텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

16번째 줄의 잘린 부분의 식은 다음과 같이 구현했습니다.

future\_price = self.current\_price \* np.exp((self.expected\_return - 0.5 \* self.volatility\*\*2) \* T + self.volatility \* np.sqrt(T) \* z)

- 멤버 변수는 dividend\_rate(배당률), EPS(당기순이익) 두 개를 추가로 삽입해

총 다섯 개의 멤버 변수를 구현했고,

멤버 함수는 (2), (3) 문제에서 요구하는 함수 두개와 추가한 멤버 변수를 사용해

배당금과 PER을 구하는 함수 두 개를 새로 만들었고,

추가로 샘플 개수 n이 커졌을 때 표본평균과 표본 분산이 각각

이론상 평균과 이론상 분산에 수렴하는지를 확인할 수 있는 그래프를 그리는

함수 두 개를 구현했습니다.

- T 시점의 주가를 구하는 get\_future\_price 함수에서는

size 만큼의 주가를 리턴하고 주가는 각각 다른 정규분포 샘플에 의해서 생성되도록

np.random.seed(seed), z = np.random.standard\_normal(size)를 사용했습니다.

- 표본평균과 표본분산은 각각 numpy 라이브러리의 np.mean과 np.var를 사용해 계산했습니다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

61번째 줄의 잘린 부분의 식은 다음과 같이 구현했습니다.

theoretical\_variance = self.current\_price\*\*2 \* np.exp(2 \* self.expected\_return \* T) \* (np.exp(self.volatility\*\*2 \* T) - 1)

- 표본평균과 이론상 평균을 비교하는 그래프를 그릴 때에는

matplotlib 라이브러리 내의 pyplot 모듈을 사용했습니다.

x축 변수는 n이고, y축에는 계산된 평균값이 나타나도록 설정합니다.

범위는 1부터 n까지로 설정했고, 표본 평균과 이론상 평균을 저장할 빈 리스트를 생성합니다.

for 루프에서는 get\_future\_price 함수를 사용해 주가를 생성한 뒤

계산된 표본 평균과 이론상의 평균이 비어 있던 리스트에 추가되어 저장됩니다.

이후 각 점들을 그래프에서 표현하도록 구현했습니다.

- 위와 같은 방식으로 표본분산과 이론상 분산을 비교하는 그래프를 구현했습니다.

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

위에서 만든 주식 클래스를 구현하기 위한 예시입니다.

모든 멤버 변수를 1회 이상 사용해 모든 멤버 함수를 1회 이상 호출했습니다.

이때 문제 (2)와 (3)에서 각각 get\_future\_price의 input parameter로 T, seed, size를 포함하고, get\_sample\_stats 함수에서는 input parameter로 T, seed, n을 포함하는 조건에 따라,

size와 n이 다른 변수이기 때문에 먼저 size = 5로 설정해서

get\_future\_price 함수를 통해 각각 다른 정규분포 샘플에 따른 3년 후의 주가 5개가 정상적으로 리턴되는 걸 보였고, 이후에 n=1000으로 설정해 뒤에 나오는 함수들에 적용되도록 했습니다.

plot\_sample\_mean를 통해 그래프를 그려보면 n이 100 이하로 작을 때에는 표본평균과 이론상 평균이 수렴하는 걸 보기 힘들지만 n이 1000이상으로 커지면 두 값이 수렴하는 걸 그래프를 통해 확인할 수 있었습니다.

n = 1000인 경우

텍스트, 스크린샷, 그래프, 도표이(가) 표시된 사진

자동 생성된 설명

추가로 n = 100인 경우

텍스트, 스크린샷, 그래프, 도표이(가) 표시된 사진

자동 생성된 설명