RESPONSI UAS PDS

SIMULASI HARGA BITCOIN

MENGGUNAKAN MONTE CARLO

Disusun oleh:

Evan Hauzal Fadhila

L0224041

PROGRAM STUDI SAINS DATA

FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA

UNIVERSITAS SEBELAS MARET

2025

- **Deskripsi DATA**

  Disini saya mengambil data Real yang saya ambil dari kaggle yang pada keterangan tersebut mengambil data dari Yahoo Finance, yang berisi data historis harga Bitcoin dari tahun 2014 sampai tahun 2025 dengan interval waktu perhari.

  Data ini memiliki beberapa kolom yaitu ada “Date”, “Close”, “High”, “Low”, “Open”, “Volume”.

	Date	Close	High	Low	Open	Volume
1	2014-09-17	457.33	468.17	452.42	465.86	21056800.0
2	2014-09-18	424.44	456.86	413.10	456.86	34483200.0
3	2014-09-19	394.80	427.83	384.53	424.10	37919700.0
4	2014-09-20	408.90	423.30	389.88	394.67	36863600.0
5	2014-09-21	398.82	412.43	393.18	408.08	26580100.0

- **Library yang dipakai**

  Disini saya menggunakan beberapa library

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.graph_objects as go
```

- **Pra-pemrosesan Data**

```
df = df[['date', 'close', 'high', 'low', 'open', 'volume']].copy()

df.loc[:, 'date'] = pd.to_datetime(df['date'], errors='coerce')

numeric_cols = ['close', 'high', 'low', 'open', 'volume']

for col in numeric_cols:
    df.loc[:, col] = (
        df[col]
        .astype(str)
        .str.replace(",", "", regex=False)
        .str.replace("USD", "", regex=False)
        .str.replace("BTC-USD", "", regex=False)
        .str.strip()
        .replace("", np.nan)
        .astype(float)
    )


df = df.sort_values('date').dropna()
```

Tahapan pra-pemrosesan data meliputi:

1. Konversi kolom tanggal ke format datetime.
2. Pembersihan data numerik agar bertipe float.
3. Pengurutan data berdasarkan tanggal.
4. Penghapusan nilai kosong (missing values).

Langkah ini dilakukan untuk memastikan data siap digunakan dalam perhitungan statistik dan simulasi.

- **Perhitungan Log Return**

```
df['log_return'] = np.log(df['close'] / df['close'].shift(1))
df = df.dropna()

print(df['log_return'].tail())
```

```
4083   -0.017950
4084   -0.005212
4085    0.025159
4086    0.016742
4087   -0.007011
Name: log_return, dtype: float64
```

Log return dihitung menggunakan rumus:

$$r_t = \ln\left(\frac{P_t}{P_{t-1}}\right)$$

Di mana:

$P_t$adalah harga penutupan pada waktu ke-t

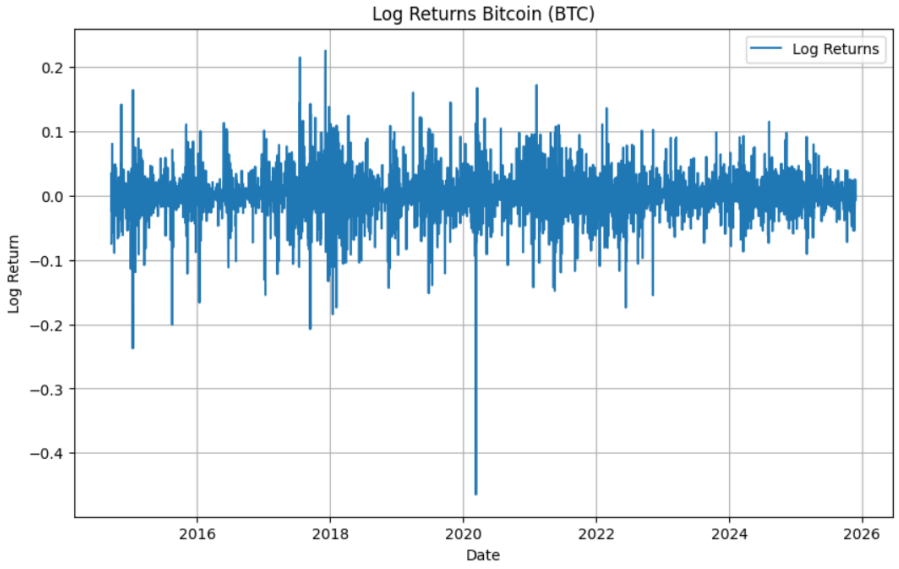
P_{t-1}adalah harga penutupan pada waktu sebelumnya

Log return dipilih karena:

- Bersifat aditif untuk periode waktu yang berbeda
- Lebih stabil secara statistik
- Umum digunakan dalam pemodelan keuangan

Hasil perhitungan log return menunjukkan fluktuasi positif dan negatif yang merepresentasikan kenaikan dan penurunan harga Bitcoin harian.

- **Visualisasi Log Return**



Visualisasi log return dilakukan untuk mengamati volatilitas harga Bitcoin dari waktu ke waktu. Grafik log return memperlihatkan bahwa Bitcoin memiliki fluktuasi yang signifikan, yang mengindikasikan risiko tinggi sekaligus peluang keuntungan besar.

- **Simulasi Monte Carlo**

```
mu = df['log_return'].mean()
sigma = df['log_return'].std()

print(f"Mean (μ): {mu}")
print(f"Volatility (σ): {sigma}")
```

```
last_price = df['close'].iloc[-1]
days = 30
n_simulations = 10000

simulation = np.zeros((n_simulations, days))

for i in range(n_simulations):
    price = last_price
    for d in range(days):
        shock = np.random.normal(mu, sigma)
        price = price * np.exp(shock)
        simulation[i, d] = price
```

Simulasi Monte Carlo dilakukan dengan langkah-langkah berikut:

1. Menghitung rata-rata (mean) log return sebagai estimasi drift (μ).
2. Menghitung standar deviasi log return sebagai volatilitas (σ).
3. Menghasilkan bilangan acak dari distribusi normal.
4. Mensimulasikan jalur harga Bitcoin ke depan berdasarkan parameter tersebut.

Model yang digunakan mengasumsikan bahwa pergerakan harga mengikuti Geometric Brownian Motion (GBM).

- **Hasil Simulasi**

```
simulation
```

```
array([[ 87498.12079539,  85099.76114286,  90403.37506182, ...,
        101756.87874349, 101061.2251379 ,  97273.01328334],
       [ 86583.14609996,  84484.23832961,  82025.11373859, ...,
         86852.89677377,  88973.53054346,  88380.00055292],
       [ 87612.46587108,  82369.42097549,  81289.97301907, ...,
         80373.72530599,  84576.8714012 ,  85417.07304003],
       ...,
       [ 88385.31816512,  87841.06179455,  96730.4552704 , ...,
        103427.69476424, 104898.00672204, 105179.79888927],
       [ 86928.73313115,  85990.82395863,  86011.67750417, ...,
         65221.55372381,  68929.04157027,  67587.46119203],
       [ 88399.4948419 ,  87055.13971965,  92160.4817031 , ...,
         90399.83434561,  87247.69715103,  84742.58545629]])
```

Hasil simulasi Monte Carlo berupa sekumpulan jalur harga Bitcoin yang merepresentasikan berbagai kemungkinan pergerakan harga di masa depan. Dari simulasi ini dapat dihitung:

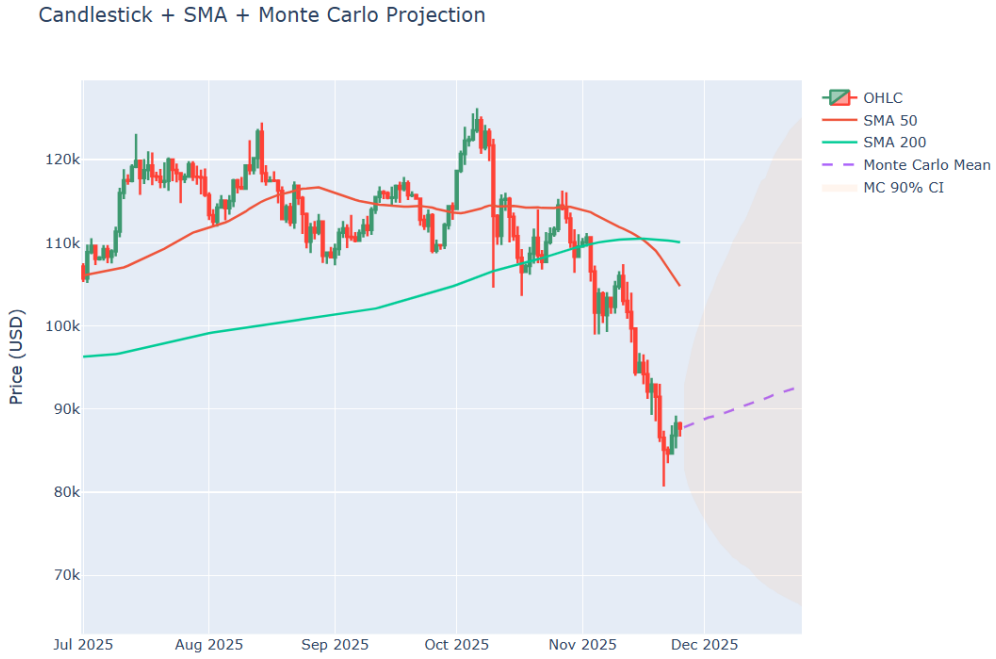
- Harga ekspektasi di akhir periode simulasi
- Rentang harga minimum dan maksimum
- Distribusi probabilitas harga

Hasil simulasi menunjukkan bahwa harga Bitcoin memiliki rentang kemungkinan yang luas, sejalan dengan karakteristik volatilitas tinggi aset kripto.

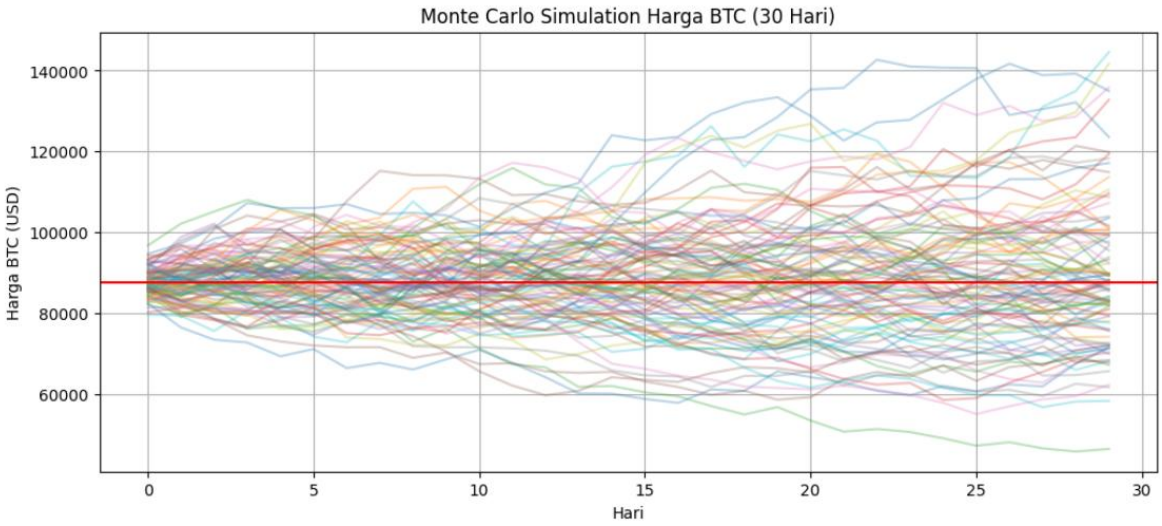
- **Visualisasi Candlestick dan Proyeksi**

  - **Harga asli**

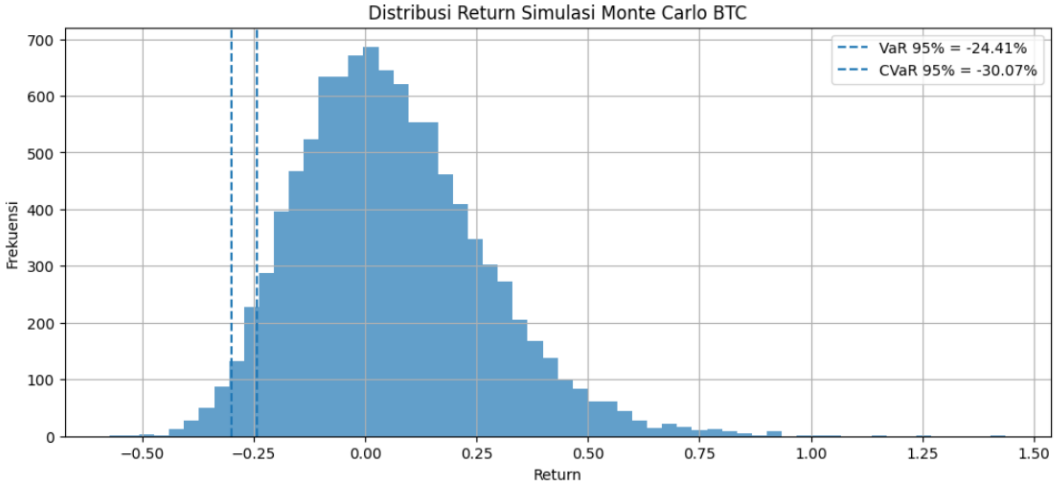
  - **Harga asli + hasil prediksi montecarlo**



➢ **Hasil prediksi monte carlo dengan 100 simulasi**



➢ **Distribusi Return simulasi monte carlo BTC**



Visualisasi candlestick digunakan untuk menampilkan pergerakan harga historis Bitcoin, khususnya pada tahun 2024. Hasil simulasi Monte Carlo dapat ditambahkan sebagai garis proyeksi untuk memberikan gambaran kemungkinan arah pergerakan harga di masa depan.

- ## Analisis output dan kesimpulan

1. Analisis Visualisasi Monte Carlo

Berdasarkan grafik *Monte Carlo Simulation Harga BTC (30 Hari)*, dapat diamati bahwa:

- Setiap garis merepresentasikan satu kemungkinan jalur pergerakan harga

Bitcoin selama 30 hari ke depan.

- Seluruh simulasi berawal dari harga awal yang sama, kemudian menyebar seiring waktu akibat unsur acak (randomness) yang berasal dari distribusi *log return* historis.
- Pola penyebaran harga semakin melebar pada hari ke-30, yang menunjukkan bahwa ketidakpastian (volatilitas) harga Bitcoin meningkat seiring bertambahnya horizon waktu.
- Garis horizontal merah menunjukkan harga awal BTC, yang berfungsi sebagai acuan untuk melihat potensi kenaikan maupun penurunan harga.

Dari simulasi tersebut terlihat bahwa:

- Sebagian skenario menunjukkan kenaikan harga yang signifikan
- Sebagian lain menunjukkan penurunan tajam, bahkan hingga jauh di bawah harga awal

Hal ini menegaskan bahwa Bitcoin merupakan aset dengan risiko tinggi (high-risk asset).

2. Analisis Value at Risk (VaR 95%)

Hasil:

VaR 95% = -0.2441 (≈ -24.41%)

Interpretasi:

- Dengan tingkat kepercayaan 95%, kerugian maksimum yang mungkin terjadi dalam 30 hari ke depan adalah sekitar 24.41% dari nilai investasi.
- Artinya, terdapat 5% kemungkinan kerugian yang lebih besar dari 24.41%.

Dalam konteks investasi:

- Jika seseorang berinvestasi Rp100 juta, maka kerugian maksimum yang diperkirakan (pada 95% confidence level) adalah sekitar Rp24,41 juta.

3. Analisis Conditional Value at Risk (CVaR 95%)

Hasil:

CVaR 95% = -0.3007 (≈ -30.07%)

Interpretasi:

- CVaR menunjukkan rata-rata kerugian pada skenario terburuk (5% kondisi ekstrem).
- Nilai ini lebih besar dari VaR karena CVaR memperhitungkan tail risk, yaitu kondisi pasar ekstrem.

  Dengan kata lain:

- Jika kerugian sudah melewati batas VaR 95%, maka rata-rata kerugian yang terjadi bisa mencapai 30.07%.

---

4. Perbandingan VaR dan CVaR

Metode	Nilai	Makna
VaR 95%	-24.41%	Kerugian maksimum normal
CVaR 95%	-30.07%	Kerugian rata-rata pada kondisi ekstrem

Perbedaan VaR dan CVaR yang cukup besar menunjukkan bahwa:

- Distribusi return Bitcoin memiliki ekor tebal (fat-tailed distribution)
- Risiko ekstrem tidak bisa diabaikan

➢ Kesimpulan Akhir

Berdasarkan simulasi Monte Carlo dan analisis risiko yang dilakukan, dapat disimpulkan bahwa:

1. Harga Bitcoin memiliki volatilitas tinggi, yang tercermin dari penyebaran jalur harga hasil simulasi.
2. Simulasi Monte Carlo efektif digunakan untuk memodelkan ketidakpastian harga aset kripto berdasarkan data historis.
3. Nilai VaR 95% sebesar 24.41% menunjukkan potensi kerugian yang signifikan dalam jangka pendek.
4. Nilai CVaR 95% sebesar 30.07% mengindikasikan bahwa pada kondisi pasar ekstrem, kerugian bisa jauh lebih besar dari batas VaR.
5. Oleh karena itu, Bitcoin tergolong aset dengan risiko tinggi, sehingga diperlukan manajemen risiko yang baik dalam pengambilan keputusan investasi.

CODE ASLI LENGKAP

[8]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.graph_objects as go
```

[9]:
```
df = pd.read_csv("/kaggle/input/betece/BTC_HISTORICAL_YFINANCE.csv")
```

[10]:
```
df = pd.read_csv("/kaggle/input/betece/BTC_HISTORICAL_YFINANCE.csv")

df = df.dropna(subset=["Date"])

numeric_cols = ["Close", "High", "Low", "Open", "Volume"]
df[numeric_cols] = df[numeric_cols].astype(float)

df.columns = ["Date", "Close", "High", "Low", "Open", "Volume"]
df["Date"] = pd.to_datetime(df["Date"])

df[numeric_cols] = df[numeric_cols].round(2)

df.head()
```

[10]:

	Date	Close	High	Low	Open	Volume
1	2014-09-17	457.33	468.17	452.42	465.86	21056800.0
2	2014-09-18	424.44	456.86	413.10	456.86	34483200.0
3	2014-09-19	394.80	427.83	384.53	424.10	37919700.0
4	2014-09-20	408.90	423.30	389.88	394.67	36863600.0
5	2014-09-21	398.82	412.43	393.18	408.08	26580100.0

untuk visualisasi candle stik, MA, dan SMA ini saya ambil code nya dari salah satu exchange yang menawarkan open resources yaitu CoinGecko https://www.coingecko.com/learn/category/api

[11]:
```
def load_data(csv_path):
    df = pd.read_csv(csv_path)

    df.columns = [c.split()[0] for c in df.columns]

    df = df.dropna(subset=["Date"])

    df["Date"] = pd.to_datetime(df["Date"], errors="coerce")
    df = df.dropna(subset=["Date"])

    num_cols = ["Close", "High", "Low", "Open", "Volume"]
```

```
    num_cols = ["Close", "High", "Low", "Open", "Volume"]
    df[num_cols] = df[num_cols].astype(float)

    df = df.sort_values("Date").reset_index(drop=True)

    df.columns = df.columns.str.lower()

    return df
```

[12]:
```
def add_sma(df, window_short, window_long):
    df["sma_short"] = df["close"].rolling(window_short).mean()
    df["sma_long"] = df["close"].rolling(window_long).mean()
    return df
```

[13]:
```
def plot_candlestick_sma(df, window_short, window_long):

    fig = go.Figure()

    fig.add_trace(go.Candlestick(
        x=df["date"],
        open=df["open"],
        high=df["high"],
        low=df["low"],
        name='OHLC'
    ))

    fig.add_trace(go.Scatter(
        x=df["date"],
        y=df["sma_short"],
        mode='lines',
        line=dict(color='blue'),
        name=f"SMA {window_short}"
    ))

    fig.add_trace(go.Scatter(
        x=df["date"],
        y=df["sma_long"],
        mode='lines',
        line=dict(color='orange'),
        name=f"SMA {window_long}"
    ))

    fig.update_layout(
        title=f"Candlestick + SMA {window_short} & {window_long}",
        xaxis_title="Date",
        yaxis_title="Price (USD)",
        xaxis_rangeslider_visible=False,
        height=600
    )

    fig.show()
```

```
plot_candlestick_sma(df, 50, 200)
```

SIMULASI MONTE CARLO

```
df = df[['date', 'close', 'high', 'low', 'open', 'volume']]
df.columns = ['date', 'close', 'high', 'low', 'open', 'volume']
```

+ Code + Markdown

Cleaning & Tipe Data

[16]:
```
df = df[['date', 'close', 'high', 'low', 'open', 'volume']].copy()

df.loc[:, 'date'] = pd.to_datetime(df['date'], errors='coerce')

numeric_cols = ['close', 'high', 'low', 'open', 'volume']

for col in numeric_cols:
    df.loc[:, col] = (
        df[col]
        .astype(str)
        .str.replace(",", "", regex=False)
        .str.replace("USD", "", regex=False)
        .str.replace("BTC-USD", "", regex=False)
```

```
        .replace("", np.nan)
        .astype(float)
    )


df = df.sort_values('date').dropna()
```

Hitung Log Return

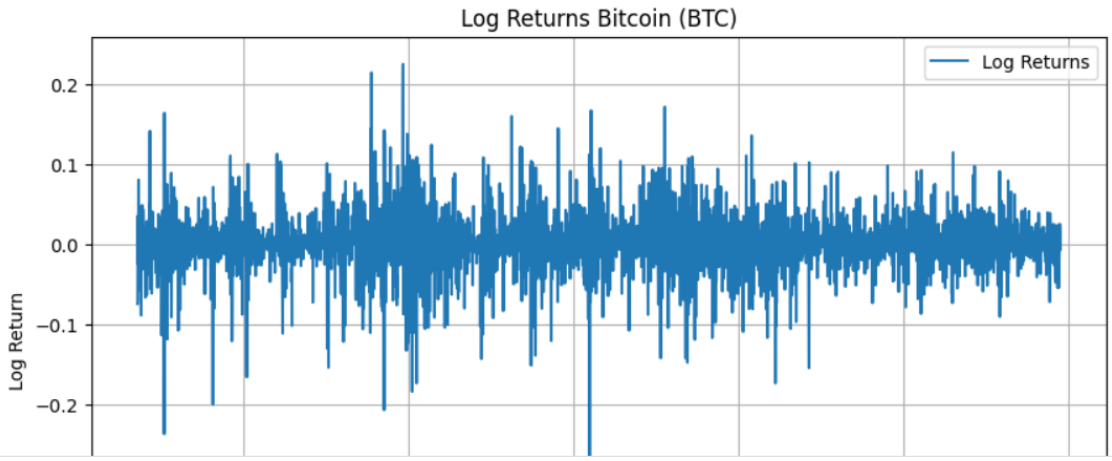
[17]:
```
df['log_return'] = np.log(df['close'] / df['close'].shift(1))
df = df.dropna()

print(df['log_return'].tail())
```

```
4083   -0.017950
4084   -0.005212
4085    0.025159
4086    0.016742
4087   -0.007011
Name: log_return, dtype: float64
/usr/local/lib/python3.11/dist-packages/pandas/core/arraylike.py:399: RuntimeWarning:

invalid value encountered in log
```

[18]:
```
plt.figure(figsize=(10, 6))
plt.plot(df['date'], df['log_return'], label='Log Returns')
plt.title('Log Returns Bitcoin (BTC)')
plt.xlabel('Date')
plt.ylabel('Log Return')
plt.legend()
plt.grid(True)
plt.show()
```



Estimasi Parameter Monte Carlo

[19]:
```
mu = df['log_return'].mean()
sigma = df['log_return'].std()

print(f"Mean (μ): {mu}")
print(f"Volatility (σ): {sigma}")
```

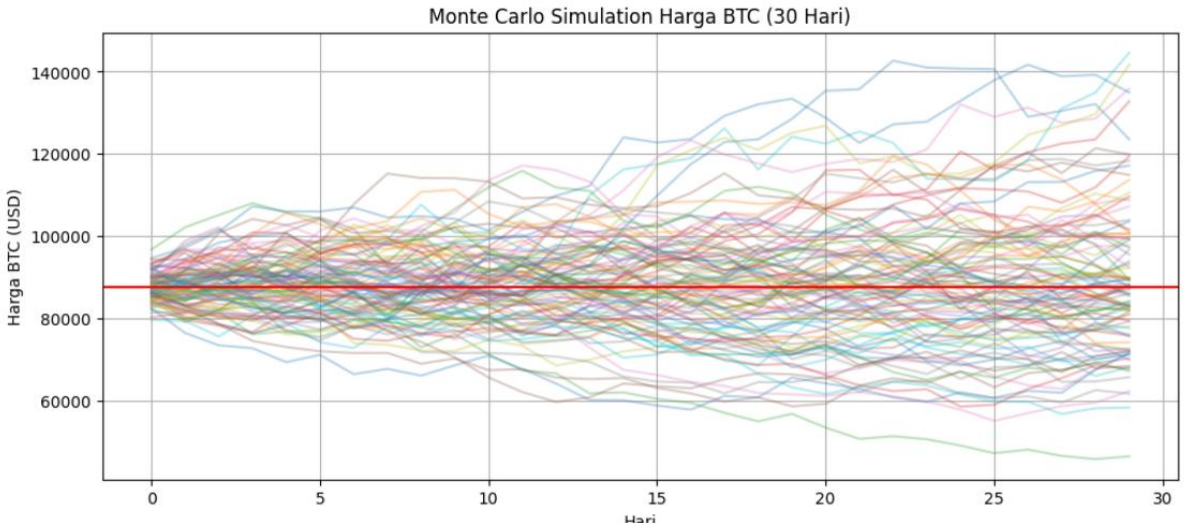
```
Mean (μ): 0.0012859644781024041
Volatility (σ): 0.03552381675868391
```

Simulasi Monte Carlo (30 Hari)

[20]:
```
last_price = df['close'].iloc[-1]
days = 30
n_simulations = 10000

simulation = np.zeros((n_simulations, days))
```

```
plt.plot(simulation[:100].T, alpha=0.3)
plt.title("Monte Carlo Simulation Harga BTC (30 Hari)")
plt.axhline(y = last_price, color = 'r', linestyle= '-')
plt.xlabel("Hari")
plt.ylabel("Harga BTC (USD)")
plt.grid(True)
plt.show()
```



Hitung VaR & CVaR (95%)

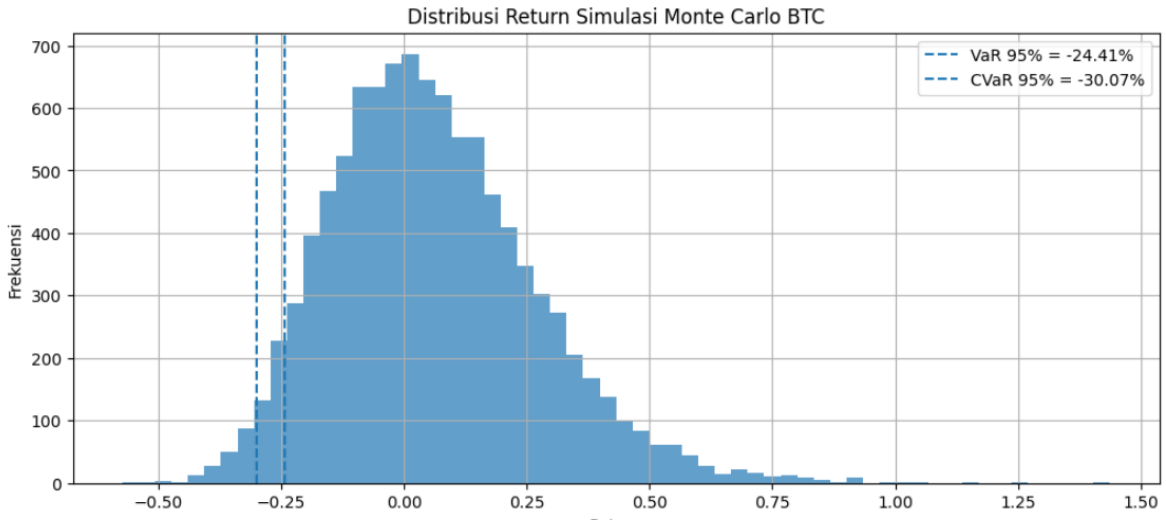
[22]:
```
final_prices = simulation[:, -1]
returns = (final_prices - last_price) / last_price

VaR_95 = np.percentile(returns, 5)
CVaR_95 = returns[returns <= VaR_95].mean()

print("VaR 95% :", VaR_95)
print("CVaR 95%:", CVaR_95)
```

```
VaR 95% : -0.24407162614891628
CVaR 95%: -0.30067591706632835
```

```
plt.title("Distribusi Return Simulasi Monte Carlo BTC")
plt.xlabel("Return")
plt.ylabel("Frekuensi")
plt.legend()
plt.grid(True)
plt.show()
```



MASUKAN HASIL SIMULASI MONTE CARLO KE CANDLE STICK

[24]:
```
simulation
```

[24]:
```
array([[ 91981.6257996 ,  94503.35387177,  98414.32747624, ...,
         84793.32727104,  84273.67454681,  88058.96128691],
       [ 87996.51828051,  86992.35763137,  85387.81091121, ...,
         85644.84133845,  85585.03892998,  82109.1904853 ],
       [ 82148.06272624,  79852.39970605,  81512.8565629 , ...,
         46524.36344802,  45734.62740059,  46391.89019078],
       ...,
       [ 88279.37536724,  86052.80366996,  87826.10653687, ...,
        109891.87864281, 106955.98845536, 110829.47630153],
       [ 89624.19160963,  87690.04902303,  91167.15242326, ...,
         98128.81890312,  98474.1667027 ,  99393.08062536],
       [ 82445.78759967,  81231.20008526,  86176.17834397, ...,
         93624.74677743,  92788.98844463,  95602.18107877]])
```

[25]:
```
mc_mean = simulation.mean(axis=0)
mc_p5   = np.percentile(simulation, 5, axis=0)
mc_p95  = np.percentile(simulation, 95, axis=0)
```

[26]:
```
last_date = df['date'].iloc[-1]
future_dates = pd.date_range(
    start=last_date + pd.Timedelta(days=1),
    periods=len(mc_mean),
    freq='D'
)
```

[27]:
```
def plot_candlestick_sma_mc(df, window_short, window_long,
                            future_dates, mc_mean, mc_p5, mc_p95):

    fig = go.Figure()

    fig.add_trace(go.Candlestick(
        x=df["date"],
        open=df["open"],
        high=df["high"],
        low=df["low"],
        close=df["close"],
        name='OHLC'
    ))

    fig.add_trace(go.Scatter(
        x=df["date"],
        y=df["sma_short"],
        mode='lines',
```

[28]:
```
df = load_data("/kaggle/input/betece/BTC_HISTORICAL_YFINANCE.csv")
df = add_sma(df, 50, 200)

df = df[df['date'] >= '2025-07-01'].reset_index(drop=True)

plot_candlestick_sma_mc(
    df,
    window_short=50,
    window_long=200,
    future_dates=future_dates,
    mc_mean=mc_mean,
    mc_p5=mc_p5,
    mc_p95=mc_p95
)
```