Theme: Statistical Related Risk: Volatility & Statistical Related Risk: Correlation

Statistics

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

import yfinance as yf
# from ydata_profiling import ProfileReport
# from IPython.display import IFrame

#statistical tests
import statsmodels.api as sm
from scipy import stats


#import yfinance as yf

btcusdt_ticker = "BTC-USD"
start_date = "2020-01-01"
end_date = "2023-04-25"

df = yf.download(btcusdt_ticker, start=start_date, end=end_date)
print(df.head())
```

```
[**********100%**********]  1 of 1 completed
                  Open         High          Low        Close    Adj Close  \
Date
2020-01-01  7194.892090  7254.330566  7174.944336  7200.174316  7200.174316
2020-01-02  7202.551270  7212.155273  6935.270020  6985.470215  6985.470215
2020-01-03  6984.428711  7413.715332  6914.996094  7344.884277  7344.884277
2020-01-04  7345.375488  7427.385742  7309.514160  7410.656738  7410.656738
2020-01-05  7410.451660  7544.497070  7400.535645  7411.317383  7411.317383

                Volume
Date
2020-01-01  18565664997
2020-01-02  20802083465
2020-01-03  28111481032
2020-01-04  18444271275
```

✓ 0s completed at 11:24 AM

Double-click (or enter) to edit

```
# installing additional package for desriptive statistics in html format
!pip install ydata_profiling

import ydata_profiling as ypr
from ydata_profiling import ProfileReport
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheel
Collecting ydata_profiling
  Downloading ydata_profiling-4.1.2-py2.py3-none-any.whl (345 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 345.9/345.9 kB 2.5 MB/s eta 0:00:00
Collecting matplotlib<3.7,>=3.2
  Downloading matplotlib-3.6.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.8/11.8 MB 79.4 MB/s eta 0:00:00
Requirement already satisfied: pydantic<1.11,>=1.8.1 in /usr/local/lib/python3.10/
Requirement already satisfied: PyYAML<6.1,>=5.0.0 in /usr/local/lib/python3.10/dis
Collecting tqdm<4.65,>=4.48.2
  Downloading tqdm-4.64.1-py2.py3-none-any.whl (78 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 78.5/78.5 kB 9.6 MB/s eta 0:00:00
Requirement already satisfied: jinja2<3.2,>=2.11.1 in /usr/local/lib/python3.10/di
Collecting scipy<1.10,>=1.4.1
  Downloading scipy-1.9.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.w
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 33.7/33.7 MB 16.4 MB/s eta 0:00:00
Requirement already satisfied: seaborn<0.13,>=0.10.1 in /usr/local/lib/python3.10/
Requirement already satisfied: numpy<1.24,>=1.16.0 in /usr/local/lib/python3.10/di
Collecting phik<0.13,>=0.11.1
  Downloading phik-0.12.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.w
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 679.5/679.5 kB 49.0 MB/s eta 0:00:00
Collecting typeguard<2.14,>=2.13.2
  Downloading typeguard-2.13.3-py3-none-any.whl (17 kB)
Requirement already satisfied: statsmodels<0.14,>=0.13.2 in /usr/local/lib/python3
Collecting multimethod<1.10,>=1.4
  Downloading multimethod-1.9.1-py3-none-any.whl (10 kB)
Requirement already satisfied: requests<2.29,>=2.24.0 in /usr/local/lib/python3.10
Collecting htmlmin==0.1.12
  Downloading htmlmin-0.1.12.tar.gz (19 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: pandas!=1.4.0,<1.6,>1.1 in /usr/local/lib/python3.1
Collecting visions[type_image_path]==0.7.5
  Downloading visions-0.7.5-py3-none-any.whl (102 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 102.7/102.7 kB 12.2 MB/s eta 0:00:00
Collecting imagehash==4.3.1
  Downloading ImageHash-4.3.1-py2.py3-none-any.whl (296 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 296.5/296.5 kB 12.6 MB/s eta 0:00:00
Requirement already satisfied: PyWavelets in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: networkx>=2.4 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: attrs>=19.3.0 in /usr/local/lib/python3.10/dist-pac
Collecting tangled-up-in-unicode>=0.0.4
```

```
  Downloading tangled_up_in_unicode-0.2.0-py3-none-any.whl (4.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.7/4.7 MB 44.5 MB/s eta 0:00:00
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist
Requirement already satisfied: pyparsing>=2.2.1 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/d
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: joblib>=0.14.1 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: typing-extensions>=4.2.0 in /usr/local/lib/python3.
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dis
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3
Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (frc
Building wheels for collected packages: htmlmin
  Building wheel for htmlmin (setup.py) ... done
  Created wheel for htmlmin: filename=htmlmin-0.1.12-py3-none-any.whl size=27096 s
  Stored in directory: /root/.cache/pip/wheels/dd/91/29/a79cecb328d01739e64017b6fb
Successfully built htmlmin
Installing collected packages: htmlmin, typeguard, tqdm, tangled-up-in-unicode, sc
  Attempting uninstall: tqdm
    Found existing installation: tqdm 4.65.0
    Uninstalling tqdm-4.65.0:
      Successfully uninstalled tqdm-4.65.0
  Attempting uninstall: scipy
    Found existing installation: scipy 1.10.1
    Uninstalling scipy-1.10.1:
      Successfully uninstalled scipy-1.10.1
  Attempting uninstall: matplotlib
    Found existing installation: matplotlib 3.7.1
    Uninstalling matplotlib-3.7.1:
      Successfully uninstalled matplotlib-3.7.1
Successfully installed htmlmin-0.1.12 imagehash-4.3.1 matplotlib-3.6.3 multimethoc
WARNING: The following packages were previously imported in this runtime:
  [matplotlib,mpl_toolkits]
You must restart the runtime in order to use newly installed versions.
```

RESTART RUNTIME

```
df_btc = df

returns_btc = df

profile = ProfileReport(df_btc)
profile.to_file('df_btc.html')
```

Summarize dataset: 51/51 [00:10<00:00, 2.94it/s,

100% Completed]

Generate report structure: 1/1 [00:06<00:00,

100% 6.96s/it]

Render HTML : 100% 1/1 [00:02<00:00, 2.95s/it]

```
profile
```

Overview

Dataset statistics

Number of variables	6
Number of observations	1210
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	66.2 KiB
Average record size in memory	56.0 B

Variable types

Numeric	6

Alerts

`Open` is highly overall correlated with `High` and 3 other fields (High, Low, Close, Adj Close)	**High correlation**
`High` is highly overall correlated with `Open` and 3 other fields (Open, Low, Close, Adj Close)	**High correlation**
`Low` is highly overall correlated with `Open` and 3 other fields (Open, High, Close, Adj Close)	**High correlation**
`Close` is highly overall correlated with `Open` and 3 other fields (Open	**High correlation**

I am interested only in the information in df_btc['Adj Close'], and I will use it for further analysis. So, in case you are interested in the other columns like: Open, High, Low, Close or Volume, the same type of analysis can be applied towards them as well. Let's do some statistical evaluations to prove whether the data is normally distributed. I will use the visual method called: "Q-Q Plot". Here you can see that the data is NOT normally distributed because otherwise it should be very close to the 45-degree line added to the plot (because normally distributed data do so).

```
returns_btc['returns'] = df['Adj Close'].pct_change()
returns_btc = returns_btc.dropna()

profile_returns = ProfileReport(returns_btc)
profile_returns
```

Summarize dataset: 100%	65/65 [00:19<00:00, 3.33it/s, Completed]
Generate report structure: 100%	1/1 [00:09<00:00, 9.93s/it]
Render HTML: 100%	1/1 [00:03<00:00, 3.52s/it]

Overview

Dataset statistics

Number of variables	7
Number of observations	1209
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%

Duplicate rows (%)	0.0%
Total size in memory	75.6 KiB
Average record size in memory	64.0 B

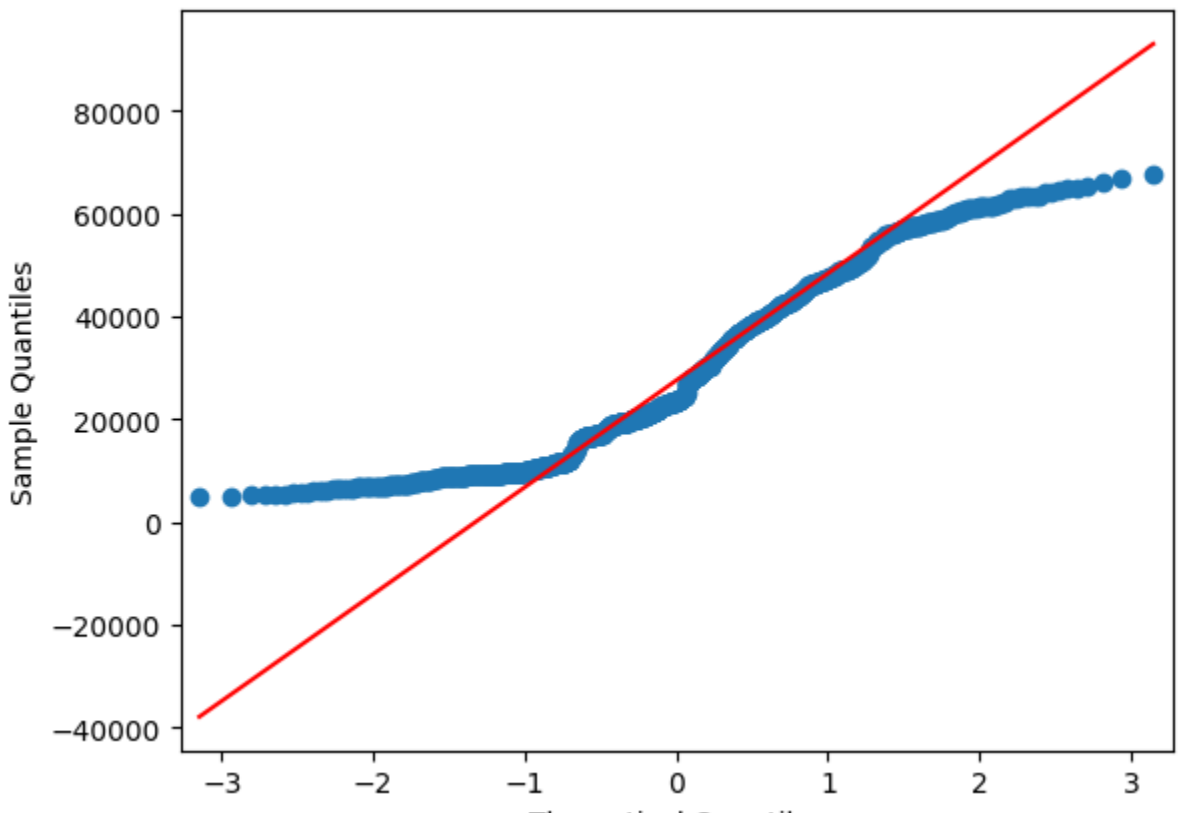
Variable types

Numeric	7

Alerts

`Open` is highly overall correlated with `High` and 3 other fields (High, Low, Close, Adj Close)	**High correlation**
`High` is highly overall correlated with `Open` and 3 other fields (Open, Low, Close, Adj Close)	**High correlation**
`Low` is highly overall correlated with `Open` and 3 other fields (Open,	**High correlation**

```
# import statsmodels.api as sm

# create Q-Q plot with 45-degree line added to the plot
"""
this plot can be uset for seen the qqplo of:
- returns_btc['Adj Close']
- returns_btc['returns']
"""
fig = sm.qqplot(returns_btc['Adj Close'], line='q')
plt.show()
```



Theoretical Quantiles

Visualizations are not exactly scientific proof for anything because every human being has his own interpretations and abstract thinking, and it is possible for someone not to be able to see it as it is. In this case I will use the very formal statistical test: Shapiro-Wilk test and it will mathematically (statistically) prove or not whether there is a normal distribution. If the value of the Shapiro test's p-value is less than 0.05. This means that with 95% confidence the data is NOT normally distributed

```
#from scipy import stats

# perform Shapiro-Wilk test
shapiro_test = stats.shapiro(returns_btc['returns'])
print(shapiro_test)
```

```
ShapiroResult(statistic=0.9171086549758911, pvalue=3.349220646237701e-25)
```

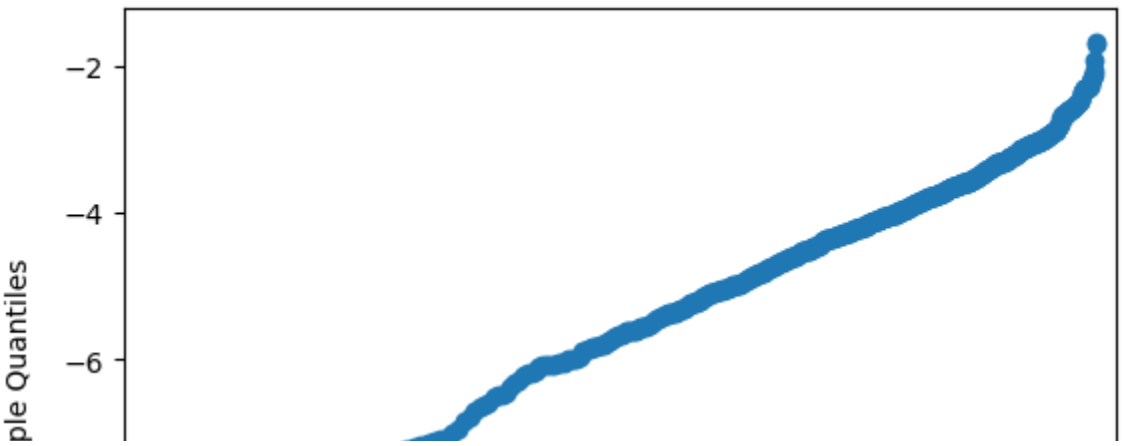
So, what can we do in this case?

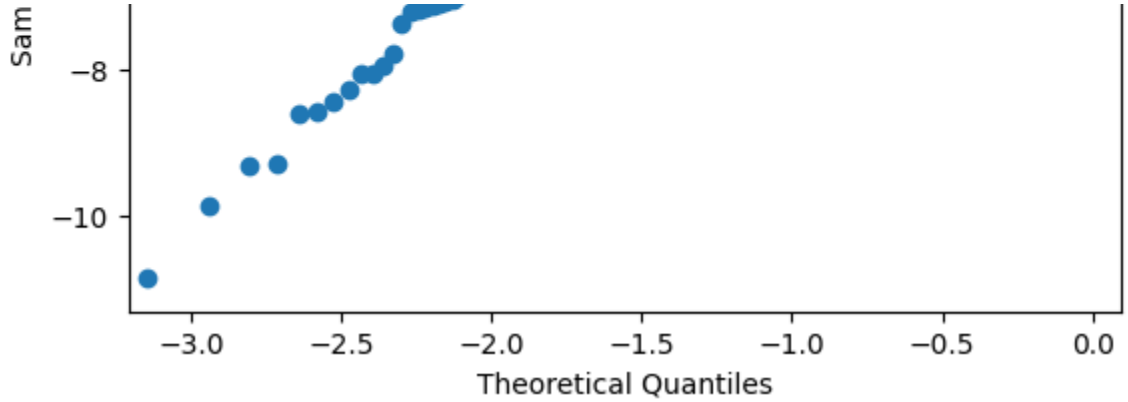
I will try to normalize the data be using the so called numpy.log() method which is a simple logarithm transformation over all the values in the data.

And the I will use once more the previous methods to check for normallity

```
logged_btc = np.log(returns_btc['returns'])

fig = sm.qqplot(logged_btc)
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/pandas/core/arraylike.py:402: RuntimeWarni
  result = getattr(ufunc, method)(*inputs, **kwargs)
```






```
# perform Shapiro-Wilk test
shapiro_test = stats.shapiro(logged_btc)
print(shapiro_test)
```

```
ShapiroResult(statistic=nan, pvalue=1.0)
```

And now with a p-value of 1 from the Shapiro Wilk test for normallity we can say that the loged data of Bitcoin returns is normally distributed with very few outliers in the tails.

Correlations

```
!pip install statsmodels
!pip install pmdarima
!pip install vectorbt
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheel
Requirement already satisfied: statsmodels in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: pandas>=0.25 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: scipy>=1.3 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (frc
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheel
Collecting pmdarima
  Downloading pmdarima-2.0.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_6
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.8/1.8 MB 18.4 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: pandas>=0.19 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.10/dis
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in /usr/local/lib/pythc
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in /usr/local/lib/
Requirement already satisfied: statsmodels>=0.13.2 in /usr/local/lib/python3.10/di
```

```
Requirement already satisfied: statsmodels>=0.13.2 in /usr/local/lib/python3.10/di
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (fro
Installing collected packages: pmdarima
Successfully installed pmdarima-2.0.3
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheel
Collecting vectorbt
  Downloading vectorbt-0.25.0-py3-none-any.whl (526 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 526.7/526.7 kB 7.7 MB/s eta 0:00:00
Collecting dateparser
  Downloading dateparser-1.1.8-py2.py3-none-any.whl (293 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 293.8/293.8 kB 26.5 MB/s eta 0:00:00
Requirement already satisfied: pytz in /usr/local/lib/python3.10/dist-packages (fr
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packag
Collecting dill
  Downloading dill-0.3.6-py3-none-any.whl (110 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 110.5/110.5 kB 12.0 MB/s eta 0:00:00
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (fr
Requirement already satisfied: imageio in /usr/local/lib/python3.10/dist-packages
Collecting schedule
  Downloading schedule-1.2.0-py2.py3-none-any.whl (11 kB)
Requirement already satisfied: numba>=0.56.0 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: numpy>=1.16.5 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: plotly>=4.12.0 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (f
Requirement already satisfied: ipywidgets>=7.0.0 in /usr/local/lib/python3.10/dist
Collecting mypy-extensions
  Downloading mypy_extensions-1.0.0-py3-none-any.whl (4.7 kB)
Requirement already satisfied: ipykernel>=4.5.1 in /usr/local/lib/python3.10/dist-
```

```
from statsmodels.tsa.stattools import adfuller, grangercausalitytests
from statsmodels.tools.tools import add_constant
from statsmodels.tsa.tsatools import lagmat2ds
from statsmodels.regression.linear_model import OLS
from statsmodels.tsa.arima.model import ARIMA
import statsmodels.api as sm
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import acf, pacf, ccf
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import yfinance as yf
import vectorbt as vbt
import pmdarima as pmd
from scipy.ndimage import shift
```

```
import plotly.graph_objects as go



df = yf.download("BTC-USD", start="2020-01-01", end="2023-04-25")

df_10yt = yf.download("^TNX", start="2020-01-01", end="2023-04-25")[['Adj Close']].rena
df_10yt['US_10_Year_Treasury_Yield'] *= 0.01
df = df.join(df_10yt, how='left')
df = df.dropna(subset=['US_10_Year_Treasury_Yield'])
df['Return'] = df['Adj Close'].pct_change()
df.index = pd.to_datetime(df.index)
df = df[['Return', 'US_10_Year_Treasury_Yield']].iloc[1:].copy()

df['Return'].describe()
```

```
[**********100%**********]  1 of 1 completed
[**********100%**********]  1 of 1 completed
count    832.000000
mean       0.002676
std        0.044690
min       -0.371695
25%       -0.016921
50%        0.001386
75%        0.022990
max        0.211097
Name: Return, dtype: float64
```

```
df
```

Date	Return	US_10_Year_Treasury_Yield
2020-01-03	0.051452	0.01788
2020-01-06	0.057773	0.01811
2020-01-07	0.050774	0.01827
2020-01-08	-0.010269	0.01874
2020-01-09	-0.024851	0.01858
...	...	...
2023-04-18	0.032349	0.03572
2023-04-19	-0.051809	0.03602
2023-04-20	-0.020008	0.03545
2023-04-21	-0.034309	0.03570
2023-04-24	0.009108	0.03515

832 rows × 2 columns

Augmented Dickey–Fuller test: The testing procedure for the ADF test is the same as for the Dickey–Fuller test [16] but it is applied to the model

$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \sum_{i=1}^{p-1} \delta_i \Delta y_{t-i} + \epsilon_t$

The unit root test is then carried out under the null hypothesis γ = 0 [17]

To ensure the stability of the results, it is essential to verify that both time series are stationary before conducting a Granger causality analysis. We will employ the Augmented Dickey-Fuller (ADF) test to confirm stationarity, with the p-value being used as a threshold, where a value exceeding 0.05 indicates non-stationarity.

```
for col in ['US_10_Year_Treasury_Yield', 'Return']:
    p_value = adfuller(df[col])[1]
    if p_value > 0.05:
        print(f'{col} is nonstationary with p-value {p_value}, so we need to do differe
    else:
        print(f'{col} is stationary with p-value {p_value}')
```

```
US_10_Year_Treasury_Yield is nonstationary with p-value 0.9419752839401622, so we
Return is stationary with p-value 1.797948926210907e-17
```

```
df['US_10_Year_Treasury_Yield'] = df['US_10_Year_Treasury_Yield'].diff()
df = df.rename(columns={'US_10_Year_Treasury_Yield': f'US_10_Year_Treasury_Yield_diff_1
df = df.dropna(subset=['US_10_Year_Treasury_Yield_diff_1'])
p_value = adfuller(df['US_10_Year_Treasury_Yield_diff_1'])[1]
print(p_value)
```

```
6.883099792070528e-13
```

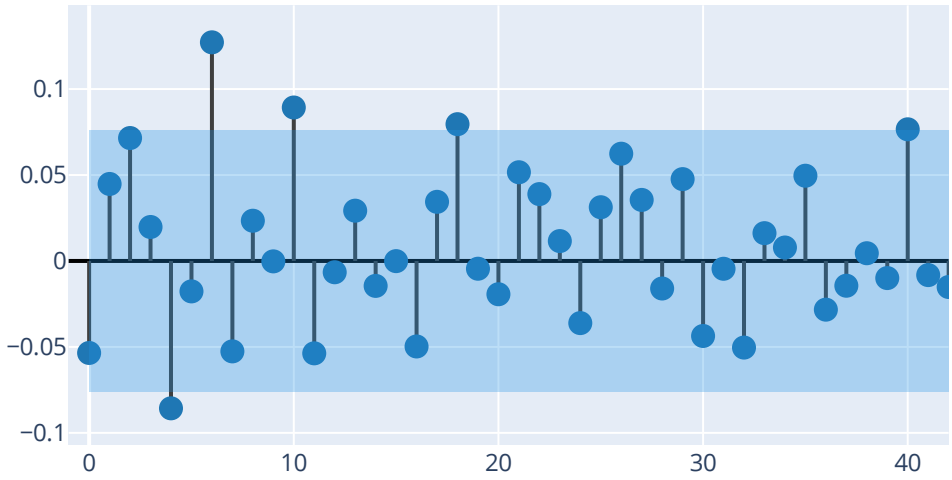
```
split_point = int(0.2*len(df))
df_test = df.iloc[-split_point:]
df = df.iloc[:-split_point]
print(f"df length: {len(df)}, df_test length: {len(df_test)}")
```

```
df length: 665, df_test length: 166
```

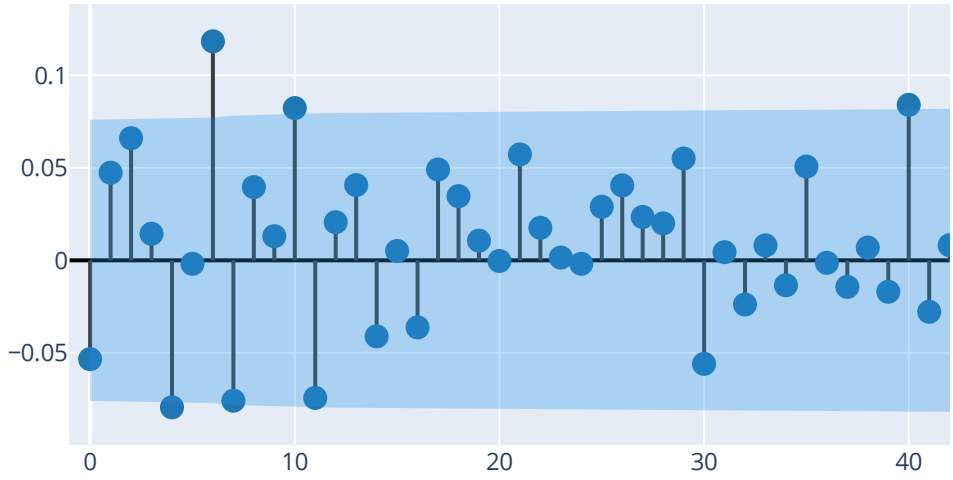
```
def create_corr_plot(df, factor='', plot_pacf=False):
    """
    Code source: https://community.plotly.com/t/plot-pacf-plot-acf-autocorrelation-plot
    """
    series = df[factor]
    corr_array = pacf(series.dropna(), nlags=100, alpha=0.05) if plot_pacf else acf(ser
    corr_array = (corr_array[0][1:], corr_array[1][1:])
```

```
    lower_y = corr_array[1][:,0] - corr_array[0]
    upper_y = corr_array[1][:,1] - corr_array[0]

    fig = go.Figure()
    [fig.add_scatter(x=(x,x), y=(0,corr_array[0][x]), mode='lines',line_color='#3f3f3f'
     for x in range(len(corr_array[0]))]
    fig.add_scatter(x=np.arange(len(corr_array[0])), y=corr_array[0], mode='markers', m
                    marker_size=12)
    fig.add_scatter(x=np.arange(len(corr_array[0])), y=upper_y, mode='lines', line_colo
    fig.add_scatter(x=np.arange(len(corr_array[0])), y=lower_y, mode='lines', fillcolor
            fill='tonexty', line_color='rgba(255,255,255,0)')
    fig.update_traces(showlegend=False)
    fig.update_xaxes(range=[-1,42])
    fig.update_yaxes(zerolinecolor='#000000')

    title=f'Partial Autocorrelation (PACF): {factor}' if plot_pacf else f'Autocorrelati
    fig.update_layout(title=title, width=600, height=400)
    fig.show()


create_corr_plot(df, 'Return', True)
create_corr_plot(df, 'Return', False)
create_corr_plot(df, 'US_10_Year_Treasury_Yield_diff_1', True)
create_corr_plot(df, 'US_10_Year_Treasury_Yield_diff_1', False)
```

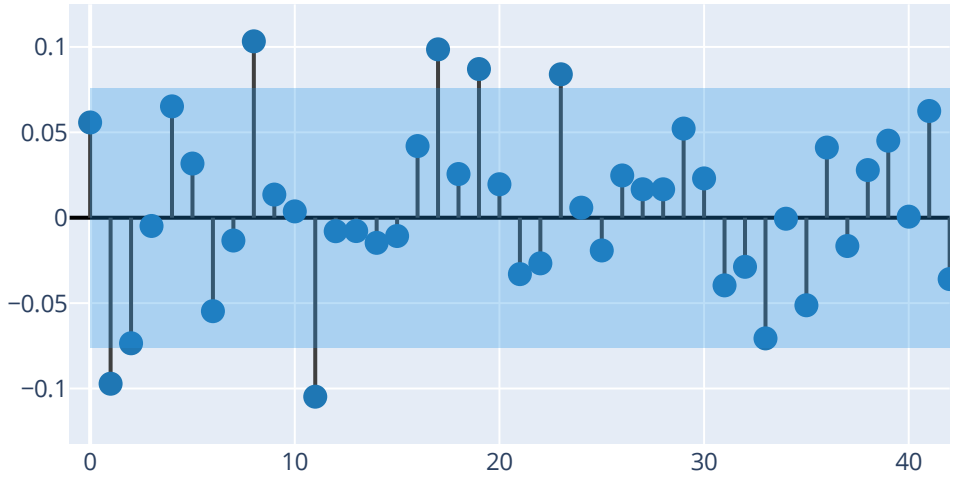
Partial Autocorrelation (PACF): Return



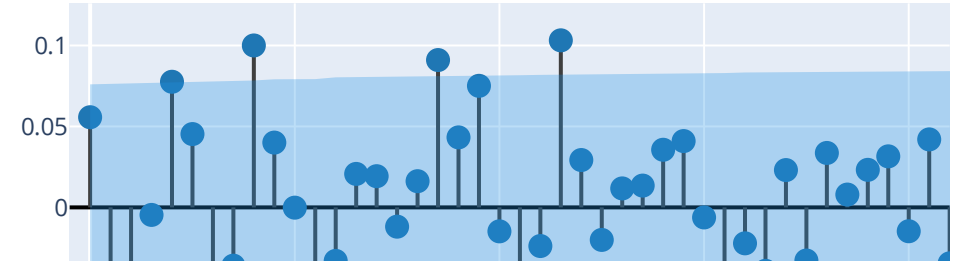
Autocorrelation (ACF): Return

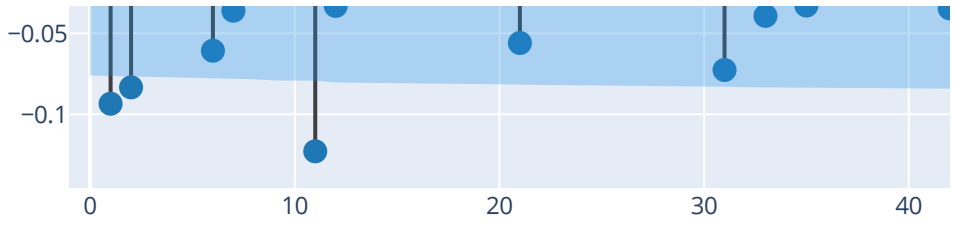


Partial Autocorrelation (PACF): US_10_Year_Treasury_Yield_diff_1



Autocorrelation (ACF): US_10_Year_Treasury_Yield_diff_1

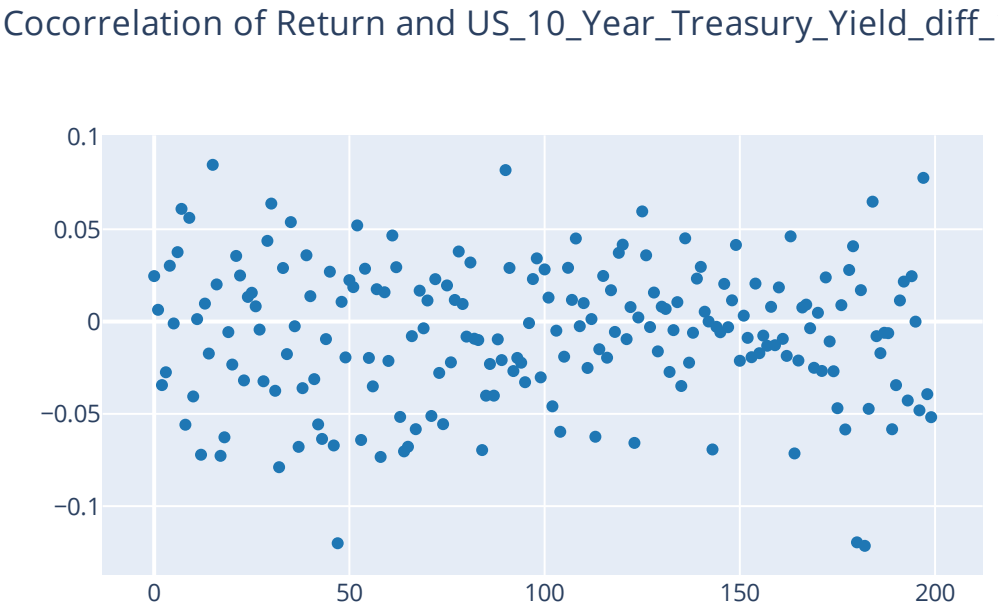




We can observe that there were no outliers in the earlier periods, but a persistent pattern of non-decay and frequent outliers outside of the range appeared in the later periods.

If we observe the correlation between US Treasury bond yields and Bitcoin we can see that there is no obvious relationship. However, we should do granger causality after we consider each lags of themself

```
corr_array = ccf(df['Return'], df['US_10_Year_Treasury_Yield_diff_1'])[:200]
fig = go.Figure()
fig.add_scatter(x=np.arange(len(corr_array)), y=corr_array, mode='markers', marker_colo
                marker_size=6)
fig.update_traces(showlegend=False)

title=f"Cocorrelation of {' and '.join(df.columns)}"
fig.update_layout(title=title, width=600, height=400)
fig.show()
```



X Granger-causes Y if the addition of the lagged values of X to the prediction of Y improves the prediction of Y, where the prediction is made using a linear regression model of the form:

$Y_t = \alpha + \sum_{i=1}^{p} \beta_i Y_{t-i} + \sum_{i=1}^{q} \gamma_i X_{t-i} + \epsilon_t$

Granger causality is a statistical technique used to determine whether a time series X can predict future values of another time series Y. Specifically, X is said to Granger-cause Y if past values of X, along with past values of Y, are statistically significant predictors of future values of Y. This is typically established using t-tests and F-tests on lagged values of X and Y.[15]

```
MAX_LAG = 100
result = grangercausalitytests(df, MAX_LAG, verbose=False)

for i in range(1, MAX_LAG + 1):
    if all(pvalue < 0.05 for pvalue in [r[1] for r in result[i][0].values()]):
        break
print(f"The lag number we consider to form the AR model with max lags = {i}")
print(result[i])
```

```
The lag number we consider to form the AR model with max lags = 47
({'ssr_ftest': (1.4640301942385112, 0.027184323824048208, 523.0, 47), 'ssr_chi2tes
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 1., 0., 0.],
       [0., 0., 0., ..., 0., 1., 0.]])])
```

```
train_data = lagmat2ds(df.values, i, trim="both", dropex=1)

model = OLS(train_data[:, 0], add_constant(train_data[:, 1:], prepend=False)).fit()
model.summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.201
Model:	OLS	**Adj. R-squared:**	0.058
Method:	Least Squares	**F-statistic:**	1.404
Date:	Wed, 03 May 2023	**Prob (F-statistic):**	0.0119
Time:	08:23:31	**Log-Likelihood:**	1109.3
No. Observations:	618	**AIC:**	-2029.
Df Residuals:	523	**BIC:**	-1608.
Df Model:	94		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
x1	-0.0560	0.041	-1.364	0.173	-0.137	0.025

x2	0.0745	0.041	1.821	0.069	-0.006	0.155
x3	0.0513	0.041	1.250	0.212	-0.029	0.132
x4	-0.0222	0.041	-0.542	0.588	-0.103	0.058
x5	-0.0815	0.041	-1.989	0.047	-0.162	-0.001
x6	-0.0223	0.041	-0.542	0.588	-0.103	0.059
x7	0.1346	0.041	3.268	0.001	0.054	0.216
x8	-0.0637	0.042	-1.530	0.127	-0.146	0.018
x9	-0.0047	0.042	-0.112	0.911	-0.087	0.077
x10	0.0174	0.042	0.414	0.679	-0.065	0.100
x11	0.0728	0.042	1.732	0.084	-0.010	0.155
x12	-0.0754	0.042	-1.790	0.074	-0.158	0.007
x13	0.0268	0.042	0.634	0.526	-0.056	0.110
x14	0.0350	0.042	0.827	0.409	-0.048	0.118
x15	-0.0417	0.042	-0.989	0.323	-0.124	0.041
x16	-0.0378	0.042	-0.899	0.369	-0.121	0.045
x17	-0.0295	0.042	-0.699	0.485	-0.112	0.053
x18	0.0459	0.042	1.094	0.274	-0.036	0.128
x19	0.0735	0.042	1.753	0.080	-0.009	0.156
x20	-0.0324	0.042	-0.771	0.441	-0.115	0.050
x21	-0.0036	0.042	-0.085	0.933	-0.086	0.079
x22	0.1019	0.042	2.433	0.015	0.020	0.184
x23	0.0362	0.042	0.860	0.390	-0.047	0.119
x24	-0.0106	0.042	-0.251	0.802	-0.093	0.072
x25	-0.0101	0.042	-0.239	0.811	-0.093	0.073
x26	0.0413	0.042	0.984	0.326	-0.041	0.124
x27	0.0470	0.042	1.113	0.266	-0.036	0.130
x28	0.0245	0.042	0.578	0.564	-0.059	0.108
x29	-0.0107	0.042	-0.254	0.800	-0.094	0.072
x30	0.0538	0.042	1.278	0.202	-0.029	0.137
x31	-0.0117	0.042	-0.278	0.781	-0.094	0.071
x32	-0.0094	0.042	-0.223	0.824	-0.092	0.073
x33	-0.0653	0.042	-1.549	0.122	-0.148	0.018
x34	0.0098	0.042	0.232	0.817	-0.073	0.093
x35	-0.0324	0.042	-0.764	0.445	-0.116	0.051
x36	0.0386	0.042	0.917	0.359	-0.044	0.121
x37	0.0076	0.042	0.183	0.855	-0.075	0.090
x38	-0.0011	0.042	-0.026	0.979	-0.084	0.081
x39	0.0143	0.042	0.342	0.733	-0.068	0.097
x40	0.0188	0.042	0.452	0.651	-0.063	0.101
x41	0.0758	0.041	1.833	0.067	-0.005	0.157
x42	0.0216	0.041	0.523	0.601	-0.060	0.103
x43	-0.0049	0.041	-0.118	0.906	-0.086	0.076
x44	-0.0191	0.042	-0.458	0.647	-0.101	0.063
x45	-0.0276	0.041	-0.666	0.505	-0.109	0.054
x46	0.0690	0.041	1.667	0.096	-0.012	0.150

x47	0.0593	0.041	1.439	0.151	-0.022	0.140
x48	-0.1405	3.526	-0.040	0.968	-7.066	6.785
x49	-1.4077	3.520	-0.400	0.689	-8.323	5.507
x50	-3.6921	3.507	-1.053	0.293	-10.581	3.197
x51	-2.3706	3.482	-0.681	0.496	-9.212	4.471
x52	-2.1286	3.439	-0.619	0.536	-8.885	4.628
x53	-0.8455	3.432	-0.246	0.805	-7.587	5.896
x54	4.6089	3.432	1.343	0.180	-2.134	11.352
x55	-5.6301	3.418	-1.647	0.100	-12.345	1.085
x56	1.9945	3.427	0.582	0.561	-4.739	8.728
x57	-5.2990	3.437	-1.542	0.124	-12.051	1.453
x58	-2.7518	3.409	-0.807	0.420	-9.449	3.945
x59	-6.9279	3.392	-2.042	0.042	-13.591	-0.264
x60	-1.2917	3.419	-0.378	0.706	-8.009	5.425
x61	-4.6094	3.450	-1.336	0.182	-11.387	2.168
x62	4.9181	3.457	1.422	0.155	-1.874	11.710
x63	-1.4209	3.444	-0.413	0.680	-8.187	5.345
x64	-6.9054	3.455	-1.998	0.046	-13.694	-0.117
x65	-6.2151	3.464	-1.794	0.073	-13.019	0.589
x66	2.6280	3.484	0.754	0.451	-4.216	9.471
x67	-6.5963	3.479	-1.896	0.059	-13.431	0.238
x68	3.4986	3.492	1.002	0.317	-3.362	10.359
x69	3.4777	3.477	1.000	0.318	-3.352	10.307
x70	0.9705	3.474	0.279	0.780	-5.854	7.795
x71	0.5608	3.481	0.161	0.872	-6.278	7.399
x72	0.6609	3.513	0.188	0.851	-6.241	7.563
x73	4.2869	3.508	1.222	0.222	-2.605	11.179
x74	2.2273	3.509	0.635	0.526	-4.666	9.120
x75	-1.6180	3.508	-0.461	0.645	-8.509	5.273
x76	1.9980	3.509	0.569	0.569	-4.895	8.891
x77	3.7962	3.500	1.085	0.279	-3.079	10.672
x78	-2.1174	3.490	-0.607	0.544	-8.973	4.738
x79	-8.2332	3.489	-2.360	0.019	-15.087	-1.379
x80	1.5128	3.497	0.433	0.665	-5.357	8.383
x81	-3.7095	3.510	-1.057	0.291	-10.604	3.185
x82	3.0143	3.518	0.857	0.392	-3.898	9.926
x83	-1.9300	3.498	-0.552	0.581	-8.801	4.941
x84	-3.5731	3.512	-1.017	0.310	-10.473	3.327
x85	1.5465	3.517	0.440	0.660	-5.363	8.456
x86	1.0002	3.488	0.287	0.774	-5.853	7.853
x87	-1.5921	3.497	-0.455	0.649	-8.463	5.279
x88	-0.0594	3.492	-0.017	0.986	-6.919	6.800
x89	-2.8451	3.498	-0.813	0.416	-9.717	4.027
x90	-5.9672	3.489	-1.710	0.088	-12.821	0.887
x91	-3.5421	3.500	-1.012	0.312	-10.418	3.334

x92	2.0373	3.531	0.577	0.564	-4.900	8.975
x93	-6.3790	3.532	-1.806	0.071	-13.318	0.560
x94	-9.9218	3.543	-2.800	0.005	-16.882	-2.962
const	0.0036	0.002	1.926	0.055	-7.24e-05	0.007

Omnibus:	28.105	**Durbin-Watson:**	1.943
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	81.417
Skew:	-0.014	**Prob(JB):**	2.09e-18
Kurtosis:	4.778	**Cond. No.**	2.59e+03

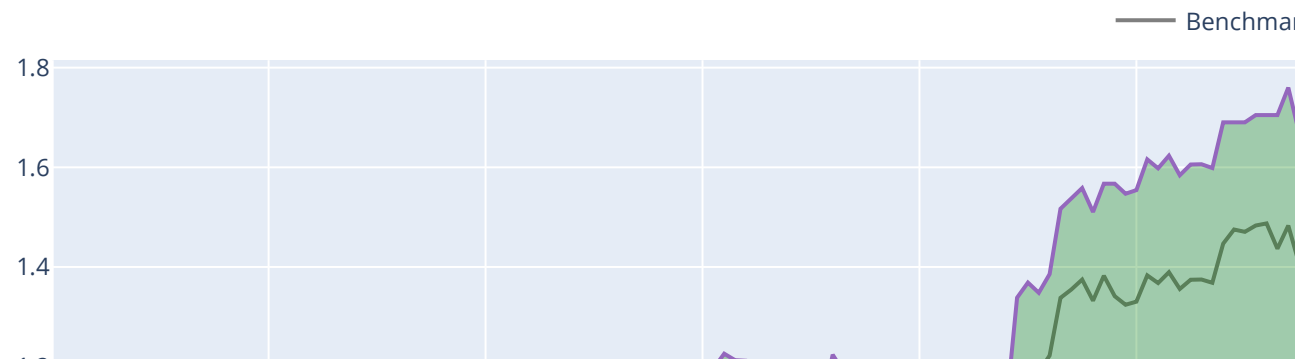
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.59e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

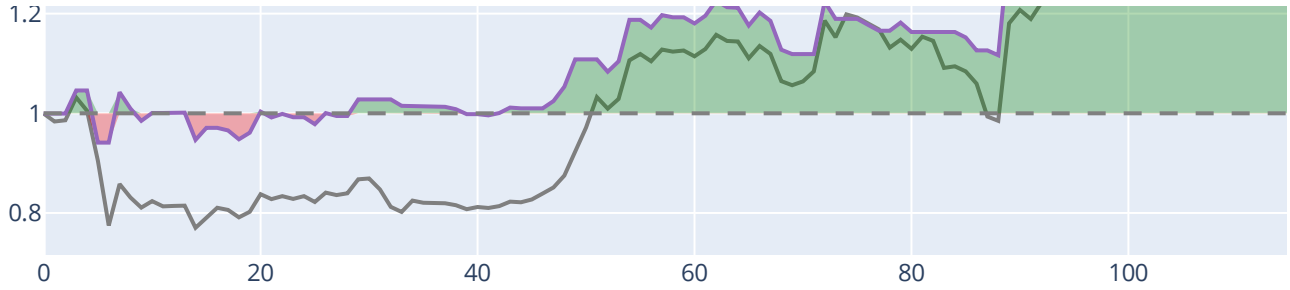
```
test_data = lagmat2ds(df_test.values, i, trim="both", dropex=1)

y = pd.Series(test_data[:, 0])
yhats = pd.Series(model.predict(add_constant(test_data[:, 1:], prepend=False)))
```

```
train_return_describe = pd.Series(model.predict()).describe()
train_return_describe
```

```
count    618.000000
mean       0.003379
std        0.020208
min       -0.077180
25%       -0.008667
50%        0.002950
75%        0.014720
max        0.096807
dtype: float64
```

```
y_model_adjusted = y.copy()
y_model_adjusted[yhats < train_return_describe['25%']] = 0.
y_model_adjusted.vbt.returns(freq='d').plot_cumulative(benchmark_rets=y).show()
```






```
y_model_adjusted.vbt.returns(freq='d').stats(settings=dict(benchmark_rets=y))
```

```
Start                                    0
End                                    118
Period                     119 days 00:00:00
Total Return [%]                   57.927419
Benchmark Return [%]               34.297639
Annualized Return [%]             306.176036
Annualized Volatility [%]          61.704038
Max Drawdown [%]                   10.266098
Max Drawdown Duration       43 days 00:00:00
Sharpe Ratio                        2.568081
Calmar Ratio                       29.823993
Omega Ratio                         1.66534
Sortino Ratio                       5.378637
Skew                                2.242645
Kurtosis                           12.329672
Tail Ratio                          1.704727
Common Sense Ratio                  6.924192
Value at Risk                      -0.030879
Alpha                               1.088425
Beta                                0.730401
dtype: object
```

REFERENCES:

1. Harsch Katara and Dheeraj Vaidya, Quartile Formula, WallStreetMojo, Quartile Formula | How to Calculate Quartile in Statistics | Example (wallstreetmojo.com)
2. Pritha Bhandari, How to find the Median, Scribbr, 02/19/2023, How to Find the Median | Definition, Examples & Calculator (scribbr.com)
3. Median - Formula, Meaning, Example | How to Find Median? (cuemath.com)
4. Quantile: Definition and How to Find Them in Easy Steps - Statistics How To
5. Avijeet Biswal, SimpleLearn, 04/03/2023, Percentile in Statistics: Overview & How to Calculate | Simplilearn
6. Jim Frost, Statistics by Jim, Interquartile Range (IQR): How to Find and Use It - Statistics By

Jim
7. Khan Academy, Calculating standard deviation step by step, Standard deviation: calculating step by step (article) | Khan Academy
8. Adam Hayes, Somer Anderson, Amanda Bellucco-Chatham, Co-efficient of Variation Meaning and How to Use It, Investopedia 09/16/2022, Co-efficient of Variation Meaning and How to Use It (investopedia.com)
9. Khan Academy; Mean, median and mode; Mean, median, and mode review (article) | Khan Academy
10. Statistics How To, Median Absolute Deviation, Median Absolute Deviation - Statistics How To
11. Ashish Kumar Srivastav, Dheeraj Vaidya, Skewness Formula, WallStreetMojo, Skewness Formula | How to Calculate Skewness? (with Examples) (wallstreetmojo.com)
12. Diksha Keni, Dheeraj Vaidya, Population Variance Formula, WallStreetMojo, Population Variance Formula | Step by Step Calculation | Examples (wallstreetmojo.com)
13. Dybvig, Philip H., and William J. Marshall. "The New Risk Management: The Good, the Bad, and the Ugly." Economic Research Federal Reserve Bank of St. Louis, vol. 79, no. 6, 1997, pp.9-21,https://research.stlouisfed.org/publications/review/1997/11/01/the-new-risk-management-the-good-the-bad-and-the-ugly. Accessed 30 April 2023.
14. Partial Auto Correlation Function Formula
15. Granger_causality
16. Dickey–Fuller test
17. Augmented Dickey–Fuller test
18. Leslie, J. R., Stephens, M. A., and Fotopoulos, S. (1986). Asymptotic distribution of the Shapiro–Wilk W for testing for normality. Ann.Statist. 14, 1497–1506

Colab paid products - Cancel contracts here