Exercício de apoio - Semana 2

Neste exercício, faremos o pré-processamento de uma base de dados, aplicando alguns dos conceitos vistos na aula da semana 2.

1. Acesse o Google Colab, lembrando que é necessário ter uma conta google para conseguir usar este ambiente. Se preferir, você pode também fazer o exercício usando o Jupyter Notebook.

2. Crie um novo notebook (New notebook) para fazer este exercício. Dê um nome para o seu caderno (sugestão: semana02-limpeza). Inclua uma descrição para o seu caderno clicando em "+ Text". Isso vai inserir uma célula de texto no caderno.

3. Em seguida, vamos importar as bibliotecas **pandas** e **numpy**.

```
import pandas as pd

import numpy as np
```

A base de dados de exemplo simula registros de casos de pessoas que foram infectadas com a COVID-19 no Brasil. O dicionário de dados da base é o seguinte:

- **id**: campo identificador do registro;
- **idade**: idade da pessoa;
- **uf**: estado onde a pessoa foi diagnosticada;
- **renda**: classe social a qual a pessoa pertence, variando entre A e E;
- **vacina**: indica se a pessoa foi vacinada (1) ou não (0).

4. Importe a base de dados direto da URL a seguir e verifique as primeiras linhas. O arquivo contém 50 registros. Olhando os 20 primeiros, podemos observar que há valores ausentes nos atributos **idade**, **uf** e **renda**.

```
url = 'https://raw.githubusercontent.com/higoramario/univesp-com360-mineracao-dados/main/dados-covid-limpeza.csv'

casos_covid = pd.read_csv(url)

casos_covid.head(20)
```

5. Podemos verificar quais são os tipos de dados de cada coluna usando o comando abaixo (função **info()**). Para cada atributo, o resultado mostra o número de colunas

não nulas e o tipo de dados. Por exemplo, o atributo idade tem 30 valores não nulos. Campos textuais são considerados como objetos.

```
casos_covid.info()
```

6. A seguir, podemos ver a análise descritiva (função **describe()**) desses dados olhando a distribuição dos dados. Por exemplo, a média de idade é 48 anos, com desvio-padrão de 22 anos. Já para os dados nominais (**uf**) e ordinais (**renda**), não temos esses valores. Como os dados do atributo **vacina** são binários, a média também não faz sentido.

```
casos_covid.describe()
```

7. Vamos olhar a quantidade de valores distintos desses atributos com a função **value_counts()**.

```
casos_covid['uf'].value_counts()
```

8. No atributo **uf**, há mais registros de São Paulo. Além disso, há registros de São Paulo e do Ceará em maiúsculo e minúsculo. Vamos fazer a padronização dos dados da coluna **uf** usando o método **upper()**. Isso vai mudar a contagem de registros por estado.

```
casos_covid['uf'] = casos_covid['uf'].str.upper()
casos_covid['uf'].value_counts()
```

9. Agora vamos olhar os valores de **renda** e **vacina**.

```
casos_covid['renda'].value_counts()
```

```
casos_covid['vacina'].value_counts()
```

Podemos observar algumas características desses dados:

- 35 das 50 pessoas diagnosticadas com COVID-19 não tomaram vacina.
- Há mais pessoas da classe C entre os infectados.

10. Agora vamos fazer alguns pré-processamentos para imputar os valores ausentes na base de dados usando as características dos dados. Na coluna **idade** vamos usar a média para imputar os valores ausentes.

11. Para isso, vamos usar a função **fillna()**, que preenche os valores nulos a partir de algum método, nesse caso, arredondando a **média** de idade, já que é um valor inteiro. Olhando a quantidade de registros por valores distintos, vemos o resultado da modificação.

```
casos_covid['idade'].fillna(round(casos_covid['idade'].mean()), inplace=True)

casos_covid['idade'].value_counts()
```

12. Para preencher o campo **uf**, vamos usar o método de **imputar o valor de acordo com a última observação**. Para isso, vamos escolher a opção **ffill** na função **fillna**.

    Olhando a contagem de registros por estado, podemos ver que a proporção foi minimamente mantida.

```
casos_covid['uf'].fillna(method="ffill",inplace=True)

casos_covid['uf'].value_counts()
```

13. Agora vamos fazer a imputação de **renda** usando o método de **imputar o valor de acordo com a observação seguinte**, que preenche os valores ausentes a partir da observação seguinte válida (ou para trás). Para isso, vamos escolher a opção **bfill** na função **fillna**.

```
casos_covid['renda'].fillna(method="bfill",inplace=True)
```

```
casos_covid['renda'].value_counts()
```

14. Com isso, fizemos um processo de limpeza de dados, que pode ser verificado usando o método **info()** ou olhando o dataset inteiro.

```
casos_covid.info()
```

```
casos_covid.head(len(casos_covid))
```

15. Com isso, concluímos o exercício de limpeza de dados desta semana. Para praticar mais, sugerimos que você escolha um outro conjunto de dados do seu interesse e verifique se há necessidade de limpeza (geralmente isso acontece). Se for o caso, faça a limpeza utilizando o conteúdo que você aprendeu esta semana.

    Algumas sugestões de bases são:

    - Portal Brasileiro de Dados Abertos
    - Basedosdados.org
    - Kaggle