|  |  |
| --- | --- |
| Project Title | **Coca Cola Stock - Live and Updated** |
| Tools | ML, Python, SQL, Excel |
| Domain | Data Analyst, |
| Project Difficulties level | intermediate |

Dataset : Dataset is available in the given link. You can download it at your convenience.

[Click](https://drive.google.com/drive/folders/19oBqp11Af9MDTz8ODGcmv7_Nj_L9tLNp?usp=sharing) [here](https://drive.google.com/drive/folders/19oBqp11Af9MDTz8ODGcmv7_Nj_L9tLNp?usp=sharing) [to](https://drive.google.com/drive/folders/19oBqp11Af9MDTz8ODGcmv7_Nj_L9tLNp?usp=sharing) [download](https://drive.google.com/drive/folders/19oBqp11Af9MDTz8ODGcmv7_Nj_L9tLNp?usp=sharing) [data](https://drive.google.com/drive/folders/19oBqp11Af9MDTz8ODGcmv7_Nj_L9tLNp?usp=sharing) [set](https://drive.google.com/drive/folders/19oBqp11Af9MDTz8ODGcmv7_Nj_L9tLNp?usp=sharing)

# About Dataset

The Coca-Cola Company is an North American multinational beverage corporation incorporated under Delaware's General Corporation Law[a] and headquartered in Atlanta, Georgia. The Coca-Cola Company has interests in the manufacturing, retailing, and marketing of non-alcoholic beverage concentrates and syrups, and alcoholic beverages. The company produces Coca-Cola, the sugary drink for which it is best known for, invented in 1886 by pharmacist John Stith Pemberton. At the time, the product was made with coca leaves, which added an amount of cocaine to the drink, and with kola nuts, which added caffeine, so that the coca and the kola together provided a stimulative effect. This stimulative effect is the reason the drink was sold to the public as a healthy "tonic", and the coca and the kola are also the source of the name of the product and of the company.In 1889, the formula and brand were sold for $2,300 (roughly $68,000 in 2021) to Asa Griggs Candler, who incorporated The Coca-Cola Company in Atlanta in 1892.

Since 1919, Coca-Cola has been a publicly traded company. Its stock is listed on the New York Stock Exchange under the ticker symbol "KO". One share of stock purchased in 1919 for $40, with all dividends reinvested, would have been worth $9.8 million in 2012, a 10.7% annual increase adjusted for inflation. A predecessor bank of SunTrust received $100,000 for underwriting Coca-Cola's 1919 public offering; the bank sold that stock for over $2 billion in 2012. In 1987, Coca-Cola once again became one of the 30 stocks which makes up the Dow Jones Industrial Average, which is commonly referenced as a proxy for stock market performance; it had previously been a Dow stock from 1932 to 1935. Coca-Cola has paid a dividend since 1920 and, as of 2019, had increased it each year for 57 years straight.

## Example: You can get the basic idea how you can create a project from here

|  |
| --- |
| To create a major ML project using Coca-Cola stock data with the specified columns, here's a structured step-by-step guide. This explanation will include **EDA**, **data cleaning**, **data visualization**, **handling missing values**, **statistical operations**, and more. The project is tailored for an experienced developer.  **Step 1: Problem Definition**   * **Objective**: Predict Coca-Cola's stock prices (e.g., Close price) and analyze trends. * **Data**: Historical data with Date, Open, High, Low, Close, Volume, Dividends, Stock Splits. * **Deliverables**: |

1.

Insights

from

the

data

(

visualizations

and

statistics).

2.

An

ML

model

to

predict

stock

prices.

3.

A

live-updating

system

for

predictions.

**Step**

**2:**

**Data**

**Collection**

Use

**Yahoo**

**Finance**

**API**

for

historical

data.

We'll

fetch

data

from

2015

to

the

present.

**Code:**

python

code

import

yfinance

as

yf

import

pandas

as

pd

#

Fetch

Coca-Cola

stock

data

ticker

=

'KO'

#

Coca-Cola

stock

ticker

data

=

yf.download(ticker,

start='2015-01-01',

end='2023-12-31')

#

Reset

index

for

easier

handling

data.reset\_index(inplace=True)

#

Display

data

structure

print(data.info())

print(data.head())

|  |
| --- |
| **Step 3: Data Cleaning**  **3.1 Handle Missing Values**  ● Check for missing values and decide on imputation or deletion.  **Code:** python code  # Check for missing values print(data.isnull().sum())  # Fill missing numerical values with the column mean data.fillna(method='ffill', inplace=True) # Forward fill for stock data continuity data.fillna(0, inplace=True) # Replace remaining missing dividends/splits with 0  # Confirm no missing values remain print(data.isnull().sum())  **Step 4: Feature Engineering**   1. **Add Moving Averages**: 20-day and 50-day for trend detection. 2. **Add Daily Returns**: Helps capture volatility. |

3.

**Add**

**Volatility**

:

Standard

deviation

over

a

rolling

window.

**Code:**

python

code

#

Add

Moving

Averages

data['MA\_20']

=

data['Close'].rolling(window=20).mean()

data['MA\_50']

=

data['Close'].rolling(window=50).mean()

#

Add

Daily

Returns

data['Daily\_Return']

=

data['Close'].pct\_change()

#

Add

Volatility

(

standard

deviation

of

returns

over

a

rolling

window)

data['Volatility']

=

data['Daily\_Return'].rolling(window=20).std()

#

Drop

rows

with

NA

due

to

rolling

calculations

data.dropna(inplace=True)

print(data.head())

**Step**

**5:**

**Exploratory**

**Data**

**Analysis**

**(**

**EDA**

**)**

**5.1**

**Summary**

**Statistics**

|  |
| --- |
| Use descriptive statistics to summarize the data.  **Code:** python code  # Summary statistics print(data.describe())  **5.2 Data Visualization**  Visualize the trends and relationships in the data.  **Code:** python code  import matplotlib.pyplot as plt import seaborn as sns  # Line plot for stock prices plt.figure(figsize=(12, 6)) plt.plot(data['Date'], data['Close'], label='Close Price') plt.plot(data['Date'], data['MA\_20'], label='MA 20', linestyle='--') plt.plot(data['Date'], data['MA\_50'], label='MA 50', linestyle='--') plt.title('Coca-Cola Stock Prices with Moving Averages') plt.xlabel('Date') |

plt.ylabel('Price')

plt.legend()

plt.show()

#

Correlation

heatmap

plt.figure(figsize=(10,

8))

sns.heatmap(data.corr(),

annot=True,

cmap='coolwarm')

plt.title('Correlation

Heatmap')

plt.show()

**Step**

**6:**

**Data**

**Splitting**

Split

the

data

into

training

and

testing

sets

for

model

training.

**Code:**

python

code

from

sklearn.model\_selection

import

train\_test\_split

#

Features

and

target

features

=

'Open',

[

'High',

'Low',

'Volume',

'Dividends',

'Stock

Splits',

'MA\_20',

'MA\_50',

'Daily\_Return',

'Volatility']

target

=

'Close'

X

=

data[features]

|  |
| --- |
| y = data[target]  # Train-test split  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42, shuffle=False)  **Step 7: Model Training**  **7.1 Use Random Forest for Initial Predictions**  A good baseline model for tabular data is Random Forest.  **Code:** python code  from sklearn.ensemble import RandomForestRegressor from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error  # Initialize the model model = RandomForestRegressor(n\_estimators=100, random\_state=42)  # Train the model model.fit(X\_train, y\_train) |

#

Predict

on

test

set

y\_pred

=

model.predict(X\_test)

#

Evaluate

model

mse

=

mean\_squared\_error(y\_test,

y\_pred)

mae

=

mean\_absolute\_error(y\_test,

y\_pred)

print(f"Mean

Squared

Error:

{

mse}"

)

print(f"Mean

Absolute

Error:

{

mae}"

)

**Step**

**8:**

**Live**

**Prediction**

**System**

Integrate

a

system

to

fetch

live

data

and

make

predictions.

**Fetching**

**Live**

**Data:**

python

code

#

Fetch

latest

stock

data

live\_data

=

yf.download(ticker,

period='1d',

interval='1m')

#

Prepare

live

data

for

prediction

live\_data['MA\_20']

=

live\_data['Close'].rolling(window=20).mean()

live\_data['MA\_50']

=

live\_data['Close'].rolling(window=50).mean()

|  |
| --- |
| live\_data['Daily\_Return'] = live\_data['Close'].pct\_change() live\_data['Volatility'] = live\_data['Daily\_Return'].rolling(window=20).std()  # Ensure no missing values live\_data.fillna(0, inplace=True)  # Use the latest data point for prediction latest\_features = live\_data[features].iloc[-1:].dropna() live\_prediction = model.predict(latest\_features) print(f"Predicted Closing Price: {live\_prediction[0]}")  **Step 9: Deploy the System**  Deploy using **Streamlit** or **Flask** for a web-based dashboard. Example with Streamlit:  **Code:** python code  import streamlit as st st.title('Coca-Cola Stock Price Prediction')  # Upload visualization |

|  |
| --- |
| st.line\_chart(data[['Close', 'MA\_20', 'MA\_50']])  # Show prediction st.write(f"Predicted Closing Price: {live\_prediction[0]}") |

## Sample Code and output

|  |
| --- |
| Stock Analysis of Coca Cola Stock[¶](https://www.kaggle.com/code/kalilurrahman/coca-cola-stock-analysis#Stock-Analysis-of-Coca-Cola-Stock)  1. Import Libraries  In [2]:  import pandas as pd import numpy as np  import matplotlib.pyplot as plt import seaborn as sns sns.set\_style('whitegrid') plt.style.use("fivethirtyeight") %matplotlib inline  *# For reading stock data from yahoo* from pandas\_datareader.data import DataReader  *# For time stamps* from datetime import datetime from math import sqrt from math import sqrt |

from

sklearn.metrics

import

mean\_squared\_error

from

sklearn.preprocessing

import

MinMaxScaler

*#ignore*

*the*

*warnings*

import

warnings

warnings

.

filterwarnings(

'ignore'

)

2.

Load

Dataset

In

[3]:

KO\_Data

=

pd

.

read\_csv(

'../input/coca-cola-stock-live-and-updated/Coca-Col

a\_stock\_history.csv'

)

3.

Basic

EDA

In

[4]:

KO\_Data

.

plot(subplots

=

True

,

figsize

=

(

10

,

12

))

plt

.

title(

'Coca

Cola

Stock

Attributes'

)

plt

.

show()

In

[5]:

def

plot\_close\_val(data\_frame,

column,

stock):

plt

.

figure(figsize

=

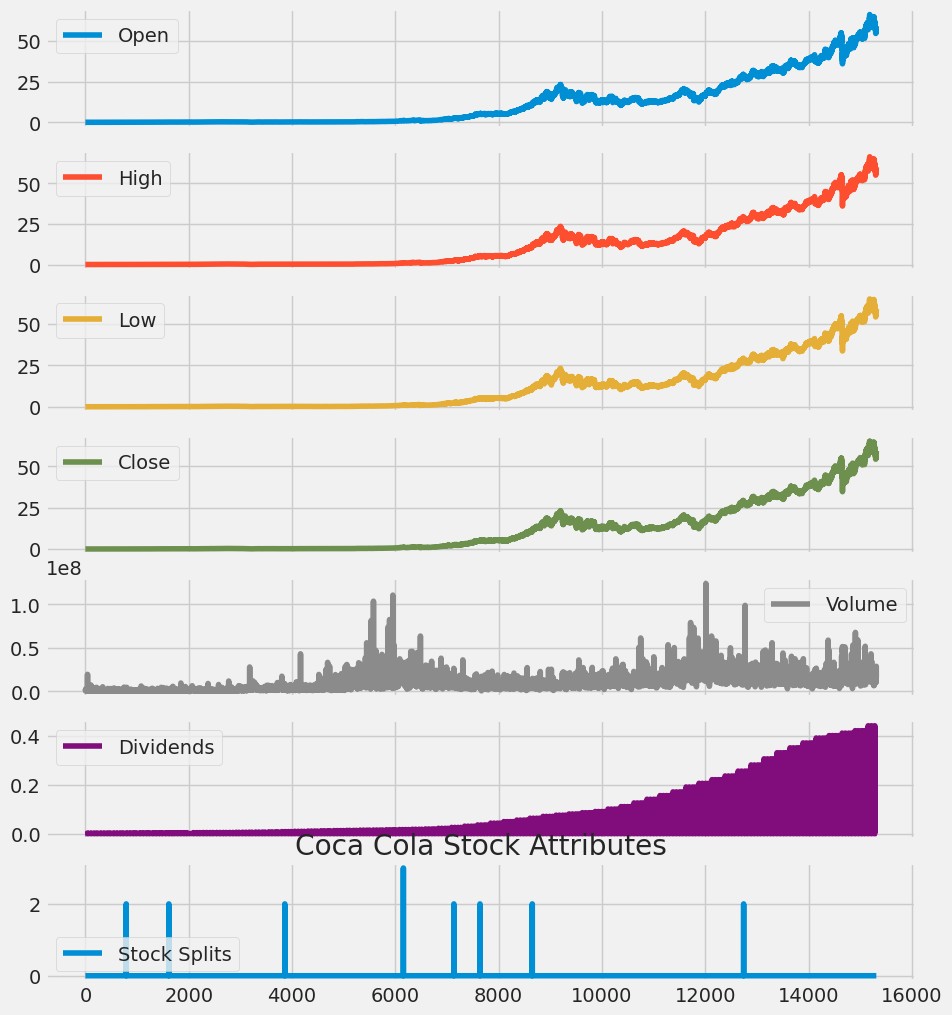
(

16

,

6

))



plt

.

title(column

+

'

Price

History

for

'

+

stock

)

plt

.

plot(data\_frame[column])

plt

.

xlabel(

'Date'

,

fontsize

=

18

)

plt

.

ylabel(column

+

'

Price

USD

(

$

)

for

'

+

stock,

fontsize

=

18

)

plt

.

show()

*#Test*

*the*

*function*

plot\_close\_val(KO\_Data,

'Close'

,

'Coca

Cola'

)

plot\_close\_val(KO\_Data,

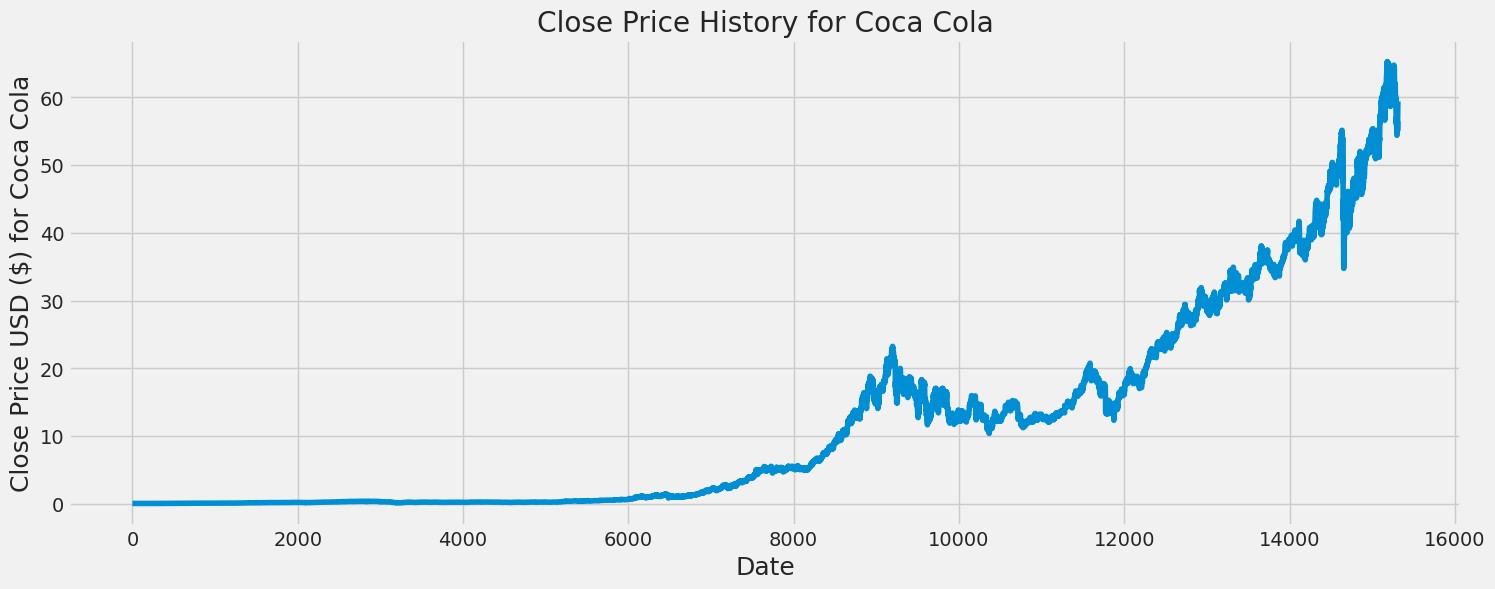
'Open'

,

'Coca

Cola'

)



In

[6]:

KO\_Data[[

"Volume"

]]

.

plot()

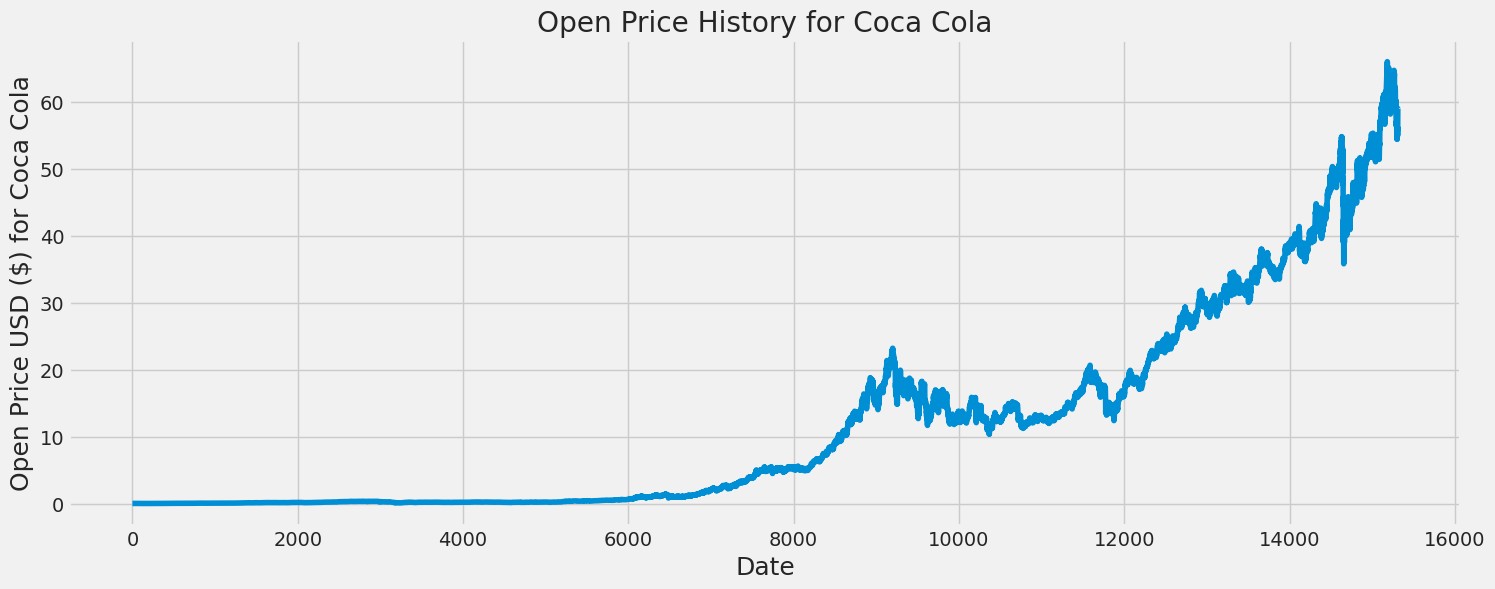
Out[6]:

<

Axes

:

>



4.

Basic

Company

Info

In

[7]:

ko\_info

=

pd

.

read\_csv(

'../input/coca-cola-stock-live-and-updated/Coca-Col

a\_stock\_info.csv'

,

header

=

None

,

names

=

([

'Description'

,

'Information'

]))

ko\_info

.

dropna()

ko\_info

.

drop(ko\_info

.

loc[ko\_info[

'Information'

]

==

'nan'

]

.

index,

inplace

=

True

)

ko

=

ko\_info

.

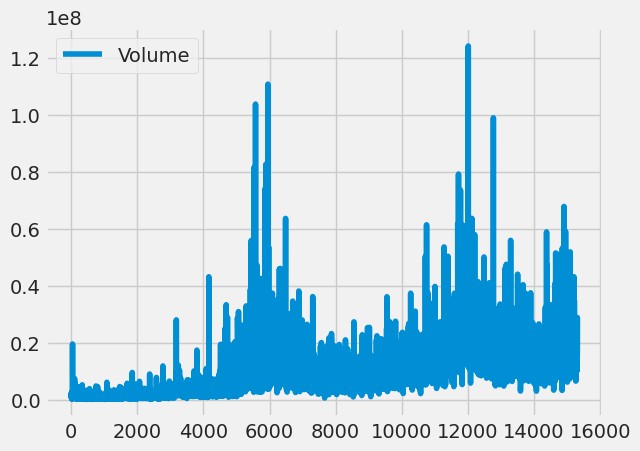
sort\_values(

'Information'

)

.

style



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ko  Out[7]:   |  |  |  | | --- | --- | --- | |  | Description | Information | | 4  9 | gmtOffSetMillisec onds | -18000000 | | 1 4 5 | bid | 0 | | 1 3 8 | ask | 0 | | 7  7 | heldPercentInside rs | 0.00636 | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | 6  8 | sharesPercentSh aresOut | 0.0074 | | 1 0 0 | shortPercentOfFl oat | 0.0074 | | 1 0 8 | trailingAnnualDivi dendYield | 0.027917083 | | 1 4 7 | dividendYield | 0.028099999 | | 2  9 | returnOnAssets | 0.07831 | | 7  5 | SandP52WeekCh ange | 0.15025425 | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | 1  8 | revenueGrowth | 0.161 | | 5  8 | 52WeekChange | 0.21709049 | | 1  5 | profitMargins | 0.23313999 | | 1  9 | operatingMargins | 0.31123 | | 1  4 | ebitdaMargins | 0.35199 | | 3  3 | returnOnEquity | 0.39722002 | | 7  4 | lastDividendValue | 0.42 | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | 9  3 | earningsQuarterly  Growth | 0.423 | | 2  7 | earningsGrowth | 0.425 | | 1  6 | grossMargins | 0.60723996 | | 7  1 | heldPercentInstitu tions | 0.7005 | | 8  4 | beta | 0.712113 | | 1 0 9 | payoutRatio | 0.82269996 | | 1 | maxAge | 1 | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | 1 |  |  | | 4  1 | quickRatio | 1.173 | | 2  8 | currentRatio | 1.516 | | 8  1 | shortRatio | 1.6 | | 1 1 6 | trailingAnnualDivi dendRate | 1.67 | | 1 2 2 | dividendRate | 1.68 | | 1  3 | askSize | 1000 | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | 9 |  |  | | 7  6 | priceToBook | 11.606621 | | 1  7 | operatingCashflo  w | 12855000064 | | 2  0 | ebitda | 13306000384 | | 8  8 | lastSplitDate | 1344816000 | | 3  5 | totalCash | 14871000064 | | 7  0 | lastFiscalYearEnd | 1609372800 | |

|  |  |  |
| --- | --- | --- |
| 8  0 | mostRecentQuart er | 1633046400 |
| 9  1 | lastDividendDate | 1638230400 |
| 8  2 | sharesShortPrevi ousMonthDate | 1638230400 |
| 1 2 3 | exDividendDate | 1638230400 |
| 9  5 | dateShortInterest | 1640908800 |
| 7  8 | nextFiscalYearEn d | 1672444800 |
| 3 | debtToEquity | 172.826 |

|  |  |  |
| --- | --- | --- |
| 2 |  |  |
| 1 3 6 | averageVolume | 17746368 |
| 1 4 0 | volume | 18219394 |
| 1 2 9 | regularMarketVol  ume | 18219394 |
| 2  3 | grossProfits | 19581000000 |
| 8  6 | priceHint | 2 |
| 7 | trailingEps | 2.031 |

|  |  |  |
| --- | --- | --- |
| 3 |  |  |
| 4  2 | recommendation  Mean | 2.1 |
| 6  0 | forwardEps | 2.43 |
| 1  5 3 | trailingPegRatio | 2.6848 |
| 9  6 | pegRatio | 2.77 |
| 1 1 9 | averageVolume1  0days | 20867790 |
| 1  1 | averageDailyVolu me10Day | 20867790 |

|  |  |  |
| --- | --- | --- |
| 3 |  |  |
| 5  7 | enterpriseToEbitd a | 21.583 |
| 9  8 | forwardPE | 24.526747 |
| 1  0 1 | sharesShortPrior  Month | 24026403 |
| 3  0 | numberOfAnalyst  Opinions | 25 |
| 1 3 3 | marketCap | 257437417472 |
| 8  5 | enterpriseValue | 287178719232 |

|  |  |  |
| --- | --- | --- |
| 1  2 8 | trailingPE | 29.34515 |
| 8  9 | lastSplitFactor | 2:1 |
| 1  4 3 | fiveYearAvgDivid endYield | 3.21 |
| 3  8 | totalCashPerShar e | 3.443 |
| 1 | zip | 30313 |
| 6  7 | sharesShort | 31874471 |
| 3 | totalRevenue | 37802000384 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | 7 |  |  | | 8  3 | floatShares | 3890760972 | | 6 | phone | 404 676 2121 | | 3  6 | totalDebt | 41707999232 | | 1 0 2 | impliedSharesOut standing | 4311130112 | | 6  2 | sharesOutstandin g | 4319419904 | | 1 4 4 | fiftyTwoWeekLow | 48.11 | |

|  |  |  |
| --- | --- | --- |
| 6  6 | bookValue | 5.135 |
| 1 0 7 | twoHundredDayA verage | 55.77645 |
| 1  1 5 | fiftyDayAverage | 57.6512 |
| 2  1 | targetLowPrice | 58 |
| 1 3 7 | dayLow | 59.21 |
| 1 2 6 | regularMarketDay  Low | 59.21 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | 1  5 0 | regularMarketPric e | 59.6 | | 2  6 | currentPrice | 59.6 | | 1  1 7 | open | 59.79 | | 1 0 6 | regularMarketOp en | 59.79 | | 1 1 4 | regularMarketPre viousClose | 59.82 | | 1  0 | previousClose | 59.82 | |

|  |  |  |
| --- | --- | --- |
| 5 |  |  |
| 9  4 | priceToSalesTraili ng12Months | 6.8101535 |
| 1 4 9 | dayHigh | 60.345 |
| 1 1 1 | regularMarketDay  High | 60.345 |
| 1 4 1 | fiftyTwoWeekHigh | 61.45 |
| 3  1 | targetMeanPrice | 63.72 |
| 2 | targetMedianPric | 64 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | 5 | e |  | | 5  5 | enterpriseToReve nue | 7.597 | | 3  4 | targetHighPrice | 70 | | 2  4 | freeCashflow | 7007374848 | | 4  0 | revenuePerShare | 8.771 | | 1 4 8 | bidSize | 800 | | 3 | fullTimeEmployee s | 80300 | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | 7  2 | netIncomeToCom mon | 8812999680 | | 4  6 | exchangeTimezo neName | America/New\_York | | 5 | city | Atlanta | | 1  3 | industry | Beverages—Non-Alcoholic | | 4  4 | shortName | Coca-Cola Company (The) | | 2 | sector | Consumer Defensive | | 5  0 | quoteType | EQUITY | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | 4  7 | exchangeTimezo neShortName | EST | | 1 4 6 | tradeable | False | | 4  8 | isEsgPopulated | False | | 7 | state | GA | | 5  1 | symbol | KO | | 4  3 | exchange | NYQ | | 1  2 | address1 | One Coca-Cola Plaza | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | 4  5 | longName | The Coca-Cola Company | | 4 | longBusinessSum mary | The Coca-Cola Company, a beverage company, manufactures, markets, and sells various nonalcoholic beverages worldwide. The company provides sparkling soft drinks; water, enhanced water, and sports drinks; juice, dairy, and plantÂbased beverages; tea and coffee; and energy drinks. It also offers beverage concentrates and syrups, as well as fountain syrups to fountain retailers, such as restaurants and convenience stores. The company sells its products under the Coca-Cola, Diet  Coke/Coca-Cola Light, Coca-Cola Zero Sugar, Fanta,  Fresca, Schweppes, Sprite, Thums Up, Aquarius, Ciel,  Dasani, glacÃ©au smartwater, glacÃ©au vitaminwater, Ice Dew, I LOHAS, Powerade, Topo Chico, AdeS, Del Valle, fairlife, innocent, Minute Maid, Minute Maid Pulpy, Simply,  Ayataka, Costa, dogadan, FUZE TEA, Georgia, Gold Peak, HONEST TEA, and Kochakaden brands. It operates through a network of independent bottling partners, distributors, wholesalers, and retailers, as well as through bottling and distribution operators. The company was founded in 1886 and is headquartered in Atlanta, Georgia. | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | 1 2 7 | currency | USD | | 3  9 | financialCurrency | USD | | 8 | country | United States | | 0 | Key | Value | | 9 | companyOfficers | [] | | 2  2 | recommendation  Key | buy | | 5  2 | messageBoardId | finmb\_26642 | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | 1 5 2 | logo\_url | https://logo.clearbit.com/coca-colacompany.com | | 1  0 | website | https://www.coca-colacompany.com | | 5  3 | market | us\_market | | 5  4 | annualHoldingsTu rnover | nan | | 5  6 | beta3Year | nan | | 5  9 | morningStarRisk  Rating | nan | | 6 | revenueQuarterly | nan | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | 1 | Growth |  | | 6  3 | fundInceptionDat e | nan | | 6  4 | annualReportExp enseRatio | nan | | 6  5 | totalAssets | nan | | 6  9 | fundFamily | nan | | 7  9 | yield | nan | | 8  7 | threeYearAverage  Return | nan | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | 9  0 | legalType | nan | | 9  2 | morningStarOver allRating | nan | | 9  7 | ytdReturn | nan | | 9  9 | lastCapGain | nan | | 1 0 3 | category | nan | | 1 0 4 | fiveYearAverageR eturn | nan | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | 1 1 0 | volume24Hr | nan | | 1 1 2 | navPrice | nan | | 1 1 8 | toCurrency | nan | | 1 2 0 | expireDate | nan | | 1 2 1 | algorithm | nan | | 1  2 | circulatingSupply | nan | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | 4 |  |  | | 1 2 5 | startDate | nan | | 1  3 0 | lastMarket | nan | | 1 3 1 | maxSupply | nan | | 1 3 2 | openInterest | nan | | 1 3 4 | volumeAllCurrenc ies | nan | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | 1  3 5 | strikePrice | nan | | 1  4 2 | fromCurrency | nan | | 1  5 1 | preMarketPrice | nan |  1. Basic CAGR   unfold\_moreShow hidden cell  In [ ]:  5.1 Basic Rolling Averages  In [9]:  *# Isolate the adjusted closing prices* |

|  |
| --- |
| adj\_close\_px = KO\_Data['Close']  *# Calculate the moving average* moving\_avg = adj\_close\_px.rolling(window=40).mean()  *# Inspect the result* moving\_avg[-10:]  Out[9]:   1. 59.573229 2. 59.329031 3. 59.103823 4. 58.921440 5. 58.725320 6. 58.504966 7. 58.298918 8. 58.171838 9. 58.088689 10. 58.030935   Name: Close, dtype: float64  In [10]:  *# Short moving window rolling mean*  KO\_Data['42'] = adj\_close\_px.rolling(window=40).mean() |

*#*

*Long*

*moving*

*window*

*rolling*

*mean*

KO\_Data[

'252'

]

=

adj\_close\_px

.

rolling(window

=

252

)

.

mean()

*#*

*Plot*

*the*

*adjusted*

*closing*

*price,*

*the*

*short*

*and*

*long*

*windows*

*of*

*rolling*

*means*

KO\_Data[[

'Close'

,

'42'

,

'252'

]]

.

plot()

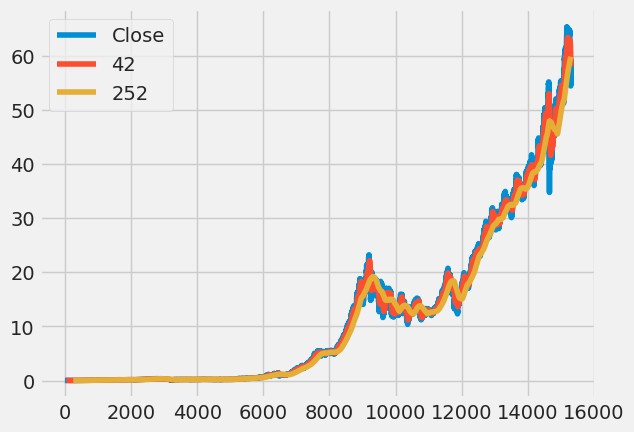
plt

.

show()

In

[11]:



daily\_close\_px

=

KO\_Data[[

'Close'

]]

*#*

*Calculate*

*the*

*daily*

*percentage*

*change*

*for*

*`daily\_close\_px`*

daily\_pct\_change

=

daily\_close\_px

.

pct\_change()

*#*

*Plot*

*the*

*distributions*

daily\_pct\_change

.

hist(bins

=

50

,

sharex

=

True

,

figsize

=

(

12

,

8

))

*#*

*Show*

*the*

*resulting*

*plot*

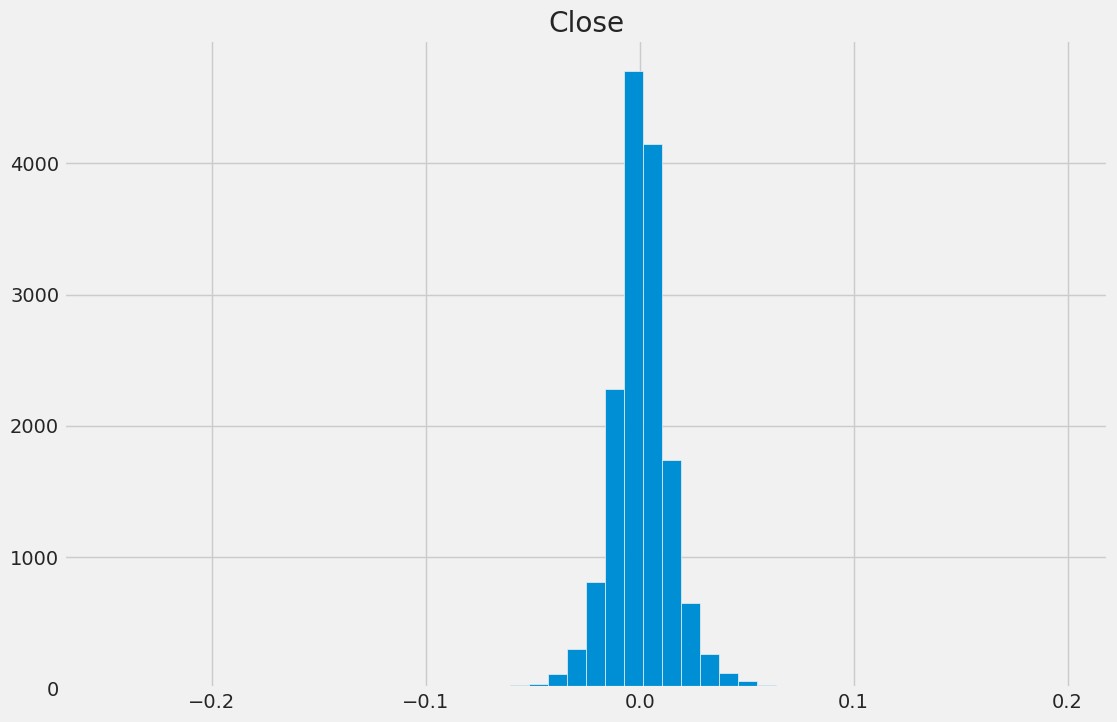
plt

.

show()

In

[12]:



|  |
| --- |
| *# Define the minumum of periods to consider* min\_periods = 75  *# Calculate the volatility* vol = daily\_pct\_change.rolling(min\_periods).std() \* np.sqrt(min\_periods)  *# Plot the volatility* vol.plot(figsize=(10, 8))  *# Show the plot* plt.show() |

In

[13]:

*#*

*Plot*

*a*

*scatter*

*matrix*

*with*

*the*

*`daily\_pct\_change`*

*data*

pd

.

plotting

.

scatter\_matrix(daily\_pct\_change,

diagonal

=

'kde'

,

alpha

=

0.1

,figsize

=

(

12

,

12

))

*#*

*Show*

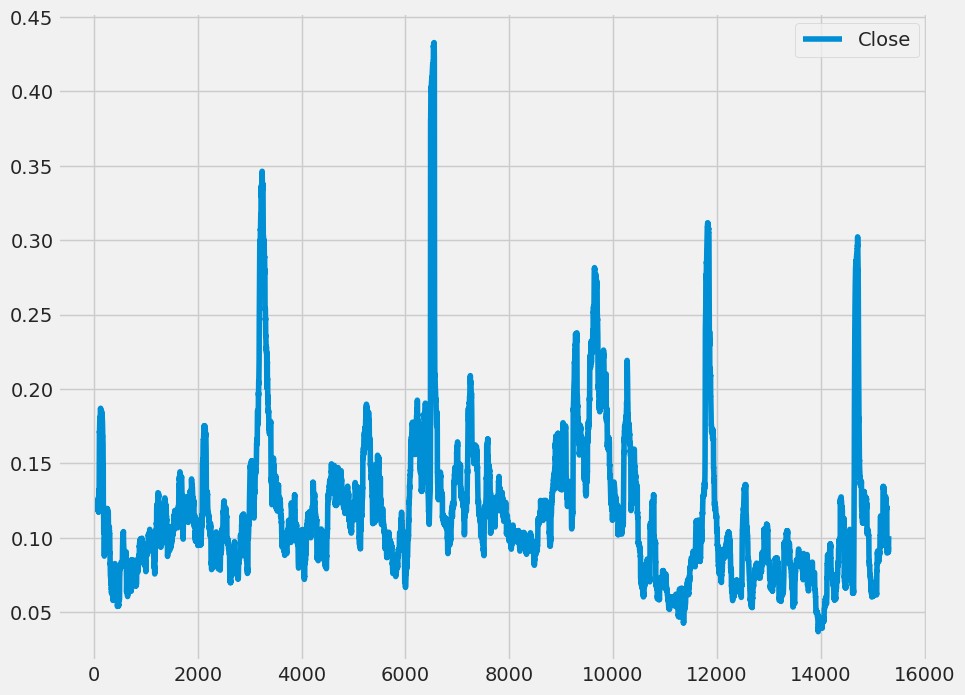
*the*

*plot*

plt

.

show()



5.2

Basic

MACD

In

[14]:

import

plotly.graph\_objects

as

go

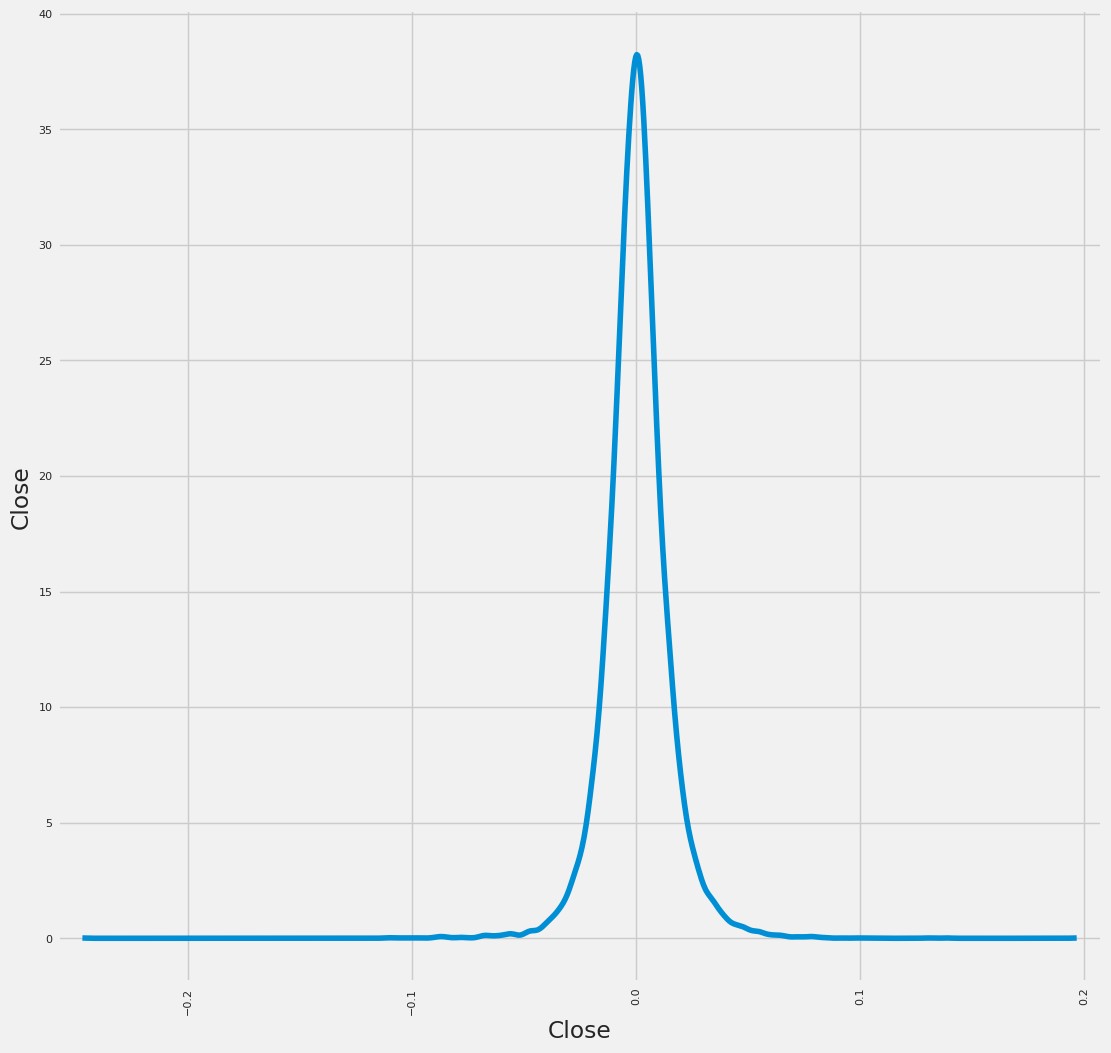
KO\_Data

=

KO\_Data

.

reset\_index()



|  |
| --- |
| fig = go.Figure(data=go.Ohlc(x=KO\_Data['Date'],  open=KO\_Data['Open'], high=KO\_Data['High'], low=KO\_Data['Low'], close=KO\_Data['Close']))  fig.show()  1970198019902000201020200102030405060  5.2.1 Basic SMA  In [15]:  *#KO\_Data=KO\_Data.reset\_index()*  KO\_Data['SMA5'] = KO\_Data.Close.rolling(5).mean()  KO\_Data['SMA20'] = KO\_Data.Close.rolling(20).mean()  KO\_Data['SMA50'] = KO\_Data.Close.rolling(50).mean()  KO\_Data['SMA200'] = KO\_Data.Close.rolling(200).mean()  KO\_Data['SMA500'] = KO\_Data.Close.rolling(500).mean()  fig =  go.Figure(data=[go.Ohlc(x=KO\_Data['Date'],open=KO\_Data['Open'], high=KO\_Data['High'],low=KO\_Data['Low'],close=KO\_Data['Close'], name = "OHLC"), |

|  |
| --- |
| go.Scatter(x=KO\_Data.Date,  y=KO\_Data.SMA5, line=dict(color='orange', width=1), name="SMA5"), go.Scatter(x=KO\_Data.Date,  y=KO\_Data.SMA20, line=dict(color='green', width=1), name="SMA20"), go.Scatter(x=KO\_Data.Date,  y=KO\_Data.SMA50, line=dict(color='blue', width=1), name="SMA50"), go.Scatter(x=KO\_Data.Date,  y=KO\_Data.SMA200, line=dict(color='violet', width=1), name="SMA200"), go.Scatter(x=KO\_Data.Date,  y=KO\_Data.SMA500, line=dict(color='purple', width=1), name="SMA500")]) fig.show()  1970198019902000201020200102030405060  OHLCSMA5SMA20SMA50SMA200SMA500  5.2.2 Basic EMA  In [16]:  KO\_Data['EMA5'] = KO\_Data.Close.ewm(span=5, adjust=False).mean() |

|  |
| --- |
| KO\_Data['EMA20'] = KO\_Data.Close.ewm(span=20, adjust=False).mean()  KO\_Data['EMA50'] = KO\_Data.Close.ewm(span=50, adjust=False).mean()  KO\_Data['EMA200'] = KO\_Data.Close.ewm(span=200, adjust=False).mean()  KO\_Data['EMA500'] = KO\_Data.Close.ewm(span=500, adjust=False).mean()  fig = go.Figure(data=[go.Ohlc(x=KO\_Data['Date'], open=KO\_Data['Open'], high=KO\_Data['High'], low=KO\_Data['Low'], close=KO\_Data['Close'], name =  "OHLC"), go.Scatter(x=KO\_Data.Date,  y=KO\_Data.SMA5, line=dict(color='orange', width=1), name="EMA5"), go.Scatter(x=KO\_Data.Date,  y=KO\_Data.SMA20, line=dict(color='green', width=1), name="EMA20"), go.Scatter(x=KO\_Data.Date,  y=KO\_Data.SMA50, line=dict(color='blue', width=1), name="EMA50"), go.Scatter(x=KO\_Data.Date, |

y

=

KO\_Data

.

SMA200,

line

=

dict

color

(

=

'violet'

,

width

=

1

,

)

name

=

"EMA200"

,

)

go

.

Scatter(x

=

KO\_Data

.

Date,

y

=

KO\_Data

.

SMA500,

line

=

dict

(

color

=

'purple'

,

width

=

1

,

)

name

=

"EMA500"

)])

fig

.

show()

1970198019902000201020200102030405060

OHLCEMA5EMA20EMA50EMA200EMA500

In

[17]:

KO\_Data

.

head()

*#KO\_Data.fillna(0)*

*#KO\_Data.set\_index('Date')*

Out[17]:

i

n

d

e

x

D

a

t

e

O

pe

n

Hi

gh

Lo

w

Cl

os

e

V

ol

u

m

e

Di

vi

de

nd

s

S

t

o

c

k

S

p

4

2

.

.

.

S

M

A

5

S

M

A

2

0

S

M

A

5

0

S

M

A

2

0

0

S

M

A

5

0

0

E

M

A

5

E

M

A

20

E

M

A

50

E

M

A

20

0

E

M

A

50

0

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  | li t  s |  |  |  |  |  |  |  |  |  |  |  |  |
| 0 | 0 | 1  9  6  2  -  0  1  -  0 2 | 0.  05 00 16 | 0.  05 13 78 | 0.  05 00 16 | 0.  05 00 16 | 8 0  6 4 0 0 | 0.  0 | 0 | N a  N | .  .  . | N a  N | N a  N | N a  N | N a  N | N a  N | 0.  05 00 16 | 0.  05 00 16 | 0.  05 00 16 | 0.  05 00 16 | 0.  05 00 16 |
| 1 | 1 | 1 9 6  2  -  0  1  -  0 3 | 0.  04 92 73 | 0.  04 92 73 | 0.  04 81 59 | 0.  04 89 02 | 1 5 7 4 4 0 0 | 0.  0 | 0 | N  a  N | .  .  . | N  a  N | N  a  N | N  a  N | N  a  N | N  a  N | 0.  04 96 44 | 0.  04 99 10 | 0.  04 99 72 | 0.  05 00 05 | 0.  05  00  11 |
|  | | | | | | | | | | | | | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | | | | | | | | | | | | | |
|  | 2 | 2 | 1  9 6  2  -  0  1  -  0 4 | 0.  04  90 26 | 0.  04  96 45 | 0.  04  90 26 | 0.  04  92 73 | 8  4  4  8 0 0 | 0.  0 | 0 | N a  N | .  .  . | N a  N | N a  N | N a  N | N a  N | N a  N | 0.  04  95 21 | 0.  04  98 49 | 0.  04  99 45 | 0.  04  99 98 | 0.  05  00 09 |
| 3 | 3 | 1 9 6  2  -  0  1  -  0 5 | 0.  04 92 73 | 0.  04 98 92 | 0.  04 80 35 | 0.  04 81 59 | 1 4 2 0 8 0 0 | 0.  0 | 0 | N  a  N | .  .  . | N  a  N | N  a  N | N  a  N | N  a  N | N  a  N | 0.  04 90 67 | 0.  04 96 88 | 0.  04 98 75 | 0.  04 99 79 | 0.  05 00 01 |
| 4 | 4 | 1 9 6 | 0.  04 77 | 0.  04 77 | 0.  04 67 | 0.  04 76 | 2 0 3 | 0.  0 | 0 | N  a | . . | 0.  04 88 | N  a | N  a | N  a | N  a | 0.  04 85 | 0.  04 94 | 0.  04 97 | 0.  04 99 | 0.  04 99 |
|  | | | | | | | | | | | | | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | | | | | | | | | | | | | |
|  |  |  | 2  -  0  1  -  0 8 | 87 | 87 | 35 | 64 | 5  2 0 0 |  |  | N | . | 03 | N | N | N | N | 99 | 95 | 88 | 56 | 92 |
| 1. rows × 21 columns 2. FINTA Tech Analysis Ratios   Let us do a financial ratios calculation using FINTA library   * + Simple Moving Average 'SMA'   + Simple Moving Median 'SMM'   + Smoothed Simple Moving Average 'SSMA'   + Exponential Moving Average 'EMA'   + Double Exponential Moving Average 'DEMA'   + Triple Exponential Moving Average 'TEMA'   + Triangular Moving Average 'TRIMA'   + Triple Exponential Moving Average Oscillator 'TRIX'   + Volume Adjusted Moving Average 'VAMA'   + Kaufman Efficiency Indicator 'ER'   + Kaufman's Adaptive Moving Average 'KAMA'   + Zero Lag Exponential Moving Average 'ZLEMA'   + Weighted Moving Average 'WMA'   + Hull Moving Average 'HMA'   + Elastic Volume Moving Average 'EVWMA' | | | | | | | | | | | | | | | |  |  |  |  |  |  |

|  |
| --- |
| * Volume Weighted Average Price 'VWAP' * Smoothed Moving Average 'SMMA' * Fractal Adaptive Moving Average 'FRAMA' * Moving Average Convergence Divergence 'MACD' * Percentage Price Oscillator 'PPO' * Volume-Weighted MACD 'VW\_MACD' * Elastic-Volume weighted MACD 'EV\_MACD' * Market Momentum 'MOM' * Rate-of-Change 'ROC' * Relative Strenght Index 'RSI' * Inverse Fisher Transform RSI 'IFT\_RSI' * True Range 'TR' * Average True Range 'ATR' * Stop-and-Reverse 'SAR' * Bollinger Bands 'BBANDS' * Bollinger Bands Width 'BBWIDTH' * Momentum Breakout Bands 'MOBO' * Percent B 'PERCENT\_B' * Keltner Channels 'KC' * Donchian Channel 'DO' * Directional Movement Indicator 'DMI' * Average Directional Index 'ADX' * Pivot Points 'PIVOT' * Fibonacci Pivot Points 'PIVOT\_FIB' ● Stochastic Oscillator %K 'STOCH' * Stochastic oscillator %D 'STOCHD' * Stochastic RSI 'STOCHRSI' * Williams %R 'WILLIAMS' * Ultimate Oscillator 'UO' |

|  |
| --- |
| * Awesome Oscillator 'AO' * Mass Index 'MI' * Vortex Indicator 'VORTEX' * Know Sure Thing 'KST' * True Strength Index 'TSI' * Typical Price 'TP' * Accumulation-Distribution Line 'ADL' * Chaikin Oscillator 'CHAIKIN' * Money Flow Index 'MFI' * On Balance Volume 'OBV' * Weighter OBV 'WOBV' * Volume Zone Oscillator 'VZO' * Price Zone Oscillator 'PZO' * Elder's Force Index 'EFI' * Cummulative Force Index 'CFI' * Bull power and Bear Power 'EBBP' * Ease of Movement 'EMV' * Commodity Channel Index 'CCI' * Coppock Curve 'COPP' * Buy and Sell Pressure 'BASP' * Normalized BASP 'BASPN' * Chande Momentum Oscillator 'CMO' * Chandelier Exit 'CHANDELIER' * Qstick 'QSTICK' * Twiggs Money Index 'TMF' * Wave Trend Oscillator 'WTO' * Fisher Transform 'FISH' * Ichimoku Cloud 'ICHIMOKU' * Adaptive Price Zone 'APZ' |

|  |
| --- |
| * Squeeze Momentum Indicator 'SQZMI' * Volume Price Trend 'VPT' * Finite Volume Element 'FVE' * Volume Flow Indicator 'VFI' * Moving Standard deviation 'MSD' * Schaff Trend Cycle 'STC'   In [18]:  try:  from finta import TA from backtesting import Backtest, Strategy from backtesting.lib import crossover  except:  !pip install finta backtesting from finta import TA from backtesting import Backtest, Strategy from backtesting.lib import crossover  Collecting finta  Downloading finta-1.3-py3-none-any.whl (29 kB)  Collecting backtesting  Downloading Backtesting-0.3.3.tar.gz (175 kB)  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━  ━━ 175.5/175.5 kB 4.3 MB/s eta 0:00:00  Preparing metadata (setup.py) ... - done |

Requirement already satisfied: numpy in

/opt/conda/lib/python3.10/site-packages (from finta) (1.23.5)

Requirement already satisfied: pandas in

/opt/conda/lib/python3.10/site-packages (from finta) (2.0.3)

Requirement already satisfied: bokeh>=1.4.0 in

/opt/conda/lib/python3.10/site-packages (from backtesting)

(3.2.2)

Requirement already satisfied: Jinja2>=2.9 in /opt/conda/lib/python3.10/site-packages (from bokeh>=1.4.0->backtesting) (3.1.2)

Requirement already satisfied: contourpy>=1 in /opt/conda/lib/python3.10/site-packages (from bokeh>=1.4.0->backtesting) (1.1.0)

Requirement already satisfied: packaging>=16.8 in /opt/conda/lib/python3.10/site-packages (from bokeh>=1.4.0->backtesting) (21.3)

Requirement already satisfied: pillow>=7.1.0 in /opt/conda/lib/python3.10/site-packages (from bokeh>=1.4.0->backtesting) (9.5.0)

Requirement already satisfied: PyYAML>=3.10 in /opt/conda/lib/python3.10/site-packages (from bokeh>=1.4.0->backtesting) (6.0)

Requirement already satisfied: tornado>=5.1 in /opt/conda/lib/python3.10/site-packages (from bokeh>=1.4.0->backtesting) (6.3.2)

|  |
| --- |
| Requirement already satisfied: xyzservices>=2021.09.1 in  /opt/conda/lib/python3.10/site-packages (from bokeh>=1.4.0->backtesting) (2023.7.0)  Requirement already satisfied: python-dateutil>=2.8.2 in  /opt/conda/lib/python3.10/site-packages (from pandas->finta)  (2.8.2)  Requirement already satisfied: pytz>=2020.1 in  /opt/conda/lib/python3.10/site-packages (from pandas->finta)  (2023.3)  Requirement already satisfied: tzdata>=2022.1 in  /opt/conda/lib/python3.10/site-packages (from pandas->finta)  (2023.3)  Requirement already satisfied: MarkupSafe>=2.0 in  /opt/conda/lib/python3.10/site-packages (from  Jinja2>=2.9->bokeh>=1.4.0->backtesting) (2.1.3)  Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in  /opt/conda/lib/python3.10/site-packages (from packaging>=16.8->bokeh>=1.4.0->backtesting) (3.0.9)  Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.10/site-packages (from python-dateutil>=2.8.2->pandas->finta) (1.16.0) Building wheels for collected packages: backtesting Building wheel for backtesting (setup.py) ... - \ done Created wheel for backtesting:  filename=Backtesting-0.3.3-py3-none-any.whl size=173804 |

sha256=cacca55c1313b2d6d721b1315d0471739920756e04b6a2004aed8c0f

21

dbc56e

Stored

in

directory:

/root/.cache/pip/wheels/e2/30/7f/19cbe31987c6ebdb47f1f510343249

066711609e3da2d57176

Successfully

built

backtesting

Installing

collected

packages:

finta,

backtesting

Successfully

installed

backtesting-0.3.3

finta-1.3

BokehJS

3.2.2

successfully

loaded.

In

[19]:

fin\_ma

=

pd

.

read\_csv(

'../input/coca-cola-stock-live-and-updated/Coca-Col

a\_stock\_history.csv'

,

parse\_dates

=

True

)

print

(

fin\_ma

.

head())

ohlc

=

fin\_ma

print

(

TA

.

SMA(ohlc,

42

))

*#ohlc.index*

*=*

*ohlc[index].dt.date*

Date

Open

High

Low

Close

Volume

Dividends

\

0

1962-01-02

0.050016

0.051378

0.050016

0.050016

806400

0.0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 1962-01-03 0.049273 | | 0.049273 | 0.048159 | 0.048902 | 1574400 |
| 0.0  2 1962-01-04 0.049026 | | 0.049645 | 0.049026 | 0.049273 | 844800 |
| 0.0  3 1962-01-05 0.049273 | | 0.049892 | 0.048035 | 0.048159 | 1420800 |
| 0.0  4 1962-01-08 0.047787  0.0  Stock Splits  0 0 | | 0.047787 | 0.046735 | 0.047664 | 2035200 |
| 1 | 0 |  |  |  |  |
| 2 | 0 |  |  |  |  |
| 3 | 0 |  |  |  |  |
| 4 | 0 |  |  |  |  |
| 0 | NaN |  |  |  |  |
| 1 | NaN |  |  |  |  |
| 2 | NaN |  |  |  |  |
| 3 | NaN |  |  |  |  |
| 4 | NaN ... |  |  |  |  |
| 15306 | 58.759467 |  |  |  |  |
| 15307 | 58.572686 |  |  |  |  |
| 15308 | 58.422110 |  |  |  |  |
| 15309 | 58.297065 |  |  |  |  |

15310 58.219369

Name: 42 period SMA, Length: 15311, dtype: float64

In [20]:

function\_dict = {' Simple Moving Average ' : 'SMA',

' Simple Moving Median ' : 'SMM',

' Smoothed Simple Moving Average ' : 'SSMA',

' Exponential Moving Average ' : 'EMA', ' Double Exponential Moving Average ' :

'DEMA',

' Triple Exponential Moving Average ' :

'TEMA',

' Triangular Moving Average ' : 'TRIMA',

' Triple Exponential Moving Average Oscillator

' : 'TRIX',

' Volume Adjusted Moving Average ' : 'VAMA',

' Kaufman Efficiency Indicator ' : 'ER',

' Kaufmans Adaptive Moving Average ' : 'KAMA', ' Zero Lag Exponential Moving Average ' :

'ZLEMA',

Weighted Moving Average ' : 'WMA',

Hull Moving Average ' : 'HMA',

Elastic Volume Moving Average ' : 'EVWMA',

Volume Weighted Average Price ' : 'VWAP',

' Smoothed Moving Average ' : 'SMMA', ' Fractal Adaptive Moving Average ' : 'FRAMA', ' Moving Average Convergence Divergence ' :

'MACD',

' Percentage Price Oscillator ' : 'PPO',

' Volume-Weighted MACD ' : 'VW\_MACD',

' Elastic-Volume weighted MACD ' : 'EV\_MACD',

' Market Momentum ' : 'MOM',

' Rate-of-Change ' : 'ROC',

' Relative Strength Index ' : 'RSI',

' Inverse Fisher Transform RSI ' : 'IFT\_RSI',

' True Range ' : 'TR',

' Average True Range ' : 'ATR',

' Stop-and-Reverse ' : 'SAR',

' Bollinger Bands ' : 'BBANDS',

' Bollinger Bands Width ' : 'BBWIDTH',

' Momentum Breakout Bands ' : 'MOBO',

' Percent B ' : 'PERCENT\_B',

' Keltner Channels ' : 'KC',

' Donchian Channel ' : 'DO',

' Directional Movement Indicator ' : 'DMI',

Average Directional Index ' : 'ADX',

Pivot Points ' : 'PIVOT',

Fibonacci Pivot Points ' : 'PIVOT\_FIB',

Stochastic Oscillator Percent K ' : 'STOCH',

' Stochastic oscillator Percent D ' : 'STOCHD',

' Stochastic RSI ' : 'STOCHRSI',

' Williams Percent R ' : 'WILLIAMS',

' Ultimate Oscillator ' : 'UO',

' Awesome Oscillator ' : 'AO',

' Mass Index ' : 'MI',

*#' Vortex Indicator ' : 'VORTEX',*

' Know Sure Thing ' : 'KST',

' True Strength Index ' : 'TSI',

' Typical Price ' : 'TP',

' Accumulation-Distribution Line ' : 'ADL',

' Chaikin Oscillator ' : 'CHAIKIN',

' Money Flow Index ' : 'MFI',

' On Balance Volume ' : 'OBV',

' Weighter OBV ' : 'WOBV',

' Volume Zone Oscillator ' : 'VZO',

' Price Zone Oscillator ' : 'PZO',

' Elders Force Index ' : 'EFI',

' Cummulative Force Index ' : 'CFI',

' Bull power and Bear Power ' : 'EBBP',

Ease of Movement ' : 'EMV',

Commodity Channel Index ' : 'CCI',

Coppock Curve ' : 'COPP',

Buy and Sell Pressure ' : 'BASP',

|  |
| --- |
| ' Normalized BASP ' : 'BASPN',  ' Chande Momentum Oscillator ' : 'CMO',  ' Chandelier Exit ' : 'CHANDELIER',  ' Qstick ' : 'QSTICK',  *#' Twiggs Money Index ' : 'TMF',*  ' Wave Trend Oscillator ' : 'WTO',  ' Fisher Transform ' : 'FISH',  ' Ichimoku Cloud ' : 'ICHIMOKU',  ' Adaptive Price Zone ' : 'APZ',  *#' Squeeze Momentum Indicator ' : 'SQZMI',*  ' Volume Price Trend ' : 'VPT',  ' Finite Volume Element ' : 'FVE',  ' Volume Flow Indicator ' : 'VFI',  ' Moving Standard deviation ' : 'MSD',  ' Schaff Trend Cycle ' : 'STC'}  for key, value **in** function\_dict .items():  function\_name = "TA." + value + "(ohlc).plot(title='" + key + "for Coca Cola / Coke Stock')"  *#print(function\_name)* result = eval(function\_name)  Unexpected exception formatting exception. Falling back to standard exception |

|  |
| --- |
| Traceback (most recent call last):  File  "/opt/conda/lib/python3.10/site-packages/IPython/core/interacti veshell.py", line 3508, in run\_code exec(code\_obj, self.user\_global\_ns, self.user\_ns)  File "/tmp/ipykernel\_20/59537527.py", line 84, in <module> result = eval(function\_name)  File "<string>", line 1, in <module> File  "/opt/conda/lib/python3.10/site-packages/finta/finta.py", line  34, in wrap return func(\*args, \*\*kwargs) File  "/opt/conda/lib/python3.10/site-packages/finta/finta.py", line  292, in KAMA sc.iteritems(), sma.shift().iteritems(),  ohlc[column].iteritems() File  "/opt/conda/lib/python3.10/site-packages/pandas/core/generic.py  ", line 5989, in \_\_getattr\_\_ return object.\_\_getattribute\_\_(self, name)  AttributeError: 'Series' object has no attribute 'iteritems'  During handling of the above exception, another exception |

|  |
| --- |
| occurred:  Traceback (most recent call last):  File  "/opt/conda/lib/python3.10/site-packages/IPython/core/interacti veshell.py", line 2105, in showtraceback stb = self.InteractiveTB.structured\_traceback( File  "/opt/conda/lib/python3.10/site-packages/IPython/core/ultratb.p y", line 1428, in structured\_traceback return FormattedTB.structured\_traceback( File  "/opt/conda/lib/python3.10/site-packages/IPython/core/ultratb.p y", line 1319, in structured\_traceback return VerboseTB.structured\_traceback( File  "/opt/conda/lib/python3.10/site-packages/IPython/core/ultratb.p y", line 1172, in structured\_traceback formatted\_exception =  self.format\_exception\_as\_a\_whole(etype, evalue, etb, number\_of\_lines\_of\_context, File  "/opt/conda/lib/python3.10/site-packages/IPython/core/ultratb.p y", line 1062, in format\_exception\_as\_a\_whole self.get\_records(etb, number\_of\_lines\_of\_context, |

|  |
| --- |
| tb\_offset) if etb else [] File  "/opt/conda/lib/python3.10/site-packages/IPython/core/ultratb.p y", line 1154, in get\_records FrameInfo(  File  "/opt/conda/lib/python3.10/site-packages/IPython/core/ultratb.p y", line 780, in \_\_init\_\_ ix = inspect.getsourcelines(frame)  File "/opt/conda/lib/python3.10/inspect.py", line 1121, in getsourcelines lines, lnum = findsource(object)  File "/opt/conda/lib/python3.10/inspect.py", line 958, in findsource raise OSError('could not get source code')  OSError: could not get source code |

7

Back

Testing

Trading

Strategy

In

[

]:

*#*

*Defining*

*DEMA*

*cross*

*strategy*

class

DemaCross

(

Strategy

):

def

init(

self

):

self

.

ma1

=

self

.

I(TA

.

DEMA,

ohlc,

10

)

self

.

ma2

=

self

.

I(TA

.

DEMA,

ohlc,

20

)

def

next(

self

):

if

crossover(

self

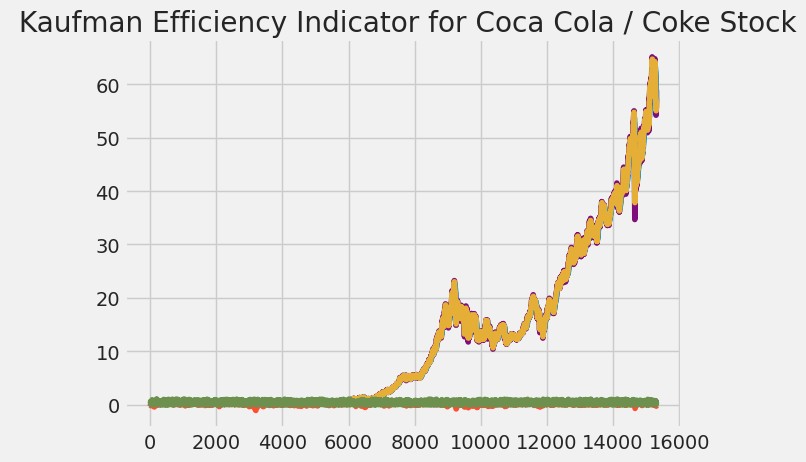
.

ma1,

self

.

ma2):



self

.

buy()

elif

crossover(

self

.

ma2,

self

.

ma1):

self

.

sell()

Let

us

do

a

bit

of

backtesting

with

a

value

of

$100000

In

[

]:

ohlc

.

head()

print

(

ohlc

.

Date)

In

[

]:

In

[

]:

bt

=

Backtest(ohlc,

DemaCross,

cash

=

100000

,

commission

=

0.015

,

exclusive\_orders

=

True

)

Back

Testing

Summary

In

[

]:

bt

.

run()

As

you

can

see,

if

you

had

invested

$100,000

in

Coca

Cola

shares,

you

would

have

got

by

now

a

return

of

118642

%

!

In

[

]:

bt

.

plot()

In

[

]:

data

=

ohlc

7.1

BackTesting

Trading

Strategy

Heatmaps

In

[

]:

from

backtesting

import

Strategy

from

backtesting.lib

import

crossover

from

backtesting.test

import

SMA

In

[

]:

def

BBANDS(data,

n\_lookback,

n\_std):

*"""Bollinger*

*bands*

*indicator"""*

hlc3

=

(

data

.

High

+

data

.

Low

+

data

.

Close)

/

3

mean,

std

=

hlc3

.

rolling(n\_lookback)

.

mean(),

hlc3.rolling(n\_lookback).std() upper = mean + n\_std\*std lower = mean - n\_std\*std return upper, lower

close = data.Close.values sma10 = SMA(data.Close, 10) sma20 = SMA(data.Close, 20) sma50 = SMA(data.Close, 50) sma100 = SMA(data.Close, 100) upper, lower = BBANDS(data, 20, 2)

*# Design matrix / independent features:*

*# Price-derived features* data['X\_SMA10'] = (close - sma10) / close data['X\_SMA20'] = (close - sma20) / close data['X\_SMA50'] = (close - sma50) / close data['X\_SMA100'] = (close - sma100) / close

data['X\_DELTA\_SMA10'] = (sma10 - sma20) / close data['X\_DELTA\_SMA20'] = (sma20 - sma50) / close data['X\_DELTA\_SMA50'] = (sma50 - sma100) / close

*#*

*Indicator*

*features*

data[

'X\_MOM'

]

=

data

.

Close

.

pct\_change(periods

=

2

)

data[

'X\_BB\_upper'

]

=

(

upper

-

close)

/

close

data[

'X\_BB\_lower'

]

=

(

lower

-

close)

/

close

data[

'X\_BB\_width'

]

=

(

upper

-

lower)

/

close

*#data['X\_Sentiment']*

*=*

*~data.index.to\_series().between('2017-09-27',*

*'2017-12-14')*

*#*

*Some*

*datetime*

*features*

*for*

*good*

*measure*

*#data['X\_day']*

*=*

*data.index.dayofweek*

*#data['X\_hour']*

*=*

*data.index.hour*

*#data*

*=*

*data.apply(pd.to\_numeric)*

*#data*

*=*

*data.dropna().astype(np.float64)*

*#data.fillna(method="ffill")*

*#data*

*data[~data.isin([np.nan,*

*=*

*np.inf,*

*np.inf]).any*

*(1)]*

*-*

*#data.replace([np.inf,*

*-*

*np.inf],*

*0.0*

*,*

*inplace=True)*

*#data*

*=*

*data.fillna(data.mean(),*

*inplace=True)*

*#data*

*=*

*data.dropna().astype(np.float64)*

In

[

]:

class

Sma4Cross

(

Strategy

):

n1

=

50

n2 = 100 n\_enter = 20 n\_exit = 10

def init(self):

self.sma1 = self.I(SMA, self.data.Close, self.n1) self.sma2 = self.I(SMA, self.data.Close, self.n2) self.sma\_enter = self.I(SMA, self.data.Close, self.n\_enter) self.sma\_exit = self.I(SMA, self.data.Close, self.n\_exit) def next(self):

if **not** self.position:

*# On upwards trend, if price closes above*

*# "entry" MA, go long*

*# Here, even though the operands are arrays, this # works by implicitly comparing the two last values* if self.sma1 > self.sma2:

if crossover(self.data.Close, self.sma\_enter):

self.buy()

*#*

*On*

*downwards*

*trend,*

*if*

*price*

*closes*

*below*

*#*

*"entry"*

*MA,*

*go*

*short*

else

:

if

crossover(

self

.

sma\_enter,

self

.

data

.

Close):

self

.

sell()

*#*

*But*

*if*

*we*

*already*

*hold*

*a*

*position*

*and*

*the*

*price*

*#*

*closes*

*back*

*below*

*(*

*above*

*)*

*"exit"*

*MA,*

*close*

*the*

*position*

else

:

if

(

self

.

position

.

is\_long

**and**

crossover(

self

.

sma\_exit,

self

.

data

.

Close)

**or**

self

.

position

.

is\_short

**and**

crossover(

self

.

data

.

Close,

self

.

sma\_exit)):

self

.

position

.

close()

In

[

]:

%%time

from

backtesting

import

Backtest

|  |
| --- |
| from backtesting.test import GOOG  backtest = Backtest(ohlc, Sma4Cross, commission=.002)  stats, heatmap = backtest.optimize( n1=range(10, 110, 10), n2=range(20, 210, 20), n\_enter=range(15, 35, 5), n\_exit=range(10, 25, 5), constraint=lambda p: p.n\_exit < p.n\_enter < p.n1 < p.n2, maximize='Equity Final [$]', max\_tries=200, random\_state=0, return\_heatmap=True)  In [ ]: heatmap  In [ ]:  hm = heatmap.groupby(['n1', 'n2']).mean().unstack() hm |

In

[

]:

from

backtesting.lib

import

plot\_heatmaps

plot\_heatmaps(heatmap,

agg

=

'mean'

)

In

[

]:

%%capture

!

pip

install

scikit

-

optimize

*#*

*This*

*is*

*a*

*run-time*

*dependency*

In

[

]:

%%time

stats\_skopt,

heatmap,

optimize\_result

=

backtest

.

optimize(

n1

=

[

10

,

100

,

]

*#*

*Note:*

*For*

*method="skopt",*

*we*

n2

=

[

20

,

200

,

]

*#*

*only*

*need*

*interval*

*end-points*

n\_enter

=

[

10

,

40

]

,

n\_exit

=

[

10

,

30

,

]

constraint

=

lambda

p:

p

.

n\_exit

<

p

.

n\_enter

<

p

.

n1

<

p

.

n2,

maximize

=

'Equity

Final

[

$]'

,

|  |  |
| --- | --- |
| method='skopt', max\_tries=200, random\_state=0, return\_heatmap=True, return\_optimization=True)  In [ ]:  from skopt.plots import plot\_objective   |  | | --- | | \_ = plot\_objective(optimize\_result, n\_points=10) | |

[Reference](https://github.com/vivekvardhan2810/Coca-Cola-Stock-Analysis) [link](https://github.com/vivekvardhan2810/Coca-Cola-Stock-Analysis)