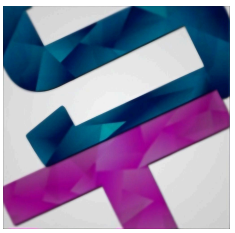


Finance and Analytics Club IITKanpur

Semester Project
Science and Technology Council
IIT Kanpur

ALGORISK INSIGHT

Documentation

CONTENTS

1. **PYTHON BASICS AND LIBRARIES**
    - Pandas
    - Numpy
    - Matplotlib.pyplot
    - Seaborn
    - Yahoo finance

2. **READING CHART PATTERNS**
    - Line graph
    - Candlestick pattern and interpretation

3. **BASIC FINANCIAL CONCEPTS**
    - Volatility
    - Time value of money
    - Cumulative returns and Log Returns
    - Moving Average(SMA and EMA)
    - Sharpe Ratio
    - Correlation Coefficient
    - Maximum Drawdown

4. **TECHNICAL ANALYSIS AND INDICATORS**
    - Understanding Volume
    - Leading vs lagging indicators
    - MACD
    - RSI and RSI divergence
    - ATR
    - Bollinger Band
    - MFI
    - Supertrend
    - Ketlner channel

5. **BASIC RISK MANAGEMENT TACTICS**
    - Stop loss
    - Take profit
    - Trailing stop loss
    - Variable stop loss

6. **UNWANTED DETAILS OF STOCK PRICE**
    - Noise
    - Denoising techniques - heiken aishi and renko chart

7. **SIGNAL GENERATION IN PYTHON**

8. **MEAN REVERSION**
    - Z score
    - Cointegration
    - Stationary time series
    - Hedging
    - Tests- Adf, granger causality, kpss

9. **ADVANCED TECHNICAL ANALYSIS**
    - Greedy chart pattern
    - Parabolic SAR
    - Inside bar
    - Pivot reversal
    - Pivot extension
    - ADX

10. **CAPM**
    - Alpha
    - Beta
    - CAPM

11. **PORTFOLIO OPTIMISATION**
    - VaR - historic and parametric
    - CoVaR - historic and parametric

12. **MONTE CARLO SIMULATIONS**
    - Monte Carlo Simulations in finance
    - Jarque Bera Test for normal distribution

13. **CPPI**
    - CPPI
    - Snenario Analysis
    - Stress Testing

PYTHON LIBRARIES

PANDAS-

Pandas is a very strong library widely used in the field of finance for data analysis, manipulation, and modelling. Its flexible data structures and powerful functions make it a valuable tool for various financial tasks. Here are some common use cases of pandas in finance:

1. Data Acquisition: Pandas can be used to retrieve financial data from various sources. It allows you to import and organise financial data into a pandas DataFrame.

2. Data Cleaning and Preprocessing: Financial datasets often contain missing values, outliers, or inconsistencies. Pandas provides functions for handling missing data, removing duplicates, handling outliers, and performing data transformations.

3. Data Exploration and Analysis: Pandas offers powerful tools for exploring financial data. It allows you to slice and filter data based on specific criteria, calculate summary statistics, and perform grouping.

4. Time Series Analysis: pandas provides excellent support for working with time-based data. It offers functions for resampling data at different frequencies, and conducting various time series.

5. Financial Modeling and Analytics: Pandas can be used for building financial models and conducting quantitative analysis.It provides functions for calculating financial metrics such as moving averages, returns, volatility, and correlations.

NUMPY-

NumPy is efficient in handling multi-dimensional arrays and mathematical functions. Here are some common use cases of NumPy in finance:

1. Efficient Data Storage: NumPy arrays provide a compact and efficient way to store and manipulate large datasets in memory.

2. Mathematical Operations: NumPy provides a wide range of mathematical functions that are crucial in financial calculations.

3. Array Operations and Manipulation: NumPy allows for efficient array operations and manipulation, such as element-wise operations, array slicing, reshaping, and merging.
4. Random Number Generation: NumPy's random module provides functions for generating random numbers and random arrays.
5. Performance Optimization: NumPy is implemented in highly optimized C code, enabling efficient computation and faster execution compared to traditional Python lists.
6. Integration with Other Libraries: NumPy seamlessly integrates with other scientific computing libraries in Python, such as pandas, SciPy, and scikit-learn.

MATPLOTLIB-

Matplotlib is a popular data visualisation library in Python, and it finds extensive use in the field of finance. It provides a wide range of functions and tools for creating insightful and visually appealing plots, charts, and graphs. Here are some common use cases of Matplotlib in finance:
1. Stock Price Visualisation: Matplotlib can be used to create line charts or candlestick charts to visualise historical stock prices.
2. Portfolio Performance Analysis: Matplotlib allows for the creation of line plots or bar plots to visualise the performance of investment portfolios.
3. Financial Time Series Analysis: Matplotlib supports the visualisation of various financial time series data, such as interest rates, economic indicators, or exchange rates.
4. Risk Analysis and VaR Visualization: Matplotlib can be used to create histograms, box plots, or violin plots to analyse the distribution of investment returns or risk factors.
5. Correlation and Heatmap Visualization:It enables the creation of correlation matrices and heatmaps to visualise the relationships and dependencies between financial assets.
6. Financial Report Generation: Matplotlib can be utilised to generate visually appealing plots and charts for financial reports or presentations

SEABORN-

Seaborn, a data visualization library built on top of Matplotlib, is widely used in the field of finance for creating visually appealing and informative plots. Here are some common use cases of Seaborn in finance:
1. Data Distribution Visualization: Seaborn offers various plots such as histograms, kernel density estimation (KDE) plots, and box plots to visualize the distribution of financial data.

2. Time Series Visualization: Seaborn provides specialized functions like line plots and area plots to visualize time series data.
3. Correlation Analysis: Seaborn offers powerful tools for visualizing and analyzing correlations between financial variables.
4. Regression Analysis: Seaborn simplifies the creation of regression plots, such as scatter plots with fitted regression lines, confidence intervals, or residual plots.
5. Categorical Data Visualization: Seaborn provides functions to visualize categorical data, such as bar plots, count plots, or categorical scatter plots.
6. Multivariate Visualization: Seaborn offers tools for visualizing multivariate relationships in financial data. Functions like pairplot and jointplot can be used to create scatter plot matrices and visualize the pairwise relationships between multiple variables.

READING CHART PATTERNS

1. Line Graph

A line graph is a graphical way to visualise the movement of stock prices over time. The following are some terms related in graph:

1. **Horizontal Axis (X-Axis)**: This represents time. It could be days, weeks, months, or even years.
2. **Vertical Axis (Y-Axis)**: This represents the stock price.
3. **Line**: The line on the graph shows how the stock price changes over the selected time period.

   Some key points for the graph line are as follows:

- **Upward Slope**: If the line is going up, the stock price is increasing.
- **Downward Slope**: If the line is going down, the stock price is decreasing.
- **Flat Line**: If the line is flat, the stock price is staying the same.

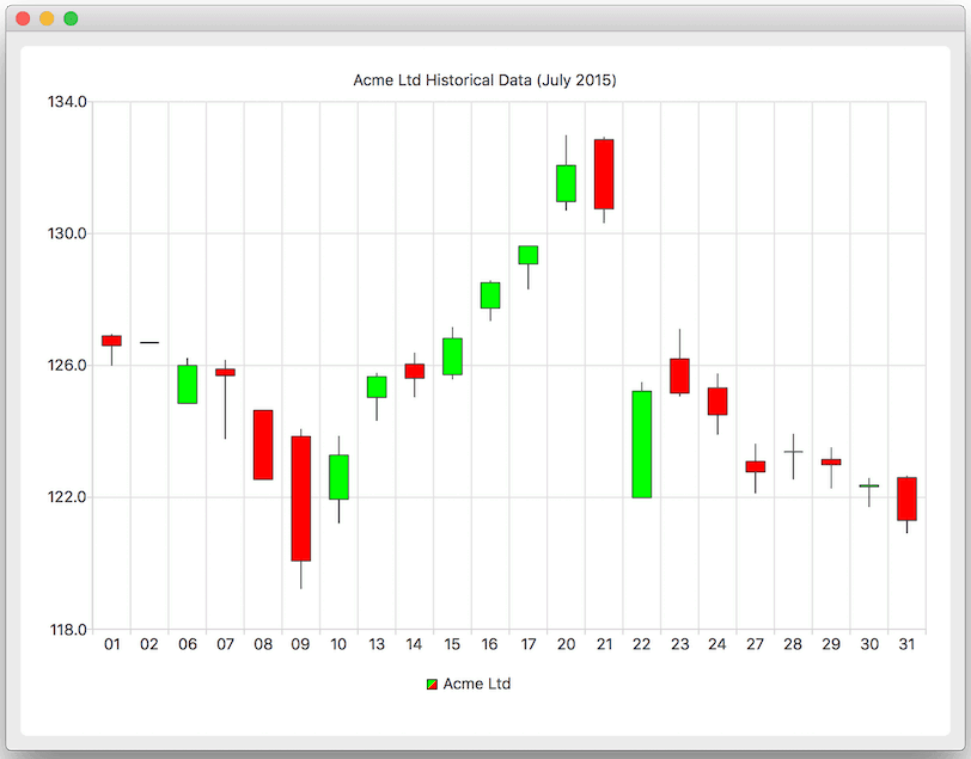
2. Candlestick Pattern:

A candlestick chart is a type of graph used to show how the price of a stock changes over time. Each "candlestick" give four pieces of information about the stock's price during a specific time period:

1. **Opening Price**: How much the stock was worth at the beginning of the period.
2. **Closing Price**: How much the stock was worth at the end of the period.
3. **Highest Price**: The highest price the stock reached during the period.
4. **Lowest Price**: The lowest price the stock reached during the period.

Parts of a Candlestick

1. **The Body**:
    - The wide part of the candlestick is called the body.
    - If the body is green, the closing price is higher than the opening price (the stock went up).
    - If the body is red, the closing price is lower than the opening price (the stock went down).
2. **The Wick (or Shadow)**:
    - The thin lines above and below the body are called wicks or shadows.
    - The top of the upper wick shows the highest price.
    - The bottom of the lower wick shows the lowest price.



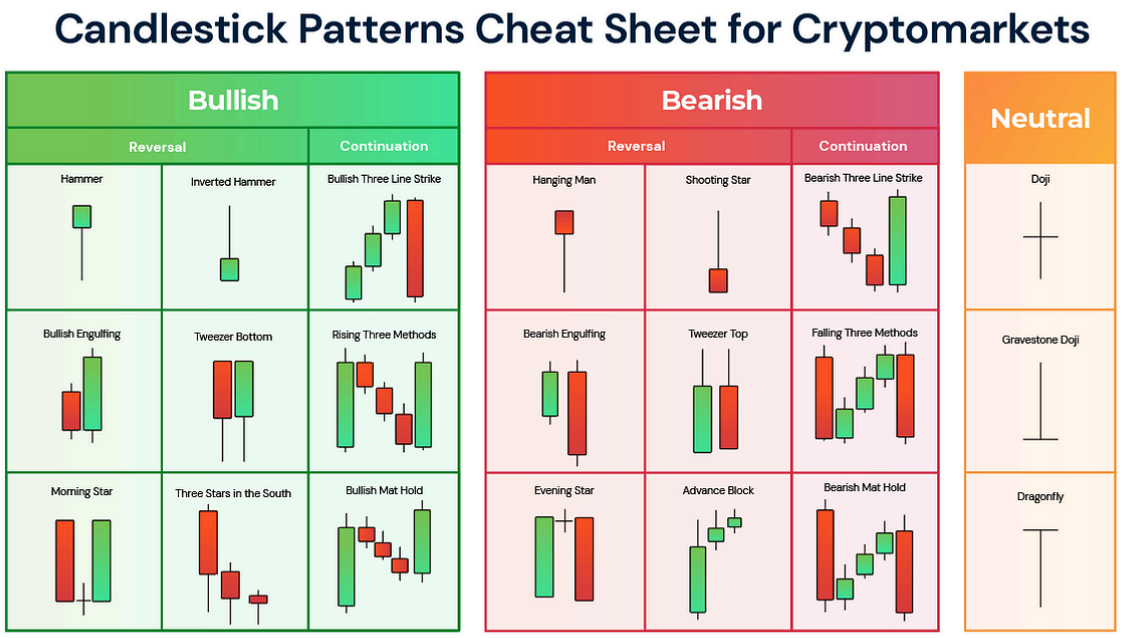
Reading Candlestick Patterns

1. **Bullish Patterns** (Stock price might go up):
    - **Hammer**: Looks like a hammer with a short body and a long lower wick. This means the price dropped a lot but then went back up.
    - **Bullish Engulfing**: A small red candle (down) followed by a big green candle (up). This suggests strong buying.
2. **Bearish Patterns** (Stock price might go down):
    - **Shooting Star**: Looks like a star with a small body and a long upper wick. This means the price went up a lot but then dropped back down.
    - **Bearish Engulfing**: A small green candle (up) followed by a big red candle (down). This suggests strong selling.
3. **Neutral Patterns**:
    - **Doji**: The opening and closing prices are almost the same, making a very small body. This means the market is undecided.

Example

If you see a hammer pattern after a period where the stock price has been dropping, it could mean the price might start going up. If you see a shooting star pattern after the price has been rising, it could mean the price might start going down.

By recognizing these patterns, you can get a better idea of when to buy or sell stocks.



BASIC FINANCIAL CONCEPTS

Volatility

1.Volatility is a statistical measure of the dispersion of returns for a given security or market index.It is often measured from either the standard deviation or variance between those returns.In most cases, the higher the volatility,the riskier the security.
In the securities markets, volatility is often associated with big price swings either up or down. For example, when the stock market rises and falls more than 1% over a sustained period of time, it is called a volatile market.An asset's volatility is a key factor when pricing options contracts.

2. Volatility is often calculated using variance and standard deviation (the standard deviation is the square root of the variance). Since volatility describes changes over a specific period of time, you simply take the standard deviation and multiply that by the square root of the number of periods in question:

$$Volatility = \sigma\sqrt{T}$$

where:

- σ = standard deviation of returns
- T = number of periods in the time horizon

Time Value of Money

1. You have won a cash prize. You have two options available to you. A) receive $10,000 now, or B) Receive $10,000 in 3 years. Which do you choose?If you're like most people, you would choose to receive the $10,000 now. After all, three years is a long time to wait. Why would any rational person defer payment into the future when they could have the same amount of money now? For most of us, taking the money in the present is just plain instinctive. So at the most basic level,the time value of money demonstrates that all things being equal, it seems better to have money now rather than later.If you choose Option A and invest the total amount at a simple annual rate of 4.5%, the future value of your investment at the end of the first year is $10,450. We arrive at this sum by multiplying the principal amount of $10,000 by the interest rate of 4.5% and then adding the interest gained to the principal amount:m $10,000×0.045=$450 $450+$10,000=$10,450

2. You can also calculate the total amount of a one-year investment with a simple manipulation of the above equation: OE=($10,000×0.045)+$10,000=$10,450 ,where:OE=Original equation. OE=($10,000×0.045)+$10,000=$10,450

3. If the $10,450 left in your investment account at the end of the first year is left untouched and you invested it at 4.5% for another year, how much would you have? To calculate this, you would take the $10,450 and multiply it again by 1.045 (0.045 +1). At the end of two years, you would have $10,920.25.

4. FV = PV × (1+*i*)^N

Where,

FV = Future value

PV = Present value (original amount of money)

i = Interest rate per period

N = Number of periods

Cumulative returns

1. A cumulative return on an investment is the aggregate amount that the investment has gained or lost over time, independent of the amount of time involved. The cumulative return is expressed as a percentage, and it is the raw mathematical return of the following calculation:
2. CR=(*Current Price of Security*)−(*Original Price of Security*)/(*Original Price of Security*)

Log returns

1. Logarithmic returns, also known as log returns, are a way of measuring the percentage change in the value of an asset over a period of time. Unlike simple returns, which measure the absolute change in the value of an asset, logarithmic returns measure the relative change in the value of an asset.

2. The formula for calculating logarithmic returns is:

`Logarithmic Return = ln(Present Value / Past Value), where ln is log`

Simple moving average(SMA)

Simple Moving Average (SMA) is a commonly used technical analysis tool in finance and investing. It is a calculation that helps smooth out price data over a specified period of time to identify trends and potential support/resistance levels. The Simple Moving Average is calculated by adding up the closing prices of an asset over a specified number of periods (e.g., days, weeks, months) and then dividing the sum by the number of periods.Here's a step-by-step process to calculate the Simple

Moving Average: Determine the number of periods for which you want to calculate the SMA (e.g., 10 days, 50 days, 200 days).Collect the closing prices of the asset for the specified number of periods.Add up the closing prices. Divide the sum by the number of periods. For example, let's calculate a 10-day Simple Moving Average for a stock's closing prices:

Closing prices: [50, 52, 48, 55, 57, 54, 51, 50, 49, 53]
SMA = (50 + 52 + 48 + 55 + 57 + 54 + 51 + 50 + 49 + 53) / 10 = 519 / 10= 51.9 .

Therefore, the 10-day Simple Moving Average of the closing prices is 51.9. Traders and analysts often use the Simple Moving Average to identify potential buy or sell signals. When the price crosses above the SMA, it may indicate a bullish signal, suggesting an upward trend. Conversely, when the price crosses below the SMA, it may indicate a bearish signal, suggesting a downward trend. It's worth noting that the Simple Moving Average is just one of many technical analysis indicators available, and its effectiveness may vary depending on the market conditions and other factors.

Exponential moving average(EMA)

Exponential Moving Average (EMA) is a type of moving average that gives more weight to recent data points, making it more responsive to price changes compared to the Simple Moving Average (SMA). It is a popular technical analysis tool used to analyze trends and identify potential entry or exit points in financial markets.The Exponential Moving Average is calculated using the following steps: Determine the number of periods for which you want to calculate the EMA (e.g., 10 days, 50 days, 200 days). Collect the closing prices of the asset for the specified number of periods.Calculate the smoothing factor (multiplier) based on the number of periods. The formula for the smoothing factor is typically: Smoothing Factor = (2 / (number of periods + 1)) For example, if you're calculating a 10-day EMA, the smoothing factor would be 0.1818 (2 / (10 + 1) = 0.1818). Calculate the initial EMA for the first period using the SMA. This can be the closing price for the first period. EMA(1) = Closing price(1) Calculate the subsequent EMAs using the following formula: EMA(current) = (Closing price(current) - EMA(previous)) * Smoothing Factor + EMA(previous) Each subsequent EMA is calculated based on the previous EMA value and the current closing price, with more weight given to recent data points due to the smoothing factor. EMA can be used in various ways, such as identifying trend direction, generating buy/sell signals, and determining support/resistance levels. Traders often look for the crossover of shorter and longer period EMAs (e.g., 10-day EMA crossing above the 50-day EMA) as potential buy or sell signal.

Code for calculating sma and ema:

In [36]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import yfinance as yf

stock_symbol = "RELIANCE.NS"
stock_data = yf.download(stock_symbol, start="2023-01-01", end="2023-04-30")
stock_data
```

```
[**********100%**********]  1 of 1 completed
```

Out[36]:

Date	Open	High	Low	Close	Adj Close	Volume
2023-01-02	2550.000000	2579.000000	2548.199951	2575.899902	2575.899902	2453414
2023-01-03	2565.050049	2573.000000	2547.800049	2557.050049	2557.050049	3534596
2023-01-04	2557.000000	2561.050049	2514.000000	2518.550049	2518.550049	4275746
2023-01-05	2523.500000	2536.399902	2504.000000	2514.050049	2514.050049	6293519
2023-01-06	2526.649902	2547.949951	2518.300049	2536.899902	2536.899902	2930338
...	...	...	...	...	...	...
2023-04-24	2375.000000	2380.899902	2348.000000	2358.000000	2358.000000	5970048
2023-04-25	2366.000000	2380.600098	2350.500000	2376.050049	2376.050049	4262471
2023-04-26	2379.000000	2386.100098	2354.050049	2362.100098	2362.100098	3977129
2023-04-27	2375.000000	2384.000000	2364.000000	2377.050049	2377.050049	4230627
2023-04-28	2382.000000	2423.899902	2381.750000	2420.500000	2420.500000	7183342

79 rows × 6 columns

In [39]:
```
for i in range(sma_period, len(close_prices)):
    sma_values[i] = np.mean(close_prices[i - sma_period : i])

stock_data["SMA"] = sma_values
stock_data
```

Out[39]:

Date	Open	High	Low	Close	Adj Close	Volume	SMA
2023-01-02	2550.000000	2579.000000	2548.199951	2575.899902	2575.899902	2453414	NaN
2023-01-03	2565.050049	2573.000000	2547.800049	2557.050049	2557.050049	3534596	NaN
2023-01-04	2557.000000	2561.050049	2514.000000	2518.550049	2518.550049	4275746	NaN
2023-01-05	2523.500000	2536.399902	2504.000000	2514.050049	2514.050049	6293519	NaN
2023-01-06	2526.649902	2547.949951	2518.300049	2536.899902	2536.899902	2930338	NaN
...	...	...	...	...	...	...	...
2023-04-24	2375.000000	2380.899902	2348.000000	2358.000000	2358.000000	5970048	2350.970020
2023-04-25	2366.000000	2380.600098	2350.500000	2376.050049	2376.050049	4262471	2349.080029
2023-04-26	2379.000000	2386.100098	2354.050049	2362.100098	2362.100098	3977129	2356.220020
2023-04-27	2375.000000	2384.000000	2364.000000	2377.050049	2377.050049	4230627	2358.240039
2023-04-28	2382.000000	2423.899902	2381.750000	2420.500000	2420.500000	7183342	2364.440039

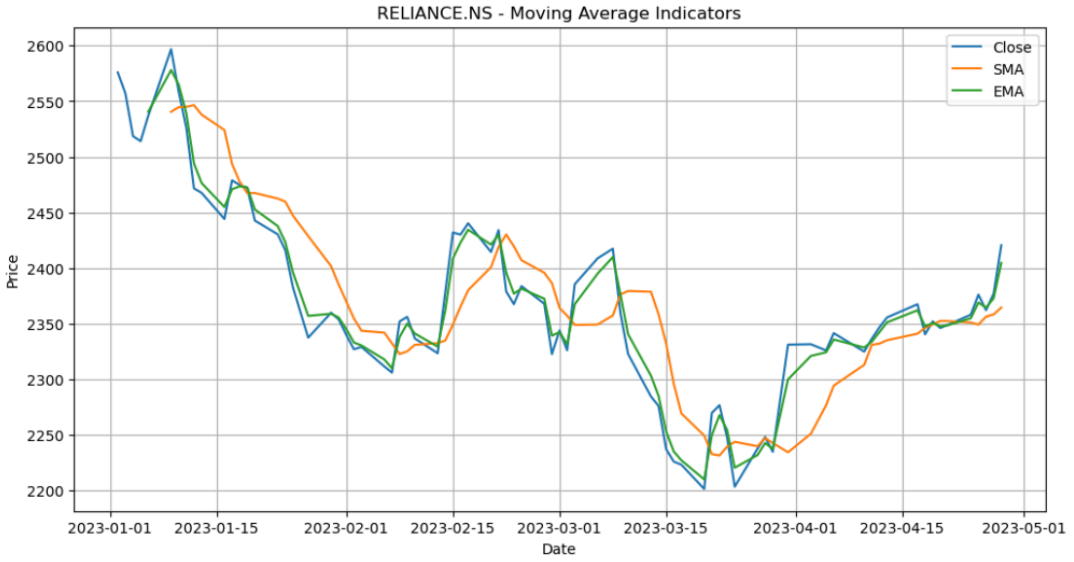
79 rows × 7 columns

In [40]:
```
ema_period = 5
weights = np.exp(-np.arange(ema_period) / ema_period)
weights /= np.sum(weights)
weights
```

```
smoothing_factor = 2 / (ema_period + 1)

ema_values = np.zeros(len(close_prices))
ema_values[:ema_period - 1] = np.nan

ema_values[ema_period - 1] = np.mean(close_prices[:ema_period])
for i in range(ema_period, len(close_prices)):
    ema_values[i] = ema_values[i - 1] * smoothing_factor + close_prices[i] * (1 - smoothing_factor)
```

In [49]:

```
plt.figure(figsize=(12, 6))
plt.plot(stock_data["Close"], label="Close")
plt.plot(stock_data["SMA"], label="SMA")
plt.plot(stock_data["EMA"], label="EMA")
plt.xlabel("Date")
plt.ylabel("Price")
plt.title(f"{stock_symbol} - Moving Average Indicators")
plt.legend()
plt.grid(True)
plt.show()
```



Sharpe ratio

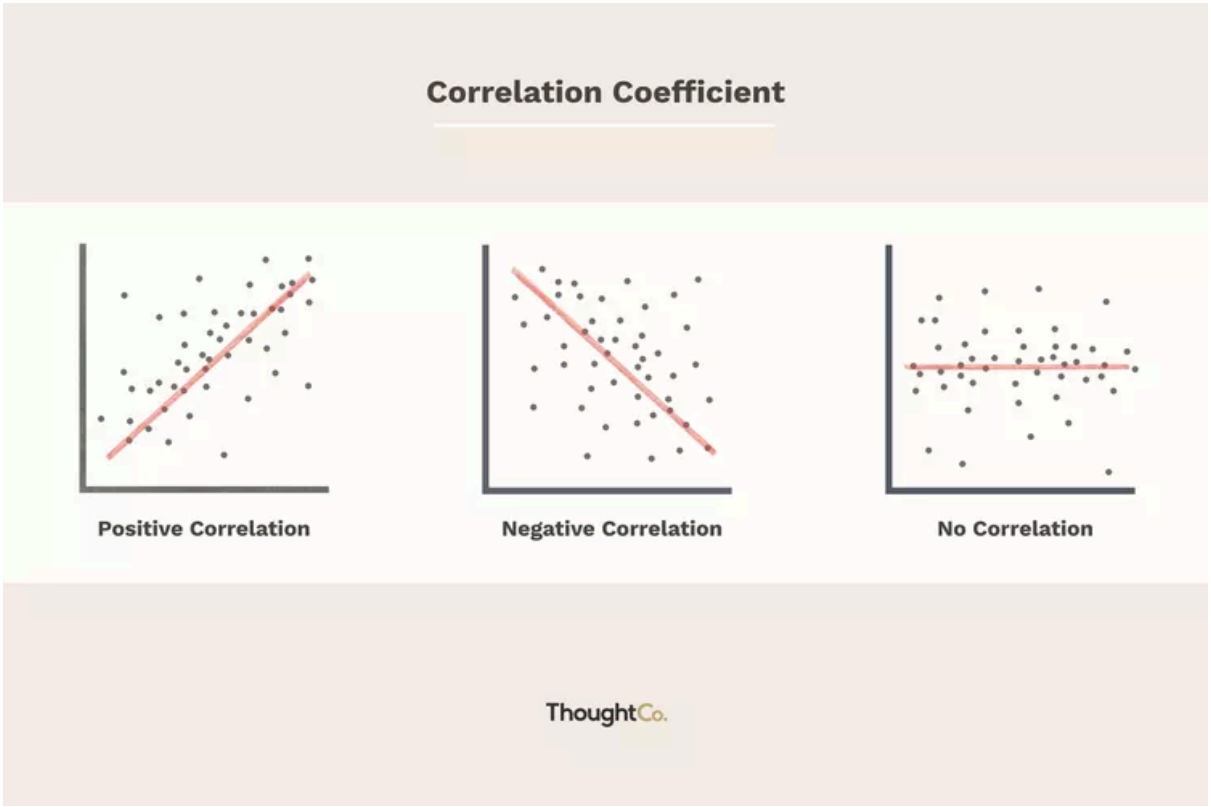
1. The Sharpe ratio compares the return on investment with its risk. It's a mathematical expression of the insight that excess returns over a period of time may signify more volatility and risk, rather than investing skill.
2. The Sharpe ratio's numerator is the difference over time between realized, or expected, returns and a benchmark such as the risk free rate of return or the performance of a particular investment category. Its denominator is the standard deviation of returns over the same period of time, a measure of volatility and risk.
3. FORMULA:

$$\text{Sharpe ratio} = (R_P - R_f)/\sigma$$

Where,
R_P=Return on portfolio
R_f=Risk free rate
σ=Standard deviation

Correlation coefficient

1.The correlation coefficient is a statistical measure of the strength of a linear relationship between two variables. Its values can range from -1 to 1. A correlation coefficient of -1 describes a perfect negative, or inverse correlation, with values in one series rising as those in the other decline, and vice versa. A coefficient of 1 shows a perfect positive correlation, or a direct relationship. A correlation coefficient of 0 means there is no linear relationship.
3. Correlation coefficients are used in science and finance to assess the degree of association between two variables, factors, or data sets. For example, since high oil prices are favourable for crude producers, one might assume the correlation between oil prices and forward returns on oil stocks is strongly positive.



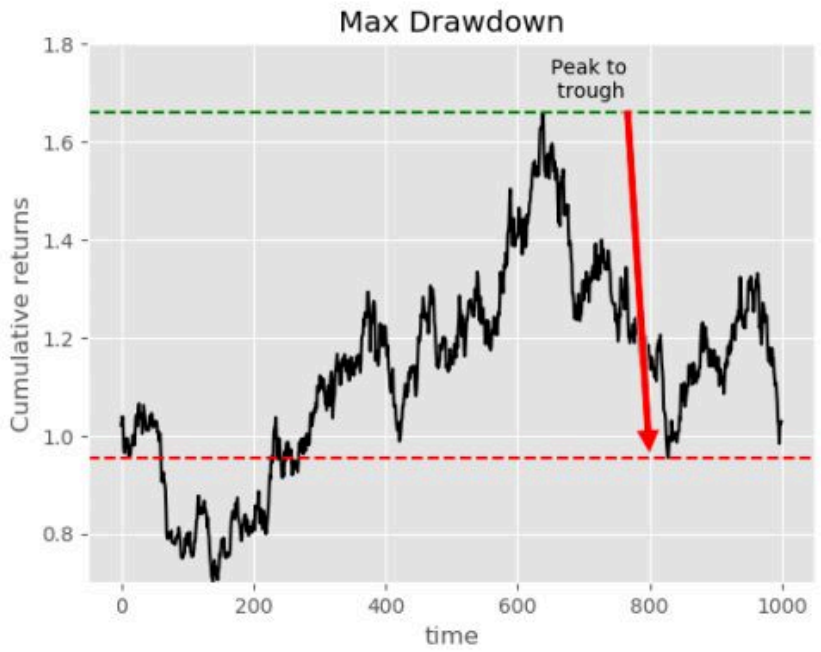
Maximum Drawdown

1. Maximum drawdown is a specific measure of drawdown that looks for the greatest movement from a high point to a low point, before a new peak is achieved. However,

it's important to note that it only measures the size of the largest loss, without taking into consideration the frequency of large losses. Because it measures only the largest drawdown, MDD does not indicate how long it took an investor to recover from the loss, or if the investment even recovered at all.

2. The Formula for Maximum Drawdown Is

MDD=(Trough Value -Peak Value)/Peak Value



PYTHON CODE FOR MAX DRAWDOWN

```
def max_drawdown(return_series):
    comp_ret = (return_series+1).cumprod()
    peak = comp_ret.expanding(min_periods=1).max()
    dd = (comp_ret/peak)-1
    return dd.min()


max_drawdowns = df.apply(max_drawdown,axis=0)
max_drawdowns.plot.bar()
plt.yabel('Max Drawdown')
```

Data denoising

Data denoising is a process where we remove the fluctuations called as noise in the chart pattern to make it look more easier to understand and analyse.

1. Heiken ashi candles

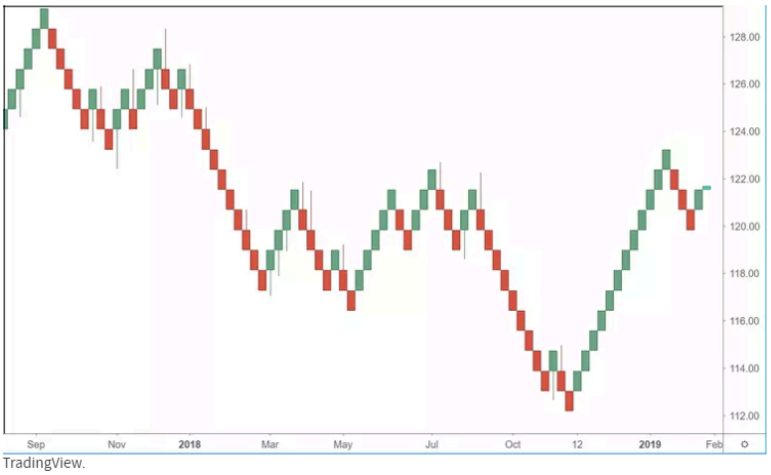
Heikin-Ashi, also sometimes spelled Heiken-Ashi, means "average bar" in Japanese. The heiken ashi technique can be used in conjunction with candlestick charts when trading securities to spot market trends and predict future prices. It's useful for making candlestick charts more readable and trends easier to analyse. For example, traders can use Heikin-Ashi charts to know when to stay in trades while a trend persists but get out when the trend pauses or reverses. Most profits are generated when markets are trending, so predicting trends correctly is necessary.

Formula:

- HA_CLOSE = (OPEN + HIGH + LOW + CLOSE)/4
- HA_OPEN = (OPEN OF PREV_BAR + CLOSE OF PREV_BAR)/2
- HA_HIGH = MAX(HIGH, HA_CLOSE, HA_OPEN)
- HA_LOW = MIN(LOW, HA_OPEN, HA_CLOSE)

2. Renko Charts

A Renko chart is a type of chart, developed by the Japanese, that is built using price movement rather than both price *and* standardised time intervals like most charts are. It is thought to be named after the Japanese word for bricks, "renga," since the chart looks like a series of bricks. A new brick is created when the price moves a specified price amount, and each block is positioned at a 45-degree angle (up or down) to the right of the prior brick. An up brick is typically coloured white or green, while a down brick is typically coloured black or red.



TECHNICAL ANALYSIS AND INDICATORS

Understanding Volume

Volume is the amount of an asset or security that changes hands over some period of time, often over the course of a trading day. For instance, a stock's trading volume refers to the number of shares traded between its daily open and close.
All of the shares of every transaction that takes place between a buyer and seller of a security contributes to the total volume count of that security on that day. For example, if five transactions occurred in one day, and each transaction involved 100 shares, the trading volume for that day would be 500.

Volume tells traders about a market's activity and liquidity. Higher trade volumes for a specific security means higher liquidity, better order execution, and a more active market for connecting buyers and sellers. Trading volume is important as it displays an investor's interest in a certain company. It reflects the momentum as well which takes place when a certain sector or stocks are trending on the higher end. It is significant to note that volume also represents the trend when it is on the verge of ending.

Technical Indicators and its types

Technical indicators are heuristic or pattern-based signals produced by the price, volume, and/or open interest of a security or contract used by traders who follow

technical analysis. By analyzing historical data, technical analysts use indicators to predict future price movements.
Now, these indicators are broadly divided into two types:

1. **Leading Indicators**: Leading indicators are designed to predict future price movements and trends. They provide signals before a new trend or reversal occurs. These indicators are useful for traders who want to anticipate potential changes in the market and act early, but is prone to generate false signals increasing the risks.
2. **Lagging Indicators**: Lagging indicators, also known as trend-following indicators, provide signals after a trend or reversal has started. They confirm the presence of a trend but do not predict it. These indicators are useful for traders who prefer to wait for confirmation before entering or exiting trades, reducing the risk of false signals.

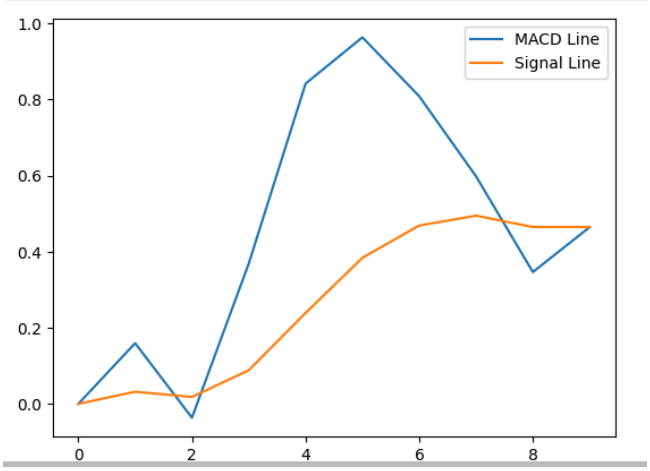
Now, we'll give a brief analysis on some of the common indicators used by investors and trade analysts.

MACD

Moving average convergence/divergence (MACD) is a popular technical indicator to help investors identify price trends, measure trend momentum, and identify market entry points for buying or selling. It is a lagging indicator as it relies on historical price data.
It is calculated by subtracting the longer-term Exponential Moving Average (EMA) from the shorter-term EMA. The MACD line is the main line, while the signal line is the EMA of the MACD line. Here's a step-by-step process to calculate the MACD:
1. Determine the periods for the short-term EMA, long-term EMA, and signal line EMA. Commonly used values are 12, 26, and 9, respectively.
2. Calculate the short-term EMA by taking the average of the closing prices over the short-term period.
3. Calculate the long-term EMA by taking the average of the closing prices over the long-term period.
4. Subtract the long-term EMA from the short-term EMA to obtain the MACD line.
5. Calculate the signal line by taking the EMA of the MACD line over the signal line period.
6. Plot the MACD line and the signal line on a chart, and analyze the crossovers and divergences between the two lines to generate trading signals.

Here's a Python code snippet that demonstrates how to calculate the MACD using the pandas and matplotlib libraries:

```python
import pandas as pd
import matplotlib.pyplot as plt
# Sample closing prices data
closing_prices = [50, 52, 48, 55, 57, 54, 51, 50, 49, 53]
# Define the periods
short_period = 12
long_period = 26
signal_period = 9
# Create a pandas Series from the closing prices
closing_series = pd.Series(closing_prices)
# Calculate the short-term EMA
short_ema = closing_series.ewm(span=short_period, adjust=False).mean()
# Calculate the long-term EMA
long_ema = closing_series.ewm(span=long_period, adjust=False).mean()
# Calculate the MACD line
macd_line = short_ema - long_ema
# Calculate the signal line
signal_line = macd_line.ewm(span=signal_period, adjust=False).mean()
# Plot the MACD line and signal line
plt.plot(macd_line, label='MACD Line')
plt.plot(signal_line, label='Signal Line')
plt.legend()
plt.show()
```



The output will be a chart displaying the MACD line and the signal line. Traders and analysts can analyze the crossovers and divergences between the two lines to generate potential buy or sell signals.

RSI

The relative strength index (RSI) is another popular and a useful technical indicator. The indicator should not be confused with relative strength (RS). RSI measures the speed and magnitude of a security's recent price changes to evaluate overvalued or undervalued conditions in the price of that security.
An asset is usually considered overbought when the RSI is above 70 and oversold when it is below 30. The RSI line crossing below the overbought line or above oversold line is often seen by traders as a signal to buy or sell.

The following is the formula to calculate Relative Strength Index:

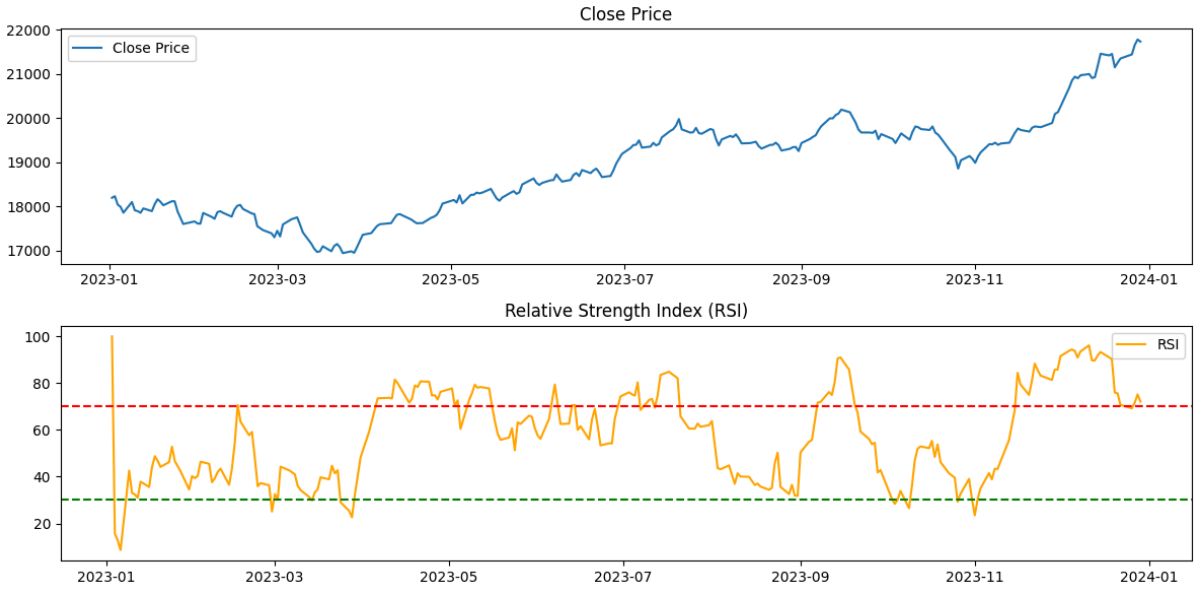
$$RS = \frac{Avg.Gain}{Avg.Loss}$$

$$RSI = 100 - \frac{100}{1 + RS}$$

In this calculation, the average percentage gain or loss is measured over a specific period. The formula treats losses as positive values. During the look-back period, periods with price losses are considered to have a zero average gain, while periods with price increases are treated as having a zero average loss. Generally, the initial RSI value is calculated using 14 periods.

An **RSI divergence** occurs when price moves in the opposite direction of the RSI. In other words, a chart might display a change in momentum before a corresponding change in price.

Below is the plot of RSI and the code snippet for calculating RSI in python.



```python
import yfinance as yf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

def calculate_rsi(data, window=14):
    delta = data['Close'].diff()
    gain = (delta.where(delta > 0, 0)).fillna(0)
    loss = (-delta.where(delta < 0, 0)).fillna(0)

    avg_gain = gain.rolling(window=window, min_periods=1).mean()
    avg_loss = loss.rolling(window=window, min_periods=1).mean()

    rs = avg_gain / avg_loss
    rsi = 100 - (100 / (1 + rs))

    return rsi


ticker = "^NSEI"
data = yf.download(ticker, start="2023-01-01", end="2024-01-01")
data['RSI'] = calculate_rsi(data)
```

MFI

The Money Flow Index (MFI) is a technical indicator that generates overbought or oversold signals using both prices and volume data. An MFI reading above 80 is considered overbought and an MFI reading below 20 is considered oversold, although levels of 90 and 10 are also used as thresholds. Also, divergence between the indicator and price is noteworthy. For example, if the indicator is rising while the price is falling or flat, the price could start rising.

Below is the formula for calculating MFI and the corresponding plot.

$$\text{Money Flow Index} = 100 - \frac{100}{1 + \text{Money Flow Ratio}}$$

where:

$$\text{Money Flow Ratio} = \frac{\text{14 Period Positive Money Flow}}{\text{14 Period Negative Money Flow}}$$

$$\text{Raw Money Flow} = \text{Typical Price} * \text{Volume}$$

$$\text{Typical Price} = \frac{\text{High} + \text{Low} + \text{Close}}{3}$$

Point to be noted here is that MFI and RSI are very closely related. The main difference is that MFI incorporates volume, while the RSI does not. Proponents of volume analysis believe it is a leading indicator. Therefore, they also believe that MFI will provide signals, and warn of possible reversals, in a more timely fashion than the RSI.

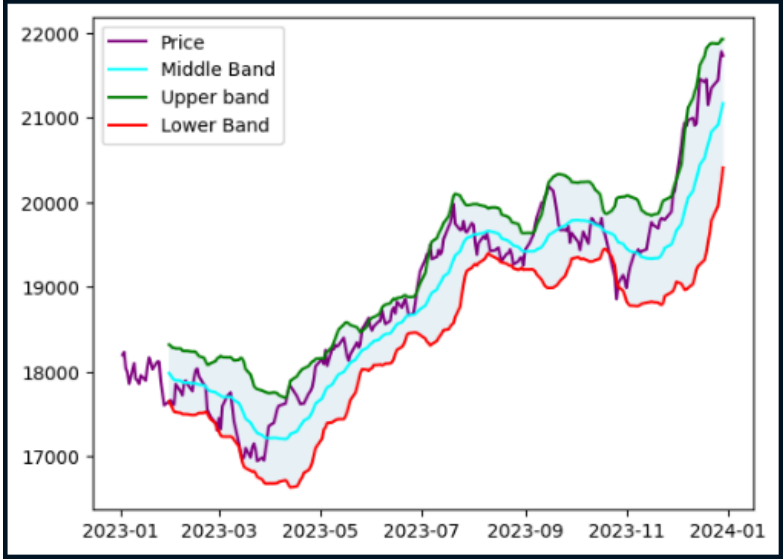
Bollinger Band

Bollinger Bands is a common lagging technical indicatorl used to determine where prices are high and low relative to each other. These bands are composed of three lines: a simple moving average (the middle band) and an upper and lower band. The upper and lower bands are typically two standard deviations above or below a 20-period simple moving average (SMA). The upper and lower bands measure volatility or the degree of variation of prices over time. Because Bollinger Bands measures volatility, the bands adjust automatically to changing market conditions. When the prices are quiet, the bands are close together. When the price moves up, the bands spread apart.

Here’s the corresponding code snippet in python:

```python
import yfinance as yf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
ticker = "INFY"
data = yf.download(ticker, start="2023-04-01", end="2024-03-31")

def bollingerBand(data,period,multiplier):

    data["Middle"] = data["Close"].rolling(20).mean()
    std = data["Close"].rolling(period).std()
    data["Upper"] = data["Middle"] + multiplier * std
    data["Lower"] = data["Middle"] - multiplier * std

    plt.plot(data["Close"], label = "Price", color="purple")
    plt.plot(data["Middle"], label = "Middle Band", color="cyan")
    plt.plot(data["Upper"], label = "Upper band", color="green")
    plt.plot(data["Lower"], label = "Lower Band", color="red")
    plt.legend()
    plt.fill_between(data.index,data["Upper"],data["Lower"],alpha=.1)
    plt.show()

bollingerBand(data,20,2)
```

✓ 0.3s



ATR

The average true range (ATR) is a price volatility indicator showing the average price variation of assets within a given time period, generally take to be 14 periods where period can be monthly, weekly, daily, or even intraday. Investors can use the indicator to determine the best time for trading. The average true range also takes into account the gaps in the movement of price. Originally developed for commodities, the ATR indicator can also be used for stocks and indices. Simply put, a stock experiencing a high level of volatility has a higher ATR, and a lower ATR indicates lower volatility for the period evaluated.

Given below shows how to calculate ATR:

If there is not a previous ATR calculated, you must use:

$$\frac{\text{Previous ATR}(n-1)+\text{TR}}{n}$$

where:

n = Number of periods

TR = True range

$$\left(\frac{1}{n}\right)\sum_{i}^{n}\text{TR}_i$$

where:

TR_i = Particular true range, such as first day's TR, then second, then third

n = Number of periods

$$\text{TR} = \text{Max}\left[(\text{H}-\text{L}), |\text{H}-\text{C}_p|, |\text{L}-\text{C}_p|\right]$$

where:

H = Today's high

L = Today's low

C_p = Yesterday's closing price

Max = Highest value of the three terms

so that:

$(\text{H}-\text{L})$ = Today's high minus the low

$|\text{H}-\text{C}_p|$ = Absolute value of today's high minus yesterday's closing price

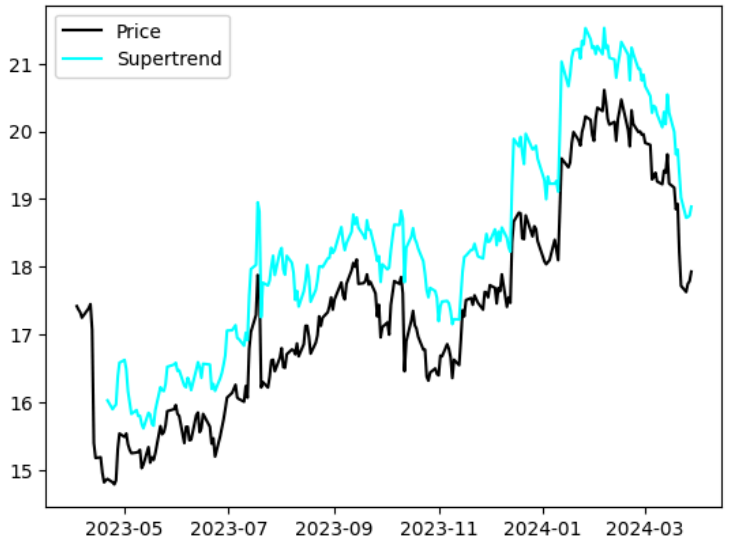
$|\text{L}-\text{C}_p|$ = Absolute value of today's low minus yesterday's closing price

Supertrend

The Supertrend Indicator is a popular technical analysis tool designed to assist traders in identifying market trends. The indicator combines the average true range (ATR) with a multiplier to calculate its value. This value is then added to or subtracted from the asset's closing price to plot the supertrend line. The Supertrend Indicator can help identify trends, manage risk, and confirm market tendencies. The indicator is limited by its lagging nature, is not very flexible, and can send up false signals.

Below is the python code and the plot of the supertrend indicator.

```
def supertrend(df, period, multiplier):
    TR_ATR(df,period)
    df["Supertrend"] = (df["High"] + df["Low"])/2 + multiplier * df["ATR"]

    plt.plot(df["Close"], label="Price", color="black")
    plt.plot(df["Supertrend"], label="Supertrend", color="cyan")
    plt.legend()
    plt.show()

ticker = "INFY"
data = yf.download(ticker, start="2023-04-01", end="2024-03-31")
supertrend(data,14,3)
```



The multiplier in the Supertrend Indicator is employed to adjust the indicator's sensitivity. A higher multiplier makes the indicator less sensitive to price changes and reduces false signals. But it could delay your investment moves. Alternatively, a lower multiplier makes the indicator more sensitive, providing quicker signals but also increasing the risk of false positives.

Keltner Channel

Keltner Channels are volatility-based bands that are placed on either side of an asset's price and can aid in determining the direction of a trend. The exponential moving average (EMA) of a Keltner Channel is typically 20 periods, although this can be adjusted if desired.

The upper and lower bands are typically set two times the average true range (ATR) above and below the EMA, although the multiplier can also be adjusted based on personal preference. Price reaching the upper Keltner Channel band is bullish while reaching the lower band is bearish. The angle of the Keltner Channel also aids in identifying the trend direction. The price may also oscillate between the upper and lower Keltner Channel bands, which can be interpreted as resistance and support levels.

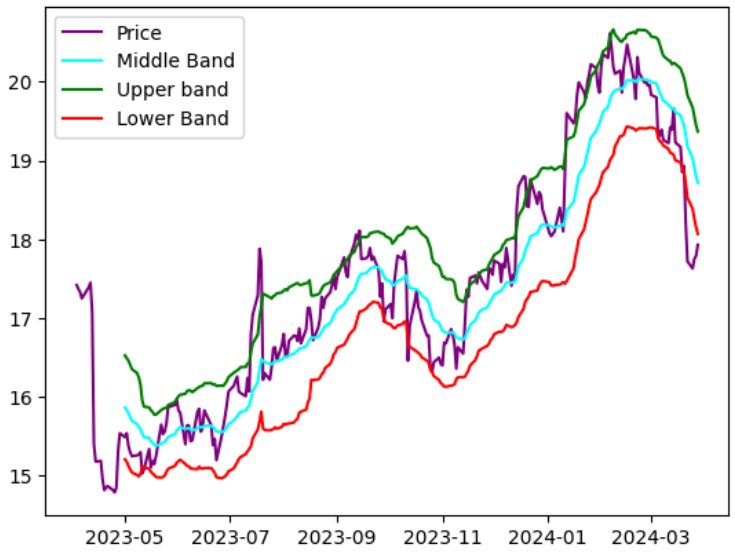
Follow these steps to calculate your Keltner channel:

1. Calculate the EMA for the asset, based on the last 20 periods or the number of periods desired.
2. Calculate the ATR of the asset, based on the last 20 periods or the number of periods desired.
3. Multiply the ATR by two (or the multiplier desired) and then add that number to the EMA value to get the upper band value.
4. Multiply the ATR by two (or the desired multiplier) and subtract that number from the EMA to get the lower band value.
5. Repeat all steps after each period ends.

Note that Keltner Channels use ATR to calculate the upper and lower bands, while Bollinger Bands use standard deviation. The interpretation of the indicators is similar, although since the calculations are different, the two indicators may provide slightly different information or trade signals.

Here's the python code and the plot of the Keltner Channel.

```python
def keltnerChannel(df,period,multiplier):
    ema(df,period)
    TR_ATR(df,period)
    df["Middle"] = df["EMA"]
    df["Upper"] = df["EMA"] + multiplier*df["ATR"]
    df["Lower"] = df["EMA"] - multiplier*df["ATR"]

    plt.plot(df["Close"], label = "Price", color="purple")
    plt.plot(df["Middle"], label = "Middle Band", color="cyan")
    plt.plot(df["Upper"], label = "Upper band", color="green")
    plt.plot(df["Lower"], label = "Lower Band", color="red")
    plt.legend()

ticker = "INFY"
data = yf.download(ticker, start="2023-04-01", end="2024-03-31")
keltnerChannel(data,20,2)
```



BASIC RISK MANAGEMENT TACTICS

While forming strategies using the technical indicators and applying these strategies on a historical data set or real life, we need to make sure to reduce our risks of potential losses as far as possible. Here are some very basic but highly useful tactics which basically involves putting an upper limit to the amount of loss in order to protect the traders capital from significant downturns in the market.

Stop Loss

A stop-loss order is a type of order used by traders to limit their loss or lock in a profit on an existing position. Traders can control their exposure to risk by placing a stop-loss order. For example, a trader may buy a stock and place a stop-loss order with a stop 10% below the stock's purchase price. Should the stock price drop to that 10% level, the stop-loss order is triggered and the stock would be sold at the best available price.
It is generally not radviced, though, to apply stop loss order in a choppy market where prices vary drastically, i.e. high volatile.

Take Profit

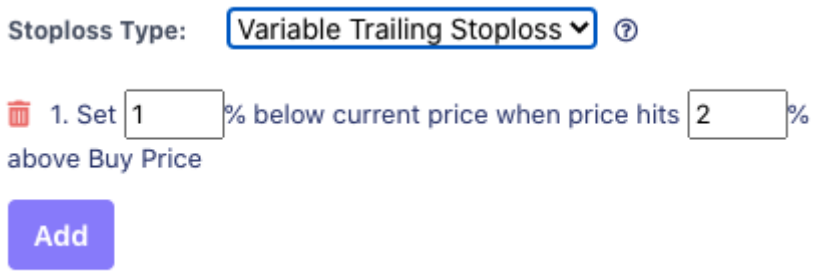
A take-profit order (T/P) is a type of limit order that specifies the exact price at which to close out an open position for a profit. If the price of the security does not reach the limit price, the take-profit order does not get filled.
Take-profit orders are best used by short-term traders interested in managing their risk. This is because they can get out of a trade as soon as their planned profit target is reached and not risk a possible future downturn in the market. Traders with a long-term strategy do not favor such orders because it cuts into their profits.

Trailing Stop Loss

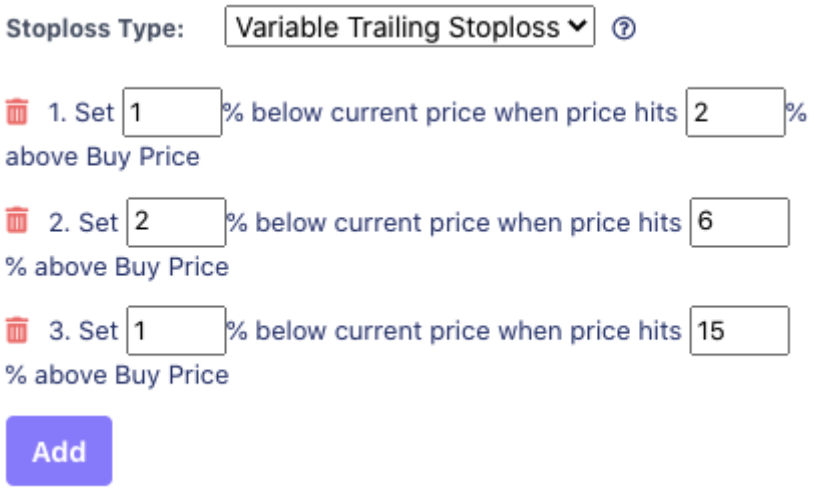
A trailing stop is a modification of a typical stop order that can be set at a defined percentage or dollar amount away from a security's current market price. Trailing stops only move if the price moves favorably. Once it moves to lock in a profit or reduce a loss, it does not move back in the other direction. For example, we can use concept of trailing stop loss like this: If a 10% trailing stop loss is added to a long position, a sell trade will be issued if the price drops 10% from its peak price after purchase.

Variable Stop Loss

As the name implies, Variable Trailing Stop Loss gives the trader the utmost flexibility as to which price level should trigger a new trailing stop loss level.



From the image above, this bot has been configured for just one level for the trailing stoploss. If the price appreciates more than 2%, the trailing stoploss level will remain the same.



For the image above, there are three TSL levels configured for the bot. The table below depicts the specific TSL price for each level.

CAPM

The Capital Asset Pricing Model (CAPM) is a financial model that helps in determining the expected return on an investment based on its risk and the overall market's risk. It provides a framework for evaluating the relationship between expected return and systematic risk, which is represented by beta.

Definitions for a few terms associated with the CAPM:

- Expected Return: The CAPM starts with the concept of expected return, which is the anticipated gain or loss on an investment. It is a measure of the average return an investor expects to receive in exchange for taking on a certain level of risk.
- Risk-Free Rate: The model considers the risk-free rate, which is the return on an investment that is considered to have no risk, such as a government bond. The risk-free rate represents the time value of money and serves as a benchmark for comparing the expected returns of other investments.
- Market Risk Premium: The CAPM takes into account the market risk premium, which is the additional return that investors demand for bearing the risk of investing in the overall market. It is the difference between the expected return on the market as a whole and the risk-free rate.
- Beta: Beta is a measure of an investment's systematic risk or volatility in relation to the overall market. It represents the sensitivity of an investment's returns to changes in the market. A beta of 1 indicates that the investment's returns move in line with the market, while a beta greater than 1 suggests higher volatility, and a beta less than 1 indicates lower volatility.

CAPM Formula:

Expected Return = Risk-Free Rate + Beta × (Market Risk Premium)
This formula implies that the expected return of an investment is equal to the risk-free rate plus a risk premium, which is determined by multiplying the investment's beta by the market risk premium.

The CAPM has several applications. It can be used to estimate the expected return on an individual stock or a portfolio of stocks. It also helps in determining the required rate of return for evaluating investment projects or making capital budgeting decisions. Additionally, the CAPM is used in the valuation of securities, such as estimating the cost of equity capital for valuing companies or calculating the discount rate for valuing future cash flows.

However, it’s important to note that the CAPM also relies on certain assumptions, including the efficient market hypothesis, which assumes that all relevant information is available to investors and that prices reflect this information accurately. It also assumes that investors are rational and risk-averse, seeking to maximize their returns for a given level of risk. While the CAPM provides a useful framework for understanding the relationship between risk and return, it has these limitations. It is argued that the model oversimplifies the complexities of the market and that beta alone is not sufficient to capture all forms of risk.

Therefore, it is often used in conjunction with other valuation models and risk assessment techniques to make more informed investment decisions.

Code for calculating the Expected Return using CAPM:

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import yfinance as yf
```

In [2]:
```
stock = "AAPL"
stock_data = yf.download(stock,'2018-06-01','2023-05-31')

stock_data.head()
```

```
[**********************100%**********************]  1 of 1 completed
```

Out[2]:

Date	Open	High	Low	Close	Adj Close	Volume
2018-06-01	46.997501	47.564999	46.937500	47.560001	45.451923	93770000
2018-06-04	47.910000	48.355000	47.837502	47.957500	45.831795	105064800
2018-06-05	48.267502	48.485001	48.090000	48.327499	46.185398	86264000
2018-06-06	48.407501	48.520000	47.980000	48.494999	46.345474	83734400
2018-06-07	48.535000	48.549999	48.084999	48.365002	46.221241	85388800

In [3]:
```
index = "^GSPC"
index_data = yf.download(index,'2018-06-01','2023-05-31')

index_data.head()
```

```
[**********************100%**********************]  1 of 1 completed
```

Out[3]:

Date	Open	High	Low	Close	Adj Close	Volume
2018-06-01	2718.699951	2736.929932	2718.699951	2734.620117	2734.620117	3694310000
2018-06-04	2741.669922	2749.159912	2740.540039	2746.870117	2746.870117	3410090000
2018-06-05	2748.459961	2752.610107	2739.510010	2748.800049	2748.800049	3523550000
2018-06-06	2753.250000	2772.389893	2748.459961	2772.350098	2772.350098	3662780000
2018-06-07	2774.840088	2779.899902	2760.159912	2770.370117	2770.370117	3742080000

In [4]:
```
stock_close_prices = stock_data["Adj Close"]
stock_returns = []

for i in range(1,len(stock_close_prices)):
    prev_price = stock_close_prices[i-1]
    current_price = stock_close_prices[i]
    return_value = (current_price - prev_price)/prev_price
    stock_returns.append(return_value)

stock_returns = pd.Series(stock_returns)

stock_returns.head()
```

Out[4]:
```
0    0.008358
1    0.007715
2    0.003466
3   -0.002681
4   -0.009097
dtype: float64
```

In [13]:
```
index_close_prices = index_data["Adj Close"]
index_returns = []

for i in range(1,len(index_close_prices)):
    prev_price = index_close_prices[i-1]
    current_price = index_close_prices[i]
    return_value = (current_price - prev_price)/prev_price
    index_returns.append(return_value)

index_returns = pd.Series(index_returns)

index_returns.head()
```

Out[13]:
```
0    0.004480
1    0.000703
2    0.008567
3   -0.000714
4    0.003126
dtype: float64
```

In [14]:
```
covariance = np.cov(stock_returns,index_returns)[0,1]
variance = np.var(index_returns)
beta = covariance/variance

print("Beta:", beta)
```

```
Beta: 1.2371807701898165
```

In [24]:
```
risk_free_rate = Rf = 0.05 #for an year
total_market_return = (index_returns.mean())*len(index_returns) #or index_returns.sum()
average_market_return = Rm = total_market_return/5 #for an year

print("Rm:", Rm)
```

```
Rm: 0.10972478980737507
```

In [26]:
```
expected_return = (Rf + beta*(Rm-Rf))

print("Average Expected Return per year:", expected_return)
```

```
Average Expected Return per year: 0.1238903614533132
```

In [14]:
```
covariance = np.cov(stock_returns,index_returns)[0,1]
variance = np.var(index_returns)
beta = covariance/variance

print("Beta:", beta)
```

```
Beta: 1.2371807701898165
```

In [24]:
```
risk_free_rate = Rf = 0.05 #for an year
total_market_return = (index_returns.mean())*len(index_returns) #or index_returns.sum()
average_market_return = Rm = total_market_return/5 #for an year

print("Rm:", Rm)
```

```
Rm: 0.10972478980737507
```

In [26]:
```
expected_return = (Rf + beta*(Rm-Rf))

print("Average Expected Return per year:", expected_return)
```

```
Average Expected Return per year: 0.1238903614533132
```

Alpha:

Alpha in finance is a measure of an investment's performance relative to a benchmark, typically an index like the S&P 500. It represents the excess return an investment generates compared to the expected return as predicted by the Capital Asset Pricing Model (CAPM).

The formula to calculate alpha is:

$$\alpha = R_i - (R_f + \beta(R_m - R_f))$$

Where:

- R_i is the actual return on the investment.
- R_f is the risk-free rate.
- β is the beta of the investment.
- R_m is the actual return of the market.

Code to Calculate Alpha for a Stock:

```python
import pandas as pd
import numpy as np
import yfinance as yf

# Function to calculate alpha
def calculate_alpha(stock_ticker, market_ticker, start_date, end_date, risk_free_rate):
    # Download stock and market data
    stock_data = yf.download(stock_ticker, start=start_date, end=end_date)
    market_data = yf.download(market_ticker, start=start_date, end=end_date)

    # Calculate daily returns
    stock_data['Returns'] = stock_data['Adj Close'].pct_change()
    market_data['Market Returns'] = market_data['Adj Close'].pct_change()

    # Drop NaN values
    returns_data = pd.DataFrame({
        'Stock Returns': stock_data['Returns'],
        'Market Returns': market_data['Market Returns']
    }).dropna()

    # Calculate beta
    covariance_matrix = np.cov(returns_data['Stock Returns'], returns_data['Market Returns'])
    beta = covariance_matrix[0, 1] / covariance_matrix[1, 1]

    # Calculate expected return using CAPM
    market_mean_return = returns_data['Market Returns'].mean()
    expected_return = risk_free_rate + beta * (market_mean_return - risk_free_rate)

    # Calculate actual mean return of the stock
    actual_return = returns_data['Stock Returns'].mean()

    # Calculate alpha
    alpha = actual_return - expected_return

    return alpha

# Example usage
stock_ticker = 'AAPL'
market_ticker = '^GSPC'  # S&P 500
start_date = '2020-01-01'
end_date = '2021-01-01'
risk_free_rate = 0.01  # 1% annual risk-free rate

alpha = calculate_alpha(stock_ticker, market_ticker, start_date, end_date, risk_free_rate)
print(f'Alpha for {stock_ticker}: {alpha:.4f}')
```

Portfolio Optimization:

Portfolio optimization is the process of choosing the proportions of various assets to include in a portfolio, in such a way that the portfolio has the best possible expected level of return for its level of risk.

Key Concepts:

- **Expected Return:** The average return that is anticipated on an investment or portfolio.
- **Risk (Variance/Standard Deviation):** The measure of the dispersion of returns from the expected return.
- **Covariance:** Measures how two stocks move together.
- **Correlation:** A standardized measure of covariance.

Steps:

1. **Calculate Expected Returns:** Use historical data to estimate future returns.
2. **Calculate Covariance Matrix:** Measure how each asset moves relative to others.
3. **Optimize Portfolio:** Use mathematical methods to find the best mix of assets.

Code for Portfolio Optimization:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import yfinance as yf
from scipy.optimize import minimize

# Define stocks and download data
stocks = ["AAPL", "MSFT", "GOOGL", "AMZN"]
data = yf.download(stocks, start="2018-01-01", end="2023-01-01")["Adj Close"]
data.head()
```

[3] ✓ 2.2s Python

```
[**********100%%**********]  4 of 4 completed
```

Ticker	AAPL	AMZN	GOOGL	MSFT
Date				
2018-01-02	40.615883	59.450500	53.598984	79.936737
2018-01-03	40.608807	60.209999	54.513435	80.308746
2018-01-04	40.797447	60.479500	54.725189	81.015572
2018-01-05	41.261944	61.457001	55.450859	82.020012
2018-01-08	41.108669	62.343498	55.646633	82.103722

```python
# Calculate daily returns
returns = data.pct_change().dropna()

# Calculate expected returns and covariance matrix
expected_returns = returns.mean()
cov_matrix = returns.cov()
```

[4] ✓ 0.0s Python

```python
# Define portfolio performance function
def portfolio_performance(weights, expected_returns, cov_matrix):
    returns = np.dot(weights, expected_returns)
    volatility = np.sqrt(np.dot(weights.T, np.dot(cov_matrix, weights)))
    return returns, volatility

# Define negative Sharpe ratio function
def negative_sharpe_ratio(weights, expected_returns, cov_matrix, risk_free_rate=0.01):
    returns, volatility = portfolio_performance(weights, expected_returns, cov_matrix)
    return -(returns - risk_free_rate) / volatility

# Constraints and bounds
constraints = ({'type': 'eq', 'fun': lambda x: np.sum(x) - 1})
bounds = tuple((0, 1) for _ in range(len(stocks)))

# Initial guess (equal distribution)
initial_guess = len(stocks) * [1. / len(stocks)]

# Optimization
result = minimize(negative_sharpe_ratio, initial_guess, args=(expected_returns, cov_matrix), method='SLSQP', bounds=bounds, constraints=constraints)

# Optimal weights
optimal_weights = result.x
```

[5] ✓ 0.0s Python

```python
# Display optimal weights
optimal_weights_df = pd.DataFrame(optimal_weights, index=stocks, columns=["Weight"])
optimal_weights_df
```

[6] ✓ 0.0s Python

Value at Risk (VaR) and Conditional Value at Risk (CVaR):

Value at Risk (VaR) is a measure that quantifies the potential loss in value of a portfolio over a defined period for a given confidence interval. For example, a 1-day 95% VaR of $1 million suggests that there is a 95% chance that the portfolio will not lose more than $1 million in a day.

Conditional Value at Risk (CVaR), also known as Expected Shortfall, provides an estimate of the average loss given that the loss is beyond the VaR threshold. It

addresses some of the limitations of VaR by considering the tail end of the loss distribution.

Historical VaR and CVaR:

Historical VaR is calculated using historical return data. It does not assume any specific distribution for returns.

Historical CVaR is calculated by averaging the losses that exceed the VaR threshold.

Parametric VaR and CVaR:

Parametric VaR assumes that returns are normally distributed and calculates VaR based on the mean and standard deviation of returns.

Parametric CVaR is calculated using the tail end of the normal distribution beyond the VaR threshold.

Historical VaR and CVaR Code:

```python
import numpy as np
import pandas as pd

def historical_var(returns, confidence_level=0.95):
    """
    confidence_level: VaR confidence level (default is 0.95).
    """
    sorted_returns = np.sort(returns)
    index = int((1 - confidence_level) * len(sorted_returns))
    var = -sorted_returns[index]
    return var

def historical_cvar(returns, confidence_level=0.95):
    var = historical_var(returns, confidence_level)
    cvar = -returns[returns <= -var].mean()
    return cvar

# Example usage
returns = pd.DataFrame({
    'Asset1': np.random.normal(0.001, 0.02, 1000),
    'Asset2': np.random.normal(0.001, 0.03, 1000)
})
portfolio_returns = returns.mean(axis=1)
historical_var_value = historical_var(portfolio_returns)
historical_cvar_value = historical_cvar(portfolio_returns)

print("Historical VaR:", historical_var_value)
print("Historical CVaR:", historical_cvar_value)
```

```
[7]  ✓ 0.0s                                                    Python
...  Historical VaR: 0.028820569366007243
     Historical CVaR: 0.035334224131475744
```

Parametric VaR and CVaR Code:

```python
import scipy.stats as stats

def parametric_var(returns, confidence_level=0.95):
    """
    confidence_level: VaR confidence level (default is 0.95).
    """
    mean = np.mean(returns)
    std_dev = np.std(returns)
    var = - (mean + std_dev * stats.norm.ppf(1 - confidence_level))
    return var

def parametric_cvar(returns, confidence_level=0.95):
    mean = np.mean(returns)
    std_dev = np.std(returns)
    var = parametric_var(returns, confidence_level)
    cvar = - (mean + std_dev * (stats.norm.pdf(stats.norm.ppf(1 - confidence_level)) / (1 - confidence_level)))
    return cvar

# Example usage
parametric_var_value = parametric_var(portfolio_returns)
parametric_cvar_value = parametric_cvar(portfolio_returns)

print("Parametric VaR:", parametric_var_value)
print("Parametric CVaR:", parametric_cvar_value)
```

✓ 0.0s Python

```
Parametric VaR: 0.028108195968458986
Parametric CVaR: -0.038903257508698
```

MEAN REVERSION

Mean reversion is a financial theory that suggests asset prices will eventually return to their long-term mean or average.

Z SCORE

Z-score is a statistical measurement that describes a value's relationship to the mean of a group of values. Z-score is measured in terms of standard deviations from the mean. In investing and trading, Z-scores are measures of an instrument's variability and can be used by traders to help determine volatility.

Z-Score Formula

The statistical formula for a value's z-score is calculated using the following formula**:**

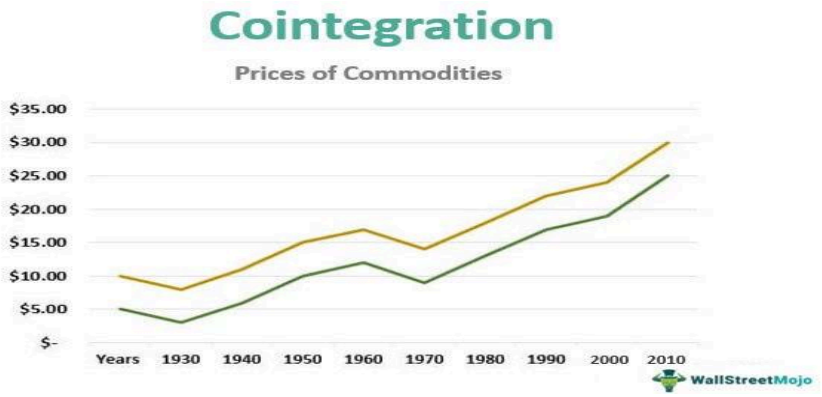
$$z = (x - \mu) / \sigma$$

Where:

- *z = Z-score*
- *x = the value being evaluated*
- *μ = the mean*
- *σ = the standard deviation*

What is Cointegration?

Cointegration is a statistical method used to test the correlation between two or more non-stationary time series in the long run or for a specified period. The method helps identify long-run parameters or equilibrium for two or more variables. In addition, it helps determine the scenarios wherein two or more stationary time series are cointegrated so that they cannot depart much from the equilibrium in the long run.



Stationary time series

A time series whose statistical properties, such as mean, variance, etc., remain constant over time, are called a stationary time series.

In other words, a stationary time series is a series whose statistical properties are independent of the point in time at which they are observed. A stationary time series has a constant variance and it always returns to the long-run mean.

Hedging

A hedge is an investment that is selected to reduce the potential for loss in other investments because its price tends to move in the opposite direction. This strategy works as a kind of insurance policy, offsetting any steep losses in other investments.

The term hedging can be used to describe diversifying a portfolio by buying shares in a conservative bond fund to offset potential losses in more volatile stock funds.

Tests- ADF, Granger causality, KPSS

ADF TEST

The Augmented Dickey-Fuller (ADF) test is a statistical test used in econometrics and finance to determine whether a unit root is present in a time series dataset. A unit root implies that a series is non-stationary, i.e., its statistical properties, such as mean and variance, are not constant over time.

Code for ADF test:

[2]
```
import yfinance as yf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

[3]
```
#Stocks Tickers are

tickers_array=["MARUTI.NS","TCS.NS","WIPRO.NS","TATASTEEL.NS","ITC.NS"]
beta_array=[0]*5
df1=pd.DataFrame(yf.download(tickers_array[0],start="2022-06-04",end="2024-06-04",period="1d"))
```

[20]
```
# ADF TEST
from statsmodels.tsa.stattools import adfuller
df1["returns"]=(df1["Close"]-df1["Open"])/df1["Open"]
adfuller(df1["returns"])
```

```
(-21.863296723114253,
 0.0,
 0,
 490,
 {'1%': -3.4437660979098843,
  '5%': -2.8674565460819896,
  '10%': -2.569921291128696},
 -2826.7910104261)
```

If p value is less than .05 then we can say time series is stationary

Granger Causality test

The **Granger Causality test** is used to determine whether or not one time series is useful for forecasting another.

This test uses the following null and alternative hypotheses:

Null Hypothesis (H0): Time series *x* does not Granger-cause time series *y*

Alternative Hypothesis (HA): Time series *x* Granger-causes time series *y*

The term “Granger-causes” means that knowing the value of time series *x* at a certain lag is useful for predicting the value of time series *y* at a later time period.

This test produces an F test statistic with a corresponding p-value. If the p-value is less than a certain significance level (i.e. α = .05), then we can reject the null hypothesis and conclude that we have sufficient evidence to say that time series *x* Granger-causes time series y.

```
from statsmodels.tsa.stattools import grangercausalitytests

#perform Granger-Causality test
grangercausalitytests(df[['column1', 'column2']], maxlag=[3])
```

```
import pandas as pd

#define URL where dataset is located
url = "https://raw.githubusercontent.com/Statology/Miscellaneous/main/chicken_egg.txt"

#read in dataset as pandas DataFrame
df = pd.read_csv(url, sep="  ")

#view first five rows of DataFrame
df.head()
```

	year	chicken	egg
0	1930	468491	3581
1	1931	449743	3532
2	1932	436815	3327
3	1933	444523	3255
4	1934	433937	3156

```
from statsmodels.tsa.stattools import grangercausalitytests

#perform Granger-Causality test
grangercausalitytests(df[['chicken', 'egg']], maxlag=[3])

Granger Causality
number of lags (no zero) 3
ssr based F test:         F=5.4050  , p=0.0030  , df_denom=44, df_num=3
ssr based chi2 test:   chi2=18.7946 , p=0.0003  , df=3
likelihood ratio test: chi2=16.0003 , p=0.0011  , df=3
parameter F test:         F=5.4050  , p=0.0030  , df_denom=44, df_num=3
```

The F test statistic turns out to be **5.405** and the corresponding p-value is **0.0030**.

KPSS TEST

The KPSS test, short for, Kwiatkowski-Phillips-Schmidt-Shin (KPSS), is a type of Unit root test that tests for the stationarity of a given series around a deterministic trend.

That is, if p-value is < significant level (say 0.05), then the series is non-stationary. Whereas in ADF test, it would mean the tested series is stationary.

[25]

```
# KPSS Test
import statsmodels.api as sm
sm.tsa.stattools.kpss(df1["Close"], regression='ct')
```

```
C:\Users\rakes\AppData\Local\Temp\ipykernel_37988\3339696010.py:3: InterpolationWarning: The test statistic is outside of the range of p-values
look-up table. The actual p-value is smaller than the p-value returned.

  sm.tsa.stattools.kpss(df1["Close"], regression='ct')
```

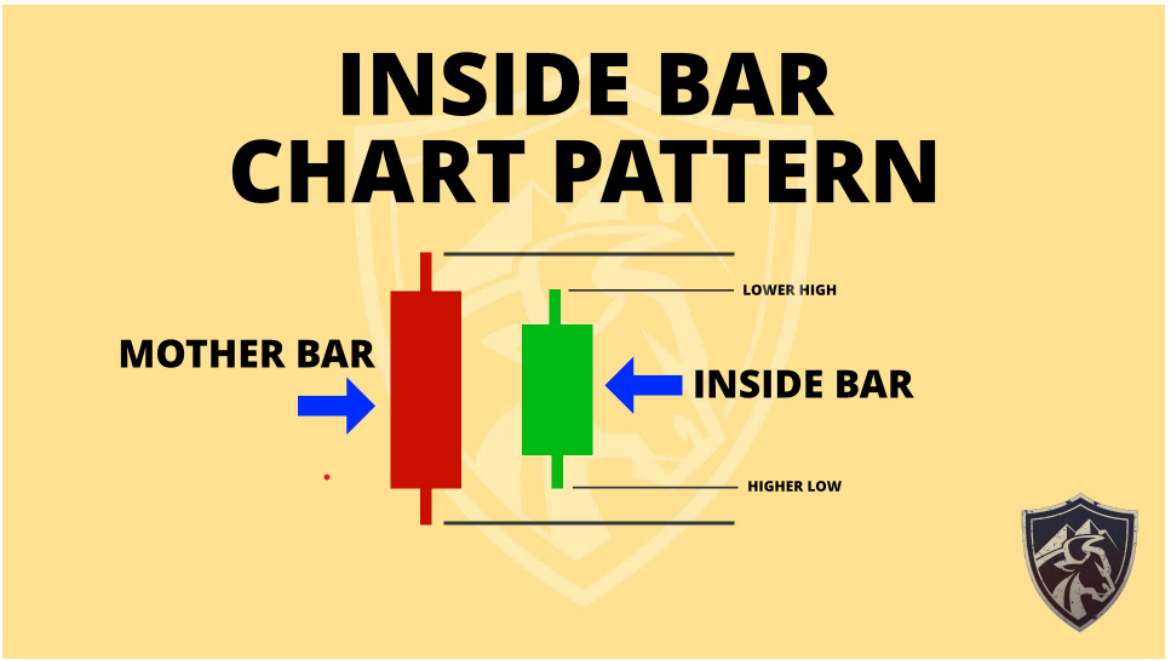
```
(0.5748066848145552,
 0.01,
 12,
 {'10%': 0.119, '5%': 0.146, '2.5%': 0.176, '1%': 0.216})
```

P-SAR

The parabolic indicator generates buy or sell signals when the position of the dots moves from one side of the asset's price to the other. For example, a buy signal occurs when the dots move from above the price to below the price, while a sell signal occurs when the dots move from below the price to above the price.

Traders also use the PSAR dots to set trailing stop loss orders. For example, if the price is rising, and the PSAR is also rising, the PSAR can be used as a possible exit if long. If the price drops below the PSAR, exit the long trade.

INSIDE BAR



An Inside Bar (or candle) is a 2-bar pattern where a bar is inside the total price action of the previous bar. In other words, the Inside Bar has a higher low and lower high than the previous bar. When this happens the previous bar is known as the mother

bar. It does not matter if the Inside Bar is bullish or bearish, all that matters is where the Inside Bar prints relative to existing price action.

Why Inside Bars Form

There can be many things that can cause an Inside Bar to form.

- A big news event might be coming up
- The traders who are in control (bulls or bears) are taking a break before continuing their campaign
- The traders who are in control (bulls or bears) are losing control, or are ready to stop their campaign, and price will reverse soon
- And more!

You don't need to know *why* Inside Bars happen, you just have to understand what the price action is telling you.

As you probably know, when price action starts to consolidate, it usually means that there will be a breakout.

Pivot Point Reversal Strategy

The pivot point reversal strategy is a trading approach used in the foreign exchange (forex) market. This strategy involves identifying significant support and resistance levels on a price chart, which are then used to determine potential entry and exit points for trades. Pivot points are calculated using the previous day's high, low, and closing prices, and are used to gauge potential price movements for the current trading day. In this article, we will explore the key components of the pivot point reversal strategy and how it can be applied in the forex market.

Pivot Point Reversal Strategy

Here's an example strategy using the pivot point reversal strategy for the forex market:

- Determine the pivot points: Calculate the pivot points for the current trading day using the previous day's high, low, and closing prices. The central pivot point is the most significant level, and it acts as a key support or resistance level for the day. The other pivot points are support and resistance levels that are above and below the central pivot point.
- Identify the trend: Determine the direction of the trend by analysing the price action on the chart. If the trend is bullish, look for buying opportunities. If the trend is bearish, look for selling opportunities.
- Wait for a price reversal: Once the trend has been identified, wait for a price reversal to occur at one of the pivot levels. This could be in the form of a candlestick pattern or a price rejection at a pivot level.

Buy Signal

pivot point reversal strategy Buy Signal

Here are the details for a buy signal using the pivot point reversal strategy:

- Identify the central pivot point (PP) and support levels (S1, S2, S3) below the current price on the chart.
- Look for a bullish trend or a potential trend reversal based on technical analysis.

- Wait for the price to reach a support level (S1, S2, or S3) and observe if it holds as a support level.
- Look for bullish candlestick patterns, such as bullish hammer, bullish engulfing, or bullish harami, at or near the support level.

Sell Signal

pivot point reversal strategy Sell Signal

Here are the details for a sell signal using the pivot point reversal strategy:

- Identify the central pivot point (PP) and resistance levels (R1, R2, R3) above the current price on the chart.
- Look for a bearish trend or a potential trend reversal based on technical analysis.
- Wait for the price to reach a resistance level (R1, R2, or R3) and observe if it holds as a resistance level.
- Look for bearish candlestick patterns, such as shooting star, bearish engulfing, or bearish harami, at or near the resistance level.

PIVOT EXTENSION

The Pivot Extension Strategy is based on Pivot Points. The strategy scans the last X+Y bars (where X and Y are leftBars and rightBars inputs in the indicator settings)

to find a pivot. When a high pivot is located, the strategy will open a short position. If a low pivot is found, it will enter a long.

What Is ADX?

The average directional index (ADX) is a technical indicator used by traders to determine the strength of a price trend for a financial security. Trading in the direction of a strong trend reduces risk and increases profit potential. Many traders consider the ADX to be the ultimate trend indicator because it is so reliable.

ADX quantifies trend strength. ADX calculations are based on a moving average of price range expansion over a given period of time. The default setting is 14 bars, although other time periods can be used.

ADX can be used with any financial security that trades, including stocks, mutual funds, exchange-traded funds, and futures.

- The average directional index, known as ADX, is a technical tool used by traders to identify trend strength.
- Trading with the trend is a longstanding and proven trading practice that reduces risk and increases profit potential.
- ADX calculations use a moving average of price range expansion.
- ADX isn't useful when prices enter a trading range.
- While ADX is a lagging indicator, it is considered very reliable.

How ADX Works

ADX is plotted as a single line with values ranging from a low of zero to a high of 100. ADX is non-directional; it registers trend strength whether price is trending up or down.

The indicator is usually plotted in the same window as the two directional movement indicator (DMI) lines, from which ADX is derived (shown below).

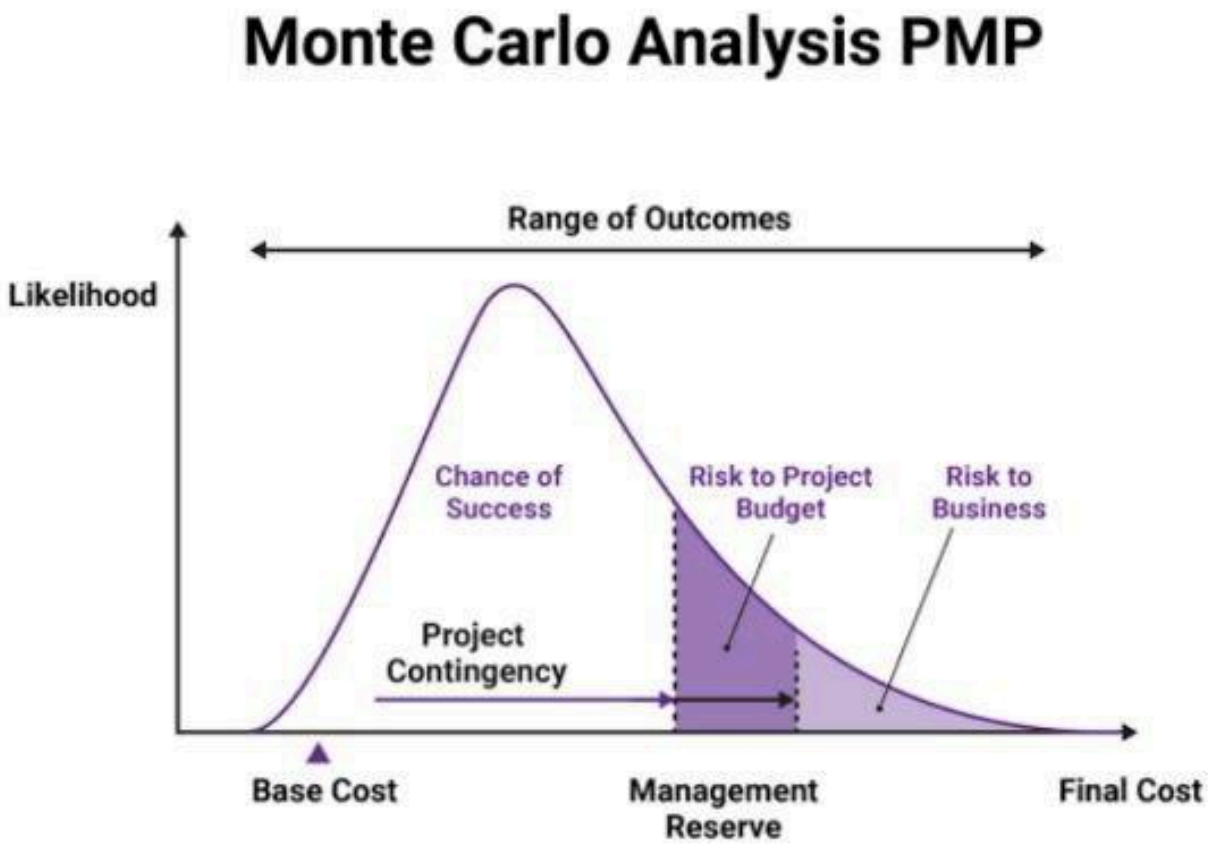
ADX is non-directional and quantifies trend strength by rising in both uptrends and downtrends.

Monte Carlo Simulation

The Monte Carlo simulation is a mathematical technique that predicts possible outcomes of an uncertain event. Computer programs use this method to analyze past data and predict a range of future outcomes based on a choice of action.
The Monte Carlo simulation provides multiple possible outcomes and the probability of each from a large pool of random data samples.
It offers a clearer picture than a deterministic forecast.
For instance, forecasting financial risks requires analyzing dozens or hundreds of risk factors. Financial analyst use the Monte Carlo simulation to produce the probability of every possible outcome.

Monte Carlo in finance:
Financial analysts often make long-term forecasts on stock prices and then advise their clients of appropriate strategies. While doing so, they must consider market factors that could cause drastic changes to the investment value. As a result, they

use the Monte Carlo simulation to predict probable outcomes to support their strategies.



Let’s use the Monte Carlo simulation to calculate pi, denoted as π.

```python
# import libraries
import numpy as np

# initialize variables
n_simulations = 100000
n_points_circle = 0
n_points_square = 0

# create lists to store x and y values
l_xs = []
l_ys = []

# loop n_simulations times
for _ in range(n_simulations):

    # x is randomly drawn from a continuous uniform distritbuion
    x = np.random.uniform(-1, 1)
    # store x in the list
    l_xs.append(x)

    # y is randomly drawn from a continuous uniform distribution
    y = np.random.uniform(-1, 1)
    # store y in the list
    l_ys.append(y)

# loop n_simulations times
for i in range(n_simulations):

    # calculate the distance between the point and the origin
    dist_from_origin = l_xs[i] ** 2 + l_ys[i] ** 2

    # if the distance is smaller than or equal to 1, the point is in the circle
    if dist_from_origin <= 1:
        n_points_circle += 1

    # by definition of the uniform distribution, the point is in the square
    n_points_square += 1

# estimate the value of pi
pi = 4 * n_points_circle / n_points_square
print(pi)
```

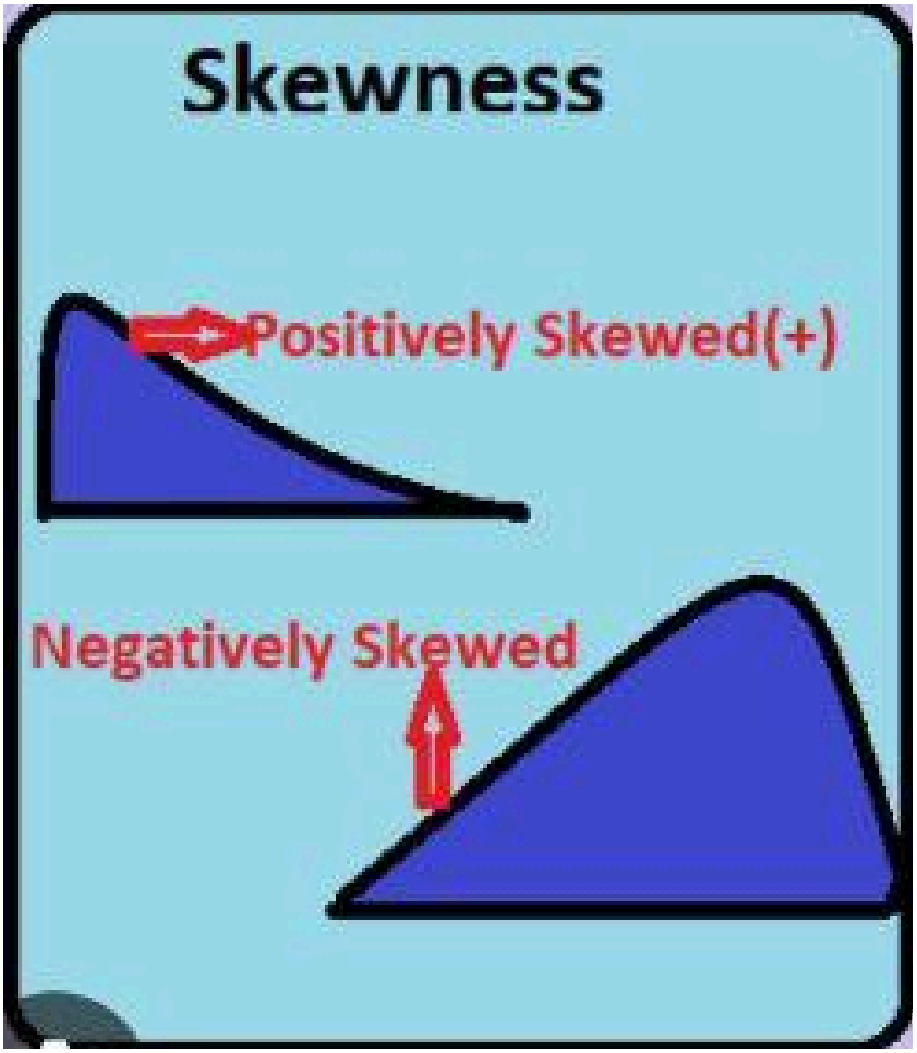
```
3.13836
```

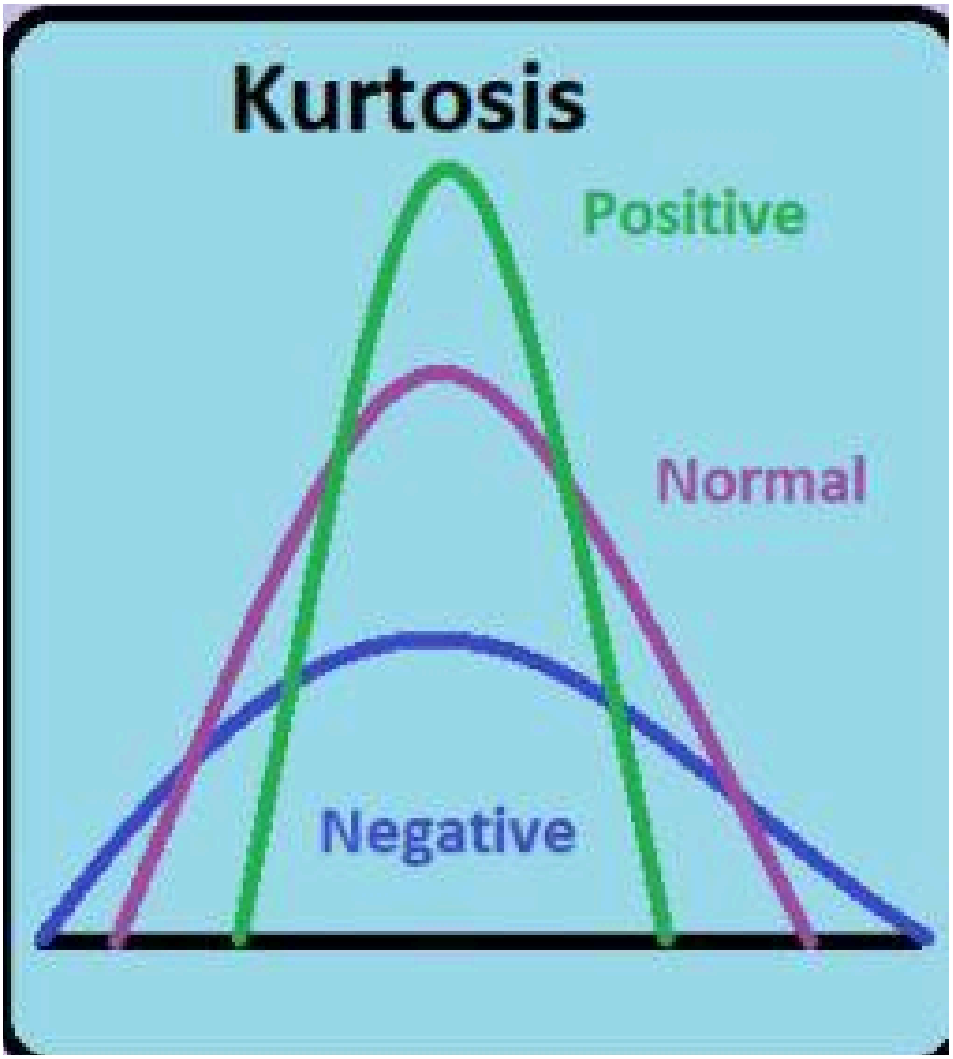
Jarque-Bera test : The Jarque-Bera Test,a type of Lagrange multiplier test, is a test for normality. Normality is one of the assumptions for many statistical tests, like the t test or F test; the Jarque-Bera test is usually run before one of these tests to confirm normality. It is usually used for large data sets, because other normality tests are not reliable when n is large.

Specifically, the test matches the skewness and kurtosis of data to see if it matches a normal distribution. The data could take many forms, including:

- Time Series Data.
- Errors in a regression model.
- Data in a Vector

A normal distribution has a skew of zero (i.e. it's perfectly symmetrical around the mean) and a kurtosis of three; kurtosis tells you how much data is in the tails and gives you an idea about how "peaked" the distribution is. It's not necessary to know the mean or the standard deviation for the data in order to run the test.





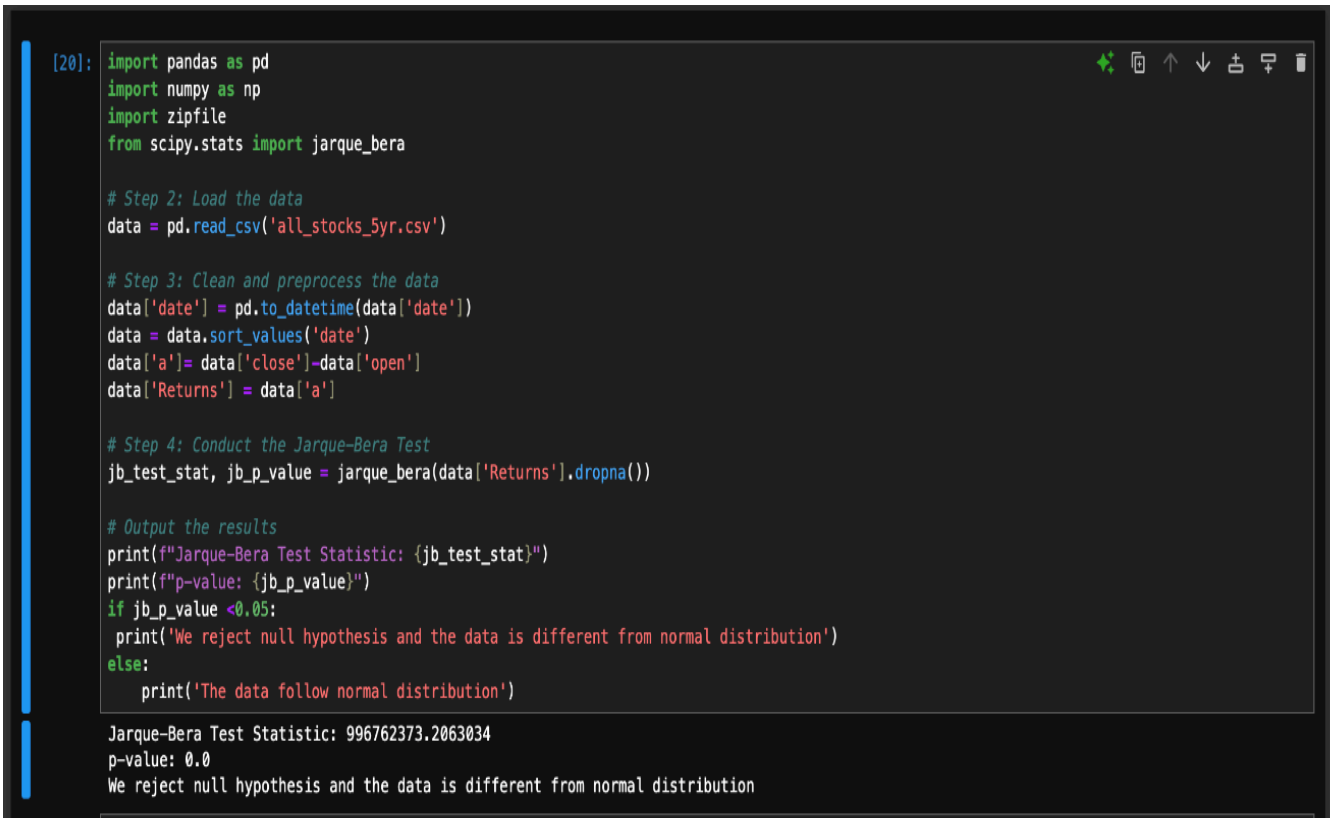
The formula for the Jarque-Bera test statistic (usually shortened to just **JB test statistic**) is:

JB = n [(√b1)2 / 6 + (b2 – 3)2 / 24].
Where:
n is the sample size,
√b1 is the sample skewness coefficient,
b2 is the kurtosis coefficient.

The **null hypothesis** for the test is that the data is normally distributed

The **alternate hypothesis** is that the data does not come from a normal distribution.

Here's a Python code snippet that demonstrates how to perform Jarque-Bera test:



```
[20]: import pandas as pd
      import numpy as np
      import zipfile
      from scipy.stats import jarque_bera

      # Step 2: Load the data
      data = pd.read_csv('all_stocks_5yr.csv')

      # Step 3: Clean and preprocess the data
      data['date'] = pd.to_datetime(data['date'])
      data = data.sort_values('date')
      data['a']= data['close']-data['open']
      data['Returns'] = data['a']

      # Step 4: Conduct the Jarque-Bera Test
      jb_test_stat, jb_p_value = jarque_bera(data['Returns'].dropna())

      # Output the results
      print(f"Jarque-Bera Test Statistic: {jb_test_stat}")
      print(f"p-value: {jb_p_value}")
      if jb_p_value <0.05:
       print('We reject null hypothesis and the data is different from normal distribution')
      else:
          print('The data follow normal distribution')

      Jarque-Bera Test Statistic: 996762373.2063034
      p-value: 0.0
      We reject null hypothesis and the data is different from normal distribution
```

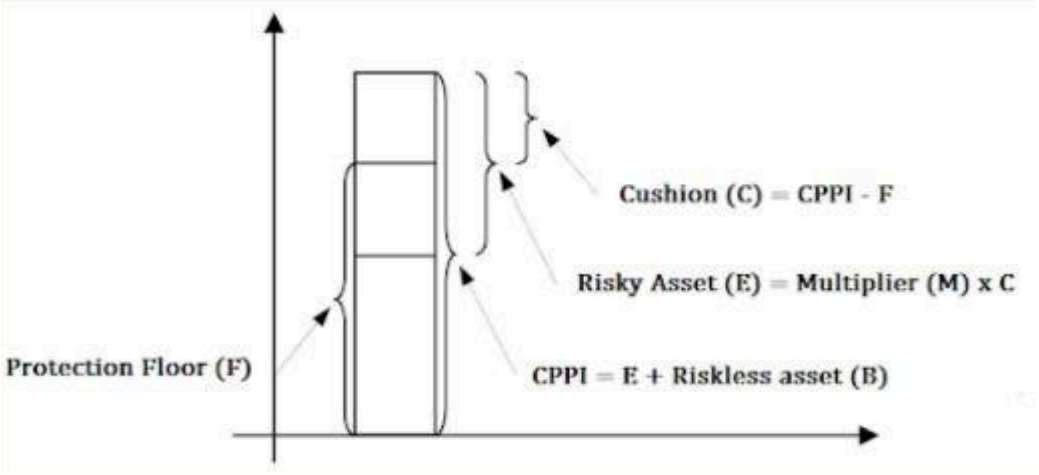
CPPI TEST

CPPI (Constant Proportion Portfolio Insurance) is a strategy that allows an investor to keep exposure to a risky asset's upside potential while providing a guarantee against the downside risk by dynamically scaling the exposure.

Imagine a portfolio consisting of two types of assets, "safe" (with a given yield y) and "risky". Firstly, choose the protection level of the portfolio, i.e. the minimum value you want to protect (for example its starting value, s) and the protection period in years, T. It's then easy to calculate how much may the portfolio value decline at each point in time t, so that the safe asset will be able to recover the loss back to the protection level.

$$F_t = \frac{s}{(1+y_t)^{(T-t)}}$$

This is the floor (the value, you don't want to go below). The difference between the current portfolio value and the floor is then called cushion; it is the value you can put at risk.
At every point in time, you can allocate multiplier M of the cushion to the risky assets

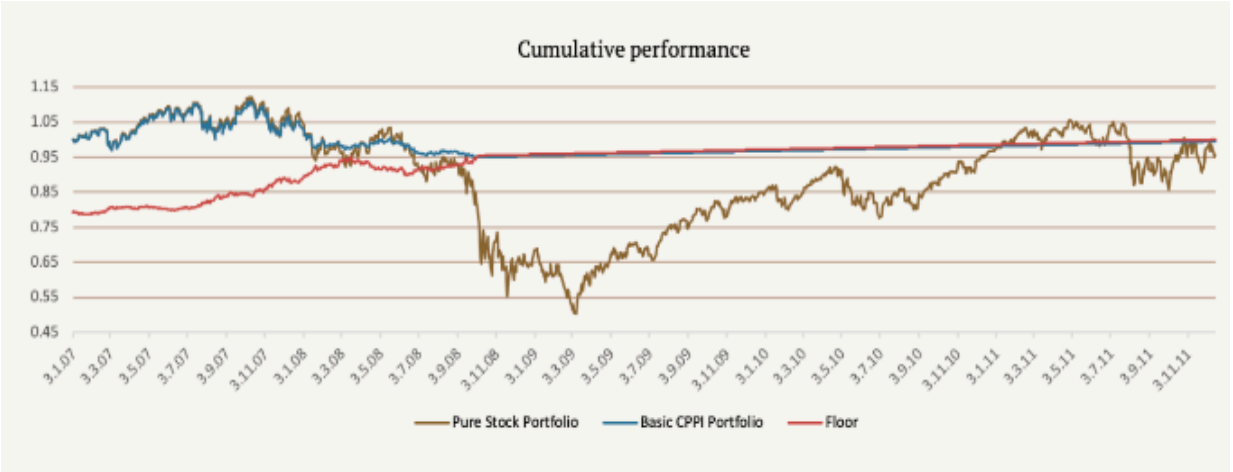


CPPI procedure creates convex option-like payoffs without actually using options. The most significant benefit of CPPI strategy is that if you were able to trade continuously, nothing could go wrong
The loss of the risky component can become large enough that you get below the floor between two trading dates. The risk of breaching the floor this way is called “the gap risk”. That gap risk materializes if and only if the loss on the risky asset exceeds 1/M within the trading interval. Because of this, it is recommended to set the multiplier as a function of the maximum potential loss within a given trading interval.

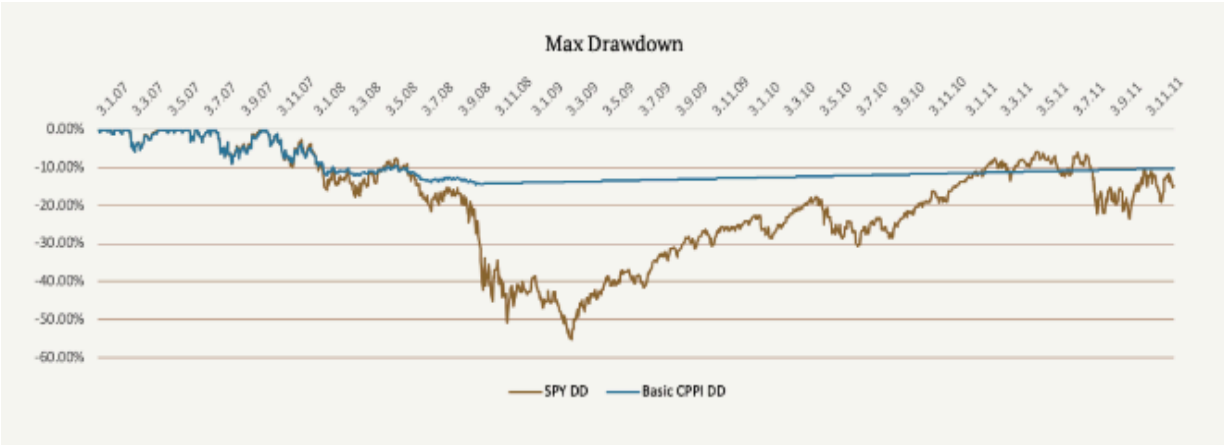
The table below shows the risk characteristics of the Basic CPPI portfolio compared to the pure SPY portfolio.

	5Y CAR (p.a.)	5Y Volatility (p.a.)	Sharpe Ratio	Max DD	95% DD	CAR/ max DD	CAR/ 95% DD
Basic CPPI	-0.09%	7.02%	-0.013	-14.55%	-14.01%	0.006	0.007
SPY	-0.91%	26.49%	-0.034	-55.19%	-44.28%	0.016	0.020

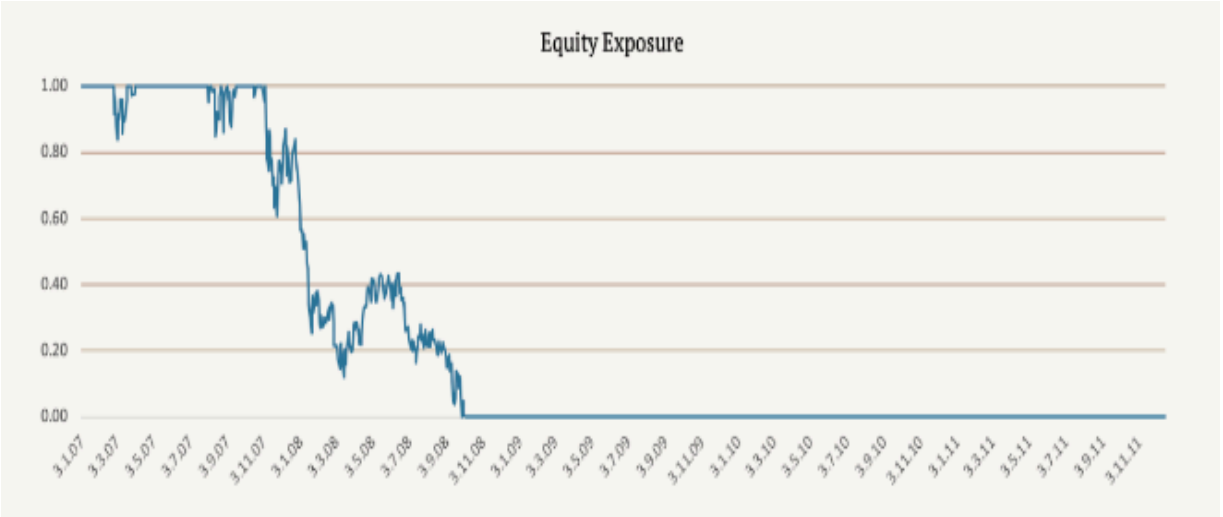
The following figure shows the cumulative performance of the Basic CPPI portfolio and pure SPY portfolio, as well as evolution of the floor in time.



The chart above shows that the Basic CPPI portfolio hit the floor in October 2008, but it managed to outperform the pure SPY portfolio. The next figure shows the drawdowns of both of these portfolios.



We can see that SPY's drawdown is much greater than the basic CPPI portfolio drawdown, which is the most significant benefit of using the CPPI strategy. The following figure shows the equity exposure during the 5-year period.



Scenario analysis is the process of estimating the expected value of a portfolio after a given period of time, assuming specific changes in the values of the portfolio's securities or keyThese assessments can be used to examine the amount of risk present within a given investment as related to a variety of potential events, ranging from highly probable to highly improbable. Depending on the results of the analysis, an investor can determine if the level of risk present falls within their comfort zone.

Scenario Analysis and Investment Strategy: A common method is to determine the standard deviation of daily or monthly security returns and then compute what value is expected for the portfolio if each security generates returns that are two or three standard deviations above and below the average return. This way, an analyst can have a reasonable amount of certainty regarding the change in the value of a portfolio during a given time period, by simulating these extremes.

Scenario Analysis in Personal Finance : The same process used for examining potential investment scenarios can be applied to various other financial situations in order to examine value shifts based on theoretical scenarios. On the consumer side, a person can use scenario analysis to examine the different financial outcomes of purchasing an item on credit, as opposed to saving the funds for a cash purchase. Additionally, a person can look at the various financial changes that may occur when deciding whether to accept a new job offer.

Scenario Analysis in Corporate Finance: Businesses can also use scenario analysis to analyze the potential financial outcomes of certain decisions, such as selecting one of two facilities or storefronts from which the business could operate. This could include considerations such as the difference in rent, utility charges, and insurance, or any benefit that may exist in one location but not the other.

Stress Testing is a computer simulation technique used to test the resilience of institutions and investment portfolios against possible future financial situations. Such testing is customarily used by the financial industry to help gauge investment risk and the adequacy of assets and help evaluate internal processes and controls. In recent years, regulators have also required financial institutions to carry out stress tests to ensure their capital holdings and other assets are adequate.

The Federal Reserve requires banks of a certain size to perform stress tests, such as the Dodd-Frank Act Stress Test (DFAST) or the Comprehensive Capital Analysis and Review (CCAR). These tests review the bank's capital and how well it can meet obligations and operate during trying economic times.

Stress testing is often performed using computer simulations, running different scenarios. Companies might use historical events, hypothetical situations, or simulations to test how well a company would operate under specific conditions.

Stress tests can be effective analytical tools in identifying whether a company has sufficient capital, strong assets, and effective plans to weather an economic storm. Companies can use historical, hypothetical, or simulated events to create test scenarios, or they may be required by a regulatory body to perform certain tests. The

results can help companies better understand their strengths, weaknesses, and areas of opportunity.