probabilidade_lista_1

May 10, 2023

[2]:
```python
from scipy.stats import norm, binom, hypergeom
from tabulate import tabulate
```

0.1 Questão 7

Temos que $S(t) = S0 * u^{N(t)} * d^{(t-N(t))}$, onde

S0 é o valor inicial,

$N(t)$ é o número de vezes que a ação subiu no período t,

u é o valor que a ação sobe (quando sobe) e

d o valor que diminui.

A $P(S(t)) = P(N(t) = n)$ é uma variável aleatória $n \sim Binom(n = t, p = 0.45)$

0.1.1 a) Qual é a probabilidade da ação valer mais que seu valor inicial no instante t = 3.

[3]:
```python
# Define as variáveis u, p_u, d, p_d, t e s0
u = 0.03          # percentual que a acao sobe
p_u = 0.45        # probabilidade de subida
d = 0.02          # percentual que a acao desvaloriza
p_d = 1 - p_u     # probabilidade de desvalorizacao
t = 3             # periodos
s0 = 12           # valor inicial

# Cria uma lista vazia chamada out_table e uma variável p_final inicializada␣
↪com 0
out_table = []
p_final = 0

# Loop que itera pelo número de subidas possíveis (de 0 a t) e calcula o valor␣
↪provável da ação, a probabilidade de esse valor acontecer,
# a distribuição acumulada da probabilidade e adiciona os valores a out_table.
# Se o valor da ação for maior que o valor inicial s0, incrementa p_final com a␣
↪probabilidade de esse valor ocorrer.
for numero_subidas in range(t+1):
```

```
    # Valor provável da ação
    s = s0 * (1+u)**numero_subidas * (1-d)**(t-numero_subidas)

    # Calcula a probabilidade de número_subidas subidas em t períodos e␣
 ↪adiciona à lista
    p_n = binom.pmf(numero_subidas, t, p_u)

    # Calcula a distribuição acumulada da probabilidade e adiciona à lista
    cdf = binom.cdf(numero_subidas, t, p_u)

    # Adiciona os valores a out_table
    out_table.append([numero_subidas, round(s,3), round(p_n,3),  round(cdf,3)])

    # Se o valor da ação for maior que s0, incrementa p_final com a␣
 ↪probabilidade de esse valor acontecer
    if s > s0:
        p_final += p_n

# Imprime a tabela out_table com os cabeçalhos e o valor de p_final com 3 casas␣
 ↪decimais.
print(tabulate(out_table, headers=["Subidas (N(3))","Valor (S(3)=s)", "P(s)",␣
 ↪"Fs(S)"]))
print(f"probabilidade = {round(p_final*100,3)}%")
```

```
  Subidas (N(3))    Valor (S(3)=s)    P(s)    Fs(S)
----------------  ----------------  ------  -------
               0            11.294   0.166    0.166
               1            11.871   0.408    0.575
               2            12.476   0.334    0.909
               3            13.113   0.091    1
probabilidade = 42.525%
```

0.1.2 b) Qual é a probabilidade da ação valer mais que R$ 13.00 no instante t = 15 (se preferir, use a planilha excel para resolver esse item).

[4]:
```
# Definir o número de períodos
t = 15

# Definir o valor inicial da ação
s0 = 12

# Definir o objetivo de preço da ação
objetivo = 13

# Criar uma lista vazia para armazenar as informações de saída
out_table = []
```

```python
# Inicializar a probabilidade final
p_final = 0

# Loop através de cada número de subidas possíveis
for numero_subidas in range(t+1):

    # Calcular o valor provável da ação dado o número de subidas
    s = s0 * (1+u)**numero_subidas * (1-d)**(t-numero_subidas)

    # Calcular a probabilidade de que a ação suba um determinado número de vezes
    p_n = binom.pmf(numero_subidas, t, p_u)

    # Calcular a função de distribuição acumulada (CDF) para a probabilidade
    cdf = binom.cdf(numero_subidas, t, p_u)

    # Adicionar os valores atuais à tabela de saída
    out_table.append([numero_subidas, round(s,4), round(p_n,4),  round(cdf,4)])

    # Se o valor da ação atual for maior do que o objetivo de preço, adicionar␣
 ↪a probabilidade atual à probabilidade final
    if s > objetivo :
        p_final += p_n

# Imprimir a tabela de saída formatada
print(tabulate(out_table, headers=["Subidas (N(15))","Valor (S(15)=s)", "P(s)",␣
 ↪"Fs(S)"]))

# Imprimir a probabilidade final formatada
print(f"probabilidade = {round(p_final*100,3)}%")
```

```
  Subidas (N(15))    Valor (S(15)=s)    P(s)    Fs(S)
-----------------  -----------------  ------  -------
                0             8.8628  0.0001   0.0001
                1             9.315   0.0016   0.0017
                2             9.7903  0.009    0.0107
                3            10.2898  0.0318   0.0424
                4            10.8148  0.078    0.1204
                5            11.3665  0.1404   0.2608
                6            11.9465  0.1914   0.4522
                7            12.556   0.2013   0.6535
                8            13.1966  0.1647   0.8182
                9            13.8699  0.1048   0.9231
               10            14.5775  0.0515   0.9745
               11            15.3213  0.0191   0.9937
               12            16.103   0.0052   0.9989
               13            16.9246  0.001    0.9999
               14            17.7881  0.0001   1
```

```
              15             18.6956   0        1
probabilidade = 34.65%
```

0.2 Questão 8

[5]:
```
# Definindo os parâmetros do problema
x = 1   # número mínimo de peças defeituosas a serem encontradas
N = 50  # tamanho do lote de peças
k = 6   # número de peças defeituosas no lote
n = 5   # número de peças retiradas para inspeção

# Calculando a probabilidade de encontrar x peças defeituosas em n retiradas
probability = hypergeom.pmf(x, N, k, n)

# Imprimindo a probabilidade encontrada
print(f"Probabilidade: {round(probability*100,2)}%")
```

```
Probabilidade: 38.44%
```

0.3 Questão 9

Uma fábrica de carros sabe que os motores de sua fabricação têm duração de acordo com uma distribuição normal

com média $\mu = 150000Km$ e

desvio-padrão $\sigma = 5000km$.

Qual é a probabilidade de que um carro qualquer tenha um motor que dure:

a) Pelo menos 160000 Km?.

[7]:
```
# Definindo os parâmetros da distribuição normal
media = 150000 # média
desvio_padrao = 5000 # desvio-padrão

# Definindo o valor da variável aleatória X para calcular a probabilidade P(X␣
↪>= x)
x = 160000

# Usando a função cdf() da biblioteca norm para calcular a probabilidade P(X <=␣
↪x)
# e subtraindo o resultado de 1 para obter a probabilidade P(X >= x)
prob = 1 - norm.cdf(x, media, desvio_padrao)

# Imprimindo a probabilidade calculada formatada em porcentagem
print(f'P(X   {x}) = 1 - P(X   {x}) = {round(prob*100,2)}%')
```

```
P(X   160000) = 1 - P(X   160000) = 2.28%
```

b) Entre 140000 e 165000 Km?

[8]:
```
media = 150000    # média
desvio_padrao = 5000    # desvio-padrão
x1 = 165000    # valor máximo
x2 = 140000    # valor mínimo
prob = norm.cdf(x1, media, desvio_padrao) - norm.cdf(x2, media, desvio_padrao) ␣
 ↪ # calcula a probabilidade
print(f'P (140000   X   160000) = {round(prob*100,2)}%')    # imprime o resultado␣
 ↪formatado
```

```
P (140000   X   160000) = 97.59%
```

0.3.1 c) Se a fábrica substitui o motor que apresente duração inferior à garantia

0.3.2 qual deve ser a garantia em Km para que a porcentagem de motores substituídos seja inferior à 0.2%?

[9]:
```
# Definindo a média e o desvio-padrão da distribuição normal
media = 150000
desvio_padrao = 5000

# Calculando o z-score correspondente a uma probabilidade de 0.2%
z_score = norm.ppf(0.2/100)

# Calculando a garantia em km correspondente ao z-score encontrado
X = z_score * desvio_padrao + media

# Exibindo os resultados
print(f'''Com Z_score de {round(z_score,4)}, correspondente a uma garantia de␣
 ↪X* ~= {int(X)} km,
a probabilidade de o motor durar pelo menos X* é P(X   X*) = {round(norm.cdf(X,␣
 ↪media, desvio_padrao)*100,5)}%''')
```

```
Com Z_score de -2.8782, correspondente a uma garantia de X* ~= 135609 km,
a probabilidade de o motor durar pelo menos X* é P(X   X*) = 0.2%
```

0.4 Questão 10

Uma pessoa decide investir em um título cambial comprando US$ 200.000,00. A taxa de câmbio atual real/dólar é 2.0. Suponha que o desvio padrão diário da taxa de câmbio R seja 5.0% e que possa ser modelada por uma variável aleatória normal. Qual é a perda máxima em real em 1 dia com 95% de chances?

[10]:
```
investimento_inicial = 200_000

cambio_medio = 2
desvio_padrao = 5/100
z_score = norm.ppf(0.05)
```

```
z_score
```

[10]: -1.6448536269514729

[11]:
```
cambio_perda_maxima = (z_score * desvio_padrao + cambio_medio)
print(f"Mínimo Câmbio: {cambio_perda_maxima}")
valor_cambio_atualizado = cambio_perda_maxima * investimento_inicial
print(f"Valor investimento: {valor_cambio_atualizado}")

lucro = valor_cambio_atualizado - cambio_medio*investimento_inicial
print(f"Perda máxima de {round(lucro,2)} R$")
```

```
Mínimo Câmbio: 1.9177573186524264
Valor investimento: 383551.4637304853
Perda máxima de -16448.54 R$
```