

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data

Lilian Campos Soares

**CARACTERIZAÇÃO DE ACIDENTES DE TRÂNSITO EM TRECHOS DE
RODOVIAS FEDERAIS E A APLICAÇÃO DE MODELOS DE MACHINE
LEARNING PARA A CLASSIFICAÇÃO DO ESTADO FÍSICO DOS ENVOLVIDOS**

Belo Horizonte
2021

Lilian Campos Soares

**CARACTERIZAÇÃO DE ACIDENTES DE TRÂNSITO EM TRECHOS DE
RODOVIAS FEDERAIS E A APLICAÇÃO DE MODELOS DE MACHINE
LEARNING PARA A CLASSIFICAÇÃO DO ESTADO FÍSICO DOS ENVOLVIDOS**

Trabalho de Conclusão de Curso apresentado
ao Curso de Especialização em Ciência de
Dados e Big Data como requisito parcial à
obtenção do título de especialista.

Belo Horizonte

2021

SUMÁRIO

1. Introdução.....	7
1.1. Contextualização	9
1.2. O problema proposto	9
2. Metodologia	11
3. Coleta de dados.....	15
4. Montagem da base de dados	18
5. Processamento dos dados.....	34
6. Análise e exploração dos dados.....	40
7. Criação do modelo de Machine Learning.....	70
8. Interpretação dos resultados	105
9. Apresentação dos resultados	116
8. Links.....	118
REFERÊNCIAS.....	119

ÍNDICE DE FIGURAS

Figura 1: Etapas e procedimentos metodológicos	12
Figura 2: Versão do Jupyter Notebook	13
Figura 3: Bibliotecas do Python das etapas 2 a 5.....	13
Figura 4: Bibliotecas do Python da etapa 6.....	14
Figura 5: Configurações dos <i>dataframes</i> da biblioteca Pandas	14
Figura 6: Configurações dos <i>Plots</i> da biblioteca Matplotlib	15
Figura 7: Coleta dos dados do DNIT	15
Figura 8: Coleta dos dados da PRF	16
Figura 9: Verificação do resumo do SNV (dados por UF)	18
Figura 10: Verificação do resumo do VMDA (dados por UF).....	19
Figura 11: Junção dos arquivos da PRF.....	20
Figura 12: Verificação do arquivo “datatran.csv” da PRF	21
Figura 13: Amostra das 10 (dez) primeiras linhas do arquivo “datatran.csv” da PRF	23
Figura 14: Verificação do arquivo “acidentes.csv” da PRF	24
Figura 15: Amostra das 10 (dez) primeiras linhas do arquivo “acidentes.csv” da PRF	26
Figura 16: Quantidade de acidentes por ano/UF e do período/UF	27
Figura 17: Indicadores por ano/UF	28
Figura 18: Montagem do arquivo “datatran_mg”	31
Figura 19: Montagem do arquivo “acidentes_mg”	31
Figura 20: Processamento e tratamento do arquivo “datatran_mg”	35
Figura 21 Processamento e tratamento do arquivo “acidentes_mg”	35
Figura 22: 2º Processamento e tratamento do arquivo “datatran_mg”	36
Figura 23: 2º Processamento e tratamento do arquivo “acidentes_mg”	36
Figura 24: Conferindo o tratamento do arquivo “datatran_mg”	37
Figura 25: Conferindo o tratamento do arquivo “acidentes_mg”	37
Figura 26: Matriz de Haddon	38
Figura 27: Enriquecimento dos dados dos arquivos “datatran_mg” e “acidentes_mg”	39
Figura 28: Acidentes, período total e por ano	40
Figura 29: Feridos, período total e por ano	41
Figura 30: Mortos, período total e por ano	42
Figura 31: Pessoas e veículos envolvidos, período total e por ano	43
Figura 32: Estado físico dos envolvidos nos acidentes, período total	44
Figura 33: Estado físico dos envolvidos nos acidentes, por ano	45
Figura 34: Classificação de acidentes, período total	46
Figura 35: Classificação de acidentes, por ano	47
Figura 36: Acidentes por dia da semana, período total	48
Figura 37: Acidentes por dia da semana, por ano	49
Figura 38: Acidentes por mês, período total	50
Figura 39: Acidentes por tipo de veículo envolvido, período total	51
Figura 40: Acidentes por causa presumível, período total	52
Figura 41: Elemento de Trânsito, período total.....	53
Figura 42: Elemento de Trânsito, por ano	54
Figura 43: Acidentes por tipo, período total.....	55
Figura 44: Acidentes por fase do dia, período total	55
Figura 45: Acidentes por tipo, período total.....	55
Figura 46: Acidentes por tipo de pista, período total	56
Figura 47: Acidentes por traçado da via, período total	56
Figura 48: Acidentes por uso do solo, período total	56
Figura 49: Acidentes por BR, período total	57
Figura 50: Acidentes por sexo e envolvido, período total	57
Figura 51: Geração de indicadores relacionados aos acidentes de MG, por ano.....	58

Figura 52: indicadores relacionados aos acidentes de MG, por ano	60
Figura 53: Carga de arquivos no MS-PowerBI	63
Figura 54: Modelo de relações no MS-PowerBI	63
Figura 55: Dashboards do MS-PowerBI	64
Figura 56: Dashboard “Resumo período” do MS-PowerBI.....	64
Figura 57: Dashboard “Resumo período” do MS-PowerBI com seleção da BR-381	65
Figura 58: Dashboard “Por ano- geral” do MS-PowerBI.....	65
Figura 59: Dashboard “Por ano- geral” do MS-PowerBI com seleção de ano, BR e tipo de pista	66
Figura 60: Dashboard “Por ano - tempo” do MS-PowerBI.....	67
Figura 61: Dashboard “Por ano - tempo” do MS-PowerBI com seleção de ano, BR e dia da semana	67
Figura 62: Dashboard “Por ano - espaço” do MS-PowerBI	68
Figura 63: Dashboard “Por ano - espaço” do MS-PowerBI com seleção de ano, BR e tipo de pista	68
Figura 64: Dashboard “Por ano - espaço” do MS-PowerBI com seleção de ano, BR, tipo de pista e KM.....	69
Figura 65: Dashboard “Por ano - gravidade” do MS-PowerBI	69
Figura 66: Dashboard “Por ano - gravidade” do MS-PowerBI com seleção de ano e BR.....	70
Figura 67: Atributo ‘estado_fisico’ de “acidentes_mg3”	71
Figura 68: Notebook Python para a criação dos modelos de Machine Learning.....	71
Figura 69: Nova verificação da amostra de “acidentes_mg3”	72
Figura 70: Novo tratamento da amostra de “acidentes_mg3” e criação do “fset”	73
Figura 71: Tratamento em “fset” e alocação de variáveis preditoras e alvo	74
Figura 72: Transformação de dados não numéricos de X.....	74
Figura 73: Verificação de X e Y	75
Figura 74: Normalização dos dados de X	79
Figura 75: Divisão do conjunto para treinamento e teste	79
Figura 76: Teste comparativo de desempenho com o SGDClassifier	80
Figura 77: Teste do algoritmo Regressão Logística com treinamento SGD do modelo	81
Figura 78: Teste do algoritmo Linear SVM com treinamento SGD do modelo	82
Figura 79: Comparação de desempenho entre diferentes modelos lineares com SGD	82
Figura 80: Parâmetros mais adequados para o SGDClassifier	83
Figura 81: Regressão Logística com treinamento SGD do modelo e os hiper parâmetros adequados.....	84
Figura 82: Armazenamento das previsões do SGDClassifier	84
Figura 83: Índice Jaccard da Regressão Logística com treinamento SGD do modelo	84
Figura 84: Erros de classificação da Regressão Logística com treinamento SGD do modelo ..	84
Figura 85: Matriz de Confusão da Regressão Logística com treinamento SGD do modelo ..	85
Figura 86: Teste do algoritmo Decision Tree Classifier	87
Figura 87: Avaliação do modelo do Decision Tree Classifier.....	87
Figura 88: Parâmetros mais adequados para o Decision Tree Classifier	88
Figura 89: Decision Tree Classifier e os hiper parâmetros adequados	88
Figura 90: Matriz de Confusão do Decision Tree Classifier	89
Figura 91: Árvore gerada pelo modelo do Decision Tree Classifier	89
Figura 92: Teste do algoritmo Random Forest Classifier	90
Figura 93: Avaliação do modelo do Random Forest Classifier	90
Figura 94: Parâmetros mais adequados para o Random Forest Classifier	91
Figura 95: Verificação da <i>feature</i> de importância	92
Figura 96: Random Forest Classifier e os hiper parâmetros adequados	93
Figura 97: Matriz de Confusão do Random Forest Classifier.....	93
Figura 98: Montagem de um novo dataframe para implementar o modelo escolhido	94
Figura 99: Procedimentos auxiliares e verificações do novo dataframe	95

Figura 100: Transformação de dados não numéricos e normalização dos dados de X2.....	95
Figura 101: Inclusão de coluna ‘estado_fisico’ com valores da predição em fset2	96
Figura 102: Verificação de “acidentes_mg” e de “fset2” para o atributo de ‘estado_fisico’	96
Figura 103: Atualização dos dados de predição em “acidentes_mg”	97
Figura 104: Verificação do dataframe “acidentes_mg” modificado	97
Figura 105: Criando uma variável <i>dummy</i> a partir de ‘estado_fisico’	99
Figura 106: Ajustes no dataframe “acidentes_mg” com a variável <i>dummy</i>	100
Figura 107: Verificação dos novos ajustes no dataframe “acidentes_mg”	101
Figura 108: Carga de dados com os dataframes “acidentes_mg” e “datatran_mg”	101
Figura 109: Criação de dataframe auxiliar “estado_vitima”	103
Figura 110: Ajustes no dataframe “datatran_mg” – parte 1	103
Figura 111: Ajustes no dataframe “datatran_mg” – parte 2	104
Figura 112: Notebook Python para interpretação dos resultados	105
Figura 113: Quantidade de acidentes, feridos, mortos e envolvidos	106
Figura 114: Estado físico das vítimas com os valores de predição	107
Figura 115: Classificação dos acidentes com os valores de predição	109
Figura 116: Classificação dos acidentes com os valores de predição	109
Figura 117: Dashboard “Resumo período” do MS-PowerBI, com valores da predição	113
Figura 118: Dashboard “Por ano- geral” do MS-PowerBI com seleção de ano, BR e tipo de pista, com valores da predição.....	113
Figura 119: Dashboard “Por ano - tempo” do MS-PowerBI com seleção de ano, BR e dia da semana, com valores da predição.....	114
Figura 120: Dashboard “Por ano - espaço” do MS-PowerBI com seleção de ano, BR e tipo de pista, com valores da predição.....	115
Figura 121: Dashboard “Por ano - gravidade” do MS-PowerBI com seleção de ano e BR, com valores da predição.....	115

ÍNDICE DE QUADROS

Quadro 1: Resultado da verificação do resumo do SNV (dados por UF) – Top 05 UFs	18
Quadro 2: Resultado da verificação do resumo do VMDA (dados por UF) – Top 05 UFs	19
Quadro 3: Resultado da verificação dos acidentes por UF – Top 05 UFs	27
Quadro 4: Indicadores por ano/UF – Top 05 UFs por quantidade de acidentes	30
Quadro 5: Causa acidente vs Elemento do Trânsito	39
Quadro 6: Índices de Acidentes, por BR, por ano	60
Quadro 7: Comparativo de ‘estado_fisico’ das vítimas	106
Quadro 8: Índices de Acidentes, por BR, por ano, com valores da predição.....	111

ÍNDICE DE TABELAS

Tabela 1: Descrição dos campos/colunas dos datasets.....	32
--	----

1. Introdução

Mortes e lesões por acidentes de trânsito são consideradas um problema de saúde pública, exigindo ações para prevenção e redução de suas consequências. Considerando as vertentes do trânsito seguro, a atuação do governo e sociedade deve estar pautada em políticas públicas de fiscalização, educação e engenharia. Conhecer os fatores contribuintes dos acidentes é fundamental para intervir preventivamente, avaliando as interações entre os elementos que compõem o sistema de trânsito (o homem, o veículo e a via), identificando as causas dos acidentes e agindo na redução de suas consequências.

De acordo com dados divulgados pela Empresa de Planejamento e Logística (EPL, 2020), quanto ao panorama 2015 de transporte inter-regional de cargas no Brasil, o modo rodoviário apresenta a maior participação na matriz de transportes, mantendo patamares históricos acima dos 60%. Segundo informações da Confederação Nacional do Transporte (CNT, 2016), o modo rodoviário é predominante na matriz de transporte de passageiros; em 2019, foram transportados 80,04 milhões de passageiros no transporte regular de passageiros (CNT, 2020).

Como a principal alternativa de movimentação de carga e de passageiros do País, o modo rodoviário é usado para deslocamentos, não apenas em curtas e médias distâncias, mas também, para viagens longas, de maneira que é cada vez mais importante uma malha rodoviária em condições ideais de utilização. Segundo as Pesquisa CNT de Rodovias, a densidade de malha rodoviária pavimentada do Brasil é pequena, quando comparado com países de dimensão territorial semelhante, e “parte das rodovias pavimentadas brasileiras não é considerada adequada para o tráfego de pessoas e bens” (CNT, 2016). Em avaliação da CNT em 2019, 59,2% das rodovias avaliadas apresentaram algum tipo de problema no estado geral (CNT, 2020).

Com uma demanda de transporte de carga e passageiros predominante nas vias rodoviárias e um ritmo crescente da frota de veículos brasileira – dados do DENATRAN (2021) indicam que de 2011 a 2020 a frota de veículos no Brasil aumentou em aproximadamente 153%, há uma tendência de aumento nos conflitos

existentes, o que implica em mais acidentes e em mais mortes e lesões de trânsito (IPEA, 2015).

De acordo com o Instituto de Pesquisas e Estudos Aplicados (IPEA, 2015), grande parte dos óbitos decorrentes dos acidentes de trânsito ocorre nas rodovias brasileiras, especialmente nas rodovias federais. No período considerado neste trabalho (2017 a 2020), a base de dados dos Boletins de Ocorrência do Departamento de Polícia Rodoviária Federal (PRF) possui registro de 289.751 acidentes em rodovias federais, dos quais 18.800 acidentes tiveram vítimas fatais. Na amostra, foram contabilizados 22.1388 mortos e 72.050 feridos graves (PRF, 2021).

Na literatura, os acidentes de trânsito são definidos como eventos aleatórios, independente do desejo do homem e causado por uma força externa (IPEA/DENATRAN/ANTP, 2006). Embora a palavra *acidente* possa “dar a impressão de inevitabilidade e imprevisibilidade – um evento sobre o qual não se tem controle”, não é esse o caso já que os acidentes de trânsito são passíveis de investigação, análise e promoção de ações preventivas (OPAS/OMS-MS, 2012).

Com base na experiência acumulada, incluindo os acidentes que ocorrem em rodovias, os acidentes de trânsito resultam de vários fatores, como o desenvolvimento urbano descontrolado no entorno da rodovia (as chamadas travessias urbanas), condições inadequadas de engenharia de tráfego, comportamento inadequado por parte dos condutores de veículos e dos pedestres, condições inadequadas da frota de veículos e condições meteorológicas desfavoráveis (IPEA/DENATRAN/ANTP, 2008).

Não existe uma caracterização sistemática e completa das situações que podem resultar nos acidentes de trânsito em rodovias federais (IPEA, 2008). Para adequadamente orientar uma política pública de segurança viária, são necessários dados com a tipificação dos acidentes, a localidade da ocorrência e das circunstâncias que ocasionaram o evento do acidente. Conhecendo os tipos de lesões e as formas como foram causadas, pode-se reunir elementos para promover intervenções (OPAS/OMS-MS, 2012). A OMS (2015) classificou o Brasil no grupo de

países que possuem um bom registro de dados sobre óbitos nos acidentes de trânsito.

Este trabalho pretende efetuar uma caracterização dos acidentes de trânsito, registrados na base de dados do Departamento de Polícia Rodoviária Federal (PRF), em trechos de rodovias federais com foco em uma unidade da federação (UF), na série temporal de 2017 a 2020, e efetuar a aplicação de modelos de *machine learning* para a classificação do estado físico dos envolvidos.

1.1. Contextualização

O tema da pesquisa é a segurança viária e está delimitado a partir da caracterização de acidentes de trânsito de uma UF brasileira nos anos de 2017 a 2020. A UF será definida por meio de identificação das extensões de malha rodoviária federal e dos totais de acidentes anuais, bem como do índice de periculosidade ou acidente por quilômetro (acidente/km).

A partir da identificação da UF do estudo, serão gerados conjuntos de dados (ou datasets) com as amostras das ocorrências de acidentes do período em estudo, efetuadas análises dos dados, geração de índices de accidentalidade e a aplicação de modelos de *machine learning* para a classificação do estado físico dos envolvidos.

1.2. O problema proposto

Partindo do problema do quadro alarmante da accidentalidade de trânsito no Brasil, a pesquisa a ser desenvolvida busca responder a seguinte questão:

De acordo com as informações registradas pela PRF na base de dados de acidentes nas rodovias federais nos anos de 2017 a 2020, qual é a extensão das lesões geradas pelos acidentes rodoviários em uma unidade da federação?

Para se chegar no problema proposto, partiu-se do método dos “5W’s”¹ e, para responder a primeira pergunta do método (o “por que?”), foi realizada uma revisão livre de literatura. Foi possível identificar que a segurança viária tem relação estreita com engenharia e comportamento (Elvik, 2003) e que análise da segurança viária envolve aspectos dos elementos que compõem o sistema de trânsito (veículo, via e homem).

A engenharia tem a função de capacitar a tomada de medidas preventivas ou corretivas, com o objetivo principal de diminuir consideravelmente a quantidade de acidentes; por exemplo, implementando controles semafóricos ou melhores concordâncias nos traçados das rodovias. O comportamento, por sua vez, tem como base o fator humano, um dos elementos mais expressivos na análise de acidentes de trânsito por ser causa de 85% a 90% deles no Brasil (Menezes, 2001; Soares et al., 2018).

Segundo Oliveira (2008), apesar de os acidentes de trânsito serem frequentes, do ponto de vista social, podem ser considerados aleatórios no tempo e no espaço, sob o ponto de vista científico. Sendo assim, para sua investigação, é necessário conhecer primeiramente o mecanismo da ocorrência, pelas características e natureza do evento, para posteriormente estudá-lo.

Para entender melhor a ocorrência de um acidente de trânsito, seus fatores e características precisam ser estudados. Coelho (1999) afirma que, para existir um acidente de trânsito, é necessária a ocorrência de uma falha na interação de um ou mais elementos do sistema, que são: usuário, veículo e via.

Múltiplos fatores de risco relacionados às vias e seus usuários, o ambiente e os veículos podem acarretar em acidentes de trânsito, como por exemplo, condições meteorológicas desfavoráveis, desatenção por parte dos pedestres e condições inadequadas da engenharia de tráfego (IPEA/DENATRAN/ANTP, 2008). Dentre eles se destaca a imprudência dos condutores, principalmente no que se refere ao excesso de velocidade. Este fator é quase universalmente considerado como o

¹ O acrônimo do 5W’s é formado a partir de cinco palavras em inglês (why, who, what, when, where) e que significam em português “por que, quem, o que, quando e onde”. Para mais informações, segue o link: <https://its.unl.edu/bestpractices/remember-5-ws>.

principal fator contribuinte, tanto para o número quanto para a gravidade dos acidentes no trânsito (OMS, 2012).

Desta forma, este estudo pretende analisar as informações registradas pela PRF em sua base de dados de acidentes nas rodovias federais nos anos de 2017 a 2020, considerando uma UF que seja representativa para a extensão de malha rodoviária federal assim como tenha registrado alta ocorrência de acidentes, e aplicar modelos de *machine learning* para a classificação do estado físico dos envolvidos.

A escolha do período do estudo foi em decorrência de três fatores: (i) que os dados registrados pela PRF a partir de 2017 já se encontram com informações de latitude e longitude, podendo assim serem georreferenciados; (ii) também em 2017, os dados para a ser agrupados por pessoa para todas as causas e tipos de acidentes; e (iii) para que fosse possível efetuar comparações em períodos anuais completos (de 2017 a 2020).

Em síntese, pelo método do 5W's, a Tabela 1 apresenta a estratificação do problema da pesquisa.

Tabela 1: Estratificação do problema da pesquisa com o método 5W's
(dados da pesquisa)

W	Resposta
Por que?	Quadro alarmante da accidentalidade de trânsito no Brasil
Quem?	Informações registradas pela PRF em sua base de dados de acidentes nas rodovias federais
O que?	A extensão das lesões geradas pelos acidentes rodoviários em uma unidade da federação
Onde?	Unidade da federação brasileira que apresentou a mais alta média de acidentes por quilômetro de rodovia federal
Quando?	2017 a 2020

2. Metodologia

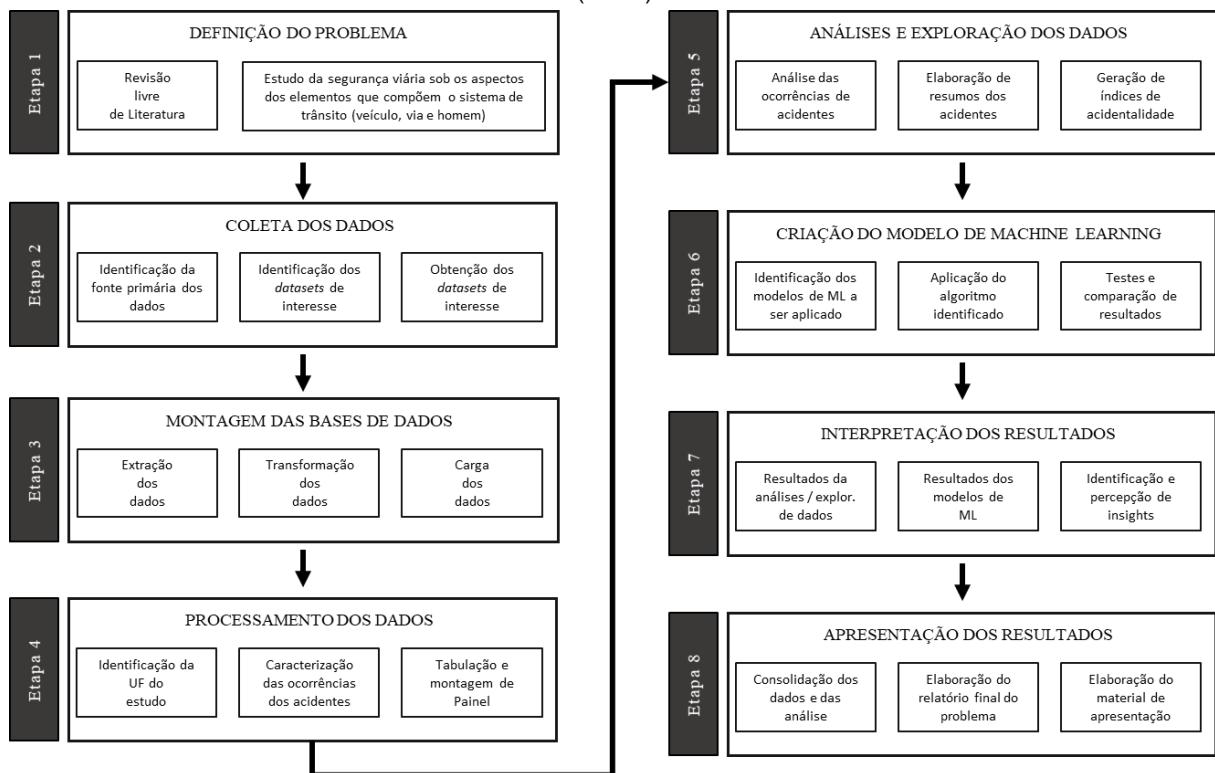
A partir da identificação de uma UF representativa no Brasil quanto à extensão total de sua malha rodoviária federal assim como também dos totais de acidentes, o estudo visa identificar caracterizar ocorrências de acidentes registradas

na base de dados da PRF, entre os anos de 2017 e 2020, e aplicar modelos de *machine learning* para a classificação do estado físico dos envolvidos.

As etapas e procedimentos metodológicos adotados neste estudo são apresentados na Figura 1. A etapa 1 de “Definição do problema” foi exposta no item 1.2 deste trabalho e as demais etapas são tratadas a seguir.

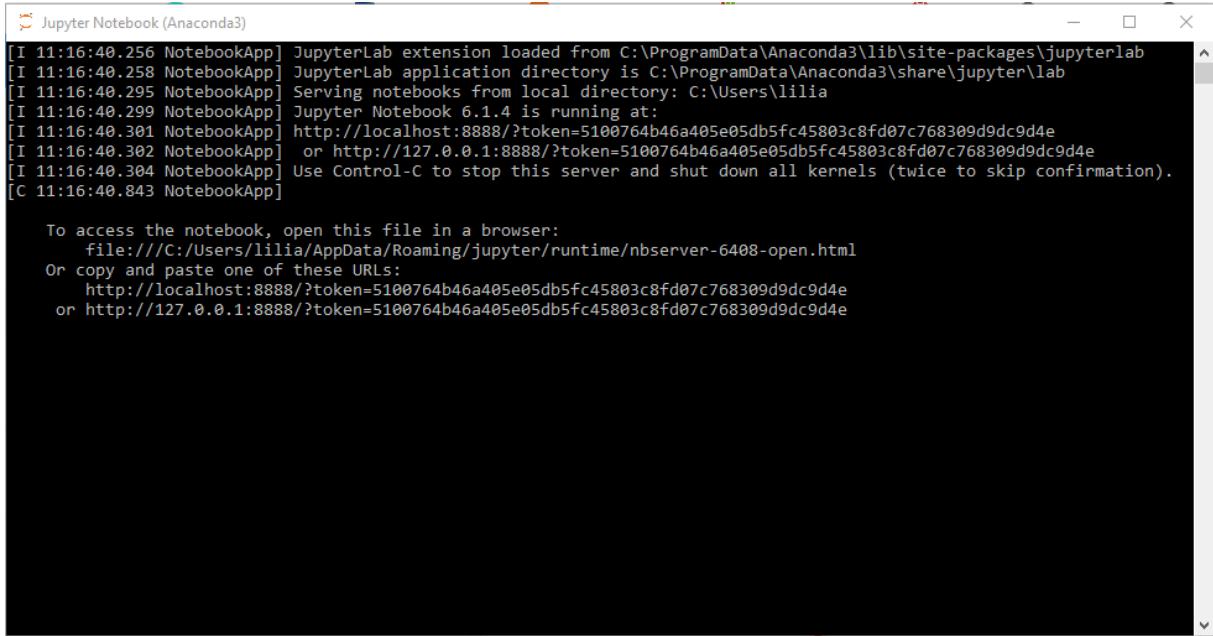
Figura 1: Etapas e procedimentos metodológicos

(autor)



Os procedimentos foram executados utilizando-se: a linguagem de programação Python na versão 3.8.2, pelo Jupyter Notebook (na versão 6.1.4) do Anaconda3 (Figura 2), para a coleta de dados, montagem das bases de dados, processamento dos dados, análise e exploração dos dados, criação do modelo de *machine learning* e interpretação dos resultados; o Microsoft PowerBI Desktop (versão de dezembro de 2020) para o painel analítico; o Microsoft Excel (para determinados suportes); o Microsoft Winword (para o relatório final) e o Microsoft Powerpoint (para o material de apresentação).

**Figura 2: Versão do Jupyter Notebook
(autor)**

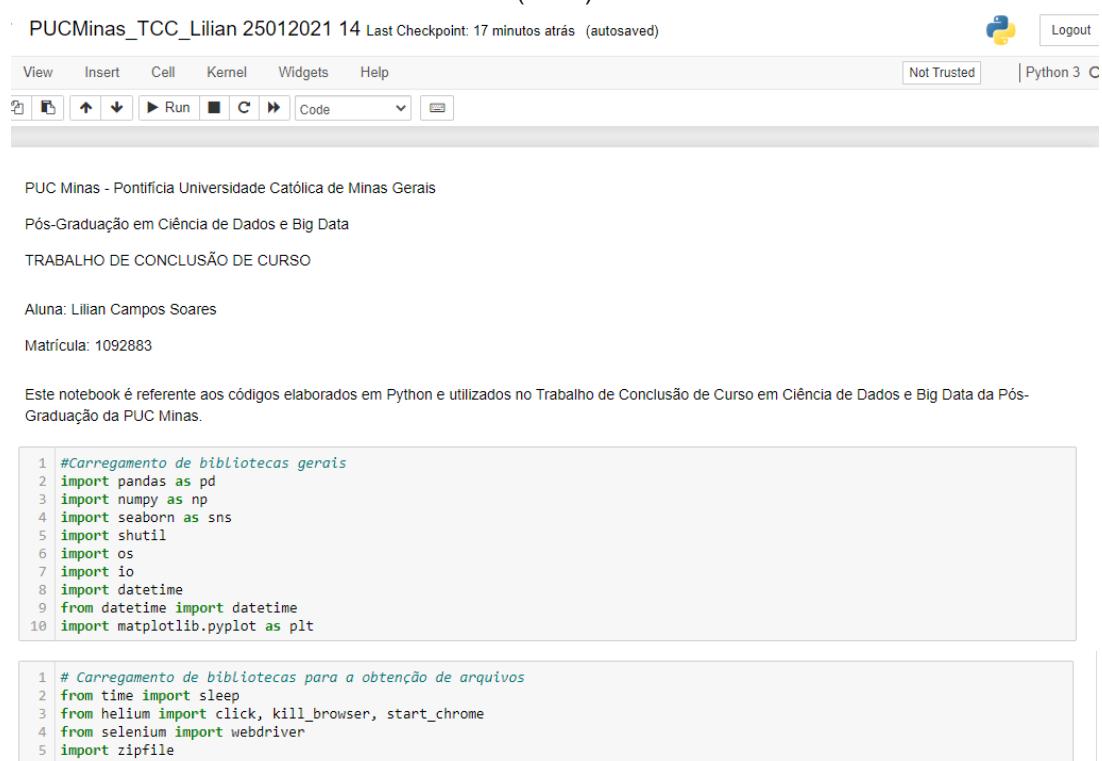


```
[I 11:16:40.256 NotebookApp] JupyterLab extension loaded from C:\ProgramData\Anaconda3\lib\site-packages\jupyterlab
[I 11:16:40.258 NotebookApp] JupyterLab application directory is C:\ProgramData\Anaconda3\share\jupyter\lab
[I 11:16:40.295 NotebookApp] Serving notebooks from local directory: C:\Users\lilia
[I 11:16:40.299 NotebookApp] Jupyter Notebook 6.1.4 is running at:
[I 11:16:40.301 NotebookApp] http://localhost:8888/?token=5100764b46a405e05db5fc45803c8fd07c768309d9dc9d4e
[I 11:16:40.302 NotebookApp] or http://127.0.0.1:8888/?token=5100764b46a405e05db5fc45803c8fd07c768309d9dc9d4e
[I 11:16:40.304 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 11:16:40.843 NotebookApp]

To access the notebook, open this file in a browser:
  file:///C:/Users/lilia/AppData/Roaming/jupyter/runtime/nbserver-6408-open.html
Or copy and paste one of these URLs:
  http://localhost:8888/?token=5100764b46a405e05db5fc45803c8fd07c768309d9dc9d4e
  or http://127.0.0.1:8888/?token=5100764b46a405e05db5fc45803c8fd07c768309d9dc9d4e
```

As bibliotecas usadas para a coleta de dados, montagem das bases de dados, processamento dos dados, e análise e exploração dos dados foram definidas (Figura 3):

**Figura 3: Bibliotecas do Python das etapas 2 a 5
(autor)**



PUCMinas_TCC_Lilian 25012021 14 Last Checkpoint: 17 minutos atrás (autosaved) Logout

View Insert Cell Kernel Widgets Help Not Trusted | Python 3 C

File Run Cell Kernel Help

PUC Minas - Pontifícia Universidade Católica de Minas Gerais
Pós-Graduação em Ciência de Dados e Big Data
TRABALHO DE CONCLUSÃO DE CURSO
Aluna: Lilian Campos Soares
Matrícula: 1092883

Este notebook é referente aos códigos elaborados em Python e utilizados no Trabalho de Conclusão de Curso em Ciência de Dados e Big Data da Pós-Graduação da PUC Minas.

```
1 #Carregamento de bibliotecas gerais
2 import pandas as pd
3 import numpy as np
4 import seaborn as sns
5 import shutil
6 import os
7 import io
8 import datetime
9 from datetime import datetime
10 import matplotlib.pyplot as plt
```

```
1 # Carregamento de bibliotecas para a obtenção de arquivos
2 from time import sleep
3 from helium import click, kill_browser, start_chrome
4 from selenium import webdriver
5 import zipfile
```

```

1 # Instalação e carregamento de bibliotecas para descompactar rar
2 !pip install pyunpack
3 from pyunpack import Archive
4 !pip install patool
5 import patoolib

```

As bibliotecas usadas para a criação do modelo de *machine learning* foram definidas conforme a Figura 4. A etapa 7, de interpretação de dados, usou basicamente as mesmas bibliotecas das etapas anteriores.

Figura 4: Bibliotecas do Python da etapa 6
(autor)

PUC Minas - Pontifícia Universidade Católica de Minas Gerais

Pós-Graduação em Ciência de Dados e Big Data

TRABALHO DE CONCLUSÃO DE CURSO

Aluna: Lilian Campos Soares

Matrícula: 1092883

Este notebook é referente aos códigos elaborados em Python e utilizados no Trabalho de Conclusão de Curso em Ciência de Dados e Big Data da Pós-Graduação da PUC Minas, especificamente para a criação dos modelos de Machine Learning.

```

#Carregamento de bibliotecas
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#Carregamento do pacote com funções estatísticas
import scipy.stats as stats

#Carregamento do pacote Sklearn
import sklearn.metrics as m
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV
from sklearn import tree
from sklearn.model_selection import RandomizedSearchCV

1 #Teste com o algoritmo SGDClassifier
2 from sklearn.linear_model import SGDClassifier

1 #Comparação com o SGDClassifier
2 from sklearn.svm import SVC

#Importando bibliotecas para trabalhar com o algoritmo Decision Tree
from sklearn.tree import DecisionTreeClassifier

1 #Teste com o algoritmo Random Forest Classifier
2 from sklearn.ensemble import RandomForestClassifier

1 #Predição dos dados de teste usando o modelo treinado e apresentação da matriz de confusão
2 from sklearn.metrics import confusion_matrix

1 #Criação de um relatório de classificação nos dados de predição para checar as métricas de acurácia
2 from sklearn.metrics import classification_report

1 #Confusion Matrix
2 import itertools

```

Também foram definidas configurações dos *dataframes* do Pandas e para que os *Plots* da Matplotlib fossem exibidos diretamente no *notebook* (Figuras 5 e 6):

Figura 5: Configurações dos *dataframes* da biblioteca Pandas
(autor)

```

1 # Definição de configurações de dataframes
2 pd.set_option('display.max_rows', 1000)
3 pd.set_option('display.max_columns', 500)
4 pd.set_option('display.width', 1000)
5 pd.set_option('display.precision', 4)
6 pd.set_option('display.expand_frame_repr', False)

```

**Figura 6: Configurações dos *Plots* da biblioteca Matplotlib
(autor)**

```
1 #Configurando os plots para serem exibidos diretamente no notebook
2 %matplotlib inline
```

3. Coleta de dados

Os dados da pesquisa foram coletados das páginas de dados / dados abertos do Departamento Nacional de Infraestrutura Terrestre (DNIT)² e da PRF³, para os dados de extensão de malha rodoviária federal / modelagem do fluxo de veículos e das ocorrências de acidentes registradas nos anos de 2017 a 2020, respectivamente.

Os dados de extensão de malha rodoviária federal se referem às planilhas do Sistema Nacional de Viação (SNV) com dados da situação da malha para os anos de 2017 a 2020 (SNV_201801B.xlsx, snv_201903a.xlsx, SNV_202001A.xlsx e SNV_202101A.xlsx). E, os dados de modelagem de fluxo de veículos se referem ao volume médio diário anual (VMDa) de veículos de 2017 a 2019 (Modelagem 2017_SNV_201801B.xlsx, Modelagem 2018_SNV_201903A.xlsx e Modelagem 2019_SNV_202001A.xlsx). A coleta destes dados se deu conforme Figura 7.

**Figura 7: Coleta dos dados do DNIT
(autor)**

```
1 # Criação do recurso para download dos arquivos de SNV do DNIT
2 # O arquivo "SNV_201801B.xls" se refere aos dados de rodovias de 2017
3 # O arquivo "SNV_201903A.xls" se refere aos dados de rodovias de 2018
4 # O arquivo "SNV_202001A.xls" se refere aos dados de rodovias de 2019
5 # O arquivo "SNV_202101A.xls" se refere aos dados de rodovias de 2020
6 opts = webdriver.ChromeOptions()
7 opts.set_capability("loggingPrefs", {"performance": "ALL"})
8
9 driver = start_chrome("http://servicos.dnit.gov.br/dnitcloud/index.php/s/oTpPRmYs5AAadiNr?path=%2FSNV%20Planilhas%20(2011-Atu
10 sleep(12)
11 click("SNV_201801B.xls")
12 sleep(12)
13 click("SNV_201903A.xls")
14 sleep(12)
15 click("SNV_202001A.xls")
16 sleep(12)
17 click("SNV_202101A.xls")
18 sleep(12)kill_browser()
```

² <https://www.gov.br/dnit/pt-br/assuntos/atlas-e-mapas/pnv-e-snv> e <http://servicos.dnit.gov.br/dadospntc>

³ <https://portal.prf.gov.br/dados-abertos>.

```
1 # Armazenamento dos arquivos SNV no diretório do trabalho e exclusão dos arquivos baixados
2 path = r"C:\Users\lilia\Downloads\TCC PUC\xls"
3 os.mkdir(path)
4 shutil.copyfile(r"C:\Users\lilia\Downloads\SNV_201801B.xls", r"C:\Users\lilia\Downloads\TCC PUC\xls\SNV_201801B.xls")
5 shutil.copyfile(r"C:\Users\lilia\Downloads\SNV_201903A.xls", r"C:\Users\lilia\Downloads\TCC PUC\xls\SNV_201903A.xls")
6 shutil.copyfile(r"C:\Users\lilia\Downloads\SNV_202001A.xls", r"C:\Users\lilia\Downloads\TCC PUC\xls\SNV_202001A.xls")
7 shutil.copyfile(r"C:\Users\lilia\Downloads\SNV_202101A.xls", r"C:\Users\lilia\Downloads\TCC PUC\xls\SNV_202101A.xls")
8 os.remove(r"C:\Users\lilia\Downloads\SNV_201801B.xls")
9 os.remove(r"C:\Users\lilia\Downloads\SNV_201903A.xls")
10 os.remove(r"C:\Users\lilia\Downloads\SNV_202001A.xls")
11 os.remove(r"C:\Users\lilia\Downloads\SNV_202101A.xls")
12 #Como as planilhas possuíam figuras e células mescladas, foi feito um tratamento no excel e geradas tabelas em formato csv d
1
2 # Criação do recurso para download dos arquivos de Estimativa do Volume Médio Diário Anual - VMDA do DNIT (fluxo de veículos
3 # O arquivo "Modelagem 2017_SNV_201801B" se refere aos dados de volume das rodovias de 2017
4 # O arquivo "Modelagem 2018_SNV_201903A" se refere aos dados de volume das rodovias de 2018
5 # O arquivo "Modelagem 2019_SNV_202001A" se refere aos dados de volume das rodovias de 2019 e também será usado para 2020 já
6 opts = webdriver.ChromeOptions()
7 opts.set_capability("loggingPrefs", {"performance": "ALL"})
8
9 driver = start_chrome("http://servicos.dnit.gov.br/dadospntc/Modelagem", options=opts)
10 sleep(12)
11 click("Estimativa VMDA 2019 (Excel)")
12 sleep(12)
13 kill_browser()
14 #Os arquivos de 2017 e 2018 não estavam mais disponíveis on-line na página do DNIT e foram obtidos em consulta por e-mail ao
1
2 # Armazenamento dos arquivos VMDA no diretório do trabalho e exclusão dos arquivos baixados
3 shutil.copyfile(r"C:\Users\lilia\Downloads\Modelagem 2019_SNV_202001A.xlsx", r"C:\Users\lilia\Downloads\TCC PUC\xls\Modelagem 2019_SNV_202001A.xlsx")
4 os.remove(r"C:\Users\lilia\Downloads\Modelagem 2019_SNV_202001A.xlsx")
5 #Os arquivos de 2017 e 2018 não estavam mais disponíveis on-line na página do DNIT e foram obtidos em consulta por e-mail ao
6 #Após a obtenção dos arquivos de 2017 e 2018, os mesmos foram armazenados no diretório de trabalho
```

Os dados de acidentes coletados são referentes aos acidentes em rodovias federais agrupados por ocorrência e agrupados por pessoa, na forma a seguir e a coleta destes dados se deu conforme Figura 8.

- Agrupados por ocorrência, arquivos *datatran2017.zip*, *datatran2018.zip*, *datatran2019.zip*, *datatran2020.zip*; e
 - Agrupados por pessoa, arquivos *acidentes2017.rar*, *acidentes2018.zip*, *acidentes2019.zip*, *acidentes2020.zip*.

Figura 8: Coleta dos dados da PRF
(autor)

```
1 # Criação do recurso para download dos arquivos da PRF de registros de acidentes
2 opts = webdriver.ChromeOptions()
3 opts.set_capability("loggingPrefs", {"performance": "ALL"})
4
5 driver = start_chrome("https://arquivos.prf.gov.br/arquivos/index.php/s/nqvFu7xEF6HhbAq", options=opts)
6 sleep(3)
7 click("Baixar datatran2017.zip")
8 sleep(12)
9
10 driver.get("https://arquivos.prf.gov.br/arquivos/index.php/s/MaC6cieXSFACNWT")
11 sleep(3)
12 click("Baixar datatran2018.zip")
13 sleep(12)
14
15 driver.get("https://arquivos.prf.gov.br/arquivos/index.php/s/kRBuylqz6DyQznN")
16 sleep(3)
17 click("Baixar datatran2019.zip")
18 sleep(12)
19
```

```

20 driver.get("https://arquivos.prf.gov.br/arquivos/index.php/s/jdDLrQIf33xXScE")
21 sleep(3)
22 click("Baixar datatran2020.zip")
23 sleep(12)
24
25 driver.get("https://arquivos.prf.gov.br/arquivos/index.php/s/PYMCvPsVa3ZHspf")
26 sleep(3)
27 click("Baixar acidentes2017.rar")
28 sleep(12)
29
30 driver.get("https://arquivos.prf.gov.br/arquivos/index.php/s/MJRqaDMQ27F60M1")
31 sleep(3)
32 click("Baixar acidentes2018.zip")
33 sleep(12)
34
35 driver.get("https://arquivos.prf.gov.br/arquivos/index.php/s/vw74viLA7WuZI4H")
36 sleep(3)
37 click("Baixar acidentes2019.zip")
38 sleep(12)
39
40 driver.get("https://arquivos.prf.gov.br/arquivos/index.php/s/35cUcYXWsrJd4tF")
41 sleep(3)
42 click("Baixar acidentes2020.zip")
43 sleep(12)
44 kill_browser()

```

```

1 # Descompactação dos arquivos da PRF
2 descompactar = zipfile.ZipFile(r"C:\Users\lilia\Downloads\datatran2017.zip")
3 descompactar.extractall(r"C:\Users\lilia\Downloads")
4 descompactar.close()
5 descompactar = zipfile.ZipFile(r"C:\Users\lilia\Downloads\datatran2018.zip")
6 descompactar.extractall(r"C:\Users\lilia\Downloads")
7 descompactar.close()
8 descompactar = zipfile.ZipFile(r"C:\Users\lilia\Downloads\datatran2019.zip")
9 descompactar.extractall(r"C:\Users\lilia\Downloads")
10 descompactar.close()
11 descompactar = zipfile.ZipFile(r"C:\Users\lilia\Downloads\datatran2020.zip")
12 descompactar.extractall(r"C:\Users\lilia\Downloads")
13 descompactar.close()
14 descompactar = zipfile.ZipFile(r"C:\Users\lilia\Downloads\acidentes2018.zip")
15 descompactar.extractall(r"C:\Users\lilia\Downloads")
16 descompactar.close()
17 descompactar = zipfile.ZipFile(r"C:\Users\lilia\Downloads\acidentes2019.zip")
18 descompactar.extractall(r"C:\Users\lilia\Downloads")
19 descompactar.close()
20 descompactar = zipfile.ZipFile(r"C:\Users\lilia\Downloads\acidentes2020.zip")
21 descompactar.extractall(r"C:\Users\lilia\Downloads")
22 descompactar.close()
23 patoolib.extract_archive(r"C:\Users\lilia\Downloads\acidentes2017.rar", outdir=r"C:\Users\lilia\Downloads")

```

```

1 # Armazenamento dos arquivos CSV no diretório de trabalho e exclusão dos arquivos baixados
2 path = r"C:\Users\lilia\Downloads\TCC PUC\csv"
3 os.mkdir(path)
4 shutil.copyfile(r"C:\Users\lilia\Downloads\datatran2017.csv", r"C:\Users\lilia\Downloads\TCC PUC\csv\datatran2017.csv")
5 shutil.copyfile(r"C:\Users\lilia\Downloads\datatran2018.csv", r"C:\Users\lilia\Downloads\TCC PUC\csv\datatran2018.csv")
6 shutil.copyfile(r"C:\Users\lilia\Downloads\datatran2019.csv", r"C:\Users\lilia\Downloads\TCC PUC\csv\datatran2019.csv")
7 shutil.copyfile(r"C:\Users\lilia\Downloads\datatran2020.csv", r"C:\Users\lilia\Downloads\TCC PUC\csv\datatran2020.csv")
8 shutil.copyfile(r"C:\Users\lilia\Downloads\acidentes2017.csv", r"C:\Users\lilia\Downloads\TCC PUC\csv\acidentes2017.csv")
9 shutil.copyfile(r"C:\Users\lilia\Downloads\acidentes2018.csv", r"C:\Users\lilia\Downloads\TCC PUC\csv\acidentes2018.csv")
10 shutil.copyfile(r"C:\Users\lilia\Downloads\acidentes2019.csv", r"C:\Users\lilia\Downloads\TCC PUC\csv\acidentes2019.csv")
11 shutil.copyfile(r"C:\Users\lilia\Downloads\acidentes2020.csv", r"C:\Users\lilia\Downloads\TCC PUC\csv\acidentes2020.csv")
12 os.remove(r"C:\Users\lilia\Downloads\datatran2017.csv")
13 os.remove(r"C:\Users\lilia\Downloads\datatran2018.csv")
14 os.remove(r"C:\Users\lilia\Downloads\datatran2019.csv")
15 os.remove(r"C:\Users\lilia\Downloads\datatran2020.csv")
16 os.remove(r"C:\Users\lilia\Downloads\acidentes2017.csv")
17 os.remove(r"C:\Users\lilia\Downloads\acidentes2018.csv")
18 os.remove(r"C:\Users\lilia\Downloads\acidentes2019.csv")
19 os.remove(r"C:\Users\lilia\Downloads\acidentes2020.csv")
20 os.remove(r"C:\Users\lilia\Downloads\datatran2017.zip")
21 os.remove(r"C:\Users\lilia\Downloads\datatran2018.zip")
22 os.remove(r"C:\Users\lilia\Downloads\datatran2019.zip")
23 os.remove(r"C:\Users\lilia\Downloads\datatran2020.zip")
24 os.remove(r"C:\Users\lilia\Downloads\acidentes2017.rar")
25 os.remove(r"C:\Users\lilia\Downloads\acidentes2018.zip")
26 os.remove(r"C:\Users\lilia\Downloads\acidentes2019.zip")
27 os.remove(r"C:\Users\lilia\Downloads\acidentes2020.zip")

```

4. Montagem da base de dados

A montagem da base de dados se deu a partir da preparação de dados para identificar a UF que será o objeto do estudo. Desta forma, iniciou-se pelos dados do DNIT de extensão de malha rodoviária federal (SNV) e de fluxo de veículos (VMDa) visando iniciar a construção de *ranking* de dados por UF.

Como as planilhas (SNV_201801B.xlsx, snv_201903a.xlsx, SNV_202001A.xlsx e SNV_202101A.xlsx) possuíam figuras e células mescladas, foi feito um tratamento no MS-Excel e geradas tabelas em formato csv para o "resumo do snv" e do "snv". Procedeu-se com a verificação do resumo do SNV, onde tem dados de malha rodoviária federal consolidados por UF.

O resultado da verificação do resumo do SNV mostrou que a UF de Minas Gerais (MG) é a que possui a maior extensão de rodovias federais em quilômetros em todos os anos (de 2017 a 2020). Ver Figura 9 e Quadro 1.

Figura 9: Verificação do resumo do SNV (dados por UF)
(autor)

```

1 #Leitura dos arquivos resumo do SNV do DNIT
2 resumo_snv_2017 = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\Resumo_2018_SNV201801B.csv", sep=';', decimal=',', enco
3 resumo_snv_2018 = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\Resumo_2019_SNV201903A.csv", sep=';', decimal=',', enco
4 resumo_snv_2019 = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\Resumo_2020_SNV202001A.csv", sep=';', decimal=',', enco
5 resumo_snv_2020 = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\Resumo_2021_SNV202101A.csv", sep=';', decimal=',', enco
6
7 #Verificação das colunas que tem UF e a Extensão total em KM (sem as planejadas) em 2017, 2018, 2019 e 2020
8 resumo_snv_2017.info()
9 resumo_snv_2018.info()
10 resumo_snv_2019.info()
11 resumo_snv_2020.info()
12 print("Extensões de rodovias por UF - 2017")
13 resumo_snv_2017 = resumo_snv_2017.sort_values(by ='TOTAL_Sem_planejada', ascending=False)
14 print(resumo_snv_2017[['UF', 'TOTAL_Sem_planejada']])
15 print("Extensões de rodovias por UF - 2018")
16 resumo_snv_2018 = resumo_snv_2018.sort_values(by ='TOTAL_Sem_planejada', ascending=False)
17 print(resumo_snv_2018[['UF', 'TOTAL_Sem_planejada']])
18 print("Extensões de rodovias por UF - 2019")
19 resumo_snv_2019 = resumo_snv_2019.sort_values(by ='TOTAL_Sem_planejada', ascending=False)
20 print(resumo_snv_2019[['UF', 'TOTAL_Sem_planejada']])
21 print("Extensões de rodovias por UF - 2020")
22 resumo_snv_2020 = resumo_snv_2020.sort_values(by ='TOTAL_Sem_planejada', ascending=False)
23 print(resumo_snv_2020[['UF', 'TOTAL_Sem_planejada']])
24 #O resultado mostra que o estado de MG é o que possui a maior extensão de rodovias federais em KM em todos os anos

```

Quadro 1: Resultado da verificação do resumo do SNV (dados por UF) – Top 05 UFs
(dados da pesquisa)

Extensões de rodovias por UF - 2017

UF	TOTAL_Sem_planejada
10 MG	9577.00
4 BA	7061.00
22 RS	5799.40
13 PA	5122.00
12 MT	5084.10

Extensões de rodovias por UF - 2018

UF	TOTAL_Sem_planejada
10 MG	8859.9

4	BA	7294.0
22	RS	5801.5
13	PA	5122.5
12	MT	5098.3
Extensões de rodovias por UF - 2019		
	UF	TOTAL_Sem_planejada
10	MG	8819.4
4	BA	7315.9
22	RS	5801.5
13	PA	5122.5
12	MT	5098.3
Extensões de rodovias por UF - 2020		
	UF	TOTAL_Sem_planejada
10	MG	8860.8
4	BA	7593.4
22	RS	5800.2
13	PA	5121.5
12	MT	5098.3

Também, como as planilhas (Modelagem 2017_SNV_201801B.xlsx, Modelagem 2018_SNV_201903A.xlsx e Modelagem 2019_SNV_202001A.xlsx) possuíam figuras e células mescladas, foi feito um tratamento no MS-Excel e geradas tabelas em formato csv para o "resumo do vmda". Procedeu-se com a verificação do resumo do VMDa, onde tem dados de fluxo de veículos anual consolidados por UF.

O resultado da verificação do resumo do VMDA mostrou que, embora a UF de MG seja o de maior extensão de rodovias federais em quilômetros, ele está em segundo lugar quanto ao fluxo de veículos (VMDa). Ver Figura 10 e Quadro 2.

Figura 10: Verificação do resumo do VMDA (dados por UF)
(autor)

```

1 #Leitura dos arquivos VMDA do DNIT (como as sheet possuíam figuras e células mescladas, foi feito um tratamento no excel e g
2 resumo_vmda_2017 = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\Modelagem 2017_SNV_201801B_2.csv", sep=';', decimal=','
3 resumo_vmda_2018 = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\Modelagem 2018_SNV_201903A_2.csv", sep=';', decimal=','
4 resumo_vmda_2019_2020 = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\Modelagem 2019_SNV_202001A_2.csv", sep=';', decimal=','
5 print("Volume médio diário anual - todos os segmentos - de rodovias por UF - 2017")
6 resumo_vmda_2017.groupby('UF')[['VMDA_Total']].sum().sort_values(ascending=False)

```

Quadro 2: Resultado da verificação do resumo do VMDA (dados por UF) – Top 05 UF's
(dados da pesquisa)

Volume médio diário anual - todos os segmentos - de rodovias por UF - 2017
SP 6.307.019
MG 5.269.878
RJ 4.069.878
PR 3.290.412
RS 2.609.081
Volume médio diário anual - todos os segmentos - de rodovias por UF - 2018
SP 6.331.202
MG 4.574.966

RJ	3.338.999
PR	3.042.041
RS	2.530.327
Volume médio diário anual - todos os segmentos - de rodovias por UF - 2019	
SP	6.409.354
MG	5.112.368
RJ	3.325.640
PR	3.218.672
SC	3.127.813

Interessante observar que a UF de São Paulo (SP) apresenta cerca de pouco mais de 1.000 quilômetros de extensão rodoviária, enquanto MG tem quase 9.000 quilômetros, contudo tem o mais alto VMDa por UF, com um elevado fluxo de veículos em rodovias federais.

Ainda, antes de se efetivamente efetuar a montagem da base de dados de acidentes do estudo, procedeu-se com a verificação dos arquivos da PRF no formato de *dataframes* e, na sequência, com a identificação dos totais de acidentes por UF para produzir um *ranking* de dados por UF; conforme a seguir:

- Foi efetuada a junção dos arquivos “datatranxxxx.csv” (acidentes agrupados por ocorrência) e “acidentesxxxx.csv” (acidentes agrupados por pessoa), de maneira distinta, criando os arquivos “datatran.csv” e “acidentes.csv” (Figura 11);
- Foi efetuada verificação do arquivo “datatran.csv” (Figura 12), o *shape* do *dataframe* Pandas, o que levou a identificar a quantidade de ocorrências registradas de acidentes (289.751) em nível Brasil em rodovias federais de 2017 a 2020; index, colunas, sumário do *dataframe*; contagem de dados não nulos, de dados únicos e de dados nulos;

Figura 11: Junção dos arquivos da PRF
(autor)

```

1 #Junção dos arquivos datatran.CSV
2 a = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\datatran2017.csv", sep=';', decimal=',', encoding = 'cp1252')
3 b = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\datatran2018.csv", sep=';', decimal=',', encoding = 'cp1252')
4 c = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\datatran2019.csv", sep=';', decimal=',', encoding = 'cp1252')
5 d = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\datatran2020.csv", sep=';', decimal=',', encoding = 'cp1252')
6 abcd = pd.concat([a,b,c,d], join="inner")
7 abcd.to_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\datatran.csv", index=False, sep=';', decimal=',', encoding = 'cp1252')
8 #abc.to_excel(r"C:\Users\lilia\Downloads\TCC PUC\csv\datatran.xlsx", index=False)

```

```

1 #Junção dos arquivos acidentes.CSV
2 e = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\acidentes2017.csv", sep=';', decimal=',', encoding = 'cp1252')
3 f = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\acidentes2018.csv", sep=';', decimal=',', encoding = 'cp1252')
4 g = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\acidentes2019.csv", sep=';', decimal=',', encoding = 'cp1252')
5 h = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\acidentes2020.csv", sep=';', decimal=',', encoding = 'cp1252')

```

```

6 efg = pd.concat([e,f,g,h], join="inner")
7 efg.to_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\acidentes.csv", index=False, sep=',', decimal=',', encoding = 'cp1252')
8 #efg.to_excel(r"C:\Users\lilia\Downloads\TCC PUC\csv\acidentes.xlsx", index=False)

```

**Figura 12: Verificação do arquivo “datatran.csv” da PRF
(autor)**

```

1 #Verificação dos arquivos datatran.CSV: quantidade de linhas e colunas do dataframe
2 datatran_original.shape

```

(289751, 30)

```

1 #Verificação dos arquivos datatran.CSV: descrição do index
2 datatran_original.index

```

RangeIndex(start=0, stop=289751, step=1)

```

1 #Verificação dos arquivos datatran.CSV: colunas presentes
2 datatran_original.columns

```

```

Index(['id', 'data_inversa', 'dia_semana', 'horario', 'uf', 'br', 'km', 'municipio', 'causa_acidente', 'tipo_acidente', 'classificacao_acidente', 'fase_dia', 'sentido_via', 'condicao_metereologica', 'tipo_pista', 'tracado_via', 'uso_solo', 'pessoas', 'mortos', 'feridos_leves', 'feridos_graves', 'ilesos', 'ignorados', 'feridos', 'veiculos', 'latitude', 'longitude', 'regional', 'delegacia', 'uop'], dtype='object')

```

```

1 #Verificação dos arquivos datatran.CSV: tipos e formatos dos dados
2 datatran_original.info()

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 289751 entries, 0 to 289750
Data columns (total 30 columns):
 #   Column            Non-Null Count  Dtype  
--- 
 0   id                289751 non-null   float64
 1   data_inversa      289751 non-null   object 
 2   dia_semana        289751 non-null   object 
 3   horario           289751 non-null   object 
 4   uf                289751 non-null   object 
 5   br                289241 non-null   float64
 6   km                289241 non-null   float64
 7   municipio         289751 non-null   object 
 8   causa_acidente   289751 non-null   object 
 9   tipo_acidente    289751 non-null   object 
 10  classificacao_acidente 289751 non-null   object 
 11  fase_dia          289751 non-null   object 
 12  sentido_via       289751 non-null   object 
 13  condicao_metereologica 289751 non-null   object 
 14  tipo_pista        289751 non-null   object 
 15  tracado_via       289751 non-null   object 
 16  uso_solo          289751 non-null   object 
 17  pessoas           289751 non-null   int64  
 18  mortos            289751 non-null   int64  
 19  feridos_leves    289751 non-null   int64  
 20  feridos_graves   289751 non-null   int64  
 21  ilesos            289751 non-null   int64  
 22  ignorados         289751 non-null   int64  
 23  feridos           289751 non-null   int64  
 24  veiculos          289751 non-null   int64  
 25  latitude          289751 non-null   float64
 26  longitude         289751 non-null   float64
 27  regional          289751 non-null   object 
 28  delegacia         289751 non-null   object 
 29  uop               278661 non-null   object 
dtypes: float64(5), int64(8), object(17)
memory usage: 66.3+ MB

```

```

1 #Verificação dos arquivos datatran.CSV: contagem de dados não nulos
2 datatran_original.count()

```

id	289751
data_inversa	289751
dia_semana	289751
horario	289751
uf	289751
br	289241
km	289241
municipio	289751
causa_acidente	289751
tipo_acidente	289751
classificacao_acidente	289751
fase_dia	289751
sentido_via	289751
condicao_metereologica	289751
tipo_pista	289751
tracado_via	289751
uso_solo	289751
pessoas	289751

```

mortos          289751
feridos_leves   289751
feridos_graves  289751
ilesos          289751
ignorados       289751
feridos          289751
veiculos         289751
latitude         289751
longitude        289751
regional         289751
delegacia        289751
uop              278661
dtype: int64

```

1	#Verificação dos arquivos datatran.CSV: valores únicos
2	datatran_original.nunique()

```

id             289751
data_inversa   1461
dia_semana     7
horario        1436
uf              27
br              128
km              9542
municipio      2004
causa_acidente 52
tipo_acidente   19
classificacao_acidente 3
fase_dia        4
sentido_via     3
condicao_metereologica 10
tipo_pista      3
tracado_via    10
uso_solo        2
pessoas         65
mortos          15
feridos_leves   43
feridos_graves  23
ilesos          57
ignorados       14
feridos          48
veiculos        17
latitude        187576
longitude       188657
regional        28
delegacia       214
uop              126
dtype: int64

```

1	#Verificação dos arquivos datatran.CSV: valores nulos / NaN nas colunas
2	datatran_original.isnull().sum()

```

id             0
data_inversa  0
dia_semana    0
horario        0
uf              0
br              510
km              510
municipio      0
causa_acidente 0
tipo_acidente   0
classificacao_acidente 0
fase_dia        0
sentido_via     0
condicao_metereologica 0
tipo_pista      0
tracado_via    0
uso_solo        0
pessoas         0
mortos          0
feridos_leves   0
feridos_graves  0
ilesos          0
ignorados       0
feridos          0
veiculos        0
latitude         0
longitude        0
regional         0
delegacia        0
uop              11090
dtype: int64

```

- Foi conhecida uma amostra das 10 (dez) primeiras linhas e das 10 (dez) últimas linhas do arquivo “datatran.csv” (Figura 13);

Figura 13: Amostra das 10 (dez) primeiras linhas do arquivo “datatran.csv” da PRF (autor)

1	#Verificação dos arquivos datatran.CSV: amostra das primeiras 10 Linhas													
2	datatran_original.head(10)													
	id	data_inversa	dia_semana	horario	uf	br	km	municipio	causa_acidente	tipo_acidente	classificacao_acidente	fase_dia	sentido_via	co
0	8.0	2017-01-01	domingo	00:00:00	PR	376.0	112.0	PARANAVAI	Fenômenos da Natureza	Queda de ocupante de veículo	Com Vítimas Feridas	Plena Noite	Crescente	
1	9.0	2017-01-01	domingo	00:01:00	SC	101.0	234.0	PALHOCA	Falta de Atenção à Condução	Colisão com objeto estático	Sem Vítimas	Plena Noite	Crescente	
2	11.0	2017-01-01	domingo	00:00:00	PR	153.0	56.9	SANTO ANTONIO DA PLATINA	Animais na Pista	Capotamento	Com Vítimas Feridas	Plena Noite	Decrescente	
3	12.0	2017-01-01	domingo	00:00:00	GO	153.0	435.0	ANAPOLIS	Avarias e/ou desgaste excessivo no pneu	Tombamento	Com Vítimas Feridas	Plena Noite	Decrescente	
4	13.0	2017-01-01	domingo	00:00:00	SC	280.0	77.3	CORUPA	Ingestão de Álcool	Saída de leito carroçável	Com Vítimas Feridas	Plena Noite	Decrescente	
5	14.0	2017-01-01	domingo	00:40:00	GO	60.0	188.0	GUAPO	Falta de Atenção à Condução	Colisão traseira	Com Vítimas Feridas	Plena Noite	Decrescente	
6	15.0	2017-01-01	domingo	00:01:00	PB	104.0	3.4	NOVA FLORESTA	Ingestão de Álcool	Tombamento	Com Vítimas Fatais	Plena Noite	Decrescente	
7	16.0	2017-01-01	domingo	00:30:00	TO	153.0	141.7	ARAGUAINA	Ingestão de Álcool	Colisão traseira	Com Vítimas Feridas	Plena Noite	Decrescente	
8	17.0	2017-01-01	domingo	01:45:00	RS	116.0	34.9	VACARIA	Defeito Mecânico no Veículo	Colisão traseira	Com Vítimas Feridas	Plena Noite	Decrescente	
9	18.0	2017-01-01	domingo	01:40:00	RS	290.0	722.0	URUGUAIANA	Animais na Pista	Atropelamento de Animal	Com Vítimas Feridas	Plena Noite	Crescente	

1	#Verificação dos arquivos datatran.CSV: amostra das últimas 10 Linhas													
2	datatran_original.tail(10)													
	id	data_inversa	dia_semana	horario	uf	br	km	municipio	causa_acidente	tipo_acidente	classificacao_acidente	fase_dia	sent	
289741	334566.0	2020-12-16	quarta-feira	17:40:00	PA	230.0	2.0	PALESTINA DO PARA	Desobediência às normas de trânsito pelo condutor	Colisão com objeto estático	Com Vítimas Feridas	Anoitecer	Crescente	
289742	334658.0	2020-12-19	sábado	03:50:00	AC	364.0	108.0	RIO BRANCO	Defeito na Via	Saída de leito carroçável	Com Vítimas Feridas	Amanhecer	Decrescente	
289743	334662.0	2020-12-18	sexta-feira	03:20:00	SP	116.0	470.0	PARIQUERA-ACU	Avarias e/ou desgaste excessivo no pneu	Incêndio	Sem Vítimas	Plena Noite	Decrescente	
289744	334668.0	2020-12-19	sábado	04:55:00	MG	153.0	7.9	ARAPORA	Objeto estático sobre o leito carroçável	Colisão com objeto estático	Sem Vítimas	Amanhecer	Crescente	
289745	334691.0	2020-11-13	sexta-feira	08:20:00	RO	364.0	409.0	JARU	Falta de Atenção à Condução	Tombamento	Com Vítimas Feridas	Pleno dia	Decrescente	
289746	334713.0	2020-12-23	quarta-feira	13:58:00	PA	316.0	62.0	CASTANHAL	Ingestão de Álcool	Colisão traseira	Com Vítimas Feridas	Pleno dia	Decrescente	
289747	334777.0	2020-12-19	sábado	23:00:00	PA	316.0	106.0	SANTA MARIA DO PARA	Desobediência às normas de trânsito pelo condutor	Colisão frontal	Com Vítimas Feridas	Plena Noite	Crescente	
289748	334780.0	2020-12-08	terça-feira	06:10:00	PR	116.0	90.9	PIRAQUARA	Condutor Dormindo	Colisão traseira	Com Vítimas Feridas	Amanhecer	Crescente	
289749	334788.0	2020-12-31	quinta-feira	22:30:00	SC	101.0	146.9	ITAPEMA	Avarias e/ou desgaste excessivo no pneu	Colisão com objeto	Com Vítimas Fatais	Plena Noite	Crescente	
289750	334959.0	2020-11-18	quarta-feira	18:00:00	AL	104.0	90.3	RIO LARGO	Falta de Atenção do Pedestre	Atropelamento de Pedestre	Com Vítimas Feridas	Plena Noite	Decrescente	

- Foi efetuada verificação do arquivo “acidentes.csv” (Figura 14), o *shape* do *dataframe* Pandas; index, colunas, sumário do *dataframe*; contagem de dados não nulos, de dados únicos e de dados nulos;

**Figura 14: Verificação do arquivo “acidentes.csv” da PRF
(autor)**

```

1 #Verificação dos arquivos acidentes.CSV: quantidade de linhas e colunas do dataframe
2 acidentes_original.shape
(678942, 35)

1 #Verificação dos arquivos acidentes.CSV: descrição do index
2 acidentes_original.index
RangeIndex(start=0, stop=678942, step=1)

1 #Verificação dos arquivos acidentes.CSV: colunas presentes
2 acidentes_original.columns
Index(['id', 'pesid', 'data_inversa', 'dia_semana', 'horario', 'uf', 'br', 'km', 'municipio', 'causa_acidente', 'tipo_acidente', 'classificacao_acidente', 'fase_dia', 'sentido_via', 'condicao_metereologica', 'tipo_pista', 'tracado_via', 'uso_solo', 'id_veiculo', 'tipo_veiculo', 'marca', 'ano_fabricacao_veiculo', 'tipo_envolvido', 'estado_fisico', 'idade', 'sexo', 'ilesos', 'feridos_leves', 'feridos_graves', 'mortos', 'latitude', 'longitude', 'regional', 'delegacia', 'uop'], dtype='object')

1 #Verificação dos arquivos acidentes.CSV: tipos e formatos dos dados
2 acidentes_original.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 678942 entries, 0 to 678941
Data columns (total 35 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   id              678942 non-null   float64
 1   pesid            678938 non-null   float64
 2   data_inversa     678942 non-null   object  
 3   dia_semana      678942 non-null   object  
 4   horario          678942 non-null   object  
 5   uf               678942 non-null   object  
 6   br               677754 non-null   float64
 7   km               677754 non-null   float64
 8   municipio        678942 non-null   object  
 9   causa_acidente  678942 non-null   object  
 10  tipo_acidente   678942 non-null   object  
 11  classificacao_acidente  678942 non-null   object  
 12  fase_dia         678942 non-null   object  
 13  sentido_via      678942 non-null   object  
 14  condicao_metereologica  678942 non-null   object  
 15  tipo_pista       678942 non-null   object  
 16  tracado_via      678942 non-null   object  
 17  uso_solo          678942 non-null   object  
 18  id_veiculo       678938 non-null   float64
 19  tipo_veiculo     678942 non-null   object  
 20  marca             645500 non-null   object  
 21  ano_fabricacao_veiculo  639545 non-null   float64
 22  tipo_envolvido   678942 non-null   object  
 23  estado_fisico    678942 non-null   object  
 24  idade              615749 non-null   float64
 25  sexo               678942 non-null   object  
 26  ilesos            678942 non-null   int64  
 27  feridos_leves    678942 non-null   int64  
 28  feridos_graves   678942 non-null   int64  
 29  mortos            678942 non-null   int64  
 30  latitude          678942 non-null   float64
 31  longitude         678942 non-null   float64
 32  regional          678942 non-null   object  
 33  delegacia         678942 non-null   object  
 34  uop               651109 non-null   object  
dtypes: float64(9), int64(4), object(22)
memory usage: 181.3+ MB

1 #Verificação dos arquivos acidentes.CSV: contagem de dados não-nulos
2 acidentes_original.count()

```

id	678942
pesid	678938
data_inversa	678942
dia_semana	678942
horario	678942
uf	678942
br	677754
km	677754
municipio	678942
causa_acidente	678942
tipo_acidente	678942
classificacao_acidente	678942
fase_dia	678942
sentido_via	678942
condicao_metereologica	678942
tipo_pista	678942
tracado_via	678942
uso_solo	678942
id_veiculo	678938
tipo_veiculo	678942

```

marca          645500
ano_fabricacao_veiculo 639545
tipo_envolvido    678942
estado_fisico     678942
idade            615749
sexo              678942
ilesos           678942
feridos_leves     678942
feridos_graves    678942
mortos           678942
latitude          678942
longitude         678942
regional          678942
delegacia         678942
uop               651109
dtype: int64

```

1	#Verificação dos arquivos acidentes.CSV: valores únicos
2	acidentes_original.nunique()

```

id             289751
pesid          678938
data_inversa    1461
dia_semana      7
horario         1436
uf              27
br              128
km              9542
municipio       2004
causa_acidente  52
tipo_acidente    19
classificacao_acidente 3
fase_dia         4
sentido_via       3
condicao_metereologica 10
tipo_pista        3
tracado_via       10
uso_solo          2
id_veiculo       475203
tipo_veiculo      25
marca            8517
ano_fabricacao_veiculo 70
tipo_envolvido     6
estado_fisico      5
idade            197
sexo              4
ilesos           2
feridos_leves     2
feridos_graves    2
mortos           2
latitude          179243
longitude         179402
regional          28
delegacia         214
uop               126
dtype: int64

```

1	#Verificação dos arquivos acidentes.CSV: valores nulos / NaN nas colunas
2	acidentes_original.isnull().sum()

```

id             0
pesid          4
data_inversa    0
dia_semana      0
horario         0
uf              0
br              1188
km              1188
municipio       0
causa_acidente  0
tipo_acidente    0
classificacao_acidente 0
fase_dia         0
sentido_via       0
condicao_metereologica 0
tipo_pista        0
tracado_via       0
uso_solo          0
id_veiculo       4
tipo_veiculo      0
marca            33442
ano_fabricacao_veiculo 39397
tipo_envolvido     0
estado_fisico      0
idade            63193
sexo              0
ilesos           0
feridos_leves     0
feridos_graves    0
mortos           0
latitude          0

```

```

longitude          0
regional          0
delegacia         0
uop              27833
dtype: int64

```

- Foi conhecida uma amostra das 10 (dez) primeiras linhas e das 10 (dez) últimas linhas do arquivo “datatran.csv” (Figura 15);

Figura 15: Amostra das 10 (dez) primeiras linhas do arquivo “acidentes.csv” da PRF (autor)

#Verificação dos arquivos acidentes.CSV: amostra das primeiras 10 Linhas acidentes_original.head(10)														
	id	pesid	data_inversa	dia_semana	horario	uf	br	km	municipio	causa_acidente	tipo_acidente	classificacao_acidente	fase_dia	sentido_v
0	8.0	1.0	2017-01-01	domingo	00:00:00	PR	376.0	112.0	PARANAVAI	Fenômenos da Natureza	Queda de ocupante de veículo	Com Vítimas Feridas	Plena Noite	Crescen
1	9.0	955.0	2017-01-01	domingo	00:01:00	SC	101.0	234.0	PALHOCA	Falta de Atenção à Condução	Colisão com objeto estático	Sem Vítimas	Plena Noite	Crescen
2	11.0	3.0	2017-01-01	domingo	00:00:00	PR	153.0	56.9	SANTO ANTONIO DA PLATINA	Animais na Pista	Capotamento	Com Vítimas Feridas	Plena Noite	Decrescen
3	11.0	2.0	2017-01-01	domingo	00:00:00	PR	153.0	56.9	SANTO ANTONIO DA PLATINA	Animais na Pista	Capotamento	Com Vítimas Feridas	Plena Noite	Decrescen
4	12.0	1499.0	2017-01-01	domingo	00:00:00	GO	153.0	435.0	ANAPOLIS	Avarias e/ou desgaste excessivo no pneu	Tombamento	Com Vítimas Feridas	Plena Noite	Decrescen
5	13.0	1892.0	2017-01-01	domingo	00:00:00	SC	280.0	77.3	CORUPA	Ingestão de Álcool	Saída de leito carroçável	Com Vítimas Feridas	Plena Noite	Decrescen
6	14.0	1563.0	2017-01-01	domingo	00:40:00	GO	60.0	188.0	GUAPO	Falta de Atenção à Condução	Colisão traseira	Com Vítimas Feridas	Plena Noite	Decrescen
7	14.0	1558.0	2017-01-01	domingo	00:40:00	GO	60.0	188.0	GUAPO	Falta de Atenção à Condução	Colisão traseira	Com Vítimas Feridas	Plena Noite	Decrescen
8	15.0	555.0	2017-01-01	domingo	00:01:00	PB	104.0	3.4	NOVA FLORESTA	Ingestão de Álcool	Tombamento	Com Vítimas Fatais	Plena Noite	Decrescen
9	16.0	990.0	2017-01-01	domingo	00:30:00	TO	153.0	141.7	ARAGUAINA	Ingestão de Álcool	Colisão traseira	Com Vítimas Feridas	Plena Noite	Decrescen

#Verificação dos arquivos acidentes.CSV: amostra das últimas 10 Linhas acidentes_original.tail(10)														
	id	pesid	data_inversa	dia_semana	horario	uf	br	km	municipio	causa_acidente	tipo_acidente	classificacao_acidente	fase_	v
678932	334668.0	745244.0	2020-12-19	sábado	04:55:00	MG	153.0	7.9	ARAPORA	Objeto estático sobre o leito carroçável	Colisão com objeto estático	Sem Vítimas	Amanh	
678933	334713.0	745335.0	2020-12-23	quarta-feira	13:58:00	PA	316.0	62.0	CASTANHAL	Ingestão de Álcool	Colisão traseira	Com Vítimas Feridas	Plen	
678934	334713.0	745336.0	2020-12-23	quarta-feira	13:58:00	PA	316.0	62.0	CASTANHAL	Ingestão de Álcool	Colisão traseira	Com Vítimas Feridas	Plen	
678935	334713.0	745334.0	2020-12-23	quarta-feira	13:58:00	PA	316.0	62.0	CASTANHAL	Ingestão de Álcool	Colisão traseira	Com Vítimas Feridas	Plen	
678936	334777.0	745470.0	2020-12-19	sábado	23:00:00	PA	316.0	106.0	SANTA MARIA DO PARA	Desobediência às normas de trânsito pelo condutor	Colisão frontal	Com Vítimas Feridas	Pl	N
678937	334780.0	745479.0	2020-12-08	terça-feira	06:10:00	PR	116.0	90.9	PIRAQUARA	Condutor Dormindo	Colisão traseira	Com Vítimas Feridas	Amanh	
678938	334780.0	745478.0	2020-12-08	terça-feira	06:10:00	PR	116.0	90.9	PIRAQUARA	Condutor Dormindo	Colisão traseira	Com Vítimas Feridas	Amanh	
678939	334788.0	745498.0	2020-12-31	quinta-feira	22:30:00	SC	101.0	146.9	ITAPEMA	Avarias e/ou desgaste excessivo no pneu	Colisão com objeto	Com Vítimas Fatais	Pl	N
678940	334959.0	745884.0	2020-11-18	quarta-feira	18:00:00	AL	104.0	90.3	RIO LARGO	Falta de Atenção do Pedestre	Atropelamento de Pedestre	Com Vítimas Feridas	Pl	N
678941	334959.0	745883.0	2020-11-18	quarta-feira	18:00:00	AL	104.0	90.3	RIO LARGO	Falta de Atenção do Pedestre	Atropelamento de Pedestre	Com Vítimas Feridas	Pl	N

Feita a verificação dos arquivos da PRF no formato de *dataframes*, procedeu-se com a identificação dos totais de acidentes por UF para que fosse possível definir a UF objeto do estudo; conforme Figura 16 e Quadro 3.

Figura 16: Quantidade de acidentes por ano/UF e do período/UF (autor)

```

1 #Verificação de dados de acidentes por UF: para definição da UF objeto do estudo
2 print("Totais de acidentes por UF - 2017")
3 a = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\datatran2017.csv", sep=';', decimal=',', encoding = 'cp1252')
4 a.groupby('uf')['id'].count().sort_values(ascending=False)
5
6 #MG é a UF que apresentou a maior quantidade de registros de ocorrências de acidentes em 2017

1 print("Totais de acidentes por UF - 2018")
2 b = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\datatran2018.csv", sep=';', decimal=',', encoding = 'cp1252')
3 b.groupby('uf')['id'].count().sort_values(ascending=False)
4
5 #MG é a UF que apresentou a maior quantidade de registros de ocorrências de acidentes em 2018

1 print("Totais de acidentes por UF - 2019")
2 c = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\datatran2019.csv", sep=';', decimal=',', encoding = 'cp1252')
3 c.groupby('uf')['id'].count().sort_values(ascending=False)
4
5 #MG é a UF que apresentou a maior quantidade de registros de ocorrências de acidentes em 2019

1 print("Totais de acidentes por UF - 2020")
2 d = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\datatran2020.csv", sep=';', decimal=',', encoding = 'cp1252')
3 d.groupby('uf')['id'].count().sort_values(ascending=False)
4
5 #MG é a UF que apresentou a maior quantidade de registros de ocorrências de acidentes em 2020

1 #Verificação de dados de acidentes por UF: para definição da UF objeto do estudo
2 abcd = pd.concat([a,b,c,d], join="inner")
3 abcd.to_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\datatran2.csv", index=False)
4 datatran_abcd = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\datatran2.csv", sep=',', decimal='.', encoding = 'utf_8')
5 print("Total de acidentes por UF - 2017 a 2020")
6 datatran_abcd.groupby('uf')['id'].count().sort_values(ascending=False)
7
8 #MG é a UF com a maior quantidade de registros de ocorrências de acidentes em todos os anos
9 #(a contagem neste ponto inclui registros que tem algum campo nulo ('0' / NaN))
10
11 #Conclusão: MG é o de maior extensão rodoviária federal, com a maior quantidade de acidentes por ano e em segundo lugar em v

```

Quadro 3: Resultado da verificação dos acidentes por UF – Top 05 UFs (dados da pesquisa)

Totais de acidentes por UF - 2017	
MG	12.730
PR	10.689
SC	10.665
RS	6.386
SP	6.011
Totais de acidentes por UF - 2018	
MG	9.066
SC	8.497
PR	7.952
RJ	4.575
SP	4.516
Totais de acidentes por UF - 2019	
MG	8.720
SC	8.428
PR	7.708
RJ	4.606
RS	4.595
Totais de acidentes por UF - 2020	
MG	8.363
SC	7.217
PR	7.168
RJ	4.222
RS	4.176

Totais de acidentes por UF - 2017 a 2020	
MG	38.879
SC	34.807
PR	33.517
RS	19.602
RJ	19.372

Para consolidar a identificação dos dados de malha rodoviária federal, de fluxo de veículos (VMDa) bem como da quantidade de acidentes em um *ranking* de dados por UF, foi construída uma tabela resumo por UF e produzidos indicadores de acidentes/km e mortes/km, de acordo com Figura 17 e Quadro 4.

**Figura 17: Indicadores por ano/UF
(autor)**

```

1 #Verificação de dados de acidentes por UF: tabela resumo por UF com indicadores (acidente/km; mortes/km)
2
3 print("Tabela resumo por UF - Indicadores com base em dados de 2017")
4 a.groupby('uf', as_index = False)[['id']].count()
5 acidentes_2017 = pd.DataFrame(a.groupby('uf', as_index = False)[['id']].count()).reset_index()
6 acidentes_2017.rename(columns={'id':'acidentes'}, inplace=True)
7 acidentes_2017.rename(columns={'uf':'UF'}, inplace=True)
8 acidentes_2017.drop(columns=['index'], axis=1, inplace=True)
9
10 a.groupby('uf', as_index = False)[['mortos']].sum()
11 mortes_2017 = pd.DataFrame(a.groupby('uf', as_index = False) [['mortos']].sum()).reset_index()
12 mortes_2017.rename(columns={'mortos':'mortes'}, inplace=True)
13 mortes_2017.rename(columns={'uf':'UF'}, inplace=True)
14 mortes_2017.drop(columns=['index'], axis=1, inplace=True)
15
16 #print(acidentes_2017)
17 #print(mortes_2017)
18 acidentes_2017.drop(columns=['UF'], axis=1, inplace=True)
19 mortes_2017.drop(columns=['UF'], axis=1, inplace=True)
20
21 resumo_snv_2017_2 = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\Resumo_2018_SNV201801B.csv", sep=';', decimal=',', encoding='latin1')
22 resumo_snv_2017_2.drop(columns=['PLANEJADA', 'RNP_TRAVESSIA', 'RNP_LEITO_NATUAL', 'RNP_EM_OBRAS_IMP', 'RNP_IMPLANT', 'RNP_EM_OBRA'])
23 resumo_snv_2017_2[['acidentes_km']] = '0'
24 resumo_snv_2017_2[['mortes_km']] = '0'
25 resumo_snv_2017_2[['acidentes_km']] = resumo_snv_2017_2.acidentes_km.astype('float64')
26 resumo_snv_2017_2[['mortes_km']] = resumo_snv_2017_2.mortes_km.astype('float64')
27 resumo_snv_2017_2.rename(columns={'TOTAL_Sem_planejada':'km_rodovia'}, inplace=True)
28
29 temp = pd.concat([acidentes_2017, resumo_snv_2017_2], axis=1, sort=False).reset_index()
30 resumo_uf_2017 = pd.concat([temp, mortes_2017], axis = 1, sort = False).reset_index()
31
32 resumo_uf_2017[['acidentes_km']] = resumo_uf_2017[['acidentes']] / resumo_uf_2017[['km_rodovia']]
33 resumo_uf_2017[['mortes_km']] = resumo_uf_2017[['mortes']] / resumo_uf_2017[['km_rodovia']]
34 resumo_uf_2017.drop(columns=['level_0'], axis=1, inplace=True)
35 resumo_uf_2017.drop(columns=['index'], axis=1, inplace=True)
36 resumo_uf_2017 = resumo_uf_2017.reindex(columns=['UF', 'km_rodovia', 'acidentes', 'acidentes_km', 'mortes', 'mortes_km'])
37 print(resumo_uf_2017)
38 resumo_uf_2017.to_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\resumo_uf_2017.csv", index=False, sep=';', decimal=',', encoding='latin1')
39
40 #Em 2017, SP apresentou o mais alto índice de acidente/Km, seguido do DF, SC e RJ.

```

```
1 print("Tabela resumo por UF - Indicadores com base em dados de 2018")
2 b.groupby('uf', as_index = False)[['id']].count()
3 acidentes_2018 = pd.DataFrame(b.groupby('uf', as_index = False)[['id']].count()).reset_index()
4 acidentes_2018.rename(columns={'id': 'acidentes'}, inplace=True)
5 acidentes_2018.rename(columns={'uf': 'UF'}, inplace=True)
6 acidentes_2018.drop(columns=['index'], axis=1, inplace=True)
7
8 b.groupby('uf', as_index = False)[['mortos']].sum()
9 mortes_2018 = pd.DataFrame(b.groupby('uf', as_index = False) [['mortos']].sum()).reset_index()
10 mortes_2018.rename(columns={'mortos': 'mortes'}, inplace=True)
11 mortes_2018.rename(columns={'uf': 'UF'}, inplace=True)
12 mortes_2018.drop(columns=['index'], axis=1, inplace=True)
13
14 #print(acidentes_2018)
15 #print(mortes_2018)
```

```

16 accidentes_2018.drop(columns=['UF'], axis=1, inplace=True)
17 mortes_2018.drop(columns=['UF'], axis=1, inplace=True)
18
19 resumo_snv_2018_2 = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\Resumo_2019_SNV201903A.csv", sep=';', decimal=',', en
20 resumo_snv_2018_2.drop(columns=['PLANEJADA','RNP_TRAVESSIA','RNP_LEITO_NATUAL','RNP_EM_OBRAS_IMP','RNP_IMPLANT','RNP_EM_OBRA
21 resumo_snv_2018_2['acidentes_km'] = '0'
22 resumo_snv_2018_2['mortes_km'] = '0'
23 resumo_snv_2018_2['acidentes_km'] = resumo_snv_2018_2.acidentes_km.astype('float64')
24 resumo_snv_2018_2['mortes_km'] = resumo_snv_2018_2.mortes_km.astype('float64')
25 resumo_snv_2018_2.rename(columns={'TOTAL_Sem_planejada':'km_rodovia'}, inplace=True)
26
27 temp = pd.concat([accidentes_2018, resumo_snv_2018_2], axis=1, sort=False).reset_index()
28 resumo_uf_2018 = pd.concat([temp, mortes_2018], axis=1, sort = False).reset_index()
29
30 resumo_uf_2018['acidentes_km'] = resumo_uf_2018['acidentes']/resumo_uf_2018['km_rodovia']
31 resumo_uf_2018['mortes_km'] = resumo_uf_2018['mortes']/resumo_uf_2018['km_rodovia']
32 resumo_uf_2018.drop(columns=['level_0'], axis=1, inplace=True)
33 resumo_uf_2018.drop(columns=['index'], axis=1, inplace=True)
34 resumo_uf_2018 = resumo_uf_2018.reindex(columns=['UF', 'km_rodovia', 'acidentes', 'acidentes_km', 'mortes', 'mortes_km'])
35 print(resumo_uf_2018)
36 resumo_uf_2018.to_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\resumo_uf_2018.csv", index=False, sep=';', decimal=',', encoding
37
38 #Em 2018, DF apresentou o mais alto índice de acidente/Km, seguido do SP, SC e RJ.
40
1 print("Tabela resumo por UF - Indicadores com base em dados de 2019")
2 c.groupby('uf', as_index = False)[['id']].count()
3 accidentes_2019 = pd.DataFrame(c.groupby('uf', as_index = False)[['id']].count()).reset_index()
4 accidentes_2019.rename(columns={'id':'acidentes'}, inplace=True)
5 accidentes_2019.rename(columns={'uf':'UF'}, inplace=True)
6 accidentes_2019.drop(columns=['index'], axis=1, inplace=True)
7
8 c.groupby('uf', as_index = False)[['mortos']].sum()
9 mortes_2019 = pd.DataFrame(c.groupby('uf', as_index = False) ['mortos'].sum()).reset_index()
10 mortes_2019.rename(columns={'mortos':'mortes'}, inplace=True)
11 mortes_2019.rename(columns={'uf':'UF'}, inplace=True)
12 mortes_2019.drop(columns=['index'], axis=1, inplace=True)
13
14 #print(accidentes_2019)
15 #print(mortes_2019)
16 accidentes_2019.drop(columns=['UF'], axis=1, inplace=True)
17 mortes_2019.drop(columns=['UF'], axis=1, inplace=True)
18
19 resumo_snv_2019_2 = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\Resumo_2020_SNV202001A.csv", sep=';', decimal=',', en
20 resumo_snv_2019_2.drop(columns=['PLANEJADA','RNP_TRAVESSIA','RNP_LEITO_NATUAL','RNP_EM_OBRAS_IMP','RNP_IMPLANT','RNP_EM_OBRA
21 resumo_snv_2019_2['acidentes_km'] = '0'
22 resumo_snv_2019_2['mortes_km'] = '0'
23 resumo_snv_2019_2['acidentes_km'] = resumo_snv_2019_2.acidentes_km.astype('float64')
24 resumo_snv_2019_2['mortes_km'] = resumo_snv_2019_2.mortes_km.astype('float64')
25 resumo_snv_2019_2.rename(columns={'TOTAL_Sem_planejada':'km_rodovia'}, inplace=True)
26
27 temp = pd.concat([accidentes_2019, resumo_snv_2019_2], axis=1, sort=False).reset_index()
28 resumo_uf_2019 = pd.concat([temp, mortes_2019], axis=1, sort = False).reset_index()
29
30 resumo_uf_2019['acidentes_km'] = resumo_uf_2019['acidentes']/resumo_uf_2019['km_rodovia']
31 resumo_uf_2019['mortes_km'] = resumo_uf_2019['mortes']/resumo_uf_2019['km_rodovia']
32 resumo_uf_2019.drop(columns=['level_0'], axis=1, inplace=True)
33 resumo_uf_2019.drop(columns=['index'], axis=1, inplace=True)
34 resumo_uf_2019 = resumo_uf_2019.reindex(columns=['UF', 'km_rodovia', 'acidentes', 'acidentes_km', 'mortes', 'mortes_km'])
35 print(resumo_uf_2019)
36 resumo_uf_2019.to_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\resumo_uf_2019.csv", index=False, sep=';', decimal=',', encoding
37
38 #Em 2019, DF apresentou o mais alto índice de acidente/Km, seguido do SP, SC e ES.
40
1 print("Tabela resumo por UF - Indicadores com base em dados de 2020")
2 d.groupby('uf', as_index = False)[['id']].count()
3 accidentes_2020 = pd.DataFrame(d.groupby('uf', as_index = False)[['id']].count()).reset_index()
4 accidentes_2020.rename(columns={'id':'acidentes'}, inplace=True)
5 accidentes_2020.rename(columns={'uf':'UF'}, inplace=True)
6 accidentes_2020.drop(columns=['index'], axis=1, inplace=True)
7
8 d.groupby('uf', as_index = False)[['mortos']].sum()
9 mortes_2020 = pd.DataFrame(d.groupby('uf', as_index = False) ['mortos'].sum()).reset_index()
10 mortes_2020.rename(columns={'mortos':'mortes'}, inplace=True)
11 mortes_2020.rename(columns={'uf':'UF'}, inplace=True)
12 mortes_2020.drop(columns=['index'], axis=1, inplace=True)
13
14 #print(accidentes_2020)
15 #print(mortes_2020)
16 accidentes_2020.drop(columns=['UF'], axis=1, inplace=True)
17 mortes_2020.drop(columns=['UF'], axis=1, inplace=True)
18
19 resumo_snv_2020_2 = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\Resumo_2021_SNV202101A.csv", sep=';', decimal=',', en
20 resumo_snv_2020_2.drop(columns=['PLANEJADA','RNP_TRAVESSIA','RNP_LEITO_NATUAL','RNP_EM_OBRAS_IMP','RNP_IMPLANT','RNP_EM_OBRA
21 resumo_snv_2020_2['acidentes_km'] = '0'
22 resumo_snv_2020_2['mortes_km'] = '0'

```

```

23 resumo_snv_2020_2['acidentes_km'] = resumo_snv_2020_2.acidentes_km.astype('float64')
24 resumo_snv_2020_2['mortes_km'] = resumo_snv_2020_2.mortes_km.astype('float64')
25 resumo_snv_2020_2.rename(columns={'TOTAL_Sem_planejada':'km_rodovia'}, inplace=True)
26
27 temp = pd.concat([acidentes_2020, resumo_snv_2020_2], axis=1, sort=False).reset_index()
28 resumo_uf_2020 = pd.concat([temp, mortes_2020], axis =1, sort = False).reset_index()
29
30 resumo_uf_2020['acidentes_km'] = resumo_uf_2019['acidentes']/resumo_uf_2020['km_rodovia']
31 resumo_uf_2020['mortes_km'] = resumo_uf_2020['mortes']/resumo_uf_2020['km_rodovia']
32 resumo_uf_2020.drop(columns=['level_0'], axis=1, inplace=True)
33 resumo_uf_2020.drop(columns=['index'], axis=1, inplace=True)
34 resumo_uf_2020 = resumo_uf_2020.reindex(columns=['UF', 'km_rodovia', 'acidentes', 'acidentes_km', 'mortes', 'mortes_km'])
35 print(resumo_uf_2020)
36 resumo_uf_2020.to_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\resumo_uf_2020.csv", index=False, sep=';', decimal=',', encoding='utf-8')
37
38 #Em 2020, DF apresentou o mais alto índice de acidente/Km, seguido do SP, SC e ES.

```

**Quadro 4: Indicadores por ano/UF – Top 05 UFs por quantidade de acidentes
(dados da pesquisa)**

Tabela resumo por UF - Indicadores com base em dados de 2017

UF	km_rodovia	acidentes	acidentes_km	mortes	mortes_km
MG	9577.00	12730	1.3292	869	0.0907
PR	4053.40	10689	2.6370	612	0.1510
SC	2352.10	10665	4.5342	381	0.1620
RS	5799.40	6386	1.1011	391	0.0674
SP	1122.30	6011	5.3560	255	0.2272

Tabela resumo por UF - Indicadores com base em dados de 2018

UF	km_rodovia	acidentes	acidentes_km	mortes	mortes_km
MG	8859.9	9066	1.0233	693	0.0782
SC	2375.6	8497	3.5768	386	0.1625
PR	3914.1	7952	2.0316	494	0.1262
RJ	1707.3	4575	2.6797	276	0.1617
SP	1122.4	4516	4.0235	207	0.1844

Tabela resumo por UF - Indicadores com base em dados de 2019

UF	km_rodovia	acidentes	acidentes_km	mortes	mortes_km
MG	8819.4	8720	0.9887	678	0.0769
SC	2375.6	8428	3.5477	403	0.1696
PR	3913.9	7708	1.9694	500	0.1277
RJ	1707.3	4606	2.6978	353	0.2068
SP	1122.4	4377	3.8997	204	0.1818

Tabela resumo por UF - Indicadores com base em dados de 2020

UF	km_rodovia	acidentes	acidentes_km	mortes	mortes_km
MG	8860.8	8363	0.9841	717	0.0809
SC	2383.8	7217	3.5355	380	0.1594
PR	3950.5	7168	1.9511	526	0.1331
RJ	1713.0	4222	2.6888	272	0.1588
SP	1122.4	4040	3.8997	240	0.2138

Com a verificação dos dados de malha rodoviária federal, de fluxo de veículos (VMDa) bem como da quantidade de acidentes e de mortes por UF, decidiu-se pela definição de MG para o estudo: (i) MG é a UF de maior extensão rodoviária federal; (ii) MG apresentou a maior quantidade de acidentes por ano e do total do período; (iii) MG apresentou a maior quantidade de mortes por ano e do total por período.

Alguns pontos são importantes de mencionar: (i) a UF de MG está em segundo lugar em fluxo de veículos – a UF de SP, embora tenha uma extensão de malha rodoviária federal baixa, tem o maior fluxo de veículos; (ii) quando comparado

a outras UFs, a UF de MG não apresentou indicadores altos de acidente/km (periculosidade) e mortes/km.

A partir deste ponto, se deu início efetivamente à montagem da base de dados com as ocorrências de acidentes da UF de MG no período de 2017 a 2020 para o estudo, conforme Figuras 18 e 19. Todo o tratamento e enriquecimento dos dados da base serão apresentados na próxima seção que trata do processamento dos dados.

Figura 18: Montagem do arquivo “datatran_mg”
(autor)

```

1 #Montagem da base de dados datatran.CSV - com a definição da UF de MG
2 datatran_mg = datatran_original[datatran_original.uf == 'MG']
3 datatran_mg.reset_index(inplace=True)
4 datatran_mg.drop(columns=['index'], axis=1, inplace=True)

1 #Montagem da base de dados datatran.CSV - com a definição da UF de MG: conferindo as alterações
2 datatran_mg.head()

```

	id	data_inversa	dia_semana	horario	uf	br	km	municipio	causa_acidente	tipo_acidente	classificacao_acidente	fase_dia	sentido_via	co
0	47.0	2017-01-01	domingo	04:50:00	MG	381.0	605.2	OLIVEIRA	Condutor Dormindo	Saída de leito carroçável	Com Vítimas Feridas	Amanhecer	Decrescente	
1	52.0	2017-01-01	domingo	05:00:00	MG	262.0	368.0	JUATUBA	Velocidade Incompatível	Colisão com objeto estático	Com Vítimas Feridas	Plena Noite	Crescente	
2	53.0	2017-01-01	domingo	05:00:00	MG	459.0	14.0	CALDAS	Falta de Atenção à Condução	Saída de leito carroçável	Sem Vítimas	Amanhecer	Crescente	
3	61.0	2017-01-01	domingo	05:00:00	MG	262.0	186.0	SAO DOMINGOS DO PRATA	Condutor Dormindo	Saída de leito carroçável	Com Vítimas Feridas	Plena Noite	Crescente	
4	63.0	2017-01-01	domingo	06:00:00	MG	135.0	408.0	BOCAIUVA	Animais na Pista	Atropelamento de Animal	Com Vítimas Feridas	Amanhecer	Decrescente	

```

1 #Montagem da base de dados datatran.CSV - exportando para CSV
2 datatran_mg.to_csv("C:\Users\lilia\Downloads\TCC PUC\csv\datatran_mg.csv", index = False, header = True, sep=';', decimal=',')

```

Figura 19: Montagem do arquivo “acidentes_mg”
(autor)

```

1 #Montagem da base de dados acidentes.CSV - com a definição da UF de MG
2 acidentes_mg = acidentes_original[acidentes_original.uf == 'MG']
3 acidentes_mg.reset_index(inplace=True)
4 acidentes_mg.drop(columns=['index'], axis=1, inplace=True)

1 #Montagem da base de dados acidentes.CSV - com a definição da UF de MG: conferindo as alterações
2 acidentes_mg.head()

```

	id	pesid	data_inversa	dia_semana	horario	uf	br	km	municipio	causa_acidente	tipo_acidente	classificacao_acidente	fase_dia	sentido_v	
0	47.0	44.0	2017-01-01	domingo	04:50:00	MG	381.0	605.2	OLIVEIRA	Condutor Dormindo	Saída de leito carroçável	Com Vítimas Feridas	Amanhecer	Decrescen	
1	47.0	45.0	2017-01-01	domingo	04:50:00	MG	381.0	605.2	OLIVEIRA	Condutor Dormindo	Saída de leito carroçável	Com Vítimas Feridas	Amanhecer	Decrescen	
2	52.0	1528.0	2017-01-01	domingo	05:00:00	MG	262.0	368.0	JUATUBA	Velocidade Incompatível	Colisão com objeto estático	Com Vítimas Feridas	Plena Noite	Crescen	
3	52.0	1526.0	2017-01-01	domingo	05:00:00	MG	262.0	368.0	JUATUBA	Velocidade Incompatível	Colisão com objeto estático	Com Vítimas Feridas	Plena Noite	Crescen	
4	52.0	1519.0	2017-01-01	domingo	05:00:00	MG	262.0	368.0	JUATUBA	Velocidade Incompatível	Colisão com objeto estático	Com Vítimas Feridas	Plena Noite	Crescen	

```

1 #Montagem da base de dados acidentes.CSV - exportando para CSV
2 acidentes_mg.to_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\acidentes_mg.csv", index = False, header = True, sep=';', decimal

```

A Tabela 2 apresenta a descrição dos campos/colunas de cada um dos datasets da PRF e que integram uma amostra de dados para a UF de MG.

Tabela 1: Descrição dos campos/colunas dos datasets
(PRF, 2021)

Dataset	Acidentes “Agrupados por ocorrência”	Arquivos “datatran-xxxx”
Nome da coluna / campo	Descrição	Tipo
<i>id</i>	Variável com valores numéricos, representando o identificador do acidente.	Numérico
<i>data_inversa</i>	Data da ocorrência no formato dd/mm/aaaa.	Data
<i>dia_semana</i>	Dia da semana da ocorrência. Ex.: Segunda, Terça, etc.	Data
<i>horario</i>	Horário da ocorrência no formato hh:mm:ss.	Hora
<i>uf</i>	Unidade da Federação. Ex.: MG, PE, DF, etc.	Texto
<i>br</i>	Variável com valores numéricos, representando o identificador da BR do acidente.	Texto
<i>km</i>	Identificação do quilômetro onde ocorreu o acidente, com valor mínimo de 0,1 km e com a casa decimal separada por ponto.	Numérico
<i>municipio</i>	Nome do município de ocorrência do acidente	Texto
<i>causa_acidente</i>	Identificação da causa principal do acidente. Neste conjunto de dados são excluídos os acidentes com a variável causa principal igual a “Não”.	Texto
<i>tipo_acidente</i>	Identificação do tipo de acidente. Ex.: Colisão frontal, Saída de pista, etc. Neste conjunto de dados são excluídos os tipos de acidentes com ordem maior ou igual a dois. A ordem do acidente demonstra a sequência cronológica dos tipos presentes na mesma ocorrência.	Texto
<i>classificação_acidente</i>	Classificação quanto à gravidade do acidente: Sem Vítimas, Com Vítimas Feridas, Com Vítimas Fatais e Ignorado.	Texto
<i>fase_dia</i>	Fase do dia no momento do acidente. Ex. Amanhecer, Pleno dia, etc.	Texto
<i>sentido_via</i>	Sentido da via considerando o ponto de colisão: Crescente e decrescente.	Texto
<i>condição_meteorologica</i>	Condição meteorológica no momento do acidente: Céu claro, chuva, vento, etc.	Texto
<i>tipo_pista</i>	Tipo da pista considerando a quantidade de faixas: Dupla, simples ou múltipla.	Texto
<i>tracado_via</i>	Descrição do traçado da via.	Texto
<i>uso_solo</i>	Descrição sobre as características do local do acidente: Urbano=Sim; Rural=Não.	Texto
<i>latitude</i>	Latitude do local do acidente em formato geodésico decimal.	Geográfico
<i>longitude</i>	Longitude do local do acidente em formato geodésico decimal.	Geográfico
<i>pessoas</i>	Total de pessoas envolvidas na ocorrência.	Numérico
<i>mortos</i>	Total de pessoas mortas envolvidas na ocorrência.	Numérico
<i>feridos_leves</i>	Total de pessoas com ferimentos leves envolvidas na ocorrência.	Numérico
<i>feridos_graves</i>	Total de pessoas com ferimentos graves envolvidas na ocorrência.	Numérico
<i>feridos</i>	Total de pessoas feridas envolvidas na ocorrência (é a soma dos feridos leves com os graves).	Numérico

<i>ilesos</i>	Total de pessoas ilesas envolvidas na ocorrência.	Numérico
<i>ignorados</i>	Total de pessoas envolvidas na ocorrência e que não se soube o estado físico.	Numérico
<i>veiculos</i>	Total de veículos envolvidos na ocorrência	Numérico

Dataset	Acidentes “Agrupados por pessoa”	Arquivos “acidentes-xxxx”
Nome da coluna / campo	Descrição	Tipo
<i>id</i>	Variável com valores numéricos, representando o identificador do acidente.	Numérico
<i>pesid</i>	Variável com valores numéricos, representando o identificador da pessoa envolvida.	Numérico
<i>data_inversa</i>	Data da ocorrência no formato dd/mm/aaaa.	Data
<i>dia_semana</i>	Dia da semana da ocorrência. Ex.: Segunda, Terça, etc.	Data
<i>horario</i>	Horário da ocorrência no formato hh:mm:ss.	Hora
<i>uf</i>	Unidade da Federação. Ex.: MG, PE, DF, etc.	Texto
<i>br</i>	Variável com valores numéricos, representando o identificador da BR do acidente.	Texto
<i>km</i>	Identificação do quilômetro onde ocorreu o acidente, com valor mínimo de 0,1 km e com a casa decimal separada por ponto.	Numérico
<i>municipio</i>	Nome do município de ocorrência do acidente	Texto
<i>causa_acidente</i>	Identificação da causa principal do acidente. Neste conjunto de dados são excluídos os acidentes com a variável causa principal igual a “Não”.	Texto
<i>tipo_acidente</i>	Identificação do tipo de acidente. Ex.: Colisão frontal, Saída de pista, etc. Neste conjunto de dados são excluídos os tipos de acidentes com ordem maior ou igual a dois. A ordem do acidente demonstra a sequência cronológica dos tipos presentes na mesma ocorrência.	Texto
<i>classificacao_acidente</i>	Classificação quanto à gravidade do acidente: Sem Vítimas, Com Vítimas Feridas, Com Vítimas Fatais e Ignorado.	Texto
<i>fase_dia</i>	Fase do dia no momento do acidente. Ex. Amanhecer, Pleno dia, etc.	Texto
<i>sentido_via</i>	Sentido da via considerando o ponto de colisão: Crescente e decrescente.	Texto
<i>condicao_meteorologica</i>	Condição meteorológica no momento do acidente: Céu claro, chuva, vento, etc.	Texto
<i>tipo_pista</i>	Tipo da pista considerando a quantidade de faixas: Dupla, simples ou múltipla.	Texto
<i>tracado_via</i>	Descrição do traçado da via.	Texto
<i>uso_solo</i>	Descrição sobre as características do local do acidente: Urbano=Sim; Rural=Não.	Texto
<i>id_veiculo</i>	Variável com valores numéricos, representando o identificador do veículo envolvido.	Numérico
<i>tipo_veiculo</i>	Tipo do veículo conforme Art. 96 do Código de Trânsito Brasileiro. Ex.: Automóvel, Caminhão, Motocicleta, etc.	Texto
<i>marca</i>	Descrição da marca do veículo.	Texto
<i>ano_fabricacao_veiculo</i>	Ano de fabricação do veículo, formato aaaa	Data
<i>tipo_envolvido</i>	Tipo de envolvido no acidente conforme sua participação no evento. Ex.: condutor, passageiro, pedestre, etc	Texto
<i>estado_fisico</i>	Condição do envolvido conforme a gravidade das lesões. Ex.: morto, ferido leve, etc.	Texto
<i>idade</i>	Idade do envolvido. O código “-1” indica que não foi possível coletar tal informação.	Numérico
<i>sexo</i>	Sexo do envolvido. O valor “inválido” indica que não foi possível coletar tal informação.	Texto
<i>ilesos</i>	Valor binário que identifica se o envolvido foi classificado como ileso.	Numérico
<i>feridos_leves</i>	Valor binário que identifica se o envolvido foi classificado	Numérico

	como ferido leve.	
<i>feridos_graves</i>	Valor binário que identifica se o envolvido foi classificado como ferido grave.	Numérico
<i>mortos</i>	Valor binário que identifica se o envolvido foi classificado como morto.	Numérico
<i>latitude</i>	Latitude do local do acidente em formato geodésico decimal.	Geográfico
<i>longitude</i>	Longitude do local do acidente em formato geodésico decimal.	Geográfico

5. Processamento dos dados

Com as bases de “datatran” e “acidentes” montadas para a UF de MG, passou-se aos procedimentos de processamento dos dados, com tratamento e enriquecimento dos dados da base.

O processamento inicial de tratamento de valores em branco e de valores nulos foi efetuado para ambas as bases (datatran e acidentes) – Figuras 20 e 21.

- A base “datatran” apresentava valores nulos / NaN nas colunas de ‘br’, ‘km’, e ‘uop’. Por algum motivo no registro da ocorrência de acidente pelos Policiais Rodoviários Federais, os atributos das colunas de ‘br’, ‘km’, e ‘uop’ não foram preenchidos. Decidiu-se por efetuar o “drop” destes registros, uma vez que os acidentes serão hierarquizados por ‘br’ e ‘km’, bem como o atributo de ‘uop’ não será considerado no estudo;
- A base “acidentes” apresentava valores nulos / NaN nas colunas de ‘br’, ‘km’, ‘marca’, ‘ano_fabricacao_veiculo’, ‘idade’ e ‘uop’. Por algum motivo no registro da ocorrência de acidente pelos Policiais Rodoviários Federais, os atributos das colunas de ‘br’, ‘km’, ‘marca’, ‘ano_fabricacao_veiculo’, ‘idade’ e ‘uop’ não foram preenchidos. Decidiu-se por efetuar o “drop” destes registros, uma vez que os acidentes serão hierarquizados por ‘br’ e ‘km’, bem como os atributos de ‘marca’, ‘ano_fabricacao_veiculo’, ‘idade’ e ‘uop’ não serão considerados no estudo.

Figura 20: Processamento e tratamento do arquivo “datatran_mg” (autor)

```

1 #Tratamento dos arquivos datatran.CSV - valores em branco para '0'
2 datatran_mg = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\datatran_mg.csv", sep=';', decimal=',', encoding = 'cp1252'
3 datatran_mg.fillna(0, inplace=True)
4 #datatran_mg.info()

1 #Tratamento dos arquivos datatran.CSV - remoção de registros em que BR está como '0' do arquivo datatran.CSV
2 datatran_mg = datatran_mg[datatran_mg.br !=0]
3 #datatran_mg.info()

1 #Tratamento dos arquivos datatran.CSV - verificação se ainda existem valores nulos / NaN nas colunas
2 datatran_mg.isnull().sum()

id               0
data_inversa     0
dia_semana       0
horario          0
uf               0
br               0
km               0
municipio        0
causa_acidente   0
tipo_acidente    0
classificacao_acidente  0
fase_dia         0
sentido_via      0
condicao_metereologica 0
tipo_pista       0
tracado_via     0
uso_solo         0
pessoas          0
mortos           0
feridos_leves    0
feridos_graves   0
ilesos           0
ignorados        0
feridos          0
veiculos         0
latitude         0
longitude        0
regional         0
delegacia        0
uop              0
dtype: int64

```

Figura 21 Processamento e tratamento do arquivo “accidentes_mg” (autor)

```

1 #Tratamento dos arquivos accidentes.CSV - valores em branco para '0'
2 accidentes_mg = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\accidentes_mg.csv", sep=';', decimal=',', encoding = 'cp1252'
3 accidentes_mg.fillna(0, inplace=True)
4 #accidentes_mg.info()

1 #Tratamento dos arquivos accidentes.CSV - remoção de registros em que BR está como '0' do arquivo accidentes.CSV
2 accidentes_mg = accidentes_mg[accidentes_mg.br !=0]
3 #accidentes_mg.info()

1 #Tratamento dos arquivos accidentes.CSV - verificação se ainda existem valores nulos / NaN nas colunas
2 accidentes_mg.isnull().sum()

id               0
pesid            0
data_inversa     0
dia_semana       0
horario          0
uf               0
br               0
km               0
municipio        0
causa_acidente   0
tipo_acidente    0
classificacao_acidente  0
fase_dia         0
condicao_metereologica 0
tipo_pista       0
tracado_via     0
uso_solo         0
tipo_veiculo    0
tipo_envolvido   0
estado_fisico    0
sexo              0

```

```

ilesos          0
feridos_leves   0
feridos_graves  0
mortos          0
latitude         0
longitude        0
dtype: int64

```

No arquivo “datatran”, em que os acidentes são agrupados por ocorrências, foi efetuado tratamento de atributos, com a conversão de atributos para inteiro, string e data; além do preenchimento da informação de uso do solo para vias em ambiente urbano e rural, bem como decidiu-se por efetuar o “drop” das colunas ‘sentido_via’, ‘regional’, ‘delegacia’ e ‘uop’, já que tais atributos não serão considerados no estudo.

Ver Figura 22.

Figura 22: 2º Processamento e tratamento do arquivo “datatran_mg”
(autor)

```

1 #Tratamento dos arquivos datatran.CSV - conversão de atributos
2 from numpy import int64
3 datatran_mg['id'] = datatran_mg.id.astype('int64')
4 datatran_mg['br'] = datatran_mg.br.astype('int64')
5 datatran_mg['km'] = datatran_mg.km.astype('int64')
6 datatran_mg['uso_solo'] = datatran_mg.uso_solo.astype('string')
7 datatran_mg['data_inversa'] = datatran_mg.data_inversa.astype('datetime64')
8 #datatran_mg.info()

1 #Tratamento dos arquivos datatran.CSV - configurando registros Urbano e Rural
2 datatran_mg['uso_solo'].replace(['Sim'], 'Urbano', inplace=True)
3 datatran_mg['uso_solo'].replace(['Não'], 'Rural', inplace=True)

1 #Tratamento dos arquivos datatran.CSV - remoção de atributos que não serão utilizados
2 datatran_mg.drop(columns = ['sentido_via', 'regional', 'delegacia', 'uop'], axis=1, inplace=True)
3 datatran_mg.reset_index(inplace=True)
4 datatran_mg.drop(columns=['index'], axis=1, inplace=True)

```

Para o arquivo “acidentes”, em que os acidentes são agrupados por pessoas, foi efetuado tratamento de atributos, com a conversão de atributos para inteiro, string e data; além do preenchimento da informação de uso do solo para vias em ambiente urbano e rural, bem como decidiu-se por efetuar o “drop” das colunas ‘sentido_via’, ‘regional’, ‘delegacia’, ‘uop’, ‘id_veiculo’, ‘marca’, ‘ano_fabricacao_veiculo’ e ‘idade’, já que tais atributos não serão considerados no estudo.

Figura 23: 2º Processamento e tratamento do arquivo “acidentes_mg”
(autor)

```

1 #Tratamento dos arquivos acidentes.CSV - conversão de atributos
2 from numpy import int64
3 acidentes_mg['id'] = acidentes_mg.id.astype('int64')
4 acidentes_mg['pesid'] = acidentes_mg.pesid.astype('int64')
5 acidentes_mg['br'] = acidentes_mg.br.astype('int64')
6 acidentes_mg['km'] = acidentes_mg.km.astype('int64')
7 acidentes_mg['uso_solo'] = acidentes_mg.uso_solo.astype('string')
8 acidentes_mg['data_inversa'] = acidentes_mg.data_inversa.astype('datetime64')
9 #acidentes_mg.info()

1 #Tratamento dos arquivos acidentes.CSV - configurando registros Urbano e Rural
2 acidentes_mg['uso_solo'].replace(['Sim'], 'Urbano', inplace=True)
3 acidentes_mg['uso_solo'].replace(['Não'], 'Rural', inplace=True)

```

```

1 #Tratamento dos arquivos acidentes.CSV - remoção de atributos que não serão utilizados
2 acidentes_mg.drop(columns = ['sentido_via','regional', 'delegacia', 'uop', 'id_veiculo', 'marca', 'ano_fabricacao_veiculo','i
3 acidentes_mg.reset_index(inplace=True)
4 acidentes_mg.drop(columns=['index'], axis=1, inplace=True)

```

Os arquivos após o tratamento acima ficaram da seguinte forma (Figuras 24 e 25):

Figura 24: Conferindo o tratamento do arquivo “datatran_mg” (autor)

#Tratamento dos arquivos datatran.CSV - conferindo as alterações															
	id	data_inversa	dia_semana	horario	uf	br	km	municipio	causa_acidente	tipo_acidente	classificacao_acidente	fase_dia	condicao_metereologica		
0	47	2017-01-01	domingo	04:50:00	MG	381	605	OLIVEIRA	Condutor Dormindo	Saída de leito carroçável	Com Vítimas Feridas	Amanhecer	Céu Claro		
1	52	2017-01-01	domingo	05:00:00	MG	262	368	JUATUBA	Velocidade Incompatível	Colisão com objeto estático	Com Vítimas Feridas	Plena Noite	Céu Claro		
2	53	2017-01-01	domingo	05:00:00	MG	459	14	CALDAS	Falta de Atenção à Condução	Saída de leito carroçável	Sem Vítimas	Amanhecer	Céu Claro		
3	61	2017-01-01	domingo	05:00:00	MG	262	186	SAO DOMINGOS DO PRATA	Condutor Dormindo	Saída de leito carroçável	Com Vítimas Feridas	Plena Noite	Céu Claro		
4	63	2017-01-01	domingo	06:00:00	MG	135	408	BOCAIUVA	Animais na Pista	Atropelamento de Animal	Com Vítimas Feridas	Amanhecer	Céu Claro		
condicao_metereologica tipo_pista tracado_via uso_solo pessoas mortos feridos_leves feridos_graves iloses ignorados feridos veiculos latitude longitude															
	Céu Claro	Dupla	Reta	Rural		2	0		2		0	0	0	2	1 -20.6370 -44.7357
	Céu Claro	Simples	Viaduto	Urbano		3	0		3		0	0	0	3	1 -19.9566 -44.3444
	Céu Claro	Simples	Curva	Rural		1	0		0		0	1	0	0	1 -21.8459 -46.4388
	Céu Claro	Simples	Curva	Rural		1	0		1		0	0	0	1	1 -19.8942 -43.0403
	Céu Claro	Simples	Reta	Rural		3	0		1		0	2	0	1	1 -16.9614 -43.8590

Figura 25: Conferindo o tratamento do arquivo “acidentes_mg” (autor)

#Tratamento dos arquivos acidentes.CSV - conferindo as alterações														
	id	pesid	data_inversa	dia_semana	horario	uf	br	km	municipio	causa_acidente	tipo_acidente	classificacao_acidente	fase_dia	condicao_meterologica
0	47	44	2017-01-01	domingo	04:50:00	MG	381	605	OLIVEIRA	Condutor Dormindo	Saída de leito carroçável	Com Vítimas Feridas	Amanhecer	Céu Claro
1	47	45	2017-01-01	domingo	04:50:00	MG	381	605	OLIVEIRA	Condutor Dormindo	Saída de leito carroçável	Com Vítimas Feridas	Amanhecer	Céu Claro
2	52	1528	2017-01-01	domingo	05:00:00	MG	262	368	JUATUBA	Velocidade Incompatível	Colisão com objeto estático	Com Vítimas Feridas	Plena Noite	Céu Claro
3	52	1526	2017-01-01	domingo	05:00:00	MG	262	368	JUATUBA	Velocidade Incompatível	Colisão com objeto estático	Com Vítimas Feridas	Plena Noite	Céu Claro
4	52	1519	2017-01-01	domingo	05:00:00	MG	262	368	JUATUBA	Velocidade Incompatível	Colisão com objeto estático	Com Vítimas Feridas	Plena Noite	Céu Claro

xígica	tipo_pista	tracado_via	uso_solo	tipo_veiculo	tipo_envolvido	estado_físico	sexo	ilesos	feridos_leves	feridos_graves	mortos	latitude	longitude
Claro	Dupla	Reta	Rural	Automóvel	Condutor	Lesões Leves	Masculino	0	1	0	0	-20.6370	-44.7357
Claro	Dupla	Reta	Rural	Automóvel	Passageiro	Lesões Leves	Masculino	0	1	0	0	-20.6370	-44.7357
Claro	Simples	Viaduto	Urbano	Automóvel	Passageiro	Lesões Leves	Masculino	0	1	0	0	-19.9566	-44.3444
Claro	Simples	Viaduto	Urbano	Automóvel	Passageiro	Lesões Leves	Feminino	0	1	0	0	-19.9566	-44.3444
Claro	Simples	Viaduto	Urbano	Automóvel	Condutor	Lesões Leves	Masculino	0	1	0	0	-19.9566	-44.3444

Na sequência, procedeu-se com o enriquecimento dos dados. Como uma estrutura analítica, a Matriz de Haddon (Haddon, 1980), desenvolvida para análise de alocação de recursos, identificação de estratégias e planejamento, permite identificar fatores de risco na interação dos três elementos do trânsito (homem, veículo e via) nos três estágios da colisão de veículos: o antes da colisão, a colisão em si e o após a colisão, conforme Figura 26.

Figura 26: Matriz de Haddon
(adaptado por OPAS/OMS-MS, 2011)

		FATORES		
FASE		HUMANO	VEÍCULO E EQUIPAMENTOS	AMBIENTE
Antes da colisão	Prevenção da colisão	Informações Atitudes Condição debilitada Aplicação da lei	Condições mecânicas Luzes Freios Direção Controle de velocidade	Projetos das vias Limites de velocidade Elementos de segurança para pedestres
Colisão	Prevenção de lesões durante a colisão	Uso de dispositivos de retenção Condição debilitada	Cintos de segurança Outros dispositivos de segurança Design com proteção a impactos	Elementos de proteção ao longo das vias
Após a colisão	Preservação da vida	Noções de primeiros socorros Acesso à atenção médica	Facilidade de acesso Risco de incêndio	Facilidade para o resgate Congestionamento

De acordo com OPAS/OMS-MS (2011), evidências mostram que com uma abordagem sistêmica, é possível perceber oportunidades de intervenção e atuar visando promover a redução de lesões causadas pelos acidentes de trânsito. Portanto, foi efetuando enriquecimento dos dados visando identificar as ocorrências de acidentes relacionadas com os elementos do sistema de trânsito (o veículo, a via ou o homem).

Nesse sentido, foi incluída coluna ‘elemento_transito’ e o atributo informado conforme a causa do acidente, como apresentado na Figura 27 e no Quadro 5, tanto no arquivo “datatran”, que tem agrupamento dos acidentes por ocorrência, quanto no arquivo “acidentes”, que tem agrupamento dos acidentes por pessoas.

Figura 27: Enriquecimento dos dados dos arquivos “datatran_mg” e “acidentes_mg” (autor)

```
#Enriquecimento de dados dos arquivos datatran.CSV - preenche elemento_transito
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Agressão Externa"), datatran_mg['elemento_transito'], "Agressão Externa")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Carga excessiva e/ou mal acondicionada"), datatran_mg['elemento_transito'], "Carga excessiva e/ou mal acondicionada")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Condutor Dormindo"), datatran_mg['elemento_transito'], "Condutor Dormindo")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Desobediência às normas de trânsito pelo condutor"), datatran_mg['elemento_transito'], "Desobediência às normas de trânsito pelo condutor")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Desobediência às normas de trânsito pelo pedestre"), datatran_mg['elemento_transito'], "Desobediência às normas de trânsito pelo pedestre")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Falta de Atenção à Condução"), datatran_mg['elemento_transito'], "Falta de Atenção à Condução")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Falta de Atenção do Pedestre"), datatran_mg['elemento_transito'], "Falta de Atenção do Pedestre")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Ingestão de Álcool"), datatran_mg['elemento_transito'], "Ingestão de Álcool")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Ingestão de álcool e/ou substâncias psicoativas"), datatran_mg['elemento_transito'], "Ingestão de álcool e/ou substâncias psicoativas")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Ingestão de Substâncias Psicoativas"), datatran_mg['elemento_transito'], "Ingestão de Substâncias Psicoativas")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Ingestão de álcool pelo condutor"), datatran_mg['elemento_transito'], "Ingestão de álcool pelo condutor")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Mal Súbito"), datatran_mg['elemento_transito'], "Mal Súbito")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Não guardar distância de segurança"), datatran_mg['elemento_transito'], "Não guardar distância de segurança")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Ultrapassagem Indevida"), datatran_mg['elemento_transito'], "Ultrapassagem Indevida")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Velocidade Incompatível"), datatran_mg['elemento_transito'], "Velocidade Incompatível")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Ausência de reação do condutor"), datatran_mg['elemento_transito'], "Ausência de reação do condutor")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Manobra de mudança de faixa"), datatran_mg['elemento_transito'], "Manobra de mudança de faixa")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Reação tardia ou ineficiente do condutor"), datatran_mg['elemento_transito'], "Reação tardia ou ineficiente do condutor")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Transitar na contramão"), datatran_mg['elemento_transito'], "Transitar na contramão")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Conversão proibida"), datatran_mg['elemento_transito'], "Conversão proibida")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Avarias e/ou desgaste excessivo no pneu"), datatran_mg['elemento_transito'], "Avarias e/ou desgaste excessivo no pneu")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Defeito Mecânico no Veículo"), datatran_mg['elemento_transito'], "Defeito Mecânico no Veículo")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Deficiência ou não Acionamento do Sistema de Iluminação"), datatran_mg['elemento_transito'], "Deficiência ou não Acionamento do Sistema de Iluminação")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Demais falhas mecânicas ou elétricas"), datatran_mg['elemento_transito'], "Demais falhas mecânicas ou elétricas")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Animais na Pista"), datatran_mg['elemento_transito'], "Animais na Pista")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Defeito na Via"), datatran_mg['elemento_transito'], "Defeito na Via")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Fenômenos da Natureza"), datatran_mg['elemento_transito'], "Fenômenos da Natureza")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Objeto estático sobre o leito carroçável"), datatran_mg['elemento_transito'], "Objeto estático sobre o leito carroçável")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Pista Escorregadia"), datatran_mg['elemento_transito'], "Pista Escorregadia")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Restrição de Visibilidade"), datatran_mg['elemento_transito'], "Restrição de Visibilidade")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Sinalização da via insuficiente ou inadequada"), datatran_mg['elemento_transito'], "Sinalização da via insuficiente ou inadequada")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Ausência de sinalização"), datatran_mg['elemento_transito'], "Ausência de sinalização")
datatran_mg['elemento_transito'] = np.where((datatran_mg['causa_acidente'] == "Chuva"), datatran_mg['elemento_transito'].replace("Chuva", "chuva"))

#Enriquecimento de dados dos arquivos acidentes.CSV - preenche elemento_transito
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Agressão Externa"), acidentes_mg['elemento_transito'], "Agressão Externa")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Carga excessiva e/ou mal acondicionada"), acidentes_mg['elemento_transito'], "Carga excessiva e/ou mal acondicionada")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Condutor Dormindo"), acidentes_mg['elemento_transito'], "Condutor Dormindo")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Desobediência às normas de trânsito pelo condutor"), acidentes_mg['elemento_transito'], "Desobediência às normas de trânsito pelo condutor")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Desobediência às normas de trânsito pelo pedestre"), acidentes_mg['elemento_transito'], "Desobediência às normas de trânsito pelo pedestre")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Falta de Atenção à Condução"), acidentes_mg['elemento_transito'], "Falta de Atenção à Condução")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Falta de Atenção do Pedestre"), acidentes_mg['elemento_transito'], "Falta de Atenção do Pedestre")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Ingestão de Álcool"), acidentes_mg['elemento_transito'], "Ingestão de Álcool")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Ingestão de álcool e/ou substâncias psicoativas"), acidentes_mg['elemento_transito'], "Ingestão de álcool e/ou substâncias psicoativas")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Ingestão de Substâncias Psicoativas"), acidentes_mg['elemento_transito'], "Ingestão de Substâncias Psicoativas")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Ingestão de álcool pelo condutor"), acidentes_mg['elemento_transito'], "Ingestão de álcool pelo condutor")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Mal Súbito"), acidentes_mg['elemento_transito'], "Mal Súbito")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Não guardar distância de segurança"), acidentes_mg['elemento_transito'], "Não guardar distância de segurança")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Ultrapassagem Indevida"), acidentes_mg['elemento_transito'], "Ultrapassagem Indevida")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Velocidade Incompatível"), acidentes_mg['elemento_transito'], "Velocidade Incompatível")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Ausência de reação do condutor"), acidentes_mg['elemento_transito'], "Ausência de reação do condutor")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Transitar na contramão"), acidentes_mg['elemento_transito'], "Transitar na contramão")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Conversão proibida"), acidentes_mg['elemento_transito'], "Conversão proibida")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Avarias e/ou desgaste excessivo no pneu"), acidentes_mg['elemento_transito'], "Avarias e/ou desgaste excessivo no pneu")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Defeito Mecânico no Veículo"), acidentes_mg['elemento_transito'], "Defeito Mecânico no Veículo")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Deficiência ou não Acionamento do Sistema de Iluminação"), acidentes_mg['elemento_transito'], "Deficiência ou não Acionamento do Sistema de Iluminação")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Demais falhas mecânicas ou elétricas"), acidentes_mg['elemento_transito'], "Demais falhas mecânicas ou elétricas")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Animais na Pista"), acidentes_mg['elemento_transito'], "Animais na Pista")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Defeito na Via"), acidentes_mg['elemento_transito'], "Defeito na Via")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Fenômenos da Natureza"), acidentes_mg['elemento_transito'], "Fenômenos da Natureza")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Objeto estático sobre o leito carroçável"), acidentes_mg['elemento_transito'], "Objeto estático sobre o leito carroçável")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Pista Escorregadia"), acidentes_mg['elemento_transito'], "Pista Escorregadia")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Restrição de Visibilidade"), acidentes_mg['elemento_transito'], "Restrição de Visibilidade")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Sinalização da via insuficiente ou inadequada"), acidentes_mg['elemento_transito'], "Sinalização da via insuficiente ou inadequada")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Ausência de sinalização"), acidentes_mg['elemento_transito'], "Ausência de sinalização")
acidentes_mg['elemento_transito'] = np.where((acidentes_mg['causa_acidente'] == "Chuva"), acidentes_mg['elemento_transito'].replace("Chuva", "chuva"))
```

Quadro 5: Causa acidente vs Elemento do Trânsito
(dados da pesquisa)

Causa do Acidente	Elemento do Trânsito
"Agressão Externa"	'Homem'
"Carga excessiva e/ou mal acondicionada"	'Homem'
"Condutor Dormindo"	'Homem'
"Desobediência às normas de trânsito pelo condutor"	'Homem'
"Desobediência às normas de trânsito pelo pedestre"	'Homem'

"Falta de Atenção à Condução"	'Homem'
"Falta de Atenção do Pedestre"	'Homem'
"Ingestão de Álcool"	'Homem'
"Ingestão de álcool e/ou substâncias psicoativas pelo pedestre"	'Homem'
"Ingestão de Substâncias Psicoativas"	'Homem'
"Ingestão de álcool pelo condutor"	'Homem'
"Mal Súbito"	'Homem'
"Não guardar distância de segurança"	'Homem'
"Ultrapassagem Indevida"	'Homem'
"Velocidade Incompatível"	'Homem'
"Ausência de reação do condutor"	'Homem'
"Manobra de mudança de faixa"	'Homem'
"Reação tardia ou ineficiente do condutor"	'Homem'
"Transitar na contramão"	'Homem'
"Conversão proibida"	'Homem'
"Avarias e/ou desgaste excessivo no pneu"	'Veículo'
"Defeito Mecânico no Veículo"	'Veículo'
"Deficiência ou não Acionamento do Sistema de Iluminação/Sinalização do Veículo"	'Veículo'
"Demais falhas mecânicas ou elétricas"	'Veículo'
"Animais na Pista"	'Via'
"Defeito na Via"	'Via'
"Fenômenos da Natureza"	'Via'
"Objeto estático sobre o leito carroçável"	'Via'
"Pista Escorregadia"	'Via'
"Restrição de Visibilidade"	'Via'
"Sinalização da via insuficiente ou inadequada"	'Via'
"Ausência de sinalização"	'Via'
"Chuva"	'Via'

6. Análise e exploração dos dados

A análise e exploração dos dados foi efetuada por meio de *scripts* em Python e com o desenvolvimento de um painel analítico MS-PowerBI.

- Análise e exploração com o Python:

- Obtendo a quantidade de acidentes em MG, total do período de 2017 a 2020 e por ano (Figura 28). Percebe-se uma tendência de queda na quantidade de acidentes por ano.

Figura 28: Acidentes, período total e por ano
(autor)

```
Quantidade_acidentes = datatran_mg.shape[0]
print("Acidentes em MG (2017 a 2020): " + str(Quantidade_acidentes))

Acidentes em MG (2017 a 2020): 38835

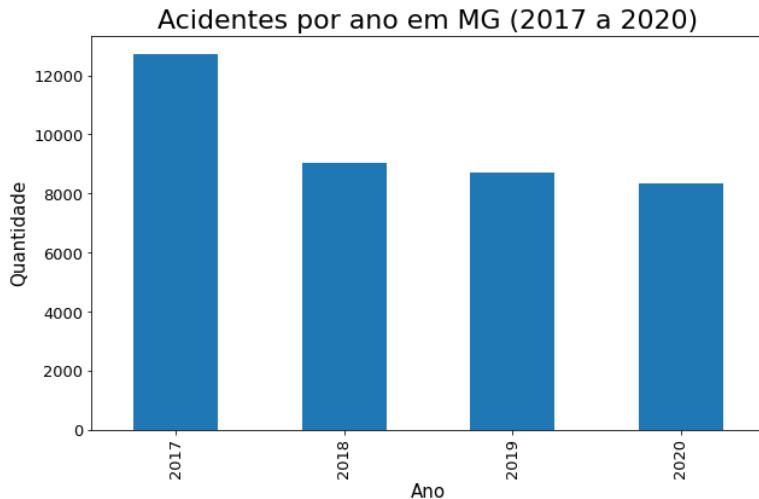
datatran_mg['data_inversa'] = datatran_mg.data_inversa.astype('datetime64')
Acidentes_por_ano = datatran_mg.groupby(datatran_mg['data_inversa']).dt.strftime('%Y')['id'].count().sort_values(ascending=False)
print("Acidentes por ano em MG (2017 a 2020): " + str(Acidentes_por_ano))

Acidentes por ano em MG (2017 a 2020): data_inversa
2017    12711
2018     9055
2019     8713
2020     8356
```

```

1 ax = Acidentes_por_ano.plot(kind='bar', figsize=(10,6), fontsize=13)
2 ax.set_alpha(0.8)
3 ax.set_title("Acidentes por ano em MG (2017 a 2020)", fontsize=22)
4 ax.set_ylabel("Quantidade", fontsize=15)
5 ax.set_xlabel("Ano", fontsize=15)
6 plt.show()

```



- Obtendo a quantidade de feridos em acidentes em MG, total do período de 2017 a 2020 e por ano (Figura 29). Percebe-se que a quantidade de feridos por ano apresentou certa oscilação, tendo uma queda de 2017 para 2018, contudo com um aumento de 2018 para 2019. Em relação ao ano de 2019 para 2020, a queda do número de feridos deveria ter sido mais acentuada já que registrou-se um menor fluxo de veículos nas rodovias em virtude das paralisações ocasionadas pelas medidas restritivas da pandemia da COVID-19.

**Figura 29: Feridos, período total e por ano
(autor)**

```

1 Total_de_feridos = datatran_mg['feridos'].sum()
2 print("Feridos em acidentes em MG (2017 a 2020): " + str(Total_de_feridos))

```

Feridos em acidentes em MG (2017 a 2020): 46100

```

1 Feridos_por_ano = datatran_mg.groupby([datatran_mg['data_inversa'].dt.year.rename('Ano')]).agg({'feridos': 'sum'})
2 print("Feridos em acidentes por ano em MG (2017 a 2020): " + str(Feridos_por_ano))

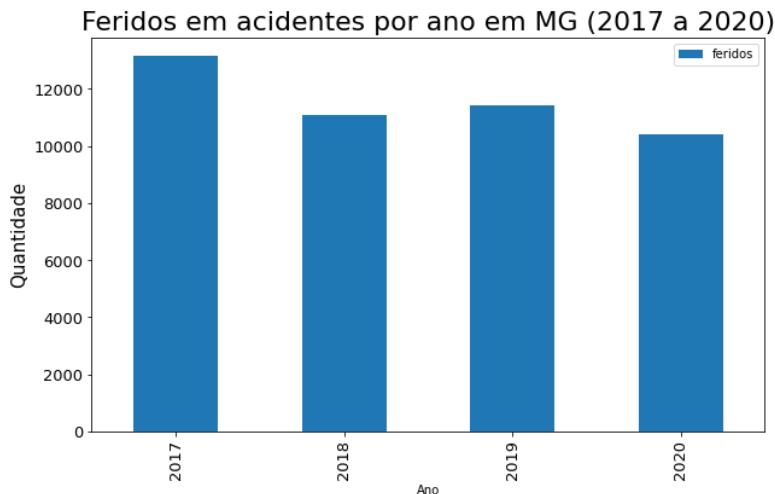
```

Ano	Feridos
2017	13155
2018	11101
2019	11429
2020	10415

```

1 ax = Feridos_por_ano.plot(kind='bar', figsize=(10,6), fontsize=13);
2 ax.set_alpha(0.8)
3 ax.set_title("Feridos em acidentes por ano em MG (2017 a 2020)", fontsize=22)
4 ax.set_ylabel("Quantidade", fontsize=15);
5 plt.show()

```



- Obtendo a quantidade de mortos em acidentes em MG, total do período de 2017 a 2020 e por ano (Figura 30). Percebe-se que a quantidade de mortos por ano também apresentou certa oscilação, tendo queda dois anos seguidos e depois um aumento em 2020. Pode-se perceber que, embora a quantidade de acidentes tenha apresentado queda ano após ano, especialmente, em 2020, os acidentes foram de maior gravidade, levando a um número mais alto de óbitos, lembrando que em 2020 registrou-se um menor fluxo de veículos nas rodovias em virtude das paralisações ocasionadas pelas medidas restritivas da pandemia da COVID-19.

**Figura 30: Mortos, período total e por ano
(autor)**

```

1 Total_de_mortos = datatran_mg['mortos'].sum()
2 print("Mortos em acidentes em MG (2017 a 2020): " + str(Total_de_mortos))

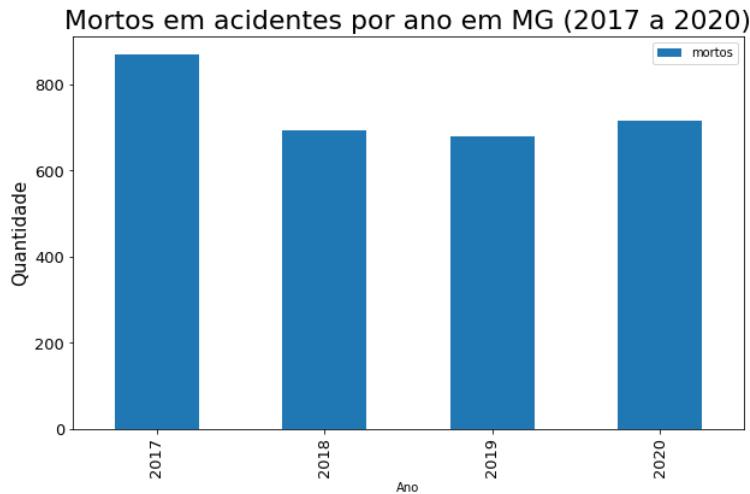
Mortos em acidentes em MG (2017 a 2020): 2957

1 Mortos_por_ano = datatran_mg.groupby([datatran_mg['data_inversa'].dt.year.rename('Ano')]).agg({'mortos': 'sum'})
2 print("Mortos em acidentes por ano em MG (2017 a 2020): " + str(Mortos_por_ano))

Mortos em acidentes por ano em MG (2017 a 2020):      mortos
Ano
2017     869
2018     693
2019     678
2020     717

1 ax = Mortos_por_ano.plot(kind='bar', figsize=(10,6), fontsize=13);
2 ax.set_alpha(0.8)
3 ax.set_title("Mortos em acidentes por ano em MG (2017 a 2020)", fontsize=22)
4 ax.set_ylabel("Quantidade", fontsize=15);
5 plt.show()

```



- Obtendo a quantidade de pessoas e veículos envolvidos em acidentes em MG, total do período de 2017 a 2020 e por ano (Figura 31).

Figura 31: Pessoas e veículos envolvidos, período total e por ano (autor)

```
Quantidade_pessoas = datatran_mg['pessoas'].sum()
Quantidade_veiculos = datatran_mg['veiculos'].sum()
print("Envolvidos em acidentes em MG (2017 a 2020): \n Pessoas: " + str(Quantidade_pessoas) + "\n " + "Veículos envolvidos: "
      + str(Quantidade_veiculos))

Envolvidos em acidentes em MG (2017 a 2020):
Pessoas: 91789
Veículos envolvidos: 59176

Quantidade_pessoas_por_ano = datatran_mg.groupby([datatran_mg['data_inversa'].dt.year.rename('Ano')]).agg({'pessoas': 'sum'})
print("Pessoas envolvidas por ano em MG (2017 a 2020): " + str(Quantidade_pessoas_por_ano))
Quantidade_veiculos_por_ano = datatran_mg.groupby([datatran_mg['data_inversa'].dt.year.rename('Ano')]).agg({'veiculos': 'sum'})
print("Veículos envolvidos por ano em MG (2017 a 2020): " + str(Quantidade_veiculos_por_ano))

Pessoas envolvidas por ano em MG (2017 a 2020):      pessoas
Ano
2017    28781
2018    21857
2019    21364
2020    19787
Veículos envolvidos por ano em MG (2017 a 2020):      veiculos
Ano
2017    18826
2018    13717
2019    13658
2020    12975
```

- Para o estado físico dos envolvidos nos acidentes, foram obtidos dados do período total e por ano.
 - No período total, foram 91.789 envolvidos em acidentes (Figura 32). Deste total, percebe-se que, 80,59% foram de ilos (41,28%) e de lesões leves (39,31%). Pouco mais de 10,92% sofreram lesões graves e 3,22% vieram a óbito. Para 5,27% dos envolvidos nos acidentes, não tiveram o seu

estado físico informado. Embora a taxa de óbitos tenha sido baixa, 50,23% tiveram algum tipo de lesão, o que pode tê-los tornados inválidos ou com sequelas pelo resto de suas vidas.

Não obstante, mesmo àqueles que saíram ilesos, podem carregar cargas emocionais decorrentes de traumas dos acidentes e ou pela perda de parentes e ou entes queridos nos acidentes pelo resto de suas vidas.

Figura 32: Estado físico dos envolvidos nos acidentes, período total (autor)

```
Estado_fisico = acidentes_mg.groupby(['estado_fisico']).size()
print("Estado físico dos envolvidos em MG (2017 a 2020): " + str(Estado_fisico))

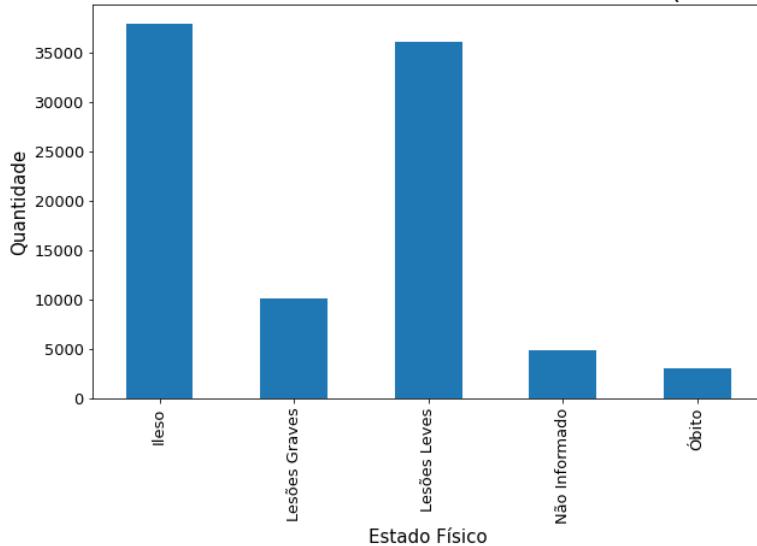
Estado fisico dos envolvidos em MG (2017 a 2020): estado_fisico
Ileso      37895
Lesões Graves   10022
Lesões Leves    36078
Não Informado   4837
Óbito        2957
dtype: int64

Quantidade_envolvidos = acidentes_mg.shape[0]
perc_ilesos = (acidentes_mg['id'][acidentes_mg['estado_fisico'] == 'Ileso'].count() / Quantidade_envolvidos) *100
perc_leagr = (acidentes_mg['id'][acidentes_mg['estado_fisico'] == 'Lesões Graves'].count() / Quantidade_envolvidos) *100
perc_leslv = (acidentes_mg['id'][acidentes_mg['estado_fisico'] == 'Lesões Leves'].count() / Quantidade_envolvidos) *100
perc_nainf = (acidentes_mg['id'][acidentes_mg['estado_fisico'] == 'Não Informado'].count() / Quantidade_envolvidos) *100
perc_obito = (acidentes_mg['id'][acidentes_mg['estado_fisico'] == 'Óbito'].count() / Quantidade_envolvidos) *100
print("% Estado físico dos envolvidos em MG (2017 a 2020): \n Ileso " + "%.2f" % perc_ilesos + "\n Lesões Graves " + "%.2f" % perc_leagr + "\n Lesões Leves " + "%.2f" % perc_leslv + "\n Não Informado " + "%.2f" % perc_nainf + "\n Óbito " + "%.2f" % perc_obito)

% Estado físico dos envolvidos em MG (2017 a 2020):
Ileso 41.28
Lesões Graves 10.92
Lesões Leves 39.31
Não Informado 5.27
Óbito 3.22

1 ax = Estado_fisico.plot(kind='bar', figsize=(10,6), fontsize=13);
2 ax.set_alpha(0.8)
3 ax.set_title("Estado físico dos envolvidos em acidentes em MG (2017 a 2020)", fontsize=22)
4 ax.set_xlabel("Quantidade", fontsize=15);
5 ax.set_ylabel("Estado Físico", fontsize=15);
6 plt.show()
```

Estado físico dos envolvidos em acidentes em MG (2017 a 2020)



- Por ano, obtendo subtotais dos envolvidos por estado físico, conforme Figura 33. Interessante perceber uma tendência de queda no número de ilesos, mas de oscilações de lesões graves, lesões leves e de óbitos, com queda em 2018, mas aumento em 2019. E, no ano de 2020, foi verificado aumento no caso de óbitos, o que pode indicar que, embora tenha tido um menor fluxo de veículos em rodovias e até uma menor quantidade de acidentes, os acidentes apresentaram maior fatalidade. Foram produzidos dois gráficos com os dados anuais.

**Figura 33: Estado físico dos envolvidos nos acidentes, por ano
(autor)**

```

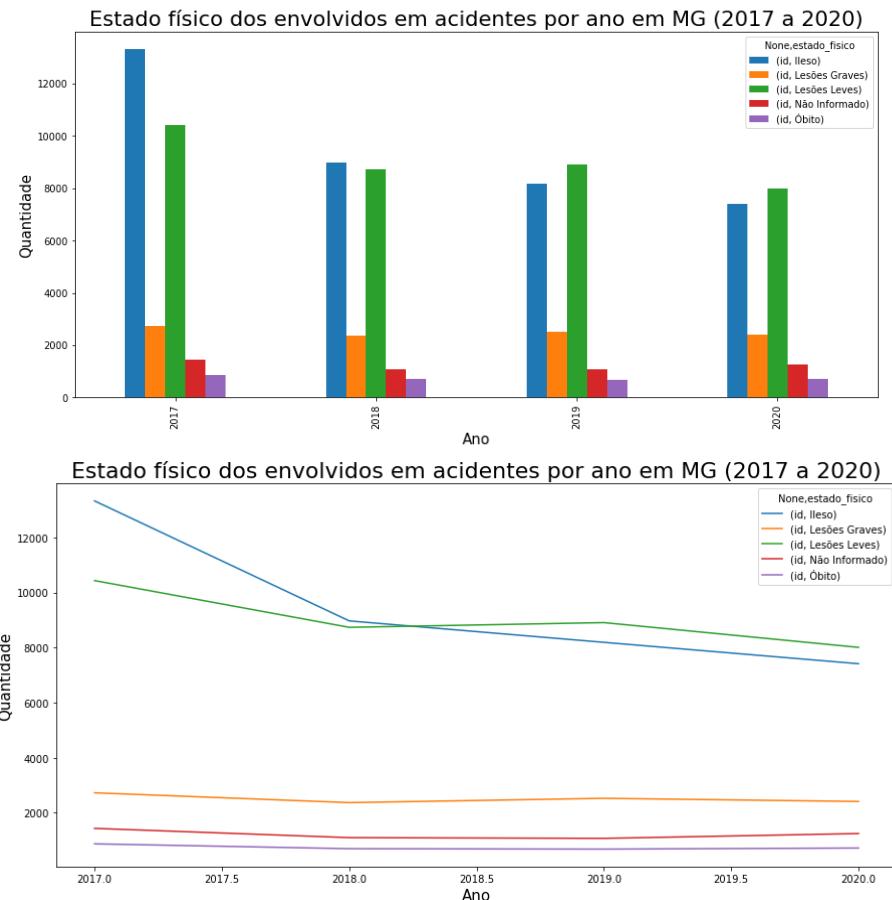
1  acidentes_mg['data_inversa'] = acidentes_mg.data_inversa.astype('datetime64')
2  Estado_fisico_por_ano = acidentes_mg.groupby([acidentes_mg['data_inversa'].dt.year.rename('Ano'), acidentes_mg['estado_fisico']])
3  print("Estado físico dos envolvidos por ano em MG (2017 a 2020): " + str(Estado_fisico_por_ano))
4

Estado físico dos envolvidos por ano em MG (2017 a 2020):
Ano  estado_fisico          id
2017 Ileso      13326
    Lesões Graves  2723
    Lesões Leves   10432
    Não Informado  1431
    Óbito           869
2018 Ileso      8968
    Lesões Graves  2367
    Lesões Leves   8734
    Não Informado  1095
    Óbito           693
2019 Ileso      8189
    Lesões Graves  2523
    Lesões Leves   8966
    Não Informado  1068
    Óbito           678
2020 Ileso      7412
    Lesões Graves  2409
    Lesões Leves   8066
    Não Informado  1243
    Óbito           717

1  fig, ax = plt.subplots(figsize=(15,7))
2  acidentes_mg.groupby([acidentes_mg['data_inversa'].dt.year.rename('Ano'), acidentes_mg['estado_fisico']]).agg({'id': 'count'})
3  ax.set_alpha(0.8)
4  ax.set_title("Estado físico dos envolvidos em acidentes por ano em MG (2017 a 2020)", fontsize=22)
5  ax.set_ylabel("Quantidade", fontsize=15);
6  ax.set_xlabel("Ano", fontsize=15);
7  plt.show()
8

1  fig, ax = plt.subplots(figsize=(15,7))
2  acidentes_mg.groupby([acidentes_mg['data_inversa'].dt.year.rename('Ano'), acidentes_mg['estado_fisico']]).agg({'id': 'count'})
3  ax.set_alpha(0.8)
4  ax.set_title("Estado físico dos envolvidos em acidentes por ano em MG (2017 a 2020)", fontsize=22)
5  ax.set_ylabel("Quantidade", fontsize=15);
6  ax.set_xlabel("Ano", fontsize=15);
7  plt.show()
8

```



- Para a classificação dos acidentes, foram obtidos dados do período total e por ano.
- No período total, observa-se que, dos 38.835 acidentes, 6,12% foram com vítimas fatais, 72,01% com vítimas feridas e 21,87% sem vítimas. Ver Figura 34.

Figura 34: Classificação de acidentes, período total (autor)

```
Classificacao_de_acidentes = datatran_mg.groupby(['classificacao_acidente']).size()
print("Classificação de acidentes em MG (2017 a 2020): " + str(Classificacao_de_acidentes))

Classificação de acidentes em MG (2017 a 2020): classificacao_acidente
Com Vítimas Fatais      2375
Com Vítimas Feridas    27966
Sem Vítimas            8494
dtype: int64

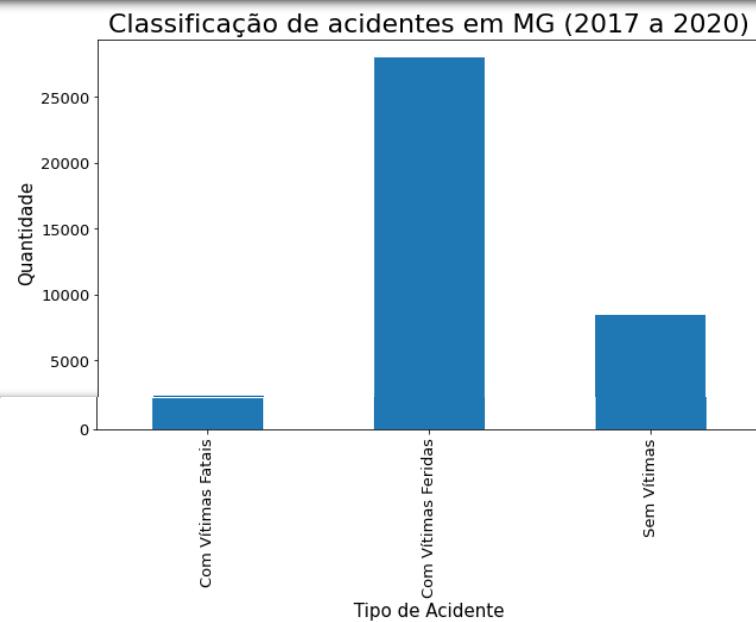
perc_cvfat = (datatran_mg['id'][datatran_mg['classificacao_acidente'] == 'Com Vítimas Fatais'].count() / Quantidade_acidentes)
perc_cvfer = (datatran_mg['id'][datatran_mg['classificacao_acidente'] == 'Com Vítimas Feridas'].count() / Quantidade_acidentes)
perc_sviti = (datatran_mg['id'][datatran_mg['classificacao_acidente'] == 'Sem Vítimas'].count() / Quantidade_acidentes) *100
print("% Classificação de acidentes em MG (2017 a 2020): \n Com Vítimas Fatais " + "%.2f" % perc_cvfat + "\n Com Vítimas Feridas " + "%.2f" % perc_cvfer + "\n Sem Vítimas " + "%.2f" % perc_sviti)

% Classificação de acidentes em MG (2017 a 2020):
Com Vítimas Fatais 6.12
Com Vítimas Feridas 72.01
Sem Vítimas 21.87
```

```

ax = Classificacao_de_acidentes.plot(kind='bar', figsize=(10,6), fontsize=13);
ax.set_alpha(0.8)
ax.set_title("Classificação de acidentes em MG (2017 a 2020)", fontsize=22)
ax.set_ylabel("Quantidade", fontsize=15);
ax.set_xlabel("Tipo de Acidente", fontsize=15);
plt.show()

```



- Por ano, foi possível observar que, embora acidentes com vítimas fatais tenha apresentado queda entre 2017 e 2018, nos anos seguintes apresentou leve aumento. Figura 35.

**Figura 35: Classificação de acidentes, por ano
(autor)**

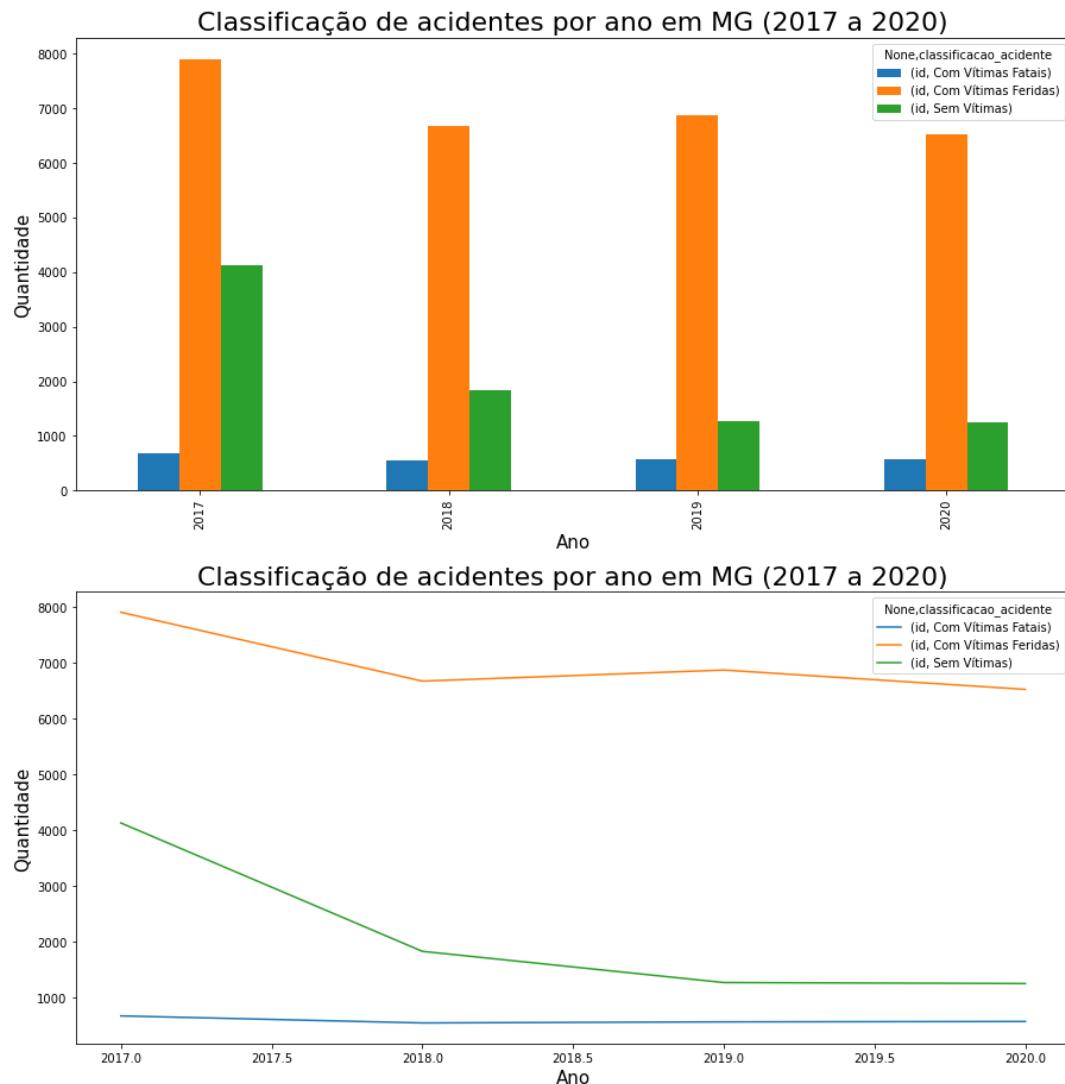
```

Classificacao_de_acidentes_por_ano = datatran_mg.groupby([datatran_mg['data_inversa'].dt.year.rename('Ano'), datatran_mg['classificacao_acidente']])
print("Classificação de acidentes por ano em MG (2017 a 2020): " + str(Classificacao_de_acidentes_por_ano))

fig, ax = plt.subplots(figsize=(15,7))
datatran_mg.groupby([datatran_mg['data_inversa'].dt.year.rename('Ano'), datatran_mg['classificacao_acidente']]).agg({'id': 'count'}).reset_index()
ax.set_alpha(0.8)
ax.set_title("Classificação de acidentes por ano em MG (2017 a 2020)", fontsize=22)
ax.set_ylabel("Quantidade", fontsize=15);
ax.set_xlabel("Ano", fontsize=15);
plt.show()

fig, ax = plt.subplots(figsize=(15,7))
acidentes_mg.groupby([datatran_mg['data_inversa'].dt.year.rename('Ano'), datatran_mg['classificacao_acidente']]).agg({'id': 'count'}).reset_index()
ax.set_alpha(0.8)
ax.set_title("Classificação de acidentes por ano em MG (2017 a 2020)", fontsize=22)
ax.set_ylabel("Quantidade", fontsize=15);
ax.set_xlabel("Ano", fontsize=15);
plt.show()

```



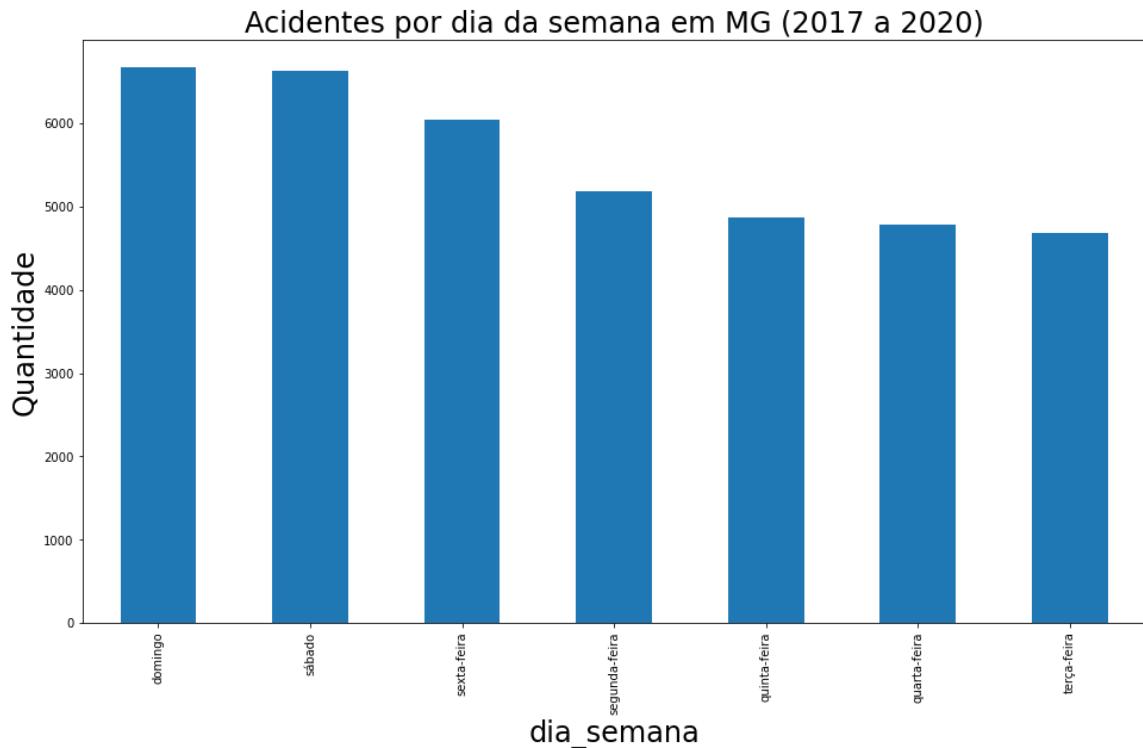
- Para a verificação dos acidentes por dia da semana, foram obtidos dados do período total e por ano.
 - No período total, percebe-se que os dias da semana de maior quantidade de acidentes são o sábado e o domingo, além da sexta-feira, como início do final de semana. Nestes dias da semana, é onde as rodovias federais apresentam maior fluxo de veículos com os deslocamentos de pessoas, sejam retornando da semana de trabalho ou indo passar o final de semana em outras localidades. Ver Figura 36.

Figura 36: Acidentes por dia da semana, período total (autor)

```
Acidentes_por_dia_semana = datatran_mg.groupby(['dia_semana']).size().sort_values(ascending=False)
print("Acidentes por dia da semana em MG (2017 a 2020): " + str(Accidentes_por_dia_semana))
```

```
Acidentes por dia da semana em MG (2017 a 2020): dia_semana
domingo      6663
sábado       6629
sexta-feira   6034
segunda-feira 5183
quinta-feira  4866
quarta-feira  4777
terça-feira   4683
dtype: int64
```

```
plt.title('Acidentes por dia da semana em MG (2017 a 2020)', fontsize=24)
plt.ylabel('Quantidade', fontsize=24)
plt.xlabel("Dia da Semana", fontsize=24)
Acidentes_por_dia_semana.plot(kind='bar')
plt.rcParams["figure.figsize"] = [16, 9]
plt.show()
```



- Por ano, percebe-se o mesmo comportamento, acidentes com predominância nos dias de sexta-feira, sábado e domingo. Interessante que, no ano de 2017, o dia da semana com maior quantidade de acidentes registrados foi o sábado, diferentemente dos demais anos da série. Ver Figura 37.

**Figura 37: Acidentes por dia da semana, por ano
(autor)**

```
Acidentes_por_dia_semana_por_ano = datatran_mg.groupby([datatran_mg['data_inversa'].dt.year.rename('Ano'), datatran_mg['dia_semana']])
print("Acidentes por dia da semana por ano em MG (2017 a 2020): " + str(Accidentes_por_dia_semana_por_ano))

Acidentes por dia da semana por ano em MG (2017 a 2020):
Ano  dia_semana
2017 domingo      2195
      quarta-feira  1537
      quinta-feira  1451
      segunda-feira 1748
      sexta-feira    1997
      sábado        2296
      terça-feira   1487
```

```

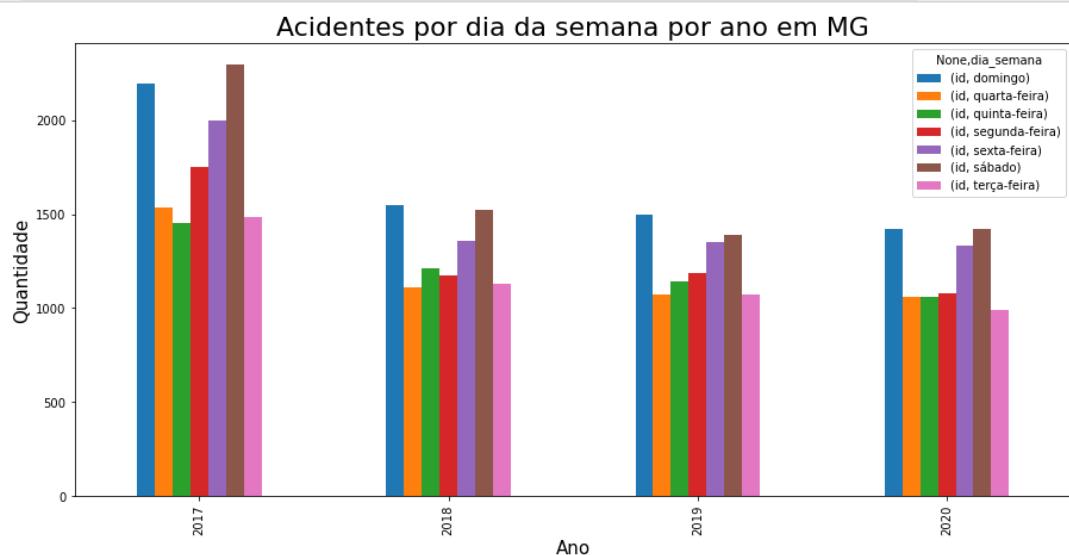
2018 domingo 1548
quarta-feira 1109
quinta-feira 1213
segunda-feira 1173
sexta-feira 1358
sábado 1522
terça-feira 1132
2019 domingo 1497
quarta-feira 1074
quinta-feira 1143
segunda-feira 1185
sexta-feira 1349
sábado 1391
terça-feira 1074
2020 domingo 1423
quarta-feira 1057
quinta-feira 1059
segunda-feira 1077
sexta-feira 1330
sábado 1420
terça-feira 990

```

```

fig, ax = plt.subplots(figsize=(15,7))
datatran_mg.groupby([datatran_mg['data_inversa'].dt.year.rename('Ano'), datatran_mg['dia_semana']]).agg({'id': 'count'}).unstack()
ax.set_alpha(0.8)
ax.set_title("Acidentes por dia da semana por ano em MG (2017 a 2020)", fontsize=22)
ax.set_ylabel("Quantidade", fontsize=15);
ax.set_xlabel("Ano", fontsize=15);
plt.show()

```



- Verificando a ocorrência de acidentes agrupados por mês, período total e por ano. No período total, o mês de maior quantidade de acidentes é o de dezembro seguido por janeiro, possivelmente em virtude dos deslocamentos em períodos de férias e de comemorações. Depois, têm-se os meses. Ver Figura 38.

**Figura 38: Acidentes por mês, período total
(autor)**

```

datatran_mg.data_inversa = pd.to_datetime(datatran_mg.data_inversa)
Acidentes_por_mes = datatran_mg.groupby(datatran_mg['data_inversa'].dt.strftime('%B'))['id'].count().sort_values(ascending=False)
print("Acidentes por mês em MG (2017 a 2020): " + str(Acidentes_por_mes))

```

Mês	Quantidade
December	3838
January	3717
October	3440
February	3375
...	...

```

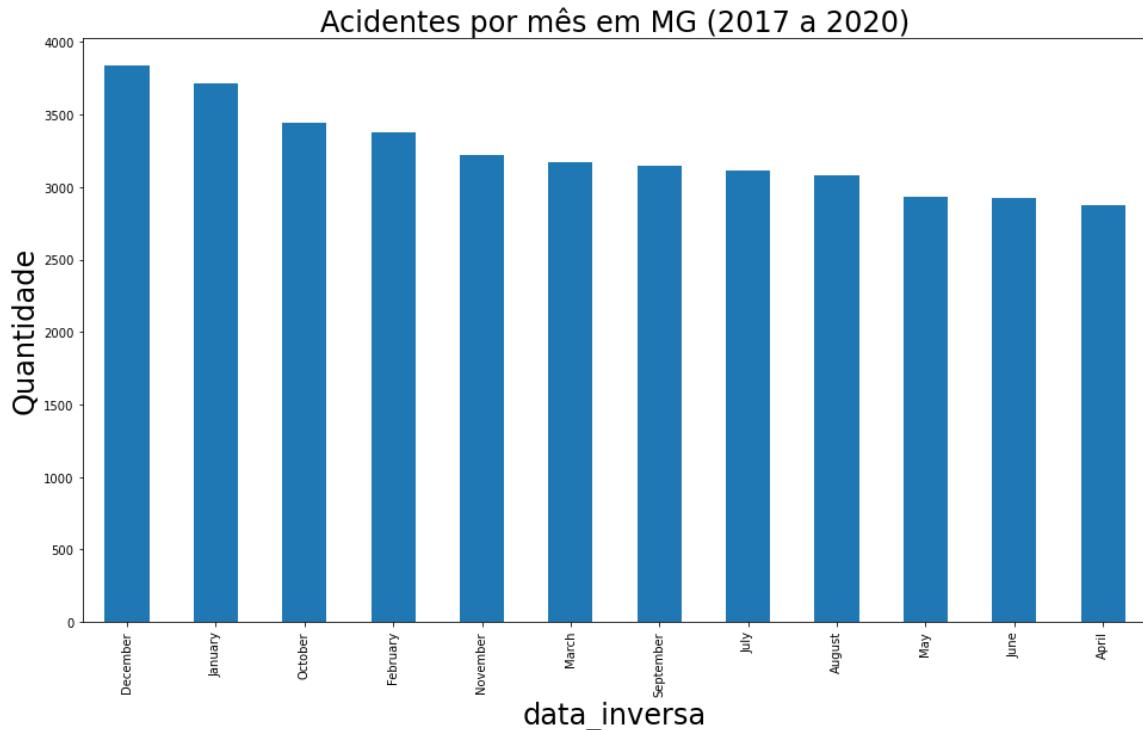
November      3220
March         3175
September    3147
July          3110
August        3084
May           2931
June          2922
April         2876
Name: id, dtype: int64

```

```

plt.title('Acidentes por mês em MG (2017 a 2020)', fontsize=24)
plt.ylabel('Quantidade', fontsize=24)
plt.xlabel('Mês', fontsize=24)
Acidentes_por_mes.plot(kind='bar')
plt.rcParams["figure.figsize"] = [32,18]
plt.show()

```



- Obtendo os tipos de veículos envolvidos nos acidentes, no período total, percebe-se que majoritariamente a participação é de veículos leves, em especial de automóveis e de motocicletas. Na sequência, aparecem os veículos de carga e, depois, ônibus. Ver Figura 39.

Figura 39: Acidentes por tipo de veículo envolvido, período total (autor)

```

Acidentes_por_tipo_veiculo = acidentes_mg.groupby(['tipo_veiculo']).size().sort_values(ascending=False)
print("Tipo de veículos(2017 a 2020): " + str(Accidentes_por_tipo_veiculo))

```

```

Tipo de veículos(2017 a 2020): tipo_veiculo
Automóvel      41970
Motocicleta    13523
Caminhão-trator  9634
Caminhão       7831
Caminhonete     7812
Ônibus          5018
Camineta        2100
Micro-ônibus     1087
Bicicleta        896
Utilitário       783
Motoneta         529
Outros           266
Semireboque      185
Ciclomotor        64
Reboque           26
Trator de rodas   23

```

```

Carroça-charrete      20
Triciclo              13
Não Informado         5
Trator de esteira     3
Chassi-plataforma    1
dtype: int64

```

- Obtendo acidentes de acordo com a sua causa, período total, percebe-se que a “Falta de Atenção à Condução” foi a maior causa presumível dos acidentes, seguido por “Velocidade Incompatível”, ambos relacionados ao fator humano, ao comportamento em especial do condutor, categorizado como o elemento de trânsito “homem”. Ver Figura 40.

**Figura 40: Acidentes por causa presumível, período total
(autor)**

```

Acidentes_por_causa = datatran_mg.groupby(['causa_acidente']).size().sort_values(ascending=False)
print("Acidentes por causa em MG (2017 a 2020): " + str(Accidentes_por_causa))

```

Acidentes por causa em MG (2017 a 2020): causa_acidente	
Falta de Atenção à Condução	12362
Velocidade Incompatível	8002
Desobediência às normas de trânsito pelo condutor	2643
Defeito Mecânico no Veículo	2404
Condutor Dormindo	2374
Não guardar distância de segurança	1853
Ingestão de Álcool	1839
Pista Escorregadia	1823
Falta de Atenção do Pedestre	905
Animais na Pista	815
Ultrapassagem Indevida	683
Avarias e/ou desgaste excessivo no pneu	586
Mal Súbito	511
Objeto estático sobre o leito carroçável	319
Restrição de Visibilidade	319
Defeito na Via	318
Carga excessiva e/ou mal acondicionada	297
Fenômenos da Natureza	183
Ingestão de álcool e/ou substâncias psicoativas pelo pedestre	134
Agressão Externa	133
Sinalização da via insuficiente ou inadequada	128
Desobediência às normas de trânsito pelo pedestre	101
Deficiência ou não Acionamento do Sistema de Iluminação/Sinalização do Veículo	58
Ingestão de Substâncias Psicoativas	34
Reação tardia ou ineficiente do condutor	2
Transitar na contramão	2
Demais falhas mecânicas ou elétricas	1
Conversão proibida	1
Chuva	1
Ingestão de álcool pelo condutor	1
Manobra de mudança de faixa	1
Ausência de sinalização	1
Ausência de reação do condutor	1

- Para a verificação dos acidentes por elemento de trânsito, em que a causa do acidente foi classificada como homem, veículo ou via; foram obtidos dados do período total e por ano.
 - No período total, observa-se que, dos 38.835 acidentes, os acidentes foram causados majoritariamente pelo elemento “homem”, com 82,09%, o que indica a relação dos acidentes com o comportamento humano. Este percentual é próximo

com o encontrado por Menezes (2001) e Soares et al. (2018). Ver Figura 41.

Figura 41: Elemento de Trânsito, período total (autor)

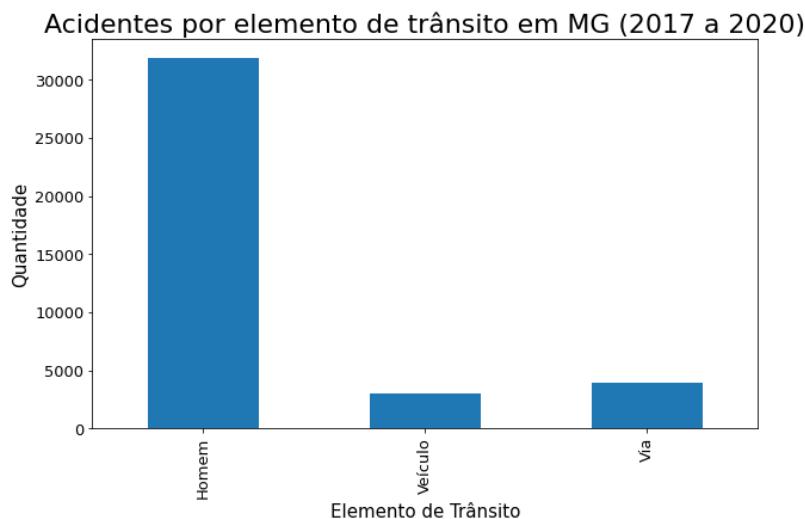
```
Elemento_de_transito = datatran_mg.groupby(['elemento_transito']).size()
print("Acidentes por elemento de trânsito em MG (2017 a 2020): " + str(Elemento_de_transito))

Acidentes por elemento de trânsito em MG (2017 a 2020): elemento_transito
Homem      31879
Veiculo     3049
Via         3907
dtype: int64

perc_homem = (datatran_mg['id'][datatran_mg['elemento_transito'] == 'Homem'].count() / Quantidade_acidentes) *100
perc_veic = (datatran_mg['id'][datatran_mg['elemento_transito'] == 'Veículo'].count() / Quantidade_acidentes) *100
perc_via = (datatran_mg['id'][datatran_mg['elemento_transito'] == 'Via'].count() / Quantidade_acidentes) *100
print("% Acidentes por elemento de trânsito em MG (2017 a 2020): \n Homem " + "%.2f" % perc_homem + "\n Veiculo " + "%.2f" % perc_veic + "\n Via " + "%.2f" % perc_via)

% Acidentes por elemento de trânsito em MG (2017 a 2020):
Homem 82.09
Veiculo 7.85
Via 10.06

ax = Elemento_de_transito.plot(kind='bar', figsize=(10,6), fontsize=13);
ax.set_alpha(0.8)
ax.set_title("Acidentes por elemento de trânsito em MG (2017 a 2020)", fontsize=22)
ax.set_ylabel("Quantidade", fontsize=15);
ax.set_xlabel("Elemento de Trânsito", fontsize=15);
plt.show()
```



- Por ano, é possível perceber uma queda substancial na quantidade de acidentes com a causa classificada no elemento de trânsito de “homem” entre os anos de 2017 e 2018; nos anos seguintes, seguiu com pequenas variações. Isso pode indicar um reforço nas ações de educação de trânsito quanto ao comportamento humano assim como também em intensificação de fiscalizações nas rodovias federais. Ver Figura 42.

**Figura 42: Elemento de Trânsito, por ano
(autor)**



- Obtendo acidentes de acordo com o seu tipo, período total, percebe-se que a “Saída de leito carroçável” foi o de maior frequência, seguido por “Colisão traseira”. Possivelmente, estas ocorrências podem estar relacionadas com a condução em velocidade incompatível com a via, desrespeito às regras de sinalização, ultrapassagens indevidas, dentre outras. Ver Figura 43.

**Figura 43: Acidentes por tipo, período total
(autor)**

```
Acidentes_por_tipo = datatran_mg.groupby(['tipo_acidente']).size().sort_values(ascending=False)
print("Acidentes por tipo em MG (2017 a 2020): " + str(Accidentes_por_tipo))
```

```
Acidentes por tipo em MG (2017 a 2020): tipo_acidente
Saída de leito carroçável      8899
Colisão traseira              5888
Tombamento                     3880
Colisão lateral                3554
Colisão com objeto estático    3422
Colisão transversal            3115
Colisão frontal                 2523
Capotamento                     2248
Queda de ocupante de veículo    1540
Atropelamento de Pedestre       1466
Incêndio                         697
Atropelamento de Animal          633
Engavetamento                   403
Derramamento de carga             196
Colisão com objeto em movimento   194
Danos eventuais                  176
Colisão com objeto                   1
dtype: int64
```

- Verificando os acidentes pela fase do dia, percebe-se que boa parte dos acidentes acontece em pleno dia. Aliado a isso, ao verificar os acidentes de acordo com as condições meteorológicas registradas, percebe-se que boa parte dos acidentes ocorreu com céu claro. Desta forma, presume-se que havia boas condições de visibilidade na via. Ver Figuras 44 e 45.

**Figura 44: Acidentes por fase do dia, período total
(autor)**

```
Acidentes_por_fase_do_dia = datatran_mg.groupby(['fase_dia']).size().sort_values(ascending=False)
print("Acidentes por fase do dia em MG (2017 a 2020): " + str(Accidentes_por_fase_do_dia))
```

```
Acidentes por fase do dia em MG (2017 a 2020): fase_dia
Pleno dia                      22162
Plena Noite                     12648
Amanhecer                        2092
Anoitecer                         1933
dtype: int64
```

**Figura 45: Acidentes por tipo, período total
(autor)**

```
Acidentes_por_condi_met = datatran_mg.groupby(['condicao_metereologica']).size().sort_values(ascending=False)
print("Acidentes por fase do dia em MG (2017 a 2020): " + str(Accidentes_por_condi_met))
```

```
Acidentes por fase do dia em MG (2017 a 2020): condicao_metereologica
Céu Claro                       20726
Nublado                          6698
Chuva                            6591
Sol                               2803
```

```

Garoa/Chuvisco      933
Ignorado           675
Nevoeiro/Neblina   313
Vento               93
Granizo              2
Neve                1
dtype: int64

```

- Verificando os acidentes pelo tipo de pista, embora os acidentes estejam mais presentes em pista simples, também ocorrem em quantidade elevada em pista dupla. Também, verifica-se que os acidentes ocorrem mais reta, possivelmente associados à velocidade incompatível com a via e às ultrapassagens indevidas. Não obstante, acidentes em perímetro urbano são de menor quantidade do que os de meio rural. Ver Figuras 46, 47 e 48.

**Figura 46: Acidentes por tipo de pista, período total
(autor)**

```

Acidentes_por_tipo_de_pista = datatran_mg.groupby(['tipo_pista']).size().sort_values(ascending=False)
print("Acidentes por tipo de pista em MG (2017 a 2020): " + str(Accidentes_por_tipo_de_pista))

Acidentes por tipo de pista em MG (2017 a 2020): tipo_pista
Simples      19440
Dupla        18243
Múltipla     1152
dtype: int64

```

**Figura 47: Acidentes por traçado da via, período total
(autor)**

```

Acidentes_por_tracado_da_via = datatran_mg.groupby(['tracado_via']).size().sort_values(ascending=False)
print("Acidentes por tipo de traçado em MG (2017 a 2020): " + str(Accidentes_por_tracado_da_via))

Acidentes por tipo de traçado em MG (2017 a 2020): tracado_via
Reta          18865
Curva         11141
Não Informado 5386
Interseção de vias 1175
Desvio Temporário 824
Rotatória       499
Retorno Regulamentado 481
Ponte          211
Viaduto        203
Túnel           50
dtype: int64

```

**Figura 48: Acidentes por uso do solo, período total
(autor)**

```

Acidentes_por_uso_solo = datatran_mg.groupby(['uso_solo']).size().sort_values(ascending=False)
print("Acidentes por tipo de traçado em MG (2017 a 2020): " + str(Accidentes_por_uso_solo))

Acidentes por tipo de traçado em MG (2017 a 2020): uso_solo
Rural         27435
Urbano        11400
dtype: int64

```

- Obtendo um *ranking* de acidentes por rodovia federal, percebe-se que a BR-381 é a que apresenta maiores números de conflitos no trânsito, possivelmente relacionado à importância da BR na malha do estado de MG e ao elevando fluxo de veículos quando comparada com outras BRs. Em segundo lugar, está a BR-040, que tem importante ligação com o Distrito Federal. Ver Figura 49.

**Figura 49: Acidentes por BR, período total
(autor)**

```
Acidentes_por_BR = datatran_mg.groupby(['br']).size().sort_values(ascending=False)
print("Acidentes por BR em MG (2017 a 2020): " + str(Acidentes_por_BR))
```

```
Acidentes por BR em MG (2017 a 2020): br
381    12291
40     7549
116    4960
262    4315
365    2702
50     2248
153    1064
251     949
267     815
459     462
354     356
356     353
146     351
135     209
364     183
452      19
451       3
265       3
352       1
122       1
120       1
dtype: int64
```

- Verificando o sexo e os tipos dos envolvidos nos acidentes, identifica-se a predominância do sexo masculino nos acidentes e, principalmente, como condutor. O sexo feminino tem maior participação nos acidentes como passageiro. Ver Figura 50.

**Figura 50: Acidentes por sexo e envolvido, período total
(autor)**

```
Acidentes_por_sexo_envolvido = acidentes_mg.groupby(['sexo']).size().sort_values(ascending=False)
print("Acidentes por sexo dos envolvidos em MG (2017 a 2020): " + str(Acidentes_por_sexo_envolvido))
```

```
Acidentes por sexo dos envolvidos em MG (2017 a 2020): sexo
Masculino    65633
Feminino     20793
Não Informado   4980
Ignorado      383
dtype: int64
```

```
Acidentes_por_sexo_e_envolvido = acidentes_mg.groupby(['tipo_envolvido','sexo']).size()
print("Acidentes por sexo por envolvido em MG (2017 a 2020): " + str(Acidentes_por_sexo_e_envolvido))
```

```
Acidentes por sexo por envolvido em MG (2017 a 2020): tipo_envolvido  sexo
Cavaleiro      Feminino      1
                           Masculino    25
                           Não Informado  1
Condutor        Feminino    4980
                           Ignorado    338
                           Masculino   51556
                           Não Informado 2209
Passageiro      Feminino   15435
                           Ignorado    43
                           Masculino   12821
                           Não Informado 125
Pedestre        Feminino    377
                           Ignorado    2
                           Masculino   1231
                           Não Informado 13
Testemunha      Feminino   2632
                           Não Informado 2632
dtype: int64
```

- Na análise e processamento dos dados de acidentes, também foram gerados indicadores por ano, apresentando dados relativos às extensões das BR; percentuais de acidentes, mortes e feridos; acidentes/km e mortes/km; totais por BR dos acidentes fatais, com

feridos e sem vítimas; e a geração do indicador de gravidade. Ver Figura 51.

Figura 51: Geração de indicadores relacionados aos acidentes de MG, por ano (autor)

```
#Geração de indicadores relacionados aos acidentes de MG

#Classificação por gravidade dos acidentes
#print(str(datatran_mg.groupby(['br','classificacao_acidente']).size()))
datatran_mg['data_inversa'] = datatran_mg.data_inversa.astype('datetime64')
class_acid = pd.DataFrame(datatran_mg.groupby([datatran_mg['data_inversa'].dt.year.rename('Ano'), datatran_mg['br'], datatran_mg['id']].rename(columns={'br':'BR'}, inplace=True)
class_acid.rename(columns={'id':'Acidentes'}, inplace=True)
class_acid['Sem_Vitimas'] = '0'
class_acid['Com_Vitimas_Feridas'] = '0'
class_acid['Com_Vitimas_Fatais'] = '0'
class_acid['BR'] = class_acid.BR.astype('object')
class_acid['Sem_Vitimas'] = class_acid.Sem_Vitimas.astype('int64')
class_acid['Com_Vitimas_Feridas'] = class_acid.Com_Vitimas_Feridas.astype('int64')
class_acid['Com_Vitimas_Fatais'] = class_acid.Com_Vitimas_Fatais.astype('int64')
class_acid.loc[class_acid['classificacao_acidente'] == 'Com_Vitimas_Fatais', 'Com_Vitimas_Fatais'] = class_acid['Acidentes']
class_acid.loc[class_acid['classificacao_acidente'] == 'Com_Vitimas_Feridas', 'Com_Vitimas_Feridas'] = class_acid['Acidentes']
class_acid.loc[class_acid['classificacao_acidente'] == 'Sem_Vitimas', 'Sem_Vitimas'] = class_acid['Acidentes']
class_acid.drop(columns=['classificacao_acidente'], axis=1, inplace=True)
class_acid.drop(columns=['Acidentes'], axis=1, inplace=True)
class_acidentes = pd.DataFrame(class_acid.groupby([class_acid['Ano'], class_acid['BR']]).agg({'Sem_Vitimas' : 'sum', 'Com_Vitimas_Feridas' : 'sum', 'Com_Vitimas_Fatais' : 'sum'}))
#class_acidentes.head(10)

#Extensões por trecho de BR
snv_2017_mg = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\SNV_201801B_3.csv", sep=';', decimal=',', encoding = 'utf_8')
snv_2018_mg = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\snv_201903a_3.csv", sep=';', decimal=',', encoding = 'utf_8')
snv_2019_mg = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\SNV_202001A_3.csv", sep=';', decimal=',', encoding = 'utf_8')
snv_2020_mg = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\SNV_202101A_3.csv", sep=';', decimal=',', encoding = 'utf_8')
snv_2017_mg.drop(columns=['Tipo de trecho', 'Desc Coinc', 'Código', 'Local de Inicio', 'Local de Fim', 'km inicial', 'km final'], axis=1)
snv_2018_mg.drop(columns=['Tipo de trecho', 'Desc Coinc', 'Código', 'Local de Inicio', 'Local de Fim', 'km inicial', 'km final'], axis=1)
snv_2019_mg.drop(columns=['Tipo de trecho', 'Desc Coinc', 'Código', 'Local de Inicio', 'Local de Fim', 'km inicial', 'km final', 'Suposta Causa'], axis=1)
snv_2020_mg.drop(columns=['Tipo de trecho', 'Desc Coinc', 'Código', 'Local de Inicio', 'Local de Fim', 'km inicial', 'km final', 'Suposta Causa'], axis=1)

#Extensões por BR
snv_2017_mg_br = pd.DataFrame(snv_2017_mg.groupby('BR', as_index = False)[['Extensão']].sum()).reset_index()
snv_2018_mg_br = pd.DataFrame(snv_2018_mg.groupby('BR', as_index = False)[['Extensão']].sum()).reset_index()
snv_2019_mg_br = pd.DataFrame(snv_2019_mg.groupby('BR', as_index = False)[['Extensão']].sum()).reset_index()
snv_2020_mg_br = pd.DataFrame(snv_2020_mg.groupby('BR', as_index = False)[['Extensão']].sum()).reset_index()
snv_2017_mg_br.rename(columns={'index':'Ano'}, inplace=True)
snv_2017_mg_br['Ano'] = '2017'
snv_2018_mg_br.rename(columns={'index':'Ano'}, inplace=True)
snv_2018_mg_br['Ano'] = '2018'
snv_2019_mg_br.rename(columns={'index':'Ano'}, inplace=True)
snv_2019_mg_br['Ano'] = '2019'
snv_2020_mg_br.rename(columns={'index':'Ano'}, inplace=True)
snv_2020_mg_br['Ano'] = '2020'

#Excluindo BRs sem acidentes registrados
snv_mg_br = pd.concat([snv_2017_mg_br, snv_2018_mg_br, snv_2019_mg_br, snv_2020_mg_br], join="inner")
snv_mg_br.set_index('Ano', inplace = True)
snv_mg_br.reset_index(inplace=True)
snv_mg_br.drop(snv_mg_br.index[[0,4,5,9,11,13,15,16,21,22,24,25,26,27,28,29,31,32,34,35,36,37,38,39,40,41,42,43,44,45,46]], inplace=True)
snv_mg_br.set_index('Ano', inplace = True)
snv_mg_br.reset_index(inplace=True)
snv_mg_br.drop(snv_mg_br.index[[19,20,24,26,28,30,31,36,37,39,40,41,42,43,45,46,47,49,50,51,52,53,54,55,56,57,58,59,60,61]], inplace=True)
snv_mg_br.set_index('Ano', inplace = True)
snv_mg_br.reset_index(inplace=True)
snv_mg_br.drop(snv_mg_br.index[[37,40,42,46,47,52,53,55,56,57,58,59,60,62,63,65,66,67,68,69,70,71,72,73,74,75,76,77]], inplace=True)
snv_mg_br.set_index('Ano', inplace = True)
snv_mg_br.reset_index(inplace=True)
snv_mg_br.drop(snv_mg_br.index[[53,54,55,58,60,64,70,71,73,74,75,76,77,78,79,80,81,83,84,85,86,87,88,89,90,91,92,93,94,95]], inplace=True)
snv_mg_br.set_index('Ano', inplace = True)
snv_mg_br.reset_index(inplace=True)
#print(snv_mg_br)
snv_mg_br.to_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\snv_mg_br.csv", index=False)

#Preparação para a geração dos indicadores
datatran_mg['data_inversa'] = datatran_mg.data_inversa.astype('datetime64')
#datatran_mg.info()
temp_mg = pd.DataFrame(datatran_mg.groupby([datatran_mg['data_inversa'].dt.year.rename('Ano'), datatran_mg['br']]).agg({'id': 'count', 'feridos': 'sum', 'mortos': 'sum', 'pessoas': 'sum', 'BR': 'count'}))
temp_mg.rename(columns={'id': 'Acidentes', 'feridos': 'Feridos', 'mortos': 'Mortes', 'pessoas': 'Envolvidos', 'BR': 'BR'}, inplace=True)
temp_mg['BR'] = temp_mg.BR.astype('object')
#print(temp_mg)
snv_mg_br.drop(columns=['BR'], axis=1, inplace=True)
temp_mg.drop(columns=['Ano'], axis=1, inplace=True)
temp_indices = pd.DataFrame(pd.concat([snv_mg_br, temp_mg], axis=1, join="inner")).reset_index()
temp_indices.drop(columns=['index'], axis=1, inplace=True)
```

```

#print(temp_indices)
#temp_indices.info()

#Gerando os indicadores - dados de 2017
temp_2017 = temp_indices.loc[(temp_indices['Ano'] == "2017")]
#print(temp_2017.loc[temp_2017['Ano'] == "2017", 'Extensão'].sum())
temp_2017['%_Extensão_BR'] = (temp_2017['Extensão'] / temp_2017['Extensão'].sum())*100
temp_2017['%_Acidente_BR'] = (temp_2017['Acidentes'] / temp_2017['Acidentes'].sum())*100
temp_2017['%_Feridos_BR'] = (temp_2017['Feridos'] / temp_2017['Feridos'].sum())*100
temp_2017['%_Mortes_BR'] = (temp_2017['Mortes'] / temp_2017['Mortes'].sum())*100
temp_2017['Acidente_KM'] = (temp_2017['Acidentes'] / temp_2017['Extensão'])
temp_2017['Mortes_KM'] = (temp_2017['Mortes'] / temp_2017['Extensão'])
temp_2017 = temp_2017.reindex(columns=['Ano', 'BR', 'Extensão', 'Acidentes', 'Envolvidos', 'Feridos', 'Mortes', '%_Extensão_BR'])
temp_2017.drop(columns=['index'], axis=1, inplace=True)
#print(temp_2017)
#temp_2017.head(50)

#Gerando os indicadores - dados de 2018
temp_2018 = temp_indices.loc[(temp_indices['Ano'] == "2018")]
#print(temp_2018.loc[temp_2018['Ano'] == "2018", 'Extensão'].sum())
temp_2018['%_Extensão_BR'] = (temp_2018['Extensão'] / temp_2018['Extensão'].sum())*100
temp_2018['%_Acidente_BR'] = (temp_2018['Acidentes'] / temp_2018['Acidentes'].sum())*100
temp_2018['%_Feridos_BR'] = (temp_2018['Feridos'] / temp_2018['Feridos'].sum())*100
temp_2018['%_Mortes_BR'] = (temp_2018['Mortes'] / temp_2018['Mortes'].sum())*100
temp_2018['Acidente_KM'] = (temp_2018['Acidentes'] / temp_2018['Extensão'])
temp_2018['Mortes_KM'] = (temp_2018['Mortes'] / temp_2018['Extensão'])
temp_2018 = temp_2018.reindex(columns=['Ano', 'BR', 'Extensão', 'Acidentes', 'Envolvidos', 'Feridos', 'Mortes', '%_Extensão_BR'])
temp_2018.drop(columns=['index'], axis=1, inplace=True)
#print(temp_2018)
#temp_2018.head(50)

#Gerando os indicadores - dados de 2019
temp_2019 = temp_indices.loc[(temp_indices['Ano'] == "2019")]
#print(temp_2019.loc[temp_2019['Ano'] == "2019", 'Extensão'].sum())
temp_2019['%_Extensão_BR'] = (temp_2019['Extensão'] / temp_2019['Extensão'].sum())*100
temp_2019['%_Acidente_BR'] = (temp_2019['Acidentes'] / temp_2019['Acidentes'].sum())*100
temp_2019['%_Feridos_BR'] = (temp_2019['Feridos'] / temp_2019['Feridos'].sum())*100
temp_2019['%_Mortes_BR'] = (temp_2019['Mortes'] / temp_2019['Mortes'].sum())*100
temp_2019['Acidente_KM'] = (temp_2019['Acidentes'] / temp_2019['Extensão'])
temp_2019['Mortes_KM'] = (temp_2019['Mortes'] / temp_2019['Extensão'])
temp_2019 = temp_2019.reindex(columns=['Ano', 'BR', 'Extensão', 'Acidentes', 'Envolvidos', 'Feridos', 'Mortes', '%_Extensão_BR'])
temp_2019.drop(columns=['index'], axis=1, inplace=True)
#print(temp_2019)
#temp_2019.head(50)

#Gerando os indicadores - dados de 2020
temp_2020 = temp_indices.loc[(temp_indices['Ano'] == "2020")]
#print(temp_2019.loc[temp_2020['Ano'] == "2020", 'Extensão'].sum())
temp_2020['%_Extensão_BR'] = (temp_2020['Extensão'] / temp_2020['Extensão'].sum())*100
temp_2020['%_Acidente_BR'] = (temp_2020['Acidentes'] / temp_2020['Acidentes'].sum())*100
temp_2020['%_Feridos_BR'] = (temp_2020['Feridos'] / temp_2020['Feridos'].sum())*100
temp_2020['%_Mortes_BR'] = (temp_2020['Mortes'] / temp_2020['Mortes'].sum())*100
temp_2020['Acidente_KM'] = (temp_2020['Acidentes'] / temp_2020['Extensão'])
temp_2020['Mortes_KM'] = (temp_2020['Mortes'] / temp_2020['Extensão'])
temp_2020 = temp_2020.reindex(columns=['Ano', 'BR', 'Extensão', 'Acidentes', 'Envolvidos', 'Feridos', 'Mortes', '%_Extensão_BR'])
temp_2020.drop(columns=['index'], axis=1, inplace=True)
#print(temp_2020)
#temp_2020.head(50)

#Consolidando os indicadores de acidentalidade em um único dataframe
indices_temp = pd.concat([temp_2017, temp_2018, temp_2019, temp_2020], sort = False).reset_index()
indices_temp.drop(columns=['index'], axis=1, inplace=True)
class_acidentes.drop(columns=['Ano'], axis=1, inplace=True)
class_acidentes.drop(columns=['BR'], axis=1, inplace=True)
indices_acidentes = pd.concat([indices_temp, class_acidentes], axis=1, sort = False).reset_index()
indices_acidentes.drop(columns=['index'], axis=1, inplace=True)
indices_acidentes['Gravidade'] = (indices_acidentes['Sem_Vítimas'] + (5 * indices_acidentes['Com_Vítimas_Feridas'])+(13 * indices_acidentes['Mortes']))
indices_acidentes.to_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\índices_acidentes.csv", index = False, header = True, sep=';')

```

- Foi então gerado dataframe com os índices dos acidentes armazenado em um arquivo csv. O dataframe possui 66 registros de dados, sendo um para cada BR e por ano. A BR-381, em todos os anos, apresentou o mais alto índice de gravidade dos acidentes. Depois, estão as BRs “040, 050 e 116”, com os mais altos índices de gravidade dos acidentes. Em termos de periculosidade (acidente/km), o ranking também é liderado pelas BRs “381, 040 e 050”, com destaque para a 381. Isto pode indicar as BRs que

devem ser foco para processos de educação no trânsito e de fiscalização, além de verificar as condições de engenharia. Ver Figura 52 e Quadro 06.

Figura 52: indicadores relacionados aos acidentes de MG, por ano (autor)

Ano	BR	Extensão	Acidentes	Envolvidos	Feridos	Mortes	%_Extensão_BR	%_Acidente_BR	Acidente_KM	%_Feridos_BR	%_Mortes_BR	Mortes_KM
0	2017	40	893.90	2374	5479	2494	145	8.7231	18.6767	2.6558	18.9586	16.6958
1	2017	50	269.80	673	1333	619	13	2.6328	5.2946	2.4944	4.7054	1.4960
2	2017	116	818.10	1626	3749	1841	170	7.9834	12.7921	1.9875	13.9947	19.5627
3	2017	135	834.70	208	553	260	34	8.1454	1.6364	0.2492	1.9764	3.9125
4	2017	146	726.80	114	256	128	6	7.0925	0.8969	0.1569	0.9730	0.6904
5	2017	153	239.90	379	855	362	29	2.3411	2.9817	1.5798	2.7518	3.3372
6	2017	251	1015.70	282	736	340	50	9.9117	2.2186	0.2776	2.5846	5.7537
7	2017	262	1082.90	1381	3012	1495	97	10.5675	10.8646	1.2753	11.3645	11.1623
8	2017	267	534.70	293	759	397	30	5.2179	2.3051	0.5480	3.0179	3.4522
9	2017	354	774.50	106	241	90	4	7.5579	0.8339	0.1369	0.6842	0.4603
10	2017	356	291.90	108	230	121	2	2.9495	0.9497	0.2700	0.9198	0.2201

Sem_Vitimas	Com_Vitimas_Feridas	Com_Vitimas_Fatais	Gravidade
720	1544	110	11.0415
236	426	11	9.2995
385	1111	130	9.3265
58	125	25	1.2076
36	72	6	0.6522
144	210	25	6.3318
87	162	33	1.3055
399	902	80	5.4936
63	205	25	2.6426
35	67	4	0.5449
21	85	2	1.6170

Quadro 6: Índices de Acidentes, por BR, por ano (dados da pesquisa)

Ano	BR	Exten-são	Aci-den-tes	En-vol-vi-dos	Fe-ri-dos	Mo-rt-es	%_Ext-en-são_BR	%_Aci-den-te_BR	Aci-den-te_KM	%_Fer-idos_BR	%_Mo-rtes_BR	Mor-tes_KM	Sem_Vi-ti-mas	Com_Vi-ti-mas_Fe-ridas	Com_Vi-ti-mas_Fatais	Gravi-dade
2017	40	893,9	2374	5479	2494	145	8,72	18,68	2,66	18,96	16,69	0,16	720	1544	110	11,04
2017	50	269,8	673	1333	619	13	2,63	5,29	2,49	4,71	1,50	0,05	236	426	11	9,30
2017	116	818,1	1626	3749	1841	170	7,98	12,79	1,99	13,99	19,56	0,21	385	1111	130	9,33
2017	135	834,7	208	553	260	34	8,15	1,64	0,25	1,98	3,91	0,04	58	125	25	1,21
2017	146	726,8	114	256	128	6	7,09	0,90	0,16	0,97	0,69	0,01	36	72	6	0,65
2017	153	239,9	379	855	362	29	2,34	2,98	1,58	2,75	3,34	0,12	144	210	25	6,33
2017	251	101,7	282	736	340	50	9,91	2,22	0,28	2,58	5,75	0,05	87	162	33	1,31
2017	262	108,2	1381	3012	1495	97	10,57	10,86	1,28	11,36	11,16	0,09	399	902	80	5,49
2017	267	534,7	293	759	397	30	5,22	2,31	0,55	3,02	3,45	0,06	63	205	25	2,64
2017	354	774,5	106	241	90	4	7,56	0,83	0,14	0,68	0,46	0,01	35	67	4	0,54
2017	356	291,9	108	230	121	2	2,85	0,85	0,37	0,92	0,23	0,01	21	85	2	1,62

2017	364	287	69	146	74	2	2,80	0,54	0,24	0,56	0,23	0,0 1	20	47	2	0,98
2017	365	869 ,9	756	1804	849	58	8,49	5,95	0,87	6,45	6,67	0,0 7	192	520	44	3,87
2017	381	108 3,9	418 0	9210	391 5	22 2	10,58	32,88	3,86	29,76	25,5 5	0,2 0	1682	2325	173	14,3 5
2017	452	306 ,2	17	33	20	0	2,99	0,13	0,06	0,15	0,00	0,0 0	5	12	0	0,21
2017	459	217 ,6	145	385	150	7	2,12	1,14	0,67	1,14	0,81	0,0 3	48	90	7	2,71
2018	40	894 ,4	177 2	4510	218 5	13 0	8,71	19,57	1,98	19,68	18,7 6	0,1 5	350	1318	104	9,27
2018	50	270 ,2	531	1179	626	18	2,63	5,86	1,97	5,64	2,60	0,0 7	123	393	15	8,45
2018	116	818 ,1	113 2	2627	141 3	10 5	7,96	12,50	1,38	12,73	15,1 5	0,1 3	155	888	89	7,03
2018	135	834 ,7	1	3	0	1	8,13	0,01	0,00	0,00	0,14	0,0 0	0	0	1	0,02
2018	146	632 ,1	77	215	120	9	6,15	0,85	0,12	1,08	1,30	0,0 1	13	56	8	0,63
2018	153	239 ,9	240	542	273	16	2,34	2,65	1,00	2,46	2,31	0,0 7	58	167	15	4,54
2018	251	101 5,7	211	599	329	52	9,89	2,33	0,21	2,96	7,50	0,0 5	41	142	28	1,10
2018	262	108 2,9	101 3	2369	131 3	89	10,54	11,19	0,94	11,83	12,8 4	0,0 8	177	764	72	4,56
2018	267	535 ,3	192	461	264	27	5,21	2,12	0,36	2,38	3,90	0,0 5	33	138	21	1,86
2018	354	775 ,4	87	260	99	2	7,55	0,96	0,11	0,89	0,29	0,0 0	20	65	2	0,48
2018	356	288 ,3	82	183	96	6	2,81	0,91	0,28	0,86	0,87	0,0 2	6	71	5	1,48
2018	364	287	52	109	54	5	2,79	0,57	0,18	0,49	0,72	0,0 2	11	36	5	0,89
2018	365	869 ,9	609	1548	829	55	8,47	6,73	0,70	7,47	7,94	0,0 6	104	460	45	3,44
2018	381	108 4,2	294 4	6981	334 6	17 1	10,55	32,51	2,72	30,14	24,6 8	0,1 6	728	2082	134	11,8 8
2018	451	429 ,1	3	5	3	0	4,18	0,03	0,01	0,03	0,00	0,0 0	0	3	0	0,03
2018	459	215 ,4	109	266	151	7	2,10	1,20	0,51	1,36	1,01	0,0 3	14	88	7	2,53
2019	40	894 ,5	173 5	4265	223 7	13 8	8,08	19,91	1,94	19,57	20,3 5	0,1 5	241	1381	113	9,63
2019	50	270 ,2	506	1067	587	12	2,44	5,81	1,87	5,14	1,77	0,0 4	88	406	12	8,42
2019	116	818 ,1	113 5	2773	152 2	10 9	7,39	13,03	1,39	13,32	16,0 8	0,1 3	117	926	92	7,26
2019	120	808 ,1	1	5	2	0	7,30	0,01	0,00	0,02	0,00	0,0 0	0	1	0	0,01
2019	122	280 ,9	1	3	1	0	2,54	0,01	0,00	0,01	0,00	0,0 0	0	1	0	0,02
2019	146	632 ,1	89	240	145	1	5,71	1,02	0,14	1,27	0,15	0,0 0	13	75	1	0,63
2019	153	239 ,9	230	525	283	20	2,17	2,64	0,96	2,48	2,95	0,0 8	34	179	17	4,79
2019	251	101 5,7	201	480	253	32	9,18	2,31	0,20	2,21	4,72	0,0 3	41	132	28	1,05
2019	262	108 3,9	956	2254	120 3	86	9,80	10,97	0,88	10,53	12,6 8	0,0 8	146	741	69	4,38
2019	265	665	2	2	2	0	6,01	0,02	0,00	0,02	0,00	0,0 0	0	2	0	0,02
2019	267	535 ,3	182	416	238	25	4,84	2,09	0,34	2,08	3,69	0,0 5	18	140	24	1,92
2019	354	775 ,4	86	192	92	5	7,01	0,99	0,11	0,80	0,74	0,0 1	15	66	5	0,53
2019	356	288 ,3	92	202	123	4	2,61	1,06	0,32	1,08	0,59	0,0 1	3	86	3	1,64
2019	364	287 ,7	34	74	37	4	2,60	0,39	0,12	0,32	0,59	0,0 1	7	23	4	0,60
2019	365	869 ,9	702	1963	106 1	75	7,86	8,06	0,81	9,28	11,0 6	0,0 9	108	537	57	4,06
2019	381	107 8,7	265 0	6630	350 5	14 4	9,75	30,41	2,46	30,67	21,2 4	0,1 3	430	2093	127	11,6 3
2019	452	306 ,2	2	3	0	1	2,77	0,02	0,01	0,00	0,15	0,0 0	1	0	1	0,05

2019	459	215 ,4	109	270	138	22	1,95	1,25	0,51	1,21	3,24	0,1 0	12	80	17	2,94
2020	40	894 ,5	166 8	4042	204 0	10 7	8,84	19,96	1,86	19,59	14,9 2	0,1 2	244	1326	98	9,11
2020	50	270 ,4	538	1070	559	24	2,67	6,44	1,99	5,37	3,35	0,0 9	109	405	24	9,05
2020	116	818 ,73	106 7	2418	134 2	11 5	8,09	12,77	1,30	12,89	16,0 4	0,1 4	109	867	91	6,87
2020	146	603	71	149	76	9	5,96	0,85	0,12	0,73	1,26	0,0 1	13	49	9	0,62
2020	153	239 ,9	215	483	237	26	2,37	2,57	0,90	2,28	3,63	0,1 1	45	147	23	4,50
2020	251	101 5,9	255	891	405	32	10,04	3,05	0,25	3,89	4,46	0,0 3	45	183	27	1,29
2020	262	108 3,9	965	2116	116 3	84	10,71	11,55	0,89	11,17	11,7 2	0,0 8	156	748	61	4,33
2020	265	658 ,3	1	1	1	0	6,51	0,01	0,00	0,01	0,00	0,0 0	0	1	0	0,01
2020	267	534	148	310	184	34	5,28	1,77	0,28	1,77	4,74	0,0 6	18	104	26	1,64
2020	352	485 ,2	1	6	1	0	4,80	0,01	0,00	0,01	0,00	0,0 0	0	1	0	0,01
2020	354	775 ,4	77	179	99	4	7,66	0,92	0,10	0,95	0,56	0,0 1	8	65	4	0,50
2020	356	287 ,96	71	140	82	2	2,85	0,85	0,25	0,79	0,28	0,0 1	9	60	2	1,16
2020	364	287 ,7	28	72	45	0	2,84	0,34	0,10	0,43	0,00	0,0 0	3	25	0	0,44
2020	365	868 ,2	635	1649	835	91	8,58	7,60	0,73	8,02	12,6 9	0,1 0	110	468	57	3,68
2020	381	107 9	251 7	6042	322 7	17 8	10,66	30,12	2,33	30,98	24,8 3	0,1 6	372	1998	147	11, 7
2020	459	215 ,4	99	219	119	11	2,13	1,18	0,46	1,14	1,53	0,0 5	15	76	8	2,32

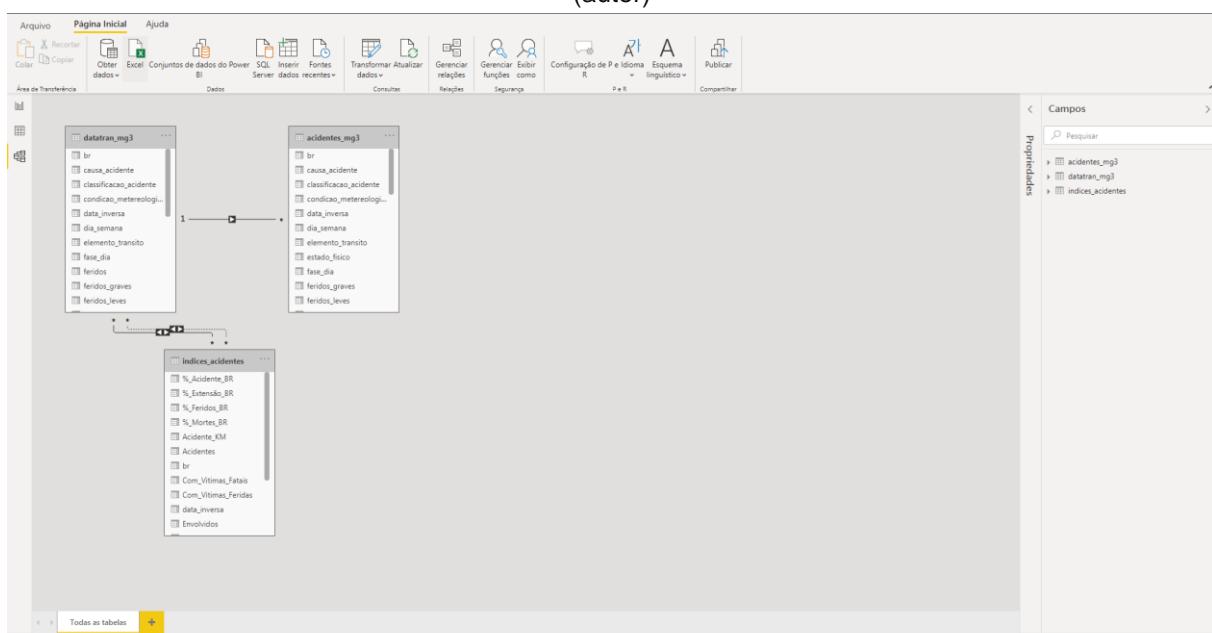
- Análise e exploração com o MS-PowerBI:

- Foram desenvolvidos painéis analíticos no MS-PowerBI como mais uma ferramenta de exploração de dados.
- A carga de dados no MS-PowerBI foi efetuada com os arquivos gerados e produzidos nas etapas de montagem da base de dados e processamento de dados com o Python.
- Os arquivos empregados na carga do MS-PowerBI foram: datatran_mg3.csv, acidentes_mg3.csv e indices_acidentes.csv, conforme pode ser verificado na Figura 53.
- O modelo de relacionamento dos dados a partir da carga ficou conforme está apresentado na Figura 54. Foi utilizado o campo de 'id' para relacionar os dados presentes nos arquivos datatran_mg3.csv e acidentes_mg3.csv. No caso entre os dados dos arquivos datatran_mg3.csv e indices_acidentes.csv, foram

utilizadas duas relações (uma pelo atributo ‘data_inversa’ e outra pelo atributo ‘br’).

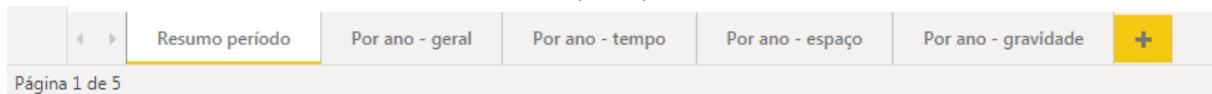
**Figura 53: Carga de arquivos no MS-PowerBI
(autor)**

**Figura 54: Modelo de relações no MS-PowerBI
(autor)**



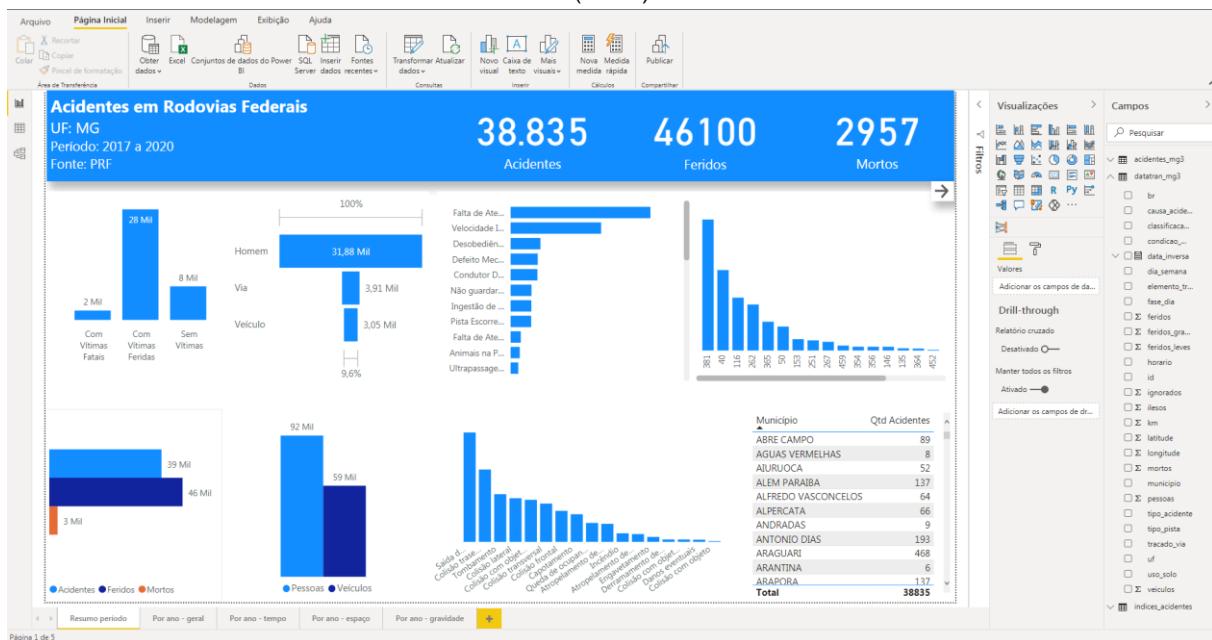
- Foram desenvolvidos 05 (cinco) dashboards, sendo 01 como um resumo geral do período de 2017 a 2020 e os outros 04 com filtros para seleção por ano e com temas específicos. Ver Figura 55.

**Figura 55: Dashboards do MS-PowerBI
(autor)**



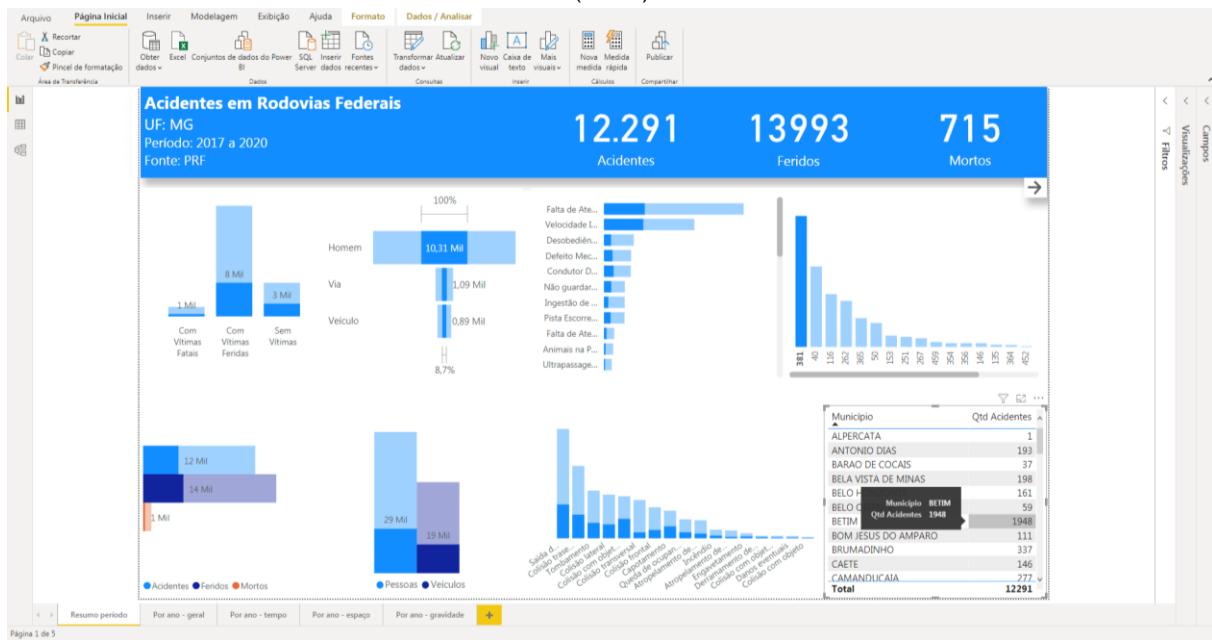
- No dashboard “Resumo período” (Figura 56), foi construída uma apresentação em que se pode verificar a quantidade total de acidentes, feridos e mortos de todo o período (todos estes dados batem com o que foi obtido e apresentado pelos scripts de Python – Figuras 28, 29 e 30). Nesse dashboard de resumo, podem ser consultados acidentes por classificação (gravidade – sem ou com vítimas, fatais ou feridas), por elemento de trânsito, pela causa presumível e tipo, BR e município.

**Figura 56: Dashboard “Resumo período” do MS-PowerBI
(autor)**



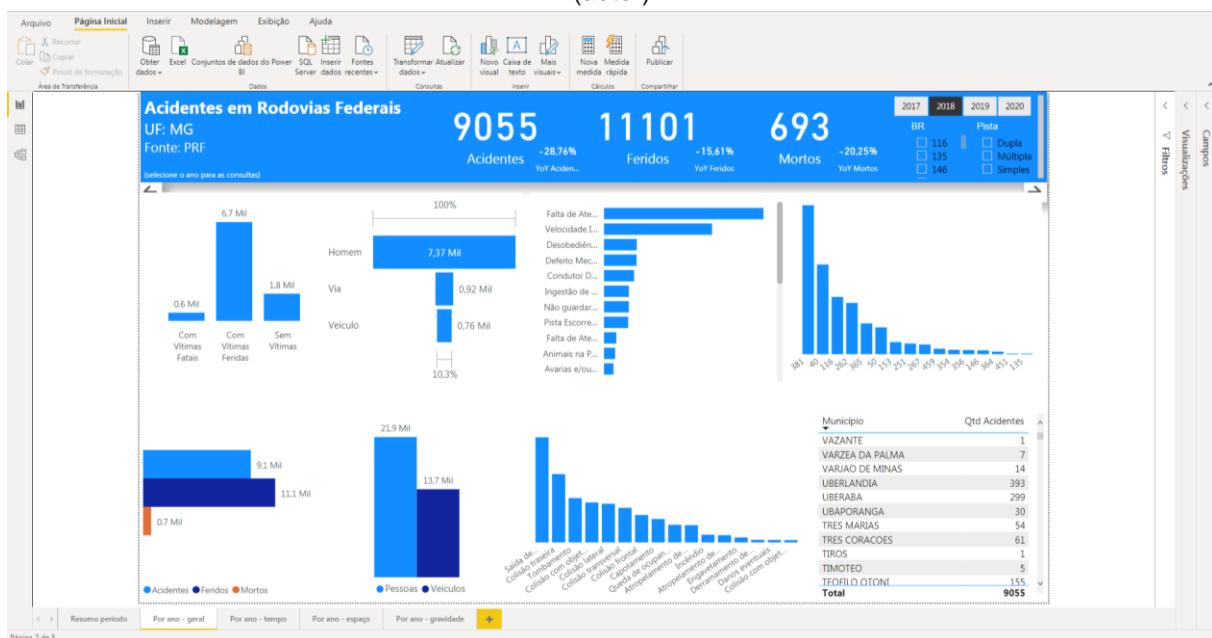
- Com o dashboard “Resumo período” (Figura 57), selecionando a BR-381, que é a BR da malha rodoviária federal da UF de MG que apresentou maior quantidade de acidentes, percebe-se que o município que teve maior participação em acidentes foi Betim, com total de 1.948 ocorrências, com as principais causas presumíveis de “Falta de Atenção à Condução” e “Velocidade Incompatível”, e principais tipos de “Saída do leito carroçável” e “Colisão traseira”.

Figura 57: Dashboard “Resumo período” do MS-PowerBI com seleção da BR-381 (autor)



- No dashboard “Por ano - geral” (Figura 58), os objetos apresentados são similares ao do dashboard “Resumo período”, contudo com a inserção de filtros por ano, por BR e por tipo de pista, visando efetuar seleções e análises.

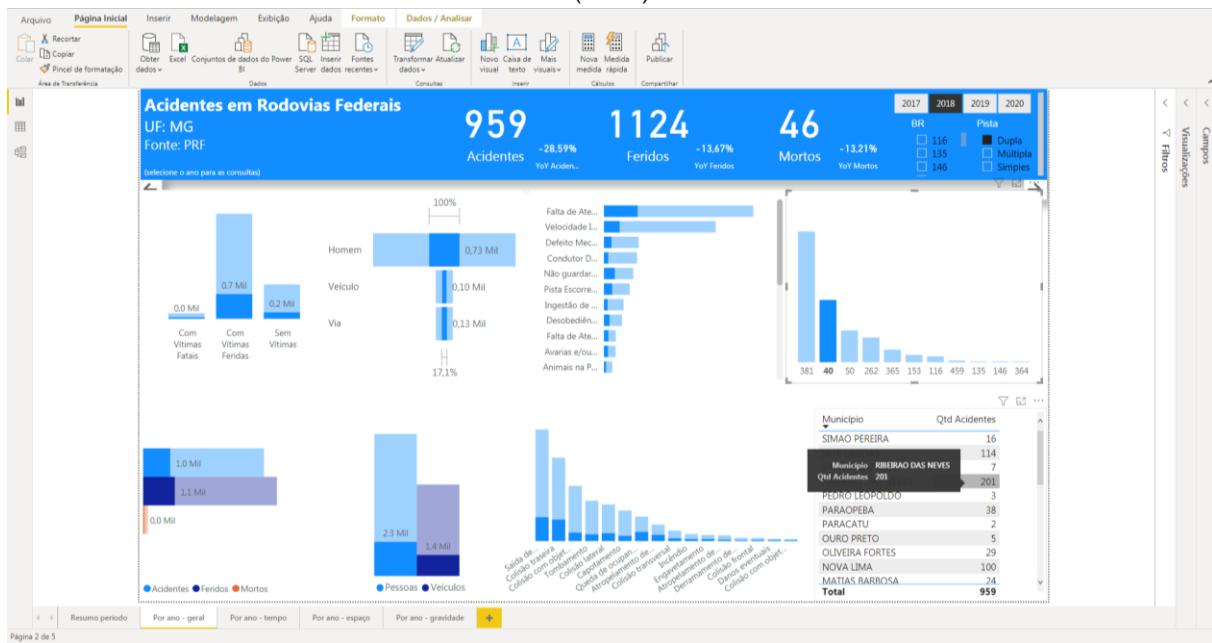
Figura 58: Dashboard “Por ano- geral” do MS-PowerBI (autor)



- Com o dashboard “Por ano - geral” (Figura 59), selecionando o ano de 2018, a BR-040, pista do tipo dupla, o município Ribeirão das

Neves é o que apresentou maior ocorrência de acidentes. Verifica-se o total de acidentes no ano selecionado (959) e que foi 28,59% menor do que o ano anterior de 2017 (YoY – year over year), bem como total de feridos (1124, 13,67% menor que em 2017) e mortos (46, 13,21% menor que em 2017).

Figura 59: Dashboard “Por ano- geral” do MS-PowerBI com seleção de ano, BR e tipo de pista (autor)



- No dashboard “Por ano - tempo” (Figura 60), foram desenvolvidos objetos que apresentam dados por mês, dia da semana e fase do dia, além das condições meteorológicas. Também foram inseridos objetos com dados de sexo dos envolvidos nos acidentes e do tipo de envolvido.
- Com o dashboard “Por ano - tempo” (Figura 61), selecionando filtro do ano para 2019 e a BR-116, verifica-se que o sábado é o dia da semana que apresentou a maior quantidade de acidentes na seleção. Na sequência, inserindo o sábado no filtro da seleção, percebe-se que o mês de agosto foi o que apresentou maior quantidade de acidentes no dia de sábado (31 acidentes), inclusive superior ao ano anterior de 2018 (19 acidentes).

Figura 60: Dashboard “Por ano - tempo” do MS-PowerBI
(autor)

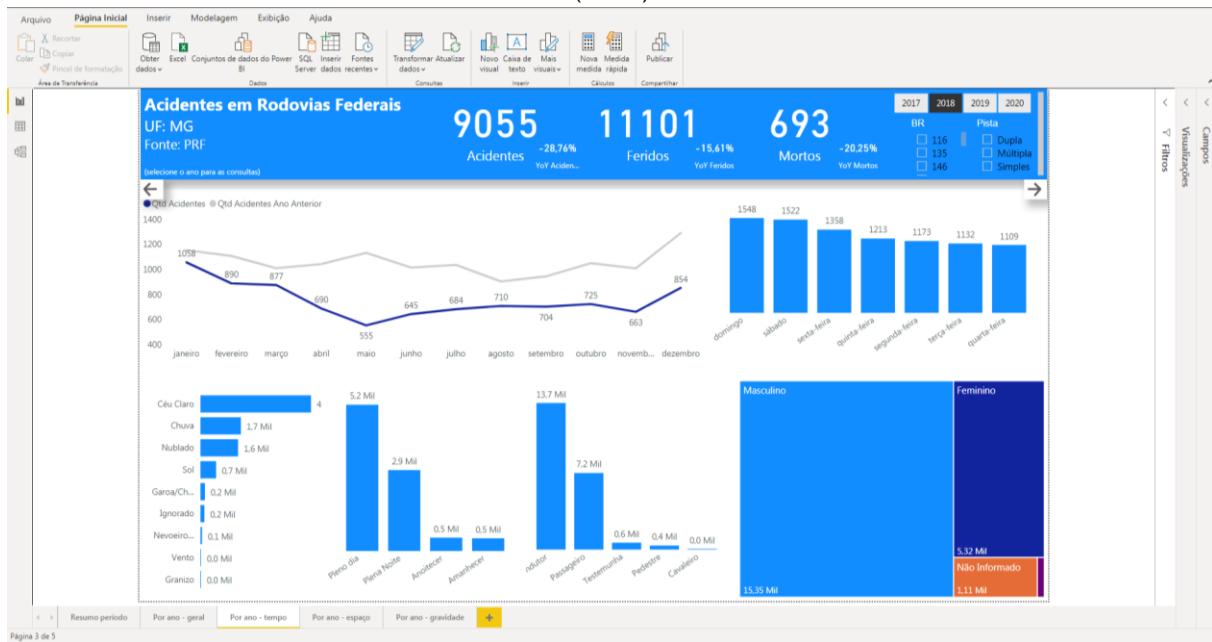
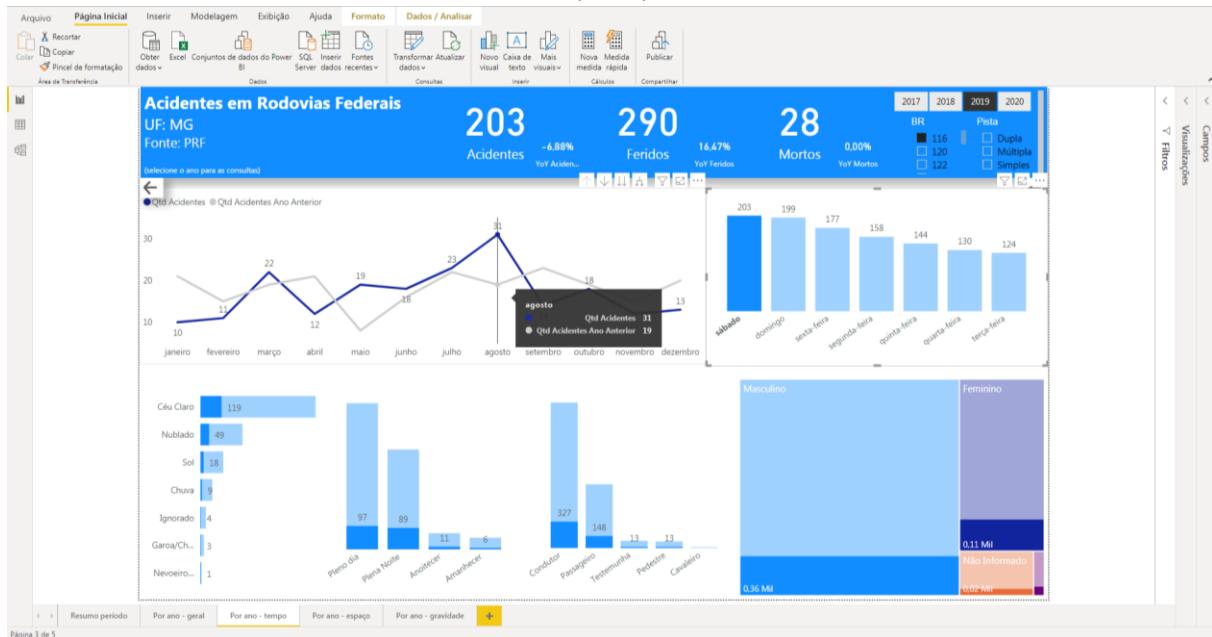
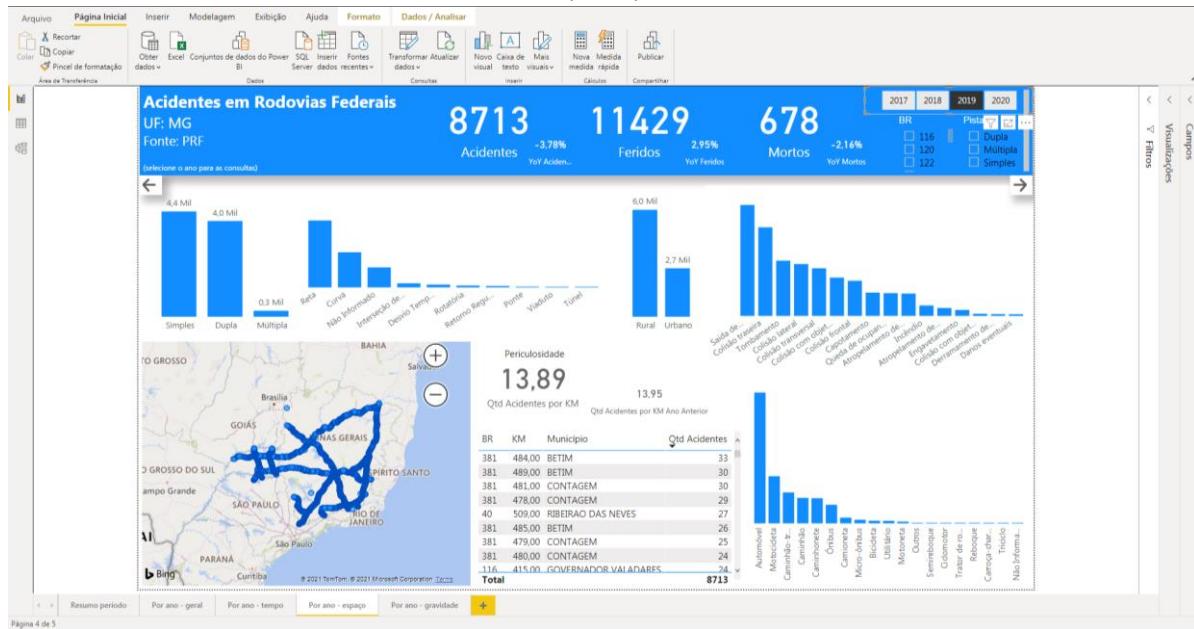


Figura 61: Dashboard “Por ano - tempo” do MS-PowerBI com seleção de ano, BR e dia da semana
(autor)



- No dashboard “Por ano - espaço” (Figura 62), foram dispostos objetos de tipo de pista, traçado da via, meio da via (urbano ou rural), tipo de veículo envolvido no acidente, e um mapa com a localização geográfica do quilômetro da rodovia em que o acidente foi registrado.

**Figura 62: Dashboard “Por ano - espaço” do MS-PowerBI
(autor)**



- Com o dashboard “Por ano - espaço” (Figura 63), selecionando filtro do ano para 2019 e a BR-116, percebe-se que o tipo de pista simples apresentou maior quantidade acidentes. Selecionando, então, a pista simples e o quilômetro 415,00 que apresentou maior quantidade de acidentes (Figura 64), o mapa é aproximado para apresentar a área dos acidentes.

**Figura 63: Dashboard “Por ano - espaço” do MS-PowerBI com seleção de ano, BR e tipo de pista
(autor)**

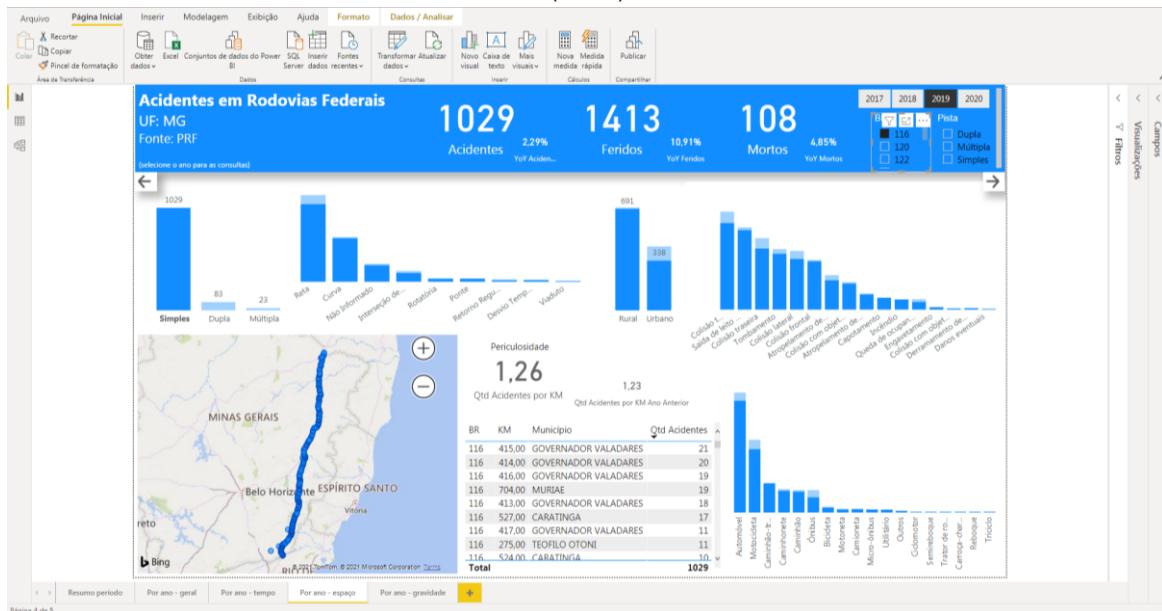
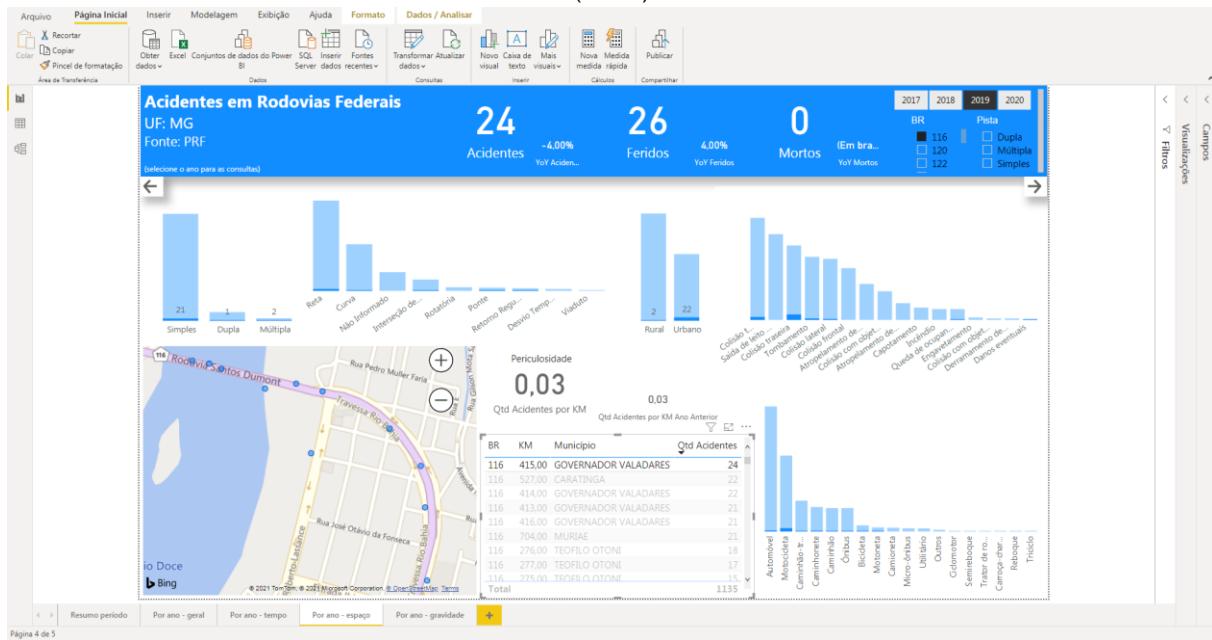


Figura 64: Dashboard “Por ano - espaço” do MS-PowerBI com seleção de ano, BR, tipo de pista e KM (autor)



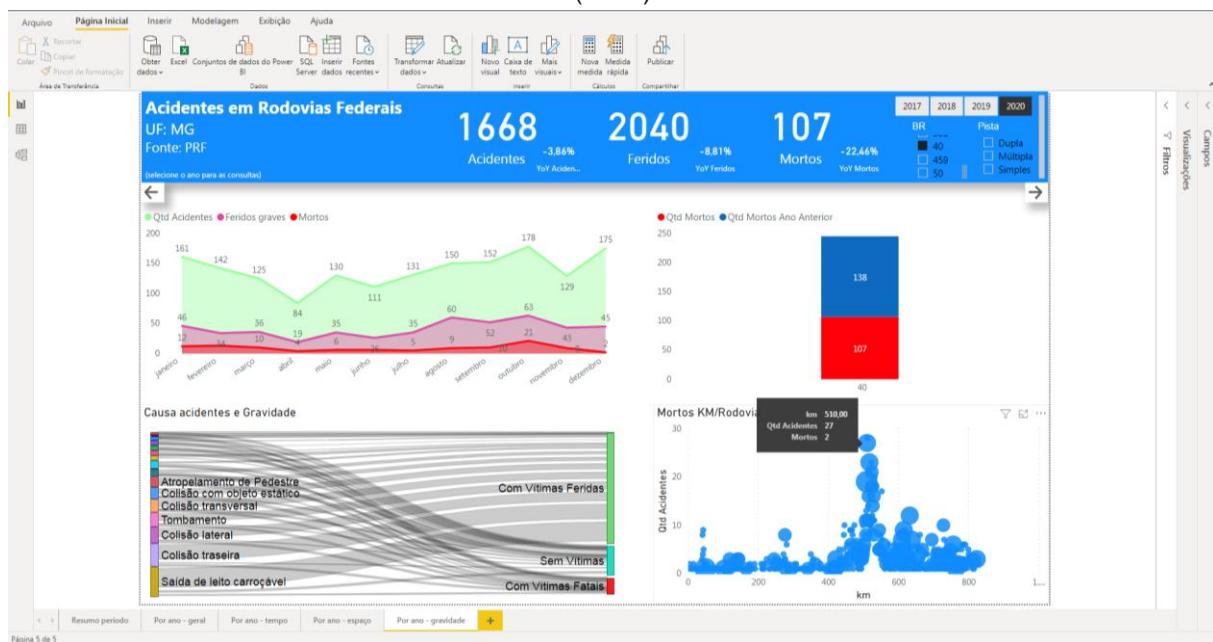
- No dashboard “Por ano - gravidade” (Figura 65), foram desenvolvidos objetos para apresentar dados que possam estar relacionados à gravidade dos acidentes. É apresentado um objeto com total de acidentes, feridos graves e mortos; um objeto que relaciona causa e acidentes sem e com vítimas (feridos e fatais), outro objeto com comparativo de mortos por ano e por quilômetro.

Figura 65: Dashboard “Por ano - gravidade” do MS-PowerBI (autor)



- Com o dashboard “Por ano - gravidade” (Figura 66), selecionando o filtro de ano de 2020 e a BR-040, verifica-se o comportamento no ano entre acidentes, feridos graves e mortos, de que foram 107 mortos em 2020 (quantidade menor do que em 2019), as principais causas presumíveis relacionadas à classificação dos acidentes (sem vítimas e com vítimas – feridas e fatais), além de identificar a distribuição de acidentes por quilômetro com a quantidade de mortos.

Figura 66: Dashboard “Por ano - gravidade” do MS-PowerBI com seleção de ano e BR (autor)



7. Criação do modelo de Machine Learning

Após a montagem da base de dados, do processamento dos dados e de uma análise e exploração dos dados, a partir deste ponto, a proposta do trabalho é efetuar a aplicação de modelos de *machine learning* para a classificação do estado físico dos envolvidos.

O atributo ‘estado_físico’, presente nos registros de ocorrência agrupados por pessoas, trata-se da condição do envolvido conforme a gravidade das lesões e apresenta os valores de “Ileso”, “Lesões Leves”, “Lesões Graves”, “Não Informado” e

“Óbito”. No caso, quando, no momento do registro da ocorrência, não se tem informações sobre o estado físico de uma (ou mais) pessoas envolvidas no acidente, são informados os valores “Não Informado” e “Ignorado”. Na amostra referente às ocorrências de acidentes na UF de MG para o período 2017 a 2020, após o tratamento dos dados, restou apenas o valor “Não Informado” (ver Figura 67).

Figura 67: Atributo ‘estado_fisico’ de “acidentes_mg3”
(autor)

```
1 acidentes_mg['estado_fisico'].value_counts()
Ileso        37895
Lesões Leves 36078
Lesões Graves 10022
Não Informado 4837
Óbito          2957
Name: estado_fisico, dtype: int64
```

O valor de “Não Informado” (e ou “Ignorado”) pode vir a distorcer as análises da amostra, até porque apresenta quantidade superior ao de óbitos. Serão aplicados modelos de *machine learning* para a classificação do estado físico dos envolvidos e recompor a amostra de dados.

Neste percurso, serão testados alguns algoritmos e recomposta a amostra de dados com a aplicação do algoritmo que apresentar melhores resultados. A intenção é prever valores de “Ileso”, “Lesões Leves”, “Lesões Graves” e “Óbito”, de maneira que o valor de “Não Informado” não pode fazer parte dos valores que o modelo irá aprender, devendo o mesmo ser removido da amostra em que serão aplicados os algoritmos de *machine learning*.

Foi criado um *notebook Jupiter* com os códigos em Python para a criação do modelo de *machine learning*. Neste *notebook*, foi efetuado carregamento de bibliotecas gerais e configurações, bem como carregado o arquivo “acidentes_mg3”, onde as ocorrências estão agrupadas por pessoas e existem atributos com a qualificação do estado físico dos envolvidos. Ver Figura 68.

Figura 68: Notebook Python para a criação dos modelos de Machine Learning
(autor)

PUC Minas - Pontifícia Universidade Católica de Minas Gerais

Pós-Graduação em Ciência de Dados e Big Data

TRABALHO DE CONCLUSÃO DE CURSO

Aluna: Lilian Campos Soares

Matrícula: 1092883

Este notebook é referente aos códigos elaborados em Python e utilizados no Trabalho de Conclusão de Curso em Ciência de Dados e Big Data da Pós-Graduação da PUC Minas, especificamente para a criação dos modelos de Machine Learning.

```

1 # Carregamento de bibliotecas
2 import pandas as pd
3 import numpy as np
4 import seaborn as sns
5 import matplotlib.pyplot as plt

1 #na EPL
2 #import io
3 #import os
4 #os.chdir("C:/TCC PUC/csv")

1 # Definição de configurações de dataframes
2 pd.set_option('display.max_rows', 1000)
3 pd.set_option('display.max_columns', 500)
4 pd.set_option('display.width', 1000)
5 pd.set_option('display.precision', 4)
6 pd.set_option('display.expand_frame_repr', False)

1 #Configurando os plots para serem exibidos diretamente no notebook
2 %matplotlib inline

1 #Importação dos dados para início da preparação do modelo de ML - arquivos datatran e acidentes.CSV
2 acidentes_mg = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\acidentes_mg3.csv", sep=';', decimal=',', encoding = 'cp12'

```

Após carga dos dados, efetuou-se verificação na amostra de “acidentes_mg3”, sendo 91.789 ocorrências por pessoa envolvida, como já identificado anteriormente. Também, confirmou-se que não existem registros nulos e foram computados os valores do atributo ‘estado_fisico’. Ver Figura 69.

Figura 69: Nova verificação da amostra de “acidentes_mg3”
(autor)

```

1 acidentes_mg.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 91789 entries, 0 to 91788
Data columns (total 26 columns):
 #   Column           Non-Null Count  Dtype  
--- 
0   id               91789 non-null    int64  
1   pesid             91789 non-null    int64  
2   data_inversa      91789 non-null    datetime64[ns]
3   dia_semana        91789 non-null    object  
4   horario            91789 non-null    object  
5   uf                91789 non-null    object  
6   br                91789 non-null    int64  
7   km                91789 non-null    float64 
8   municipio          91789 non-null    object  
9   causa_acidente    91789 non-null    object  
10  tipo_acidente     91789 non-null    object  
11  classificacao_acidente 91789 non-null    object  
12  fase_dia           91789 non-null    object  
13  condicao_metereologica 91789 non-null    object  
14  tipo_pista          91789 non-null    object  
15  tracado_via         91789 non-null    object  
16  uso_solo            91789 non-null    object  
17  tipo_veiculo        91789 non-null    object  
18  tipo_envolvido      91789 non-null    object  
19  estado_fisico        91789 non-null    object  
20  sexo               91789 non-null    object  
21  ilesos              91789 non-null    int64  
22  feridos_leves       91789 non-null    int64  
23  feridos_graves      91789 non-null    int64  
24  mortos              91789 non-null    int64  
25  elemento_transito    91789 non-null    object  
dtypes: datetime64[ns](1), float64(1), int64(7), object(17)
memory usage: 18.2+ MB

1 #Verificação de valores nulos / NaN nas colunas
2 acidentes_mg.isnull().sum()
3 #sem valores nulos / NaN

```

id 0

```

pesid          0
data_inversa   0
dia_semana    0
horario        0
uf             0
br             0
km             0
municipio      0
causa_acidente 0
tipo_acidente   0
classificacao_acidente 0
fase_dia       0
condicao_metereologica 0
tipo_pista     0
tracado_via    0
uso_solo       0
tipo_veiculo   0
tipo_envolvido 0
estado_fisico  0
sexo            0
ilesos          0
feridos_leves   0
feridos_graves  0
mortos          0
elemento_transito 0
dtype: int64

```

```

1 accidentes_mg['estado_fisico'].value_counts()

Ileso          37895
Lesões Leves   36078
Lesões Graves  10022
Não Informado  4837
Óbito           2957
Name: estado_fisico, dtype: int64

```

Foi realizada a conversão do tipo de atributo de ‘km’ para *float* e criado um novo *dataset* chamado “*fset*” com os seguintes atributos: ‘dia_semana’, ‘br’, ‘km’, ‘causa_acidente’, ‘tipo_acidente’, ‘classificacao_acidente’, ‘fase_dia’, ‘condicao_metereologica’, ‘tipo_pista’, ‘tracado_via’, ‘uso_solo’, ‘tipo_veiculo’, ‘tipo_envolvido’, ‘estado_fisico’, ‘sexo’ e ‘elemento_transito’. Ver Figura 70.

Figura 70: Novo tratamento da amostra de “accidentes_mg3” e criação do “fset” (autor)

```

#Escolha dos atributos para um novo dataset chamado 'fset'
fset = accidentes_mg[['dia_semana','br','km','causa_acidente','tipo_acidente','classificacao_acidente','fase_dia', 'condicao_metereologica','tipo_pista','tracado_via','uso_solo','tipo_v']]
fset.head()

```

	dia_semana	br	km	causa_acidente	tipo_acidente	classificacao_acidente	fase_dia	condicao_metereologica	tipo_pista	tracado_via	uso_solo	tipo_v
0	domingo	381	605.0	Condutor Dormindo	Saída de leito carroçável	Com Vítimas Feridas	Amanhecer		Céu Claro	Dupla	Reta	Rural
1	domingo	381	605.0	Condutor Dormindo	Saída de leito carroçável	Com Vítimas Feridas	Amanhecer		Céu Claro	Dupla	Reta	Rural
2	domingo	262	368.0	Velocidade Incompatível	Colisão com objeto estático	Com Vítimas Feridas	Plena Noite		Céu Claro	Simples	Viaduto	Urbano
3	domingo	262	368.0	Velocidade Incompatível	Colisão com objeto estático	Com Vítimas Feridas	Plena Noite		Céu Claro	Simples	Viaduto	Urbano
4	domingo	262	368.0	Velocidade Incompatível	Colisão com objeto estático	Com Vítimas Feridas	Plena Noite		Céu Claro	Simples	Viaduto	Urbano

Foram então retirados os registros em que o ‘estado_fisico = ‘Não Informado’ do *dataset* “*fset*” (com essa retirada, *fset* ficou com 86.952 registros), bem como alocadas as variáveis preditoras (X) e alvo ou *target* (Y). Ver Figura 71.

Figura 71: Tratamento em “fset” e alocação de variáveis preditoras e alvo (autor)

```

1 #Tratamento dos registros que o estado_fisico está como Não Informado
2 f = fset.loc[fset['estado_fisico'] == 'Não Informado'].index
3 fset.drop(f, axis= 0, inplace = True)

1 #Alocação de variáveis preditoras
2 X = fset[['dia_semana','br','km','causa_acidente','tipo_acidente','classificacao_acidente','fase_dia','condicao_metereologica','tipo_pista','tracado_via','uso_solo','tipo_veiculo','tipo_envolvido','sexo','elemento_transito']]
3
4 #Alocação da variável target
5 Y = fset['estado_fisico'].values

```

As variáveis preditoras armazenadas em X são: 'dia_semana', 'br', 'km', 'causa_acidente', 'tipo_acidente', 'classificacao_acidente', 'fase_dia', 'condicao_metereologica', 'tipo_pista', 'tracado_via', 'uso_solo', 'tipo_veiculo', 'tipo_envolvido', 'sexo', 'elemento_transito'. E a variável alvo armazenada em Y é: 'estado_fisico'.

Na sequência, deu-se a transformação de dados não numéricos em dados numéricos. Neste passo, foi utilizada a função *LabelEncoder* do módulo *preprocessing* da biblioteca Sklearn ou *scikit-learn*⁴. A Sklearn fornece uma ferramenta eficiente para codificar níveis de recursos categóricos em valores numéricos. A LabelEncoder codifica rótulos com um valor entre 0 e n_classes-1, onde n é o número de rótulos distintos. Se um rótulo se repetir, ele atribuirá o mesmo valor atribuído anteriormente.

Na transformação, a nomenclatura das variáveis seguiu um padrão com o prefixo 'LE_' de Label Encoder. E, o método '.fit()' recebe uma lista com os valores presentes na variável X. Foi utilizado a '.unique()' para trazer todos os valores únicos das colunas do dataset "fset".

Figura 72: Transformação de dados não numéricos de X (autor)

```

1 #Importação do módulo preprocessing
2 from sklearn import preprocessing

1 #Transformação de dados não numéricos em dados numéricos - utilização da função LabelEncoder do módulo preprocessing da biblioteca Sklearn ou scikit-learn
2 LE_Dia_Semana = preprocessing.LabelEncoder()
3 LE_Dia_Semana.fit(fset.dia_semana.unique())
4 X[:,0] = LE_Dia_Semana.transform(X[:,0])

1 LE_Causa = preprocessing.LabelEncoder()
2 LE_Causa.fit(fset.causa_acidente.unique())
3 X[:,3] = LE_Causa.transform(X[:,3])

1 LE_Tipo_Accidente = preprocessing.LabelEncoder()
2 LE_Tipo_Accidente.fit(fset.tipo_acidente.unique())

```

⁴<http://scikit-learn.org>: `sklearn.preprocessing.LabelEncoder`

```

3 X[:,3] = LE_Causa.transform(X[:,3])

1 LE_Tipo_Accidente = preprocessing.LabelEncoder()
2 LE_Tipo_Accidente.fit(fset.tipo_acidente.unique())
3 X[:,4] = LE_Tipo_Accidente.transform(X[:,4])

1 LE_Class_acidente = preprocessing.LabelEncoder()
2 LE_Class_acidente.fit(fset.classificacao_acidente.unique())
3 X[:,5] = LE_Class_acidente.transform(X[:,5])

1 LE_Fase_Dia = preprocessing.LabelEncoder()
2 LE_Fase_Dia.fit(fset.fase_dia.unique())
3 X[:,6] = LE_Fase_Dia.transform(X[:,6])

1 LE_Condicao_Met = preprocessing.LabelEncoder()
2 LE_Condicao_Met.fit(fset.condicao_metereologica.unique())
3 X[:,7] = LE_Condicao_Met.transform(X[:,7])

1 LE_Tipo_Pista = preprocessing.LabelEncoder()
2 LE_Tipo_Pista.fit(fset.tipo_pista.unique())
3 X[:,8] = LE_Tipo_Pista.transform(X[:,8])

1 LE_Tracado_Via = preprocessing.LabelEncoder()
2 LE_Tracado_Via.fit(fset.tracado_via.unique())
3 X[:,9] = LE_Tracado_Via.transform(X[:,9])

1 LE_Uso_Solo = preprocessing.LabelEncoder()
2 LE_Uso_Solo.fit(fset.uso_solo.unique())
3 X[:,10] = LE_Uso_Solo.transform(X[:,10])

1 LE_Tipo_Veiculo = preprocessing.LabelEncoder()
2 LE_Tipo_Veiculo.fit(fset.tipo_veiculo.unique())
3 X[:,11] = LE_Tipo_Veiculo.transform(X[:,11])

1 LE_Tipo_Envolvido = preprocessing.LabelEncoder()
2 LE_Tipo_Envolvido.fit(fset.tipo_envolvido.unique())
3 X[:,12] = LE_Tipo_Envolvido.transform(X[:,12])

1 LE_Sexo = preprocessing.LabelEncoder()
2 LE_Sexo.fit(fset.sexo.unique())
3 X[:,13] = LE_Sexo.transform(X[:,13])

1 LE_Elemento_Transito = preprocessing.LabelEncoder()
2 LE_Elemento_Transito.fit(fset.elemento_transito.unique())
3 X[:,14] = LE_Elemento_Transito.transform(X[:,14])

```

Concluída a transformação de dados não numéricos em dados numéricos, foram efetuadas algumas verificações nas variáveis X e Y. No caso, para conhecer o conteúdo dos arrays das variáveis que foram montados e confirmar que se obteve sucesso na transformação de dados de X. Foi criado um *dataframe* auxiliar com o intuito de verificar as variáveis preditoras X e a distribuição de dados.

Figura 73: Verificação de X e Y
(autor)

```

1 print(Y)
['Lesões Leves' 'Lesões Leves' 'Lesões Leves' ... 'Lesões Graves' 'Ileso'
 'Ileso']

1 print(X)
[[0 381 605.0 ... 1 2 0]
 [0 381 605.0 ... 2 2 0]
 [0 262 368.0 ... 2 2 0]
 ...
 [4 116 648.0 ... 1 0 0]
 [4 116 648.0 ... 1 2 0]
 [5 153 7.0 ... 1 2 2]]

```

```

1 dfx = pd.DataFrame(X)

```

```
1 dfx.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 86952 entries, 0 to 86951
Data columns (total 15 columns):
 #   Column  Non-Null Count  Dtype  
---  -- 
 0   0       86952 non-null   object 
 1   1       86952 non-null   object 
 2   2       86952 non-null   object 
 3   3       86952 non-null   object 
 4   4       86952 non-null   object 
 5   5       86952 non-null   object 
 6   6       86952 non-null   object 
 7   7       86952 non-null   object 
 8   8       86952 non-null   object 
 9   9       86952 non-null   object 
 10  10      86952 non-null   object 
 11  11      86952 non-null   object 
 12  12      86952 non-null   object 
 13  13      86952 non-null   object 
 14  14      86952 non-null   object 
dtypes: object(15)
memory usage: 10.0+ MB
```

```
1 dfx.head()
   0   1   2   3   4   5   6   7   8   9   10  11  12  13  14
0  0  381  605  7  15  1  0  1  0  5  0  0  1  2  0
1  0  381  605  7  15  1  0  1  0  5  0  0  2  2  0
2  0  262  368  32  5  1  2  1  2  9  1  0  2  2  0
3  0  262  368  32  5  1  2  1  2  9  1  0  2  0  0
4  0  262  368  32  5  1  2  1  2  9  1  0  1  2  0
```

```
1 dfx.count()
0    86952
1    86952
2    86952
3    86952
4    86952
5    86952
6    86952
7    86952
8    86952
9    86952
10   86952
11   86952
12   86952
13   86952
14   86952
dtype: int64
```

```
1 dfx.nunique()
0      7
1     21
2    950
3     33
4     17
5      3
6      4
7     10
8      3
9     10
10    2
11    21
12      5
13      4
14      3
dtype: int64
```

```
1 dfx.isnull().sum()
0    0
1    0
2    0
3    0
4    0
5    0
6    0
7    0
8    0
9    0
10   0
11   0
```

```
12    0
13    0
14    0
dtype: int64
```

```
1 print(dfx.describe())
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
count	86952	86952	86952.0	86952	86952	86952	86952	86952	86952	86952	86952	86952	86952	86952	86952
unique	7	21	950.0	33	17	3	4	10	3	10	2	21	5	4	3
top	5	381	510.0	16	9	1	3	1	2	5	0	0	1	2	0
freq	15313	27617	491.0	28510	17467	65797	49861	46412	46598	43599	61919	40017	56996	65633	72826

```
1 dfx.to_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\dfx.csv", index = False, header = True, sep=';', decimal='.', encoding = 'cp1252')
```

```
1 dfx = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\dfx2.csv", sep=';', decimal='.', encoding = 'cp1252')
```

```
1 dfx.head()
```

	col0	col1	col2	col3	col4	col5	col6	col7	col8	col9	col10	col11	col12	col13	col14
0	0	381	605.0	7	15	1	0	1	0	5	0	0	1	2	0
1	0	381	605.0	7	15	1	0	1	0	5	0	0	2	2	0
2	0	262	368.0	32	5	1	2	1	2	9	1	0	2	2	0
3	0	262	368.0	32	5	1	2	1	2	9	1	0	2	0	0
4	0	262	368.0	32	5	1	2	1	2	9	1	0	1	2	0

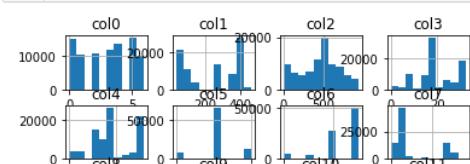
```
1 dfx.info()
```

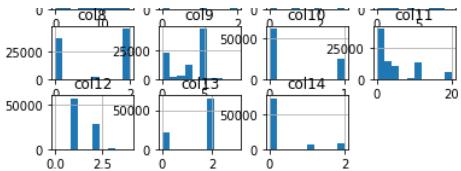
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 86952 entries, 0 to 86951
Data columns (total 15 columns):
 #   Column  Non-Null Count  Dtype  
 --- 
 0   col0    86952 non-null  int64  
 1   col1    86952 non-null  int64  
 2   col2    86952 non-null  float64 
 3   col3    86952 non-null  int64  
 4   col4    86952 non-null  int64  
 5   col5    86952 non-null  int64  
 6   col6    86952 non-null  int64  
 7   col7    86952 non-null  int64  
 8   col8    86952 non-null  int64  
 9   col9    86952 non-null  int64  
 10  col10   86952 non-null  int64  
 11  col11   86952 non-null  int64  
 12  col12   86952 non-null  int64  
 13  col13   86952 non-null  int64  
 14  col14   86952 non-null  int64  
dtypes: float64(1), int64(14)
memory usage: 10.0 MB
```

```
1 print(dfx.describe())
```

	col0	col1	col2	col3	col4	col5	col6	col7	col8	col9	col10	col11	col12	col13	col14	
count	86952.0000	86952.0000	86952.0000	86952.0000	86952.0000	86952.0000	86952.0000	86952.0000	86952.0000	86952.0000	86952.0000	86952.0000	86952.0000	86952.0000	86952.0000	
mean	2.9618	234.3929	458.9134	19.0324	9.1285	1.0576	2.4139	2.4975	1.0995	3.2484	0.2879	3.9606	1.3626	1.5190	0.2558	
std	2.0172	142.9484	239.0857	8.5091	4.3668	0.4899	0.8180	2.8601	0.9810	2.2158	0.4528	5.5344	0.5187	0.8551	0.6139	
min	0.0000	40.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	
25%	1.0000	116.0000	289.0000	15.0000	6.0000	1.0000	2.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	
50%	3.0000	262.0000	489.0000	16.0000	9.0000	1.0000	3.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	
75%	5.0000	381.0000	621.0000	26.0000	14.0000	1.0000	3.0000	6.0000	0.0000	0.0000	1.0000	5.0000	2.0000	0.0000	0.0000	
max	6.0000	459.0000	949.0000	32.0000	16.0000	2.0000	3.0000	9.0000	0.0000	0.0000	1.0000	20.0000	4.0000	3.0000	2.0000	9.0000

```
1 dfx.hist()
2 plt.show()
```

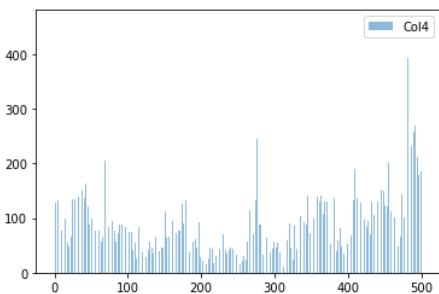




```

1 plt.hist(dfx['col2'], bins, alpha=0.5, label='Col4')
2 plt.legend(loc='upper right')
3 plt.show()

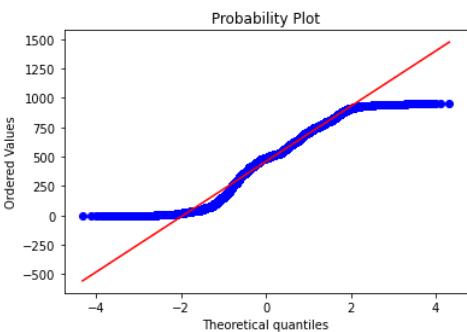
```



```

1 import scipy.stats as stats
2 stats.probplot(dfx['col2'], dist="norm", plot=pylab)
3 pylab.show()

```



O próximo passo foi a normalização dos valores visando um melhor desempenho dos algoritmos de classificação que serão usados (Figura 74). O objetivo da normalização é alterar os valores das colunas numéricas no conjunto de dados para uma escala comum, sem distorcer as diferenças nos intervalos de valores.

Para *machine learning*, nem todos os conjuntos de dados requerem normalização, devendo o método ser necessário apenas quando os parâmetros tiverem intervalos muito diferentes⁵. Optou-se por efetuar a normalização, considerando as diferenças percebidas na distribuição dos dados de X (vide Figura 73). Foram efetuados testes com *StandardScaler*, *MinMaxScaler* e *RobustScaler*, mas definiu-se pelo *StandardScaler* pelos resultados de média e desvio padrão apresentados, conforme Figura 74.

⁵ <https://medium.com/tentando-ser-um-unic%C3%B3rnio/porqu%C3%A9-e-quando-%C3%A9-necess%C3%A1rio-normalizar-os-dados-92e5cce445aa>

Figura 74: Normalização dos dados de X
(autor)

```

1 #Normalização dos valores para uma melhor performance do algoritmo de classificação
2 #Foram efetuados testes com StandardScaler, MinMaxScaler e RobustScaler, mas optou-se pelo StandardScaler

1 #Normalização dos valores com StandardScaler
2 std = preprocessing.StandardScaler()
3 std.fit(X)
4 Xstd = std.transform(X)
5 print ('Média antes da normalização: {:.2f}'.format(X.mean()),'\nDesvio padrão antes da normalização: {:.2f}'.format(X.std())
6 print('')
7 print ('Média depois da normalização: {:.2f}'.format(Xstd.mean()),'\nDesvio padrão depois da normalização: {:.2f}'.format(Xs
8
9
10 Média antes da normalização: 49.48
11 Desvio padrão antes da normalização: 143.08
12
13 Média depois da normalização: -0.00
14 Desvio padrão depois da normalização: 1.00

```

Uma vez normalizado, o array “Xstd”, que contém os valores para predição, foi dividido em 04 (quatro) partes com o objetivo de constituir conjuntos para treinamento e teste do modelo:

- X_trainset: uma de treinamento do modelo, onde o algoritmo irá 'apreender' a relação entre as variáveis preditoras;
- Y_trainset: uma parte também de treinamento do modelo, mas somente com as variáveis alvo, ou os valores que queremos prever;
- X_testset: uma parte para teste, onde o modelo irá exercer o aprendizado obtido com os arrays X_trainset e Y_trainset, ou irá testar o que foi observado com os arrays de treinamento; e
- Y_testset: uma parte para avaliar o desempenho do modelo, em que Y_testset contém as saídas corretas que se deseja prever e é usado para comparar com as previsões feitas pelo modelo.

Figura 75: Divisão do conjunto para treinamento e teste
(autor)

```

1 #Depois de normalizado, o array Xstd, que contém os valores para a predição, foi dividido em 4 partes:
2
3 #X_trainset: Uma de treino do modelo, onde o algoritmo irá 'apreender' a relação entre as variáveis preditoras
4 #Y_trainset: Uma parte também de treino do modelo, mas somente com as variáveis target, ou os valores que queremos prever
5 #X_testset: Uma parte para teste, onde o modelo irá exercer o aprendizado obtido com os arrays X_trainset e Y_trainset, ou i
6 #Y_testset: Uma parte para avaliar a performance do modelo, Y_testset contém as saídas corretas que desejamos prever,e é usa
7
8
9 #Importação do modulo train_test_split
10 from sklearn.model_selection import train_test_split
11
12 #Divisão do dataset seguindo a proporção 70/30 (70% para treino e 30% para teste)
13 X_trainset, X_testset, Y_trainset, Y_testset = train_test_split(Xstd, Y, test_size=0.3, random_state=3)

```

A exemplo do trabalho de outros pesquisadores com ocorrências de acidentes de trânsito em rodovias federais⁶, o primeiro algoritmo que se decidiu testar foi o SGDClassifier (Stochastic Gradient Descent Classifier)⁷. SGDClassifier é um classificador linear (como o Linear SVM – Support Vector Machine e o Logistic Regression), contudo é otimizado pelo SGD. SGD é um método de otimização que é aplicado no algoritmo. O modelo de *machine learning* define uma função de perda (loss) e o método de otimização a minimiza ou maximiza (Fuchs, 2019).

O *Gradient Descent* é usado para minimizar uma função de custo, sendo um dos algoritmos mais populares para realizar a otimização, ao calcular o gradiente usando uma única amostra; inclusive é o método mais comum para otimizar redes neurais. Pode-se, então, utilizar este algoritmo para otimizar um classificador linear, como *Logistic Regression* e *LinearSVM*, por meio do SGDClassifier com parâmetro `loss='log'` e SGDClassifier com parâmetro `loss='hinge'`, respectivamente (Fuchs, 2019).

Embora a amostra do estudo tenha ficado com menos de 100 mil registros (SKLEARN, 2020), o SGDClassifier é mais eficiente do que o Linear SVM sem SGD (SVC com `kernel='linear'` ou Linear SVC) para *labeled data*. Foi efetuado teste comparativo e identificado que o SGDClassifier executa bem mais rápido (3.32s) do que o Linear SVC (953.10s) nesta amostra de dados do estudo (Figura 76).

Figura 76: Teste comparativo de desempenho com o SGDClassifier (autor)

```
click to scroll output; double click to hide [sem SGD com o SGDClassifier]
3 import time
4 start = time.time()
5 clf = SGDClassifier(loss="hinge", penalty="l2")
6 clf.fit(X_trainset,Y_trainset)
7 stop = time.time()
8 print(f"Tempo de treinamento para linear SVM com SGD: {stop - start}s")
9
10 start = time.time()
11 clf = SVC(kernel='linear')
12 clf.fit(X_trainset,Y_trainset)
13 stop = time.time()
14 print(f"Tempo de treinamento para linear SVM sem SGD: {stop - start}s")
15 #SGD classifier executa bem mais rápido do que o Linear SVM
```

Tempo de treinamento para linear SVM com SGD: 3.3240966796875s
 Tempo de treinamento para linear SVM sem SGD: 953.1051347255707s

⁶ <https://www.linkedin.com/pulse/tr%C3%A2nsito-brasil-o-que-mudou-em-10-anos-parte-4-mais-ramon-nogueira/> e <https://www.linkedin.com/pulse/an%C3%A1lide-das-v%C3%ADtimas-de-acidentes-tr%C3%A2nsito-nas-rodovias-gabriel-gomes/?published=1>

⁷ https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Como não se conhecia o funcionamento do algoritmo e da configuração de seus parâmetros, foram realizados alguns testes a até se chegar em uma definição adequada dos parâmetros do algoritmo. Iniciou-se com a regressão logística, obtendo uma acurácia de 0.58, e com a verificação da influência no número de iterações sob o `score`. Por padrão, o número máximo de passes ('`max_iter`') sob os dados de treinamento (`epoch`s) é 1000. Percebeu-se que o `score` é maior com um menor número de iterações.

Figura 77: Teste do algoritmo Regressão Logística com treinamento SGD do modelo (autor)

```

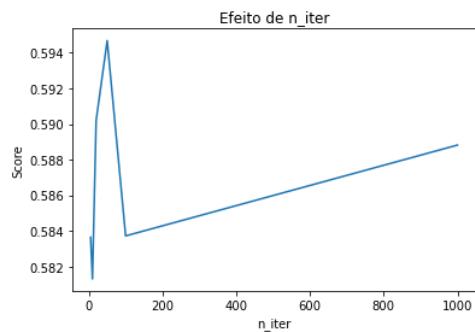
1 #Regressão Logística (loss='log') com treinamento SGD do modelo
2 clf = SGDClassifier(loss="log", penalty="l2")
3 clf.fit(X_trainset, Y_trainset)

SGDClassifier(loss='log')

1 Y_Pred = clf.predict(X_testset)
2 print('Acurácia: {:.2f}'.format(accuracy_score(Y_testset, Y_Pred)))
Acurácia: 0.58

1 #Verificando a influência sob o número de iterações
2 n_iters = [5, 10, 20, 50, 100, 1000]
3 scores = []
4 for n_iter in n_iters:
5     clf = SGDClassifier(loss="log", penalty="l2", max_iter=n_iter)
6     clf.fit(X_trainset, Y_trainset)
7     scores.append(clf.score(X_testset, Y_testset))

1 plt.title("Efeito de n_iter")
2 plt.xlabel("n_iter")
3 plt.ylabel("Score")
4 plt.plot(n_iters, scores)
[<matplotlib.lines.Line2D at 0x1b54aec4ee0>]
```



Na sequência, efetuou-se o mesmo cálculo para o modelo linear do SVM, com treinamento SGD, e também se verificou a influência no número de iterações sob o `score` (Figura 78). Obteve-se uma acurácia um pouco maior (0.59) com o número padrão de '`max_iter`' (=1000); contudo, percebeu-se um comportamento distinto do `score` sob a influência de outros números de iterações – inclusive, comparando os gráficos das Figuras 77 e 78, a Regressão Logística com treinamento SGD aparentemente tem desempenho melhor.

Figura 78: Teste do algoritmo Linear SVM com treinamento SGD do modelo (autor)

```

1 #Linear SVM (Loss='hinge') com treinamento SGD do modelo
2 clf = SGDClassifier(loss="hinge", penalty="l2")
3 clf.fit(X_trainset,Y_trainset)

SGDClassifier()

1 Y_Pred = clf.predict(X_testset)
2 print('Acurácia: {:.2f}'.format(accuracy_score(Y_testset, Y_Pred)))

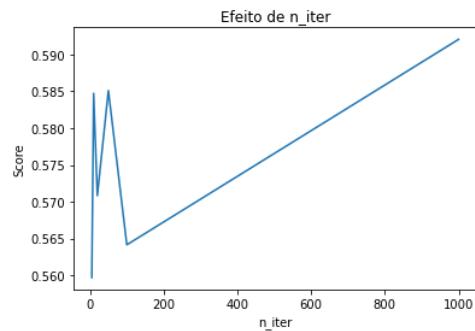
Acurácia: 0.59

1 #Verificando a influência sob o número de iterações
2 n_iters = [5, 10, 20, 50, 100, 1000]
3 scores = []
4 for n_iter in n_iters:
5     clf = SGDClassifier(loss="hinge", penalty="l2", max_iter=n_iter)
6     clf.fit(X_trainset,Y_trainset)
7     scores.append(clf.score(X_testset, Y_testset))

1 plt.title("Efeito de n_iter")
2 plt.xlabel("n_iter")
3 plt.ylabel("Score")
4 plt.plot(n_iters, scores)

[<matplotlib.lines.Line2D at 0x1b54ab2b700>]

```



Explorando os diferentes modelos lineares do SGDClassifier, verificou-se o desempenho dos classificadores a partir do parâmetro ‘loss’ entre *hinge*, *log*, *modified_hubber*, *perceptron* e *squared_hinge*. Tornou-se claro que o modelo linear do SVM (‘loss=hinge’) entrega o melhor score e a aplicação do *perceptron* entrega o pior score. Figura 79.

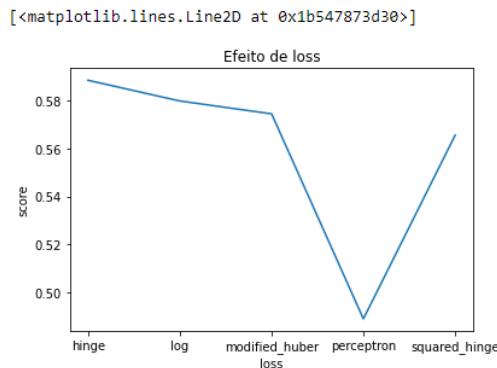
Figura 79: Comparação de desempenho entre diferentes modelos lineares com SGD (autor)

```

1 #Comparação de desempenho "Performance" entre diferentes modelos lineares
2 losses = ["hinge", "log", "modified_hubber", "perceptron", "squared_hinge"]
3 scores = []
4 for loss in losses:
5     clf = SGDClassifier(loss=loss, penalty="l2", max_iter=1000)
6     clf.fit(X_trainset,Y_trainset)
7     scores.append(clf.score(X_testset, Y_testset))

1 plt.title("Efeito de loss")
2 plt.xlabel("loss")
3 plt.ylabel("score")
4 x = np.arange(len(losses))
5 plt.xticks(x, losses)
6 plt.plot(x, scores)

```



Mesmo, já tendo algumas indicações do que seria a melhor configuração de parâmetros para o SGDClassifier, empregou-se o método *GridSearch* para encontrar os mais adequados hiper parâmetros para uso do algoritmo. Embora a acurácia não conseguisse ser muito aumentada, obteve-se um ganho de 0.01, encontrando uma acuraria de 0.60 com os parâmetros ‘alpha’: 0.01, ‘loss’: ‘log’ e ‘penalty’:‘l1’. Concluiu-se que a Regressão Logística com treinamento SGD do modelo é o algoritmo mais adequado para a amostra deste trabalho.

Figura 80: Parâmetros mais adequados para o SGDClassifier
(autor)

```

1 #Encontrando os hyper parâmetros mais adequados por meio do método GridSearch
2 from sklearn.model_selection import GridSearchCV
3 params = {
4     "loss" : ["hinge", "log", "squared_hinge", "modified_huber", "perceptron"],
5     "alpha" : [0.0001, 0.001, 0.01, 0.1],
6     "penalty" : ["l2", "l1", "elasticnet", "none"],
7 }
8
9 clf = SGDClassifier(max_iter=50)
10 grid = GridSearchCV(clf, param_grid=params, cv=10)

1 grid.fit(X_trainset,Y_trainset)

1 print(grid.best_params_)

{'alpha': 0.01, 'loss': 'log', 'penalty': 'l1'}

1 grid_predictions = grid.predict(X_testset)
2
3 print('Acurácia: {:.2f}'.format(accuracy_score(Y_testset, grid_predictions)))

```

Acurácia: 0.60

Apenas para se certificar e deixar a configuração adequada do algoritmo, executou-se novamente com os mais adequados hiper parâmetros para uso do algoritmo, conforme pode ser verificado na Figura 81. Embora 0.60 não seja uma acurácia ideal, prosseguiu-se com o armazenamento das previsões com dados de treino e de teste (Figura 82), bem como com a aplicação do *jaccard_similarity_score* sobre estes dados (Figura 83) para teste e a verificação dos erros de classificação nas amostras de treino e de teste (Figura 84).

Figura 81: Regressão Logística com treinamento SGD do modelo e os hiper parâmetros adequados
 (autor)

```

1 #Teste com o algoritmo SGDClassifier e os hyper parâmetros mais adequados
2 clf = SGDClassifier(loss="log", penalty="l1", alpha=0.01, max_iter=50)
3 clf.fit(X_trainset,Y_trainset)

SGDClassifier(alpha=0.01, loss='log', max_iter=50, penalty='l1')

1 Y_Pred = clf.predict(X_testset)
2 print('Acurácia: {:.2f}'.format(accuracy_score(Y_testset, Y_Pred)))

Acurácia: 0.60

```

Figura 82: Armazenamento das predições do SGDClassifier
 (autor)

```

1 #Armazenando as predições com dados de treino e dados de teste
2 pTrain = clf.predict(X_trainset)
3 pTest = clf.predict(X_testset)

```

Figura 83: Índice Jaccard da Regressão Logística com treinamento SGD do modelo
 (autor)

```

1 #Obtendo o Índice Jaccard dos dados de treino e de teste
2 print('Índice Jaccard p/ dados de Treino loss = log: {:.3f}'.format(m.jaccard_score(Y_trainset, pTrain, average='micro')))
3 print('Índice Jaccard p/ dados de Teste loss = log: {:.3f}'.format(m.jaccard_score(Y_testset, pTest, average='micro')))

Índice Jaccard p/ dados de Treino loss = log: 0.427%
Índice Jaccard p/ dados de Teste loss = log: 0.429%

```

Figura 84: Erros de classificação da Regressão Logística com treinamento SGD do modelo
 (autor)

```

1 #Verificando os erros de classificação nas amostras de treino e de teste
2 print('Erros de classificação das amostras Treino: {}'.format((Y_trainset != pTrain).sum()),'\nAmostras de treino "Y": {}'.format((Y_trainset == 1).sum()))
3 print('Erros de classificação das amostras Teste: {}'.format((Y_testset != pTest).sum()),'\nAmostras de teste "Y": {}'.format((Y_testset == 1).sum()))

Erros de classificação das amostras Treino: 24427
Amostras de treino "Y": 60866
Erros de classificação das amostras Teste: 10436
Amostras de teste "Y": 26086

```

Se o conjunto inteiro de *labels* previstas para uma amostra (*y_pred*) combina exatamente com o conjunto de *labels* verdadeiro (*y_true*), então a acurácia (precisão) é 1.0; caso contrário, a acurácia é 0.0. Em todas as verificações do score de acurácia do algoritmo SGDClassifier, havia sido utilizado o método *.accuracy_score* (que é a quantidade de amostras classificadas corretamente dividida pelo total).

De qualquer maneira, foi feito um teste com o método *.jaccard_score* (Figura 83) e que retornou valores mais baixos de acurácia. O coeficiente de similaridade Jaccard⁸ é definido como o tamanho da interseção dividido pelo tamanho da união de dois conjuntos de rótulos (no caso, *Y_trainset* e *pTrain*; ou *Y_testset* e *pTest*) e é usado para comparar o conjunto de rótulos previstos *y_pred* (*pTrain* e *pTest*) para

⁸ https://scikit-learn.org/stable/modules/generated/sklearn.metrics.jaccard_score.html

uma amostra com o conjunto correspondente de rótulos em y_{true} ($Y_{trainset}$ e $Y_{testset}$).

Passou-se a verificar o relatório de classificação e a matriz de confusão⁹ visando a avaliar o desempenho do modelo de *machine learning* com o algoritmo do SGDClassifier (Figura 85). A matriz de confusão contabiliza a quantidade de resultados positivos verdadeiros, falsos positivos, negativos verdadeiros e falsos negativos; refletindo o fato de que nos torna mais fácil ver que tipo de confusões ocorrem no algoritmo de classificação.

Figura 85: Matriz de Confusão¹⁰ da Regressão Logística com treinamento SGD do modelo (autor)

```

1 #Matriz de confusão
2 import itertools
3 def plot_confusion_matrix(cm, classes,
4                           normalize=False,
5                           title='Matriz de Confusão',
6                           cmap=plt.cm.Blues):
7     """
8         This function prints and plots the confusion matrix.
9         Normalization can be applied by setting `normalize=True`.
10    """
11    if normalize:
12        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
13        print("Matriz de confusão normalizada")
14    else:
15        print('Matriz de confusão sem normalização')
16
17    print(cm)
18
19    plt.imshow(cm, interpolation='nearest', cmap=cmap)
20    plt.title(title)
21    plt.colorbar()
22    tick_marks = np.arange(len(classes))
23    plt.xticks(tick_marks, classes, rotation=45)
24    plt.yticks(tick_marks, classes)
25
26    fmt = '.2f' if normalize else 'd'
27    thresh = cm.max() / 2.
28    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
29        plt.text(j, i, format(cm[i, j], fmt),
30                  horizontalalignment="center",
31                  color="white" if cm[i, j] > thresh else "black")
32
33    plt.tight_layout()
34    plt.ylabel('Label verdadeira')
35    plt.xlabel('Label predita')
36
37 #retirada de um dos notebooks do curso de Machine Learning da Cognitiveclass, assim como o bloco Logo em seguida, que faz a
4
1 #Cálculo da matriz de confusão
2 cnf_matrix = m.confusion_matrix(Y_testset, pTest, labels = fset.estado_fisico.unique())
3 np.set_printoptions(precision=2)
4
5
1 print (m.classification_report(Y_testset, pTest))
2
3 # Gráfico da matriz de confusão sem normalização
4 plt.figure()
5 plot_confusion_matrix(cnf_matrix, classes= fset.estado_fisico.unique(), normalize= False, title='Matriz de Confusão')

          precision    recall  f1-score   support
Ileso          0.65      0.73      0.68     11379
Lesões Graves    0.00      0.00      0.00      3055
Lesões Leves       0.55      0.69      0.61     10766
Óbito            0.00      0.00      0.00      886

```

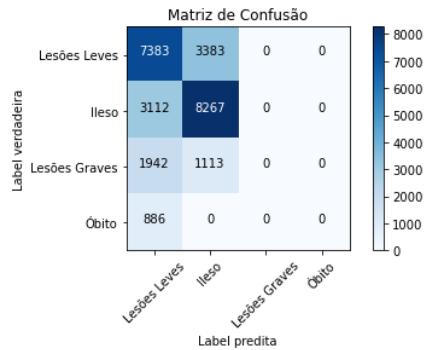
⁹ https://www.python-course.eu/confusion_matrix.php

¹⁰ Foi retirada do curso de Machine Learning da Cognitiveclass

accuracy			0.60	26086
macro avg	0.30	0.35	0.32	26086
weighted avg	0.51	0.60	0.55	26086

Matriz de confusão sem normalização

```
[[7383 3383 0 0]
 [3112 8267 0 0]
 [1942 1113 0 0]
 [ 886 0 0 0]]
```



Passou-se ao teste com um algoritmo de árvore de decisão, o `DecisionTreeClassifier`, e que é um método de aprendizado supervisionado não paramétrico usado para classificação e regressão¹¹. O objetivo é criar um modelo que preveja o valor de uma variável de destino, aprendendo regras de decisão simples inferidas dos recursos de dados. Uma árvore pode ser vista como uma aproximação constante por partes.

O `DecisionTreeClassifier` usa o índice de *Gini* e a entropia como critérios para calcular a qualidade da divisão do nó: (i) ‘gini’, observa a impureza do nó; (ii) ‘entropy’, observa o ganho de informação. Um nó com várias classes é impuro, enquanto um nó com apenas uma classe é puro¹².

A medição de Gini é a probabilidade de uma amostra aleatória ser classificada incorretamente quando se escolhe aleatoriamente um rótulo de acordo com a distribuição em um ramo. Entropia é uma medida de informação. Assim que uma determinada observação tem sua entropia reduzida a zero, forma-se uma folha na árvore, e observações com as mesmas características passam a ser agrupadas nesta determinada folha. Enquanto as outras informações continuam sendo processadas e subdividindo-se em galhos e novas folhas.

¹¹ <https://scikit-learn.org/stable/modules/tree.html>

¹² <https://michael-fuchs-python.netlify.app/2019/11/30/introduction-to-decision-trees/>

O primeiro teste com o algoritmo de árvore de decisão empregou a configuração padrão do algoritmo e que implementa o critério do índice de Gini. Ver Figura 86.

**Figura 86: Teste do algoritmo Decision Tree Classifier
(autor)**

```
#Importando bibliotecas para trabalhar com o algoritmo Decision Tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
import sklearn.metrics as m

#Teste com o algoritmo Decision Tree
dtclf = DecisionTreeClassifier()
dtclf.fit(X_trainset,Y_trainset)

DecisionTreeClassifier()

#Armazenamento de previsões em duas variáveis: uma para os dados de treino e outra para os dados de teste
Y_PredTE = dtclf.predict(X_testset)
Y_PredTR = dtclf.predict(X_trainset)
```

O modelo foi então avaliado de acordo com a acurácia obtida, o índice de Jaccard obtido e uma validação cruzada (cross validation). A acurácia por meio do *accuracy_score* e a média do *cross_validation* apresentaram scores muito próximos, 0.64 e 0.635, respectivamente. Ver Figura 87.

**Figura 87: Avaliação do modelo do Decision Tree Classifier
(autor)**

```
#Verificando a acurácia nos dados de teste e de treino
print('Acurácia p/ dados de Teste: {:.2f}'.format(accuracy_score(Y_testset, Y_PredTE)))
print('Acurácia p/ dados de Treino: {:.2f}'.format(accuracy_score(Y_trainset, Y_PredTR)))

Acurácia p/ dados de Teste: 0.64
Acurácia p/ dados de Treino: 0.96

#Verificando erros de classificação nos dados de teste e de treino
print('Erros de classificação das amostras Teste: {}'.format((Y_testset != Y_PredTE).sum()),'\nAmostras de teste "y": {}'.format((Y_testset == 1).sum()))
print('Erros de classificação das amostras Treino: {}'.format((Y_trainset != Y_PredTR).sum()),'\nAmostras de treino "y": {}'.format((Y_trainset == 1).sum()))

Erros de classificação das amostras Teste: 9407
Amostras de teste "y": 26086
Erros de classificação das amostras Treino: 2623
Amostras de treino "y": 60866

#Obtendo o índice Jaccard dos dados de teste e treino
print('Índice Jaccard p/ dados de Teste: {:.3f}%'.format(m.jaccard_score(Y_testset, Y_PredTE, average='micro')))
print('Índice Jaccard p/ dados de Treino: {:.3f}%'.format(m.jaccard_score(Y_trainset, Y_PredTR, average='micro')))

Índice Jaccard p/ dados de Teste: 0.470%
Índice Jaccard p/ dados de Treino: 0.917%

#Avaliação do modelo com cross validation
scores = cross_val_score(dtclf, X_trainset, Y_trainset, cv=10)
print("Média Cross-Validation: {:.3f} (std: {:.3f})".format(scores.mean(),
                                                       scores.std()),
      end="\n\n")
#a acurácia não aumentou

Média Cross-Validation: 0.635 (std: 0.004)
```

O próximo passo foi identificar os hiper parâmetros mais adequados para o modelo. Empregou-se o método *GridSearch* para encontrar os referidos parâmetros, conforme a Figura 88. O resultado mostrou que os parâmetros “criterion”: ‘gini’,

'max_depth': 10, 'max_leaf_nodes': None, 'min_samples_leaf': 10, 'min_samples_split': 15" fornecem um melhor score para o modelo. Na sequência, tais parâmetros foram implementados.

Figura 88: Parâmetros mais adequados para o Decision Tree Classifier (autor)

```
#Encontrando os hiper parâmetros mais adequados por meio do método GridSearch
param_grid = {"criterion": ["gini", "entropy"],
             "min_samples_split": [2, 5, 10, 15, 20],
             "max_depth": [None, 2, 3, 5, 7, 10],
             "min_samples_leaf": [1, 3, 5, 7, 10],
             "max_leaf_nodes": [None, 3, 5, 7, 10, 15, 20],
             }

grid = GridSearchCV(dtclf, param_grid, cv=10, scoring='accuracy')
grid.fit(X_trainset, Y_trainset)

GridSearchCV(cv=10,
            estimator=DecisionTreeClassifier(criterion='entropy',
                                              min_samples_split=4),
            param_grid=[{"criterion": ["gini", "entropy"],
                         "max_depth": [None, 2, 3, 5, 7, 10],
                         "max_leaf_nodes": [None, 3, 5, 7, 10, 15, 20],
                         "min_samples_leaf": [1, 3, 5, 7, 10],
                         "min_samples_split": [2, 5, 10, 15, 20]},
                         ],
            scoring='accuracy')

#O melhor score obtido entre todos os parâmetros
print(grid.best_score_)

0.7130583616107357

#Relação dos hiper parâmetros usados para gerar o melhor score
print(grid.best_params_)

{'criterion': 'gini', 'max_depth': 10, 'max_leaf_nodes': None, 'min_samples_leaf': 10, 'min_samples_split': 15}
```

Com a implementação dos hiper parâmetros mais adequados e nova execução do algoritmo, obteve-se um ganho de acurácia e uma quantidade menor de erros de classificação das amostras de teste, embora tenham tidos mais erros de classificação nas amostras de treinamento.

Figura 89: Decision Tree Classifier e os hiper parâmetros adequados (autor)

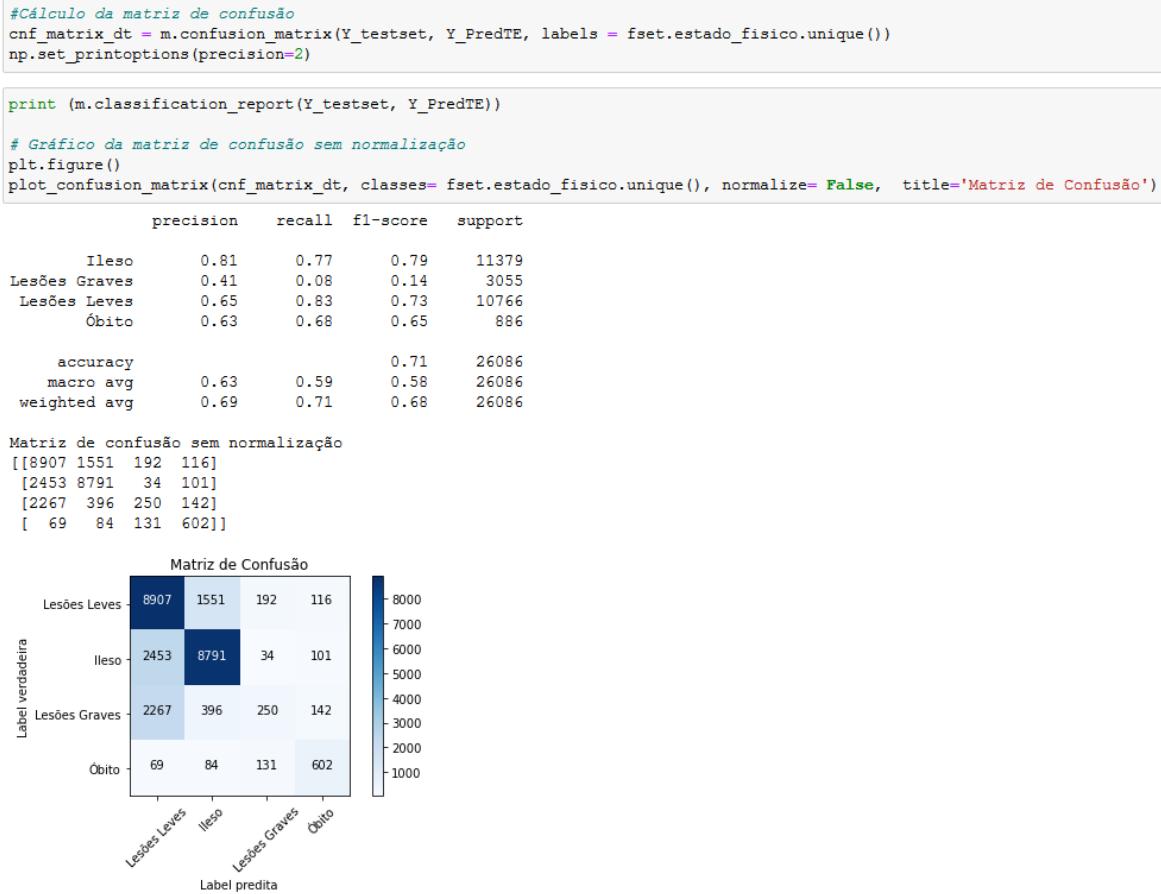
```
#Teste com o algoritmo DecisionTree e os hiper parâmetros mais adequados
dtclf = DecisionTreeClassifier(criterion = 'gini', max_depth=10, max_leaf_nodes=None, min_samples_leaf=10, min_samples_split=1
dtclf.fit(X_trainset,Y_trainset)
Y_PredTE = dtclf.predict(X_testset)
Y_PredTR = dtclf.predict(X_trainset)
print('Acurácia p/ dados de Teste: {:.2f}'.format(accuracy_score(Y_testset, Y_PredTE)))
print('Acurácia p/ dados de Treino: {:.2f}'.format(accuracy_score(Y_trainset, Y_PredTR)))
print('Erros de classificação das amostras Teste: {}'.format((Y_testset != Y_PredTE).sum()),'\nAmostras de teste "y": {}'.format(Y_testset.sum()))
print('Erros de classificação das amostras Treino: {}'.format((Y_trainset != Y_PredTR).sum()),'\nAmostras de treino "y": {}'.format(Y_trainset.sum()))
print('Índice Jaccard p/ dados de Teste: {:.3f}%'.format(m.jaccard_score(Y_testset, Y_PredTE, average='micro')))
print('Índice Jaccard p/ dados de Treino: {:.3f}%'.format(m.jaccard_score(Y_trainset, Y_PredTR, average='micro')))

Acurácia p/ dados de Teste: 0.71
Acurácia p/ dados de Treino: 0.72
Erros de classificação das amostras Teste: 7536
Amostras de teste "y": 26086
Erros de classificação das amostras Treino: 16989
Amostras de treino "y": 60866
Índice Jaccard p/ dados de Teste: 0.552%
Índice Jaccard p/ dados de Treino: 0.564%
```

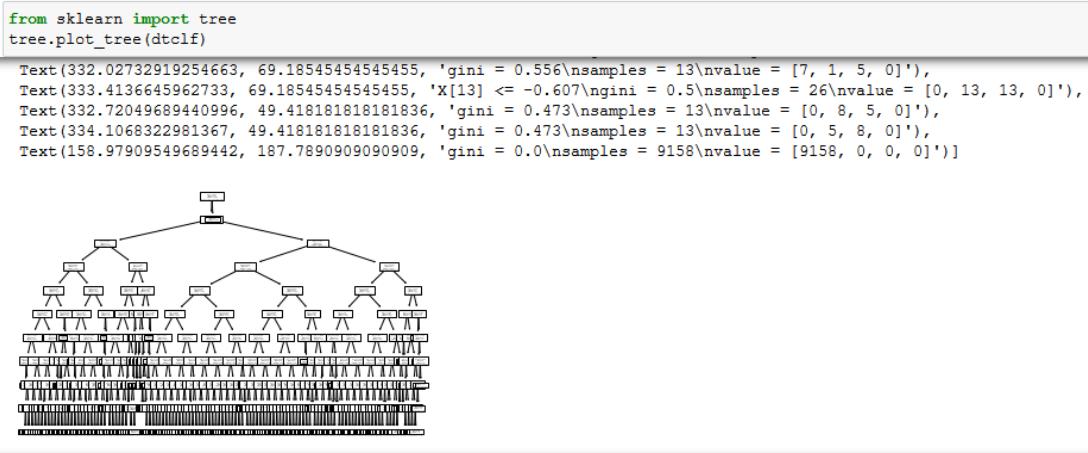
Na sequência, produziu-se um relatório de classificação e a matriz de confusão com o resultado do modelo do algoritmo DecisionTreeClassifier. A acurácia

obtida foi de 0.71, conforme Figura 90. E foi também gerado o *plot* da árvore do modelo, conforme Figura 91.

**Figura 90: Matriz de Confusão do Decision Tree Classifier
(autor)**



**Figura 91: Árvore gerada pelo modelo do Decision Tree Classifier
(autor)**



O último algoritmo testado foi o RandomForestClassifier. O Random Forest consiste em um grande número de árvores de decisão individuais que operam como

um conjunto ou *ensemble*¹³. Cada árvore individual em uma floresta aleatória exibe uma previsão de classe e a classe com mais votos torna-se a previsão de nosso modelo.

O conceito fundamental por trás do Random Forest é simples, mas poderoso: a sabedoria das multidões. Na ciência de dados, o motivo pelo qual o modelo aleatório de floresta funciona tão bem é: “um grande número de modelos (árvores) relativamente não correlacionados operando como um comitê terá um desempenho melhor do que qualquer um dos modelos constituintes individuais”¹⁴.

O teste com o algoritmo Random Forest foi realizado, inicialmente, com as configurações padrão, conforme Figura 92. Foi efetuada a avaliação do modelo quanto ao resultado do primeiro teste e obteve-se uma acurácia aproximada de 0.71, segundo a Figura 93.

Figura 92: Teste do algoritmo Random Forest Classifier
(autor)

```
#Importando bibliotecas para trabalhar com o algoritmo Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV

#Teste com o algoritmo Random Forest Classifier
rfclf = RandomForestClassifier()
rfclf.fit(X_trainset,Y_trainset)

RandomForestClassifier()

#Armazenamento de previsões em duas variáveis: uma para os dados de treino e outra para os dados de teste
Y_PredTE2 = rfclf.predict(X_testset)
Y_PredTR2 = rfclf.predict(X_trainset)
```

Figura 93: Avaliação do modelo do Random Forest Classifier
(autor)

```
#Verificando a acurácia nos dados de teste e de treino
print('Acurácia p/ dados de Teste: {:.2f}'.format(accuracy_score(Y_testset, Y_PredTE2)))
print('Acurácia p/ dados de Treino: {:.2f}'.format(accuracy_score(Y_trainset, Y_PredTR2)))

Acurácia p/ dados de Teste: 0.71
Acurácia p/ dados de Treino: 0.96

#Verificando erros de classificação nos dados de teste e de treino
print('Erros de classificação das amostras Teste: {}'.format((Y_testset != Y_PredTE2).sum()),'\nAmostras de teste "y": {}'.format((Y_testset == Y_PredTE2).sum()))
print('Erros de classificação das amostras Treino: {}'.format((Y_trainset != Y_PredTR2).sum()),'\nAmostras de treino "y": {}'.format((Y_trainset == Y_PredTR2).sum()))

Erros de classificação das amostras Teste: 7601
Amostras de teste "y": 26086
Erros de classificação das amostras Treino: 2623
Amostras de treino "y": 60866
```

¹³ Em estatística e *machine learning*, os métodos de *ensemble* usam vários algoritmos para obter um melhor desempenho preditivo do que poderia ser obtido com qualquer um dos algoritmos sozinho.

¹⁴ <https://towardsdatascience.com/understanding-random-forest-58381e0602d2>

```
#Obtendo o índice Jaccard dos dados de teste e treino
print('Índice Jaccard p/ dados de Teste: {:.3f}'.format(m.jaccard_score(Y_testset, Y_PredTE2, average='micro')))
print('Índice Jaccard p/ dados de Treino: {:.3f}'.format(m.jaccard_score(Y_trainset, Y_PredTR2, average='micro')))

Índice Jaccard p/ dados de Teste: 0.549%
Índice Jaccard p/ dados de Treino: 0.917%


#Avaliação do modelo com cross validation
scores = cross_val_score(fclf, X_trainset, Y_trainset, cv=5)
scores

array([0.7 , 0.71, 0.7 , 0.71, 0.7 ])


print('KFold: Score Médio da Acurácia = {}'.format(np.mean(scores)))

KFold: Score Médio da Acurácia = 0.7043669673169584
```

O passo seguinte foi identificar os hiper parâmetros mais adequados para o modelo. Empregou-se o método *RandomizedSearchCV* para encontrar os referidos parâmetros, conforme a Figura 94. O resultado mostrou que os parâmetros "n_estimators": 140, 'min_samples_split': 9, 'max_features': 8, 'max_depth': 14, 'criterion': 'entropy', 'bootstrap': True fornecem um melhor score para o modelo. Foi realizado um *ranking* do modelo com as combinações de parâmetros.

Figura 94: Parâmetros mais adequados para o Random Forest Classifier (autor)

```
#Encontrando os hiper parâmetros mais adequados por meio do método GridSearch
param_dist = {"n_estimators": list(range(10,210,10)),
              "max_depth": list(range(3,20)),
              "max_features": list(range(1, 10)),
              "min_samples_split": list(range(2, 11)),
              "bootstrap": [True, False],
              "criterion": ["gini", "entropy"]}

n_iter_search = 50

random_search = RandomizedSearchCV(rfclf, param_distributions=param_dist, scoring='accuracy',
                                    n_iter=n_iter_search)
random_search.fit(X_trainset, Y_trainset)

RandomizedSearchCV(estimator=RandomForestClassifier(), n_iter=50,
                    param_distributions={'bootstrap': [True, False],
                                         'criterion': ['gini', 'entropy'],
                                         'max_depth': [3, 4, 5, 6, 7, 8, 9, 10,
                                                       11, 12, 13, 14, 15, 16,
                                                       17, 18, 19],
                                         'max_features': [1, 2, 3, 4, 5, 6, 7, 8,
                                                          9],
                                         'min_samples_split': [2, 3, 4, 5, 6, 7,
                                                               8, 9, 10],
                                         'n_estimators': [10, 20, 30, 40, 50, 60,
                                                          70, 80, 90, 100, 110,
                                                          120, 130, 140, 150,
                                                          160, 170, 180, 190,
                                                          200]},
                    scoring='accuracy')

#Identificando os melhores hiper parâmetros
print('Melhor número de estimadores:', random_search.best_estimator_.get_params()['n_estimators'])
print('Melhor min_samples_split:', random_search.best_estimator_.get_params()['max_depth'])

Melhor número de estimadores: 140
Melhor min_samples_split: 14

#Relação dos hiper parâmetros usados para gerar o melhor score
random_search.best_params_
```

```
{'n_estimators': 140,
 'min_samples_split': 9,
 'max_features': 8,
 'max_depth': 14,
 'criterion': 'entropy',
 'bootstrap': True}
```

```
#Rank de modelos de acordo com a configuração de hyper parâmetros
results = pd.DataFrame(random_search.cv_results_).sort_values('rank_test_score')
for i, row in results.head().iterrows():
    print("Rank do modelo: {}".format(row.rank_test_score))
    print("Score médio de validação: {:.3f} (std: {:.3f})".format(row.mean_test_score, row.std_test_score))
    print("Modelo de hiper parâmetros: {}\\n".format(row.params))

Rank do modelo: 1
Score médio de validação: 0.723 (std: 0.004)
Modelo de hiper parâmetros: {'n_estimators': 140, 'min_samples_split': 9, 'max_features': 8, 'max_depth': 14, 'criterion': 'entropy', 'bootstrap': True}

Rank do modelo: 2
Score médio de validação: 0.723 (std: 0.003)
Modelo de hiper parâmetros: {'n_estimators': 90, 'min_samples_split': 4, 'max_features': 9, 'max_depth': 13, 'criterion': 'gini', 'bootstrap': True}

Rank do modelo: 3
Score médio de validação: 0.722 (std: 0.004)
Modelo de hiper parâmetros: {'n_estimators': 100, 'min_samples_split': 4, 'max_features': 6, 'max_depth': 15, 'criterion': 'gini', 'bootstrap': False}

Rank do modelo: 4
Score médio de validação: 0.722 (std: 0.003)
Modelo de hiper parâmetros: {'n_estimators': 120, 'min_samples_split': 3, 'max_features': 8, 'max_depth': 13, 'criterion': 'entropy', 'bootstrap': False}

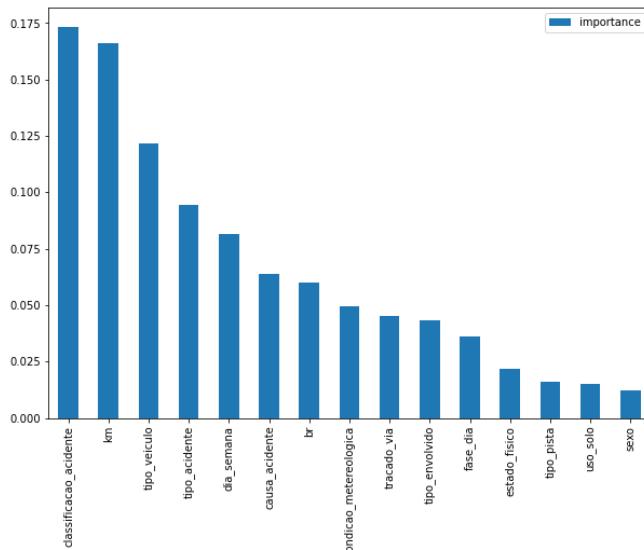
Rank do modelo: 5
Score médio de validação: 0.722 (std: 0.004)
Modelo de hiper parâmetros: {'n_estimators': 50, 'min_samples_split': 4, 'max_features': 9, 'max_depth': 16, 'criterion': 'gini', 'bootstrap': True}
```

Aproveitou-se e realizou-se uma verificação do atributo (ou feature) de importância para o modelo. Percebe-se, pela Figura 96, que a 'classificacao_acidente' (acidentes sem ou com vítimas – feridas ou graves), seguido de 'km', é a *feature* mais relevante no modelo. Conhecer a *feature* de importância, indicada pelo modelo de *machine learning*, pode favorecer a um melhor entendimento da lógica do modelo, concentrando-se nas variáveis importantes.

Figura 95: Verificação da *feature* de importância (autor)

```
#Determinação de atributo (feature) de importância
feat_imps = pd.DataFrame({'importance': fclf.feature_importances_}, index=fset.columns[:-1])
feat_imps.sort_values(by='importance', ascending=False, inplace=True)
```

```
feat_imps.plot(kind='bar', figsize=(10,7))
plt.legend()
plt.show()
```



A partir da implementação dos hiper parâmetros mais adequados e com uma nova execução do algoritmo, obteve-se um ganho de acurácia (0.72) e uma quantidade menor de erros de classificação das amostras de teste, embora tenham tidos mais erros de classificação nas amostras de treinamento.

Figura 96: Random Forest Classifier e os hiper parâmetros adequados (autor)

```
#Teste com o algoritmo RandomForest e os hyper parâmetros mais adequados
rfclf = RandomForestClassifier(n_estimators=140, min_samples_split=9, max_features=9, max_depth=13, criterion='gini', bootstrap=True)
rfclf.fit(X_trainset,Y_trainset)
Y_PredTE2 = rfclf.predict(X_testset)
Y_PredTR2 = rfclf.predict(X_trainset)

print('Acurácia p/ dados de Teste: {:.2f}'.format(accuracy_score(Y_testset, Y_PredTE2)))
print('Acurácia p/ dados de Treino: {:.2f}'.format(accuracy_score(Y_trainset, Y_PredTR2)))
print('Erros de classificação das amostras Teste: {}'.format((Y_testset != Y_PredTE2).sum()),'\nAmostras de teste "y": {}'.format(len(Y_testset)))
print('Erros de classificação das amostras Treino: {}'.format((Y_trainset != Y_PredTR2).sum()),'\nAmostras de treino "y": {}'.format(len(Y_trainset)))
print('Índice Jaccard p/ dados de Teste: {:.3f}%'.format(m.jaccard_score(Y_testset, Y_PredTE2, average='micro')))
print('Índice Jaccard p/ dados de Treino: {:.3f}%'.format(m.jaccard_score(Y_trainset, Y_PredTR2, average='micro')))

Acurácia p/ dados de Teste: 0.72
Acurácia p/ dados de Treino: 0.75
Erros de classificação das amostras Teste: 7283
Amostras de teste "y": 26086
Erros de classificação das amostras Treino: 15029
Amostras de treino "y": 60866
Índice Jaccard p/ dados de Teste: 0.563%
Índice Jaccard p/ dados de Treino: 0.604%
```

Por fim, produziu-se um relatório de classificação e a matriz de confusão com o resultado do modelo do algoritmo RandomForestClassifier. A acurácia obtida foi de 0.72, conforme Figura 97.

De todos os algoritmos testados (SGDclassifier – log e hinge; DecisionTreeClassifier; e RandomForestClassifier), o Random Forest foi o que entregou o score de acurácia mais elevado, embora se desejasse um valor superior a 0.72, foi o melhor que se obteve nos presentes testes.

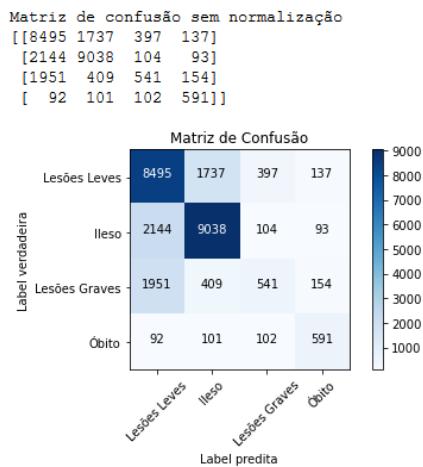
Figura 97: Matriz de Confusão do Random Forest Classifier (autor)

```
#Cálculo da matriz de confusão
cnf_matrix_rf = m.confusion_matrix(Y_testset, Y_PredTE2, labels = fset.estado_fisico.unique())
np.set_printoptions(precision=2)

print (m.classification_report(Y_testset, Y_PredTE2))

# Gráfico da matriz de confusão sem normalização
plt.figure()
plot_confusion_matrix(cnf_matrix_rf, classes= fset.estado_fisico.unique(), normalize= False, title='Matriz de Confusão')

precision    recall   f1-score   support
Ileso        0.80      0.79      0.80     11379
Lesões Graves   0.47      0.18      0.26      3055
Lesões Leves    0.67      0.79      0.72     10766
Óbito         0.61      0.67      0.64      886
accuracy                           0.72     26086
macro avg       0.64      0.61      0.60     26086
weighted avg    0.70      0.72      0.70     26086
```



Considerando os resultados acima, passou-se à implementação do modelo escolhido, com o Random Forest, para efetivar os resultados do algoritmo num conjunto de dados para estudo. Desta forma, criou-se um novo *dataframe* (fset2) com os mesmos *feature sets* de “acidentes_mg”, usados anteriormente nos testes dos algoritmos, ('dia_semana', 'br', 'km', 'causa_acidente', 'tipo_acidente', 'classificacao_acidente', 'fase_dia', 'condicao_metereologica', 'tipo_pista', 'tracado_via', 'uso_solo', 'tipo_veiculo', 'tipo_envolvido', 'estado_fisico', 'sexo', 'elemento_transito'). Figura 98.

Figura 98: Montagem de um novo dataframe para implementar o modelo escolhido (autor)

```
#Montagem de um novo dataset para implementar o modelo escolhido de ML, o do RandomForest
acidentes_mg.reset_index(inplace = True)
fset2 = acidentes_mg[['dia_semana','br','km','causa_acidente','tipo_acidente','classificacao_acidente','fase_dia','condicao_metereologica','tipo_pista','tracado_via','uso_solo','tipo_v']]
fset2.head()
```

	dia_semana	br	km	causa_acidente	tipo_acidente	classificacao_acidente	fase_dia	condicao_metereologica	tipo_pista	tracado_via	uso_solo	tipo_v
0	domingo	381	605.0	Condutor Dormindo	Saída de leito carroável	Com Vítimas Feridas	Amanhecer	Céu Claro	Dupla	Reta	Rural	Autc
1	domingo	381	605.0	Condutor Dormindo	Saída de leito carroável	Com Vítimas Feridas	Amanhecer	Céu Claro	Dupla	Reta	Rural	Autc
2	domingo	262	368.0	Velocidade Incompatível	Colisão com objeto estático	Com Vítimas Feridas	Plena Noite	Céu Claro	Simples	Viaduto	Urbano	Autc
3	domingo	262	368.0	Velocidade Incompatível	Colisão com objeto estático	Com Vítimas Feridas	Plena Noite	Céu Claro	Simples	Viaduto	Urbano	Autc
4	domingo	262	368.0	Velocidade Incompatível	Colisão com objeto estático	Com Vítimas Feridas	Plena Noite	Céu Claro	Simples	Viaduto	Urbano	Autc

Foram necessários alguns procedimentos auxiliares para o trabalho, como a criação de uma variável (f1) com os índices dos registros de ‘estado_fisico’ = ‘Não Informado’, de maneira que se pudesse saber onde seriam realizadas as alterações com os valores preditivos. Na sequência, eliminou-se a coluna de ‘estado_fisico’ do *dataframe* fset2. Figura 99.

Figura 99: Procedimentos auxiliares e verificações do novo dataframe (autor)

```
#Juntando os indices dos registros de 'estado_fisico' = 'Não Informado' na variável f1
f1 = fset2.loc[fset2['estado_fisico'] == 'Não Informado'].index

#Verificando a variável f1
f1

Int64Index([ 22,    45,    62,   102,   163,   164,   165,   180,   242,   243,
             ...
            91643, 91644, 91654, 91655, 91701, 91702, 91711, 91729, 91775, 91784], dtype='int64', length=4837)

#Tratamento dos registros que o estado_fisico está como Não Informado - retirando a coluna alvo (target)
fset2.drop(labels= 'estado_fisico', axis= 1, inplace= True)

#Verificando as colunas de fset2
fset2.columns

Index(['dia_semana', 'br', 'km', 'causa_acidente', 'tipo_acidente', 'classificacao_acidente', 'fase_dia', 'condicao_meteorologica', 'tipo_pista', 'tracado_via', 'uso_solo', 'tipo_veiculo', 'tipo_envolvido', 'sexo', 'elemento_transito'], dtype='object')

#Verificando os valores de fset2
fset2.values

array([['domingo', 381, 605.0, ..., 'Condutor', 'Masculino', 'Homem'],
       ['domingo', 381, 605.0, ..., 'Passageiro', 'Masculino', 'Homem'],
       ['domingo', 262, 368.0, ..., 'Passageiro', 'Masculino', 'Homem'],
       ...
       ['sexta-feira', 116, 648.0, ..., 'Condutor', 'Feminino', 'Homem'],
       ['sexta-feira', 116, 648.0, ..., 'Condutor', 'Masculino', 'Homem'],
       ['sabado', 153, 7.0, ..., 'Condutor', 'Masculino', 'Via']],
      dtype=object)
```

Os valores de fset2 foram armazenados na variável X2 e, então, transformados de não numéricos para numéricos usando a função *LabelEncoder*, já existente no código. Na sequência, procedeu-se com a normalização dos valores (Figura 100).

Figura 100: Transformação de dados não numéricos e normalização dos dados de X2 (autor)

```
#Transformando os valores de X2 em numéricos e normalizando os dados (X2std)
X2 = fset2.values
X2[:, 0] = LE_Dia_Semana.transform(X2[:, 0])
X2[:, 3] = LE_Causa.transform(X2[:, 3])
X2[:, 4] = LE_Tipo_Accidente.transform(X2[:, 4])
X2[:, 5] = LE_Class_acidente.transform(X2[:, 5])
X2[:, 6] = LE_Fase_Dia.transform(X2[:, 6])
X2[:, 7] = LE_Condicao_Met.transform(X2[:, 7])
X2[:, 8] = LE_Tipo_Pista.transform(X2[:, 8])
X2[:, 9] = LE_Tracado_Via.transform(X2[:, 9])
X2[:, 10] = LE_Uso_Solo.transform(X2[:, 10])
X2[:, 11] = LE_Tipo_Veiculo.transform(X2[:, 11])
X2[:, 12] = LE_Tipo_Envoltido.transform(X2[:, 12])
X2[:, 13] = LE_Sexo.transform(X2[:, 13])
X2[:, 14] = LE_Elemento_Transito.transform(X2[:, 14])
X2std = std.transform(X2)

#O novo dataset (fset2) recebe uma nova coluna 'estado_fisico' apenas com as previsões do modelo escolhido de ML, o do Random Forest
fset2['estado_fisico'] = rfclf.predict(X2std)
```

E, então, o novo *dataframe* fset2 recebe uma coluna (ou feature) chamado 'estado_fisico' com os dados das previsões do modelo escolhido, o Random Forest. É realizada uma verificação das colunas de fset2 e de seus valores. Tanto a coluna 'estado_fisico' quanto os seus valores estão presentes em fset2. Ver Figura 101.

Figura 101: Inclusão de coluna ‘estado_fisico’ com valores da predição em fset2 (autor)

```
#O novo dataset (fset2) recebe uma nova coluna 'estado_fisico' apenas com as previsões do modelo escolhido de ML, o do Random
fset2['estado_fisico'] = rfclf.predict(X2std)
!!!

#Verificando as colunas fset2 com a inclusão de 'estado_fisico'
fset2.columns

Index(['dia_semana', 'br', 'km', 'causa_acidente', 'tipo_acidente', 'classificacao_acidente', 'fase_dia', 'condicao_meteorologica', 'tipo_pista', 'tracado_via', 'uso_solo', 'tipo_veiculo', 'tipo_envolvido', 'sexo', 'elemento_transito', 'estado_fisico'], dtype='object')

#Verificando os valores de fset2 com a inclusão de 'estado_fisico'
fset2.values

array([['domingo', 381, 605.0, ..., 'Masculino', 'Homem', 'Lesões Leves'],
       ['domingo', 381, 605.0, ..., 'Masculino', 'Homem', 'Lesões Leves'],
       ['domingo', 262, 368.0, ..., 'Masculino', 'Homem', 'Lesões Leves'],
       ...,
       ['sexta-feira', 116, 648.0, ..., 'Feminino', 'Homem',
        'Lesões Leves'],
       ['sexta-feira', 116, 648.0, ..., 'Masculino', 'Homem', 'Ileso'],
       ['sábado', 153, 7.0, ..., 'Masculino', 'Via', 'Ileso']],
      dtype=object)
```

Agora, é realizada uma comparação entre a *feature* ‘estado_fisico’ presente em “acidentes_mg” e em “fset2”. Percebe-se que a *feature* existe em ambos os *dataframes*. Ver Figura 102.

Figura 102: Verificação de “acidentes_mg” e de “fset2” para o atributo de ‘estado_fisico’ (autor)

```
#Verificando as colunas de acidentes_mg
acidentes_mg.columns

Index(['index', 'id', 'pesid', 'data_inversa', 'dia_semana', 'horario', 'uf', 'br', 'km', 'municipio', 'causa_acidente', 'tipo_acidente', 'classificacao_acidente', 'fase_dia', 'condicao_meteorologica', 'tipo_pista', 'tracado_via', 'uso_solo', 'tipo_veiculo', 'tipo_envolvido', 'estado_fisico', 'sexo', 'ilesos', 'feridos_leves', 'feridos_graves', 'mortos', 'latitude', 'longitude', 'elemento_transito'], dtype='object')

acidentes_mg.iloc[:10,20]

0    Lesões Leves
1    Lesões Leves
2    Lesões Leves
3    Lesões Leves
4    Lesões Leves
5        Ileso
6    Lesões Leves
7        Ileso
8        Ileso
9    Lesões Leves
Name: estado_fisico, dtype: object

fset2.iloc[:10,15]

0    Lesões Leves
1    Lesões Leves
2    Lesões Leves
3    Lesões Leves
4    Lesões Leves
5        Ileso
6    Lesões Leves
7    Lesões Leves
8    Lesões Leves
9    Lesões Leves
Name: estado_fisico, dtype: object
```

Agora, executam-se duas rotinas que: (i) emprega o método ‘.iloc’ com indexador para selecionar dados por posição, nesse caso o número da linha, usando como referência a variável ‘f1’ – ambos os *dataframes* possuem a mesma quantidade de linhas e com os registros, embora com *features* distintas; (ii) usa o

método ‘.at’ para acessar um valor para linha / coluna de maneira que a *feature* ‘estado_fisico’ de “acidentes_mg” recebe o valor da *feature* ‘estado_fisico’ de “fset2”. Na sequência, salvam-se as modificações e armazenam-se os dados em uma nova versão csv de acidentes. Figura 103.

Figura 103: Atualização dos dados de predição em “acidentes_mg”
(autor)

```
#Atualização das informações em um novo "dataset" acidentes_mg - parte 1
with pd.option_context('max_column',45): #Usado na construção do loop para ver cada registro
    for i in f1:
        acidentes_mg.iloc[i,20] = fset2.iloc[i,15]

#Atualização das informações em um novo "dataset" acidentes_mg - parte 2
for i in f1:
    acidentes_mg.at[i,'estado_fisico'] = fset2.at[i,'estado_fisico']

#Salvando as modificações feitas do novo "dataset" acidentes_mg
acidentes_mg.to_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\acidentes_mg4.csv", index = False, header = True, sep=';', decimal=',')

!!!
```

Superada esse passo, ainda são necessários ajustes no *dataframe* “acidentes_mg” para que as *features* ‘ilesos’, ‘feridos_leves’, ‘feridos_graves’, ‘mortos’ represente o que foi alterado em ‘estado_fisico’ com os valores das predições. Inicia-se com a leitura e verificação do novo arquivo “acidentes_mg”. Figura 104.

Figura 104: Verificação do dataframe “acidentes_mg” modificado
(autor)

```
#Verificação e ajuste dos dados em acidentes com as predições
acidentes_mg = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\acidentes_mg4.csv", sep=';', decimal=',', encoding = 'cp1252')
acidentes_mg.drop(labels=['index'],axis=1, inplace=True)

#Verificando o dataset modificado de acidentes
acidentes_mg.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 91789 entries, 0 to 91788
Data columns (total 28 columns):
 #   Column           Non-Null Count  Dtype  
 ---  --  
 0    id               91789 non-null   int64  
 1    pesid             91789 non-null   int64  
 2    data_inversa      91789 non-null   datetime64[ns]
 3    dia_semana        91789 non-null   object 
 4    horario            91789 non-null   object 
 5    uf                91789 non-null   object 
 6    br                91789 non-null   int64  
 7    km                91789 non-null   float64
 8    municipio          91789 non-null   object 
 9    causa_acidente    91789 non-null   object 
 10   tipo_acidente     91789 non-null   object 
 11   classificacao_acidente 91789 non-null   object 
 12   fase_dia           91789 non-null   object 
 13   condicao_metereologica 91789 non-null   object 
 14   tipo_pista          91789 non-null   object 
 15   tracado_via         91789 non-null   object 
 16   uso_solo            91789 non-null   object 
 17   tipo_veiculo        91789 non-null   object 
 18   tipo_envolvido      91789 non-null   object 
 19   estado_fisico        91789 non-null   object 
 20   sexo                91789 non-null   object 
 21   ilesos              91789 non-null   int64  
 22   feridos_leves       91789 non-null   int64  
 23   feridos_graves      91789 non-null   int64  
 24   mortos              91789 non-null   int64  
 25   latitude             91789 non-null   float64
 26   longitude            91789 non-null   float64
 27   elemento_transito    91789 non-null   object
```

```
#Verificando a f1 para que seja comparada com acidentes_mg
f1
```

```
Int64Index([ 22, 45, 62, 102, 163, 164, 165, 180, 242, 243,
... 91643, 91644, 91654, 91655, 91701, 91702, 91711, 91729, 91775, 91784], dtype='int64', length=4837)
```

```
#Verificando o dataset modificado de acidentes
acidentes_mg.head(23)
```

	id	pesid	data_inversa	dia_semana	horario	uf	br	km	municipio	causa_acidente	tipo_acidente	classificacao_acidente	fase_dia	condica
0	47	44	2017-01-01	domingo	04:50:00	MG	381	605.0	OLIVEIRA	Condutor Dormindo	Saída de leito carroçável	Com Vítimas Feridas	Amanhecer	
1	47	45	2017-01-01	domingo	04:50:00	MG	381	605.0	OLIVEIRA	Condutor Dormindo	Saída de leito carroçável	Com Vítimas Feridas	Amanhecer	
2	52	1528	2017-01-01	domingo	05:00:00	MG	262	368.0	JUATUBA	Velocidade Incompatível	Colisão com objeto estático	Com Vítimas Feridas	Plena Noite	
3	52	1526	2017-01-01	domingo	05:00:00	MG	262	368.0	JUATUBA	Velocidade Incompatível	Colisão com objeto estático	Com Vítimas Feridas	Plena Noite	
4	52	1519	2017-01-01	domingo	05:00:00	MG	262	368.0	JUATUBA	Velocidade Incompatível	Colisão com objeto estático	Com Vítimas Feridas	Plena Noite	
5	53	49	2017-01-01	domingo	05:00:00	MG	459	14.0	CALDAS	Falta de Atenção à Condução	Saída de leito carroçável	Sem Vítimas	Amanhecer	
6	61	38	2017-01-01	domingo	05:00:00	MG	262	186.0	SAO DOMINGOS DO PRATA	Condutor Dormindo	Saída de leito carroçável	Com Vítimas Feridas	Plena Noite	
7	63	1608	2017-01-01	domingo	06:00:00	MG	135	408.0	BOCAIUVA	Animais na Pista	Atropelamento de Animal	Com Vítimas Feridas	Amanhecer	
8	63	1606	2017-01-01	domingo	06:00:00	MG	135	408.0	BOCAIUVA	Animais na Pista	Atropelamento de Animal	Com Vítimas Feridas	Amanhecer	
9	63	1604	2017-01-01	domingo	06:00:00	MG	135	408.0	BOCAIUVA	Animais na Pista	Atropelamento de Animal	Com Vítimas Feridas	Amanhecer	
10	81	56	2017-01-01	domingo	07:50:00	MG	40	774.0	JUIZ DE FORA	Condutor Dormindo	Colisão com objeto estático	Com Vítimas Feridas	Pleno dia	
11	81	58	2017-01-01	domingo	07:50:00	MG	40	774.0	JUIZ DE FORA	Condutor Dormindo	Colisão com objeto estático	Com Vítimas Feridas	Pleno dia	
12	95	78	2017-01-01	domingo	08:30:00	MG	381	896.0	CAMBUI	Falta de Atenção à Condução	Saída de leito carroçável	Com Vítimas Feridas	Pleno dia	
13	95	81	2017-01-01	domingo	08:30:00	MG	381	896.0	CAMBUI	Falta de Atenção à Condução	Saída de leito carroçável	Com Vítimas Feridas	Pleno dia	
14	95	80	2017-01-01	domingo	08:30:00	MG	381	896.0	CAMBUI	Falta de Atenção à Condução	Saída de leito carroçável	Com Vítimas Feridas	Pleno dia	
15	95	79	2017-01-01	domingo	08:30:00	MG	381	896.0	CAMBUI	Falta de Atenção à Condução	Saída de leito carroçável	Com Vítimas Feridas	Pleno dia	
16	95	73	2017-01-01	domingo	08:30:00	MG	381	896.0	CAMBUI	Falta de Atenção à Condução	Saída de leito carroçável	Com Vítimas Feridas	Pleno dia	
17	96	89	2017-01-01	domingo	09:00:00	MG	50	81.0	UBERLANDIA	Ingestão de Álcool	Tombamento	Com Vítimas Feridas	Pleno dia	
18	98	1114	2017-01-01	domingo	08:30:00	MG	365	117.0	JEQUITAI	Velocidade Incompatível	Queda de ocupante de veículo	Com Vítimas Feridas	Pleno dia	
19	103	2061	2017-01-01	domingo	07:20:00	MG	40	550.0	NOVA LIMA	Velocidade Incompatível	Capotamento	Com Vítimas Feridas	Pleno dia	
20	103	2084	2017-01-01	domingo	07:20:00	MG	40	550.0	NOVA LIMA	Velocidade Incompatível	Capotamento	Com Vítimas Feridas	Pleno dia	
21	104	520	2017-01-01	domingo	08:35:00	MG	365	619.0	UBERLANDIA	Falta de Atenção à Condução	Colisão com objeto estático	Com Vítimas Feridas	Pleno dia	
22	104	3281	2017-01-01	domingo	08:35:00	MG	365	619.0	UBERLANDIA	Falta de Atenção à Condução	Colisão com objeto estático	Com Vítimas Feridas	Pleno dia	

tracado_via	uso_solo	tipo_veiculo	tipo_envolvido	estado_fisico	sexo	ilesos	feridos_leves	feridos_graves	mortos	latitude	longitude	elemento_transito
Reta	Rural	Automóvel	Condutor	Lesões Leves	Masculino	0	1	0	0	-20.6370	-44.7357	Homem
Reta	Rural	Automóvel	Passageiro	Lesões Leves	Masculino	0	1	0	0	-20.6370	-44.7357	Homem
Viaduto	Urbano	Automóvel	Passageiro	Lesões Leves	Masculino	0	1	0	0	-19.9566	-44.3444	Homem
Viaduto	Urbano	Automóvel	Passageiro	Lesões Leves	Feminino	0	1	0	0	-19.9566	-44.3444	Homem
Viaduto	Urbano	Automóvel	Condutor	Lesões Leves	Masculino	0	1	0	0	-19.9566	-44.3444	Homem
Curva	Rural	Automóvel	Condutor	Ileso	Feminino	1	0	0	0	-21.8459	-46.4388	Homem
Curva	Rural	Automóvel	Condutor	Lesões Leves	Masculino	0	1	0	0	-19.8942	-43.0403	Homem

```
#Análise e exploração da amostra dos dados - arquivo acidentes.CSV
acidentes_mg = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\acidentes_mg3.csv", sep=';', decimal=',', encoding = 'cp1251')
acidentes_mg.head(23)
```

	Reta	Rural	Automóvel	Passageiro	Iluso	Masculino	1	0	0	0	-16.9614	-43.8590	Via	
	Reta	Urbano	Automóvel	Testemunha	Lesões Leves	Não Informado	0	0	0	0	-18.8861	-48.2673	Homem	
#Análise e exploração da amostra dos dados - arquivo acidentes.CSV														
acidentes_mg = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\acidentes_mg3.csv", sep=';', decimal=',', encoding = 'cp1251')														
	id	pesid	data_inversa	dia_semana	horario	uf	br	km	municipio	causa_acidente	tipo_acidente	classificacao_acidente	fase_dia	condicao_
0	47	44	2017-01-01	domingo	04:50:00	MG	381	605	OLIVEIRA	Condutor Dormindo	Saída de leito carroçável	Com Vítimas Feridas	Amanhecer	
1	47	45	2017-01-01	domingo	04:50:00	MG	381	605	OLIVEIRA	Condutor Dormindo	Saída de leito carroçável	Com Vítimas Feridas	Amanhecer	
2	52	1528	2017-01-01	domingo	05:00:00	MG	262	368	JUATUBA	Velocidade Incompatível	Colisão com objeto estático	Com Vítimas Feridas	Plena Noite	
3	52	1526	2017-01-01	domingo	05:00:00	MG	262	368	JUATUBA	Velocidade Incompatível	Colisão com objeto estático	Com Vítimas Feridas	Plena Noite	
4	52	1519	2017-01-01	domingo	05:00:00	MG	262	368	JUATUBA	Velocidade Incompatível	Colisão com objeto estático	Com Vítimas Feridas	Plena Noite	
5	53	49	2017-01-01	domingo	05:00:00	MG	459	14	CALDAS	Falta de Atenção à Condução	Saída de leito carroçável	Sem Vítimas	Amanhecer	
22	104	3281	2017-01-01	domingo	08:35:00	MG	365	619	UBERLANDIA	Falta de Atenção à Condução	Colisão com objeto estático	Com Vítimas Feridas	Pleno dia	
	Reta	Urbano	Automóvel	Testemunha	Lesões Leves	Não Informado	0	0	0	0	-18.8861	-48.2673	Homem	

Foi realizado uma comparação entre o conteúdo de f1 e “acidentes_mg” com carga de dados antes da modificação (acidentes_mg3.csv) e após a modificação (acidentes_mg4.csv) quando recebeu os valores de predição. Pela Figura 104, percebe-se que o registro de índice 22 teve seu valor modificado de “Não Informado” para “Lesões Leves”.

Desta feita, foi empregado o método ‘get_dummies()’ para manipulação dos dados, convertendo o dado categórico existente em ‘estado_fisico’ em uma variável dummy ou de indicadores. Esta conversão foi armazenada em ‘vitimas’. Figura 105.

Figura 105: Criando uma variável dummy a partir de ‘estado_fisico’
(autor)

```
#Convertendo 'estado_fisico' numa variável dummy 'vitimas'
vitimas = pd.get_dummies(acidentes_mg['estado_fisico'])
```

```
#Verificando a variável dummy 'vitimas'
vitimas
```

	Iluso	Lesões Graves	Lesões Leves	Óbito
0	0	0	1	0
1	0	0	1	0
2	0	0	1	0
3	0	0	1	0
4	0	0	1	0
...
91784	0	0	1	0

91785	0	0	1	0
91786	0	1	0	0
91787	1	0	0	0
91788	1	0	0	0

91789 rows × 4 columns

No *dataframe* “acidentes_mg”, foi realizada a retirada das colunas (ou features) de ‘ilesos’, ‘feridos_leves’, ‘feridos_graves’ e ‘mortos’, já que seus valores não condizem mais com o valor da predição em ‘estado_físico’. Na sequência, concatenou-se “acidentes_mg” com a variável *dummy* ‘vitimas’. Ver Figura 106.

Figura 106: Ajustes no dataframe “acidentes_mg” com a variável *dummy* (autor)

```
#Eliminando as colunas de 'ilesos', 'feridos_leves', 'feridos_graves', 'mortos' de acidentes_mg
acidentes_mg.drop(labels=['ilesos','feridos_leves','feridos_graves','mortos'],axis=1, inplace = True)

#Unindo acidentes_mg com a variável dummy 'vitimas'
acidentes_mg = pd.concat([acidentes_mg, vitimas], axis=1, join='outer')

#Verificando o dataset novamente modificado de acidentes
acidentes_mg.head()

jica tipo_pista tracado_via uso_solo tipo_veiculo tipo_envolvido estado_físico sexo latitude longitude elemento_transito ileso Lesões Graves Lesões Leves Óbito
Iaró Dupla Reta Rural Automóvel Condutor Lesões Leves Masculino -20.6370 -44.7357 Homem 0 0 1 0
Iaró Dupla Reta Rural Automóvel Passageiro Lesões Leves Masculino -20.6370 -44.7357 Homem 0 0 1 0
Iaró Simples Viaduto Urbano Automóvel Passageiro Lesões Leves Masculino -19.9566 -44.3444 Homem 0 0 1 0
Iaró Simples Viaduto Urbano Automóvel Passageiro Lesões Leves Feminino -19.9566 -44.3444 Homem 0 0 1 0
Iaró Simples Viaduto Urbano Automóvel Condutor Lesões Leves Masculino -19.9566 -44.3444 Homem 0 0 1 0

#Renomeando as colunas de acidentes
acidentes_mg.rename(mapper={'Ileso':'ilesos', 'Lesões Leves':'feridos_leves', 'Lesões Graves':'feridos_graves','Óbito':'mortos'}, axis=1, inplace=True)

#Verificando o dataset novamente modificado de acidentes
acidentes_mg.head()

tracado_via uso_solo tipo_veiculo tipo_envolvido estado_físico sexo latitude longitude elemento_transito ileos feridos_graves feridos_leves mortos
Reta Rural Automóvel Condutor Lesões Leves Masculino -20.6370 -44.7357 Homem 0 0 1 0
Reta Rural Automóvel Passageiro Lesões Leves Masculino -20.6370 -44.7357 Homem 0 0 1 0
Viaduto Urbano Automóvel Passageiro Lesões Leves Masculino -19.9566 -44.3444 Homem 0 0 1 0
Viaduto Urbano Automóvel Passageiro Lesões Leves Feminino -19.9566 -44.3444 Homem 0 0 1 0
Viaduto Urbano Automóvel Condutor Lesões Leves Masculino -19.9566 -44.3444 Homem 0 0 1 0
```

Após a concatenação de “acidentes_mg” com a variável *dummy* ‘vitimas’, realizou-se a alteração dos nomes das colunas (ou features): ‘Ileso’ : ‘ilesos’, ‘Lesões

'Leves' : 'feridos_leves', 'Lesões Graves' : 'feridos_graves', 'Óbito' : 'mortos'; retornando ao estado original de nomes de colunas. Ver Figura 106.

Procedeu-se com nova verificação no *dataframe* “acidentes_mg” com as modificações a partir dos indicadores da variável *dummy* e dos ajustes de estrutura (colunas). Pela Figura 107, é possível conferir os valores de ‘estado_fisico’ e compará-los com os de cada uma das colunas de ‘ilesos’, ‘feridos_leves’, ‘feridos_graves’ e ‘mortos’. Desta forma, o *dataframe* “acidentes_mg” foi plenamente ajustado com os dados da predição e, então, os dados foram salvos em “acidentes_mg5.csv”.

Figura 107: Verificação dos novos ajustes no dataframe “acidentes_mg” (autor)

```
#Verificando o dataset novamente modificado de acidentes
acidentes_mg['estado_fisico'].value_counts()

Ileso        40160
Lesões Leves 37775
Lesões Graves 10382
Óbito         3472
Name: estado_fisico, dtype: int64

#Confirmando os valores alterados de acidentes
print('Total de pessoas envolvidas: ', acidentes_mg['id'].count())
print('Ilesos: ',acidentes_mg.ilesos.sum())
print('Feridos Leves: ',acidentes_mg.feridos_leves.sum())
print('Feridos Graves: ',acidentes_mg.feridos_graves.sum())
print('Mortos: ',acidentes_mg.mortos.sum())

Total de pessoas envolvidas:  91789
Ilesos: 40160
Feridos Leves: 37775
Feridos Graves: 10382
Mortos: 3472

#Salvando as modificações feitas do novo "dataset" acidentes_mg
acidentes_mg.to_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\acidentes_mg5.csv", index = False, header = True, sep=';', decimal=',')


```

Concluído o saneamento do *dataframe* “acidentes_mg”, agora se faz necessário promover os ajustes no outro conjunto de dados, o “datatran_mg”, que possui os acidentes agrupados por ocorrência. Ambos arquivos foram carregados, gerando dois *dataframes* (“acidentes_mg” e “datatran_mg”). A estrutura de “datatran_mg” foi verificada, sendo identificadas colunas que necessitaram serem compatibilizadas com os valores da predição. Figura 108.

Figura 108: Carga de dados com os dataframes “acidentes_mg” e “datatran_mg” (autor)

```
#Lendo os arquivos de acidentes e datatran
datatran_mg = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\datatran_mg3.csv", sep=';', decimal=',', encoding = 'cp1252',
acidentes_mg = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\acidentes_mg5.csv", sep=';', decimal=',', encoding = 'cp1252')

#Confirmando a estrutura de datatran
datatran_mg.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38835 entries, 0 to 38834
Data columns (total 27 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   id               38835 non-null    int64  
 1   data_inversa     38835 non-null    datetime64[ns]
 2   dia_semana      38835 non-null    object  
 3   horario          38835 non-null    object  
 4   uf               38835 non-null    object  
 5   br               38835 non-null    int64  
 6   km               38835 non-null    int64  
 7   municipio        38835 non-null    object  
 8   causa_acidente  38835 non-null    object  
 9   tipo_acidente   38835 non-null    object  
 10  classificacao_acidente 38835 non-null    object  
 11  fase_dia         38835 non-null    object  
 12  condicao_metereologica 38835 non-null    object  
 13  tipo_pista       38835 non-null    object  
 14  tracado_via     38835 non-null    object  
 15  uso_solo         38835 non-null    object  
 16  pessoas          38835 non-null    int64  
 17  mortos           38835 non-null    int64  
 18  feridos_leves   38835 non-null    int64  
 19  feridos_graves  38835 non-null    int64  
 20  ilesos           38835 non-null    int64  
 21  ignorados        38835 non-null    int64  
 22  feridos          38835 non-null    int64  
 23  veiculos          38835 non-null    float64 
 24  latitude          38835 non-null    float64 
 25  longitude         38835 non-null    float64 
 26  elemento_transito 38835 non-null    object  
dtypes: datetime64[ns](1), float64(2), int64(11), object(13)
memory usage: 8.0+ MB

```

```
#Verificando a estrutura e dados de datatran
datatran_mg.head()
```

ca	tipo_pista	tracado_via	uso_solo	pessoas	mortos	feridos_leves	feridos_graves	ilesos	ignorados	feridos	veiculos	latitude	longitude	elemento_transito
ro	Dupla	Reta	Rural	2	0	2	0	0	0	2	1	-20.6370	-44.7357	Homem
ro	Simples	Viaduto	Urbano	3	0	3	0	0	0	3	1	-19.9566	-44.3444	Homem
ro	Simples	Curva	Rural	1	0	0	0	1	0	0	1	-21.8459	-46.4388	Homem
ro	Simples	Curva	Rural	1	0	1	0	0	0	1	1	-19.8942	-43.0403	Homem
ro	Simples	Reta	Rural	3	0	1	0	2	0	1	1	-16.9614	-43.8590	Via

ca	tipo_pista	tracado_via	uso_solo	pessoas	mortos	feridos_leves	feridos_graves	ilesos	ignorados	feridos	veiculos	latitude	longitude	elemento_transito
ro	Dupla	Reta	Rural	2	0	2	0	0	0	2	1	-20.6370	-44.7357	Homem
ro	Simples	Viaduto	Urbano	3	0	3	0	0	0	3	1	-19.9566	-44.3444	Homem
ro	Simples	Curva	Rural	1	0	0	0	1	0	0	1	-21.8459	-46.4388	Homem
ro	Simples	Curva	Rural	1	0	1	0	0	0	1	1	-19.8942	-43.0403	Homem
ro	Simples	Reta	Rural	3	0	1	0	2	0	1	1	-16.9614	-43.8590	Via

Criou-se um *dataframe* auxiliar denominado de 'estado_vitima' para armazenar os valores totalizados de 'ilesos' : 'sum', 'feridos_graves' : 'sum', 'feridos_leves' : 'sum', 'mortos' : 'sum', a partir do *dataframe* "acidentes_mg" (e que já possui os valores de predição e teve a sua estrutura ajustada com os resultados da predição). Pela Figura 109, pode se verificar que foram calculados e armazenados o somatório por coluna de 'ilesos', 'feridos_graves', 'feridos_leves' e 'mortos'.

Figura 109: Criação de dataframe auxiliar “estado_vitima” (autor)

```
#Agregando os valores por estado das vítimas em acidentes
estado_vitimas = pd.DataFrame(acidentes_mg.groupby(acidentes_mg['id']).agg({'ilesos' : 'sum', 'feridos_graves' : 'sum', 'feridos_leves' : 'sum', 'mortos' : 'sum'}))
estado_vitimas.drop(labels=['id'],axis=1, inplace = True)

#Verificando 'estado_vitimas'
estado_vitimas
```

	ilesos	feridos_graves	feridos_leves	mortos
0	0	0	2	0
1	0	0	3	0
2	1	0	0	0
3	0	0	1	0
4	2	0	1	0
...
38830	0	1	0	0
38831	0	0	1	0
38832	0	0	2	0
38833	1	1	0	0
38834	1	0	0	0

38835 rows × 4 columns

Efetuou-se a verificação da quantidade de ‘Ignorados’ em “datatran_mg” e que correspondeu ao que total de ‘Não Informado’ (4837) que existia na feature ‘estado_fisico’ de “acidentes_mg”. Feito isso, realizou-se a eliminação das colunas de ‘ilesos’, ‘feridos_leves’, ‘feridos_graves’, ‘mortos’ e ‘ignorados’ em “datatran_mg” e, na sequência, a concatenação com as colunas do *dataframe* auxiliar ‘estado_vitima’. Ver Figura 110.

Figura 110: Ajustes no dataframe “datatran_mg” – parte 1 (autor)

```
#Verificando a quantidade de 'ignorados' em datatran
datatran_mg['ignorados'].sum()
#valor é igual ao de 'Não Informado' em acidentes
4837

#Unindo datatran_mg com a 'estado_vitimas'
datatran_mg.drop(labels=['ilesos','feridos_leves','feridos_graves','mortos','ignorados'],axis=1, inplace = True)
datatran_mg = pd.concat([datatran_mg, estado_vitimas], axis=1, join='outer')

#Verificando o dataset modificado de datatran
datatran_mg.head()
```

metereologica	tipo_pista	tracado_via	uso_solo	pessoas	feridos	veiculos	latitude	longitude	elemento_transito	ilesos	feridos_graves	feridos_leves	mortos
Céu Claro	Dupla	Reta	Rural	2	2	1	-20.6370	-44.7357	Homem	0	0	2	0
Céu Claro	Simples	Viaduto	Urbano	3	3	1	-19.9566	-44.3444	Homem	0	0	3	0
Céu Claro	Simples	Curva	Rural	1	0	1	-21.8459	-46.4388	Homem	1	0	0	0
Céu Claro	Simples	Curva	Rural	1	1	1	-19.8942	-43.0403	Homem	0	0	1	0
Céu Claro	Simples	Reta	Rural	3	1	1	-16.9614	-43.8590	Via	2	0	1	0

Desta forma, o *dataframe* “*datatran_mg*” passou por um primeiro ajuste de suas colunas e agora, para concluir os ajustes, deve-se compatibilizar os valores de ‘feridos’ e que é o somatório de ‘feridos_graves’ com ‘feridos_leves’, além de ajustar a ‘classificacao_acidente’. Foram obtidos os valores ajustados (ilesos, feridos leves, feridos graves e mortos) e que, comparativamente, são equivalentes aos presentes em “*acidentes_mg*”, conforme Figuras 111 e 107. Por fim, realizou-se o armazenamento dos dados de “*datatran_mg*” em “*datatran_mg4.csv*”.

**Figura 111: Ajustes no dataframe “*datatran_mg*” – parte 2
(autor)**

```
#Corrigindo o valor do atributo de 'feridos' para os novos dados de 'feridos_graves' e 'feridos_leves'
datatran_mg['feridos'] = datatran_mg['feridos_graves'] + datatran_mg['feridos_leves']
```

```
#Confirmando os valores alterados de datatran segundo o modelo que rodou em acidentes
print('Total de pessoas envolvidas: ', datatran_mg['pessoas'].sum())
print('Ilesos: ', datatran_mg.ilesos.sum())
print('Feridos Leves: ', datatran_mg.feridos_leves.sum())
print('Feridos Graves: ', datatran_mg.feridos_graves.sum())
print('Mortos: ', datatran_mg.mortos.sum())
```

```
Total de pessoas envolvidas: 91789
Ilesos: 40160
Feridos Leves: 37775
Feridos Graves: 10382
Mortos: 3472
```

```
#Realizando ajustes na coluna 'classificacao_acidente'
fset3 = datatran_mg[['classificacao_acidente','ilesos','feridos_leves','feridos_graves','mortos']].copy()
fset3.head()
```

	classificacao_acidente	ilesos	feridos_leves	feridos_graves	mortos
0	Com Vítimas Feridas	0	2	0	0
1	Com Vítimas Feridas	0	3	0	0
2	Sem Vítimas	1	0	0	0
3	Com Vítimas Feridas	0	1	0	0
4	Com Vítimas Feridas	2	1	0	0

```
fset3['classificacao_acidente'] = '0'
fset3.loc[(fset3['feridos_leves'] == 0) & (fset3['feridos_graves'] == 0) & (fset3['mortos'] == 0), 'classificacao_acidente'] =
fset3.loc[fset3['mortos'] > 0, 'classificacao_acidente'] = 'Com Vítimas Fatais'
fset3.loc[(fset3['feridos_leves'] > 0) & (fset3['feridos_graves'] > 0) & (fset3['mortos'] == 0), 'classificacao_acidente'] =
fset3.loc[(fset3['feridos_leves'] > 0) & (fset3['mortos'] == 0), 'classificacao_acidente'] = 'Com Vítimas Feridas'
fset3.loc[(fset3['feridos_graves'] > 0) & (fset3['mortos'] == 0), 'classificacao_acidente'] = 'Com Vítimas Feridas'
fset3.drop(labels=['ilesos','feridos_leves','feridos_graves','mortos'],axis=1, inplace = True)
```

```
#Aplicando a correção de fset3 em datatran
datatran_mg.drop(labels=['classificacao_acidente'],axis=1, inplace = True)
datatran_mg = pd.concat([datatran_mg, fset3], axis=1, join='outer')
datatran_mg.head()
```

a	tracado_via	uso_solo	pessoas	feridos	veiculos	latitude	longitude	elemento_transito	ilesos	feridos_graves	feridos_leves	mortos	classificacao_acidente
3	Reta	Rural	2	2	1	-20.6370	-44.7357	Homem	0	0	2	0	Com Vítimas Feridas
3	Viaduto	Urbano	3	3	1	-19.9566	-44.3444	Homem	0	0	3	0	Com Vítimas Feridas
3	Curva	Rural	1	0	1	-21.8459	-46.4388	Homem	1	0	0	0	Sem Vítimas
3	Curva	Rural	1	1	1	-19.8942	-43.0403	Homem	0	0	1	0	Com Vítimas Feridas
3	Reta	Rural	3	1	1	-16.9614	-43.8590	Via	2	0	1	0	Com Vítimas Feridas

```
#Salvando as modificações feitas do novo "dataset" datatran_mg
datatran_mg.to_csv(r'C:\Users\lilia\Downloads\TCC PUC\csv\datatran_mg4.csv", index = False, header = True, sep=';', decimal=',')
```

8. Interpretação dos resultados

A etapa da interpretação dos resultados foi executada em duas fases: (i) via Python, verificando e analisando a amostra de dados ajustados com os valores de predição; e (ii) pelo painel de MS-PowerBI, o qual recebeu carga com os arquivos de dados ajustados com os valores de predição.

Iniciando pela fase do Python, entendeu-se que seria interessante criar um *notebook* separado para ser executado nesta fase (ver Figura 112), o qual contou com carregamento de bibliotecas, configurações de dataframes, de exibição de gráficos e a importação de dados dos arquivos “acidentes_mg” e “datatran_mg” com os valores de predição.

**Figura 112: Notebook Python para interpretação dos resultados
(autor)**

```

PUC Minas - Pontifícia Universidade Católica de Minas Gerais
Pós-Graduação em Ciência de Dados e Big Data
TRABALHO DE CONCLUSÃO DE CURSO

Aluna: Lilian Campos Soares
Matrícula: 1092883

Este notebook é referente aos códigos elaborados em Python e utilizados no Trabalho de Conclusão de Curso em Ciência de Dados

#Carregamento de bibliotecas gerais
import pandas as pd
import numpy as np
import seaborn as sns
import shutil
import os
import io
import datetime
from datetime import datetime
import matplotlib.pyplot as plt

# Definição de configurações de dataframes
pd.set_option('display.max_rows', 1000)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
pd.set_option('display.precision', 4)
pd.set_option('display.expand_frame_repr', False)

#Configurando os plots para serem exibidos diretamente no notebook
%matplotlib inline

#Importação dos dados para interpretação dos resultados - arquivos datatran e acidentes.CSV
acidentes_mg = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\acidentes_mg5.csv", sep=';', decimal=',', encoding = 'cp1252')
datatran_mg = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\datatran_mg4.csv", sep=';', decimal=',', encoding = 'cp1252')

```

Com a exploração dos arquivos alterados, foi possível identificar as no estado físico das vítimas (no total e por ano), segundo Figura 113 e Quadro 7.

Figura 113: Quantidade de acidentes, feridos, mortos e envolvidos (autor)

```
#Nova análise e exploração dos dados
Quantidade_acidentes = datatran_mg.shape[0]
print("Acidentes em MG (2017 a 2020): " + str(Quantidade_acidentes))

Acidentes em MG (2017 a 2020): 38835

datatran_mg['data_inversa'] = datatran_mg.data_inversa.astype('datetime64')
Acidentes_por_ano = datatran_mg.groupby(datatran_mg['data_inversa'].dt.strftime('%Y'))['id'].count().sort_values(ascending=False)
print("Acidentes por ano em MG (2017 a 2020): " + str(Accidentes_por_ano))

Acidentes por ano em MG (2017 a 2020): data_inversa
2017    12711
2018     9055
2019     8713
2020     8356
Name: id, dtype: int64

Total_de_feridos = datatran_mg['feridos'].sum()
print("Feridos em acidentes em MG (2017 a 2020): " + str(Total_de_feridos))
#O total de feridos antes dos valores de predição era de 46100

Feridos em acidentes em MG (2017 a 2020): 48157

Feridos_por_ano = datatran_mg.groupby([datatran_mg['data_inversa'].dt.year.rename('Ano')]).agg({'feridos': 'sum'})
print("Feridos em acidentes por ano em MG (2017 a 2020): " + str(Feridos_por_ano))
#Percebe-se a alteração de totais de feridos por ano

Feridos em acidentes por ano em MG (2017 a 2020):      feridos
Ano
2017    13678
2018    11677
2019    11875
2020    10927

Total_de_mortos = datatran_mg['mortos'].sum()
print("Mortos em acidentes em MG (2017 a 2020): " + str(Total_de_mortos))
#O total de mortos antes dos valores de predição era de 2957

Mortos em acidentes em MG (2017 a 2020): 3472

Mortos_por_ano = datatran_mg.groupby([datatran_mg['data_inversa'].dt.year.rename('Ano')]).agg({'mortos': 'sum'})
print("Mortos em acidentes por ano em MG (2017 a 2020): " + str(Mortos_por_ano))
#Percebe-se a alteração de totais de mortos por ano

Mortos em acidentes por ano em MG (2017 a 2020):      mortos
Ano
2017    1015
2018     822
2019     799
2020     636

Quantidade_pessoas = datatran_mg['pessoas'].sum()
Quantidade_veiculos = datatran_mg['veiculos'].sum()
print("Envolvidos em acidentes em MG (2017 a 2020): \n Pessoas: " + str(Quantidade_pessoas) + "\n " + "Veículos envolvidos: "
#Sem alterações nestes valores

Envolvidos em acidentes em MG (2017 a 2020):
Pessoas: 91789
Veículos envolvidos: 59176
```

**Quadro 7: Comparativo de ‘estado_físico’ das vítimas
(dados da pesquisa)**

Estado físico real	Estado físico com predição
Ileso	37895
Lesões Graves	10022
Lesões Leves	36078
Óbito	2957
Não Informado	4837

	40160
	10382
	37775
	3472

Percebe-se que o valor de ‘4837’ da categoria de ‘Não Informado’ foi preedito entre todos os demais estados físicos, especialmente entre ilesos e óbitos. Isto indica que, das vítimas com os estados de ‘Não Informado’, a probabilidade maior é de que teriam vindo a óbito ou saído ilesos do acidente. Figura 114.

Figura 114: Estado físico das vítimas com os valores de predição (autor)

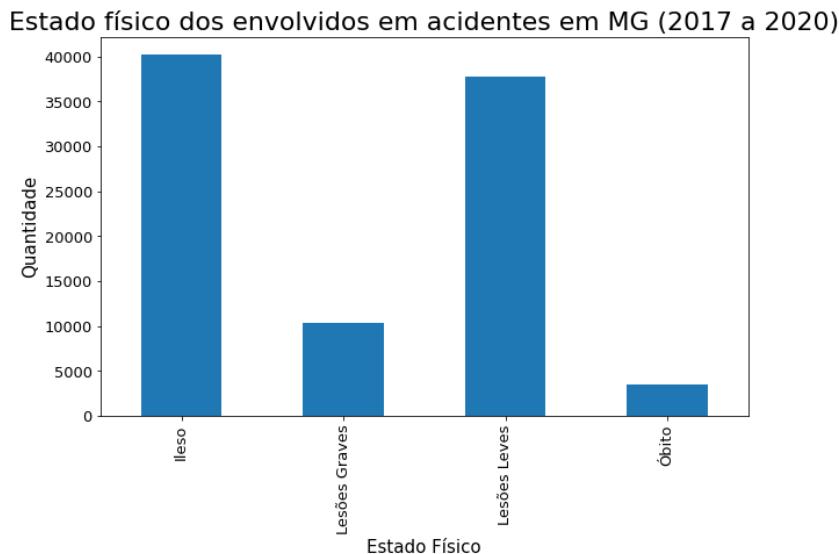
```
Estado_fisico = acidentes_mg.groupby(['estado_fisico']).size()
print("Estado físico dos envolvidos em MG (2017 a 2020): " + str(Estado_fisico))
#Os totais por categoria sofreram alterações

Estado físico dos envolvidos em MG (2017 a 2020): estado_fisico
Ileso          40160
Lesões Graves 10382
Lesões Leves   37775
Óbito           3472
dtype: int64

Quantidade_envolvidos = acidentes_mg.shape[0]
perc_ilesos = (acidentes_mg['id'][acidentes_mg['estado_fisico'] == 'Ileso'].count() / Quantidade_envolvidos) *100
perc_leogr = (acidentes_mg['id'][acidentes_mg['estado_fisico'] == 'Lesões Graves'].count() / Quantidade_envolvidos) *100
perc_lelev = (acidentes_mg['id'][acidentes_mg['estado_fisico'] == 'Lesões Leves'].count() / Quantidade_envolvidos) *100
perc_nainf = (acidentes_mg['id'][acidentes_mg['estado_fisico'] == 'Não Informado'].count() / Quantidade_envolvidos) *100
perc_obito = (acidentes_mg['id'][acidentes_mg['estado_fisico'] == 'Óbito'].count() / Quantidade_envolvidos) *100
#print(f"% Estado físico dos envolvidos em MG (2017 a 2020): \n Ileso-{perc_ilesos},\n Lesões Graves-{perc_leogr},\n Lesões Leves-{perc_lelev},\n Não Informado-{perc_nainf},\n Óbito-{perc_obito}")
print("% Estado físico dos envolvidos em MG (2017 a 2020): \n Ileso " + "%.2f" % perc_ilesos + "\n Lesões Graves " + "%.2f" % perc_leogr + "\n Lesões Leves " + "%.2f" % perc_lelev + "\n Não Informado " + "%.2f" % perc_nainf + "\n Óbito " + "%.2f" % perc_obito)
#Agora não existe mais a categoria de Não Informado

% Estado físico dos envolvidos em MG (2017 a 2020):
Ileso 43.75
Lesões Graves 11.31
Lesões Leves 41.15
Não Informado 0.00
Óbito 3.78

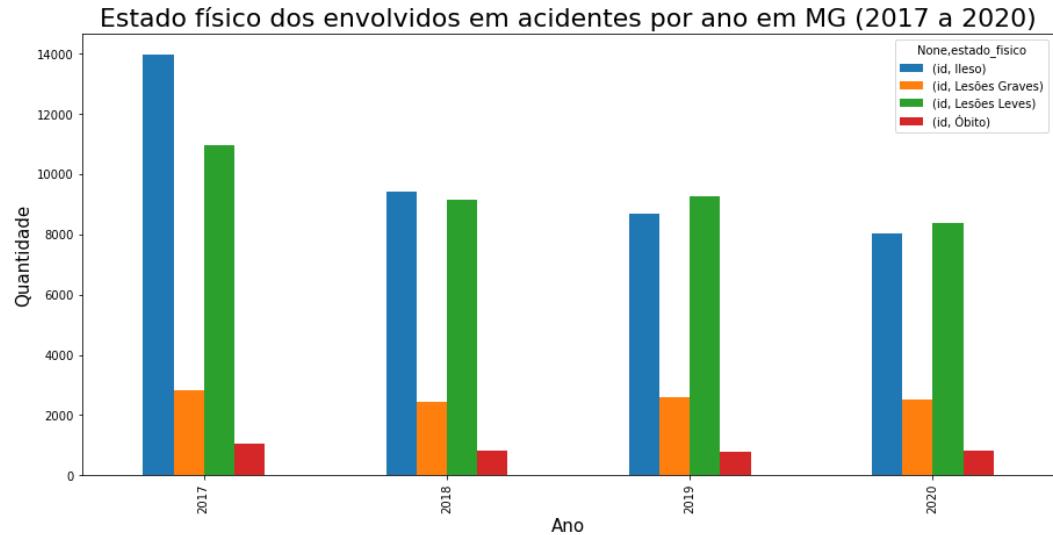
ax = Estado_fisico.plot(kind='bar', figsize=(10,6), fontsize=13);
ax.set_alpha(0.8)
ax.set_title("Estado físico dos envolvidos em acidentes em MG (2017 a 2020)", fontsize=22)
ax.set_ylabel("Quantidade", fontsize=15);
ax.set_xlabel("Estado Físico", fontsize=15);
plt.show()
```



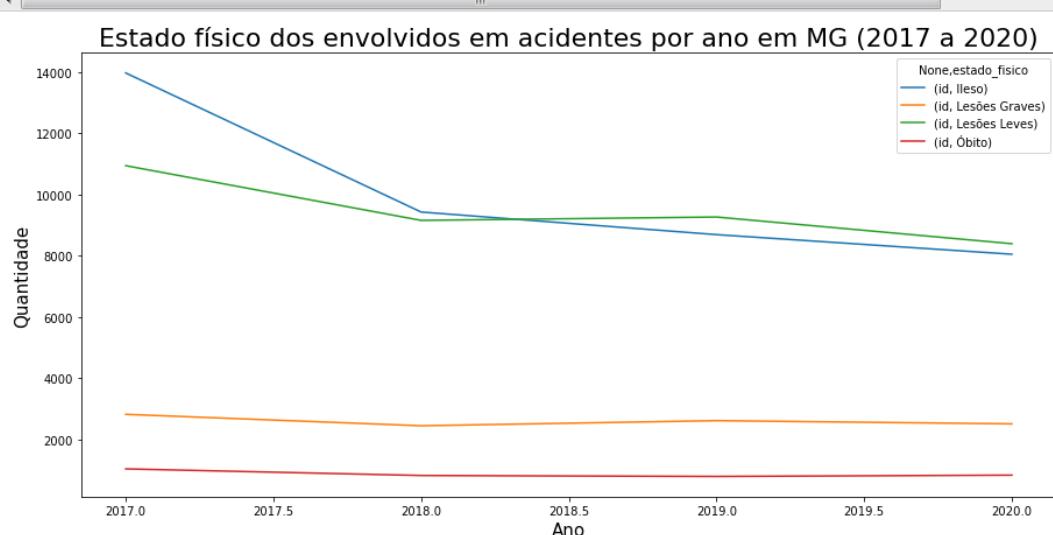
```
acidentes_mg['data_inversa'] = acidentes_mg.data_inversa.astype('datetime64')
Estado_fisico_por_ano = acidentes_mg.groupby([acidentes_mg['data_inversa'].dt.year.rename('Ano'), acidentes_mg['estado_fisico']])
print("Estado físico dos envolvidos por ano em MG (2017 a 2020): " + str(Estado_fisico_por_ano))

Estado físico dos envolvidos por ano em MG (2017 a 2020):
Ano  estado_fisico
2017 Ileso          13980
      Lesões Graves  2818
      Lesões Leves   10948
      Óbito            1035
2018 Ileso          9433
      Lesões Graves  2445
      Lesões Leves   9160
      Óbito            819
2019 Ileso          8694
      Lesões Graves  2612
      Lesões Leves   9270
      Óbito            788
2020 Ileso          8053
      Lesões Graves  2507
      Lesões Leves   8397
      Óbito            830
```

```
fig, ax = plt.subplots(figsize=(15,7))
acidentes_mg.groupby([acidentes_mg['data_inversa'].dt.year.rename('Ano'), acidentes_mg['estado_fisico']]).agg({'id': 'count'})
ax.set_alpha(0.8)
ax.set_title("Estado físico dos envolvidos em acidentes por ano em MG (2017 a 2020)", fontsize=22)
ax.set_ylabel("Quantidade", fontsize=15);
ax.set_xlabel("Ano", fontsize=15);
plt.show()
```



```
fig, ax = plt.subplots(figsize=(15,7))
acidentes_mg.groupby([acidentes_mg['data_inversa'].dt.year.rename('Ano'), acidentes_mg['estado_fisico']]).agg({'id': 'count'})
ax.set_alpha(0.8)
ax.set_title("Estado físico dos envolvidos em acidentes por ano em MG (2017 a 2020)", fontsize=22)
ax.set_ylabel("Quantidade", fontsize=15);
ax.set_xlabel("Ano", fontsize=15);
plt.show()
```



Quanto à variável categórica do dataset ‘classificacao_acidente’, embora as quantidades totais de classificação de acidentes não tenham sofrido mudanças por categoria (Sem Vítimas, Com Vítimas Feridas, Com Vítimas Fatais) – Figura 115, ocorreram mudanças nos valores destas categorias por BR. Isto pode ser verificado comparando com o resultado da nova geração dos índices de acidentes, segundo a Figura 116 e Quadro 08.

Figura 115: Classificação dos acidentes com os valores de predição (autor)

```
Classificacao_de_acidentes = datatran_mg.groupby(['classificacao_acidente']).size()
print("Classificação de acidentes em MG (2017 a 2020): " + str(Classificacao_de_acidentes))
#A classificação de acidentes não sofreu mudanças em quantidades totais

Classificação de acidentes em MG (2017 a 2020): classificacao_acidente
Com Vítimas Fatais      2375
Com Vítimas Feridas    27966
Sem Vítimas            8494
dtype: int64
```

Figura 116: Classificação dos acidentes com os valores de predição (autor)

```
#Ajustes nos indicadores relacionados aos acidentes de MG

#Classificação por gravidade dos acidentes
#print(str(datatran_mg.groupby(['br','classificacao_acidente']).size()))
class_acid = pd.DataFrame(datatran_mg.groupby([datatran_mg['data_inversa']].dt.year.rename('Ano'), datatran_mg['br'], datatran_mg['classificacao_acidente']).size())
class_acid.rename(columns={'br':'BR'}, inplace=True)
class_acid.rename(columns={'id':'Acidentes'}, inplace=True)
class_acid['Sem_Vitimas'] = '0'
class_acid['Com_Vitimas_Feridas'] = '0'
class_acid['Com_Vitimas_Fatais'] = '0'
class_acid['BR'] = class_acid.BR.astype('object')
class_acid['Sem_Vitimas'] = class_acid.Sem_Vitimas.astype('int64')
class_acid['Com_Vitimas_Feridas'] = class_acid.Com_Vitimas_Feridas.astype('int64')
class_acid['Com_Vitimas_Fatais'] = class_acid.Com_Vitimas_Fatais.astype('int64')
class_acid.loc[class_acid['classificacao_acidente'] == 'Com Vítimas Fatais', 'Com_Vitimas_Fatais'] = class_acid['Acidentes']
class_acid.loc[class_acid['classificacao_acidente'] == 'Com Vítimas Feridas', 'Com_Vitimas_Feridas'] = class_acid['Acidentes']
class_acid.loc[class_acid['classificacao_acidente'] == 'Sem Vítimas', 'Sem_Vitimas'] = class_acid['Acidentes']
class_acid.drop(columns=['classificacao_acidente'], axis=1, inplace=True)
class_acid.drop(columns=['Acidentes'], axis=1, inplace=True)
class_acidentes = pd.DataFrame(class_acid.groupby([class_acid['Ano'], class_acid['BR']]).agg({'Sem_Vitimas' : 'sum', 'Com_Vitimas_Feridas' : 'sum', 'Com_Vitimas_Fatais' : 'sum'}))
#class_acidentes.head(10)

#Extensões por trecho de BR
snv_2017_mg = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\SNV_201801B_3.csv", sep=';', decimal=',', encoding = 'utf_8')
snv_2018_mg = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\snv_201903a_3.csv", sep=';', decimal=',', encoding = 'utf_8')
snv_2019_mg = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\SNV_202001A_3.csv", sep=';', decimal=',', encoding = 'utf_8')
snv_2020_mg = pd.read_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\SNV_202101A_3.csv", sep=';', decimal=',', encoding = 'utf_8')
snv_2017_mg.drop(columns=['Tipo de trecho','Desc Coint','Código','Local de Início','Local de Fim','km inicial','km final'], axis=1, inplace=True)
snv_2018_mg.drop(columns=['Tipo de trecho','Desc Coint','Código','Local de Início','Local de Fim','km inicial','km final'], axis=1, inplace=True)
snv_2019_mg.drop(columns=['Tipo de trecho','Desc Coint','Código','Local de Início','Local de Fim','km inicial','km final','Superação'], axis=1, inplace=True)
snv_2020_mg.drop(columns=['Tipo de trecho','Desc Coint','Código','Local de Início','Local de Fim','km inicial','km final','Superação'], axis=1, inplace=True)

#Extensões por BR
snv_2017_mg_br = pd.DataFrame(snv_2017_mg.groupby('BR', as_index = False)[['Extensão']].sum()).reset_index()
snv_2018_mg_br = pd.DataFrame(snv_2018_mg.groupby('BR', as_index = False)[['Extensão']].sum()).reset_index()
snv_2019_mg_br = pd.DataFrame(snv_2019_mg.groupby('BR', as_index = False)[['Extensão']].sum()).reset_index()
snv_2020_mg_br = pd.DataFrame(snv_2020_mg.groupby('BR', as_index = False)[['Extensão']].sum()).reset_index()
snv_2017_mg_br.rename(columns={'index':'Ano'}, inplace=True)
snv_2017_mg_br['Ano'] = '2017'
snv_2018_mg_br.rename(columns={'index':'Ano'}, inplace=True)
snv_2018_mg_br['Ano'] = '2018'
snv_2019_mg_br.rename(columns={'index':'Ano'}, inplace=True)
snv_2019_mg_br['Ano'] = '2019'
snv_2020_mg_br.rename(columns={'index':'Ano'}, inplace=True)
snv_2020_mg_br['Ano'] = '2020'

#Excluindo BRs sem acidentes registrados
snv_mg_br = pd.concat([snv_2017_mg_br,snv_2018_mg_br,snv_2019_mg_br,snv_2020_mg_br], join="inner")
snv_mg_br.set_index('Ano', inplace = True)
snv_mg_br.reset_index(inplace=True)
snv_mg_br.drop(snv_mg_br.index[[0,4,5,9,11,13,15,16,21,22,24,25,26,27,28,29,31,32,34,35,36,37,38,39,40,41,42,43,44,45,46]], inplace=True)
snv_mg_br.set_index('Ano', inplace = True)
snv_mg_br.reset_index(inplace=True)
snv_mg_br.drop(snv_mg_br.index[[19,20,24,26,28,30,31,36,37,39,40,41,42,43,45,46,47,49,50,51,52,53,54,55,56,57,58,59,60,61]], inplace=True)
snv_mg_br.set_index('Ano', inplace = True)
snv_mg_br.reset_index(inplace=True)
snv_mg_br.drop(snv_mg_br.index[[37,40,42,46,47,52,53,55,56,57,58,59,60,62,63,65,66,67,68,69,70,71,72,73,74,75,76,77]], inplace=True)
snv_mg_br.set_index('Ano', inplace = True)
snv_mg_br.reset_index(inplace=True)
snv_mg_br.drop(snv_mg_br.index[[53,54,55,58,60,64,70,71,73,74,75,76,77,78,79,80,81,83,84,85,86,87,88,89,90,91,92,93,94,95]], inplace=True)
snv_mg_br.set_index('Ano', inplace = True)
snv_mg_br.reset_index(inplace=True)
#print(snv_mg_br)
snv_mg_br.to_csv(r"C:\Users\lilia\Downloads\TCC PUC\csv\snv_mg_br2.csv", index=False)

#Preparação para a geração dos indicadores
temp_mg = pd.DataFrame(datatran_mg.groupby([datatran_mg['data_inversa']].dt.year.rename('Ano'), datatran_mg['br']).agg({'id': 'count', 'Acidentes': 'sum', 'Feridos': 'sum', 'Mortes': 'sum', 'Envolvidos': 'sum'}))
temp_mg.rename(columns={'id':'Acidentes'}, inplace=True)
temp_mg.rename(columns={'feridos':'Feridos'}, inplace=True)
temp_mg.rename(columns={'mortos':'Mortes'}, inplace=True)
temp_mg.rename(columns={'pessoas':'Envolvidos'}, inplace=True)
temp_mg.rename(columns={'br':'BR'}, inplace=True)
temp_mg['BR'] = temp_mg.BR.astype('object')
#print(temp_mg)
```

```

snv_mg_br.drop(columns=['BR'], axis=1, inplace=True)
temp_mg.drop(columns=['Ano'], axis=1, inplace=True)
temp_indices = pd.DataFrame(pd.concat([snv_mg_br, temp_mg], axis=1, join="inner")).reset_index()
temp_indices.drop(columns=['index'], axis=1, inplace=True)
#print(temp_indices)
#temp_indices.info()

#Gerando os indicadores - dados de 2018
temp_2018 = temp_indices.loc[(temp_indices['Ano'] == "2018")]
#print(temp_2018.loc[temp_2018['Ano'] == "2018", 'Extensão'].sum())
temp_2018['%_Extensão_BR'] = (temp_2018['Extensão'] / temp_2018['Extensão'].sum())*100
temp_2018['%_Acidente_BR'] = (temp_2018['Acidentes'] / temp_2018['Acidentes'].sum())*100
temp_2018['%_Feridos_BR'] = (temp_2018['Feridos'] / temp_2018['Feridos'].sum())*100
temp_2018['%_Mortes_BR'] = (temp_2018['Mortes'] / temp_2018['Mortes'].sum())*100
temp_2018['Acidente_KM'] = (temp_2018['Acidentes'] / temp_2018['Extensão'])
temp_2018['Mortes_KM'] = (temp_2018['Mortes'] / temp_2018['Extensão'])
temp_2018 = temp_2018.reindex(columns=['Ano', 'BR', 'Extensão', 'Acidentes', 'Envolvidos', 'Feridos', 'Mortes', '%_Extensão_BR'])
temp_2018.drop(columns=['index'], axis=1, inplace=True)
#print(temp_2018)
#temp_2018.head(50)

#Gerando os indicadores - dados de 2019
temp_2019 = temp_indices.loc[(temp_indices['Ano'] == "2019")]
#print(temp_2019.loc[temp_2019['Ano'] == "2019", 'Extensão'].sum())
temp_2019['%_Extensão_BR'] = (temp_2019['Extensão'] / temp_2019['Extensão'].sum())*100
temp_2019['%_Acidente_BR'] = (temp_2019['Acidentes'] / temp_2019['Acidentes'].sum())*100
temp_2019['%_Feridos_BR'] = (temp_2019['Feridos'] / temp_2019['Feridos'].sum())*100
temp_2019['%_Mortes_BR'] = (temp_2019['Mortes'] / temp_2019['Mortes'].sum())*100
temp_2019['Acidente_KM'] = (temp_2019['Acidentes'] / temp_2019['Extensão'])
temp_2019['Mortes_KM'] = (temp_2019['Mortes'] / temp_2019['Extensão'])
temp_2019 = temp_2019.reindex(columns=['Ano', 'BR', 'Extensão', 'Acidentes', 'Envolvidos', 'Feridos', 'Mortes', '%_Extensão_BR'])
temp_2019.drop(columns=['index'], axis=1, inplace=True)
#print(temp_2019)
#temp_2019.head(50)

#Gerando os indicadores - dados de 2020
temp_2020 = temp_indices.loc[(temp_indices['Ano'] == "2020")]
#print(temp_2020.loc[temp_2020['Ano'] == "2020", 'Extensão'].sum())
temp_2020['%_Extensão_BR'] = (temp_2020['Extensão'] / temp_2020['Extensão'].sum())*100
temp_2020['%_Acidente_BR'] = (temp_2020['Acidentes'] / temp_2020['Acidentes'].sum())*100
temp_2020['%_Feridos_BR'] = (temp_2020['Feridos'] / temp_2020['Feridos'].sum())*100
temp_2020['%_Mortes_BR'] = (temp_2020['Mortes'] / temp_2020['Mortes'].sum())*100
temp_2020['Acidente_KM'] = (temp_2020['Acidentes'] / temp_2020['Extensão'])
temp_2020['Mortes_KM'] = (temp_2020['Mortes'] / temp_2020['Extensão'])
temp_2020 = temp_2020.reindex(columns=['Ano', 'BR', 'Extensão', 'Acidentes', 'Envolvidos', 'Feridos', 'Mortes', '%_Extensão_BR'])
temp_2020.drop(columns=['index'], axis=1, inplace=True)
#print(temp_2020)
#temp_2020.head(50)

#Consolidando os indicadores de accidentalidade em um único dataframe
indices_temp = pd.concat([temp_2017, temp_2018, temp_2019, temp_2020], sort = False).reset_index()
indices_temp.drop(columns=['index'], axis=1, inplace=True)
class_acidentes.drop(columns=['Ano'], axis=1, inplace=True)
class_acidentes.drop(columns=['BR'], axis=1, inplace=True)
indices_acidentes = pd.concat([indices_temp, class_acidentes], axis=1, sort = False).reset_index()
indices_acidentes.drop(columns=['index'], axis=1, inplace=True)
indices_acidentes['Gravidade'] = (indices_acidentes['Sem_Vítimas'] + (5 * indices_acidentes['Com_Vítimas_Feridas'])+(13 * indices_acidentes['Mortes']))
indices_acidentes.to_csv(r'C:\Users\lilia\Downloads\TCC_FUC\indices_acidentes_ajustado.csv", index = False, header = True,

```

```

#Relação dos indicadores de accidentalidade ajustados
indices_acidentes.head(66)
#Embora as quantidades totais de classificação de acidentes não tenham sofrido mudanças por categoria (Sem Vítimas, Com Vítimas, Mortes), ocorreram mudanças nos valores destas categorias por BR
#Isto poderá ser verificado comparando os arquivos de índices de acidentes

```

Ano	BR	Extensão	Acidentes	Envolvidos	Feridos	Mortes	%_Extensão_BR	%_Acidente_BR	Acidente_KM	%_Feridos_BR	%_Mortes_BR	Mortes_KM	Sem_Vítimas
0	2017	40	893.90	2374	5479	2600	182	8.7231	18.6767	2.6558	19.0086	17.9310	0.2036
1	2017	50	269.80	673	1333	638	21	2.6328	5.2946	2.4944	4.6644	2.0690	0.0778
2	2017	116	818.10	1626	3749	1935	204	7.9834	12.7921	1.9875	14.1468	20.0985	0.2494
3	2017	135	834.70	208	553	263	39	8.1454	1.6364	0.2492	1.9228	3.8424	0.0467
4	2017	146	726.80	114	256	137	8	7.0925	0.8969	0.1569	1.0016	0.7882	0.0110
5	2017	153	239.90	379	855	377	31	2.3411	2.9817	1.5798	2.7563	3.0542	0.1292
6	2017	251	1015.70	282	736	333	50	9.9117	2.2186	0.2776	2.4346	4.9261	0.0492
7	2017	262	1082.90	1381	3012	1547	102	10.5675	10.8646	1.2753	11.3101	10.0493	0.0942
8	2017	267	534.70	293	759	425	35	5.2179	2.3051	0.5480	3.1072	3.4483	0.0655
9	2017	354	774.50	106	241	96	4	7.5579	0.8339	0.1369	0.7019	0.3941	0.0052

Pode-se citar, como exemplo, a BR-040 que no ano de 2017 apresentou inicialmente a composição de 720 acidentes sem vítimas, 1544 com vítimas feridas

em 110 com vítimas fatais (ver Quadro 6), e, após a predição, passou a apresentar 723 acidentes sem vítimas, 1542 com vítimas feridas em 109 com vítimas fatais (ver Quadro 8).

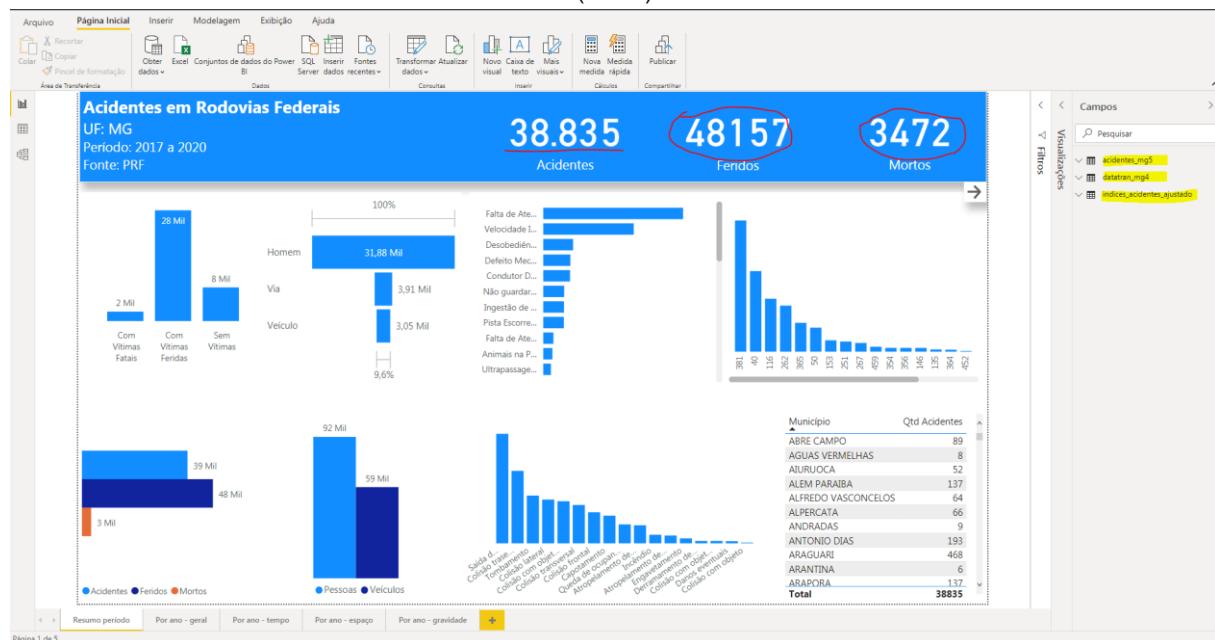
**Quadro 8: Índices de Acidentes, por BR, por ano, com valores da predição
(dados da pesquisa)**

Ano	BR	Ex- ten- são	Aci- den- tes	En- vol- vi- dos	Fe- ri- dos	Mo- rt- es	%_Ext- en- são_B- R	%_Aci- den- te_BR	Aci- den- te_K- M	%_Fer- idos_- BR	%_Mo- rtes_- BR	Mor- tes _KM	Sem- Vi- ti- mas	Com_Vi- ti- mas_Fe- ridas	Com_Vi- ti- mas_F- atais	gra- vi- dade
2017	40	893, 9	237 4	5479	260 0	18 2	8,72	18,68	2,66	19,01	17,9 3	0,2 0	723	1542	109	11,0 2
2017	50	269, 8	673	1333	638	21	2,63	5,29	2,49	4,66	2,07	0,0 8	236	423	14	9,39
2017	116	818, 1	162 6	3749	193 5	20 4	7,98	12,79	1,99	14,15	20,1 0	0,2 5	393	1105	128	9,27
2017	135	834, 7	208	553	263	39	8,15	1,64	0,25	1,92	3,84	0,0 5	59	124	25	1,20
2017	146	726, 8	114	256	137	8	7,09	0,90	0,16	1,00	0,79	0,0 1	36	72	6	0,65
2017	153	239, 9	379	855	377	31	2,34	2,98	1,58	2,76	3,05	0,1 3	144	211	24	6,30
2017	251	101, 5,7	282	736	333	50	9,91	2,22	0,28	2,43	4,93	0,0 5	87	162	33	1,31
2017	262	108, 2,9	138 1	3012	154 7	10 2	10,57	10,86	1,28	11,31	10,0 5	0,0 9	407	897	77	5,44
2017	267	534, 7	293	759	425	35	5,22	2,31	0,55	3,11	3,45	0,0 7	64	205	24	2,62
2017	354	774, 5	106	241	96	4	7,56	0,83	0,14	0,70	0,39	0,0 1	36	66	4	0,54
2017	356	291, 9	108	230	124	6	2,85	0,85	0,37	0,91	0,59	0,0 2	21	84	3	1,64
2017	364	287	69	146	80	2	2,80	0,54	0,24	0,58	0,20	0,0 1	20	47	2	0,98
2017	365	869, 9	756	1804	909	69	8,49	5,95	0,87	6,65	6,80	0,0 8	197	516	43	3,83
2017	381	108, 3,9	418 0	9210	403 3	25 2	10,58	32,88	3,86	29,49	24,8 3	0,2 3	1685	2331	164	14,2 7
2017	452	306, 2	17	33	21	0	2,99	0,13	0,06	0,15	0,00	0,0 0	5	12	0	0,21
2017	459	217, 6	145	385	160	10	2,12	1,14	0,67	1,17	0,99	0,0 5	48	90	7	2,71
2018	40	894, 4	177 2	4510	235 6	19 8	8,71	19,57	1,98	20,18	24,0 9	0,2 2	369	1284	119	9,32
2018	50	270, 2	531	1179	730	39	2,63	5,86	1,97	6,25	4,74	0,1 4	99	406	26	9,13
2018	116	818, 1	113 2	2627	142 0	11 1	7,96	12,50	1,38	12,16	13,5 0	0,1 4	224	839	69	6,50
2018	135	834, 7	1	3	1	0	8,13	0,01	0,00	0,01	0,00	0,0 0	0	1	0	0,01
2018	146	632, 1	77	215	90	6	6,15	0,85	0,12	0,77	0,73	0,0 1	19	54	4	0,54
2018	153	239, 9	240	542	320	12	2,34	2,65	1,00	2,74	1,46	0,0 5	41	188	11	4,69
2018	251	101, 5,7	211	599	271	11	9,89	2,33	0,21	2,32	1,34	0,0 1	39	162	10	0,96
2018	262	108, 2,9	101 3	2369	130 4	78	10,54	11,19	0,94	11,17	9,49	0,0 7	180	772	61	4,46
2018	267	535, 3	192	461	249	12	5,21	2,12	0,36	2,13	1,46	0,0 2	52	131	9	1,54
2018	354	775, 4	87	260	101	8	7,55	0,96	0,11	0,86	0,97	0,0 1	14	67	6	0,55
2018	356	288, 3	82	183	113	4	2,81	0,91	0,28	0,97	0,49	0,0 1	13	66	3	1,33
2018	364	287	52	109	76	6	2,79	0,57	0,18	0,65	0,73	0,0 2	11	39	2	0,81
2018	365	869, 9	609	1548	721	51	8,47	6,73	0,70	6,17	6,20	0,0 6	131	443	35	3,22
2018	381	108, 4,2	294 4	6981	377 6	27 6	10,55	32,51	2,72	32,34	33,5 8	0,2 5	594	2155	195	12,8 2

2018	451	429 ,1	3	5	6	2	4,18	0,03	0,01	0,05	0,24	0,0 0	0	2	1	0,05
2018	459	215 ,4	109	266	143	8	2,10	1,20	0,51	1,22	0,97	0,0 4	21	82	6	2,36
2019	40	894 ,5	173	4265	242 5	15 9	8,08	19,91	1,94	20,42	19,9 0	0,1 8	233	1393	109	9,63
2019	50	270 ,2	506	1067	703	37	2,44	5,81	1,87	5,92	4,63	0,1 4	72	408	26	9,07
2019	116	818 ,1	113	2773	151 2	11 1	7,39	13,03	1,39	12,73	13,8 9	0,1 4	176	881	78	6,84
2019	120	808 ,1	1	5	1	0	7,30	0,01	0,00	0,01	0,00	0,0 0	0	1	0	0,01
2019	122	280 ,9	1	3	1	0	2,54	0,01	0,00	0,01	0,00	0,0 0	0	1	0	0,02
2019	146	632 ,1	89	240	123	4	5,71	1,02	0,14	1,04	0,50	0,0 1	20	66	3	0,62
2019	153	239 ,9	230	525	283	17	2,17	2,64	0,96	2,38	2,13	0,0 7	36	181	13	4,63
2019	251	101 5,7	201	480	253	32	9,18	2,31	0,20	2,13	4,01	0,0 3	23	158	20	1,06
2019	262	108 3,9	956	2254	128 8	90	9,80	10,97	0,88	10,85	11,2 6	0,0 8	129	762	65	4,41
2019	265	665	2	2	4	0	6,01	0,02	0,00	0,03	0,00	0,0 0	0	2	0	0,02
2019	267	535 ,3	182	416	255	11	4,84	2,09	0,34	2,15	1,38	0,0 2	22	150	10	1,69
2019	354	775 ,4	86	192	106	8	7,01	0,99	0,11	0,89	1,00	0,0 1	16	62	8	0,55
2019	356	288 ,3	92	202	135	11	2,61	1,06	0,32	1,14	1,38	0,0 4	9	74	9	1,72
2019	364	287 ,7	34	74	37	3	2,60	0,39	0,12	0,31	0,38	0,0 1	10	23	1	0,48
2019	365	869 ,9	702	1963	989	92	7,86	8,06	0,81	8,33	11,5 1	0,1 1	101	540	61	4,13
2019	381	107 8,7	265 0	6630	358 9	20 9	9,75	30,41	2,46	30,22	26,1 6	0,1 9	418	2070	162	11,9 3
2019	452	306 ,2	2	3	1	1	2,77	0,02	0,01	0,01	0,13	0,0 0	0	1	1	0,06
2019	459	215 ,4	109	270	170	14	1,95	1,25	0,51	1,43	1,75	0,0 6	12	89	8	2,60
2020	40	894 ,5	166 8	4042	214 5	13 7	8,84	19,96	1,86	19,63	16,3 9	0,1 5	250	1315	103	9,13
2020	50	270 ,4	538	1070	599	30	2,67	6,44	1,99	5,48	3,59	0,1 1	107	405	26	9,13
2020	116	818 ,73	106 7	2418	143 7	12 5	8,09	12,77	1,30	13,15	14,9 5	0,1 5	111	865	91	6,86
2020	146	603	71	149	85	12	5,96	0,85	0,12	0,78	1,44	0,0 2	13	50	8	0,61
2020	153	239 ,9	215	483	238	36	2,37	2,57	0,90	2,18	4,31	0,1 5	45	147	23	4,50
2020	251	101 5,9	255	891	404	30	10,04	3,05	0,25	3,70	3,59	0,0 3	44	187	24	1,27
2020	262	108 3,9	965	2116	124 1	84	10,71	11,55	0,89	11,36	10,0 5	0,0 8	156	756	53	4,27
2020	265	658 ,3	1	1	1	0	6,51	0,01	0,00	0,01	0,00	0,0 0	0	1	0	0,01
2020	267	534	148	310	187	42	5,28	1,77	0,28	1,71	5,02	0,0 8	18	104	26	1,64
2020	352	485 ,2	1	6	1	0	4,80	0,01	0,00	0,01	0,00	0,0 0	0	1	0	0,01
2020	354	775 ,4	77	179	110	7	7,66	0,92	0,10	1,01	0,84	0,0 1	8	63	6	0,52
2020	356	287 ,96	71	140	87	2	2,85	0,85	0,25	0,80	0,24	0,0 1	10	59	2	1,15
2020	364	287 ,7	28	72	46	0	2,84	0,34	0,10	0,42	0,00	0,0 0	4	24	0	0,43
2020	365	868 ,2	635	1649	892	10 7	8,58	7,60	0,73	8,16	12,8 0	0,1 2	106	471	58	3,70
2020	381	107 9	251 7	6042	331 9	21 2	10,66	30,12	2,33	30,37	25,3 6	0,2 0	363	2000	154	11,4 6
2020	459	215 ,4	99	219	135	12	2,13	1,18	0,46	1,24	1,44	0,0 6	14	78	7	2,30

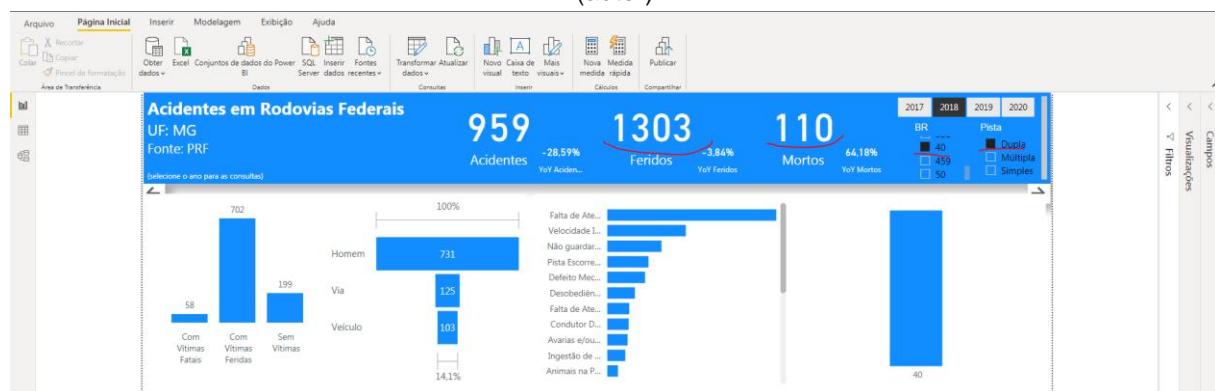
Passando para a apresentação dos dados com valores de predição no painel de MS-PowerBI, foi realizada nova carga de dados no painel, agora com os arquivos e valores ajustados com a predição, conforme Figura 117. Percebe-se alteração nos totais de feridos e mortos, assim como obtido na predição via Python (Figura 114).

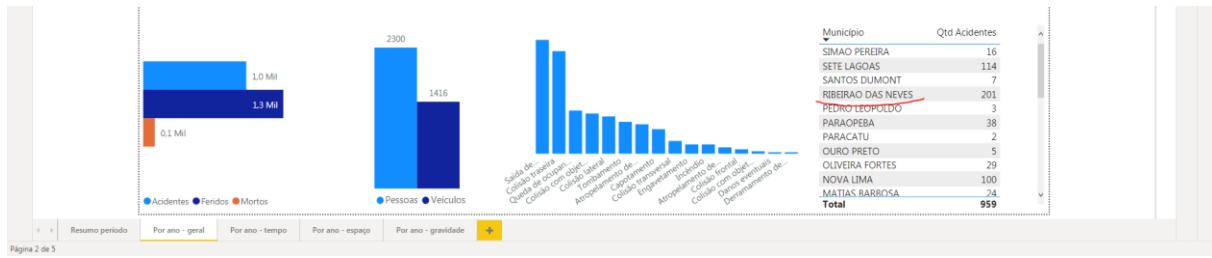
Figura 117: Dashboard “Resumo período” do MS-PowerBI, com valores da predição (autor)



Com o dashboard “Por ano - geral” (Figura 118), selecionou-se o ano de 2018, a BR-040, pista do tipo dupla, e confirmou-se que o município Ribeirão das Neves ainda é o que apresentou maior ocorrência de acidentes. Contudo, verificou-se mudança na quantidade de feridos (de 1124 para 1303) e de mortos (de 46 para 110), confrontando com a Figura 58.

Figura 118: Dashboard “Por ano- geral” do MS-PowerBI com seleção de ano, BR e tipo de pista, com valores da predição (autor)

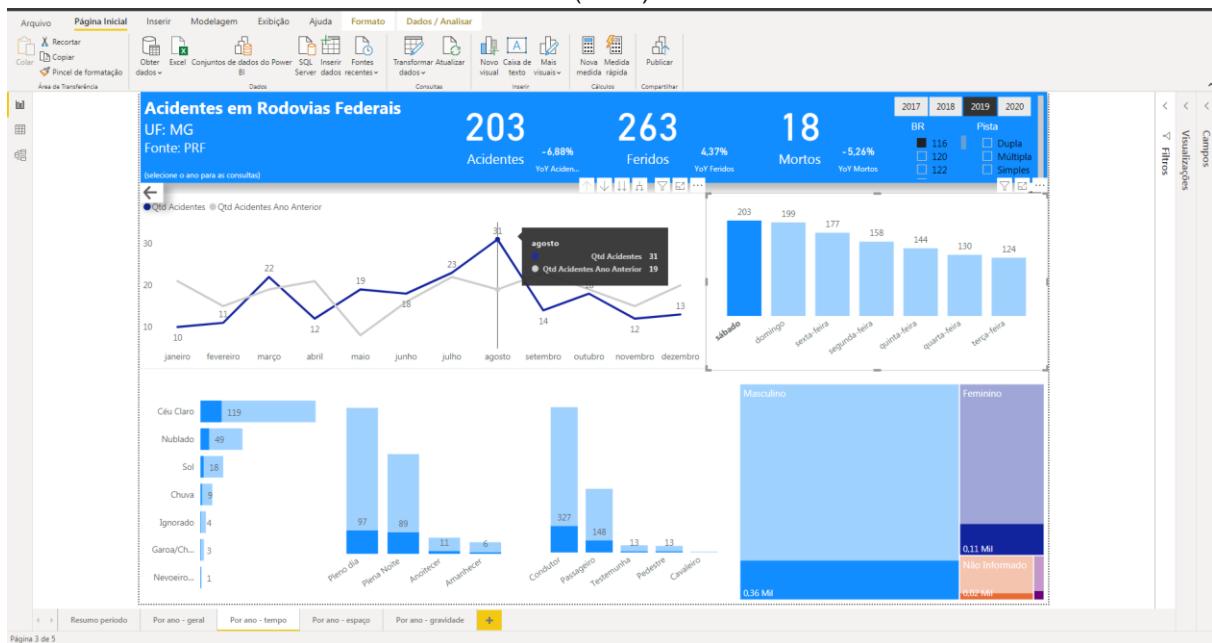




Com o dashboard “Por ano - tempo” (Figura 119), selecionando filtro do ano para 2019 e a BR-116, verificou-se que o sábado continuou como o dia da semana que apresentou a maior quantidade de acidentes na seleção.

Na sequência, inserindo o sábado no filtro da seleção, percebeu-se que o mês de agosto também foi o que apresentou maior quantidade de acidentes no dia de sábado, contudo verificou-se mudança no comportamento nos demais meses do ano (Figura 61). Também, a quantidade total de feridos diminuiu de 290 para 263 assim como de mortos que também diminuiu de 28 para 18.

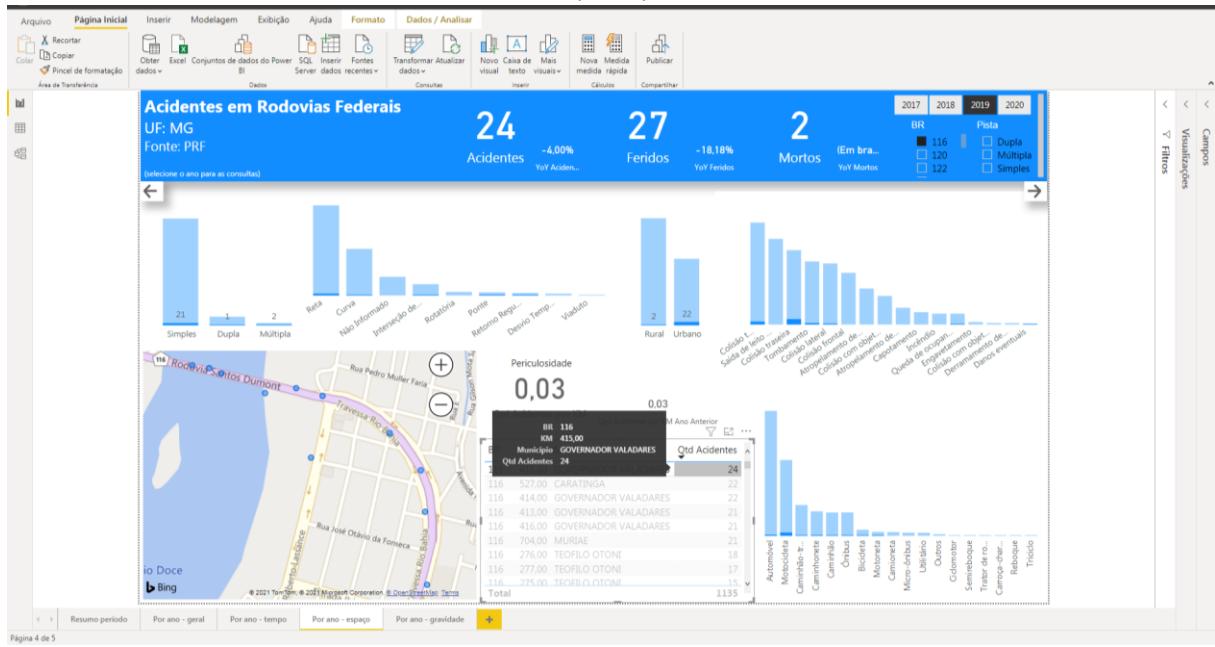
Figura 119: Dashboard “Por ano - tempo” do MS-PowerBI com seleção de ano, BR e dia da semana, com valores da predição
(autor)



Com o dashboard “Por ano - espaço” (Figura 120), selecionando filtro do ano para 2019 e a BR-116, percebe-se que o tipo de pista simples ainda apresentou maior quantidade acidentes. Selezionando, então, a pista simples e o quilômetro 415,00 que apresentou maior quantidade de acidentes, notou-se que a quantidade

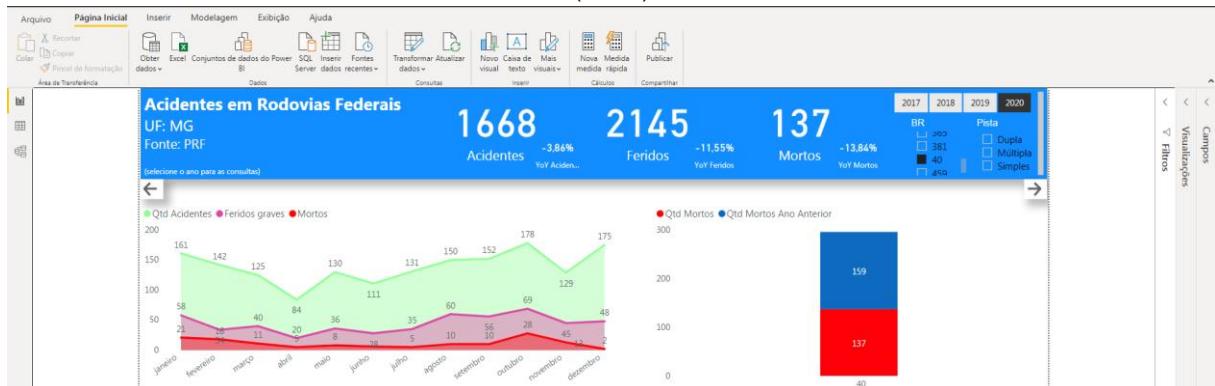
aumentou de 21 para 24 bem como aumento de feridos de 26 para 27 e de mortos de 0 para 2, conforme pode ser verificado confrontando com a Figura 64.

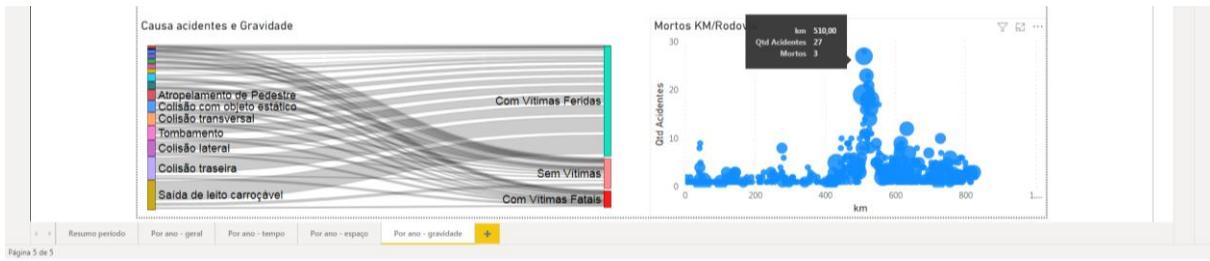
Figura 120: Dashboard “Por ano - espaço” do MS-PowerBI com seleção de ano, BR e tipo de pista, com valores da previsão (autor)



Com o dashboard “Por ano - gravidade” (Figura 66), selecionando o filtro de ano de 2020 e a BR-040, verificou-se o comportamento no ano com diferenças nas linhas de feridos graves e mortos, bem como na quantidade de mortos do ano que passou de 107 para 137. As principais causas presumíveis relacionadas à classificação dos acidentes (sem vítimas e com vítimas – feridas e fatais) mantiveram-se as mesmas. E, no caso da distribuição de acidentes por quilômetro com a quantidade de mortos, no km 510, teve um aumento de um morto.

Figura 121: Dashboard “Por ano - gravidade” do MS-PowerBI com seleção de ano e BR, com valores da previsão (autor)

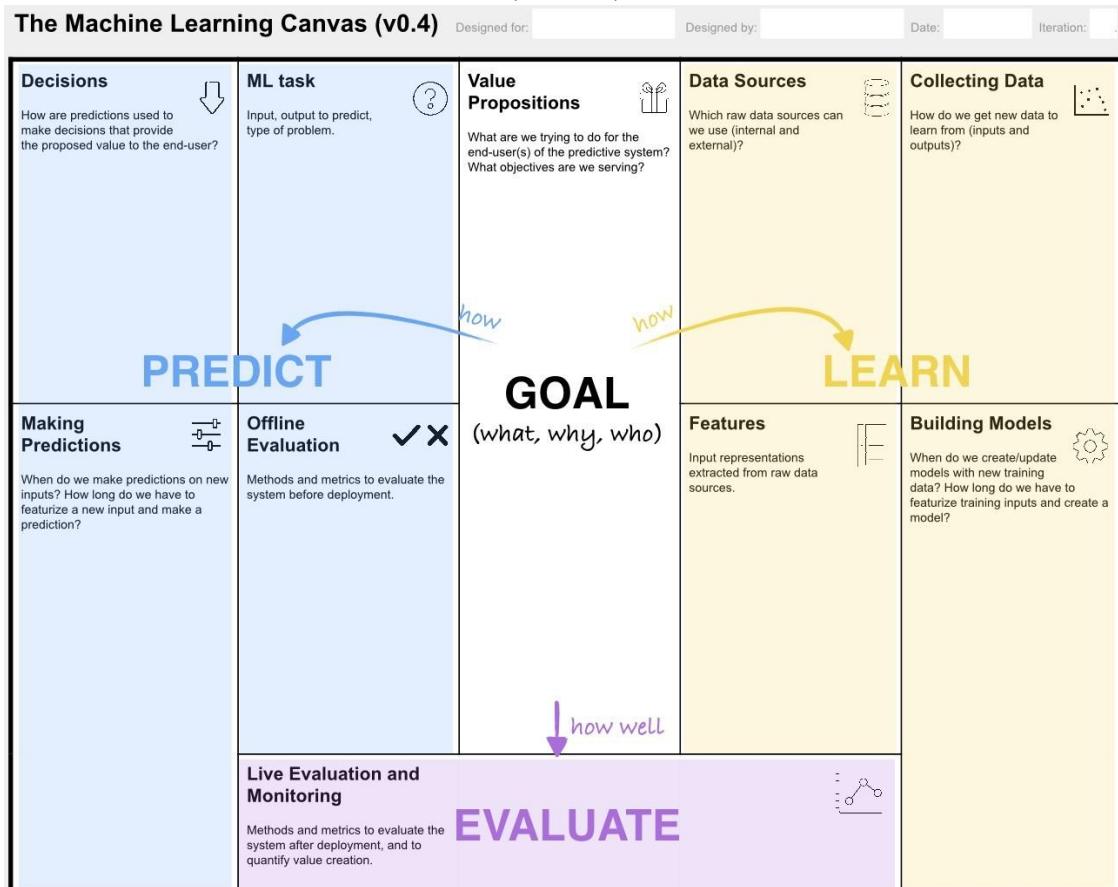




9. Apresentação dos resultados

Para a apresentação dos resultados obtidos, foi utilizado o modelo de Canvas proposto por Dourard¹⁵ (Figura 122).

**Figura 122: Modelo de Canvas para projetos de Machine Learning
(Dourard)**



¹⁵ <https://www.louisdorard.com/machine-learning-canvas#download>

<p>Decisions</p>  <p>O resultado do modelo é a predição do estado físico de vítimas em acidentes de trânsito nas rodovias federais, possibilitando determinar ações de fiscalização e de educação no trânsito.</p>	<p>ML task</p>  <p>A tarefa de <i>machine learning</i> é a classificação do estado físico das vítimas, identificando o estado dos que estavam como "Não Informado" na base de dados e que poderiam ser "Ilesos, Feridos Leves, Feridos Graves ou Óbito".</p> <p>A variável alvo é 'estado_fisico'.</p> <p>Dos algoritmos testados, o Random Forest, após ter sido aprimorado, foi o que apresentou mais alta acurácia.</p>	<p>Value Propositions</p>  <p>A intenção é de ter todos os valores de estado físico de vítimas identificados podendo analisar e estudar a gravidade dos acidentes.</p> <p>Conhecendo a gravidade dos acidentes, o local em que ocorreram, a quantidade de ocorrências, dentre outros, pode-se identificar ações preventivas, como: fiscalização, educação, mudanças de geometria na via, manutenção de vias, sinalização de vias.</p>	<p>Data Sources</p>  <p>A principal fonte de dados foram os arquivos com as ocorrências de acidentes registrados pela PRF.</p>	<p>Collecting Data</p>  <p>Os novos dados poderão ser coletados no portal de dados abertos da PRF.</p>		
<p>Making Predictions</p>  <p>As previsões são realizadas após o tratamento e o processamento dos dados e de forma muito rápida.</p>	<p>Offline Evaluation</p>  <p>Técnicas de validação cruzada para avaliação do modelo, bem como as métricas de desvio-padrão, coeficiente de variação, valor mínimo e valor máximo dos parâmetros: 1. "accuracy" 2. "precision" 3. "recall" 4. "f1" 5. "jaccard"</p>	<p>Features</p>  <p>As variáveis preditoras X são: 'dia_semana','br','km','causa_acidente','tipo_acidente','classificacao_acidente','fase_dia','condicao_meteorologica','tipo_pista','tracado_via','uso_solo','tipo_veiculo','tipo_envolvido','sexo','elemento_transito'.</p> <p>E a variável alvo Y é: 'estado_fisico'.</p>				<p>Building Models</p>  <p>Foram testados três algoritmos: SGDClassifier, DecisionTree Classifier e o RandomForest Classifier.</p> <p>O modelo foi construído com o algoritmo que teve o melhor desempenho na classificação, no caso o Random Forest.</p>
<p>Live Evaluation and Monitoring</p> <p>A avaliação e o monitoramento do modelo foi realizada por meio da classificação do estado físico 'Não Informado', medindo a acurácia das classes (accuracy_score e jaccard_score), bem como pela medição do tempo gasto para classificação de novos registros pelo modelo.</p>						

8. Links

A seguir, estão os links do vídeo de apresentação e do repositório contendo os dados utilizados no projeto.

Link para o vídeo: <https://www.youtube.com/watch?v=QWwKYImVfNY>

Link para o repositório: <https://github.com/liliancs74/TCCPUCMG>

REFERÊNCIAS

CNT. Confederação Nacional do Transporte (CNT), Anuário do Transporte 2020, 2020, disponível em <https://anuariodotransporte.cnt.org.br/2020/Rodoviario/1-1-Principais-dados>.

CNT. Confederação Nacional do Transporte (CNT), Pesquisa CNT de rodovias 2016: relatório gerencial. – 20.ed. – Brasília: CNT: SEST: SENAT, 2016.

COELHO, Heleny da Silva. Análise da influência das características físicoperacionais das vias na ocorrência de acidentes de trânsito nas rodovias federais. 1999. 186 f. Dissertação (Mestrado em Engenharia Civil) - Universidade de Brasília, Brasília, 1999.

DENATRAN. Departamento Nacional de Trânsito (DENATRAN), Dados de Frota de Veículos, 2021, disponível em <https://www.gov.br/infraestrutura/pt-br/assuntos/transito/conteudo-denatran>.

DNIT. Departamento Nacional de Infraestrutura de Transportes (DNIT). PLANO NACIONAL DE CONTAGEM DE TRÁFEGO (PNCT), 2020. Disponível em <http://servicos.dnit.gov.br/dadospnct>.

DNIT. Departamento Nacional de Infraestrutura de Transportes (DNIT). SNV e PNV, 2021. Disponível em <https://www.gov.br/dnit/pt-br/assuntos/atlas-e-mapas/pnv-e-snv>.

ELVIK, R. To what extent can theory account for the findings of road safety evaluation studies? Institute of Transport Economics. Oslo, Noruega, 2003.

EPL. Empresa de Planejamento e Logística (EPL), Transporte inter-regional de carga no Brasil - Panorama 2015, 2020, disponível em <http://www.epl.gov.br/transporte-inter-regional-de-carga-no-brasil-panorama-2015>.

FUCHS, M. Introduction to SGD Classifier, 2019, disponível em <https://michael-fuchs-python.netlify.app/2019/11/11/introduction-to-sgd-classifier/>.

IPEA. Instituto de Pesquisas e Estudos Aplicados (IPEA), Acidentes de Trânsito nas Rodovias Federais Brasileiras: Caracterização, Tendências e Custos para a Sociedade, 2015.

IPEA/DENATRAN/ANTP. Instituto De Pesquisa Econômica Aplicada (IPEA) / Departamento Nacional de Trânsito (DENATRAN) / Associação Nacional de Transportes Públicos (ANTP). Impactos sociais e econômicos dos acidentes de trânsito nas rodovias brasileiras. Brasília, 2006.

IPEA/DENATRAN/ANTP. Instituto De Pesquisa Econômica Aplicada (IPEA) / Departamento Nacional de Trânsito (DENATRAN) / Associação Nacional de Transportes Públicos (ANTP). Fatores condicionantes da gravidade dos acidentes de trânsito nas rodovias brasileiras. Brasília, 2008.

MENEZES, F. A. B. Análise e Tratamento de Trechos Rodoviários Críticos em Ambientes de Grandes Centros Urbanos. Dissertação de Mestrado em Engenharia de Transportes. Universidade Federal do Rio de Janeiro – UFRJ. Rio de Janeiro, Brasil, 2001.

OLIVEIRA, Marcos Pimentel de. O impacto da utilização de medidores eletrônicos de velocidade na redução de acidentes de trânsito em área urbana. 2008. 78 f. Dissertação (Mestrado em Engenharia Civil) - Universidade Federal de Uberlândia, Uberlândia, 2008.

OMS. Organização Mundial da Saúde (OMS), Sumário do Relatório Global sobre o Estado da Segurança Viária 2015, 2015, disponível em http://www.who.int/violence_injury_prevention/road_safety_status/2015/Summary_GSRRS2015_POR.pdf.

OPAS/OMS-MS Prevenção de lesões causadas pelo trânsito: Manual de Treinamento, 2011.

OPAS/OMS-MS. Organização Mundial da Saúde (OMS). Gestão da velocidade: um manual de segurança viária para gestores e profissionais da área. Brasília, D.F.: OPAS, 2012.

PRF. Departamento de Polícia Rodoviária Federal (DPRF), Acidentes em rodovias federais, 2021, disponível em: <https://www.prf.gov.br/portal/dados-abertos/acidentes/acidentes>.

SKLEARN. Scikit-Learn. Choosing the right estimator, 2020, disponível em https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html.

SOARES, Lilian C.; PRADO, H.; BALANIUK, R.; FERNEDA, E.; BORTOLI, A. Caracterização de acidentes rodoviários e as ações governamentais para a redução de mortes e lesões no trânsito: Um estudo de dados da rodovia BR-101 no período de 2014 a 2016. Revista Transporte y Territorio, no. 19, pp. 182-220, 2018. DOI: <https://doi.org/10.34096/rtt.v19.i19.5331>