Data Science Code Series 1: Bitcoin Analysis Project

Description

In the realm of finance, numbers and trends often hold the key to understanding and making informed decisions. In this notebook, we embark on a financial journey to analyze Bitcoin's performance and understand its investment potential. Let's dissect it step by step, unraveling the financial concepts along the way.

Coding Playground

This script is a foundational setup for financial analysis in a Jupyter Notebook environment. It incorporates four essential libraries: Numpy (`np`) for numerical operations, Numpy Financial (`npf`) for advanced financial calculations, Matplotlib (`plt`) for data visualization, and Yahoo Finance (`yf`) for accessing real-time financial data.

Together, these libraries equip users, whether investors or data analysts, with the necessary tools for in-depth financial analysis and exploration. This concise and powerful script streamlines the process of deriving meaningful insights from financial data within a Jupyter Notebook environment.

In [1]:
```python
import numpy as np
import numpy_financial as npf
import matplotlib.pyplot as plt
import yfinance as yf
```

Setting the Stage: The Analysis Period

Our stage is set with an analysis period spanning four years, from 2018 to 2021. The years '2018', '2019', '2020', and '2021' are not mere labels; they represent the timeline during which our financial protagonist, Bitcoin, underwent significant price fluctuations.

We also introduce Bitcoin's price data. The array 'bitcoin' houses the closing prices of Bitcoin for each year within our analysis period. Here, we observe the evolution of Bitcoin's price, from the relatively humble USD 3,869.47 in 2018 to a staggering USD 29,391.78 in 2021.

In [2]:
```python
years = ['2018', '2019', '2020', '2021'] # Analysis period
bitcoin = [3869.47, 7188.46, 22203.31, 29391.78] # Price for 2018-2021
```

Commit 1: Bitcoin Standard Deviation

The journey begins with a calculation of Bitcoin's ***standard deviation***, represented by `bitcoin_std`. Standard deviation measures the dispersion or volatility of an asset's returns. It's a crucial metric in finance, helping us ***assess the risk associated with an investment***. In this case, we're quantifying the price volatility of Bitcoin from 2018 to 2021.

In [3]:
```python
# First commit: Bitcoin Standard Deviation
bitcoin_std = np.std(bitcoin)

print("The Bitcoin's standard deviation is:", bitcoin_std)
```

```
The Bitcoin's standard deviation is: 10513.803224771947
```

Commit 2: Bitcoin Internal Rate of Return (IRR)

Next, we delve into the realm of returns with the **Internal Rate of Return (IRR)**. `bitcoin_irr` represents the IRR of Bitcoin, indicating the annualized return rate an investor can expect from Bitcoin mining operations. It's a critical metric for evaluating investment opportunities, ***giving us insights into the profitability of Bitcoin mining***.

```
In [4]: # Second commit: Bitcoin Internal Return of Investment (IRR)
miner_yearly = 10
bitcoin_ymined = np.multiply(bitcoin, miner_yearly)
bitcoin_equip_invest = np.array([-500000])
bitcoin_invest = np.concatenate([bitcoin_equip_invest, bitcoin_ymined])
bitcoin_irr = npf.irr(bitcoin_invest)

print("The Bitcoin's internal rate of return is:", bitcoin_irr)
```

```
The Bitcoin's internal rate of return is: 0.07297215919675293
```

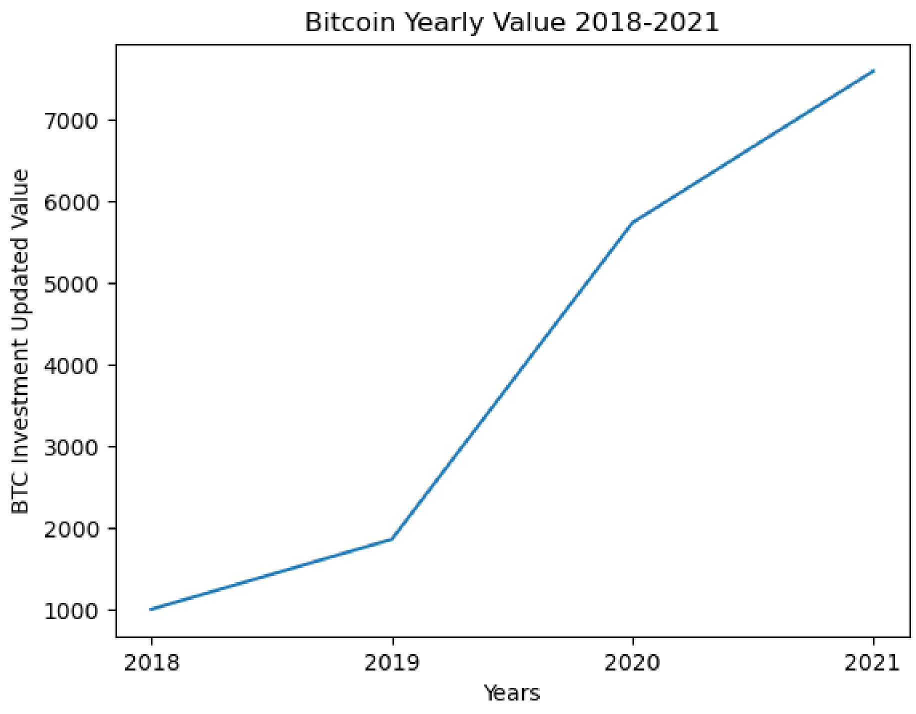
Commit 3: Bitcoin Yearly Value Plot

We visualize Bitcoin's yearly value in this step. The script calculates the initial investment amount in 2018 based on an investment of $1,000. It then tracks how that investment would have grown each year, considering Bitcoin's price fluctuations. This visual representation helps us understand Bitcoin's price trends over time.

```
In [5]: # Third commit: Bitcoin plotted yearly value
bitcoin2018_invest = 1000
bitcoin_initial_amount = bitcoin2018_invest/bitcoin[0]
bitcoin_yearly_val = np.multiply(bitcoin,bitcoin_initial_amount)

# Create the plot, add title and labels
plt.plot(years, bitcoin_yearly_val)
plt.title('Bitcoin Yearly Value 2018-2021')
plt.xlabel('Years')
plt.ylabel('BTC Investment Updated Value')

plt.show('bitcoin_yearly_val.png')
```



Commit 4: Bitcoin Price Using yfinance

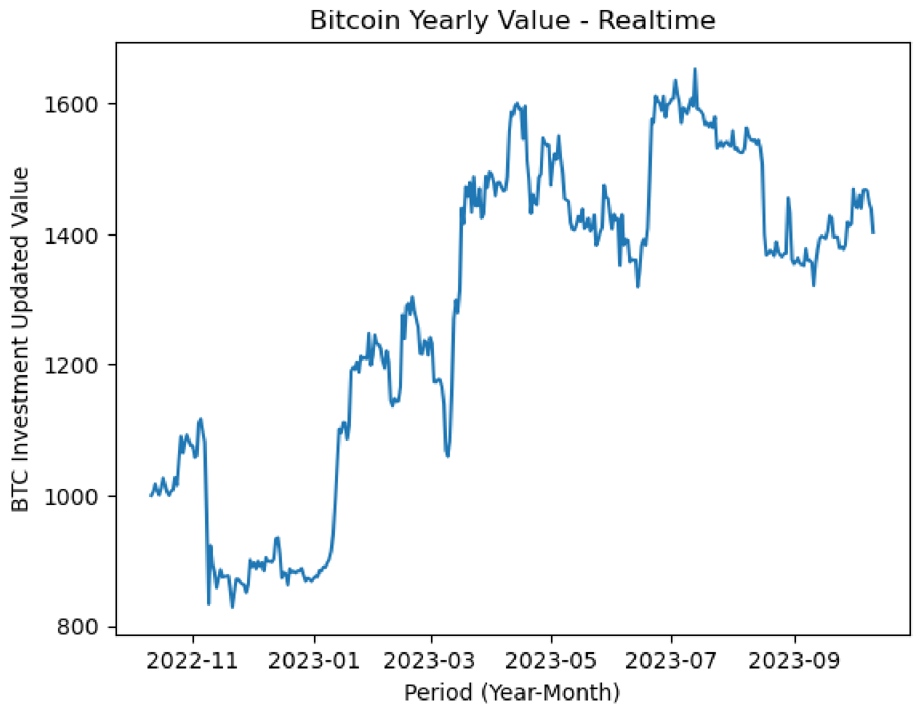
Now, we bring real-time data into the equation. We use the `yfinance` package to fetch Bitcoin's price history from **Yahoo Finance** for the past year (from last year's present month to the current month, hence the result given is updated depending on which date and time this code is run). This real-time data allows us to analyze Bitcoin's recent performance and its potential as an investment.

```
In [6]: # Fourth Commit: Bitcoin Price using yfinance
data = yf.Ticker('BTC-USD') # Data scrap from Yahoo Finance

bitcoin_price = data.history('1y')['Close'] # One-year price history (last year's present month - current month)
```

```
btc_initial_invest = 1000
btc_initial_amount = btc_initial_invest/bitcoin_price[0]
btc_yearly_val = np.multiply(bitcoin_price, btc_initial_amount)

# Create the plot, add title and labels
plt.plot(btc_yearly_val)
plt.title('Bitcoin Yearly Value - Realtime')
plt.xlabel('Period (Year-Month)')
plt.ylabel('BTC Investment Updated Value')
plt.show('btc_yearly_val.png')
```



Commit 5: Bitcoin Metrics Analysis

In the final step, we dive deeper into Bitcoin's metrics. We calculate two crucial indicators: risk and Sharpe ratio.

Risk (Annual Standard Deviation): `btc_risk` measures the ***annualized standard deviation of Bitcoin returns***. It ***quantifies the asset's volatility over a year***, a vital factor for assessing its risk profile.

Sharpe Ratio: `btc_sharpe` is the Sharpe ratio, a key metric for ***evaluating the risk-adjusted return of an investment***. It considers both returns and risk and helps investors make informed decisions.

In [7]:
```
# Fifth Commit: Bitcoin Metrics Analysis
btc_ret = bitcoin_price # Bitcoin last year price data variable assignment
trade_days = 252 # Total trading days in one year
```

In [8]:
```
# Calculate Bitcoin risk
btc_risk = np.std(btc_ret) * np.sqrt(trade_days) # Also called annual standard deviation
print('Risk:',btc_risk)
```

```
Risk: 73330.38192028097
```

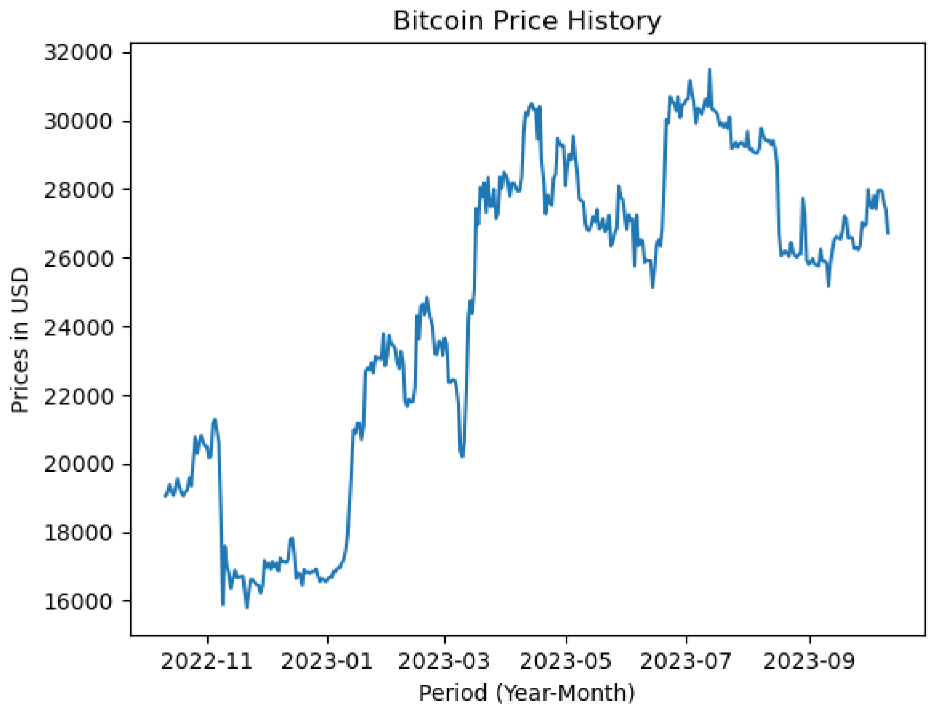
In [9]:
```
#Calculate Bitcoin sharpe
btc_sharpe = (np.mean(btc_ret) / np.std(btc_ret))*np.sqrt(trade_days)
print('Sharpe:',btc_sharpe)
```

```
Sharpe: 84.17336665107877
```

Data Visualization

Our financial journey concludes with a plot of Bitcoin's price history for the last year, offering a visual summation of our exploration.

In [10]:
```
# Create the plot, add title and labels
plt.plot(btc_ret)
plt.title('Bitcoin Price History')
plt.xlabel('Period (Year-Month)')
plt.ylabel('Prices in USD')
plt.show('btc_lastyear_history.png')
```



Summary and Closing Notes

Throughout this journey, we leveraged libraries like Numpy for numerical operations, Numpy Financial for advanced financial calculations, Matplotlib for data visualization, and Yahoo Finance for accessing real-time data. Armed with these tools and insights, we now have a comprehensive understanding of Bitcoin's financial story, empowering us to make informed investment decisions in the dynamic world of cryptocurrencies.

In this script, we've navigated the complexities of financial analysis, from historical data to real-time insights. Armed with statistical tools and metrics, we've unraveled the story of Bitcoin's financial performance, equipping ourselves with knowledge to make informed investment choices.

Credits:

By curlypetrol, 2023