CÓDIGOS

terça-feira, 19 de setembro de 2023 22:30

Distribuição de Probabilidade Binomial:

```
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import binom

# Condições do experimento probabilistico binomial
exp = 7
sucesso = 0.46
val_x = list(range(exp + 1))
# Geração da tabela de distribuição de probabilidade binomial
dist_binomial = pd.DataFrame({'Sucessos': val_x})
dist_binomial['Probabilidade'] = binom.pmf(val_x, exp, sucesso)
display(dist_binomial)
# Representação gráfica da distribuição discreta de probabilidades
# Gráfico de segmentos verticais (caules)
print('** Gráfico de segmentos verticais (caules) **')
dist_stem = plt.stem(dist_binomial['Sucessos'],
                     dist_binomial['Probabilidade'], use_line_collection=True)
plt.xlabel('Indivíduos que usam o celular para acessar redes sociais')
plt.ylabel('Probabilidades')
plt.title('Distribuição discreta de probabilidades')
plt.grid()
plt.show()
# Gráfico de barras verticais
print('** Gráfico de barras verticais **')
dist_bar = plt.bar(dist_binomial['Sucessos'],
                   dist_binomial['Probabilidade'])
plt.xlabel('Indivíduos que usam o celular para acessar redes sociais')
plt.ylabel('Probabilidades')
plt.title('Distribuição discreta de probabilidades')
plt.grid(False)
plt.show()
```

Neste código, você está realizando uma análise probabilística binomial e visualizando a distribuição discreta de probabilidades. Vou explicar cada parte do código:

1. exp = 7` e `sucesso = 0.46`: Aqui, você define os parâmetros do experimento probabilístico binomial. `exp` representa o número de experimentos ou tentativas, enquanto `sucesso` é a probabilidade de sucesso em cada tentativa.

2. val_x = list(range(exp + 1))`: Essa linha cria uma lista `val_x` que contém todos os possíveis valores de sucesso, variando de 0 a `exp`. Isso representa o número de sucessos que você deseja analisar em um experimento binomial.

3. dist_binomial`: Aqui, você cria um DataFrame `dist_binomial` para armazenar a distribuição de probabilidade binomial. A coluna 'Sucessos' conterá os valores de sucesso, e a coluna 'Probabilidade' será preenchida com as probabilidades associadas a cada valor de sucesso. Você calcula essas probabilidades usando a função `binom.pmf` da biblioteca SciPy.

4. display(dist_binomial)`: Isso exibe a tabela de distribuição de probabilidade binomial na saída.

5. Em seguida, você cria dois gráficos para visualizar a distribuição discreta de probabilidades:

- *Gráfico de Segmentos Verticais (Caules)*: Você utiliza `plt.stem` para criar um gráfico de caules, onde os valores de sucesso são exibidos no eixo x e as probabilidades no eixo y. Este gráfico mostra a probabilidade de cada valor de sucesso.

- *Gráfico de Barras Verticais*: Você utiliza `plt.bar` para criar um gráfico de barras verticais, onde

os valores de sucesso são exibidos no eixo x e as probabilidades no eixo y. Cada barra representa a probabilidade de um determinado valor de sucesso.

6. Ambos os gráficos têm rótulos de eixos, título e uma grade para melhorar a legibilidade.

Essencialmente, o código calcula e visualiza a distribuição de probabilidade binomial para um experimento com 7 tentativas e uma probabilidade de sucesso de 0.46 em cada tentativa. Ele fornece duas representações gráficas diferentes dessa distribuição discreta de probabilidades.

Parâmetros populacionais de uma distribuição binomial:

```
# ** Cálculo dos parâmetros usando as fórmulas **
# Cálculo da média (mu = np)
media = exp*sucesso
# Cálculo da variância (sigma^2 = np(1-p))
variancia = exp*sucesso*(1-sucesso)
# Cálculo do desvio padrão
desv_pad = variancia**(1/2)
# Exibição dos resultados
print('** Parâmetros populacionais - Fórmulas **')
print(f'Média = {media:.2f} indivíduos')
print(f'Variância = {variancia:.2f} indivíduos^2')
print(f'Desvio padrão = {desv_pad:.2f} indivíduos')
# Cálculo dos parâmetros usando o método stats de binom
med, var = binom.stats(exp, sucesso)
print('** Parâmetros populacionais - stats **')
print(f'Média = {med:.2f} ; Variância = {var:.2f}')
```

Neste trecho do código, você está calculando os parâmetros populacionais da distribuição binomial, como a média, a variância e o desvio padrão. Você calcula esses parâmetros de duas maneiras diferentes: usando fórmulas e usando o método `stats` da biblioteca SciPy. Vou explicar cada parte deste trecho:

1. *Cálculo da Média, Variância e Desvio Padrão Usando Fórmulas*:
   - media = exp * sucesso`: A média (ou valor esperado) de uma distribuição binomial é calculada multiplicando o número de experimentos (`exp`) pela probabilidade de sucesso (`sucesso`). A fórmula é dada por μ = np, onde μ representa a média, n é o número de experimentos e p é a probabilidade de sucesso.
   - variancia = exp * sucesso * (1 - sucesso)`: A variância de uma distribuição binomial é calculada usando a fórmula $\sigma^2 = np(1-p)$, onde σ^2 representa a variância e n é o número de experimentos. Essa fórmula leva em consideração tanto o número de experimentos quanto a probabilidade de sucesso e fracasso.
   - desv_pad = variancia**(1/2)`: O desvio padrão é a raiz quadrada da variância e é calculado como $\sigma = \sqrt{(\sigma^2)}$.

2. *Exibição dos Resultados*:
   - Aqui, você exibe os valores calculados da média, variância e desvio padrão usando as fórmulas.

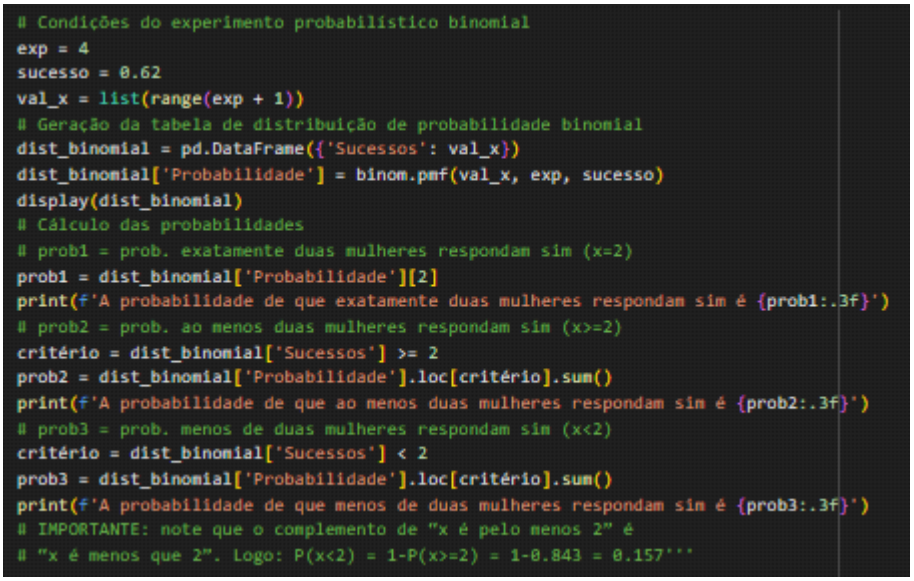
3. *Cálculo dos Parâmetros Usando o Método `stats`*:
   - med, var = binom.stats(exp, sucesso)`: Você usa o método `stats` da distribuição binomial da biblioteca SciPy para calcular a média (`med`) e a variância (`var`). Esse método fornece uma maneira conveniente de calcular esses parâmetros.

4. *Exibição dos Resultados Usando o Método `stats`*:
   - Aqui, você exibe os valores calculados da média e variância usando o método `stats` da biblioteca SciPy.

Ambos os métodos devem fornecer os mesmos resultados para a média e a variância, uma vez que as fórmulas e o método `stats` são equivalentes para esses cálculos em uma distribuição binomial. O

desvio padrão é a raiz quadrada da variância, portanto, também será o mesmo em ambos os casos.

Mais probabilidades binomiais:



```
# Condições do experimento probabilístico binomial
exp = 4
sucesso = 0.62
val_x = list(range(exp + 1))
# Geração da tabela de distribuição de probabilidade binomial
dist_binomial = pd.DataFrame({'Sucessos': val_x})
dist_binomial['Probabilidade'] = binom.pmf(val_x, exp, sucesso)
display(dist_binomial)
# Cálculo das probabilidades
# prob1 = prob. exatamente duas mulheres respondam sim (x=2)
prob1 = dist_binomial['Probabilidade'][2]
print(f'A probabilidade de que exatamente duas mulheres respondam sim é {prob1:.3f}')
# prob2 = prob. ao menos duas mulheres respondam sim (x>=2)
critério = dist_binomial['Sucessos'] >= 2
prob2 = dist_binomial['Probabilidade'].loc[critério].sum()
print(f'A probabilidade de que ao menos duas mulheres respondam sim é {prob2:.3f}')
# prob3 = prob. menos de duas mulheres respondam sim (x<2)
critério = dist_binomial['Sucessos'] < 2
prob3 = dist_binomial['Probabilidade'].loc[critério].sum()
print(f'A probabilidade de que menos de duas mulheres respondam sim é {prob3:.3f}')
# IMPORTANTE: note que o complemento de "x é pelo menos 2" é
# "x é menos que 2". Logo: P(x<2) = 1-P(x>=2) = 1-0.843 = 0.157'''
```

Neste código, você está trabalhando com um experimento probabilístico binomial e calculando diferentes probabilidades relacionadas a esse experimento. Vou explicar cada parte deste trecho:

1. *Condições do Experimento Binomial*:
   - exp = 4`: Você define o número de experimentos (ensaios) como 4. No contexto de um experimento binomial, isso geralmente significa que você está repetindo um processo quatro vezes, cada vez com uma probabilidade de sucesso constante.
   - sucesso = 0.62`: Você define a probabilidade de sucesso em cada experimento como 0.62. Isso indica a probabilidade de um resultado desejado em cada tentativa do experimento.

2. *Geração da Tabela de Distribuição Binomial*:
   - Você cria um DataFrame chamado `dist_binomial` para armazenar a distribuição de probabilidade binomial. Ele terá duas colunas: "Sucessos", que representa o número de sucessos possíveis (0 a 4), e "Probabilidade", que armazena as probabilidades correspondentes calculadas usando a função de massa de probabilidade (`pmf`) da distribuição binomial da biblioteca SciPy.

3. *Cálculo das Probabilidades*:
   - prob1 = dist_binomial['Probabilidade'][2]`: Aqui, você calcula a probabilidade de exatamente duas mulheres responderem "sim" (x=2) no experimento. Isso é simplesmente obtido acessando a probabilidade correspondente no DataFrame `dist_binomial`.
   - prob2 = dist_binomial['Probabilidade'].loc[critério].sum()`: Você calcula a probabilidade de que ao menos duas mulheres respondam "sim" (x>=2) no experimento. Isso é feito somando as probabilidades para x igual ou maior que 2, o que é determinado pelo critério `critério = dist_binomial['Sucessos'] >= 2`.
   - prob3 = dist_binomial['Probabilidade'].loc[critério].sum()`: Aqui, você calcula a probabilidade de menos de duas mulheres responderem "sim" (x<2) no experimento. Isso é feito somando as probabilidades para x menor que 2, conforme determinado pelo critério `critério = dist_binomial['Sucessos'] < 2`.

4. *Complemento das Probabilidades*:
   - Você observa que o complemento de "x é pelo menos 2" é "x é menos que 2". Portanto, você calcula `prob3` como o complemento de `prob2` usando a fórmula: `P(x<2) = 1 - P(x>=2)`. Nesse caso, `prob3` é a probabilidade de que menos de duas mulheres respondam "sim".

Essas probabilidades calculadas fornecem informações sobre diferentes cenários dentro do experimento binomial, como o número exato de sucessos, ao menos um sucesso, e menos de um sucesso. Cada probabilidade é calculada com base na distribuição binomial com os parâmetros fornecidos (`exp` e `sucesso`).