RELATED ARTICLES

LangChain for Equity Inv

Stock Market Data: Obtaining Data, Visualization & Analysis in Python

Automated Trading | Aug 30, 2024 | 13 min ead

By Chainika Thakar (Originally written by Ishan Shah)

Success in the trading journey requires the trader to know the key concepts before starting trading and one of them is mastering the stock market data analysis. For conducting the data analysis, the trader first needs

RELATED ARTICLES

LangChain for Equity Inv

to fetch the data and visualise it for the "identification of historical price trends and patterns".

You must be wondering **"What is the benefit of this identification"**?

The answer is that forecasting future price movements becomes possible with this analysis of historical movements in price. For instance, an analysis of the historical performance of S&P 500 stock tickers can be done to predict future movements of the same. If you are looking to fetch the stock market data and analyse the historical data in Python, you have come to the right place.

After reading this blog, you will be able to:

- Get historical data for stocks
- Plot the data and analyse the performance
- Get the fundamental, futures and options data

For easy navigation through the blog, we have mentioned below what this blog covers, and that is:

Importance and techniques of data analysis in stock trading

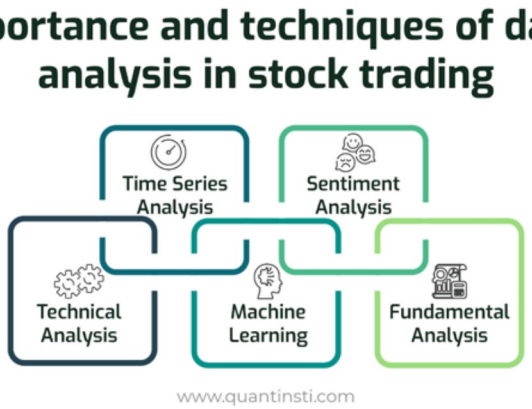
Data analysis is fundamental to stock trading as it transforms previous market data into actionable insights for the future.

Through rigorous analysis, traders can identify historical patterns, forecast future price movements, and make informed decisions. It helps in understanding market trends, volatility, and potential risks, thereby enhancing the ability to devise robust trading strategies.

Here are some key techniques:

RELATED ARTICLES

Lang Chain for Equity Inv



- **Technical Analysis:** Uses historical price and volume data to identify patterns and trends, helping to forecast future price movements.
- **Time Series Analysis:** Analyses stock price data over time to identify trends, cycles, and seasonal effects, providing insights into future performance.
- **Machine Learning:** Employs algorithms to model and predict stock prices based on historical data, improving the accuracy of predictions.
- **Sentiment Analysis:** Gauges market sentiment by analysing news articles, social media, and other sources, offering insights into market psychology.
- **Fundamental Analysis:** Examines a company's financial statements, health, and industry position to determine its intrinsic value and potential for future growth.

Effective data analysis reduces emotional bias and enhances precision, leading to improved trading performance and gainful returns. In an era driven by

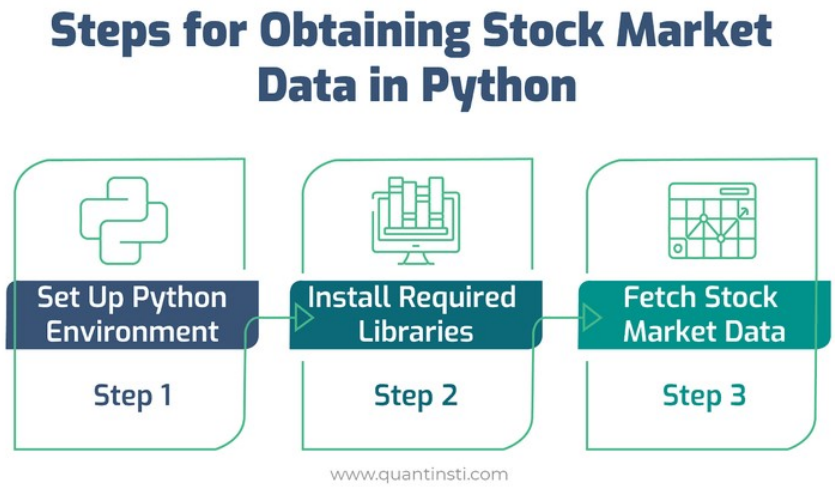
RELATED ARTICLES

LangChain for Equity Inv

vast amounts of data, leveraging analytical tools is indispensable for gaining a competitive edge in stock trading.

Let us now see the steps for obtaining the stock market data.

Steps for obtaining stock market data in Python



Step 1: Set Up Python Environment: Ensure Python is installed on your system. Create a virtual environment using Anaconda or virtualenv to isolate project dependencies and maintain a clean workspace.

Step 2: Install Required Libraries: Use pip or conda

RELATED ARTICLES

LangChain for Equity Inv

to install essential libraries such as Pandas, NumPy, and yfinance. These libraries will help in data manipulation, numerical operations, and fetching stock market data.

Step 3: Fetch Stock Market Data: Utilise the yfinance library to download historical market data. This can be done using the yf.download() function, specifying the stock ticker, start and end dates, and data interval.

Now, we will discuss how to fetch the stock market data in Python by installing and importing the libraries.

How to fetch stock market data in Python?

Yahoo Finance

One of the first sources from which you can get historical daily price-volume stock market data is Yahoo finance. You can use pandas_datareader or yfinance module to get the data and then can download or store it in a CSV file by using pandas.to_csv method. Additionally, you can explore Yahoo Finance futures data to analyze market trends

and incorporate it into your trading strategies.

If yfinance is not installed on your computer, then run the below line of code from your Jupyter Notebook to install yfinance.

```
!pip install yfinance
```

Output:

```
Collecting yfinance==0.1.63
  Downloading yfinance-0.1.63.tar.gz (26 kB)
Requirement already satisfied: pandas>=0.24 in c:\users\academy\miniconda3\lib\site-packages (from yfinance==0.1.63) (1.2.4)
Requirement already satisfied: numpy>=1.15 in c:\users\academy\miniconda3\lib\site-packages (from yfinance==0.1.63) (1.20.3)
Requirement already satisfied: requests>=2.20 in c:\users\academy\miniconda3\lib\site-packages (from yfinance==0.1.63) (2.25.0)
Requirement already satisfied: multitasking>=0.0.7 in c:\users\academy\miniconda3\lib\site-packages (from yfinance==0.1.63) (0.
0.9)
Requirement already satisfied: lxml>=4.5.1 in c:\users\academy\miniconda3\lib\site-packages (from yfinance==0.1.63) (4.6.3)
Requirement already satisfied: numpy>=1.15 in c:\users\academy\miniconda3\lib\site-packages (from yfinance==0.1.63) (1.20.3)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\academy\miniconda3\lib\site-packages (from pandas>=0.24->yfin
ance==0.1.63) (2.8.1)
Requirement already satisfied: pytz>=2017.3 in c:\users\academy\miniconda3\lib\site-packages (from pandas>=0.24->yfinance==0.1.
63) (2021.1)
Requirement already satisfied: six>=1.5 in c:\users\academy\miniconda3\lib\site-packages (from python-dateutil>=2.7.3->pandas>=
0.24->yfinance==0.1.63) (1.15.0)
Requirement already satisfied: chardet<4,>=3.0.2 in c:\users\academy\miniconda3\lib\site-packages (from requests>=2.20->yfinanc
e==0.1.63) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\academy\miniconda3\lib\site-packages (from requests>=2.20->yfinan
ce==0.1.63) (2021.5.30)
Requirement already satisfied: idna<3,>=2.5 in c:\users\academy\miniconda3\lib\site-packages (from requests>=2.20->yfinance==0.
1.63) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\academy\miniconda3\lib\site-packages (from requests>=2.20->yfi
nance==0.1.63) (1.25.11)
Building wheels for collected packages: yfinance
  Building wheel for yfinance (setup.py): started
  Building wheel for yfinance (setup.py): finished with status 'done'
  Created wheel for yfinance: filename=yfinance-0.1.63-py2.py3-none-any.whl size=23908 sha256=091b3fb3632e1d5221ca8707446548c8f
bb9cfdb03dbc9014ec2379234ef8d77
  Stored in directory: c:\users\academy\appdata\local\pip\cache\wheels\44\19\52\4db92c8786b13e717b9664529da13bf8d7b74aff580ca64
fc0
Successfully built yfinance
Installing collected packages: yfinance
  Attempting uninstall: yfinance
    Found existing installation: yfinance 0.1.59
    Uninstalling yfinance-0.1.59:
      Successfully uninstalled yfinance-0.1.59
Successfully installed yfinance-0.1.63
```

```
# Import yfinance package
import yfinance as yf

# Set the start and end date
start_date = '1990-01-01'
end_date = '2024-08-1'

# Set the ticker
ticker = 'AMZN'

# Get the data
data = yf.download(ticker, start_date, end_date)

# Print 5 rows
```

```
data.tail()
```

AMZN_data.py hosted with ❤ by GitHub view raw

Output:

```
[**********100%%**********
```

	Open	High	Lo
Date			
2024-07-25	182.910004	183.899994	176.80
2024-07-26	180.389999	183.190002	180.24
2024-07-29	183.839996	184.750000	182.38
2024-07-30	184.720001	185.860001	179.38
2024-07-31	185.050003	187.940002	184.46

To visualise the adjusted close price data, you can use the matplotlib library and plot method as shown below.

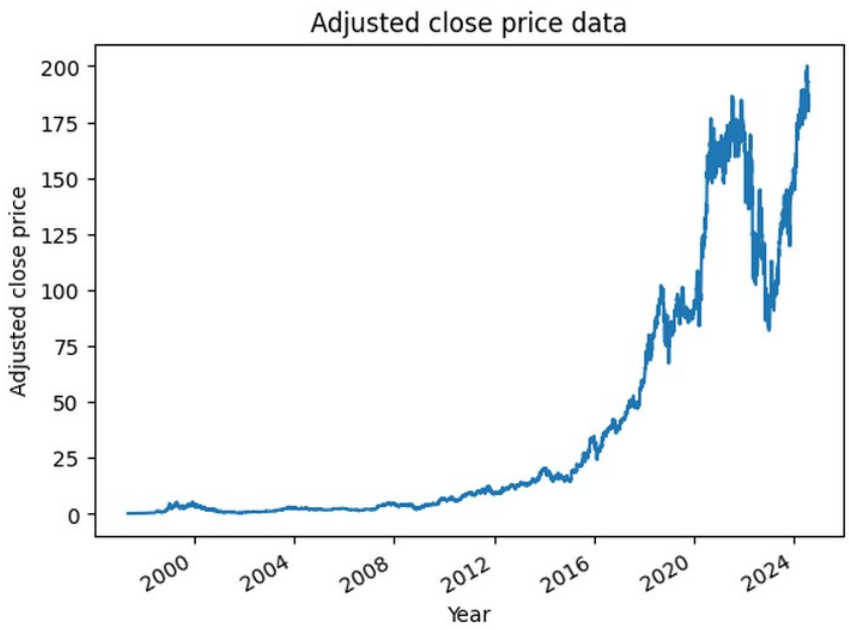
```
# Import matplotlib for plotting
import matplotlib.pyplot as plt
%matplotlib inline

# Plot adjusted close price data
data['Adj Close'].plot()
plt.xlabel('Year')
plt.ylabel('Adjusted close price')
plt.title('Adjusted close price data')
plt.show()
```

RELATED ARTICLES

LangChain for Equity Inv

Visualise.py hosted with ❤ by GitHub view raw

Output:



Data Source: Yahoo Finance

Let us improve the plot by resizing, giving appropriate labels and adding grid lines for better readability.

```
# Plot the adjusted close price
data['Adj Close'].plot(figsize=(10, 7))


# Define the label for the title of the figure
plt.title("Adjusted Close Price of %s" % ticker, fontsi


# Define the labels for the x-axis and y-axis
plt.ylabel('Price', fontsize=14)
plt.xlabel('Year', fontsize=14)

```

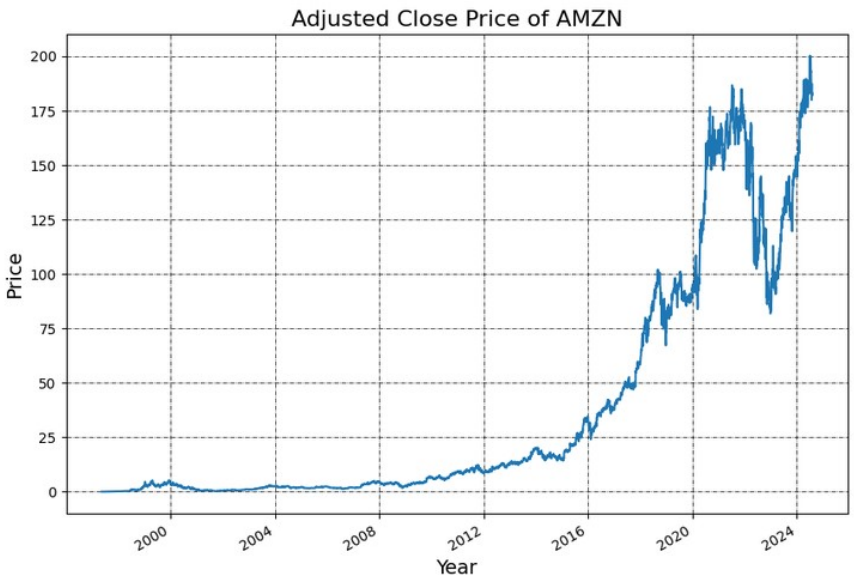
RELATED ARTICLES

LangChain for Equity Inv

```
# Plot the grid lines
plt.grid(which="major", color='k', linestyle='-.', linew

# Show the plot
plt.show()
```

Adj_close_AMZN.py hosted with ❤ by GitHub view raw

Output:



Data Source: Yahoo Finance

Advantages of Yahoo Finance

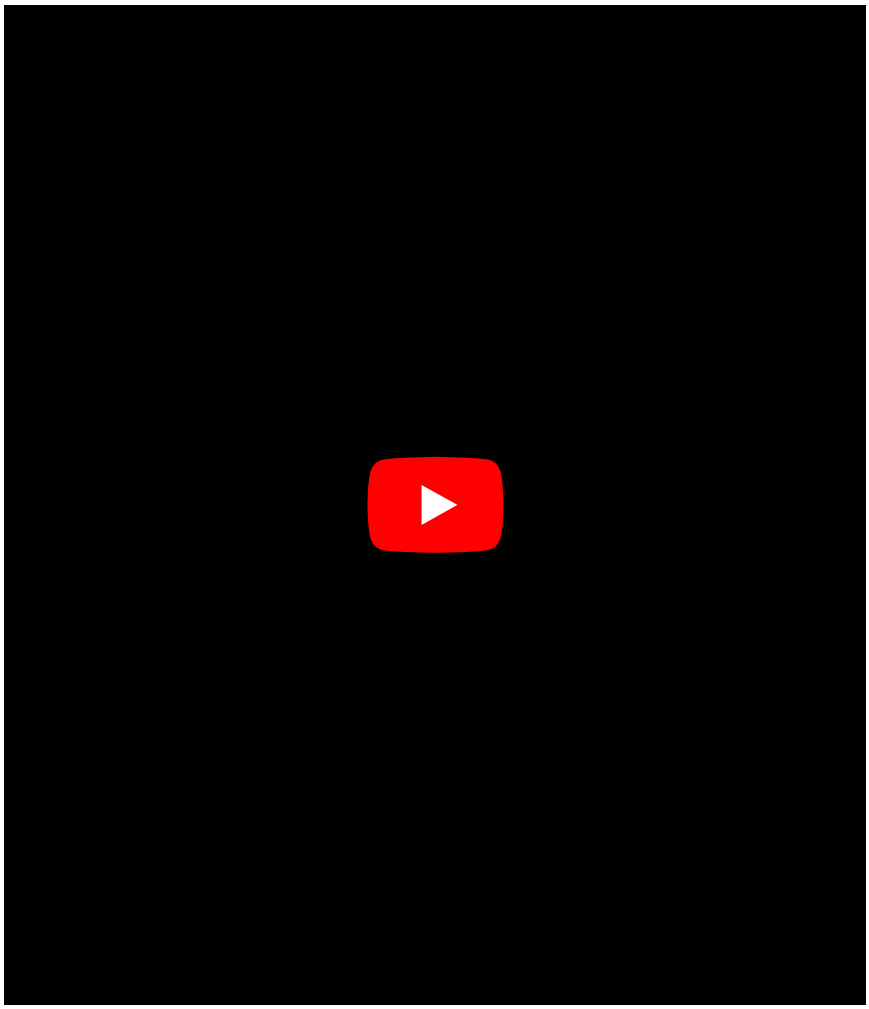
1. Adjusted close price stock market data is available
2. Most recent stock market data is available
3. Doesn't require an API key to fetch the stock market data

Below is an interesting video by Nitesh Khandelwal (Co-Founder and CEO, of QuantInsti) that answers all

RELATED ARTICLES

LangChain for Equity Inv

your questions related to getting Data for Algo Trading.



Now we will discuss how we can get the stock market data for various geographies.

How to get stock market data for different geographies?

To get stock market data for different geographies,

search the ticker symbol on Yahoo finance and use that as the ticker.

To get the stock market data of multiple stock tickers, you can create a list of tickers and call the yfinance download method for each stock ticker.

For simplicity, I have created a dataframe data to store the adjusted close price of the stocks.

```
# Import packages
import yfinance as yf
import pandas as pd

# Set the start and end date
start_date = '1990-01-01'
end_date = '2024-08-1'

# Define the ticker list
tickers_list = ['AAPL', 'IBM', 'MSFT', 'WMT']

# Create placeholder for data
data = pd.DataFrame(columns=tickers_list)

# Fetch the data
for ticker in tickers_list:
    data[ticker] = yf.download(ticker,
                               start_date,
                               end_date)['Adj Close']

# Print first 5 rows of the data
data.tail()
```

Stock_data_geographies.py hosted with ❤ by GitHub view raw

Output:

RELATED ARTICLES

LangChain for Equity Inv

```
[**********************100%%**********************
[**********************100%%**********************
[**********************100%%**********************
[**********************100%%**********************
```

	AAPL	IBM	MS
Date			
2024-07-25	217.490005	191.979996	418.39
2024-07-26	217.960007	191.750000	425.26
2024-07-29	218.240005	191.500000	426.73
2024-07-30	218.800003	191.039993	422.92
2024-07-31	222.080002	192.139999	418.35

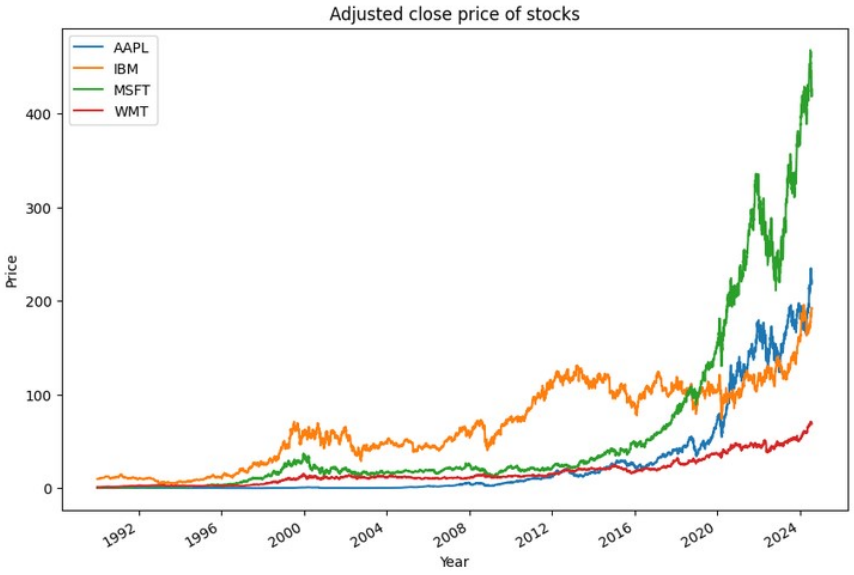
```
1  import matplotlib.pyplot as plt
2
3  data.plot(figsize=(10, 7))
4
5  # Add labels and title
6  plt.xlabel('Year')
7  plt.ylabel('Price')
8  plt.title('Adjusted close price of stocks')
9
10 # Show the plot
11 plt.show()
```

Adj_close_price.py hosted with ❤ by GitHub view raw

Output:

RELATED ARTICLES

LangChain for Equity Inv



Data Source: Yahoo Finance

Let us now check the real life example of stock market data fetching as well as the analysis.

Real-life example of stock market data fetching and analysis in Python

If you want to analyse the stock market data for all the stocks which make up S&P 500 then the below code will help you. It gets the list of stocks from the Wikipedia page and then fetches the stock market data from yahoo finance.

```
# Import packages
import yfinance as yf
```

RELATED ARTICLES

LangChain for Equity Inv

```
import pandas as pd

# Read and print the stock tickers that make up S&P500
tickers = pd.read_html(
    'https://en.wikipedia.org/wiki/List_of_S%26P_500_comp
print(tickers.head())
```

S&P500.py hosted with ❤ by GitHub view raw

Output:

```
  Symbol     Security             GICS Sector
0    MMM           3M             Industrials
1    AOS  A. O. Smith             Industrials
2    ABT       Abbott             Health Care
3   ABBV       AbbVie             Health Care
4    ACN    Accenture  Information Technology

     Headquarters Location  Date added      C
0    Saint Paul, Minnesota  1957-03-04    667
1     Milwaukee, Wisconsin  2017-07-26    911
2  North Chicago, Illinois  1957-03-04     18
3  North Chicago, Illinois  2012-12-31  15511
4          Dublin, Ireland  2011-07-06  14673
```

```
# Get the data for these tickers from yahoo finance
data = yf.download(tickers.Symbol.to_list(),'2021-1-1','
print(data.head())
```

Data_yfinance.py hosted with ❤ by GitHub view raw

Output:

```
Ticker                 A    AAL        AAPL
Date
```

```
2021-01-04  115.980736  15.13  126.830078  90
2021-01-05  116.928986  15.43  128.398163  91
2021-01-06  120.135468  15.52  124.076103  90
2021-01-07  123.332176  15.38  128.309967  91
2021-01-08  124.212006  15.13  129.417419  92

Ticker            ACGL          ACN         ADBE
Date
2021-01-04  34.900002  243.104004  485.339996
2021-01-05  35.040001  244.488007  485.690002
2021-01-06  36.580002  247.161118  466.309998
2021-01-07  36.240002  249.493027  477.739990
2021-01-08  36.439999  250.403015  485.100006
```

Intraday or minute frequency stock data

The below code fetches the stock market data for MSFT for the past 5 days of 1-minute frequency.

```
import yfinance as yf
intraday_data = yf.download(tickers="MSFT",
                            period="5d",
                            interval="1m",
                            auto_adjust=True)
intraday_data.head()
```

Intraday_data.py hosted with ❤ by GitHub view raw

Output:

```
[*********************100%%**********************
```

RELATED ARTICLES

LangChain for Equity Inv

	Open	High	
Datetime			
2024-08-02 09:30:00-04:00	412.744995	413.940002	41
2024-08-02 09:31:00-04:00	413.000000	413.399994	41
2024-08-02 09:32:00-04:00	413.500000	414.714996	41
2024-08-02 09:33:00-04:00	414.600006	415.000000	41
2024-08-02 09:34:00-04:00	412.899994	413.149994	41

Resample stock data

Convert 1-minute data to 1-hour data or resample stock data

During strategy modelling, you might be required to work with a custom frequency of stock market data such as 15 minutes or 1 hour or even 1 month.

If you have minute level data, then you can easily construct the 15 minutes, 1 hour or daily candles by resampling them. Thus, you don't have to buy them

RELATED ARTICLES

LangChain for Equity Inv

separately.

In this case, you can use the pandas resample method to convert the stock market data to the frequency of your choice. The implementation of these is shown below where a 1-minute frequency data is converted to 10-minute frequency data.

The first step is to define the dictionary with the conversion logic. For example, to get the open value the first value will be used, to get the high value the maximum value will be used and so on.

The name Open, High, Low, Close and Volume should match the column names in your dataframe.

```
ohlcv_dict = {
'Open': 'first',
'High': 'max',
'Low': 'min',
'Close': 'last',
'Volume': 'sum'
}
```

Define_dictionary.py hosted with ❤ by GitHub view raw

Convert the index to datetime timestamp as by default string is returned. Then call the resample method with the frequency such as:

- 10T for 10 minutes,
- D for 1 day and

- M for 1 month

```
# Import package & get the data
import yfinance as yf
intraday_data = yf.download(tickers="MSFT",
                            period="5d",
                            interval="1m",
                            auto_adjust=True)

# Define the resampling logic
ohlcv_dict = {
     'Open': 'first',
     'High': 'max',
     'Low': 'min',
     'Close': 'last',
     'Volume': 'sum'
}

# Resample the data
intraday_data_10 = intraday_data.resample('10T').agg(oh
intraday_data_10.head()
```

Resample_data.py hosted with ❤ by GitHub view raw

Output:

```
[*********************100%%**********************
```

	Open	High	
Datetime			
2024-08-02 09:30:00-04:00	412.744995	415.000000	41
2024-08-02	411.589996	413.434998	40

RELATED ARTICLES

LangChain for Equity Inv

RELATED ARTICLES

LangChain for Equity Inv

	Open	High	
Datetime			
09:40:00-04:00			
2024-08-02 09:50:00-04:00	408.000000	408.695007	40
2024-08-02 10:00:00-04:00	407.570007	408.000000	40
2024-08-02 10:10:00-04:00	405.859985	406.970001	40

Suggested read:

- How to Get Historical Market Data Through Python API

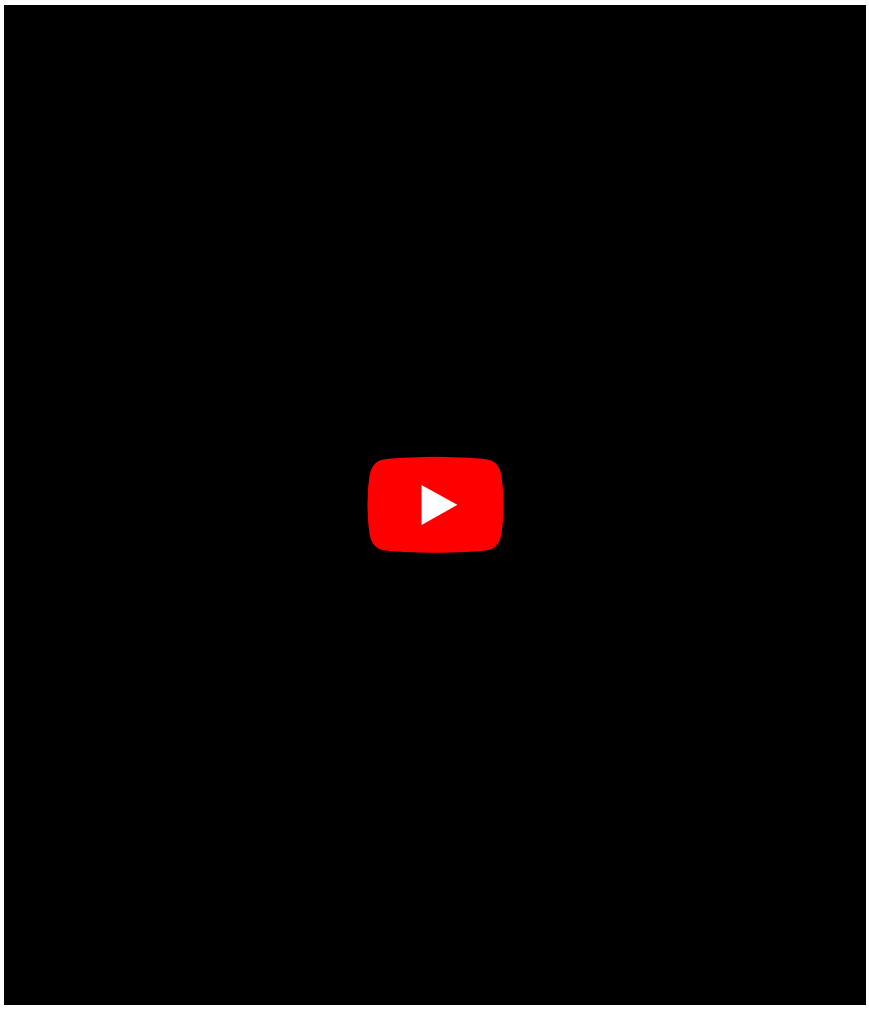
Fundamental data

We have used yfinance to get the fundamental data.

Below is a video that covers fundamental data analysis in detail.

RELATED ARTICLES

LangChain for Equity Inv



The first step is to set the ticker and then call the appropriate properties to get the right stock market data.

If yfinance is not installed on your computer, then run the below line of code from your Jupyter Notebook to install yfinance.

```
!pip install yfinance
# Import yfinance
import yfinance as yf

# Set the ticker as MSFT
msft = yf.Ticker("MSFT")
```

MSFT_ticker.py hosted with ❤ by GitHub view raw

Key Ratios

You can fetch the latest price to book ratio and price to earnings ratio as shown below.

```
# fetch the latest price to book ratio and price to earn

# Get the latest price to book ratio
pb_ratio = msft.info['priceToBook']
print('Price to Book Ratio is: ', pb_ratio)
# Get the latest price to earnings ratio
pe_ratio = msft.info['trailingPE']
print('Price to Earnings Ratio is: ', pe_ratio)
```

Ratios.py hosted with ❤ by GitHub view raw

Output:

```
Price to Book Ratio is:  11.540634
Price to Earnings Ratio is:  35.321186
```

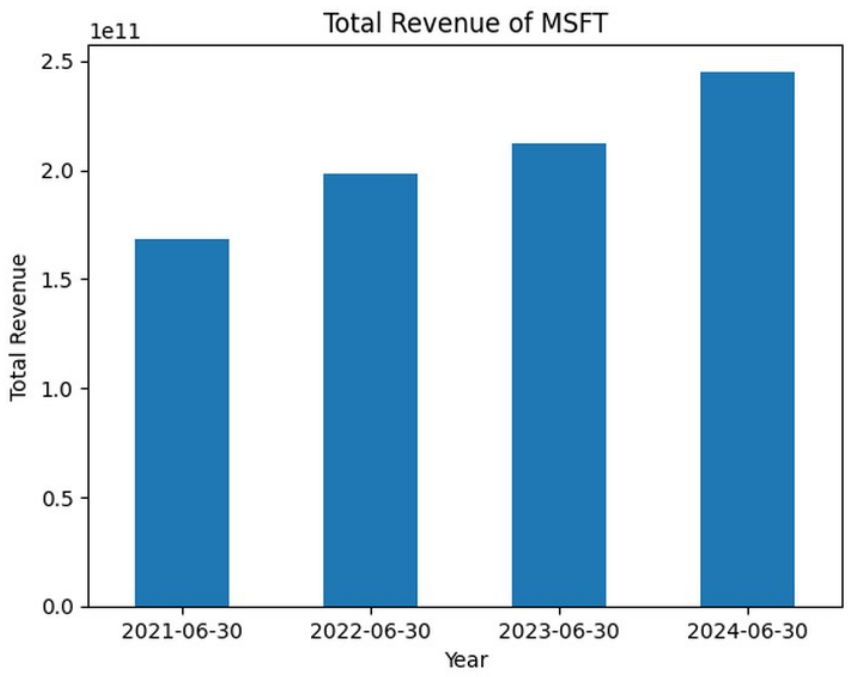
Revenues

```
import matplotlib.pyplot as plt
import yfinance as yf

# Set the ticker as MSFT
msft = yf.Ticker("MSFT")

# Get the financials data and sort the columns (dates)
financials = msft.financials.sort_index(axis=1)

# Select the 'Total Revenue' row
```

```
total_revenue = financials.loc['Total Revenue']

# Plot the total revenue as a bar chart
total_revenue.plot.bar()

# Format the x-axis to show dates without time
plt.xticks(range(len(total_revenue.index)), total_reven
plt.xlabel('Year')
plt.ylabel('Total Revenue')
plt.title('Total Revenue of MSFT')

# Show the plot
plt.show()
```

Total_revenue_MSFT.py hosted with ❤ by GitHub view raw

Output:



Data Source: Yahoo Finance

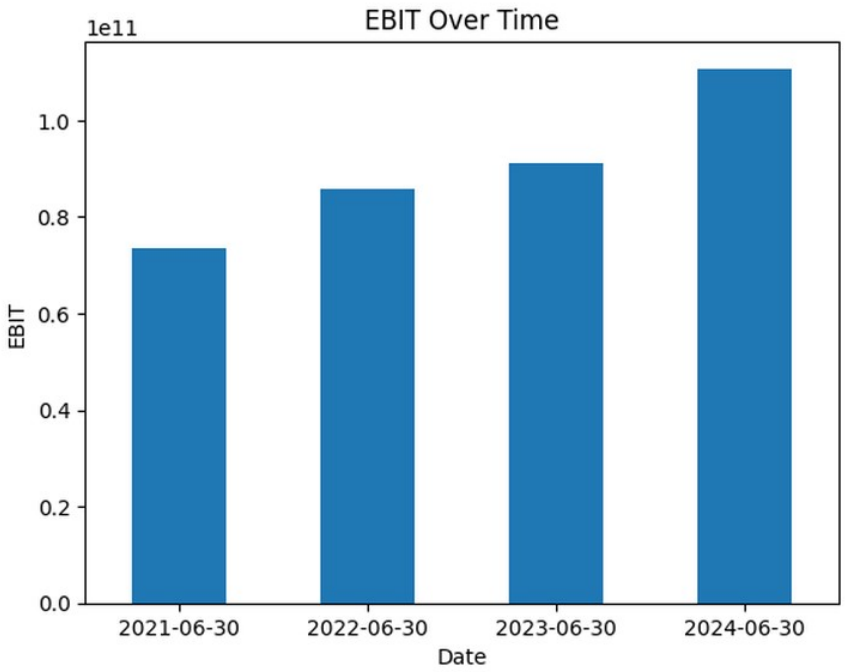
Earnings Before Interest and Taxes

(EBIT)

```
# Assuming EBIT is a pandas Series with a DateTimeIndex
EBIT_sorted = EBIT.sort_values(ascending=True)

# Plotting the sorted EBIT
EBIT_sorted.plot.bar()
plt.xticks(range(len(EBIT_sorted.index)), EBIT_sorted.i
plt.xlabel('Date')
plt.ylabel('EBIT')
plt.title('EBIT Over Time')
plt.show()
```

EBIT.py hosted with ❤ by GitHub view raw

Output:



Data Source: Yahoo Finance

Balance sheet, cash flows and other information

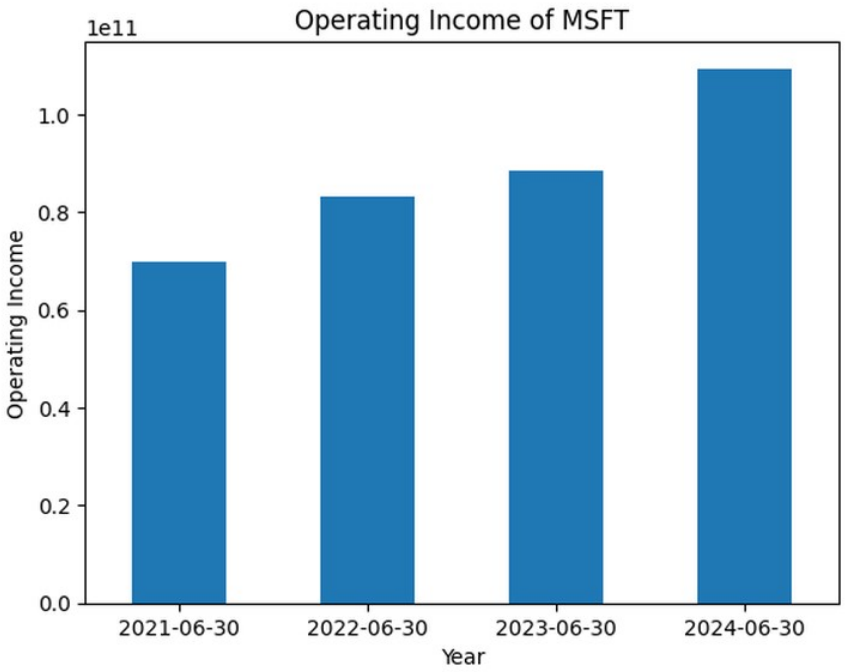
RELATED ARTICLES

LangChain for Equity Inv

```python
# Checking Operating income or EBIT (Earnings before int
import yfinance as yf

# Set the ticker as MSFT
msft = yf.Ticker("MSFT")

# Get the financials data
financials = msft.financials

# Display the available rows in the financials DataFrame
print(financials.index)
import matplotlib.pyplot as plt
import yfinance as yf

# Set the ticker as MSFT
msft = yf.Ticker("MSFT")

# Get the financials data and sort the columns (dates) :
financials = msft.financials.sort_index(axis=1)

# Check if 'Operating Income' is available
if 'Operating Income' in financials.index:
    # Select the 'Operating Income' row
    operating_income = financials.loc['Operating Income

    # Plot Operating Income as a bar chart (acting as a
    operating_income.plot.bar()

    # Format the x-axis to show dates without time
    plt.xticks(range(len(operating_income.index)), opera
    plt.xlabel('Year')
    plt.ylabel('Operating Income')
    plt.title('Operating Income of MSFT')
    # Show the plot
    plt.show()
else:
    print("Operating Income data is not available.")
```

Operating_income_MSFT.py hosted with ❤ by GitHub view raw

Output:



Stock market data analysis

After you have the stock market data, the next step is to create trading strategies and analyse the performance. The ease of analysing the performance is the key advantage of Python.

We will analyse the cumulative returns and different ratios such as

- Sharpe ratio,
- Sortino ratio, and
- Calmar ratio.

I have created a simple buy-and-hold strategy for illustration purposes with four stocks namely:

RELATED ARTICLES

LangChain for Equity Inv

- Apple
- Amazon
- Microsoft
- Walmart

Install pyfolio if not already installed, as shown in the first line of the code, following which we will analyse the performance using the pyfolio tear sheet.

```
# If you already have a version of pyfolio on your syste
!pip install pyfolio-reloaded==0.9.5
# Define the ticker list
tickers_list = ['AAPL', 'AMZN', 'MSFT', 'WMT']

# Import pandas and create a placeholder for the data
import pandas as pd
data = pd.DataFrame(columns=tickers_list)

# Fetch the data
import yfinance as yf
for ticker in tickers_list:
    data[ticker] = yf.download(ticker, period='5y',)['

# Compute the returns of individual stocks and then com
# The mean return is the daily portfolio returns with t
data = data.pct_change().dropna().mean(axis=1)

# Import Pyfolio
import pyfolio as pf

# Get the full tear sheet
pf.create_simple_tear_sheet(data)
```

Tear_sheet.py hosted with ❤ by GitHub view raw

Output:

RELATED ARTICLES

LangChain for Equity Inv

Start date	2019-08-09
End date	2024-08-07
Total months	59

	Backtest
Annual return	23.101%
Cumulative returns	181.993%
Annual volatility	24.33%
Sharpe ratio	0.98
Calmar ratio	0.79
Stability	0.74
Max drawdown	-29.312%
Omega ratio	1.19
Sortino ratio	1.42
Skew	-0.07
Kurtosis	6.07
Tail ratio	0.96
Daily value at risk	-2.971%

Above in the output you can see the tear sheet which includes all the analysed backtested figures for a total of 59 months.

It is important to note that backtesting results do not guarantee future performance. The presented strategy results are intended solely for educational purposes and should not be interpreted as investment advice. A comprehensive evaluation of the strategy across multiple parameters is necessary to assess its effectiveness.

Now we will see the various techniques used for data visualisation for you to be able to use any one.

Data visualisation techniques

Data visualisation techniques help interpret and communicate insights from stock market data. Here are some common techniques and their uses:

1. Line Charts: Line charts plot stock prices over time, showing trends and patterns. They are ideal for visualising price movements and historical performance.

RELATED ARTICLES

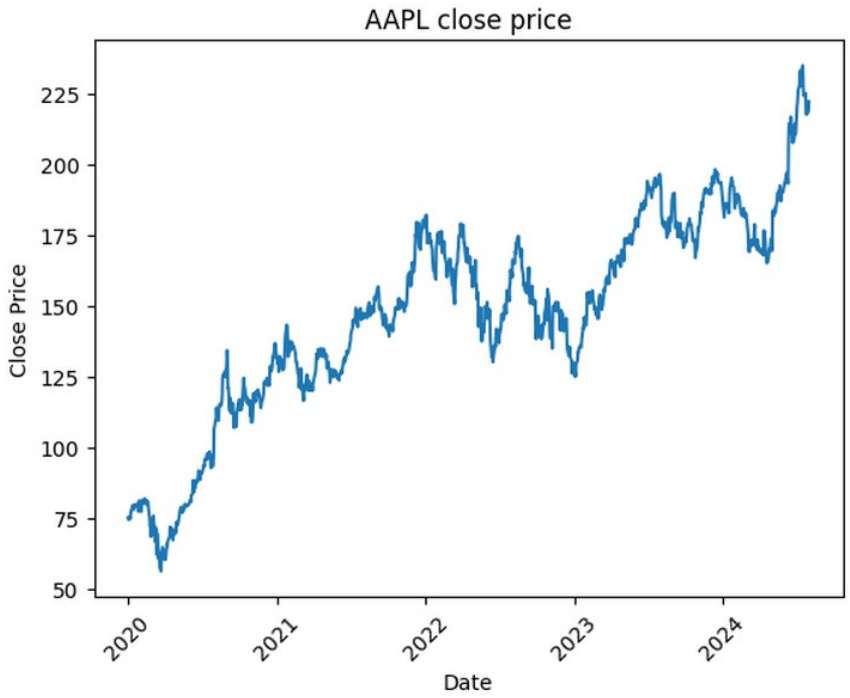
LangChain for Equity Inv

Code Example:

```python
import yfinance as yf
import matplotlib.pyplot as plt

# Download historical stock data for Apple (AAPL)
stock_data = yf.download('AAPL', start='2020-01-01', en

# Plotting the closing price over time
plt.plot(stock_data.index, stock_data['Close'])
plt.title('AAPL close price')
plt.xlabel('Date')
plt.ylabel('Close Price')
# Rotate x-axis labels
plt.xticks(rotation=45)
plt.show()
```

AAPL_close_price.py hosted with ❤ by GitHub view raw

Output:



The above plot shows the line chart displaying close

RELATED ARTICLES

LangChain for Equity Inv

price of AAPL over a period of time.

2. Candlestick Charts: Candlestick charts display the open, high, low, and close prices for a given period, revealing market sentiment and trends. They are commonly used for technical analysis.

Code Example:

```
import yfinance as yf
import plotly.graph_objects as go

# Download historical stock data for Apple (AAPL)
stock_data = yf.download('AAPL', start='2020-01-01', en

# Plotting the candlestick chart with Plotly
fig = go.Figure(data=[go.Candlestick(x=stock_data.index
                                     open=stock_data['O
                                     high=stock_data['H
                                     low=stock_data['Lo
                                     close=stock_data['
fig.update_layout(title='AAPL Candlestick Chart', xaxis

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

AAPL_candlestick_chart.py hosted with ❤ by GitHub view raw

Output:

Above plot shows a candlestick chart using Plotly for the specified date range and a line chart below the for the closing prices.

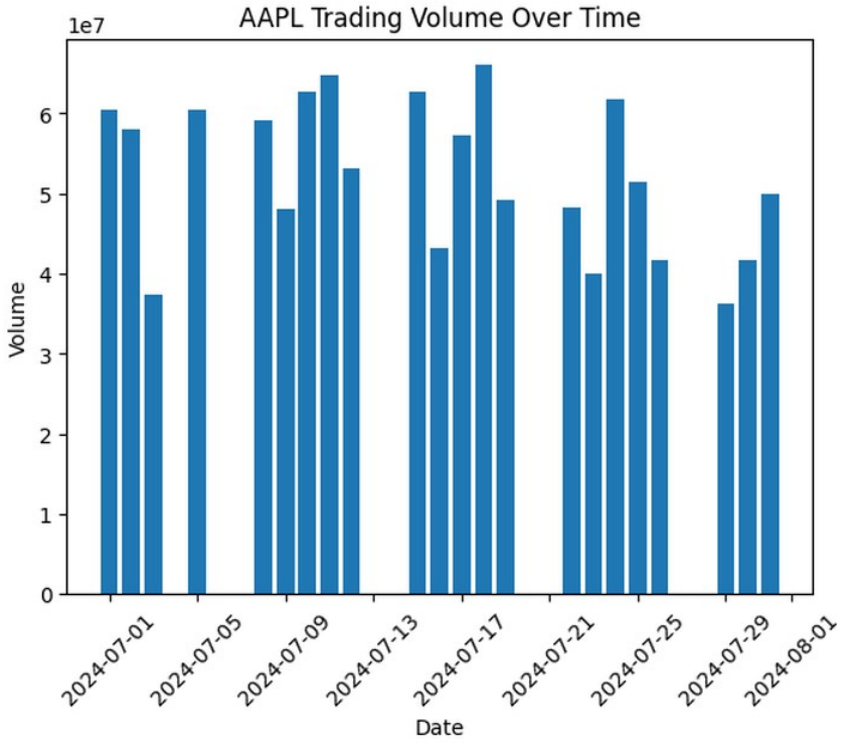
3. Bar Charts: Bar charts compare different stock metrics such as trading volume or price changes. They are useful for visualising discrete data points.

Code Example:

```
import yfinance as yf
import matplotlib.pyplot as plt

# Download historical stock data for Apple (AAPL)
stock_data = yf.download('AAPL', start='2024-07-01', en

# Plotting the trading volume over time with a bar chart
plt.bar(stock_data.index, stock_data['Volume'])
plt.title('AAPL Trading Volume Over Time')
plt.xlabel('Date')
plt.ylabel('Volume')
plt.xticks(rotation=45)  # Rotate x-axis labels for bet
plt.show()
```

Bar_chart.py hosted with ❤ by GitHub view raw

Output:



Above plot is a bar chart displaying the trading volume for Apple Inc. over the specified date range.

4. Histogram: Histograms show the distribution of stock returns or other numerical data. They help understand the frequency distribution of returns.

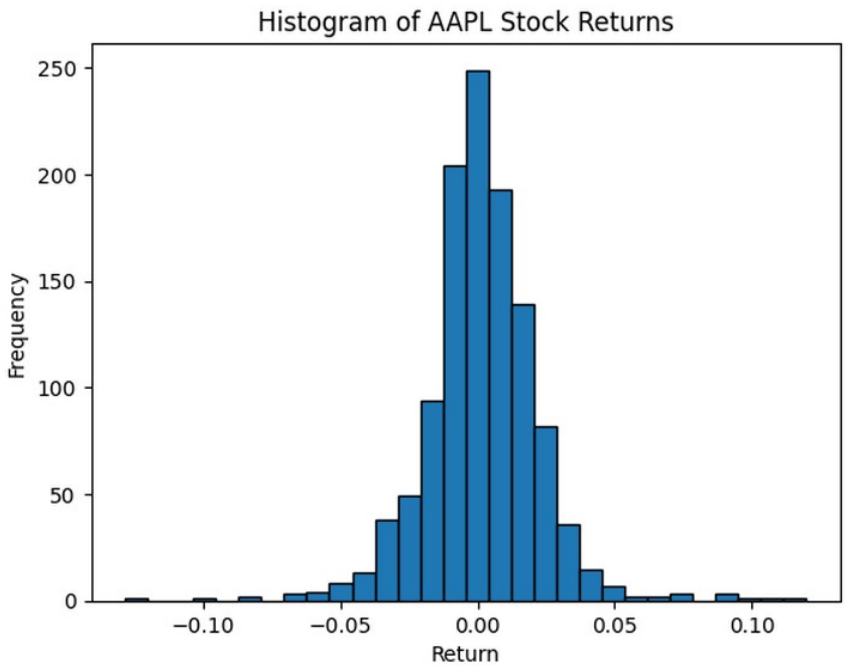
Code Example:

```
import yfinance as yf
import matplotlib.pyplot as plt

# Download historical stock data for Apple (AAPL)
stock_data = yf.download('AAPL', start='2020-01-01', en
```

```
stock_data['Return'] = stock_data['Close'].pct_change()

# Drop NaN values that result from the percentage change
stock_data = stock_data.dropna()

# Plotting the histogram of stock returns
plt.hist(stock_data['Return'], bins=30, edgecolor='black')
plt.title('Histogram of AAPL Stock Returns')
plt.xlabel('Return')
plt.ylabel('Frequency')
plt.show()
```

Histogram.py hosted with ❤ by GitHub view raw

Output:



The above histogram shows the distribution of daily returns for Apple Inc. over the specified period.

5. Scatter Plots: Scatter plots visualise the relationship between two variables, such as stock

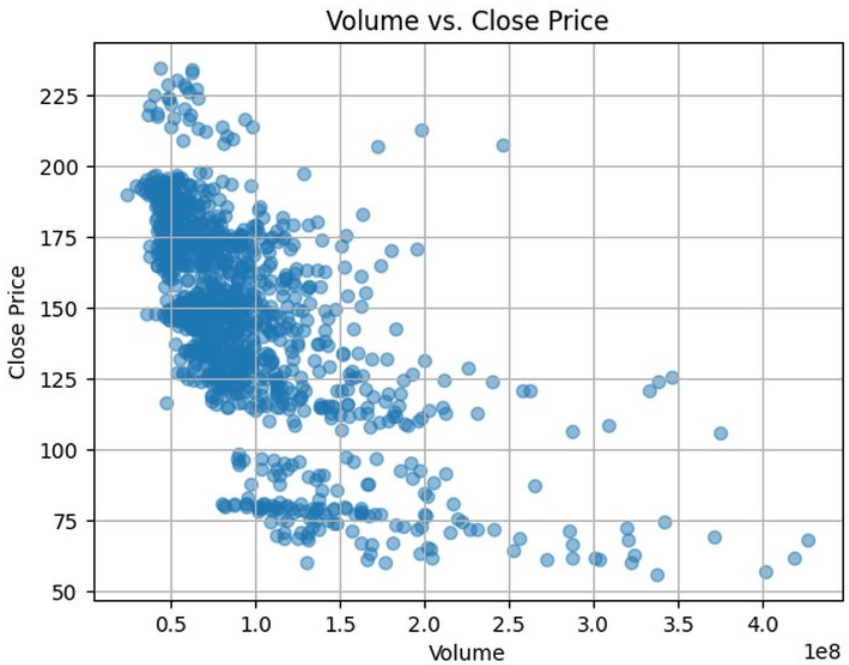
price and trading volume, helping to identify correlations.

Code Example:

```
import yfinance as yf
import matplotlib.pyplot as plt

# Download historical stock data for Apple (AAPL)
stock_data = yf.download('AAPL', start='2020-01-01', en

# Scatter plot of Volume vs. Close Price
plt.scatter(stock_data['Volume'], stock_data['Close'],
plt.title('Volume vs. Close Price')
plt.xlabel('Volume')
plt.ylabel('Close Price')
plt.grid(True)
plt.show()
```

Scatter_plots.py hosted with ❤ by GitHub view raw

Output:



RELATED ARTICLES

LangChain for Equity Inv

RELATED ARTICLES

LangChain for Equity Inv

The above scatter plot shows the relationship between the trading volume and the closing price of the stock. Each point on the scatter plot represents a single trading day's volume and closing price.

6. Heatmaps: Heatmaps display data intensity through colour variations, useful for visualising correlations between different stocks or metrics.

Code Example:

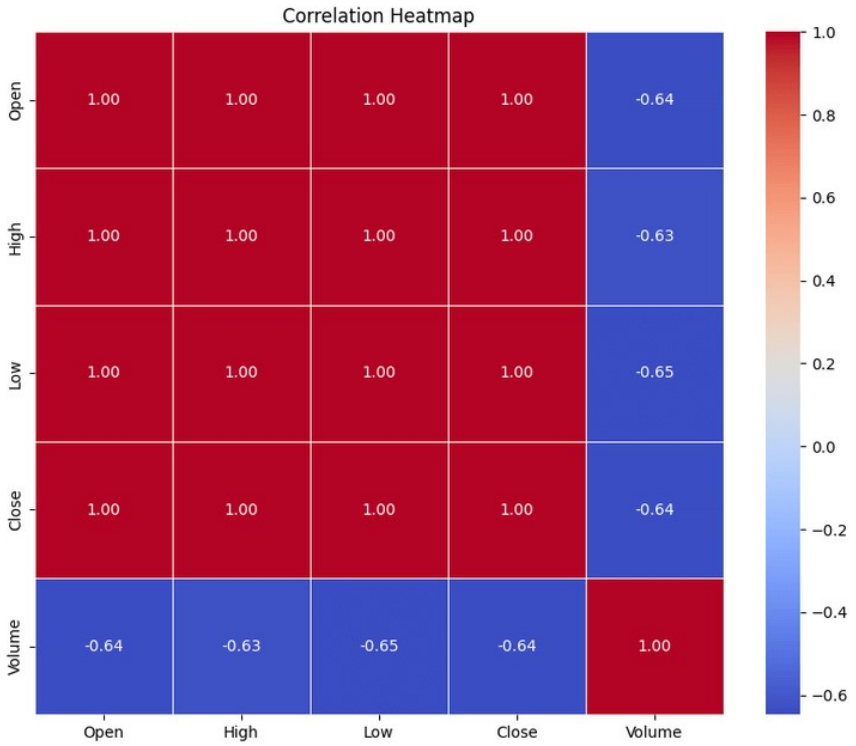
```
import yfinance as yf
import seaborn as sns
import matplotlib.pyplot as plt

# Download historical stock data for Apple (AAPL)
stock_data = yf.download('AAPL', start='2020-01-01', en

# Calculate the correlation matrix
correlation_matrix = stock_data[['Open', 'High', 'Low',

# Plot the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolw
plt.title('Correlation Heatmap')
plt.show()
```

Heatmap.py hosted with ❤ by GitHub view raw

Output:

RELATED ARTICLES

LangChain for Equity Inv



The heatmap above visualises the correlation between the selected numeric columns of Apple Inc.'s stock data, with a colour map that highlights the strength of the correlations.

7. Box Plots: Box plots summarise the distribution of stock returns, showing median, quartiles, and outliers. They are useful for understanding volatility and return distributions.

Code Example:

```
import yfinance as yf
import seaborn as sns
import matplotlib.pyplot as plt

# Download historical stock data for Apple (AAPL)
```

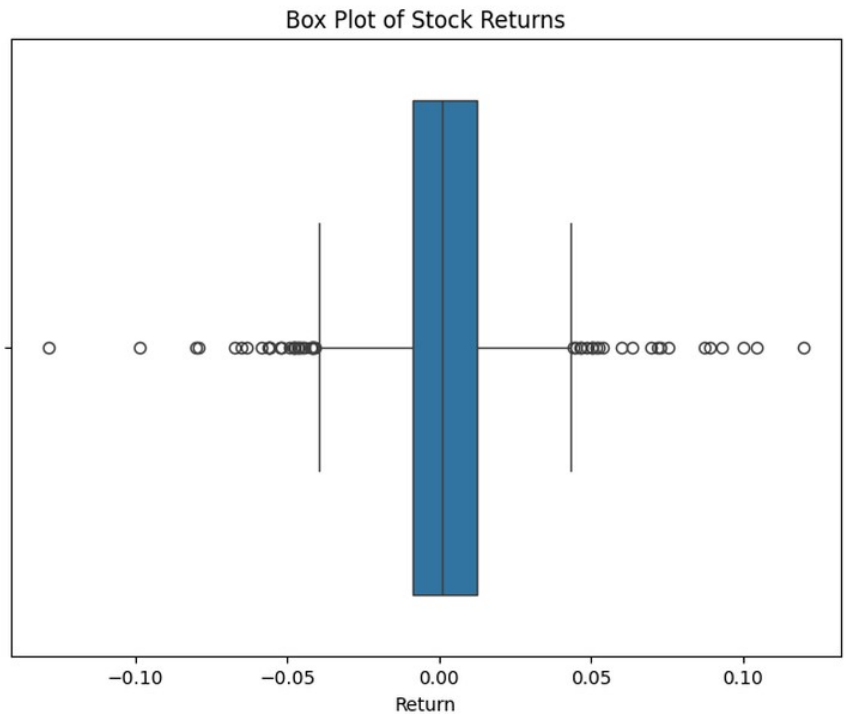
RELATED ARTICLES

LangChain for Equity Inv

```
stock_data = yf.download('AAPL', start='2020-01-01', en
# Calculate daily returns
stock_data['Return'] = stock_data['Close'].pct_change()
# Drop NaN values resulting from the percentage change
stock_data = stock_data.dropna()
# Plotting the box plot of stock returns
plt.figure(figsize=(8, 6))
sns.boxplot(x=stock_data['Return'])
plt.title('Box Plot of Stock Returns')
plt.xlabel('Return')
plt.show()
```

Box_plot.py hosted with ❤ by GitHub view raw

Output:



The box plot above visualises the distribution of daily stock returns, showing key statistical summaries such as the median, quartiles (one of the quantiles), and

RELATED ARTICLES

LangChain for Equity Inv

potential outliers (an important part of data cleaning).

Each technique provides unique insights into stock market data, helping to uncover trends, relationships, and anomalies in the market.

Suggested reads on Data Visualisation using Python:

- Seaborn for Python Data Visualization
- Plotly Python for an Interactive Data Visualization
- Bokeh for Data Visualization in Python

You will find it very useful and knowledgeable to read through this list consisting of some of our top blogs on:

- Python for Trading
- Machine Learning
- Sentiment Trading
- Algorithmic Trading
- Options Trading
- Technical Analysis

Conclusion

Data analysis is vital in stock trading, transforming

raw data into actionable insights that inform trading strategies and decisions. Setting up a robust Python environment and following systematic steps to obtain and visualise stock market data are essential for effective analysis. Also, utilising various visualisation techniques helps in identifying trends, patterns, and correlations within the data.

Fetching stock market data in Python can be done using libraries like yfinance, which allows for the retrieval of historical data across different geographies. We also discussed real-life examples, such as analysing S&P 500 stock tickers, intraday data, and resampling, to demonstrate the practical applications of these techniques.

Additionally, incorporating fundamental data enriches the analysis, providing a comprehensive view of market conditions. By mastering these tools and techniques, traders can enhance their ability to make informed, data-driven decisions in the stock market.

Moreover, Getting market data is a comprehensive course to help with learning how to fetch various data like pricing data of stocks, fundamental data and news headlines data. This course is available **FREE** of cost and can be accessed to gain a thorough knowledge for fetching data, performing quality checks,

RELATED ARTICLES

LangChain for Equity Inv

visualisation as well as the analysis of the data with Python language.

With this course, you will learn all the abovementioned essentials of stock market data with the help of various formats such as videos, documentation, codes, etc. Also, you can take the quiz to confirm the gained information.

File in the download

- Stock market data analysis in Python - Python notebook

Login to Download

Note: The original post has been revamped on 30th August 2024 for recentness, and accuracy.

Disclaimer: All investments and trading in the stock market involve risk. Any decision to place trades in the financial markets, including trading in stock or options or other financial instruments is a personal decision that should only be made after thorough research, including a personal risk and financial assessment and the engagement of professional assistance to the extent you believe necessary. The

trading strategies or related information mentioned in this article is for informational purposes only.

Live Webinar: GenAI & Automated Trading

Share Article:

https://blog.quantinsti.com/stock-market-data-analys

RELATED ARTICLES

LangChain for Equity Inv

Jun 06, 2024

Automated Forex Trading: A Step-by-Step Guide

RELATED ARTICLES

LangChain for Equity Inv

Sep 11, 2024

Automated Trading Systems: Architecture, Protocols, Types of Latency

31 Comments

Sign Up To Comment

Mohamed Afkar • 5 Years Ago

Merci

QuantInsti • 5 Years Ago

De rien

Bryce Viorst • 4 Years Ago

Thank you! unfortunately I keep coming up with the error 'numpy.int64' object has no attribute 'to_pydatetime' when I try to run the full tear sheet. Do you know how to fix this? Thanks.

QuantInsti • 4 Years Ago

Hi! Thank you for the comment. You can downgrade your pandas to 0.25 and it should work fine. You can read

RELATED ARTICLES

LangChain for Equity Inv

through this thread if it doesn't fix: https://github.com/quantopian/pyfolio/issues/520 We hope this helps?

Alejandro Mazzuca • 4 Years Ago

Gracias!

QuantInsti • 4 Years Ago

De nada...

sandeep bhanu teja • 4 Years Ago

For indian market we can get futures and options data from nsepy. But where can we get equity data.

QuantInsti • 4 Years Ago

Hi! You can just replace the ticker with the stock which you want the data for. You can get the ticker information from Yahoo Finance. For example, for HDFC Bank the ticker is HDFCBANK.NS and for Nifty it is ^NSEI.

from pandas_datareader import data

Set the start and end date

start*date* = *'1990-01-01' end*date =

'2019-02-01'

Set the ticker

ticker = 'HDFCBANK.NS'

Get the data

data = data.get*data*yahoo(ticker, start*date, end*date)

Jeremy Karsenty • 4 Years Ago

Bonjour, Merci beaucoup très instructif. Y a t-il un moyen de récupérer les metrics de Ticker.info sur plusieurs années et non uniquement le last ?

QuantInsti • 4 Years Ago

Unfortunately, the free data is only of the last 1 year. However, you can use Quandl to get fundamental data which is older than a year. The link to it is as follows. https://www.quandl.com/databases/ZFA/data

Supriya Devidutta • 4 Years Ago

thx , beautifully explained, nice piece of article,

QuantInsti • 4 Years Ago

RELATED ARTICLES

LangChain for Equity Inv

Thank you. We're glad you liked it.

Pradhyumn Jain • 4 Years Ago

How do I get Implied volatility data while importing option chain data?

Ishan Shah • 4 Years Ago

Thanks, Pradhyumn for your question. IV is a derived data. You can use the black Scholes model to get the IV. For example, you can use python mibian library, pass the option price, strike price, underlying price, days to expiry, type of option (call/put) to get the implied volatility.

Lif Moon • 4 Years Ago

Visualization and Analysis with pyfolio

Hi, I am trying to follow your outstanding tutorial. But, when trying to use "pyfolio" I get some error messages: AttributeError: module 'pandas.core.indexing' has no attribute 'get*indexers*list' Any suggestions? Thank you for your time, Li

D

Dan Sch • 4 Years Ago

In the course : Course Name: Python for Trading: Basic, Section No: 5, Unit No: 7, Unit

RELATED ARTICLES

LangChain for Equity Inv

type: Notebook

Why to import data from Yahoo and also from Quandle?

DS

QuantInsti • 4 Years Ago

Hi Dan, thanks for your comment. To address your question, these are used only for illustration purpose. And considering that Quandl and Yahoo are widely used. Yahoo provides daily data and some days of minute level data for free and also has wide coverage in terms of asset class. We hope this helps.

S

swetha biofincapital • 3 Years Ago

where can we get historical market cap data for a specified date

QuantInsti • 3 Years Ago

Thank you for the comment, Swetha. Perhaps this article may be of assistance to you: https://blog.quantinsti.com/historical-market-data-python-api

RELATED ARTICLES

LangChain for Equity Inv

RELATED ARTICLES

Langchain for Equity Inv

R **Rajesh Kumar Kumar** • 3 Years Ago

When I give the following code, I am getting the error mentioned below. Kindly help.

import pandas as pd

!pip install pandas_datareader==0.7.0

from pandas*datareader import data*

*data=data.get*data_yahoo('AAPL',start='2017-01-01',end='2017-04-30') data.head()

ImportError: cannot import name 'urlencode' from 'pandas.io.common' (C:\Users\rciinturk2\Anaconda3\lib\site-packages\pandas\io\common.py)

QuantInsti • 3 Years Ago

Hi, Rajesh. Thanks for your comment. The solution is to upgrade the pandas-datareader version. You can use the below command to do that.

```
!pip install pandas-datareader --upgrade
```

We hope this helps.

DrSaaie K B • 3 Years Ago

RELATED ARTICLES

LangChain for Equity Inv

Hello, Though I studied programming 30 years back, I found it an uphill task to install yfinance.py on jupyter notebook. At last, I could install it. After its installation the following messages appeared:

```
Requirement already satisfied: yfinance
Requirement already satisfied: numpy>=1
Requirement already satisfied: multitasl
Requirement already satisfied: lxml>=4.5
Requirement already satisfied: pandas>=0
Requirement already satisfied: requests>
Requirement already satisfied: pytz>=201
Requirement already satisfied: python-da
Requirement already satisfied: six>=1.5
Requirement already satisfied: chardet<5
Requirement already satisfied: urllib3<1
Requirement already satisfied: idna<3,>=
Requirement already satisfied: certifi>=
    import yfinance as yf
```

In windows settings>>App; I could not find yfinance under installed files.

I am determined to install this. I request you to help me. If possible inform me by email me at bless.vskp@gmail.com

QuantInsti • 3 Years Ago

Thank you for the comment. Could you kindly confirm if you had

activated the environment in which you installed yfinance prior to running the python files?

DrSaaie K B • 3 Years Ago

Very grateful to you for your prompt response. Your webpage inspired me a lot and I studied Python for 3 hours. I have also learned about Jupyter and Anaconda environment today after receiving your comments. My child helped me in activating the environment. After activating the environment today, I tried to install yfinance 0.1.54.

The last sentence of anaconda activation is as follows: "'D:\Python-Anaconda\envs\stock-screener\Scripts\spyder-script.py', '--reset'] To activate this environment, use

$ conda activate stock-screener

To deactivate an active environment, use

RELATED ARTICLES

LangChain for Equity Inv

RELATED ARTICLES

LangChain for Equity Inv

$ conda deactivate"

After this activation using activate command, installation of yfinance 0.1.54 returned the following lines:

(stock-screener) D:\STOCK SCANNER GITHUB\yfinance-python\yfinance-0.1.54\dist*yfinance-0.1.54\yfinance-0.1.54>python setup.py usage: setup.py [global*opts] cmd1 [cmd1*opts] [cmd2 [cmd2*opts] ...] or: setup.py --help [cmd1 cmd2 ...] or: setup.py --help-commands or: setup.py cmd --help

error: no commands supplied Now I am stuck-up here. Kindly guide me further. I am determined to run this package with your gracious help. Regards.

QuantInsti • 3 Years Ago

Hi! Thanks for your comment. After activating your environment by running `conda activate stock-screener`, please run the command

`pip install yfinance==0.1.63` .

This should install yfinance on your system.

Manu Bhatnagar • 3 Years Ago

Hi, I get this error when I run your program Exception in thread Thread-6: Traceback (most recent call last): File "C:\Users\myname\anaconda3\lib\threading.py", line 926, in *bootstrap*inner self.run() . . . File "C:\Users\myname\anaconda3\lib\json\decoder.py", line 355, in raw_decode raise JSONDecodeError("Expecting value", s, err.value) from None json.decoder.JSONDecodeError: Expecting value: line 1 column 1 (char 0)

QuantInsti • 3 Years Ago

Hi Manu, thank you for your comment. Could you please provide more information on what code cell you were trying to run when you faced this error? This would help us understand your query better and advise accordingly.

joydip biswas • Last Year

Thanks for the post. Is there any module available to download data from MCX ?

RELATED ARTICLES

LangChain for Equity Inv

QuantInsti • Last Year

Hi Joydip, thanks for your comment. Sure, you can get the MCX commodities data from here: https://www.mcxindia.com/market-data/bhavcopy

PRINCE SISHOLIA • 6 Months Ago

I am facing issue in ploting bar graph for EBIT and Revenue

QuantInsti • 6 Months Ago

Hi! Thank you for your comment and for sharing your observation. The same has been updated on the blog. Do have a look.

RELATED ARTICLES

LangChain for Equity Inv

Live Webinar: GenAI & Automated Trading

About

QuantInsti

EPAT

CSAF

Quantra

Blueshift

RELATED ARTICLES

LangChain for Equity Inv

Events & Announcements

Announcements

Webinars

Quick Links

Contact Us

Privacy Policy

Blog Contribution