Aula 13 - Estatística e Probabilidade - Python

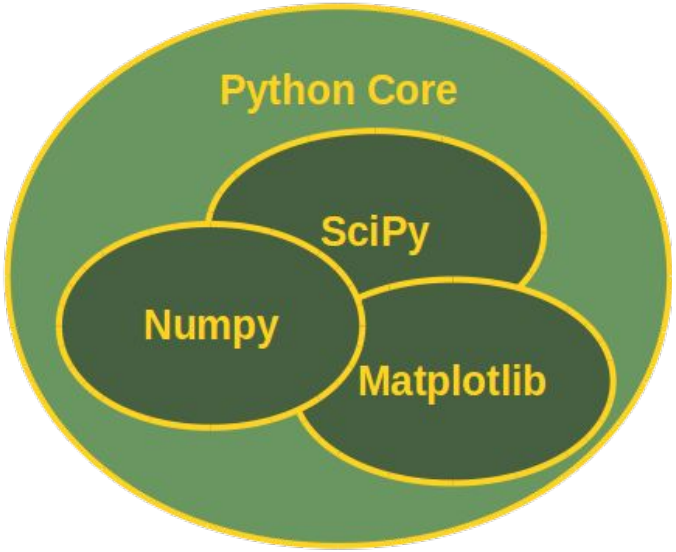
Profa. Gabrielly Queiroz

Numpy e Scipy

NumPy (Numerical Python) é uma biblioteca de código aberto para Python que suporta operações matemáticas e computação científica.

Scipy - computação científica.

Fundamentais para análise de dados e machine learning.



import numpy as np

from scipy import stats

https://numpy.org/

https://docs.scipy.org/doc/scipy/tutorial/stats.html

Scipy

O SciPy (Scientific Python) é outra biblioteca importante no ecossistema da IA e Ciência de Dados. Enquanto o NumPy foca em arrays e álgebra linear, o SciPy expande isso com módulos avançados de:

- Estatística
- Otimização
- Integração
- Álgebra linear
- Interpolação
- Transformadas
- Probabilidade (usado bastante em ML)

Em Inteligência Artificial, o módulo mais usado do SciPy é `scipy.stats`, que oferece ferramentas para distribuições de probabilidade, testes estatísticos, análise de hipóteses, entre outros.

Numpy

np.array() - **Cria vetores e matrizes (arrays)**, que são a base para cálculos matemáticos em IA.

```
import numpy as np

x = np.array([1, 2, 3])         # vetor 1D

m = np.array([[1, 2], [3, 4]])  # matriz 2D
```

https://colab.research.google.com/drive/1wOGA1pQVdI104q7q5OJN1-rRxtlHZyfc?usp=sharing

Numpy

`np.mean()`, `np.median()`, `np.std()`, `np.var()` - **Estatística descritiva**, usada para explorar o desempenho de modelos e dados de entrada.

media = np.mean([10, 20, 30])

mediana = np.median([1, 2, 3, 4, 5])

variância = np.var([10, 12, 14])

desvio = np.std([10, 12, 14])

Numpy

np.dot() - **Produto escalar de vetores ou multiplicação de matrizes**, usado em redes neurais e regressão linear

```
w = np.array([0.2, 0.5])

x = np.array([1, 2])

resultado = np.dot(w, x)  # 0.2*1 + 0.5*2 = 1.2
```

Numpy

np.random - **Geração de dados aleatórios**, ideal para criar datasets simulados para treinar modelos.

`np.random.rand()` – números aleatórios entre 0 e 1

`np.random.randint()` – inteiros aleatórios

`np.random.normal()` – dados com distribuição normal

```
# Geração de vetores aleatórios
x = np.random.rand(5)
y = np.random.normal(loc=0, scale=1, size=1000)
```

Numpy

np.where() - Criação de decisões condicionais – semelhante a um if vetorial, muito útil para rótulos em classificação.

```
valores = np.array([0.7, 0.85, 0.95])

classificacao = np.where(valores > 0.9, 'ótimo', 'regular')

print(classificacao)
```

Numpy

`np.argmax()` e `np.argmin()` - **Índice do maior ou menor valor**, útil para identificar a classe com maior probabilidade em modelos de classificação.

```
probs = np.array([0.1, 0.3, 0.6])

classe_predita = np.argmax(probs)  # retorna 2
```

Numpy

`np.linspace()` e `np.arange()` – **Criação de sequências numéricas**, comuns em visualização, testes ou algoritmos genéticos.

valores = np.linspace(0, 1, 5) # [0. , 0.25, 0.5, 0.75, 1.]

`np.sum()`, `np.max()`, `np.min()` – **Operações agregadas** essenciais para cálculo de perdas, métricas e análise de desempenho.

dados = np.array([[1, 2], [3, 4]])

print(np.sum(dados)) # soma total: 10

print(np.max(dados)) # maior valor: 4

Scipy

Módulo	Função	Aplicação
`scipy.stats`	`norm`, `binom`, `poisson`	Distribuições probabilísticas
`scipy.stats`	`mode`, `ttest_ind`	Estatísticas e testes
`scipy.optimize`	`minimize`	Minimização de funções
`scipy.linalg`	`inv`, `eig`, `solve`	Álgebra linear
`scipy.integrate`	`quad`, `dblquad`	Integração numérica
`scipy.cluster`	`kmeans`, `vq`	Agrupamento de dados
`scipy.spatial`	`distance.euclidean`	Distância vetorial

Scipy

`scipy.stats.mode()` – Moda - Determinar o valor mais frequente (usado para rótulos, predição mais comum).

```
from scipy import stats


valores = [1, 2, 2, 3, 3, 3, 4]

moda = stats.mode(valores)

print(moda.mode[0], moda.count[0])
```

Distribuição Normal

norm.cdf(x, loc=media, scale=desvio_padrao)

x é Valor que você quer saber a probabilidade acumulada até ele. Ex: `altura < x`

from scipy.stats import norm

prob = norm.cdf(valor, loc=media, scale=desvio_padrao)

```
from scipy.stats import norm

media = 0.90
desvio = 0.02

# Qual a probabilidade de um modelo ter precisão menor que 0.88?
prob = norm.cdf(0.88, loc=media, scale=desvio)
print(f"Probabilidade de precisão < 0.88: {prob:.4f}")
```

Distribuição Binominal

```
n   # número de transações
p    # probabilidade de acerto
k    # número de acertos desejados

prob_binomial = binom.pmf(k, n, p)
```

```
from scipy.stats import binom

# Probabilidade de acertar exatamente 7 de 10 previsões com 80% de chance de acerto

prob = binom.pmf(k=7, n=10, p=0.8)

print(f"P(X = 7): {prob:.4f}")
```

Distribuição de Poisson

```
mu      # taxa média de falhas
k       # número de falhas que queremos

prob_poisson = poisson.pmf(k, mu)
```

```
from scipy.stats import poisson
```

```
# Qual a chance de ocorrerem exatamente 3 erros, sabendo que a média é 2 por hora?

prob = poisson.pmf(k=3, mu=2)

print(f"P(X = 3): {prob:.4f}")
```

Atividade

Faça o download do seguinte conjunto de dados em formato `.csv`:
https://people.sc.fsu.edu/~jburkardt/data/csv/hw_200.csv

Esse arquivo contém informações sobre **altura (em polegadas)** e **peso (em libras)** de 200 indivíduos. Com base nesses dados, utilize Python (com Pandas, NumPy e SciPy) para realizar uma análise estatística.

Calcule a média e o desvio padrão das colunas `Altura e peso`. Em seguida, **utilize a função `scipy.stats.norm.cdf()` para calcular a probabilidade de um indivíduo ter altura menor que 65 polegadas e peso menor que 120 libras**, assumindo que os dados seguem uma distribuição normal. Crie uma classificação entre altura baixa, média e alta.