GEIA 25.1- INTRODUÇÃO A CIÊNCIA DE DADOS

Aluno:

Arthur Arruda Mourao (2510307)

Monitor:

Arthur Veras

- Importação dos dados:

```python
import pandas as pd

df = pd.read_csv("base_ent2.csv")
class ImportarDados:
    def __init__(self):
        self.ID = df["ID"].sort_values()
    #-------------------------------------------------------------#
        self.Idade = df["Idade"].sort_values()
    #-------------------------------------------------------------#
        self.Nota_Matematica = df["Nota_Matematica"].sort_values()
    #-------------------------------------------------------------#
        self.Nota_Estatistica = df["Nota_Estatistica"].sort_values()
    #-------------------------------------------------------------#
        self.Nota_Programacao = df["Nota_Programacao"].sort_values()
    #-------------------------------------------------------------#
        self.Faltas = df["Faltas"].sort_values()
    #-------------------------------------------------------------#
        self.Sexo = df["Sexo"].sort_values()
    #-------------------------------------------------------------#
        self.Cidade = df["Cidade"].sort_values()
    #-------------------------------------------------------------#

```

Primeiramente, o Dataframe (nesse caso o arquivo.csv) foi atribuido a uma variavel df, que sera utilizada ao longo do codigo para visualisamos algumas propriedades do dataframe ao longo do codigo. Logo em seguida foi criado a classe ImportarDados, a qual sera utilizada para importar os valores do dataframe ao codigo. Na segunda linha da função "__init__" por exemplo, Declarei a "Idade" como a coluna da idade no dataframe (df["Idade"]) e com os valores ordenados em ordem crescente utilizando o .sort_values() no final da declaração.

- Analise estatistica:

```python
class AnaliseEstatistica:
    def __init__(self, dados):
        self.dados = dados
    def media(self):
        Idade_Media = float(self.dados.Idade.mean())
        Nota_Matematica_Media = float(self.dados.Nota_Matematica.mean())
        Nota_Estatistica_Media = float(self.dados.Nota_Estatistica.mean())
        Nota_Programacao_Media = float(self.dados.Nota_Programacao.mean())
        Faltas_Media = float(self.dados.Faltas.mean())
        return (f"Media das idades: {Idade_Media}\n"
                f"Media das notas de matematica: {Nota_Matematica_Media}\n"
                f"Media das notas de estatistica: {Nota_Estatistica_Media}\n"
                f"Media das notas de programacao: {Nota_Programacao_Media}\n"
                f"Media das faltas: {Faltas_Media}\n"
                )
    def median(self):
        Idade_Median = float(self.dados.Idade.median())
        Nota_Matematica_Median = float(self.dados.Nota_Matematica.median())
        Nota_Estatistica_Median = float(self.dados.Nota_Estatistica.median())
        Nota_Programacao_Median = float(self.dados.Nota_Programacao.median())
        Faltas_Median = float(self.dados.Faltas.median())
        return (f"Mediana das idades: {Idade_Median}\n"
                f"Mediana das notas de matematica: {Nota_Matematica_Median}\n"
                f"Mediana das notas de estatistica: {Nota_Estatistica_Median}\n"
                f"Mediana das notas de programacao: {Nota_Programacao_Median}\n"
                f"Mediana das faltas: {Faltas_Median}\n"
                )
```

```python
    def mode(self):
        Idade_Mode = self.dados.Idade.mode().tolist()
        Nota_Matematica_Mode = self.dados.Nota_Matematica.mode().tolist()
        Nota_Estatistica_Mode = self.dados.Nota_Estatistica.mode().tolist()
        Nota_Programacao_Mode = self.dados.Nota_Programacao.mode().tolist()
        Faltas_Mode = self.dados.Faltas.mode().tolist()
        Sexo_Mode= self.dados.Sexo.mode().tolist()
        Cidade_Mode = self.dados.Cidade.mode().tolist()
        return (f"Moda(s) das idades: {Idade_Mode}\n"
                f"Moda(s) das notas de matematica: {Nota_Matematica_Mode}\n"
                f"Moda(s) das notas de estatistica: {Nota_Estatistica_Mode}\n"
                f"Moda(s) das notas de programacao: {Nota_Programacao_Mode}\n"
                f"Moda(s) das faltas: {Faltas_Mode}\n"
                f"Moda(s) dos Sexos: {Sexo_Mode}\n"
                f"Moda(s) das Cidades: {Cidade_Mode}\n"
                )
    def std(self):
        Idade_Std = round(float(self.dados.Idade.std()), 2)
        Nota_Matematica_Std = round(float(self.dados.Nota_Matematica.std()),2)
        Nota_Estatistica_Std = round(float(self.dados.Nota_Estatistica.std()),2)
        Nota_Programacao_Std = round(float(self.dados.Nota_Programacao.std()),2)
        Faltas_Std = round(float(self.dados.Faltas.std()),2)
        return (f"Mediana das idades: {Idade_Std}\n"
                f"Mediana das notas de matematica: {Nota_Matematica_Std}\n"
                f"Mediana das notas de estatistica: {Nota_Estatistica_Std}\n"
                f"Mediana das notas de programacao: {Nota_Programacao_Std}\n"
                f"Mediana das faltas: {Faltas_Std}\n"
                )
```

```
    def interval(self):
        Idade_Interval_min = round(float(self.dados.Idade.min()), 2)
        Idade_Interval_max = round(float(self.dados.Idade.max()), 2)
        Nota_Matematica_min = round(float(self.dados.Nota_Matematica.min()),2)
        Nota_Matematica_max = round(float(self.dados.Nota_Matematica.max()),2)
        Nota_Estatistica_min = round(float(self.dados.Nota_Estatistica.min()),2)
        Nota_Estatistica_max = round(float(self.dados.Nota_Estatistica.max()),2)
        Nota_Programacao_min = round(float(self.dados.Nota_Programacao.min()),2)
        Nota_Programacao_max = round(float(self.dados.Nota_Programacao.max()),2)
        Faltas_min = round(float(self.dados.Faltas.min()),2)
        Faltas_max = round(float(self.dados.Faltas.max()),2)
        return (f"Intervalo das idades: [{Idade_Interval_min},{Idade_Interval_max}]\n"
                f"Mediana das notas de matematica: [{Nota_Matematica_min},{Nota_Matematica_max}]\n"
                f"Mediana das notas de estatistica: [{Nota_Estatistica_min},{Nota_Estatistica_max}]\n"
                f"Mediana das notas de programacao: [{Nota_Programacao_min},{Nota_Programacao_max}]\n"
                f"Mediana das faltas: [{Faltas_min},{Faltas_max}]\n"
                )
    def variance(self):
        Idade_variance = float(self.dados.Idade.var())
        Nota_Matematica_variance = float(self.dados.Nota_Matematica.var())
        Nota_Estatistica_variance = float(self.dados.Nota_Estatistica.var())
        Nota_Programacao_variance = float(self.dados.Nota_Programacao.var())
        Faltas_variance = float(self.dados.Faltas.var())
        return (f"Variancia das idades: {Idade_variance}\n"
                f"Variancia das notas de matematica: {Nota_Matematica_variance}\n"
                f"Variancia das notas de estatistica: {Nota_Estatistica_variance}\n"
                f"Variancia das notas de programacao: {Nota_Programacao_variance}\n"
                f"Variancia das faltas: {Faltas_variance}\n"
                )
```

Na classe AnaliseEstatistica, como o proprio nome ja diz, foi utilizada para calcular medidas de estatisticas como: media, mediana, moda, desvio padrao, intervalo e variancia utilizando-se as funções: media(), median(), mode(), std(), interval() e variance() respectivamente. Antes de tudo, porem, foi declarado uma variavel dados como uma forma de se referir a classe ImportarDados

```
dados = ImportarDados()
```

Ou seja, na linha de codigo "Idade_Media = float(self.dados.Idade.mean())" ele pega a coluna "Idade" do dataframe, calcula a media pelo metodo .mean(), e retorna o valor desse float para a variavel "Idade_Media ". E foi de forma similar que foi feito as funções media(), median(), mode(), std() e variance() (A diferença sendo que cada função estava utilizando seu metodo adequado). ja na função variance(), primeiro foi descoberto o menor valor e depois o maior valor de cada coluna do dataframe para em seguida determinar o intervalo dos dados.

OBS: E importante notar que as colunas "Sexo" e "Cidade" so apareceram na função moda pois não são numeros, portanto não e possivel calcular uma media, desvio padrão e etc.

- Deteção de outliers:

```python
class DeteccaoOutliers:
    def __init__(self, dados):
        self.dados = dados
    def zscore(self):
        Idade_Zscore = ((df['Idade'] - self.dados.Idade.mean())/self.dados.Idade.std())
        Nota_Matematica_Zscore = ((df['Nota_Matematica'] - self.dados.Nota_Matematica.mean())/self.dados.Nota_Matematica.std())
        Nota_Estatistica_Zscore = ((df['Nota_Estatistica'] - self.dados.Nota_Estatistica.mean())/self.dados.Nota_Estatistica.std())
        Nota_Programacao_Zscore = ((df['Nota_Programacao'] - self.dados.Nota_Programacao.mean())/self.dados.Nota_Programacao.std())
        Faltas_Zscore = ((df['Faltas'] - self.dados.Faltas.mean())/self.dados.Faltas.std())
        Idade_outliers = (abs(Idade_Zscore) > 3).sum()
        Nota_Matematica_outliers = (abs(Nota_Matematica_Zscore) > 3).sum()
        Nota_Estatistica_outliers = (abs(Nota_Estatistica_Zscore) > 3).sum()
        Nota_Programacao_outliers = (abs(Nota_Programacao_Zscore) > 3).sum()
        Faltas_outliers = (abs(Faltas_Zscore) > 3).sum()
        outliers_totais = Idade_outliers + Nota_Matematica_outliers + Nota_Estatistica_outliers + Nota_Programacao_outliers + Faltas_outliers
        return (f"Foram detectados {outliers_totais} outlies utlizando-se o metodo Zscore \n"
                f"{Idade_outliers} deles foram detectados na sessao de: Idade \n"
                f"{Nota_Matematica_outliers} deles foram detectados na sessao de: Nota_Matematica \n"
                f"{Nota_Estatistica_outliers} deles foram detectados na sessao de: Nota_Estatistica \n"
                f"{Nota_Programacao_outliers} deles foram detectados na sessao de: Nota_Programacao \n"
                f"{Faltas_outliers} deles foram detectados na sessao de: Faltas \n"
                )
```

```python
    def cerca_tukey(self):
        idade_limite_inferior = df['Idade'].quantile(0.25) - 1.5 * (df['Idade'].quantile(0.75) - df['Idade'].quantile(0.25))
        idade_limite_superior = df['Idade'].quantile(0.75) + 1.5 * (df['Idade'].quantile(0.75) - df['Idade'].quantile(0.25))
        Idade_outliers = (df['Idade'] < idade_limite_inferior) | (df['Idade'] > idade_limite_superior)
        #-----------------------------------------------------------------------------------------------------------------------------------
        Matematica_limite_inferior = df['Nota_Matematica'].quantile(0.25) - 1.5 * (df['Nota_Matematica'].quantile(0.75) - df['Nota_Matematica']
        Matematica_limite_superior = df['Nota_Matematica'].quantile(0.75) + 1.5 * (df['Nota_Matematica'].quantile(0.75) - df['Nota_Matematica']
        Nota_Matematica_outliers = (df['Nota_Matematica'] < Matematica_limite_inferior) | (df['Nota_Matematica'] > Matematica_limite_superior)
        #-----------------------------------------------------------------------------------------------------------------------------------
        Estatistica_limite_inferior = df['Nota_Estatistica'].quantile(0.25) - 1.5 * (df['Nota_Estatistica'].quantile(0.75) - df['Nota_Estatisti
        Estatistica_limite_superior = df['Nota_Estatistica'].quantile(0.75) + 1.5 * (df['Nota_Estatistica'].quantile(0.75) - df['Nota_Estatisti
        Nota_Estatistica_outliers = (df['Nota_Estatistica'] < Estatistica_limite_inferior) | (df['Nota_Estatistica'] > Estatistica_limite_super
        #-----------------------------------------------------------------------------------------------------------------------------------
        Programacao_limite_inferior = df['Nota_Programacao'].quantile(0.25) - 1.5 * (df['Nota_Programacao'].quantile(0.75) - df['Nota_Programac
        Programacao_limite_superior = df['Nota_Programacao'].quantile(0.75) + 1.5 * (df['Nota_Programacao'].quantile(0.75) - df['Nota_Programac
        Nota_Programacao_outliers = (df['Nota_Programacao'] < Programacao_limite_inferior) | (df['Nota_Programacao'] > Programacao_limite_super
        #-----------------------------------------------------------------------------------------------------------------------------------
        Faltas_limite_inferior = df['Faltas'].quantile(0.25) - 1.5 * (df['Faltas'].quantile(0.75) - df['Faltas'].quantile(0.25))
        Faltas_limite_superior = df['Faltas'].quantile(0.75) + 1.5 * (df['Faltas'].quantile(0.75) - df['Faltas'].quantile(0.25))
        Faltas_outliers = (df['Faltas'] < Faltas_limite_inferior) | (df['Faltas'] > Faltas_limite_superior)
        tukey_outliers = Faltas_outliers.sum() + Nota_Programacao_outliers.sum() + Nota_Estatistica_outliers.sum() + Nota_Matematica_outliers.s
        return (f"Foram detectados {tukey_outliers} outlies utlizando-se o metodo da cerca de tukey: \n"
                f"{Idade_outliers.sum()} deles foram detectados na sessao de: Idade \n"
                f"{Nota_Matematica_outliers.sum()} deles foram detectados na sessao de: Nota_Matematica \n"
                f"{Nota_Estatistica_outliers.sum()} deles foram detectados na sessao de: Nota_Estatistica \n"
                f"{Nota_Programacao_outliers.sum()} deles foram detectados na sessao de: Nota_Programacao \n"
                f"{Faltas_outliers.sum()} deles foram detectados na sessao de: Faltas \n"
                )
```

A classe DetecaoOutliers, serve para detectar os outliers (valores que se distanciam significativamente da tendencia geral do restante dos dados), ou seja, valores anormais fora do esperado. Existem 2 metodos para detectar outliers, sendo eles o Zscore, e a Cerca de Tukey. Primeiramente, calculamos Zscore de acordo com a seguinte formula:

$$Z = (x-u)/o$$

sendo "x" igual ao valor que se deseja testar, "u" a media da amostra e "o" o desvio padrao. Se o Z tiver o resultado (score) maior que 3, ele podera ser considerado um outlier. Então o codigo ficara algo como:

```
coluna_Zscore = (df['coluna'] - self.dados.coluna.mean())/self.dados.coluna.std()
```

sendo df['coluna'] o "x", o self.dados.coluna.mean() a media dos dados, e o self.dados.coluna.std() o desvio padrão dos dados. Porem, essa linha retornara o index dos valores que sao outliers, para calculamos a quantidade total, devemos utilizar: coluna_outliers = (abs(coluna_Zscore) > 3).sum()

Para calcular os outliers da cerca de tukey, devemos calcular o primeiro e o terceiro quartil (que nesse caso podemos utilizar o metodo .quantile(valor), que nesse caso o valor sera 0.25 e 0.75 respectivamente), calcular o Intervalo Interquartil (IQR) que e a diferença entre o terceiro e o primeiro quartil (q3 - q1). Depois devemos calcular o limite inferior (q1 - 1.5 * iqr) e o limite superior (q3 + 1.5 * iqr). Se o dado for menor que o limite inferior ou maior que o limite superior, ele sera considerado um outlier, o que em codigo fica assim:

```
coluna_limite_inferior = df['coluna'].quantile(0.25) - 1.5 * (df['Idade'].quantile(0.75) - df['coluna'].quantile(0.25))

coluna_limite_superior = df['coluna'].quantile(0.75) + 1.5 * (df['coluna'].quantile(0.75) - df['coluna'].quantile(0.25))

    coluna_outliers = (df['coluna'] < coluna_limite_inferior) | (df['coluna'] > coluna_limite_superior)
```

Porem, a mesma coisa ocorre com esse calculo, ele fornece o index dos outliers, e não as quantidades, para isso devemos usar o .sum() ao final de coluna_outliers

Report:

```
158 class Report:
159     def __init__(self):
160         self.dados = dados
161     def generate_report(self):
162         return (
163             f"Dataframe original: {df} \n"
164             f"\n"
165             f"||-----------||-----Analise Estatistica:-----||-----------||\n"
166             f"--------------Media-----------------\n"
167             f"{AnaliseEstatistica.media(self)}\n"
168             f"--------------Mediana-----------------\n"
169             f"{AnaliseEstatistica.median(self)}\n"
170             f"--------------Moda-----------------\n"
171             f"{AnaliseEstatistica.mode(self)}\n"
172             f"--------------Desvio padrao-----------------\n"
173             f"{AnaliseEstatistica.std(self)}\n"
174             f"--------------Intervalo-----------------\n"
175             f"{AnaliseEstatistica.interval(self)}\n"
176             f"--------------Variancia-----------------\n"
177             f"{AnaliseEstatistica.variance(self)}\n"
178             f"||-----------||-----Detecao de outliers:-----||-----------||\n"
179             f"--------------Zscore-----------------\n"
180             f"{DeteccaoOutliers.zscore(self)}\n"
181             f"--------------Cerca de Tukey-----------------\n"
182             f"{DeteccaoOutliers.cerca_tukey(self)}\n"
183             )
184 dados = ImportarDados()
185 resultado = Report()
186 print(resultado.generate_report())
```

Por fim, criei uma ultima classe Report para gerar um relatorio dos dados.