# DOCUMENTAÇÃO

## TRATAMENTO DATASET: ‘ACIDENTES’

**1- Importação da biblioteca ‘Pandas’:**

*Para que possamos utilizar as funcionalidades de cada uma das bibliotecas no Google Colab, é necessário que inicialmente façamos a importação.*

**2- Integração com a Google Cloud Platform (GCP):**

*Cumprindo o requisito de armazenar os datasets originais no ‘bucket’ da GCP, esse passo é essencial para que possamos conectar a sessão do Colab com a conta administradora da GCP, fornecendo, assim, permissão para que possamos importar para o ‘notebook’ o dataset desejado.*

**3- Importação dos Datasets por meio do comando ‘gsutil’:**

*O ‘gsutil’ é uma aplicação em Python que permite acesso a GCP pela linha de comando. Utilizamos-na em conjunto com o comando ‘cp’ para copiar o arquivo desejado no ‘bucket’ para uma pasta temporária (tmp) no Colab. A seguir, por meio da função ‘pd.read’, fazemos a leitura dos datasets localizados na pasta tmp, obtendo os dataframes (df).*

**4- Listagem dos Dataframes:**

*Para dar base ao comando seguinte, criamos uma lista contendo os dois dataframes importados para o Colab.*

**5- Concatenando Dataframes:**

*Para aumentar o escopo da análise, optamos pela utilização de dois datasets (2019-2020). Portanto, no sentido de torná-los um - facilitando os tratamentos - fizemos uso do comando ‘pd.concat’.*

**6- Eliminando colunas sem pertinência para o estudo:**

*Visto que já dispúnhamos de informações suficientes para as correlações e análises pretendidas, tomamos a liberdade de ‘dropar’ colunas de baixa pertinência a fim de reduzir os dados mostrados e facilitando o tratamento e visualização*

**7- Renomeando os títulos das colunas:**

*Apesar de já estarem em pt-br, optamos por renomear as colunas para simplificar a interpretação e melhorar a visualização dos dados.*

**8- Verificando o sucesso das etapa 6 e 7:**

*Para verificarmos as alterações feitas nos passos anteriores, utilizamos o comando 'dff'. Por meio desse, obtemos a exibição das cinco primeiras e cinco últimas linhas do dataframe. O suficiente para verificarmos as alterações propostas.*

**9- Analisando tipo de dado e nulos:**

*É de grande importância no processo de tratamento de dados a verificação percentual de quanto do dataframe está comprometido com valores nulos. Paralelamente, é também interessante estudar o tipo dos dados (dt type) para garantir que estejam na formatação correta para as manipulações pretendidas. Para tanto, utilizamos o comando 'df.info', que nos revela a contagem de não nulos, juntamente com o 'dt type'.*

**10- Somando o total de nulos para possível eliminação:**

*Seguindo a linha do estudo da etapa 9, agora utilizamos de um comando 'df.isnull' + 'sum' para que possamos identificar e somar o total de nulos. O objetivo desse movimento é para entendermos a significância na exclusão dos nulos. Caso haja muitos nulos, excluí-los, representa perigo à integridade dos dados*

**11- Eliminando possíveis duplicatas:**

*Nessa etapa utilizamos o comando 'df.drop_duplicates' para que, na existência de valores duplicados, as cópias fossem eliminadas. Melhorando, portanto, a qualidade dos dados.*

**12- Eliminando valores nulos em 'id_pessoa' e 'rodovia_br':**

*Seguindo o racional exposto na etapa 10, percebemos que a quantidade de nulos em 'id_pessoa' e 'rodovia_br' representava menos que 0,2% do dataframe. Logo, com o objetivo de clareza e qualidade dos dados, optamos por 'dropar' os nulos nessas colunas.*

**13- Verificando o sucesso da etapa anterior:**

*Com o comando 'df.info' visualizamos a contagem de nulos no dataframe. Exatamente o que precisávamos para verificar o sucesso do 'drop'.*

**14- Alterando o tipo dos dados:**

*De acordo com a lógica estabelecida na etapa 9, agora, tomamos a decisão de alterar o tipo dos dados. Cada qual para o formato ideal tendo em vista as manipulações e análises que serão feitas futuramente.*

**15- Verificando o sucesso da etapa anterior:**

*Com o comando 'df.info' visualizamos os tipos de dados no dataframe. Exatamente o que precisávamos para verificar o sucesso da operação anterior.*

**16- Analisando valores únicos na coluna ‘uf’**

*Por meio do comando ‘pd.unique’, analisamos a coluna ‘uf’ para identificarmos quais valores únicos estavam presentes. Felizmente, não haviam dados incongruentes.*

**17- Verificando o número de estados:**

*Apesar de já termos verificado que não existiam valores discrepantes na coluna dos estados, optamos por sermos extra cautelosos e rodar mais uma verificação da quantidade de valores únicos. Como temos 27 unidades federativas, a quantidade deveria ser exatamente 27. O funcionamento é: (para cada) + (valor único) + (somar 1 ao contador). A transcrição para código: (for) + (pd.unique) + (cont += 1).*

**18- Analisando valores únicos na coluna ‘dia_semana’**

*Por meio do comando ‘pd.unique’, analisamos a coluna ‘dia_semana’ para identificarmos quais valores únicos estavam presentes. Por meio dessa análise pudemos verificar que, para além da grafia tradicional (xxxx-feira), havia dias escritos somente pelo formato popular do primeiro nome.*

**19- Renomeando dias da semana escritos fora da grafia padrão:**

*Tendo em mente as descobertas do passo 18, resolvemos renomear os dias da semana que estivessem escritos fora da grafia tradicional (acompanhada por ‘-feira’). O motivo dessa alteração é para que os dados fiquem corretamente agrupados por cada dia da semana único, ou seja, ao invés de termos ‘X casos na terça’ e ‘Y casos na terça-feira’, teremos ‘X + Y casos na terça-feira’.*

**20- Verificando o sucesso da etapa anterior:**

*Com o comando ‘pd.unique’ visualizamos os valores únicos na coluna ‘dia_semana’. Exatamente o que precisávamos para verificar o sucesso da operação anterior.*

**21- Renomeando os dados da coluna ‘causa_acidente’**

*Visando melhor organização e clareza nos dados, optamos por renomear os valores da coluna ‘causa_acidentes’.*

**22- Verificando o sucesso da etapa anterior:**

*Com o comando ‘df.head(2)’ visualizamos apenas as duas primeiras linhas do dataframe. Exibição suficiente para, minimizando o processamento da máquina, obtermos feedback sobre a alteração feita na etapa anterior.*

**23- Análises de colunas:**

*Por meio do comando 'pd.unique', analisamos a coluna 'classificacao_acidente', 'condicao_metereologica', 'tipo_envolvido', 'estado_fisico' e 'sexo' para identificarmos quais valores únicos estavam presentes. Sem informações alarmantes.*

**24- Contagem de resultados 'cavaleiro' na coluna 'tipo_envolvido':**

*Com o uso da estrutura em Python do 'contador' (cont += 1) e uma relação de equivalência, estudamos quantas vezes o valor 'cavaleiro' era presente na coluna de tipo. Nenhum insight relevante.*

**25- Contagem de resultados 'Não Informado' na coluna 'estado_físico':**

*Novamente, com o uso da estrutura do contador e da relação de equivalência, investigamos quantas vezes o valor 'Não informado' era presente na coluna 'estado_físico'. Obtivemos um número significativo de 'NI'.*

**26- Contagem de resultados 'Não Informado' na coluna 'sexo':**

*No mesmo procedimento de análise supracitado, investigamos quantas vezes o valor 'Não informado' era presente na coluna 'sexo'. Obtivemos um número significativo de 'NI'.*

**27- Contagem de resultados 'Ignorado' na coluna 'sexo':**

*Repetindo o uso do contador e da relação de equivalência, investigamos quantas vezes o valor 'Ignorado' era presente na coluna 'sexo'. Nenhum insight relevante.*

**28- Contagem de valores únicos por coluna:**

*A título de maximizarmos nosso conhecimento sobre os pormenores do dataset, resolvemos somar os valores únicos por cada coluna.*

**29- Construção de gráficos para observações pontuais:**

*Seguindo as diretrizes propostas para o projeto, 'plotamos' gráficos para que pudéssemos visualizar alguns pontos específicos no relacionamento entre as colunas do dataframe e, paralelamente, demonstrar nosso conhecimento nas construções gráficas em Pandas.*

**30- Exportando o Dataframe tratado para o 'bucket' na GCP:**

*Finalizando as operações, realizamos o salvamento diretamente no 'bucket'. Para tanto, utilizamos primeiramente o comando 'df.to_csv' para criar um novo arquivo csv a partir do dataframe tratado. A seguir, assim como na importação, usamos o 'gsutil cp' a fim de copiar o arquivo pronto, presente nas dependências do Colab, para o Cloud Storage. Ao término do comando de cópia, temos o último comando 'gsutil cat', que é posto para que recebamos uma confirmação do sucesso da operação.*