GEIA 25.1- Introdução à Ciência de Dados

Aluno:

Leandro Quintela de Araujo (**2514910**)

Monitor:

Arthur Veras

Fortaleza

2025

Sumário

```python
class ImportarDados:
    def __init__(self, caminho_arquivo):
        self.array = np.genfromtxt(caminho_arquivo, delimiter=",", skip_header=1)
        self.idade = self.array[:, 1]
        self.mat = self.array[:, 2]
        self.est = self.array[:, 3]
        self.prog = self.array[:, 4]
        self.faltas = self.array[:, 5]

        self.variaveis = {
            'Idade': self.idade,
            'Notas de Matemática': self.mat,
            'Notas de Estatística': self.est,
            'Notas de Programação': self.prog,
            'Faltas': self.faltas
        }
```

Para a importação de dados, meu método foi separar cada coluna em uma variavel, por exemplo, coluna de idade seria a coluna 1.

Após isso, um método que achei para visualizar/fazer o loop facilmente foi pegar essas variaveis e atribuir um dicionário pra facilitar o loop em generate_report().

```python
class AnaliseEstatistica:
    def mean(self, array):
        return np.mean(array)

    def median(self, array):
        return np.median(array)

    def mode(self, array):
        valores, counts = np.unique(array, return_counts=True)
        indice_max = np.argmax(counts)
        return valores[indice_max]

    def std(self, array):
        return np.std(array)

    def variance(self, array):
        return np.var(array)

    def interval(self, array):
        return np.max(array) - np.min(array)

    def z_score(self, array):
        return (array - np.mean(array)) / np.std(array)
```

Na parte de análise, descobri que o numpy ja tem várias funções que iria facilitar o trabalho, como o numpy.mean, median, std pra desvio padrão, var. Não achei a necessidade de usar a biblioteca Math visto que já tinham muitas funções pelo proprio Numpy.

Alguns não tinham as funções diretamente ditas como median pra mediana, mas com outras como max e min eu consegui fazer a função de intervalo, por exemplo.

```python
class DeteccaoOutliers:
    def cerca_turkey(self, array):
        q1 = np.percentile(array, 25)
        q3 = np.percentile(array, 75)
        iqr = q3 - q1
        limite_inferior = q1 - 1.5 * iqr
        limite_superior = q3 + 1.5 * iqr
        return limite_inferior, limite_superior

    def outliers_tukey(self, array):
        li, ls = self.cerca_turkey(array)
        return array[(array < li) | (array > ls)]

    def outliers_z_score(self, array, limite=3):
        z = (array - np.mean(array)) / np.std(array)
        return array[(z < -limite) | (z > limite)]
```

Muito simples, a formula de tukey e os outliers foram ensinados em aula, bastou saber os limites superior e inferior e passar o parâmetro de outliers. (array < (x)), (array > x)

```python
dados = ImportarDados("base.csv")
analise = AnaliseEstatistica()
outlier = DeteccaoOutliers()

def generate_report():
    print("Relatório\n")

    for nome, var in dados.variaveis.items():
        print(f"--- {nome} ---")
        print(f"Média: {analise.mean(var):.2f}")
        print(f"Mediana: {analise.median(var):.2f}")
        print(f"Moda: {analise.mode(var)}")
        print(f"Desvio Padrão: {analise.std(var):.2f}")
        print(f"Variância: {analise.variance(var):.2f}")
        print(f"Intervalo: {analise.interval(var):.2f}")
        print(f"Z-Score (amostragem): {np.round(analise.z_score(var), 2)}")

        inf, sup = outlier.cerca_turkey(var)
        print(f"Cerca de Tukey: Inferior = {inf:.2f}, Superior = {sup:.2f}")

        outliers_tukey = outlier.outliers_tukey(var)
        outliers_z = outlier.outliers_z_score(var)

        print(f"Outliers (Tukey): {outliers_tukey}")
        print(f"Outliers (Z-Score): {outliers_z}")
        print()
```

Como dito no começo, na hora de fazer o relatório eu puxei os valores que guardei no dicionario e fui exibindo através de um loop de lá.

Em primeira mão eu tinha feito de uma forma que extendia muito:

```python
def generate_report():

    print("A média: ")
    for nome, variavel in variaveis.items():
        print(f"{nome}, {mean(variavel)}")

    print("A mediana: ")
    for nome, variavel in variaveis.items():
        print(f"{nome}, {median(variavel)}")

    print("A moda: ")
    for nome, variavel in variaveis.items():
        print(f"{nome}, {mode(variavel)}")

    print("O desvio padrão: ")
    for nome, variavel in variaveis.items():
        print(f"{nome}, {std(variavel):.2f}")
```

Daí eu notei que dava pra encurtar o código indo no dicionário diretamente.

```python
for nome, var in dados.variaveis.items():
    print(f"--- {nome} ---")
    print(f"Média: {analise.mean(var):.2f}")
    print(f"Mediana: {analise.median(var):.2f}")
    print(f"Moda: {analise.mode(var)}")
    print(f"Desvio Padrão: {analise.std(var):.2f}")
    print(f"Variância: {analise.variance(var):.2f}")
    print(f"Intervalo: {analise.interval(var):.2f}")
    print(f"Z-Score (amostragem): {np.round(analise.z_score(var), 2)}")
```

Pontos a melhorar:

1. Em moda, encontrar uma forma de implementar a bimodal, sem o uso de bibliotecas externas como scipy e a counter(que foi demonstrada em aula).

```python
def mode(self, array):
    valores, counts = np.unique(array, return_counts=True)
    indice_max = np.argmax(counts)
    return valores[indice_max]
```

Notei que só estava pegando o primeiro indice.

2. Implementar corretamente o ID para cada outliers, dessa forma é possivel gerar um relatório mais completo e aprofundado.