh) (0.5.3)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.9/dist-packages (from tsfresh) (2.2.1)
Requirement already satisfied: scipy>=1.2.0 in /usr/local/lib/python3.9/dist-packages (from tsfresh) (1.10.1)
Requirement already satisfied: pandas>=0.25.0 in /usr/local/lib/python3.9/dist-packages (from tsfresh) (1.5.3)
Requirement already satisfied: requests>=2.9.1 in /usr/local/lib/python3.9/dist-packages (from tsfresh) (2.27.1)
Requirement already satisfied: dask[dataframe]>=2.9.0 in /usr/local/lib/python3.9/dist-packages (from tsfresh) (2
Requirement already satisfied: click>=7.0 in /usr/local/lib/python3.9/dist-packages (from dask[dataframe]>=2.9.0-
Requirement already satisfied: partd>=0.3.10 in /usr/local/lib/python3.9/dist-packages (from dask[dataframe]>=2.9
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.9/dist-packages (from dask[dataframe]>=2
Requirement already satisfied: pyyaml>=5.3.1 in /usr/local/lib/python3.9/dist-packages (from dask[dataframe]>=2.9
Requirement already satisfied: toolz>=0.8.2 in /usr/local/lib/python3.9/dist-packages (from dask[dataframe]>=2.9
Requirement already satisfied: fsspec>=0.6.0 in /usr/local/lib/python3.9/dist-packages (from dask[dataframe]>=2.9
Requirement already satisfied: psutil>=5.0 in /usr/local/lib/python3.9/dist-packages (from distributed>=2.11.0->t
Requirement already satisfied: urllib3 in /usr/local/lib/python3.9/dist-packages (from distributed>=2.11.0->tsfre
Requirement already satisfied: locket>=1.0.0 in /usr/local/lib/python3.9/dist-packages (from distributed>=2.11.0-
Requirement already satisfied: jinja2 in /usr/local/lib/python3.9/dist-packages (from distributed>=2.11.0->tsfres
Requirement already satisfied: zict>=0.1.3 in /usr/local/lib/python3.9/dist-packages (from distributed>=2.11.0->t
Requirement already satisfied: msgpack>=0.6.0 in /usr/local/lib/python3.9/dist-packages (from distributed>=2.11.0
Requirement already satisfied: tornado>=6.0.3 in /usr/local/lib/python3.9/dist-packages (from distributed>=2.11.0
Requirement already satisfied: sortedcontainers!=2.0.0,!=2.0.1 in /usr/local/lib/python3.9/dist-packages (from di
Requirement already satisfied: tblib>=1.6.0 in /usr/local/lib/python3.9/dist-packages (from distributed>=2.11.0->
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (from pandas>=0.25.0->tsfre
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.9/dist-packages (from pandas>=0.2
Requirement already satisfied: six in /usr/local/lib/python3.9/dist-packages (from patsy>=0.4.1->tsfresh) (1.16.0
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.9/dist-packages (from requests
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-packages (from requests>=2.9.1->tsfr
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.9/dist-packages (from requests>=2.9.1
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.9/dist-packages (from scikit-learn>
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.9/dist-packages (from scikit-learn>=0.22.0
Requirement already satisfied: numba>=0.54 in /usr/local/lib/python3.9/dist-packages (from stumpy>=1.7.2->tsfresh
Requirement already satisfied: llvmlite<0.40,>=0.39.0dev0 in /usr/local/lib/python3.9/dist-packages (from numba>=
Requirement already satisfied: setuptools in /usr/local/lib/python3.9/dist-packages (from numba>=0.54->stumpy>=1
```

```
Requirement already satisfied: heapdict in /usr/local/lib/python3.9/dist-packages (from zict>=0.1.3->distributed
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.9/dist-packages (from jinja2->distribute
Installing collected packages: stumpy, tsfresh
Successfully installed stumpy-1.11.1 tsfresh-0.20.0
```

```
dataZ = data[['Date','Open','High','Low','Close','Volume','year','month','day']]
dataZ.head()
```

	Date	Open	High	Low	Close	Volume	year	month	day
0	2009-04-06	29.658203	30.937500	29.541016	30.703125	3253043	2009	4	6
1	2009-04-07	29.482422	29.960938	29.072266	29.482422	1795584	2009	4	7
2	2009-04-08	30.361328	30.908203	29.746094	30.644531	1202688	2009	4	8
3	2009-04-09	31.240234	33.027344	31.025391	32.841797	2358579	2009	4	9
4	2009-04-13	32.470703	34.423828	32.285156	33.974609	1897062	2009	4	13

```
# settings for feature extraction
from tsfresh.feature_extraction import ComprehensiveFCParameters
settings = ComprehensiveFCParameters()
# e.g.
kind_to_fc_parameters = {
    "Open": {"mean": None},
    "Close": {"maximum": None, "minimum": None}
}

# automated feature extraction
from tsfresh.feature_extraction import extract_features
features = extract_features(dataZ, column_id="Date", column_sort="Date", default_fc_parameters=settings)

#features = extract_features(dataZ, column_id="Date", column_sort="Date")
```

```
WARNING:tsfresh.feature_extraction.settings:Dependency not available for matrix_profile, this feature w
Feature Extraction:   8%|▉         | 2145/28192 [01:13<14:50, 29.26it/s]
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-28-234c3714e780> in <cell line: 12>()
     10 # automated feature extraction
     11 from tsfresh.feature_extraction import extract_features
---> 12 features = extract_features(dataZ, column_id="Date", column_sort="Date",
default_fc_parameters=settings)
     13
     14 #features = extract_features(dataZ, column_id="Date", column_sort="Date")

                                  15 frames
/usr/local/lib/python3.9/dist-packages/pandas/core/dtypes/inference.py in is_array_like(obj)
    186
    187
--> 188 def is_array_like(obj) -> bool:
    189     """
    190     Check if the object is array-like.

KeyboardInterrupt:
```

SEARCH STACK OVERFLOW

```
features.info()
```

```
features.head()
```

```
features.describe()
```

Zum Bearbeiten doppelklicken (oder Eingabe)

✓ 1 s Abgeschlossen um 10:37