Monte Carlo simulation is a robust statistical approach that uses stochastic variables to emulate the dynamics of complex systems. In the financial markets, this methodology facilitates the prediction of future price trajectories for stocks based on their historical attributes - expected returns and volatility.

https://www.investing.com/equities/bank-uralsib-historical-data

**import numpy as np**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**from math import sqrt**

This part covers the inclusion of several essential libraries:

numpy (np) - offers functions for performing mathematical operations on arrays and for generating random numbers

pandas (pd) and read\_csv - make it easy to manipulate tabular data and ingest CSV files

matplotlib.pyplot (plt) - provides data visualization capabilities

sqrt from mathematics - used to calculate square roots

These libraries are an integral part of data analysis and simulations, as they provide the necessary tools for data processing, mathematical calculations, and visual representation.

**data = pd.read\_csv('BANK URALSIB Stock Price History.csv', parse\_dates=['Date'], dayfirst=True, index\_col='Date')**

**data['Price'] = pd.to\_numeric(data['Price'], errors='coerce')**

**data.dropna(subset=['Price'], inplace=True)**

**data.sort\_index(inplace=True)**

**if data.empty:**

**raise ValueError("DataFrame is empty after loading and cleaning. Check the CSV file.")**

**print(f"Data loaded successfully. Number of observations: {data.shape}")**

This code block is dedicated to loading and preparing the data from the CSV file. I use pd.read\_csv, giving it specific instructions: to understand the dates in the 'Date' column as such (parse\_dates) and to know that the day is the first (dayfirst), then to use these dates to index the data (index\_col). To avoid problems with calculations, the prices are converted to numeric values with pd.to\_numeric. I remove any rows with missing data (dropna), as they cannot be used in the analysis. Sorts everything is sorted by date (sort\_index) to have the correct time sequence. Finally, a check is made (if data.empty) to prevent it from continuing if something went wrong with the loading.

**start\_price = data['Price'].iloc[0]**

**end\_price = data['Price'].iloc[-1]**

**start\_date = data.index[0]**

**end\_date = data.index[-1]**

**days = (end\_date - start\_date).days**

**if days <= 0 or start\_price <= 0:**

**print("Warning: Invalid period or starting price for CAGR calculation. We set mu = 0.")**

**cagr = 0.0**

**else:**

**cagr = ((end\_price / start\_price) \*\* (365.0 / days)) - 1**

**mu = cagr**

**print(f'Calculated CAGR (mu): {mu:.4f} ({mu\*100:.2f}%)')**

Determining the expected return (mu) is key to our simulation, as it sets the expected price trend. For this purpose, the CAGR (Compound Annual Growth Rate) is calculated. This indicator is preferred over a simple average return because it correctly reflects the effect of accumulation (compound interest) on the investment over the years. The calculation involves finding the initial and final prices, the period in days between them, and applying the formula ((FinalPrice / InitialPrice) ^ (365 / NumberDays)) - 1. Annualization is achieved by exponentiation, and subtracting 1 gives the percentage result. After checking the data, the calculated CAGR becomes our value for mu.

**returns = data['Price'].pct\_change()**

**returns = returns.dropna()**

**if returns.empty:**

**print("Warning: Cannot calculate daily returns. Set vol = 0.")**

**volume = 0.0**

**else:**

**vol = returns.std() \* sqrt(252)**

**print(f"Calculated annual volatility (vol): {vol:.4f} ({vol\*100:.2f}%)")**

Besides the expected trend (mu), the other key parameter for this simulation is volatility, which measures the degree of price fluctuations. It is calculated by first obtaining the daily percentage changes (pct\_change()), then finding their standard deviation (std()) – this is the daily volatility. To bring it to an annual basis, I scale it by multiplying by the square root of 252. I check if there is data before the calculation. The importance of volatility is to determine the amount of random movements in the simulated price paths. It represents the "noise" around the trend and directly affects the degree of uncertainty – higher volatility leads to a larger potential price range.

**S = data['Price'].iloc[-1]**

**T = 252**

**num\_simulations = 1000**

**print(f'\n--- Simulation parameters ---')**

**print(f'Starting price (S): {S:.4f}')**

**print(f'Simulation period (T): {T} days')**

**print(f'Number of simulations: {num\_simulations}')**

I define the main inputs that will drive the simulation: S serves as the starting point (the last price), T defines the forecast duration (in trading days, e.g. 1 year), and num\_simulations controls the number of scenarios generated. Together, these parameters establish the boundaries of the experiment. The choice of num\_simulations is an important balance – a larger number improves the estimation of the probability distribution, but increases the computational burden.

**result = []**

**plt.figure(figsize=(12, 7))**

**for i in range(num\_simulations):**

**daily\_returns\_sim = np.random.normal(mu / T, vol / sqrt(T), T) + 1**

**price\_list = [S]**

**for x in daily\_returns\_sim:**

**price\_list.append(price\_list[-1] \* x)**

**result.append(price\_list[-1])**

**plt.plot(price\_list, alpha=0.1, color='blue')**

This block executes the basic Monte Carlo simulation loop num\_simulations times. In each iteration, an independent price path is generated: np.random.normal creates T random daily returns (with center mu/T and spread vol/sqrt(T)), which are converted into multipliers (+1). The price series is iteratively constructed from S by successive multiplication by these factors. The final price is stored in result. Each path is visualized with alpha=0.1. This process of repeatedly generating random paths is the essence of Monte Carlo simulation for estimating future prices using the geometric Brownian motion model.

**plt.title(f'Monte Carlo Simulation: {num\_simulations} Price Paths for Bank Uralsib (USBN) for {T} days')**

**plt.xlabel('Trading Days')**

**plt.ylabel('Simulated Price (RUB)')**

**plt.grid(True, alpha=0.3)**

**plt.show()**

Here is shows the graph on simulations . I add title , labels on axes and grid ( plt.grid ()) for clarity , after which we visualize with plt.show (). This The schedule is important because effectively illustrates the concept for " funnel " on uncertainty " – how the possible range on the prices is growing proportionally on temporal horizon on the forecast .

**plt.figure ( figsize =(10, 6))**

**plt.hist (result, bins=50, color=' skyblue ', edgecolor ='black', alpha=0.7)**

**plt.title (' Distribution on The extremes Prices after The simulation ')**

**plt.xlabel (' End Price (RUB)')**

**plt.ylabel (' Frequency ')**

Here I create histogram on the extremes prices from all simulations . Generated​ new figure and applies plt.hist with 50 intervals (bins=50) and set colors . This visualization shows the distribution on the probabilities for the final price , as us helps yes understand​ range and probability on the possible exits .

**mean\_price = np.mean (result)**

**quantile\_5 = np.percentile (result, 5)**

**quantile\_95 = np.percentile (result, 95)**

**print(f'\n--- Analysis on results ({ num\_simulations } simulations ) ---')**

**print( f'Average final price : {mean\_price:.4f} RUB')**

**print(f'5% quantile ( pessimistic ): {quantile\_5:.4f} RUB')**

**print(f'95% quantile ( optimistic ): {quantile\_95:.4f} RUB')**

**plt.axvline ( mean\_price , color='black', linestyle ='solid', linewidth=2, label= f'Mean : {mean\_price:.2f}')**

**plt.axvline (quantile\_5, color='red', linestyle ='dashed', linewidth=2, label=f'5% quantile : {quantile\_5:.2f}')**

**plt.axvline (quantile\_95, color='green', linestyle ='dashed', linewidth=2, label=f'95% quantile : {quantile\_95:.2f}')**

**plt.legend ()**

**plt.grid (True, alpha=0.3)**

**plt.show ()**

Here I quantify the results from the simulation , as calculate the mean ( mean\_price ), 5th (quantile\_5) and 95th (quantile\_95) percentile on the extremes prices . These values is are printed and add as vertical lines to the histogram ( black for medium , red / green dotted for percentiles ), along with a legend ( plt.legend ()). The meaning is in the score on risk : average price shows expectations score , and the percentiles outline 90%- complexion confidential interval , which helps at the taking on informed investment solutions .

***Analysis on The simulations Data for Bank Uralsib***

*The results from Monte Carlo the simulation us give multilayer look on the potential future on the shares of Bank Uralsib .*

*Basic Input and Output Data :*

*CAGR ( Annual Average Growth Rate ) growth ): 14.05% ( Shows positive expected trend based​ on history ).*

*Annual Volatility : 244.22% ( Signals) for exceptionally high degree on uncertainty and hesitation ).*

*Home price ( for simulation ): 0.1495 RUB.*

*Medium forecast price ( after 1 year): 0.1824 RUB ( Average result from 1000 simulations ).*

*5th percentile ( Pessimistic ): 0.0001 RUB ( Scenario with almost full loss ).*

*95th percentile ( Optimistic ): 0.4802 RUB ( Scenario with significant profit ).*

*Interpretation on The average Estimated Price :*

*The value from 0.1824 RUB represents the arithmetic mean final price from all 1000 simulations . It suggests expected height from about 22% compared to the initial price , which is in line with the calculations historical CAGR of 14.05%.*

*Key Note : This is important . average value not to be accepted as the " most likely " price . The histogram shows asymmetrically distribution , typical for shares , where the most common values ( mode ) can yes are lower from average . Despite this , the average value remains useful as indicator for the expected long-term profitability .*

*Quantiles as Meter on Risk :*

*5th and 95th percentiles outline the frames at 90% of simulated results , giving us idea for the potential range on movement :*

*Risk from Loss (5% percentile = 0.0001 RUB): This the result is exceptional worrying . He shows that​ according to the simulation there is a 5% probability the investment yes losses almost the whole you value within​ on one year .*

*Potential for Profit (95% percentile = 0.4802 RUB): There is also a 5% probability the price yes exceeded this level , carrying profit from over 220%.*

*Main conclusion : The huge difference between these two percentile ( of almost zero up to 0.48 RUB) is directly consequence from the extreme annual volatility (244.22%) and highlights the huge uncertainty in the forecast .*

*Consequences from The extreme Volatility (244.22%):*

*Such high volatility is rare and there is significant consequences :*

*Tall Risk and Potential : Explains why the simulation generates as scenarios with almost full loss , as well as scenarios with huge profits .*

*" Fat " tails ": The distribution on the returns probably there are " thick" tails " which means higher from the normal one probability for extreme ( very) good or very bad ) results .*

*Inadequacy on standard Metrics : Traditional measures for risk as VaR can yes no catch adequately the potential losses at such volatility .*

*Behavioral Aspect : The Strong fluctuations can yes provoke emotional reactions among investors , which only by myself you can yes contributed for the market instability .*

*Asymmetry on The distribution :*

*The histogram shows positive asymmetry ( longer right tail ). This is typical for shares and means :*

*Most simulated extreme prices are grouped at the bottom part on range .*

*Exists smaller probability for achievement on significantly higher prices ( long right tail ).*

*The median price (50th percentile ) is probably lower from the average (0.1824 RUB).*

*Comparison Risk / Return :*

*If comparable the expected average return ( about 22%) with risk-free alternative ( e.g. 5-7%), we get risky premium from about 15-17%. The main question for the investor has given this one premium is adequate remuneration for poets risk , especially considering the 5% chance for almost full loss .*

*Practical Conclusions for Various Investors :*

*The simulation suggests the following approaches :*

*Conservative Investors : The High risk from significant loss does this one action probably inappropriate .*

*Speculative Investors : The Big One potential for profit can to be tempting , but it is reasonable the exposure yes be small part from the general wallet .*

*Diversified Wallets : Small position can yes raise the expected profitability on the wallet , but with security will also increased his volatility .*

*Time Horizon : Extreme hesitations make her more suitable for long -term investors horizon , capable to " survive " big temporary declines .*