Revisão: Estatística para Ciência de Dados

> **Estatística** é a ciência que se dedica à coleta, análise e interpretação de dados. Ela é uma ferramenta essencial para a tomada de decisões em diversas áreas do conhecimento.

- **Estatística Descritiva**
  - Descrever e resumir os dados
  - Tabelas, gráficos e medidas resumo
  - **Exemplo**: média, mediana, moda, variância, desvio padrão, etc.
- **Estatística Inferencial**
  - Inferir conclusões sobre uma população a partir de uma amostra
  - Testes de hipóteses, intervalos de confiança, regressão, etc.
  - **Exemplo**: teste t de Student, ANOVA, regressão linear, etc.

Estatística Descritiva

Medidas de Posição

- **Média**: Balanço entre todos os valores. Busca o valor central.
  - Aritmética: $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$
  - Geométrica: $\bar{x} = \sqrt[n]{\prod_{i=1}^{n} x_i}$
  - Harmônica: $\bar{h} = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}$

```
import numpy as np
x = np.array([1, 2, 3, 4, 5])
a = np.mean(x)
g = np.prod(x)**(1/len(x))
h = len(x) / np.sum(1/x)
```

Aplicações:

- Média Aritmética: Média de salários, média de notas, média de preços, etc.
- Média Geométrica: Dados que estão normalizados por um valor de referência. Ex.: taxa de crescimento, taxa de juros, etc.
- Média Harmônica: Dados que envolvem razões ou taxas. Ex.: velocidade média, média de resistências, etc.

- **Mediana**: Valor central de um conjunto de dados ordenados.
  - Se n é ímpar, a mediana é o valor central.
  - Se n é par, a mediana é a média dos dois valores centrais.

```
import numpy as np
x = np.array([15, 22, 35, 4, 85])
mediana = np.median(x)
```

- Em comparação com a média, a mediana é menos sensível a valores extremos.
- Mediana é mais representativa em distribuições assimétricas.
- Mediana é demanda mais tempo computacional para calcular.

- **Moda**: Valor mais frequente em um conjunto de dados.

```
import numpy as np
from scipy import stats
x = np.array([15, 22, 35, 4, 85, 22])
moda = stats.mode(x)
```

- O conjunto de dados pode ser:
  - unimodal: uma moda
  - bimodal: duas modas
  - multimodal: mais de uma moda
  - amodal: sem moda

Aplicações:

- Conjuntos de dados com valores discretos
- Conjuntos de dados categóricos

Medidas de Dispersão

- **Variância**: Média dos quadrados das diferenças entre os valores e a média.
  - População: $\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}$
  - Amostra: $s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$

```
import numpy as np
x = np.array([15, 22, 35, 4, 85])
variancia = np.var(x, ddof=1)
```

- **Desvio Padrão**: Raiz quadrada da variância.
  - População: $\sigma = \sqrt{\sigma^2}$
  - Amostra: $s = \sqrt{s^2}$

```
import numpy as np
x = np.array([15, 22, 35, 4, 85])
desvio_padrao = np.std(x, ddof=1)
```

> Em média, quanto os valores se desviam da média.

- **Coeficiente de Variação**: Medida de dispersão relativa.
  - $CV = \frac{s}{\bar{x}} \times 100\%$

```python
import numpy as np
x = np.array([15, 22, 35, 4, 85])
cv = np.std(x, ddof=1) / np.mean(x) * 100
```

> Em média, quanto os valores se desviam **percentualmente** da média.

- Variância e desvio padrão medem a **dispersão** dos dados.
- O desvio padrão é a medida de dispersão mais comum.
- A variância é menos intuitiva, pois está em unidades ao quadrado.
- Coeficiente de variação é útil para comparar a dispersão entre conjuntos de dados com escalas diferentes.

Medidas de Forma

- **Assimetria**: Medida de desvio da simetria em relação à média.
  - Positiva: cauda à direita
  - Negativa: cauda à esquerda

```
import numpy as np
from scipy import stats
x = np.array([15, 22, 35, 4, 85])
assimetria = stats.skew(x)
```

> skew > 0: assimetria positiva; skew < 0: assimetria negativa
>
> skew = 0: simetria

https://blog.proffernandamaciel.com.br/assimetria-e-curtose-dos-dados/

- **Curtose**: Medida de achatamento da distribuição.
  - Leptocúrtica: mais alta e fina
  - Mesocúrtica: normal
  - Platicúrtica: mais baixa e larga

```
import numpy as np
from scipy import stats
x = np.array([15, 22, 35, 4, 85])
curtose = stats.kurtosis(x)
```

- Curtose é uma medida de forma que indica a forma da distribuição dos dados.
  - Curtose > 0: mais picos
  - Curtose < 0: menos picos
  - Curtose = 0: normal

Medidas de Associação

- **Covariância**: Medida de associação linear entre duas variáveis.
  - População: $\sigma_{xy} = \frac{\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)}{n}$
  - Amostra: $s_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n-1}$

```
import numpy as np
x = np.array([15, 22, 35, 4, 85])
y = np.array([10, 20, 30, 40, 50])
covariancia = np.cov(x, y, ddof=1)
```

- Coeficiente de correlação de Pearson: $r = \frac{s_{xy}}{s_x \times s_y}$

```
import numpy as np
x = np.array([15, 22, 35, 4, 85])
y = np.array([10, 20, 30, 40, 50])
correlacao = np.corrcoef(x, y)
```

- Correlação varia entre -1 e 1.
  - Correlação = 1: associação positiva perfeita
  - Correlação = -1: associação negativa perfeita
  - Correlação = 0: não há associação linear
  - Correlação entre 0.7 e 1: associação forte
  - Correlação entre 0.3 e 0.7: associação moderada
  - Correlação entre 0 e 0.3: associação fraca
  - Correlação entre -0.3 e 0: associação fraca
  - Correlação entre -0.7 e -0.3: associação moderada
  - Correlação entre -1 e -0.7: associação forte

Medidas de Probabilidade

- **Probabilidade**: Medida de incerteza associada a um evento.
  - Probabilidade de Laplace: $P(A) = \frac{\text{número de casos favoráveis}}{\text{número de casos possíveis}}$

```
import numpy as np
from scipy import stats
x = np.array([15, 22, 35, 4, 85]) # Casos possíveis
A = lambda x: x%2 == 0 # Evento: número par
p = np.sum(A(x)) / len(x) # Probabilidade de A em x
```

Calculando a probabilidade de uma variável aleatória discreta:

```
import numpy as np
x = np.array([2,2,3,3,2,2,15, 22, 35, 4, 85]) # Variável aleatória
vals, freq = np.unique(x, return_counts=True)
prob = freq / len(x)
p = {v: p for v, p in zip(vals, prob)}
print(p[2]) # Probabilidade de x = 2
```

- **Distribuição de Probabilidade**: Função que descreve a probabilidade de cada valor de uma variável aleatória baseada em um modelo.
  - discreta: as variáveis podem assumir apenas valores discretos.
  - contínua: as variáveis podem assumir qualquer valor real em um intervalo.

- **Distribuição Normal**: Distribuição de probabilidade contínua mais comum.
  - Média: μ
  - Desvio padrão: σ
  - Função de densidade de probabilidade: $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

```python
import numpy as np
from scipy import stats
mu, sigma = 0, 1 # Média e desvio padrão
p = stats.norm.pdf(0.5, mu, sigma) # Probabilidade de x = 0.5
print(p)
```

> Quando assumimos que os dados seguem uma distribuição normal, podemos inferir muitas informações sobre os dados apenas conhecendo a média e o desvio padrão.

- Teste de normalidade: Teste de hipóteses para verificar se os dados seguem uma distribuição normal.

```
import numpy as np
from scipy import stats
x = np.array([15, 22, 35, 4, 85])
p = stats.normaltest(x)
```

- Se o valor-p é menor que 0.05, os dados não seguem uma distribuição normal.
- Se o valor-p é maior que 0.05, os dados seguem uma distribuição normal.

- **Distribuição Binomial**: Distribuição de probabilidade discreta.
  - Número de tentativas: n
  - Probabilidade de sucesso: p
  - Probabilidade de fracasso: $1 - p$
  - Função de probabilidade: $P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$

```python
import numpy as np
from scipy import stats
n, p = 10, 0.5 # Número de tentativas e probabilidade de sucesso
p = stats.binom.pmf(5, n, p) # Probabilidade de k = 5
print(p)
```

- **Distribuição de Poisson**: Distribuição de probabilidade discreta.
  - Número médio de ocorrências: λ
  - Função de probabilidade: $P(X = k) = \frac{e^{-\lambda}\lambda^k}{k!}$

```python
import numpy as np
from scipy import stats
lamb = 2 # Número médio de ocorrências
p = stats.poisson.pmf(3, lamb) # Probabilidade de k = 3
print(p)
```

- **Distribuição Exponencial**: Distribuição de probabilidade contínua.
  - Taxa de ocorrências: λ
  - Função de densidade de probabilidade: $f(x) = \lambda e^{-\lambda x}$

```
import numpy as np
from scipy import stats
lamb = 2 # Taxa de ocorrências
p = stats.expon.pdf(3, scale=1/lamb) # Probabilidade de x = 3
print(p)
```

- **Distribuição Qui-Quadrado**: Distribuição de probabilidade contínua.
  - Graus de liberdade: k
  - Função de densidade de probabilidade: $f(x) = \frac{1}{2^{k/2}\Gamma(k/2)} x^{k/2-1} e^{-x/2}$

```
import numpy as np
from scipy import stats
k = 2 # Graus de liberdade
p = stats.chi2.pdf(3, k) # Probabilidade de x = 3
print(p)
```

- **Distribuição t de Student**: Distribuição de probabilidade contínua.
  - Graus de liberdade: k
  - Função de densidade de probabilidade: $f(x) = \frac{\Gamma((k+1)/2)}{\sqrt{k\pi}\Gamma(k/2)}(1 + x^2/k)^{-(k+1)/2}$

```python
import numpy as np
from scipy import stats
k = 2 # Graus de liberdade
p = stats.t.pdf(3, k) # Probabilidade de x = 3
print(p)
```

- **Distribuição F de Snedecor**: Distribuição de probabilidade contínua.
  - Graus de liberdade: k_1, k_2
  - Função de densidade de probabilidade: $f(x) = \frac{\sqrt{\frac{(k_1 x)^{k_1} k_2^{k_2}}{(k_1 x + k_2)^{k_1 + k_2}}}}{x B(k_1/2, k_2/2)}$

```
import numpy as np

from scipy import stats
k1, k2 = 2, 3 # Graus de liberdade
p = stats.f.pdf(3, k1, k2) # Probabilidade de x = 3
print(p)
```

Testes de Hipóteses

- **Teste t de Student**: Teste de hipóteses para médias de duas amostras.
  - Hipótese nula: as médias são iguais.
  - Hipótese alternativa: as médias são diferentes.

```
import numpy as np
from scipy import stats
x = np.array([15, 22, 35, 4, 85])
y = np.array([10, 20, 30, 40, 50])
t, p = stats.ttest_ind(x, y)
```

Percentis e Quartis

- **Percentil**: Valor da variável maior que uma certa porcentagem dos dados.

```
import numpy as np
x = np.array([15, 22, 35, 4, 85])
v = np.percentile(x, 32)
print(v)
```

> `v` é o valor que é maior que 32% dos dados.

- **Quartil**: Valores de variáveis que dividem os dados em quatro partes iguais.

```
import numpy as np
x = np.array([15, 22, 35, 4, 85])
q1, q2, q3 = np.percentile(x, [25, 50, 75])
```

> `q1`, `q2` e `q3` são os valores que dividem os dados em quatro partes iguais.

Normalização de Dados

- **Padronização**: Transformação dos dados para ter média zero e desvio padrão um.
  - $z = \frac{x - \bar{x}}{s}$

```
import numpy as np
x = np.array([15, 22, 35, 4, 85])
z = (x - np.mean(x)) / np.std(x)
```

- Padronização é útil para comparar variáveis com diferentes escalas.
- Padronização é útil para algoritmos de aprendizado de máquina sensíveis à escala.
- Padronização não remove outliers.

Outliers

- **Outliers**: Valores extremos que se desviam significativamente do restante dos dados.
    - Critério de Tukey: $Q_1 - 1.5 \times IQR$ e $Q_3 + 1.5 \times IQR$

```
import numpy as np
x = np.array([15, 22, 35, 4, 85])
q1, q3 = np.percentile(x, [25, 75])
iqr = q3 - q1
outliers = x[(x < q1 - 1.5*iqr) | (x > q3 + 1.5*iqr)]
```

- **Tratamento de Outliers**:
  - Remoção: remover os outliers do conjunto de dados.

```
import numpy as np
x = np.array([15, 22, 35, 4, 85])
q1, q3 = np.percentile(x, [25, 75])
iqr = q3 - q1
xa = x[(x >= q1 - 1.5*iqr) & (x <= q3 + 1.5*iqr)]
```

- Substituição: substituir os outliers por valores mais próximos.

```
xb = x.copy()
xb[(xb < q1 - 1.5*iqr)] = q1 - 1.5*iqr
xb[(xb > q3 + 1.5*iqr)] = q3 + 1.5*iqr
```

- Ignorar: ignorar os outliers e continuar a análise.
- Análise separada: analisar os outliers separadamente.

Estatística Inferencial

- **Intervalo de Confiança**: Intervalo que contém o valor real do parâmetro com uma certa probabilidade.
  - Intervalo de confiança para a média: $\bar{x} \pm t_{\alpha/2} \times \frac{s}{\sqrt{n}}$

```
import numpy as np
from scipy import stats
x = np.array([15, 22, 35, 4, 85])
intervalo = stats.t.interval(0.95, len(x)-1, loc=np.mean(x), scale=stats.sem(x))
```

- **Teste de Hipóteses**: Teste estatístico para verificar se uma afirmação sobre uma população é verdadeira.
  - Hipótese nula: afirmação a ser testada.
  - Hipótese alternativa: afirmação oposta à hipótese nula.
  - Valor-p: probabilidade de obter um resultado igual ou mais extremo que o observado, assumindo que a hipótese nula é verdadeira.

```
import numpy as np
from scipy import stats
x = np.array([15, 22, 35, 4, 85])
t, p = stats.ttest_1samp(x, 0)
```

- **ANOVA**: Análise de variância para comparar médias de três ou mais amostras.
  - Hipótese nula: as médias são iguais.
  - Hipótese alternativa: pelo menos uma média é diferente.

```
import numpy as np
from scipy import stats
x = np.array([15, 22, 35, 4, 85])
y = np.array([10, 20, 30, 40, 50])
z = np.array([5, 10, 15, 20, 25])
f, p = stats.f_oneway(x, y, z)
```

- **Regressão Linear**: Modelo estatístico para prever o valor de uma variável a partir de outra.
  - Coeficiente de regressão: $b = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$
  - Intercepto: $a = \bar{y} - b\bar{x}$

```
import numpy as np
from scipy import stats
x = np.array([15, 22, 35, 4, 85])
y = np.array([10, 20, 30, 40, 50])
b, a, r, p, std_err = stats.linregress(x, y)

f = lambda x: a + b*x # Função de regressão
print(f(10)) # Previsão de y para x = 10
```

- **Regressão Logística**: Modelo estatístico para prever a probabilidade de um evento binário.
  - Função logística: $p = \frac{1}{1+e^{-(a+bx)}}$

```
import numpy as np
from scipy import stats
x = np.array([15, 22, 35, 4, 85])
y = np.array([0, 1, 1, 0, 1])
b, a = stats.linregress(x, np.log(y/(1-y)))[:2]
f = lambda x: 1 / (1 + np.exp(-(a + b*x))) # Função de regressão logística
print(f(10)) # Previsão de y para x = 10
```

- **Regressão Polinomial**: Modelo estatístico para prever o valor de uma variável a partir de outra.
  - Coeficientes de regressão: $b_0, b_1, b_2, \ldots, b_n$
  - Função de regressão: $f(x) = b_0 + b_1 x + b_2 x^2 + \ldots + b_n x^n$

```
import numpy as np
from scipy import stats
x = np.array([15, 22, 35, 4, 85])
y = np.array([10, 20, 30, 40, 50])
b = np.polyfit(x, y, 2) # Coeficientes de regressão polinomial para grau 2
f = np.poly1d(b) # Função de regressão polinomial
print(f(10)) # Previsão de y para x = 10
```

- **Regressão Múltipla**: Modelo estatístico para prever o valor de uma variável a partir de duas ou mais variáveis.
  - Coeficientes de regressão: $b_0, b_1, b_2, \ldots, b_n$
  - Função de regressão: $f(x_1, x_2, \ldots, x_n) = b_0 + b_1x_1 + b_2x_2 + \ldots + b_nx_n$

```python
import numpy as np
from scipy import stats
x1 = np.array([15, 22, 35, 4, 85])
x2 = np.array([10, 20, 30, 40, 50])
y = np.array([5, 10, 15, 20, 25])
X = np.column_stack((x1, x2))
b = np.linalg.lstsq(X, y, rcond=None)[0] # Coeficientes de regressão múltipla
f = lambda x1, x2: b[0] + b[1]*x1 + b[2]*x2 # Função de regressão múltipla
print(f(10, 10)) # Previsão de y para x1 = 10 e x2 = 10
```

- **Regressão Não Linear**: Modelo estatístico para prever o valor de uma variável a partir de outra.
    - Função de regressão: $f(x) = ae^{bx}$

```
import numpy as np
from scipy import stats
x = np.array([15, 22, 35, 4, 85])
y = np.array([10, 20, 30, 40, 50])
a, b = stats.linregress(x, np.log(y))[:2]
f = lambda x: a * np.exp(b*x) # Função de regressão não linear
print(f(10)) # Previsão de y para x = 10
```

- **Bootstrap**: Método de reamostragem para estimar a distribuição de uma estatística.
  - Amostras de bootstrap: amostras de uma população com reposição.
  - Estimativa bootstrap: estatística calculada para cada amostra de bootstrap.

```
import numpy as np
from scipy import stats
x = np.array([15, 22, 35, 4, 85])
estimativas = [np.mean(np.random.choice(x, len(x))) for _ in range(1000)]
```

- **Cross-Validation**: Método de avaliação de modelos de aprendizado de máquina.
  - Conjunto de treinamento: conjunto de dados para treinar o modelo.
  - Conjunto de teste: conjunto de dados para avaliar o modelo.
  - Validação cruzada: técnica para avaliar o desempenho do modelo.

```
import numpy as np
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression
x = np.array([15, 22, 35, 4, 85])
y = np.array([10, 20, 30, 40, 50])
model = LinearRegression()
scores = cross_val_score(model, x.reshape(-1, 1), y, cv=5)
```

- **Regularização**: Técnica para evitar overfitting em modelos de aprendizado de máquina.
  - Regressão Ridge: penaliza os coeficientes de regressão.
  - Regressão Lasso: penaliza os coeficientes de regressão e seleciona variáveis.

```
import numpy as np
from sklearn.linear_model import Ridge, Lasso
x = np.array([15, 22, 35, 4, 85])
y = np.array([10, 20, 30, 40, 50])
model = Ridge(alpha=0.1) # Regressão Ridge
model = Lasso(alpha=0.1) # Regressão Lasso
model.fit(x.reshape(-1, 1), y)
```