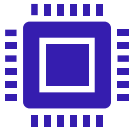
Python 101

Algo Markov Chain Monte Carlo

Le but du Workshop

Introduction

Plongez dans le fascinant univers des chaînes de Markov et leur application dans la science des données et l'interaction avec les API.



Public Cible

Ce cours est destiné aux personnes ayant un niveau intermédiaire en programmation jusqu'aux développeurs chevronnés désireux d'approfondir leurs connaissances en modélisation stochastique.

Objectif

L'objectif est de vous doter de compétences avancées et directement applicables dans l'utilisation des chaînes de Markov et Monte Carlo Simulation pour la résolution de problèmes complexes.

Langage

Les concepts et applications seront enseignés et implémentés en Python, un langage puissant pour la modélisation statistique et la science des données.

Table de matière

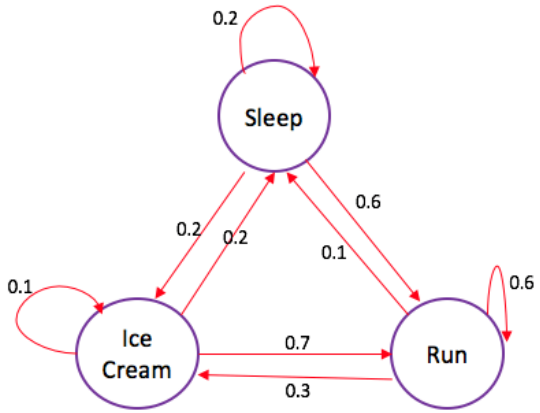
APIs et Sources de Données: Maîtrisez l'art de récupérer des données via des APIs pour alimenter vos modèles de chaînes de Markov, une compétence clé pour traiter des données réelles et dynamiques.

Science des Données avec Pandas: Apprenez les principes fondamentaux des chaînes de Markov et Monte Carlo Simulation, y compris la théorie sous-jacente, la construction de modèles, et l'analyse de la probabilité de transition pour prédire des états futurs.

Visualisation de Données: Découvrez comment visualiser efficacement les matrices de transition, les états stables, et les parcours de chaînes de Markov à l'aide de bibliothèques Python telles que Matplotlib.

Projet sur l'Application des Chaînes de Markov et Monte Carlo Simulation **en** Finance : Explorez comment les chaînes de Markov et Monte Carlo Simulation peuvent être utilisées pour modéliser les fluctuations des marchés financiers.

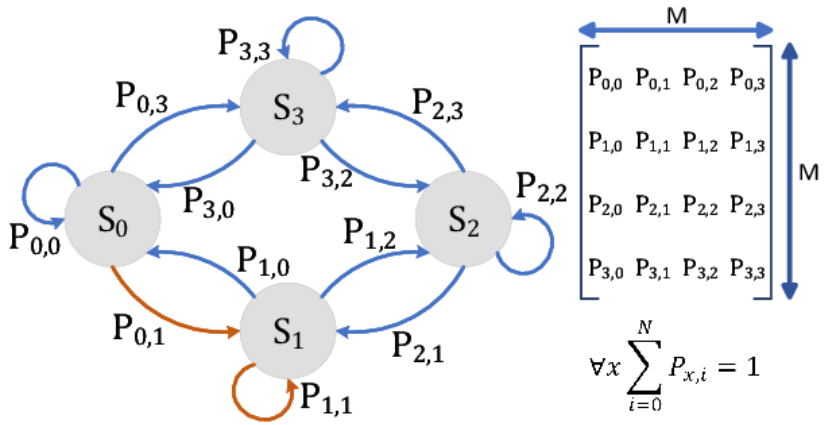
Markov Chain



CURRENT STATE

NEXT STATE

	SLEEP	RUN	ICE CREAM
SLEEP	0.2	0.6	0.2
RUN	0.1	0.6	0.3
ICE CREAM	0.2	0.7	0.1



https://setosa.io/ev/markov-chains/

https://www.datacamp.com/tutorial/markov-chains-python-tutorial

Utilisation

- Prediction de états
- Google Page Rank
- Probability
- Optimisation de échantionnage aléatoire

Monte Carlo

Intégration par Monte Carlo

$$\frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p_{uni}(x_i)}$$

```
m = 0
for i in range(N):
     xi = a+(b-a)*rand()
     pdf = 1/(b-a)
     m += f(xi)/pdf
m *= 1/N
```

$$\frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}$$

Générateur de nombre aléatoire

```
m = 0
for i in range(N):
     xi = sample(rand())
     m += f(xi)/pdf(xi)
m *= 1/N
```

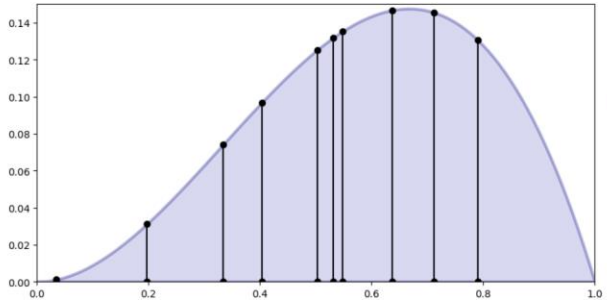
Forme générale

Poids

Estimateur de Monte Carlo

Pourquoi il faut diviser par la densité des échantillons?

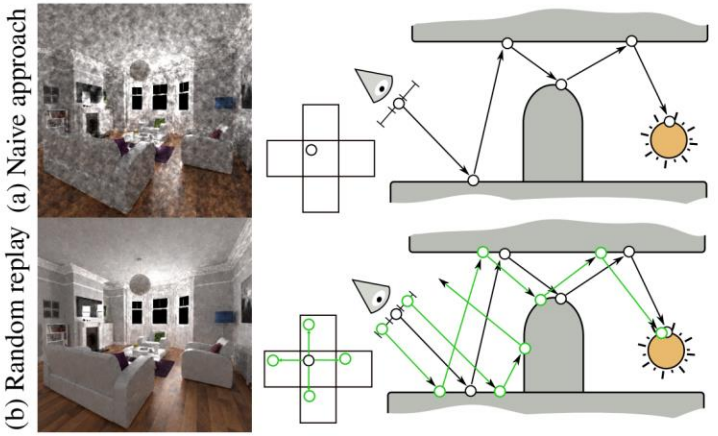
$$\int_a^b f(x)dx \approx \frac{1}{N}\sum_{i=1}^{N}(b-a)f(x_i) = \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p_{uni}(x_i)}$$



Densité uniforme

$$p_{uni}(x) = \frac{1}{b-a}$$

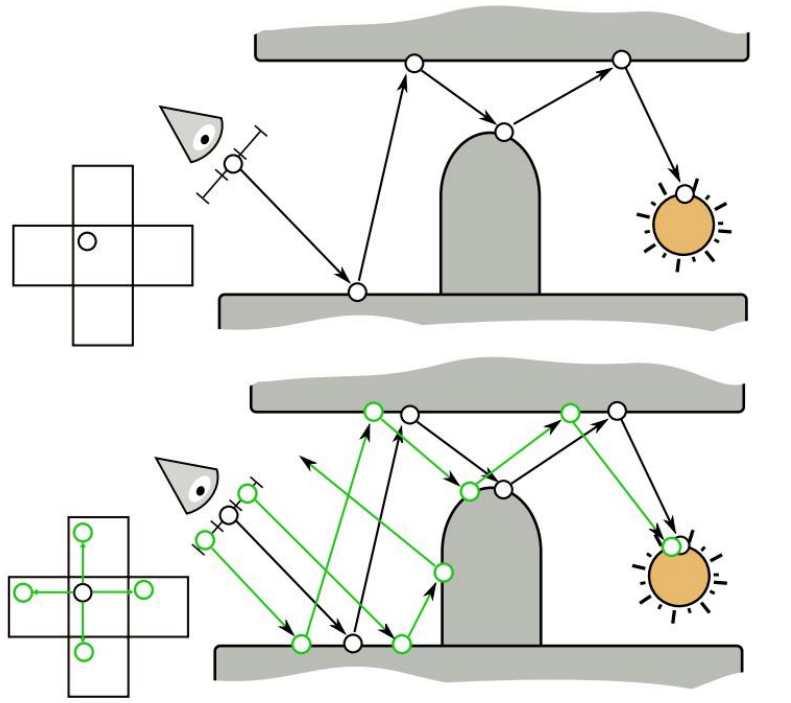
$$\int_a^b p_{uni}(x)\mathrm{d}x = 1$$



https://medium.com/@juee_thete/understanding-monte-carlo-simulation-and-its-implementation-with-python-3ecacb958cd4

Utilisation

- Rendu 3D
- Random Walk
- Prediction AI
- Calcul de Risque
- Casino
- Probabilité



6. Modelling stock prices

Here is an example when classical Monte Carlo method cannot be applied to mo
el stock prices.

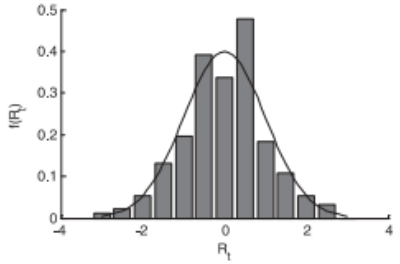


Fig. 8. Distribution of normalized logarithms of continuous day returns.

The histogram of normalized logarithms of continuous day returns R_i of
Corporation (TESO) is depicted in figure 8. Although the hypothesis of normality
cepted, there exist two peaks. If one is confident about the shape of histogram, th
sumption of normality should be rejected and standard Monte Carlo cannot be app

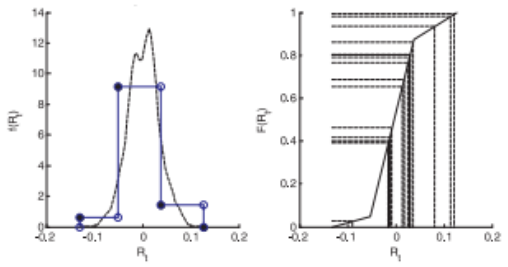
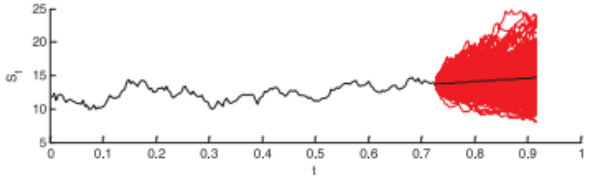


Fig. 9. Forecasting the stock prices.

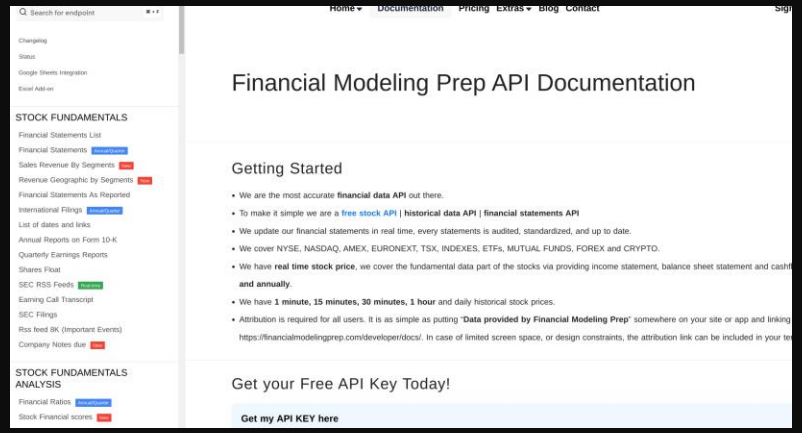
Constructing kernel density estimate for $r_i = \frac{S_i - S_{i-1}}{S_{i-1}}$ using Gaussian kernel function also gives PDF with 2 peaks (figure 9). MCMC with piecewise-linear distribution as a proposal density was applied to this PDF.



Requirement

- panda
- lxml
- bs4
- httpx
- backtesting
- yfinance
- pandas_ta
- scipy
- rich
- matplotlib

```python
def make_api_request(api_endpoint, params):
    with httpx.Client() as client:
        # Make the GET request to the API
        response = client.get(api_endpoint, params=params)
        if response.status_code == 200:
            return response.json()
        print("Error: Failed to retrieve data from API")
        return None
```




```python
def get_historical_price_full_stock(symbol):
    api_endpoint = f"{BASE_URL_FMP}/historical-price-full/{symbol}"
    params = {"apikey": FMP_API_KEY}

    return make_api_request(api_endpoint, params)
```

1. Les Données

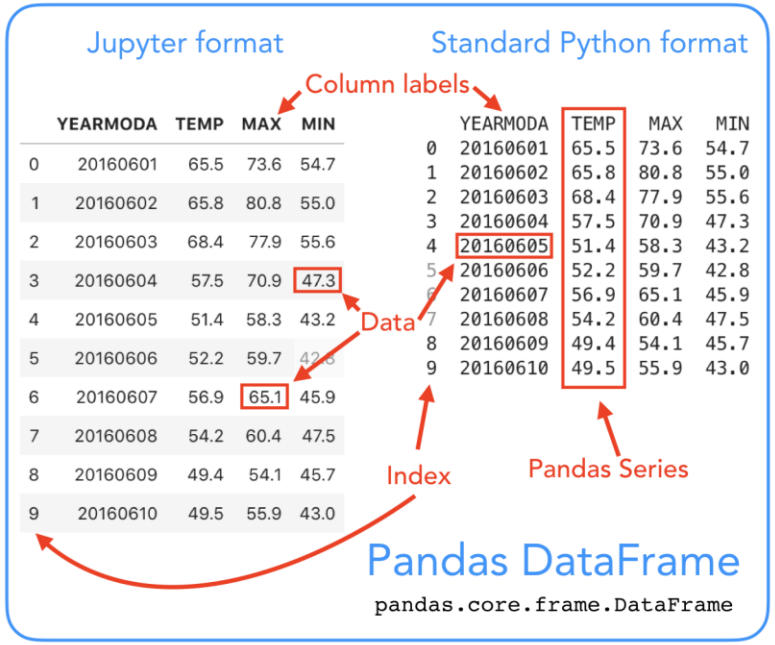
DataFrame

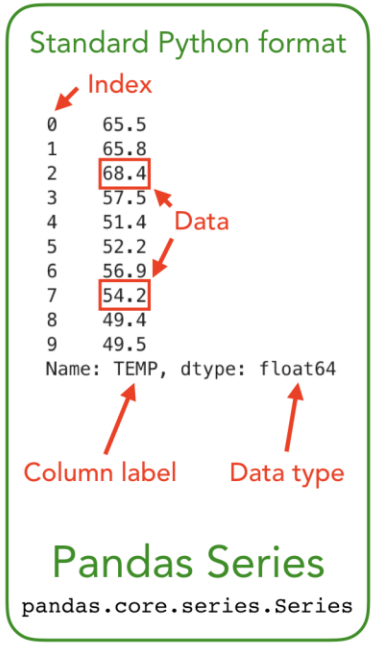
- **Structure Tabulaire**
- **Trier, Filtrer, Agréger**
- **Intégration avec Matplotlib**
- **Column, Row**

```python
# Load our necessary libraries
import pandas as pd
import numpy as np
```

```python
# Create data into CSV (that we'll import later on)
# Let's say the data is for a veterinarian keeping tabs on his clients.

raw_data = {'pet_name': ['Woof', 'Chester', 'Rex', 'Mystery', 'Pumpkin'],
        'pet_last_name': ['Smith', 'Kim', "", 'Taylor', ""],
        'good_pet_score': [96, 34, 89, 92, 79],
        'type': ['dog', 'cat', 'mini-dinosaur', "unknown", "bird"],
        'amount_owed': ["5000", "9,000", 570, 622, 190]}
df = pd.DataFrame(raw_data, columns = ['pet_name', 'pet_last_name', 'good_pet_score', 'type', 'amount_owed'])
df
```

	pet_name	pet_last_name	good_pet_score	type	amount_owed
0	Woof	Smith	96	dog	5000
1	Chester	Kim	34	cat	9,000
2	Rex		89	mini-dinosaur	570
3	Mystery	Taylor	92	unknown	622
4	Pumpkin		79	bird	190

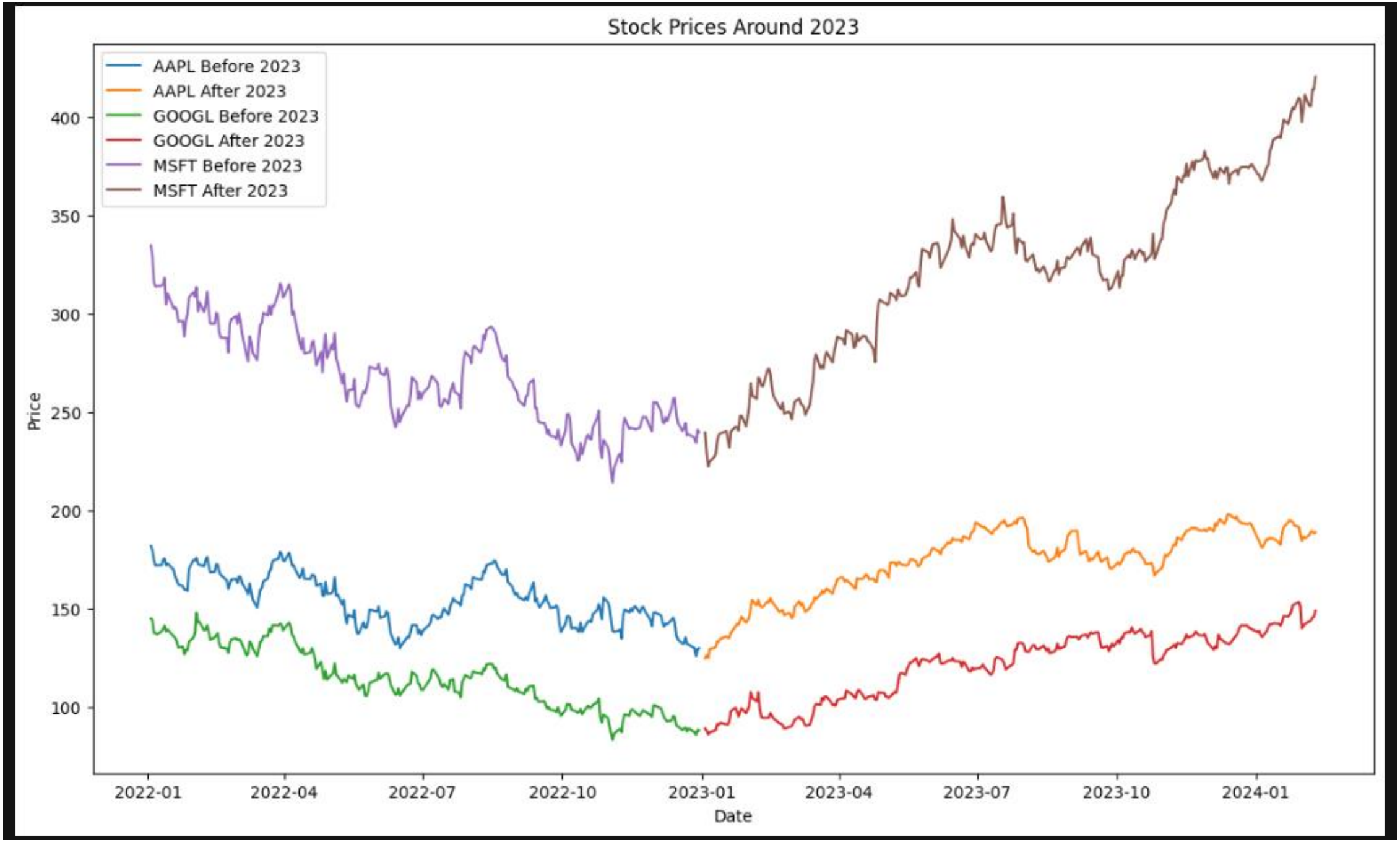


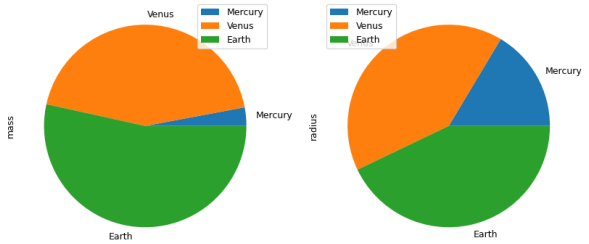


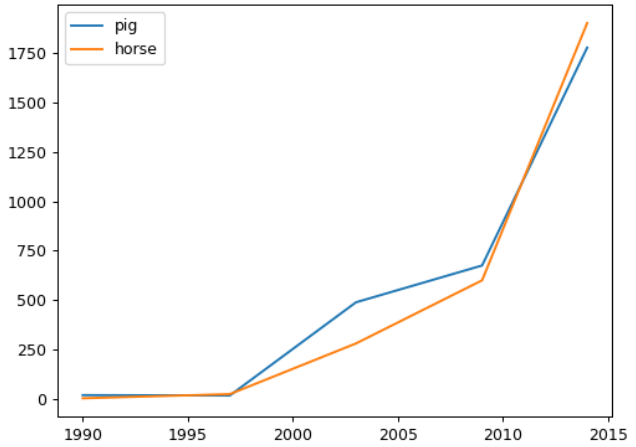
```
january_2023_index = data[(data['date'] < '2023-01-01')].index
january_2023_index
```

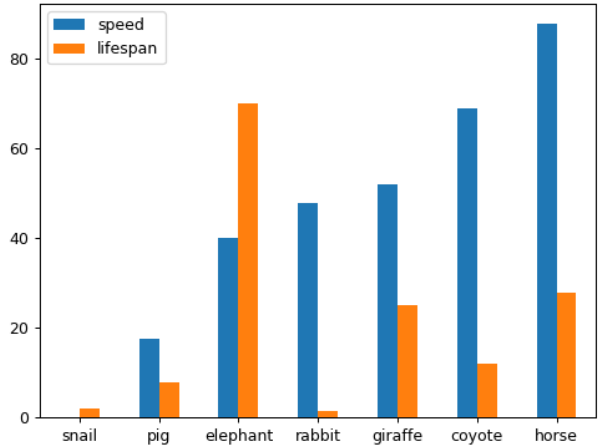
```
prices_before_january_2023 = data.loc[january_2023_index]
prices_before_january_2023

# Assuming prices_before_january_2023 is your DataFrame
prices_before_january_2023 = prices_before_january_2023.rename(columns={
    'open': 'Open',
    'high': 'High',
    'low': 'Low',
    'close': 'Close',
    'volume': 'Volume'  # Only if you have a volume column
})

# Drop any additional columns that are not required
required_columns = ['date', 'Open', 'High', 'Low', 'Close', 'Volume']
prices_before_january_2023 = prices_before_january_2023[required_columns]

# sort by date ascending
prices_before_january_2023 = prices_before_january_2023.sort_values(by=['date'], ascending=True)

prices_before_january_2023
```

2. Préparation des Données


Stock Prices Around 2023
AAPL Before 2023
AAPL After 2023
GOOGL Before 2023
GOOGL After 2023
MSFT Before 2023
MSFT After 2023
Price
400
350
300
250
200
150
100
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01
Date



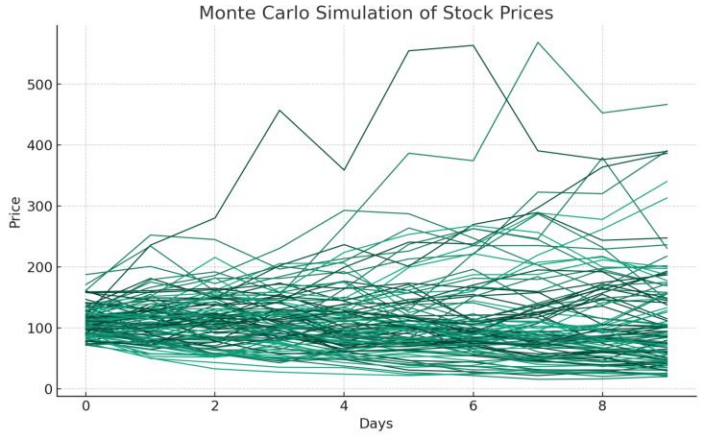


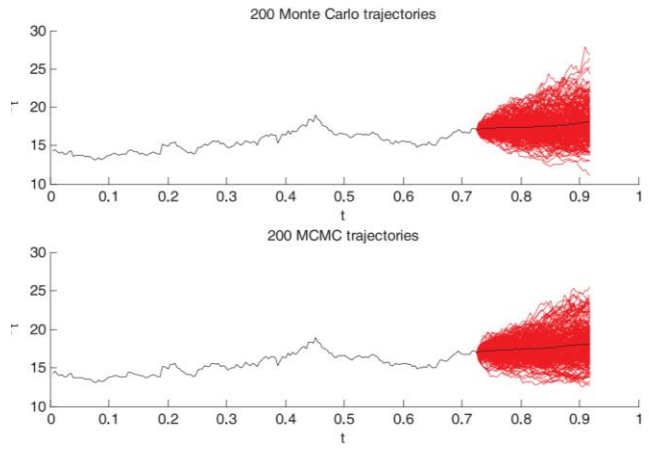


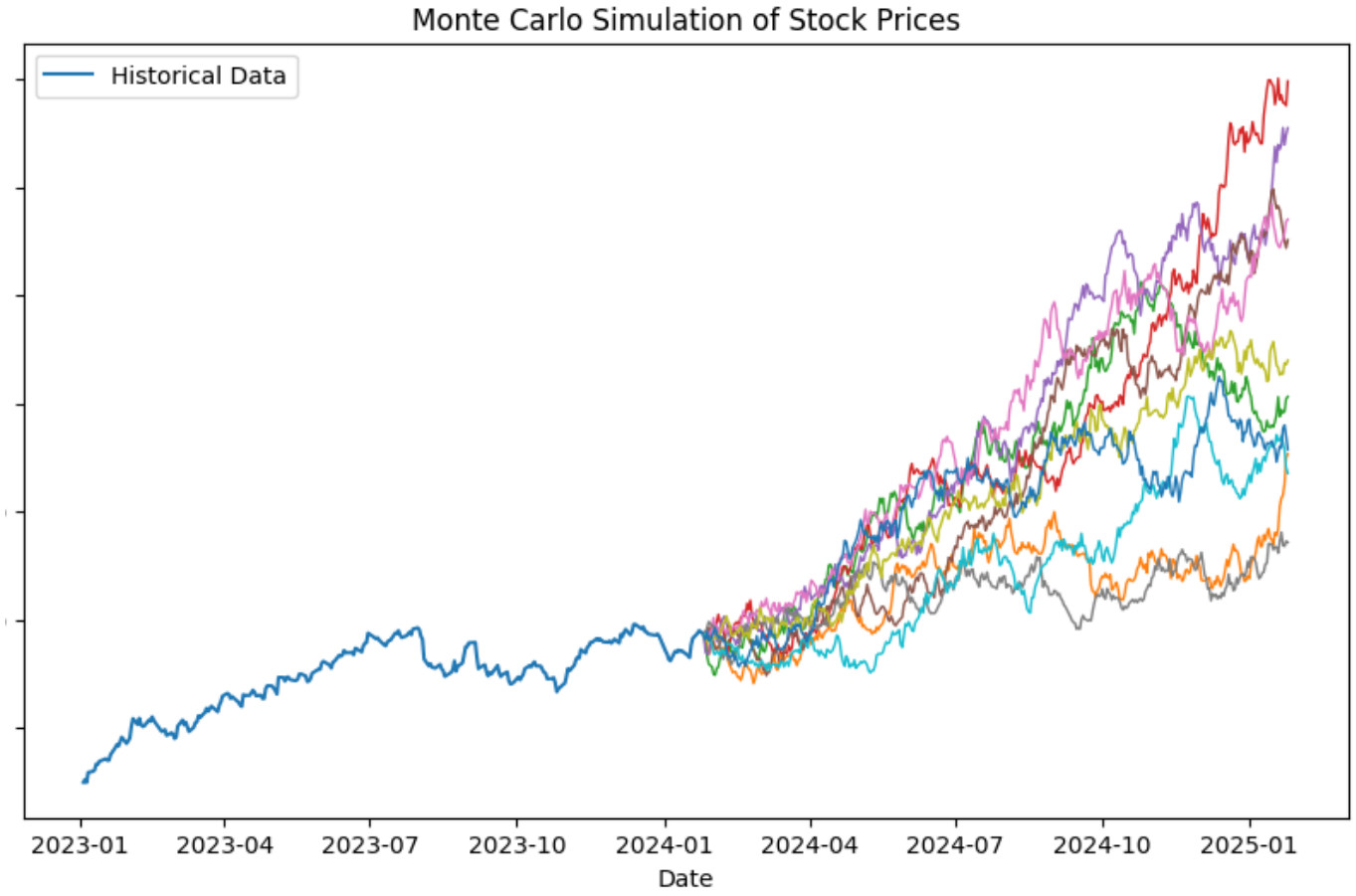
matplotlib

- Interactive
- Popular
- Custom

3. Simulation



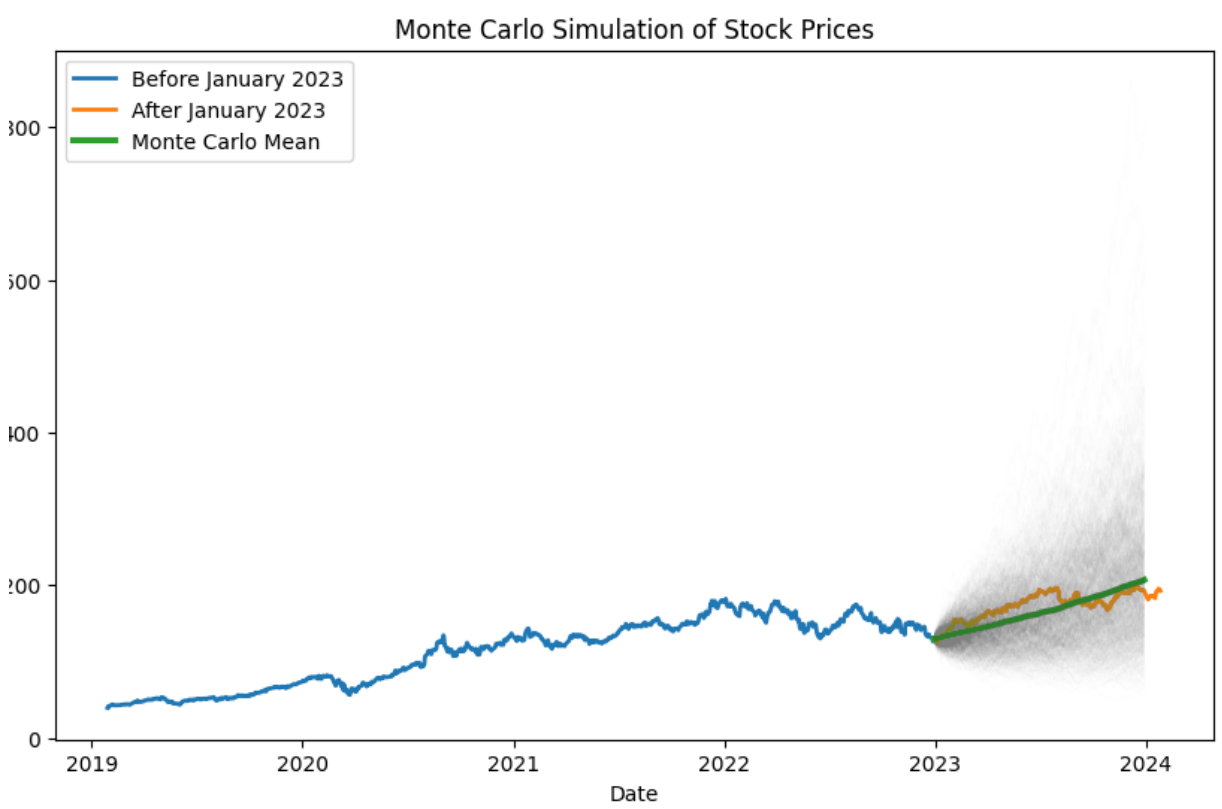


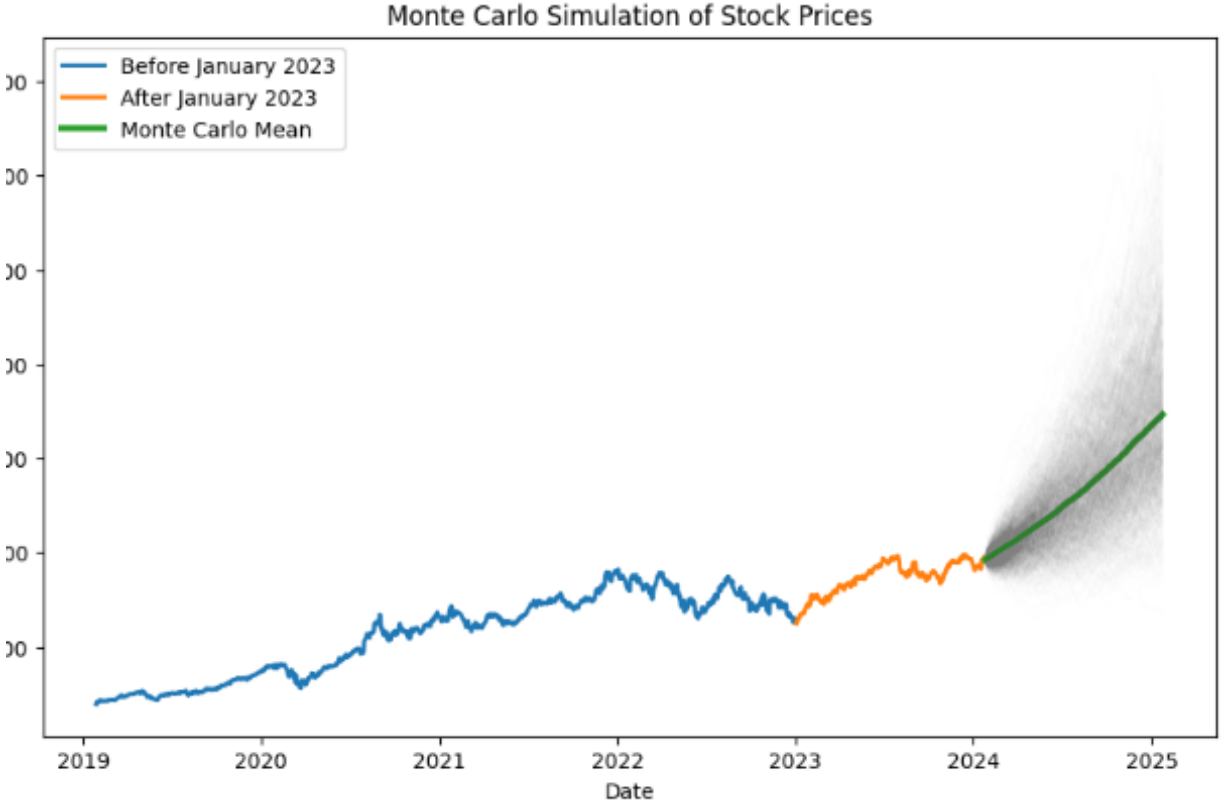


Parameters

- **Moyenne** : Calcul de la moyenne pour estimer le rendement moyen de l'actif.
- **Variance** : Mesure de la variabilité des rendements, indiquant la volatilité.
- **Drift** : Tendance générale des rendements, ajustée pour la volatilité.
- **Chocs aléatoires** : Variations simulées des prix futurs, basées sur la volatilité et l'aléatoire.
- **SimulationDays** : Indique le nombre de jours sur lesquels on fait la simulation, presque une année.
- **MCIterations** : Représente le nombre de fois que la simulation est répétée pour avoir différents résultats possibles.

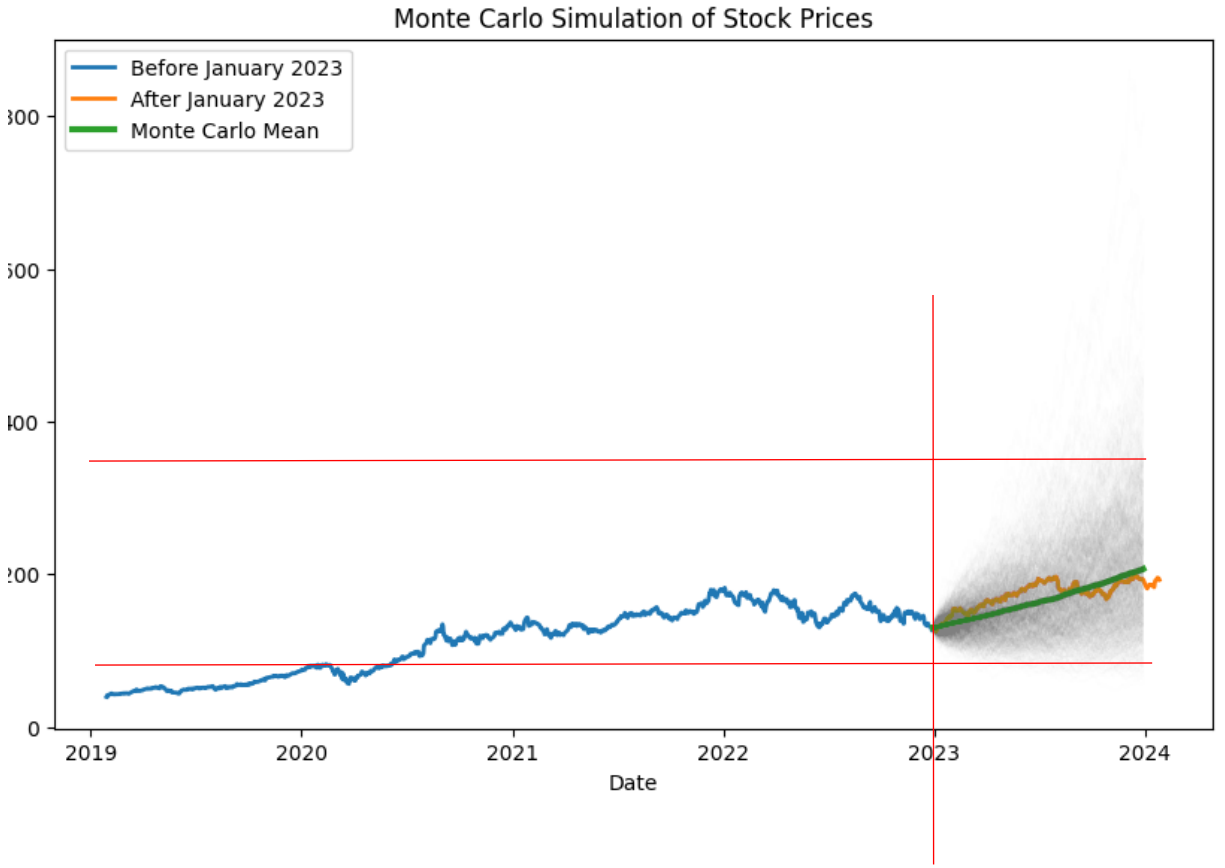
4. Prediction



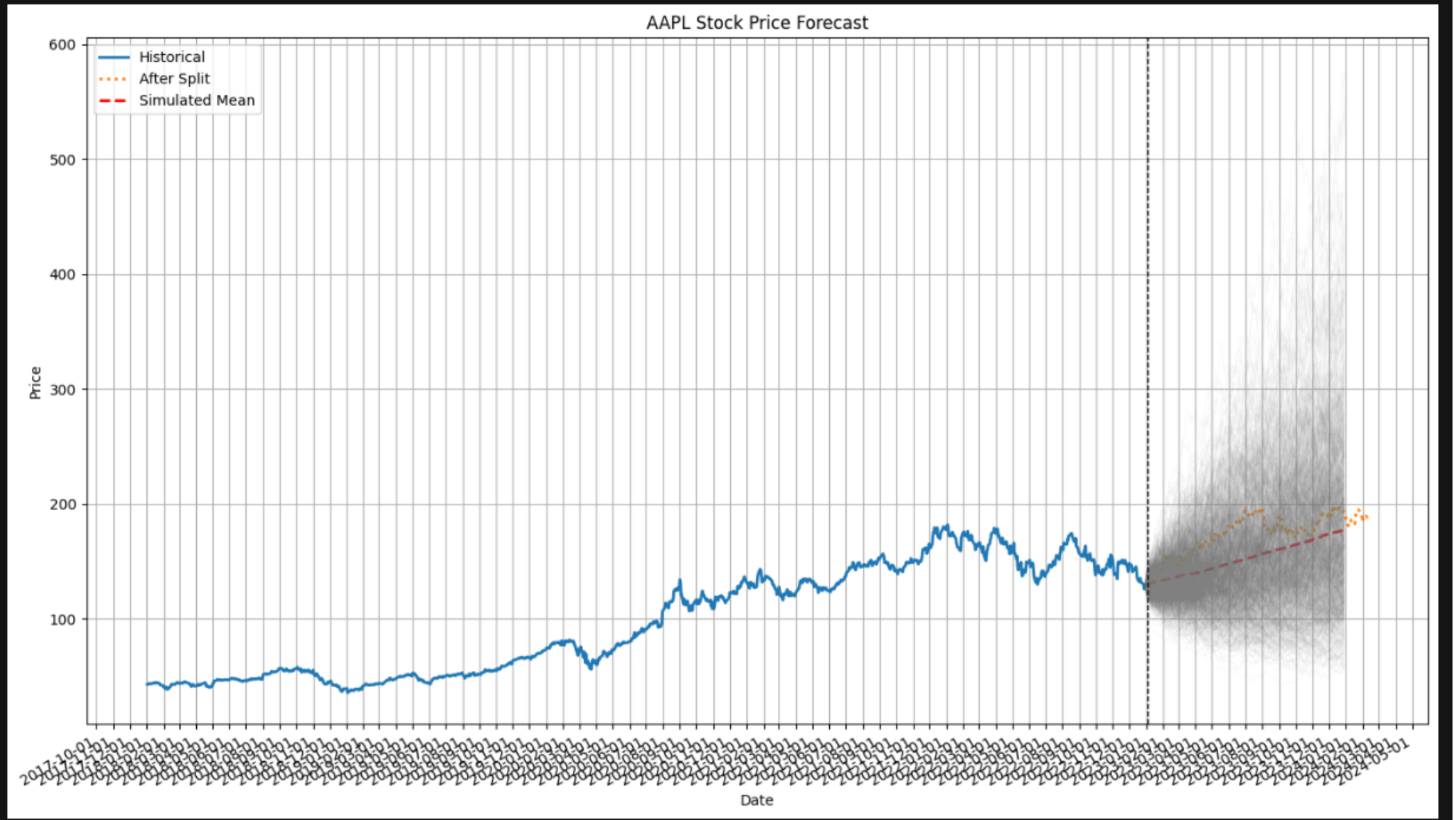


5. Analyse de AAPL

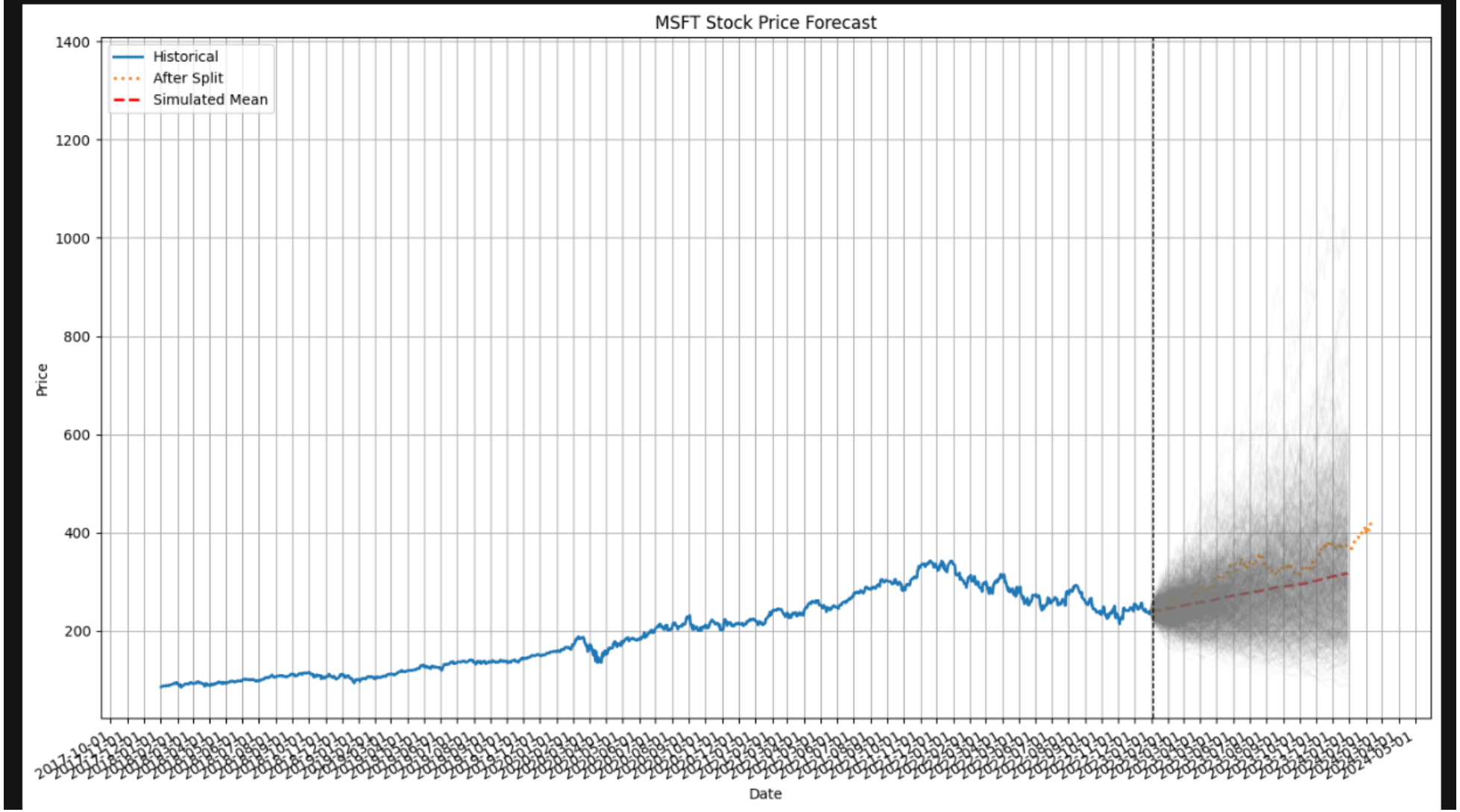
- Average future price after 364 days: $206.63
- Date of simulation: 2023-12-29
- Real price on 2023-12-29: $192.53
- Real date: 2023-12-29
- Price difference: $14.10
- Price difference percentage: 7.32%
- Model accuracy: 92.68%



```
AAPL - Average future price after 364 days: $177.35
Real price on 2024-02-09: $188.85
Price difference: $-11.50, Percentage difference: -6.09%
Model accuracy: 93.91%
```

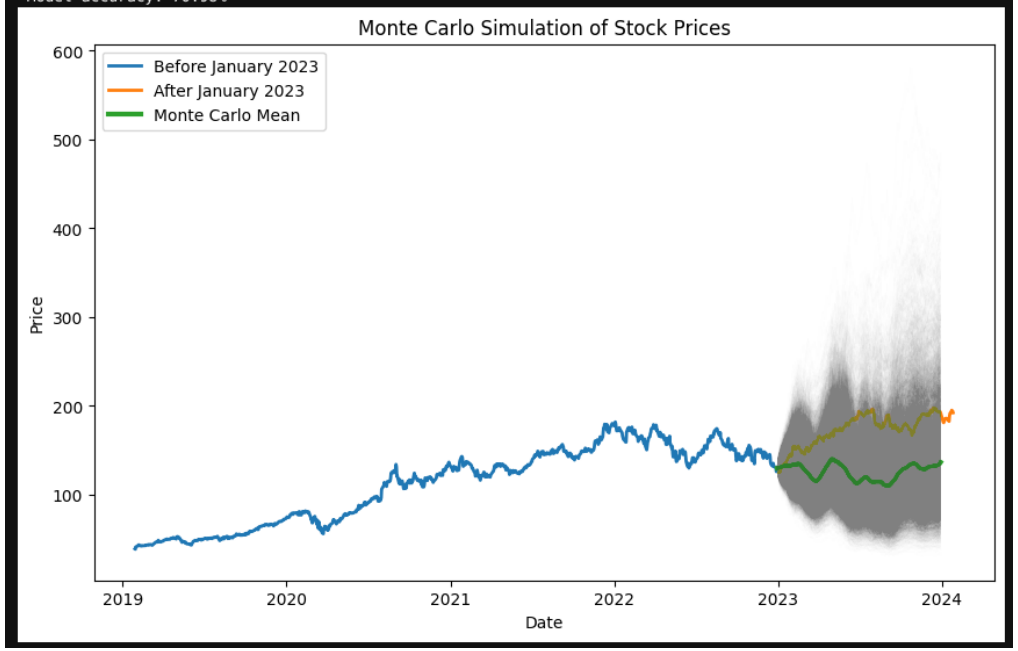


```
MSFT - Average future price after 364 days: $317.63
Real price on 2024-02-09: $420.55
Price difference: $-102.92, Percentage difference: -24.47%
Model accuracy: 75.53%
```

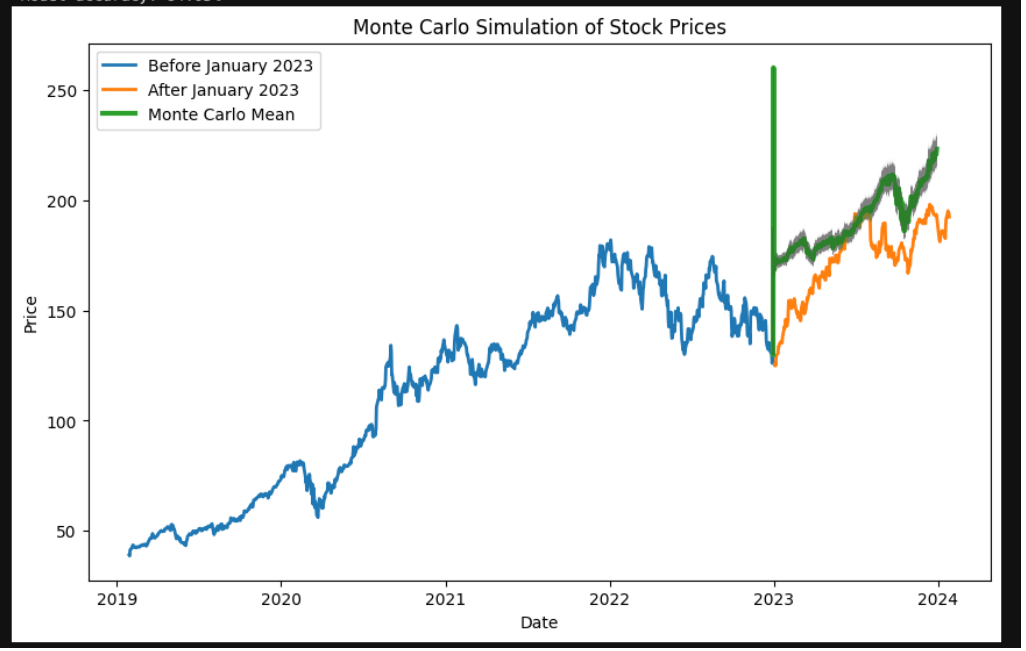


Fail

```
Average future price after 364 days: $136.65
Date of simulation: 2023-12-29
Real price on 2023-12-29: $192.53
Real date: 2023-12-29
Price difference: $-55.88
Price difference percentage: -29.02%
Model accuracy: 70.98%
```



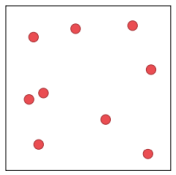

```
Average future price after 364 days: $223.28
Date of simulation: 2023-12-29
Real price on 2023-12-29: $192.53
Real date: 2023-12-29
Price difference: $30.75
Price difference percentage: 15.97%
Model accuracy: 84.03%
```



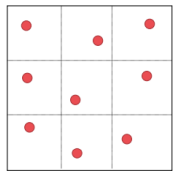
6. Optimization

- Roulette russe: Améliore l'efficacité si nous nous arrêtons sur des échantillons ne nous donnant pas beaucoup de contribution
- Stratification: Meilleur répartition
- Optimiser les paramètre
- Combiner Markov Chain et Monte Carlo
- Loi des grands nombres

Indépendant vs stratifié



```
for i in 0..N:
  s[i].x = rand()
  s[i].y = rand()
```

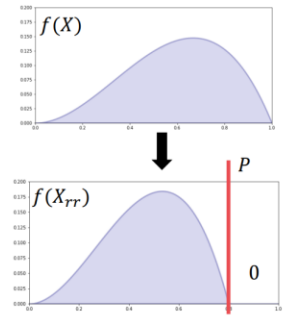


```
for i in 0..Nx:
  for j in 0..Ny:
    s[i,j].x = (i+rand())/Nx
    s[i,j].y = (j+rand())/Ny
```

Roulette Russe

- Arrêter de façon aléatoire la récursion
- Cela nous donne un nouveau estimateur

$$X_{rr} = \begin{cases} \frac{X}{P} & \xi < P \\ 0 & \text{sinon} \end{cases}$$



- **Non biasé**: a la même espérance mathématique

$$E[X_{rr}] = P \cdot E[X]/P + (1-P) \cdot 0 = E[X]$$

https://profs.etsmtl.ca/agruson/competition/2022/

7. Exploration

- Courbe de Risque
- Adjuster le Take profit et Stop lost
- Combiner avec EMA Indicator
- Faire Plus Crypto et Stock
- Combiner avec autres statégies
- Combiner avec la regression lineaire
- Plus gros range de données