

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

NÚCLEO DE EDUCAÇÃO A DISTÂNCIA

Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data

Henrique Borsatti Lisboa

ANÁLISE DO COMÉRCIO INTERNACIONAL E PREVISÃO DOS MODAIS DE
TRANSPORTE INTERNACIONAL DE MERCADORIAS

São Paulo

2021

Henrique Borsatti Lisboa

ANÁLISE DO COMÉRCIO INTERNACIONAL E PREVISÃO DOS MODAIS DE
TRANSPORTE INTERNACIONAL DE MERCADORIAS

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.

São Paulo
2021

SUMÁRIO

1. Introdução	4
1.1. Contextualização	4
1.2. O Problema Proposto	5
2. Coleta de Dados.....	7
3. Processamento/Tratamento de Dados.....	15
4. Análise e Exploração dos Dados	22
4.1. Análise dos Parceiros Comerciais do Brasil no ano de 2020	22
4.2. Análise da Balança Comercial.....	31
4.3. Análise dos Tipos das Mercadorias	36
4.4. Análise Temporal.....	42
4.5. Análise dos Modais de Transporte.....	53
5. Criação de Modelos de Machine Learning	68
5.1. Naive Bayes (NB) e Random Forest (RF) na Exportação	78
5.2. Naive Bayes (NB) e Random Forest (RF) na Importação	81
5.3. Demais Modelos na Exportação	82
5.4. Demais Modelos na Importação	88
6. Apresentação dos Resultados	94
6.1. Random Forest (RF) na Exportação.....	95
6.2. Random Forest (RF) na Importação	97
6.3. Demais Modelos na Exportação	99
6.4. Demais Modelos na Importação	106
7. Links	116
REFERÊNCIAS	117
APÊNDICE	119

1. Introdução

1.1. Contextualização

Na condição de Auditor-Fiscal da Receita Federal, tendo minha participação no Curso de Especialização em Ciência de Dados e Big Data da PUC Minas Virtual sido custeada pela Secretaria da Receita Federal do Brasil e sendo minha aprovação no curso condição para minha promoção funcional, resolvi buscar um tema que cumprisse as seguintes condições:

- a) Atender os requisitos estabelecidos pela instituição de ensino PUC Minas Virtual para a elaboração do Trabalho de Conclusão de Curso (TCC);
- b) Ser do interesse da Secretaria da Receita Federal do Brasil, conforme o inventário de Competências institucionais e individuais no âmbito da Secretaria da Receita Federal do Brasil constante na Portaria RFB nº 925/2018;
- c) Respeitar o sigilo fiscal (art. 2º da Portaria RFB nº 2344/2011) e o sigilo funcional (art. 116, inciso VIII da Lei nº 8.112, de 1990).

Com base nisso e dada minha experiência profissional com o Despacho Aduaneiro de Importação, busquei na internet dados de domínio público das importações e exportações brasileiras, base extensa e rica, que possibilitasse as mais diversas análises e previsões. Além disso, decidi propor, no modelo de aprendizado de máquina a ser desenvolvido, algum possível uso do resultado nas atividades do órgão, em especial para as atividades de fiscalização e gestão de riscos aduaneiros.

Note que não há afronta ao sigilo fiscal, por serem os dados de domínio público, não possuindo informações sobre a situação econômica ou financeira do sujeito passivo ou de terceiros e sobre a natureza e o estado de seus negócios ou atividades. Além disso, também não há afronta ao sigilo funcional, pois não tenho conhecimentos e nem realizo ou realizei parte das atividades de gerenciamento de risco e o uso proposto dos dados públicos (básicos ante os dados realmente disponíveis à fiscalização) é puramente teórico e educativo.

1.2. O Problema Proposto

O comércio internacional é a troca de bens e serviços através de fronteiras internacionais ou territórios. A importância do comércio internacional reside no fato de que as economias nacionais não são capazes de produzir todos os bens de que necessitam, fazendo com que cada nação se especialize na produção de bens que consigam alcançar maior competitividade.

A Receita Federal é responsável pelo controle aduaneiro, que consiste em realizar o controle e a fiscalização da entrada e saída de mercadorias de origem estrangeira no país, o acompanhamento do despacho aduaneiro, a verificação da correta informação da base de cálculo de incidência dos tributos devidos na operação e o controle da aplicação de medidas de defesa comercial.

Assim sendo, propõe-se inicialmente uma análise minuciosa das importações e exportações brasileiras. Utilizaremos os dados mais recentes disponíveis, do ano de 2020. Além do mais, não é possível pensarmos no ano de 2020 sem lembrarmos do início da pandemia da doença do coronavírus (COVID-19). Por isso, incluiremos na análise os marcos temporais importantes da doença e veremos como o avanço dela afetou (ou não) as operações.

Após isso, conforme requisito da instituição de ensino, criaremos um modelo de Aprendizado de Máquina (*Machine Learning*) com base nos dados disponíveis, alinhado aos objetivos do órgão. Optou-se por fazer uma previsão dos modais de transporte de carga internacional.

Os cinco principais modais de transporte de carga no Brasil, conforme pesquisas na internet, são o rodoviário, aéreo, ferroviário, aquaviário e dutoviário. Cada qual possui vantagens e desvantagens em relação ao demais. Por exemplo, o aquaviário marítimo possui como vantagens a capacidade de transportar grandes quantidades de carga e o baixo custo e como desvantagem o longo tempo de trânsito. O aéreo, por outro lado, possui uma maior limitação na quantidade de carga transportada e um alto custo, mas é o modal mais rápido. As diferenças entre os modais não se limitam a isso e podem incluir outros assuntos como flexibilidade de rotas, custos de manuseio e embalagem, dependência de outros modais etc.

Diversos podem ser os motivos que guiam a escolha de um importador ou exportador por determinado modal em detrimento dos outros: tipo de mercadoria transportada, volume, padrões de segurança exigidos, urgência da entrega,

orçamento disponível etc. Por exemplo, o padrão para mercadorias como commodities (grãos, minérios etc.) é o modal marítimo e para mercadorias de alto valor agregado ou de caráter urgente, o modal aéreo.

Algumas vezes, contudo, isso não é o que ocorre de fato. Diversos motivos, muitas vezes legítimos, como a urgência na entrega ou o transporte em conjunto com outras mercadorias de características distintas, fazem com que uma operação de comércio exterior fuja a essa regra. Contudo, em outros casos, uma operação aparentemente incomum pode ser um indício de um erro de preenchimento da declaração ou até de uma fraude. Retornando ao exemplo anterior dos modais marítimo e aéreo: por que motivo uma empresa estaria exportando uma mercadoria de alto valor agregado e de baixo volume pelo modal marítimo? Por que motivo uma empresa estaria importando grãos pelo modal aéreo e pagando 10 (dez) vezes o valor da mercadoria em frete, se todos seus concorrentes utilizam outro modal? Poderiam ser casos de subfaturamento, superfaturamento, falsa declaração de conteúdo ou outra operação com suspeita de irregularidade sujeita ao Procedimento Especial de Controle Aduaneiro – PECA, regulamentado pela IN RFB nº 1.169/2011? Nesses casos, pode ser necessária a análise fiscal para verificar o motivo dessa discrepância.

Utilizando a técnica dos [5-Ws](#) (principais perguntas que devem ser feitas e respondidas ao investigar e relatar um fato ou situação, sendo aplicável a várias atividades profissionais), podemos organizar assim o problema para uma melhor sistematização do projeto:

Why? (Por quê?): A análise dos dados do Comércio Exterior é importante pois dela resultam as políticas de fomento e estímulo ao comércio exterior que, em última instância, impactam grandemente o crescimento do país. A análise e a busca de fraudes no comércio exterior, parte essencial do controle aduaneiro, tem como função, entre outras, a proteção contra a evasão fiscal, concorrência desleal e entrada e saída de mercadorias proibidas.

Who? (Quem?): Os dados objeto da análise são do Governo Federal do Brasil, disponibilizados pelo sistema Comex Stat do Ministério da Indústria, Comércio Exterior e Serviços (MDIC).

What? (O quê?): O problema proposto é a análise dos dados das exportações e importações brasileiras no ano de 2020 e os efeitos da pandemia do

novo coronavírus, assim como a previsão do modal de transporte internacional de mercadorias com o objetivo de detectar a escolha de modais incomuns que, em alguns casos, poderiam indicar alguma infração.

Where? (Onde?): Os aspectos geográficos são de abrangência nacional.

When? (Quando?): O período analisado é da totalidade do ano de 2020.

Todo o desenvolvimento do projeto foi realizado utilizando a linguagem de programação de alto nível Python. Escolheu-se essa linguagem por sua robustez e pelas diversas bibliotecas disponíveis para análise de dados e aprendizado de máquina. Utilizou-se o ambiente computacional web *Jupyter Notebook* baseado na versão Python 3.7.10 da linguagem Python presente na plataforma *Google Colaboratory*, executado na nuvem. *Jupyter Notebook* (ou simplesmente notebook) é uma aplicação web que permite criar e compartilhar documentos que possuem ao mesmo tempo código interativo e textos explicativos.

2. Coleta de Dados

Os dados das Estatísticas de Comércio Exterior em Dados Abertos foram obtidos do sítio do Governo Federal na data de 07/02/2021 no seguinte endereço base: <https://www.gov.br/produtividade-e-comercio-exterior/pt-br/assuntos/comercio-exterior/estatisticas/base-de-dados-bruta>.

Não se trata de um *dataset* único e consolidado. Foram inicialmente utilizadas duas bases de dados detalhadas por NCM (Nomenclatura Comum do Mercosul, código utilizado para classificação fiscal das mercadorias), uma para exportação e uma para importação (EXP_2020.csv e IMP_2020.csv), enriquecidas por outros 5 (cinco) *datasets*, a saber: NCM.csv, NCM_SH.csv, PAIS.csv, URF.csv e VIA.csv. A descrição de cada campo/coluna desses *datasets* será detalhada nas tabelas abaixo. Em relação a granularidade dos dados, cada linha do *dataset* é uma operação de comércio exterior correspondente a uma adição de uma declaração, definida por um único código NCM. Na declaração real há ainda a divisão das adições em itens e é possível colocar uma descrição para cada item. Essas informações, assim como diversas outras referentes à identificação do sujeito passivo, não estão disponíveis nesses *datasets*.

Nem todos os dados utilizados no trabalho são provenientes da página do sítio do Governo Federal. Parte dos dados é proveniente de raspagem de dados (*data scraping*) das páginas das próprias resoluções do Governo Federal que concederam reduções temporárias das alíquotas do Imposto de Importação para determinadas mercadorias utilizadas na prevenção e no combate do novo coronavírus (<https://www.in.gov.br/en/web/dou/-/resolucao-n-17-de-17-de-marco-de-2020-248564246> e <https://www.in.gov.br/en/web/dou/-/resolucao-n-31-de-7-de-abril-de-2020-251704729>).

Para realizar essa raspagem poderíamos ter utilizado diversas bibliotecas, como a conhecida Beautiful Soup, utilizada para a análise de documentos HTML e XML. Contudo, a própria biblioteca Pandas, já utilizada em outras partes da análise, possui ferramentas prontas para esse propósito.

Iremos agora começar a mostrar trechos dos códigos presentes no *notebook* elaborado e precisamos, antes de tudo, mostrar as bibliotecas que serão utilizadas nesse trabalho de importação e análise de dados. As bibliotecas utilizadas para criação dos modelos de aprendizagem de máquina serão mostradas no tópico específico.

```
#Importando as bibliotecas que iremos utilizar na importação e análise
import string
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
import requests
```

- 1) string – biblioteca para operações comuns de strings (textos);
- 2) numpy – biblioteca para cálculos em vetores e matrizes multidimensionais;
- 3) pandas – biblioteca para análise e manipulação de dados;
- 4) seaborn – biblioteca para visualização de dados baseada no matplotlib;
- 5) matplotlib – biblioteca para visualização de dados;
- 6) wordcloud – biblioteca utilizada para criação de nuvens de palavras;
- 7) requests – biblioteca para fazer requisições HTTP.

Superada a questão das bibliotecas utilizadas, passemos primeiramente à implementação da raspagem de dados da resolução. As explicações de cada linha seguem em comentários junto do código:

```
#URL da Resolução
url = 'https://www.in.gov.br/en/web/dou/-/resolucao-n-17-de-17-de-marco-de-2020-248564246'

try:
    r = requests.get(url)
except requests.exceptions.RequestException as e:
    print(e)

#Colocando todas tabelas da página em uma lista
df_list = pd.read_html(r.text)
#Selecionando a primeira tabela
ncm_marco = df_list[0]
#Colocando a primeira linha como nome das colunas
ncm_marco.columns = ncm_marco.iloc[0]
ncm_marco = ncm_marco[1:]
#Retirando os pontos da coluna NCM e descartando as descrições e valores nulos de NCM (usados para os ex-tarifários)
ncm_marco['NCM'].str.replace('.','').dropna()
ncm_marco.head(10)
```

Que fornece como resultado os seguintes dados:

```
1    22072019
2    29349934
3    38089419
4    38089429
5    39262000
7    39269040
8    39269090
16   40151100
17   40151900
18   56012299
Name: NCM, dtype: object
```

Todo código do *notebook* elaborado é possuí comentários como o trecho acima, de modo a explicar o uso de todas as classes, objetos, funções e métodos. Não faremos todos esses comentários novamente aqui, para não poluir demais o documento, exceto em ocasiões pontuais em que acreditarmos que tal explicação seja necessária para o entendimento pleno da questão.

Nessa raspagem obtivemos todos os dados de NCMs isentas de determinada resolução e podemos com isso realizar as análises que serão vistas posteriormente. As outras raspagens realizadas no *notebook* seguem o mesmo procedimento e podem ser encontradas diretamente no *notebook*.

Por fim, obtidos todos os dados que serão utilizados no trabalho, destacamos abaixo os campos, descrições e tipos iniciais de cada *dataset*.

EXP_2020 e IMP_2020

Nome da coluna/campo	Descrição	Tipo
CO_ANO	Ano da Importação. Ex: 2020	int64
CO_MES	Mês da Importação. Ex: 10	int64
CO_NCM	Código NCM da Mercadoria. Código de 8 dígitos adotado como convenção da categorização de mercadorias no Mercosul. Baseado no Sistema Harmonizado (código internacional de mesmo propósito com 6 dígitos) é utilizado para classificar e definir o tratamento tributário e administrativo de cada mercadoria importada ou exportada. Ex: 0710.10.00 é a NCM de Batatas congeladas não cozidas ou cozidas em água ou vapor.	int64
CO_UNID	Código de 2 dígitos da unidade de medida estatística da mercadoria. Ex: 17 é Litro.	int64
CO_PAIS	Código do país. Ex: 105 é o Brasil	int64
SG_UF_NCM	Sigla do Estado da Operação. Ex: SP é São Paulo	object
CO_VIA	Código da via de transporte da operação. Ex: 1 é Marítima.	int64
CO_URF	Código de 7 dígitos da Unidade da RFB responsável pela operação. Ex: 0817600 é o Aeroporto Internacional de Guarulhos	int64
QT_ESTAT	Quantidade da mercadoria na unidade de medida estatística	int64

	determinada pelo CO_UNID. Ex: 2000.	
KG_LIQUIDO	Peso em kg líquidos (sem considerar peso da embalagem). Ex: 50 kgs.	int64
VL_FOB	Valor da mercadoria em si, no local de embarque (sem considerar os custos de transporte ou seguro) em dólares. FOB vem da sigla em inglês de Free On Board, que significa “livre a bordo”, sendo toda a responsabilidade pelo transporte da mercadoria é do cliente, incluindo os riscos e os custos. Ex: 3.500,00 USD.	int64

Em relação aos *datasets* utilizados para enriquecimento dos dados, apenas serão detalhadas as colunas efetivamente utilizadas e não todas presentes.

NCM

Nome da coluna/campo	Descrição	Tipo
CO_NCM	Código NCM da Mercadoria. Código de 8 dígitos adotado como convenção da categorização de mercadorias no Mercosul. É utilizado para classificar e definir o tratamento tributário e administrativo de cada mercadoria importada ou exportada. É dividido em Capítulo, Posição, Item e Subitem, a cada 2 dígitos. Ex: 8517.12.11	int64
NO_NCM_POR	Descrição dessa NCM em português. Ex: a descrição do código	object

	8517.12.11 é “Portáteis (por exemplo, walkie talkie e handle talkie)”.	
--	--	--

NCM_SH

Nome da coluna/campo	Descrição	Tipo
CO_SH6	Código do Sistema Harmonizado da Mercadoria. São os 6 primeiros dígitos do código NCM, englobando Capítulo, Posição e Item. Ex: 8517.12	int64
CO_SH4	Código do Sistema Harmonizado da Mercadoria com 4 dígitos, englobando Capítulo, Posição. Ex: 8517	int64
CO_SH2	Código do Sistema Harmonizado da Mercadoria com 2 dígitos, englobando Capítulo. Ex: 85	int64
NO_SH6_POR	Descrição do item em português. Ex: a descrição do código 8517.12 é “Telefones para redes celulares e para outras redes sem fio”.	object
NO_SH4_POR	Descrição da posição em português. Ex: a descrição do código 8517 é “APARELHOS TELEFÔNICOS, INCLUINDO OS TELEFONES PARA REDES CELULARES E PARA OUTRAS REDES SEM FIO; OUTROS APARELHOS PARA A TRANSMISSÃO OU RECEPÇÃO DE	object

	VOZ, IMAGENS OU OUTROS DADOS, INCLUINDO OS APARELHOS PARA COMUNICAÇÃO EM REDES POR FIO OU REDES SEM FIO (TAL COMO UMA REDE LOCAL (LAN) OU UMA REDE DE ÁREA ESTENDIDA (ALARGADA*) (WAN)), EXCETO OS APARELHOS DAS POSIÇÕES 84.43, 85.25, 85.27 OU 85.28.”.	
NO_SH2_POR	Descrição do capítulo em português. Ex: a descrição do código 85 é “Máquinas, aparelhos e materiais elétricos, e suas partes; aparelhos de gravação ou de reprodução de som, aparelhos de gravação ou de reprodução de imagens e de som em televisão, e suas partes e acessórios”.	object

PAIS

Nome da coluna/campo	Descrição	Tipo
CO_PAIS	Código do país. Ex: 105	int64
NO_PAIS	Nome do país em português: Ex: “Brasil”	object

URF

Nome da coluna/campo	Descrição	Tipo

CO_URF	Código de 7 dígitos da Unidade da RFB responsável pela operação. Ex: 0817600	int64
NO_URF	Nome do URF em português: Ex: “0817600 - AEROPORTO INTERNACIONAL DE SAO PAULO/GUARULHOS”	object

VIA

Nome da coluna/campo	Descrição	Tipo
CO_VIA	Código da via de transporte da operação. Ex: 1	int64
NO_VIA	Nome da via em português: Ex: “MARITIMA”	object

E os *datasets* obtidos pela raspagem de dados:

NCM_MARCO E NCM_ABRIL

Nome da coluna/campo	Descrição	Tipo
NCM	Código NCM da Mercadoria. Código de 8 dígitos adotado como convenção da categorização de mercadorias no Mercosul. É utilizado para classificar e definir o tratamento tributário e administrativo de cada mercadoria importada ou exportada. É dividido em Capítulo, Posição, Item e Subitem, a cada 2 dígitos. Ex: 8517.12.11	object

3. Processamento/Tratamento de Dados

O conjunto de dados obtidos anteriormente teve que passar por algumas etapas de tratamento/transformação/junção para possibilitar a posterior análise e desenvolvimento dos modelos. Nesse tratamento, normalmente precisamos realizar a correção dos tipos de dados, tratamento de valores ausentes e de *outliers* (que se diferenciam drasticamente de todos os outros, pontos fora da curva normal). Note que essa é uma etapa iterativa. Por diversas vezes, durante a análise ou etapas posteriores, notamos diversas ausências, falhas ou necessidades de alteração dos dados que necessitaram de novos processamentos e tratamentos. Por isso, alguns tratamentos aqui feitos serão justificados e compreendidos apenas quando analisarmos o tópico seguinte da análise de dados. No mesmo sentido, as etapas da análise serão aqui brevemente mencionadas para justificar esses procedimentos.

Como estamos executando o notebook na nuvem do Google Colab, armazenamos os dados no Google Drive (em uma pasta “data” na raiz do Google Drive) e devemos, antes de tudo, montar o drive para podermos importar e trabalhar com os *datasets*. Isso é feito com o seguinte comando:

```
#Montando o Google Drive
from google.colab import drive
drive.mount('/content/gdrive')
```

Mounted at /content/gdrive

Inicialmente, devemos carregar os *datasets* para os *Dataframes*. *Dataframe* é uma estrutura de dados bidimensional da biblioteca Pandas, com os dados alinhados de forma tabular em linhas e colunas, mutável em tamanho e potencialmente heterogênea, semelhantemente a uma matriz. Feito isso, transformamos as colunas CO_NCM do tipo int64 para o tipo *object*, utilizado para textos. Isso foi feito pois, posteriormente, iremos realizar a junção dos *Dataframes* e precisaremos dos 6 primeiros dígitos da coluna CO_NCM dos *datasets* EXP_2020 e IMP_2020 para realizar a junção com a coluna CO_SH do *dataset* NCM_SH, que contém as descrições das mercadorias por capítulo, posição, item e subitem.

Se tentarmos realizar essa junção fazendo a divisão inteira do CO_NCM por 100, encontramos um problema. As duas primeiras seções da NCM (Animais Vivos e Produtos do Reino Animal e Produtos do Reino Vegetal) começam com zero. Por exemplo, a NCM de Primatas é 0106.11.00. Ao tratar esse número como inteiro e

dividir por 100, teremos apenas 5 dígitos em vez de 6, 10611, impossibilitando a junção. Desse modo, optou-se por tratar esses códigos como textos e realizar operações diretamente nos textos. Diversos outros tratamentos, que serão explicitados adiante, foram feitos para permitir a integração das diversas fontes de dados.

Voltando então à importação dos dados e criação dos *dataframes*, temos o seguinte código que realiza o carregamento e transformação dos dados:

```
#Carregando e já aplicando uma conversão na coluna do código NCM para termos sempre 8 dígitos (adicionando zeros no começo)
df_exp = pd.read_csv('/content/gdrive/MyDrive/data/EXP_2020.csv', sep=';', converters={'CO_NCM': '{:0>8}'.format()})
df_imp= pd.read_csv( '/content/gdrive/MyDrive/data/IMP_2020.csv', sep=';', converters={'CO_NCM': '{:0>8}'.format()})
```

Em seguida, verificamos brevemente os dados e seus tipos:

```
#Exibindo os cabeçalhos
display(df_exp.head())
display(df_imp.head())

#Verificando os tipos dos dados
display(df_exp.info())
display(df_imp.info())
```

	CO_ANO	CO MES	CO_NCM	CO_UNID	CO_PAIS	SG_UF_NCM	CO_VIA	CO_URF	QT_ESTAT	KG_LIQUIDO	VL_FOB
0	2020	12	44189900	10	245	SC	1	927800	44064	44064	37651
1	2020	9	06029029	11	845	SP	7	1017701	20944	17274	41051
2	2020	8	87082999	11	586	PR	7	917500	6754	3712	19696
3	2020	12	22089000	17	351	RS	0	1017700	24	24	128
4	2020	9	85182100	11	538	SP	4	817600	897	157	7086
	CO_ANO	CO MES	CO_NCM	CO_UNID	CO_PAIS	SG_UF_NCM	CO_VIA	CO_URF	QT_ESTAT	KG_LIQUIDO	VL_FOB
0	2020	11	39199020	10	493	SP	1	817800	11	11	396
1	2020	11	90291010	11	249	RJ	4	717700	1	0	151
2	2020	11	38119090	10	249	SC	1	927800	982	982	6339
3	2020	11	84819090	10	399	SP	4	817600	409	409	19329
4	2020	11	84099112	11	23	SP	4	817700	12	205	9610

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1417821 entries, 0 to 1417820
Data columns (total 11 columns):
 #   Column      Non-Null Count   Dtype  
--- 
 0   CO_ANO       1417821 non-null    int64  
 1   CO_MES       1417821 non-null    int64  
 2   CO_NCM       1417821 non-null    object  
 3   CO_UNID      1417821 non-null    int64  
 4   CO_PAIS      1417821 non-null    int64  
 5   SG_UF_NCM    1417821 non-null    object  
 6   CO_VIA       1417821 non-null    int64  
 7   CO_URF       1417821 non-null    int64  
 8   QT_ESTAT     1417821 non-null    int64  
 9   KG_LIQUIDO   1417821 non-null    int64  
 10  VL_FOB      1417821 non-null    int64  
dtypes: int64(9), object(2)
memory usage: 119.0+ MB
None

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1748493 entries, 0 to 1748492
Data columns (total 11 columns):
 #   Column      Dtype  
--- 
 0   CO_ANO       int64  
 1   CO_MES       int64  
 2   CO_NCM       object  
 3   CO_UNID      int64  
 4   CO_PAIS      int64  
 5   SG_UF_NCM    object  
 6   CO_VIA       int64  
 7   CO_URF       int64  
 8   QT_ESTAT     int64  
 9   KG_LIQUIDO   int64  
 10  VL_FOB      int64  
dtypes: int64(9), object(2)
memory usage: 146.7+ MB
None

```

Como podemos observar, todos dados são inteiros com 64 bits, exceto SG_UF_NCM e CO_NCM (que foi transformado de int64 para object).

Vamos verificar agora se há dados nulos nos *dataframes*:

```
#Verificando se atualmente temos valores nulos nos dataframes
display(df_exp.isnull().sum())
display(df_imp.isnull().sum())

CO_ANO      0
CO_MES      0
CO_NCM      0
CO_UNID     0
CO_PAIS     0
SG_UF_NCM   0
CO_VIA      0
CO_URF      0
QT_ESTAT    0
KG_LIQUIDO  0
VL_FOB      0
dtype: int64
CO_ANO      0
CO_MES      0
CO_NCM      0
CO_UNID     0
CO_PAIS     0
SG_UF_NCM   0
CO_VIA      0
CO_URF      0
QT_ESTAT    0
KG_LIQUIDO  0
VL_FOB      0
dtype: int64
```

Podemos observar que não há dados nulos nos *dataframes*.

Em seguida, o mesmo tratamento foi feito para os demais *dataframes*:

```
#Carregando os datasets complementares e aplicando a formatação nos campos de modo tratar diferenças e manter a correspondência
df_ncm = pd.read_csv('/content/gdrive/MyDrive/data/NCM.csv', sep=";", encoding = "ISO-8859-1", converters={'CO_NCM': '{:0>8}'.format})
df_ncm_sh = pd.read_csv('/content/gdrive/MyDrive/data/NCM_SH.csv', sep=";", encoding = "ISO-8859-1",
                       converters={'CO_SH6': '{:0>6}'.format, 'CO_SH4': '{:0>4}'.format, 'CO_SH2': '{:0>2}'.format})
df_pais = pd.read_csv('/content/gdrive/MyDrive/data/PAIS.csv', sep=";", encoding = "ISO-8859-1")
df_urf = pd.read_csv('/content/gdrive/MyDrive/data/URF.csv', sep=";", encoding = "ISO-8859-1")
df_via = pd.read_csv('/content/gdrive/MyDrive/data/VIA.csv', sep=';')
```

Retiramos as colunas desnecessárias e podemos verificar os cabeçalhos:

	CO_NCM	NO_NCM_POR
0	38085910	Outras mercadorias mencionadas na Nota de subp...
1	38085921	Mercadorias à base de metamidofós (ISO) ou mon...
2	38085922	Mercadorias à base de endossulfan (ISO), apres...
3	38085923	Mercadorias à base de alaclor (ISO), apresenta...
4	38085929	Mercadorias à base de outras substâncias, apre...

	CO_SH6	CO_SH4	CO_SH2	NO_SH6_POR	NO_SH4_POR	NO_SH2_POR
0	010110	0101	01	Animais vivos das espécies cavalar, asinina e ...	Cavalos, asininos e muares, vivos	Animais vivos
1	010111	0101	01	Cavalos reprodutores, de raça pura	Cavalos, asininos e muares, vivos	Animais vivos
2	010119	0101	01	Outros cavalos, vivos	Cavalos, asininos e muares, vivos	Animais vivos
3	010120	0101	01	Asininos e muares vivos	Cavalos, asininos e muares, vivos	Animais vivos
4	010121	0101	01	Cavalos reprodutores de raça pura	Cavalos, asininos e muares, vivos	Animais vivos

	CO_PAIS	NO_PAIS
0	0	Não Definido
1	13	Afeganistão
2	15	Aland, Ilhas
3	17	Albânia
4	20	Alboran-Perejil, Ilhas

	CO_URF	NO_URF
0	215200	0215200 - MONTE DOURADO
1	220101	0220101 - ITACOATIARA
2	510102	0510102 - CAMACARI
3	130101	0130101 - ALTO ARAGUAIA
4	812700	0812700 - SAO SEBASTIAO

	CO_VIA	NO_VIA
0	10	ENTRADA/SAIDA FICTA
1	99	VIA DESCONHECIDA
2	13	POR REBOQUE
3	11	COURIER
4	15	VICINAL FRONTEIRICO

Agora chegamos ao motivo de termos convertido para texto o campo CO_NCM dos *datasets* EXP_2020 E IMP_2020. Inicialmente criamos uma coluna nova nos *dataframes* de exportação e importação (denominada CO_SH6 nos *dataframes* df_exp e df_imp) com os 6 primeiros dígitos do código NCM e em seguida usamos essa coluna como chave da junção com o *dataset* NCM_SH. Fizemos o mesmo com os demais *datasets* e chaves, obtendo dois *dataframes* completos:

```
#Criando uma nova coluna nos dataframes principais que é necessária para a junção
df_exp['CO_SH6'] = df_exp['CO_NCM'].str[:6]
df_imp['CO_SH6'] = df_imp['CO_NCM'].str[:6]

#Realizando a junção dos datasets complementares com os principais
df_exp_full = df_exp.join(df_ncm.set_index('CO_NCM'),
                           on='CO_NCM').join(df_ncm_sh.set_index('CO_SH6'),
                                             on='CO_SH6').join(df_pais.set_index('CO_PAIS'),
                                                               on='CO_PAIS').join(df_urf.set_index('CO_URF'),
                                                                 on='CO_URF').join(df_via.set_index('CO_VIA'), on='CO_VIA')

df_imp_full = df_imp.join(df_ncm.set_index('CO_NCM'),
                           on='CO_NCM').join(df_ncm_sh.set_index('CO_SH6'),
                                             on='CO_SH6').join(df_pais.set_index('CO_PAIS'),
                                                               on='CO_PAIS').join(df_urf.set_index('CO_URF'),
                                                                 on='CO_URF').join(df_via.set_index('CO_VIA'), on='CO_VIA')

#Exibindo os dataframes
display(df_exp_full)
display(df_imp_full)
```

Os cabeçalhos dos *dataframes* completos (df_exp_full e df_imp_full) podem ser observados na seção 1.5. do *notebook*. Verificamos novamente que não há valores nulos após as junções, como esperado (especialmente depois do tratamento da coluna CO_NCM) e reordenamos as colunas para melhorar a visualização.

```
#Mudando as ordens das colunas nos dataframes para facilitar a visualização
df_exp_full = df_exp_full[['CO_ANO', 'CO_MES', 'CO_NCM', 'NO_NCM_POR', 'CO_SH6',
                           'NO_SH6_POR', 'CO_SH4', 'NO_SH4_POR', 'CO_SH2', 'NO_SH2_POR',
                           'CO_UNID', 'CO_PAIS', 'NO_PAIS', 'SG_UF_NCM', 'CO_VIA', 'NO_VIA',
                           'CO_URF', 'NO_URF', 'QT_ESTAT', 'KG_LIQUIDO', 'VL_FOB']]

df_imp_full = df_imp_full[['CO_ANO', 'CO_MES', 'CO_NCM', 'NO_NCM_POR', 'CO_SH6',
                           'NO_SH6_POR', 'CO_SH4', 'NO_SH4_POR', 'CO_SH2', 'NO_SH2_POR',
                           'CO_UNID', 'CO_PAIS', 'NO_PAIS', 'SG_UF_NCM', 'CO_VIA', 'NO_VIA',
                           'CO_URF', 'NO_URF', 'QT_ESTAT', 'KG_LIQUIDO', 'VL_FOB']]
```

Após esses tratamentos já passamos para a análise dos dados. Contudo, durante a análise, a depender do objetivo, novos *dataframes* foram criados e os dados tratados e processados novamente. Por motivos de organização desse trabalho, trataremos aqui apenas da parte de tratamento de dados para posteriormente apresentar as análises e os resultados. Os motivos que ensejaram esses tratamentos serão sucintamente explicitados, deixando para o tópico da análise os demais pormenores.

Primeiro observe que os *datasets* possuem exclusivamente a informação do país de origem/destino e não de procedência/aquisição. Não é o objetivo desse trabalho explicar o significado dessas definições, mas simplificadamente podemos dizer que: origem é onde o produto foi fabricado (ou sofreu ou última modificação substancial), procedência é de onde ele veio e aquisição é onde foi comprado. Sendo assim, durante a análise, verificamos que há operações com valor “Brasil” na coluna “NO_PAIS”. Isso pode parecer estranho, mas dada a explicação anterior é compreensível e esperado. Por exemplo, toda reimportação de mercadorias em que

não haja alteração substancial da mercadoria terá como dado de origem o Brasil, por mais que o país de procedência (exportador) seja outro. Sendo assim, iremos criar um segundo *dataframe* sem o Brasil para determinadas análises, mantendo o *dataframe* original para outras análises. Isso é feito com o código a seguir:

```
#Retirando as linhas que possuem como país de origem/destino Brasil e criando um novo dataframe
df_exp_full_2=df_exp_full[df_exp_full['NO_PAIS']!='Brasil']
df_imp_full_2=df_imp_full[df_imp_full['NO_PAIS']!='Brasil']
```

Em outro momento da análise dos dados, quando utilizamos as descrições das mercadorias para produzir nuvens de palavras, precisamos realizar os tratamentos das colunas, retirando pontuações, dígitos e palavras com pouco significado. Não colocaremos essa parte do código aqui (por ser demasiadamente extensa) e deixaremos para apresentar no momento da análise dos dados.

Quando da análise dos modais de transporte, verificou-se que há uma grande quantidade de operações em que a via de transporte não foi declarada, principalmente nos dados de exportação. Isso se deve principalmente a alterações nos sistemas, em especial a implantação do Portal Único de Comércio Exterior - Siscomex e a implantação da DUE, Declaração Única de Exportação. Diversos outros sistemas, em especial de presença e armazenamento de carga, não estão, pelo que tudo indica, suficientemente integrados aos novos sistemas. Sendo assim, não há como, no momento, recuperar a real situação sem um reprocessamento dos dados, que poderá ser feito posteriormente, como feito em 2019 (<http://editor.economia.gov.br/Economia/noticias/2019/06/secretaria-de-comercio-exterior-atualiza-estatisticas-de-2018>). Por esse motivo, foram criados ainda outros *dataframes*, retirando-se as linhas com a informação de "VIA NAO DECLARADA".

```
#Criando um novo dataframe retirando os dados de VIA NAO DECLARADA
df_exp_full_3=df_exp_full[df_exp_full['NO_VIA']!='VIA NAO DECLARADA']
df_imp_full_3=df_imp_full[df_imp_full['NO_VIA']!='VIA NAO DECLARADA']
```

Finalmente, foi feito também um tratamento dos dados focado nos modelos de *Machine Learning*, de modo a aumentar as métricas de desempenho e selecionar os atributos mais preditivos do *dataset*. Isso é a chamada *Feature Engineering* (Engenharia de Variáveis), processo de extrair e selecionar as melhores características que possam ser utilizadas de forma efetiva em modelos preditivos. Isso será feito em um tópico dedicado (5. Criação de Modelos de Machine Learning) para manter a organização lógica do texto.

4. Análise e Exploração dos Dados

Finalmente atingimos o momento da análise dos dados. A sistemática adotada foi a de seguir, tanto no notebook quanto nesse documento, a divisão da análise em assuntos de interesse. Serão expostos os códigos comentados e as saídas obtidas e, eventualmente, quando se mostrar necessário, serão feitos comentários em relação ao código ou ao resultado obtido.

4.1. Análise dos Parceiros Comerciais do Brasil no ano de 2020

O Objetivo dessa análise é verificar os países dos quais o Brasil mais importa e para os quais mais exporta utilizando diversas métricas.

Primeiramente analisemos a quantidade de países importadores/exportadores de/para o Brasil no ano de 2020:

```
#Transformando a coluna em lista e verificando os valores únicos
lista_paises_exp=list(df_exp_full['NO_PAIS'].unique())
lista_paises_imp=list(df_imp_full['NO_PAIS'].unique())

#Imprimindo o tamanho das listas resultantes
print('Quantidade de países para os quais o Brasil exportou em 2020:')
display(len(lista_paises_exp))
print('Quantidade de países dos quais o Brasil importou em 2020:')
display(len(lista_paises_imp))

Quantidade de países para os quais o Brasil exportou em 2020:
244
Quantidade de países dos quais o Brasil importou em 2020:
236
```

Vemos que há uma diferença nas quantidades obtidas. Vejamos então os países para os quais o Brasil exportou mas não importou e os países dos quais o Brasil importou mas não exportou:

```
| #Definindo uma função que transforma a lista e set e realiza a diferença entre os sets
def Diff(li1, li2):
    return (list(list(set(li1)-set(li2)))))

| #Calculando a diferença entre as listas
print('Países para os quais o Brasil exportou mas não importou em 2020:')
display(Diff(lista_paises_exp, lista_paises_imp))

Países para os quais o Brasil exportou mas não importou em 2020:
['Guernsey',
 'Guam',
 'São Martinho, Ilha de (parte francesa)',
 'Mayotte',
 'São Bartolomeu',
 'Benin',
 'Martinica',
 'Groenlândia',
 'Norfolk, Ilha',
 'São Vicente e Granadinas',
 'Cook, Ilhas',
 'Mauritânia',
 'Marianas do Norte, Ilhas',
 'Sudão do Sul']
```

```
#Calculando a diferença entre as listas
print('Países dos quais o Brasil importou mas não exportou em 2020:')
display(Diff(lista_paises_imp, lista_paises_exp))
```

Países dos quais o Brasil importou mas não exportou em 2020:

```
['Svalbard e Jan Mayen',
 'Pitcairn',
 'Bouvet, Ilha',
 'Bancos Centrais',
 'Geórgia do Sul e Sandwich do Sul, Ilhas',
 'Cocos (Keeling), Ilhas']
```

Vejamos então os cinco maiores importadores e exportadores, em valor FOB:

```
#Agrupando e ordenando os dados
df_paises_exp = df_exp_full.groupby(['NO_PAIS'])['VL_FOB'].sum().sort_values(ascending=False).to_frame()
print('Importadores de mercadorias brasileiras em 2020 em valor US$:')
df_paises_exp
```

Importadores de mercadorias brasileiras em 2020 em valor US\$:

	VL_FOB
NO_PAIS	
China	67788071101
Estados Unidos	21481528307
Argentina	8488717954
Países Baixos (Holanda)	7382820316
Canadá	4229942341

```
#Agrupando e ordenando os dados
df_paises_imp = df_imp_full.groupby(['NO_PAIS'])['VL_FOB'].sum().sort_values(ascending=False).to_frame()
print('Exportadores de mercadorias para o Brasil em 2020 em valor US$:')
df_paises_imp
```

Exportadores de mercadorias para o Brasil em 2020 em valor US\$:

	VL_FOB
NO_PAIS	
China	34041250902
Estados Unidos	24122448007
Brasil	12620789898
Alemanha	8597645213
Argentina	7788098367

Observando os resultados anteriores vemos que um dos maiores exportadores de mercadorias para o Brasil é, estranhamente, o próprio Brasil. Já explicamos o motivo disso na etapa de tratamento dos dados (há apenas a informação de origem no *dataset*) e a necessidade de, para determinadas análises, retirar o Brasil do *dataframe*:

```
#Retirando as linhas que possuem como país de origem/destino Brasil e criando um novo dataframe
df_exp_full_2=df_exp_full[df_exp_full['NO_PAIS']!='Brasil']
df_imp_full_2=df_imp_full[df_imp_full['NO_PAIS']!='Brasil']

#Agrupando e ordenando os dados
df_paises_exp = df_exp_full_2.groupby(['NO_PAIS'])['VL_FOB'].sum().sort_values(ascending=False).to_frame()
df_paises_imp = df_imp_full_2.groupby(['NO_PAIS'])['VL_FOB'].sum().sort_values(ascending=False).to_frame()

print('Importadores de mercadorias brasileiras em 2020 em valor US$:')
display(df_paises_exp)
print('Exportadores de mercadorias para o Brasil em 2020 em valor US$:')
display(df_paises_imp)
```

Importadores de mercadorias brasileiras em 2020 em valor US\$:

	VL_FOB
NO_PAIS	
China	67788071101
Estados Unidos	21481528307
Argentina	8488717954
Países Baixos (Holanda)	7382820316
Canadá	4229942341

Exportadores de mercadorias para o Brasil em 2020 em valor US\$:

VL_FOB

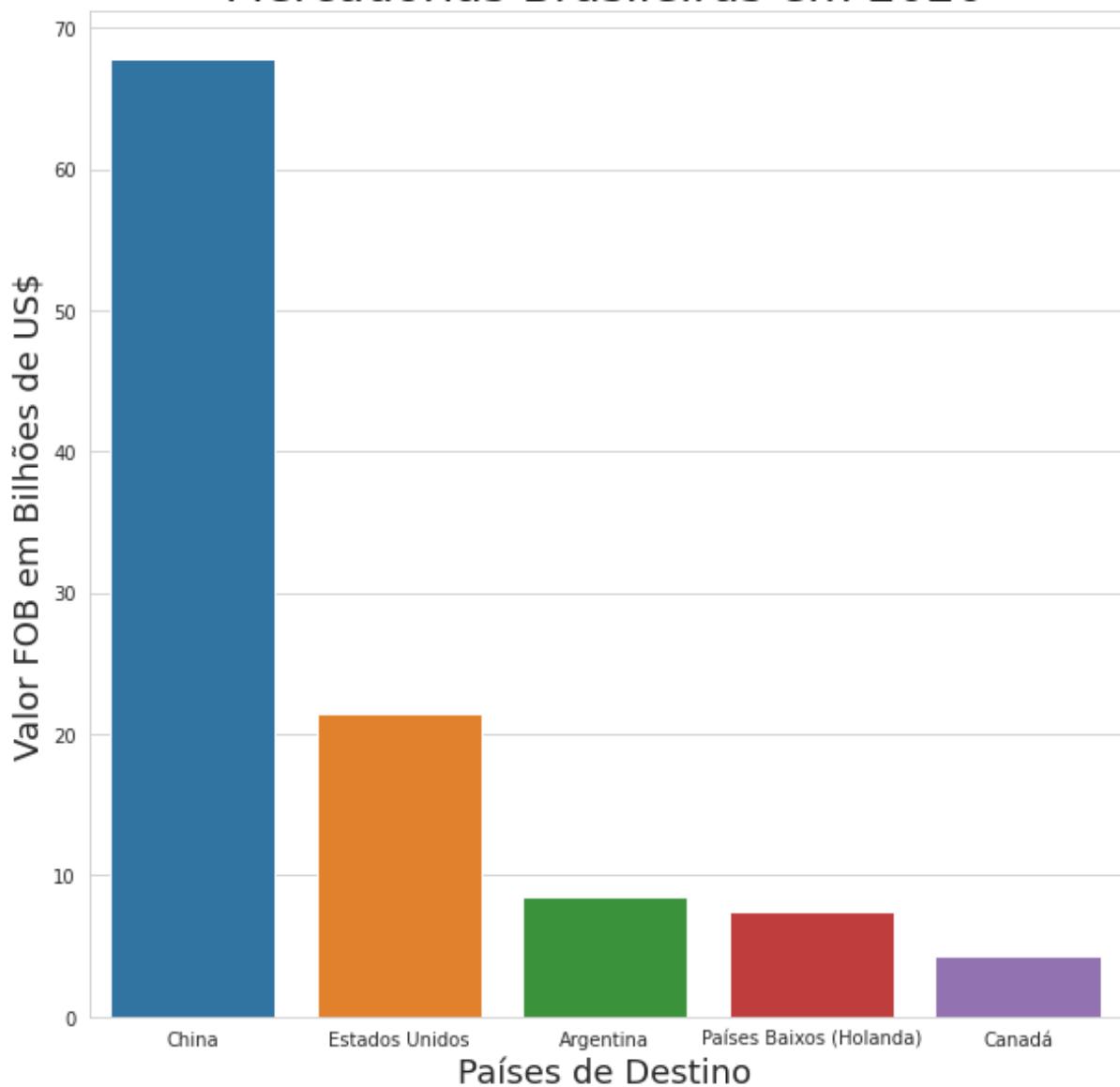
	NO_PAIS
China	34041250902
Estados Unidos	24122448007
Alemanha	8597645213
Argentina	7788098367
Coreia do Sul	4087766930

Para facilitar a visualização podemos desenhar gráficos. Nesse caso, podemos usar o gráfico de barras, um gráfico com barras retangulares e comprimento proporcional aos valores que ele apresenta. Para isso usaremos as bibliotecas *matplotlib* e *seaborn*:

```
#Selecionando os maiores por valor FOB
df_maiores_exp=df_paises_exp.nlargest(5,['VL_FOB'])
#Convertendo de dólares para bilhões de dólares
df_maiores_exp['VL_FOB']=df_maiores_exp['VL_FOB']/1000000000

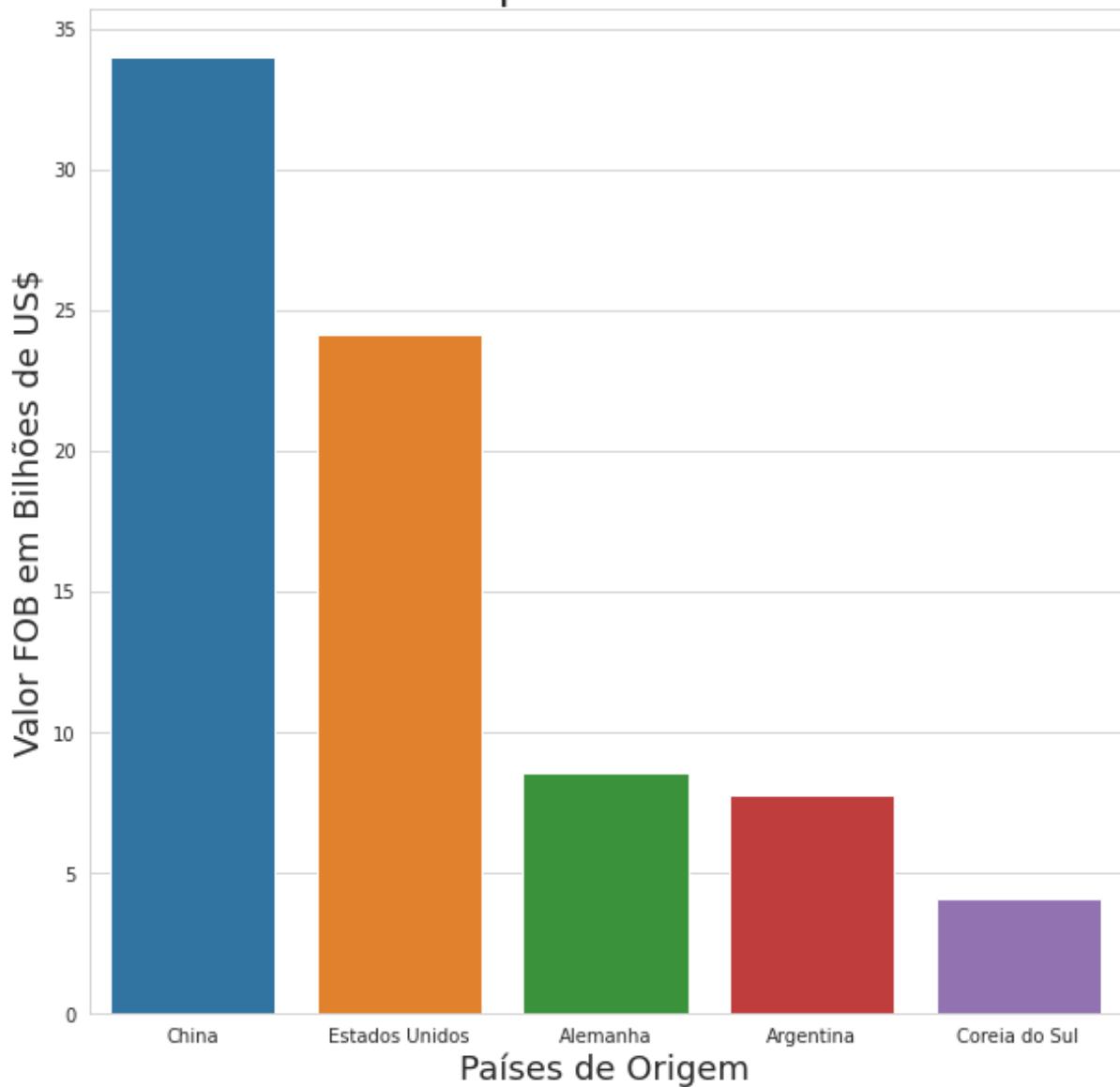
#Plotando os gráficos
plt.figure(figsize=(10,10))
sns.set_style('whitegrid')
bar_maiores_exp=sns.barplot(x=df_maiores_exp.index,y=df_maiores_exp['VL_FOB'])
plt.xlabel('Países de Destino',size=18)
plt.ylabel('Valor FOB em Bilhões de US$',size=18)
plt.title('5 Maiores Importadores de\nMercadorias Brasileiras em 2020', fontdict = {'fontsize': 25})
```

5 Maiores Importadores de Mercadorias Brasileiras em 2020



Podemos fazer o mesmo para obter os maiores exportadores para ao Brasil:

5 Maiores Exportadores de Mercadorias para o Brasil em 2020



Uma outra análise interessante que pode ser realizada é da proporção em valor dos cinco maiores parceiros em cada tipo de operação em relação ao valor total de operações realizadas. Para isso, podemos usar um gráfico de pizza, um diagrama circular onde os valores de cada categoria estatística representada são proporcionais às respectivas frequências (nesse caso, valor). Vejamos o código e o resultado gráfico dessa análise.

```

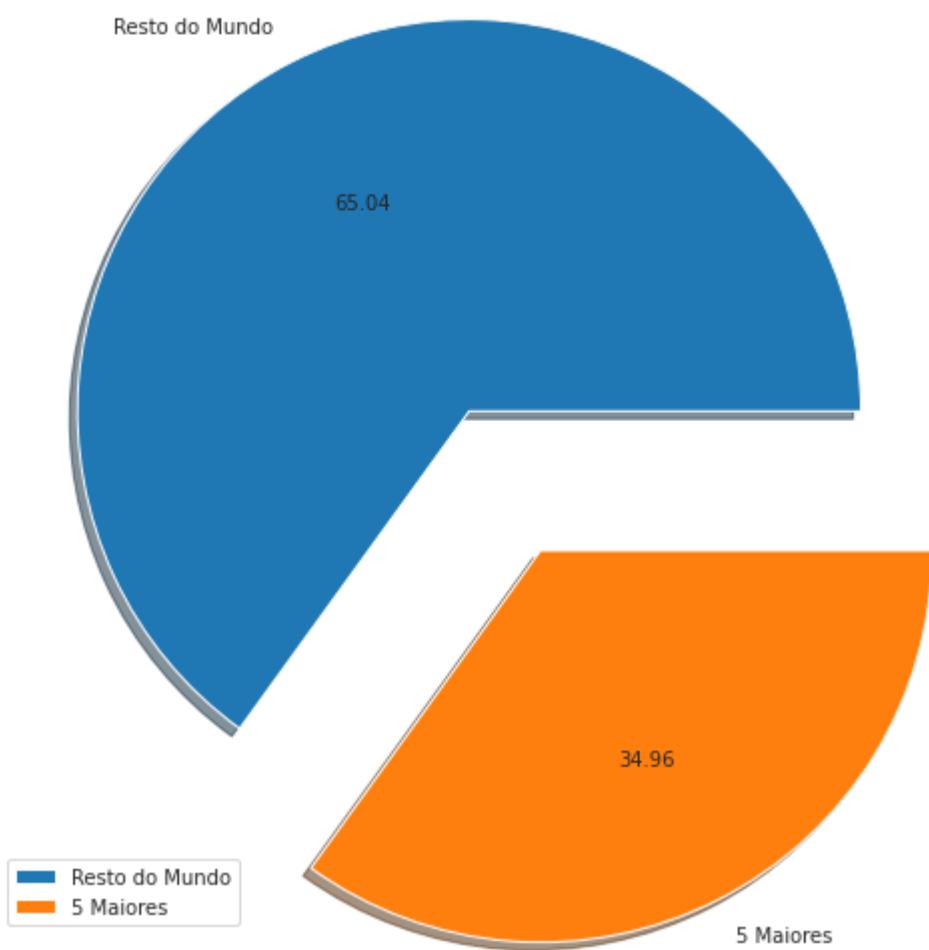
#Somando o valor FOB dos 5 maiores
soma_top_5_exp=df_maiores_exp['VL_FOB'].sum()
#Somando o valor FOB total excluindo os 5 maiores
soma_resto_exp = df_exp_full_2.sort_values('VL_FOB',ascending=False)[5:]['VL_FOB'].sum()/1000000000 # convertendo dólares para bilhões de dólares

#Plotando o Gráfico
labels=['Resto do Mundo','5 Maiores']
#Definindo os tamanhos das fatias com base nas somas
sizes=[soma_resto_exp,soma_top_5_exp]

explode = [ 0.1, 0.3]
plt.rcParams['figure.figsize'] = (9, 9)
plt.pie(sizes, labels = labels, explode = explode, shadow = True, autopct='%1.2f')
plt.title('5 Maiores Importadores de Mercadorias\nBrasileiras x Resto do Mundo', fontsize = 25)
plt.legend()
plt.show()

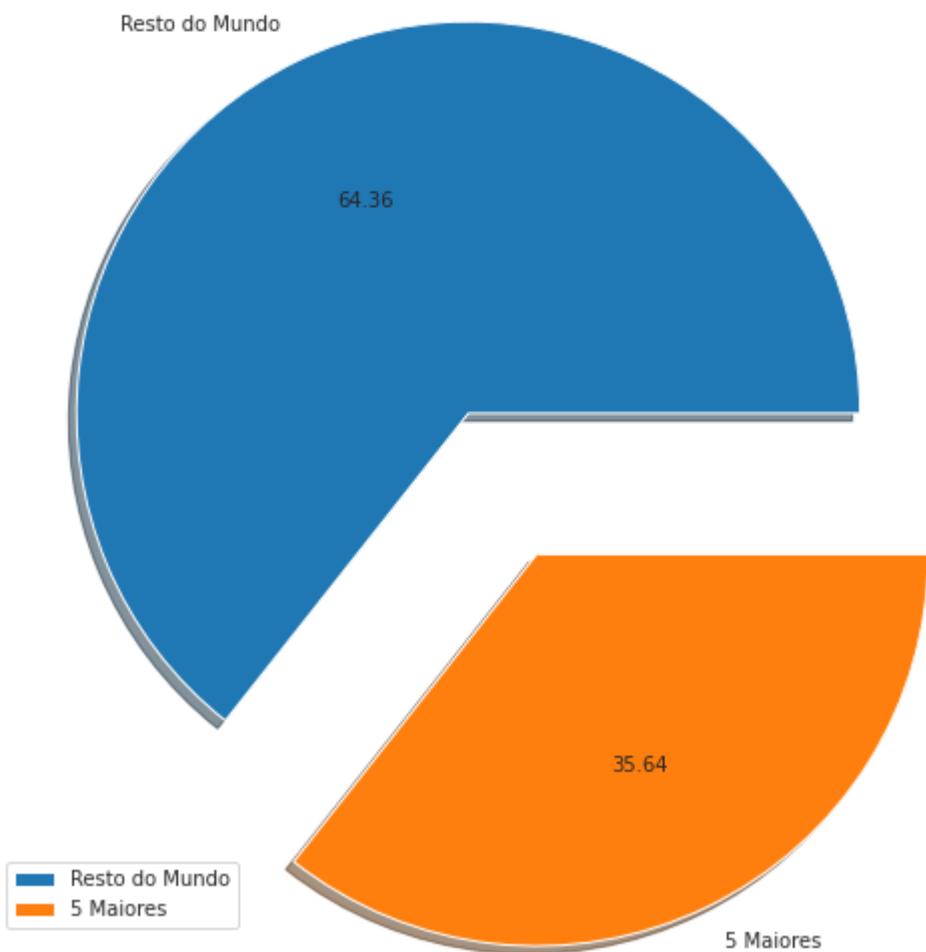
```

5 Maiores Importadores de Mercadorias Brasileiras x Resto do Mundo em valor



Fazendo o mesmo com o outro *dataframe*:

5 Maiores Exportadores de Mercadorias para o Brasil x Resto do Mundo em valor



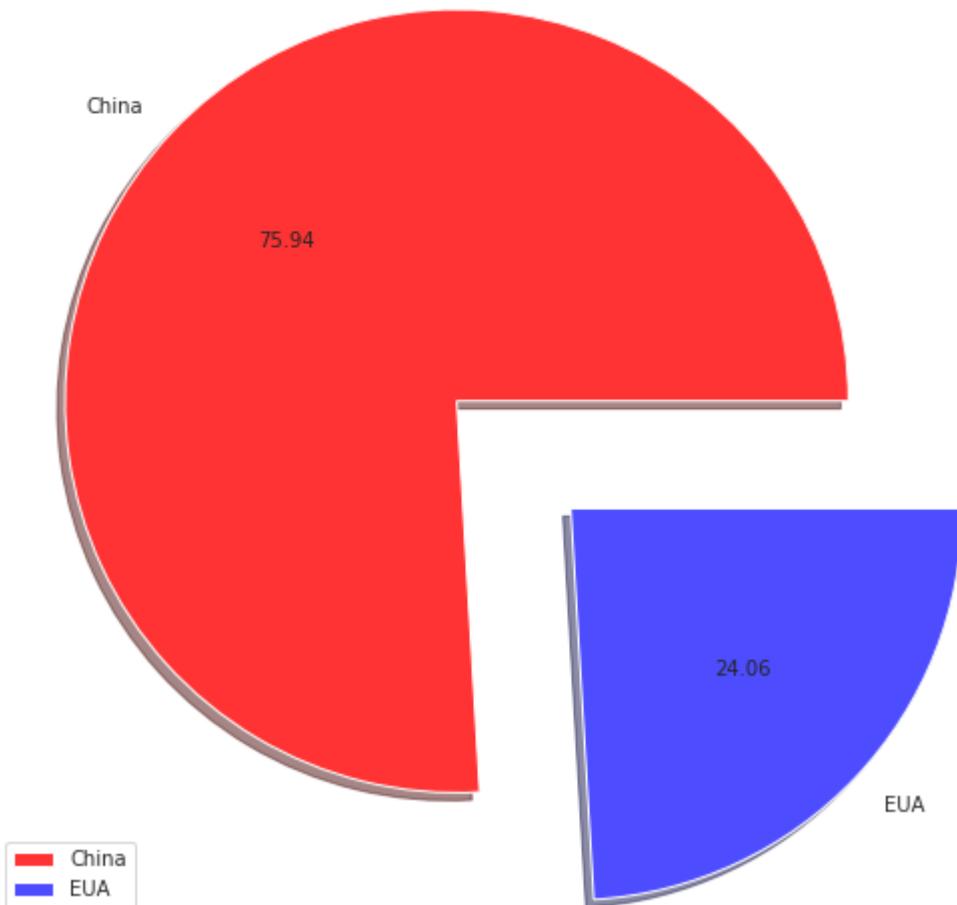
Observa-se a grande importância dos cinco maiores parceiros em cada tipo de operação em relação ao total, correspondendo eles por aproximadamente 65% do total de operações.

Voltando e verificando o gráfico dos maiores parceiros comerciais do Brasil, destacam-se China e Estados Unidos no contexto do comércio internacional brasileiro. Podemos então criar comparativos entre as estatísticas de ambos. Façamos então outro gráfico de pizza, dessa vez verificando a proporção de cada país no total das operações do Brasil com ambos países.

```
#Plotando o Gráfico
labels=['China','EUA']
colors = ['#ff3333','#4d4dff']
#Definindo o tamanho das fatias selecionando apenas as linhas com os dados dos dois países
sizes=[df_maiores_exp.loc['China','VL_FOB'],df_maiores_exp.loc['Estados Unidos','VL_FOB']]

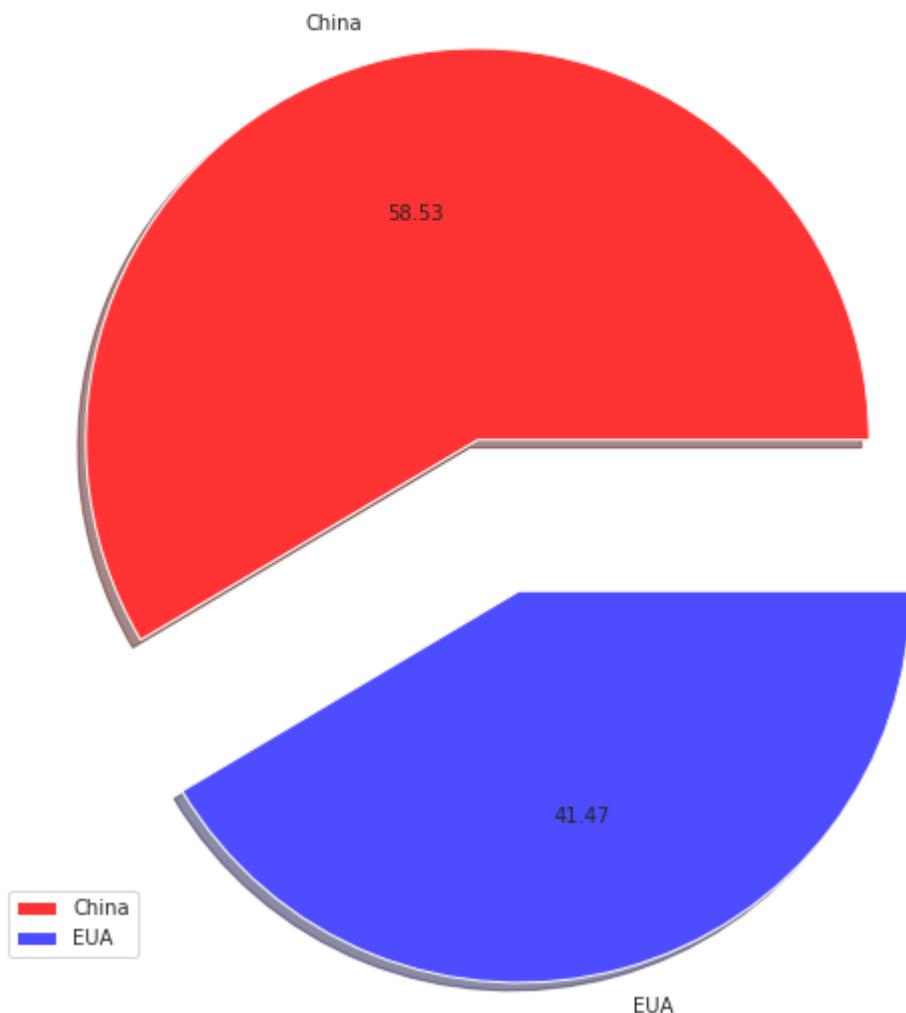
explode = [ 0.1, 0.3]
plt.rcParams['figure.figsize'] = (9, 9)
plt.pie(sizes, labels = labels, colors = colors, explode = explode, shadow = True, autopct='%.2f')
plt.title('Importações de Mercadorias Brasileiras:\nChina x EUA', fontsize = 25)
plt.legend()
plt.show()
```

Importações de Mercadorias Brasileiras: China x EUA



Realizando os cálculos equivalentes no outro *dataframe*:

Exportações de Mercadorias para o Brasil: China x EUA



Podemos então observar que a China é quem mais importa do Brasil e quem mais exporta para o Brasil, em valores totais. A China é, reconhecidamente, o principal e maior parceiro comercial do Brasil em termos de valor das operações.

4.2. Análise da Balança Comercial

Vamos agora analisar a Balança Comercial do país. Balança Comercial é um termo econômico correspondente a diferença entre as exportações e importações, refletindo a situação econômica de um país. Se o valor for positivo então o saldo é favorável e se for negativo é desfavorável.

```
#Somando os valores e convertendo em bilhões de dólares
positivo = df_exp_full['VL_FOB'].sum()/10000000000
negativo = df_imp_full['VL_FOB'].sum()/10000000000
#Calculando o saldo
saldo = (positivo-negativo)

print("Positivo: ")
display(positivo)
print("Negativo: ")
display(negativo)
print("Saldo: ")
display(saldo)

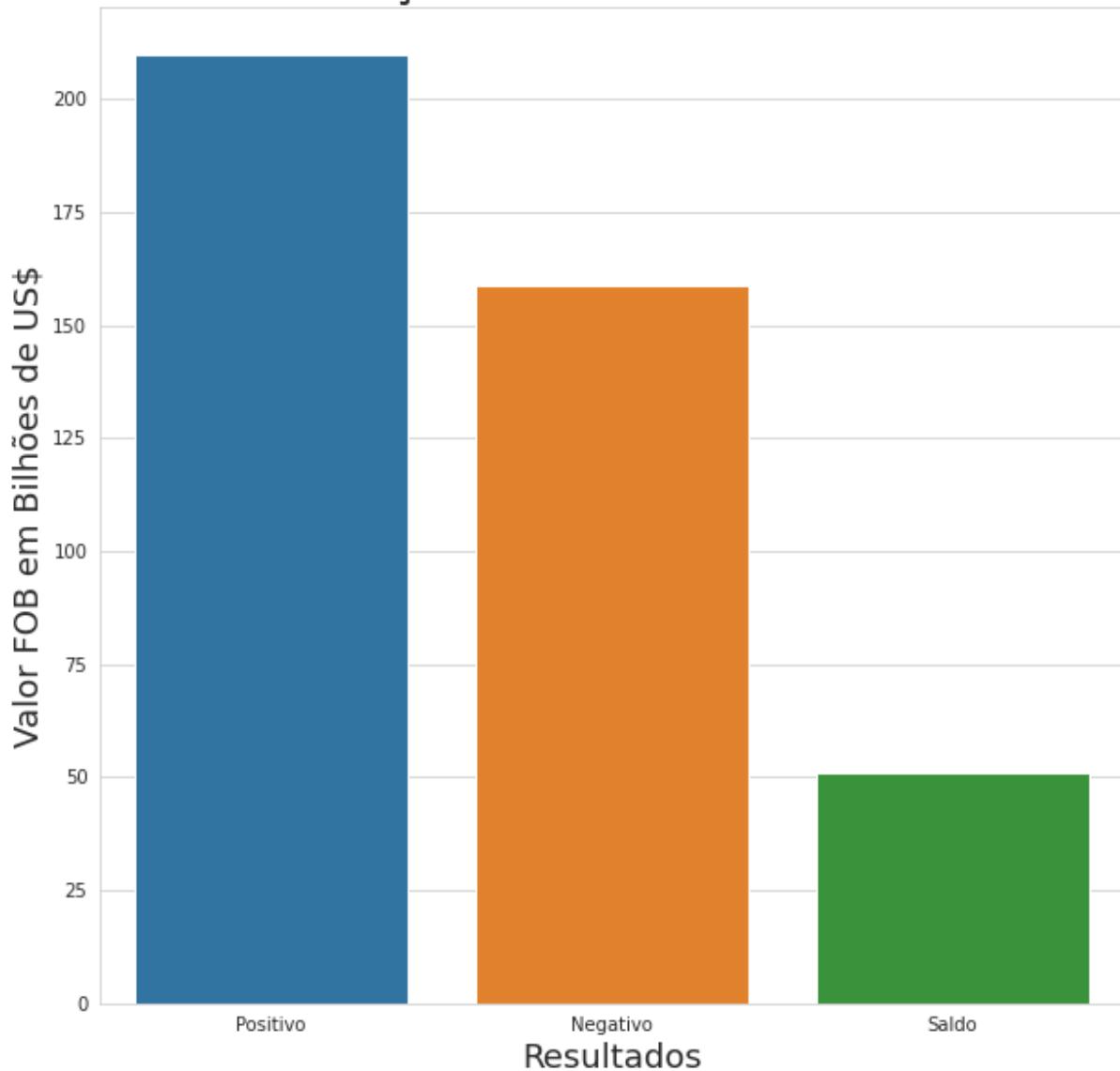
Positivo:
209.878384964
Negativo:
158.937295209
Saldo:
50.94108975499998
```

Esses valores vão ao encontro do divulgado pelo Governo Federal (<https://www.gov.br/pt-br/noticias/financas-impostos-e-gestao-publica/2021/01/balanca-comercial-fecha-2020-com-superavit-de-us-50-9-bilhoes>). O saldo positivo da balança comercial brasileira em 2020 é resultado de US\$ 209,9 bilhões em exportações e US\$ 158,9 bilhões em importações com superávit de US\$ 50,9 bilhões. Note que estamos usando o *dataset* original nessa análise e não o *dataset* em que retiramos as operações em que constava "Brasil" na coluna "NO_PAIS". Isso deve ser feito pois, diferentemente da análise anterior, nesse caso estamos interessados nos valores recebidos pelas exportações e nos valores pagos pelas importações. E nisso nada importa a origem da mercadoria, e sim e somente se houve entrada ou saída de valores. O país pode, por exemplo, pagar por uma mercadoria de procedência estrangeira, mas de origem brasileira e com isso diminuir o saldo da balança comercial.

Plotando em gráfico temos o seguinte:

```
#Plotando o gráfico
plt.figure(figsize=(10,10))
sns.set_style('whitegrid')
sns.barplot(['Positivo', 'Negativo', 'Saldo'], [positivo, negativo, saldo])
plt.xlabel('Resultados',size=18)
plt.ylabel('Valor FOB em Bilhões de US$',size=18)
plt.title('Balança Comercial Brasil 2020', fontdict = {'fontsize': 25})
```

Balança Comercial Brasil 2020



Voltemos agora, mais uma vez, à análise dos parceiros comerciais do Brasil. Sabemos que a China é o maior importador de produtos brasileiros e o maior exportador de produtos para o Brasil, mas não sabemos o saldo disso. Os saldos dessas operações podem ser tanto negativos quanto positivos. O mesmo pode ser dito dos EUA.

```
#Calculando os saldos selecionando as linhas dos dois países
positivo_ch = df_maiores_exp.loc['China','VL_FOB']
negativo_ch = df_maiores_imp.loc['China','VL_FOB']
saldo_ch = (positivo_ch-negativo_ch)

positivo_eua = df_maiores_exp.loc['Estados Unidos','VL_FOB']
negativo_eua = df_maiores_imp.loc['Estados Unidos','VL_FOB']
saldo_eua = (positivo_eua-negativo_eua)

print("Positivo China: ")
display(positivo_ch)

print("Negativo China: ")
display(negativo_ch)

print("Saldo China: ")
display(saldo_ch)

print("Positivo EUA: ")
display(positivo_eua)

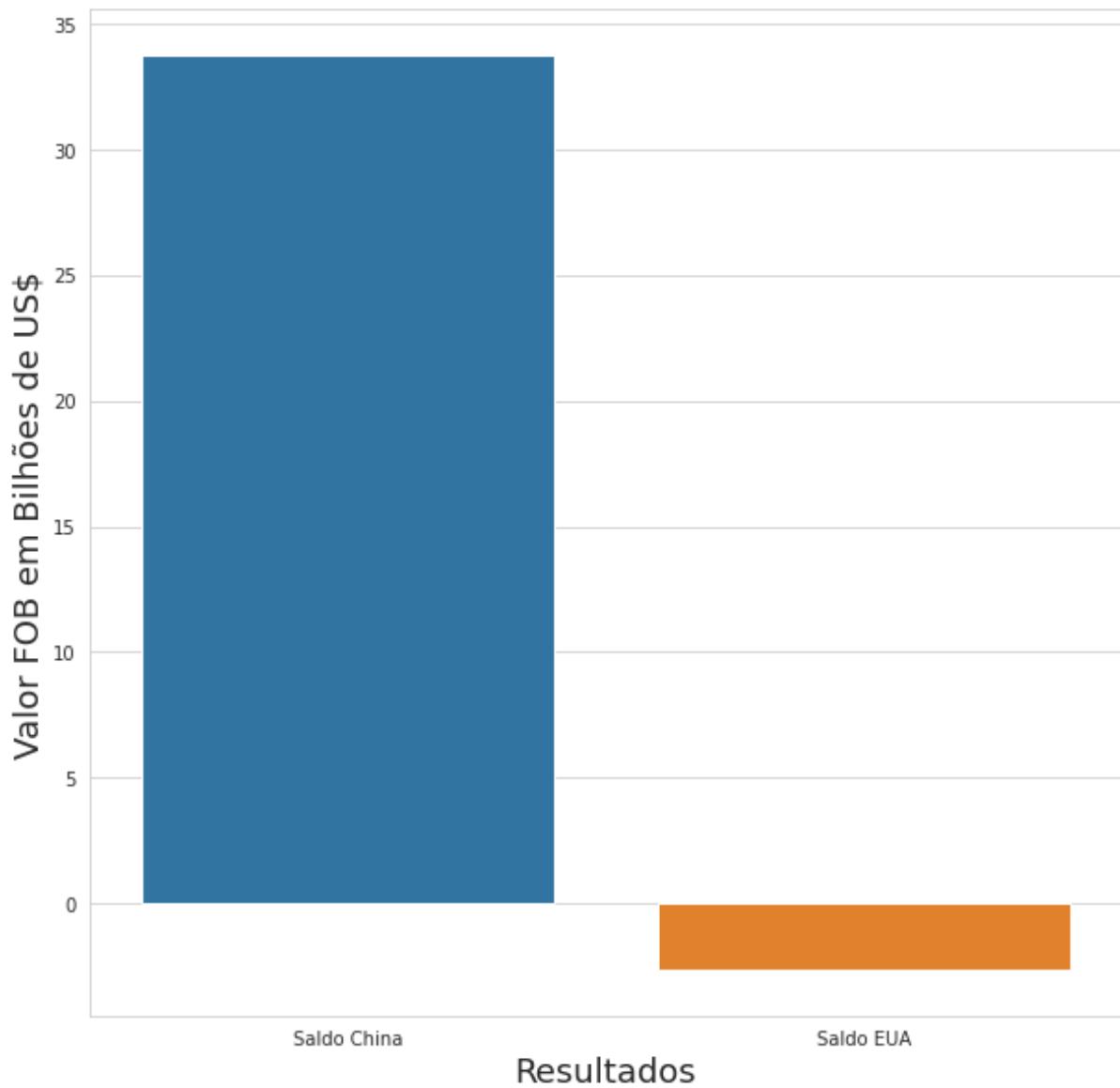
print("Negativo EUA: ")
display(negativo_eua)

print("Saldo EUA: ")
display(saldo_eua)
```

Positivo China:
67.788071101
Negativo China:
34.041250902
Saldo China:
33.746820199
Positivo EUA:
21.481528307
Negativo EUA:
24.122448007
Saldo EUA:
-2.6409196999999978

```
#Plotando o gráfico
plt.figure(figsize=(10,10))
sns.set_style('whitegrid')
sns.barplot(['Saldo China', 'Saldo EUA'], [saldo_ch, saldo_eua])
plt.xlabel('Resultados',size=18)
plt.ylabel('Valor FOB em Bilhões de US$',size=18)
plt.title('Resultados das Operações do Brasil\ncom China e EUA 2020', fontdict = {'fontsize': 25})
```

Resultados das Operações do Brasil com China e EUA 2020



O resultado é claro. Nossa saldo com a China é superavitário e com os EUA é deficitário. Só com isso podemos notar a gigantesca importância da China para o superávit comercial brasileiro e a importância das nossas relações com o país. Isso vai ao encontro dos dados divulgados pelo governo brasileiro (<https://www.legisweb.com.br/noticia/?id=25074>): “A forte resiliência das exportações brasileiras foi em muito influenciada pelo ritmo de recuperação da região asiática, sobretudo a China. Esse padrão difere muito do observado no resto do mundo, onde o volume exportado foi mais duramente atingido que no caso brasileiro e a recuperação ocorre mais claramente a partir de maio de 2020”.

4.3. Análise dos Tipos das Mercadorias

Uma nuvem de palavras, também conhecida como nuvem de *tags*, é uma visualização simples de dados em que as palavras são mostradas em tamanhos variados, dependendo da frequência com que aparecem no texto.

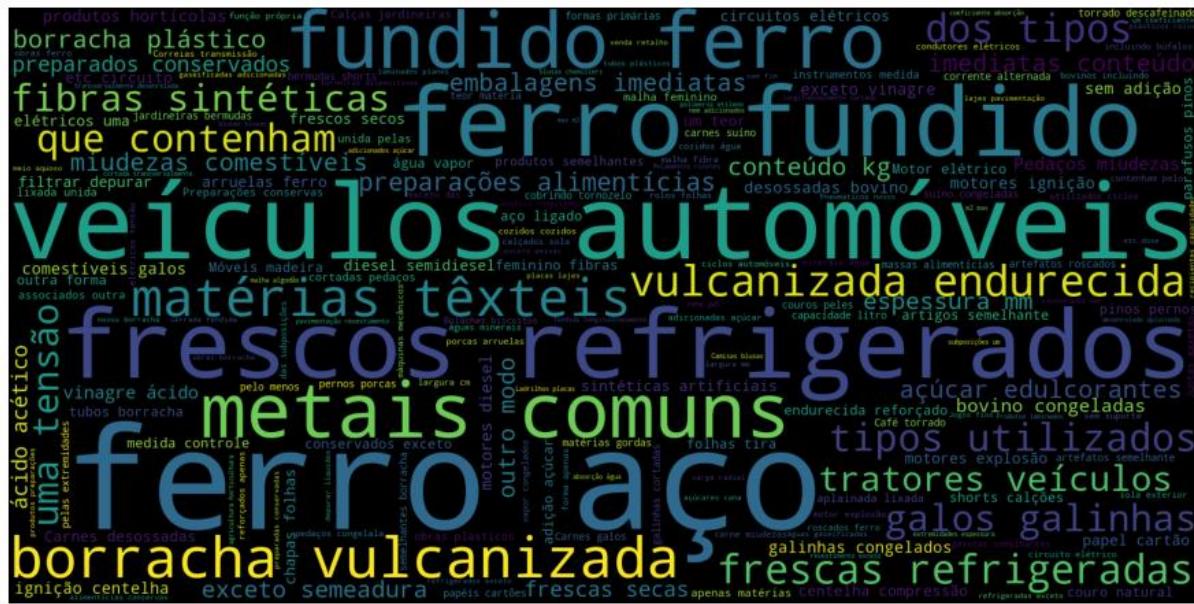
Para construir uma, utilizaremos a biblioteca *WordCloud* do Python. Inicialmente, sempre quando trabalhamos com textos, é comum fazermos um pré-processamento com remoção de dígitos, pontuações e *stopwords*. *Stopwords* são palavras ou artigos muito usados, como “a”, “de”, “o”, “da”, “para” etc. que não ajudam o modelo e não fornecem informações úteis.

```
#Tratamento dos dados
#Adicionando StopWords Comuns nas descrições das NCMs que em nada agregam na análise
stopwords = set(STOPWORDS)
#Adicionando novas palavras a serem excluídas. Etapa iterativa.
stopwords.update(["da", "meu", "em", "você", "de", "do", "ao", "os", "ou", "seus", "suas", "e", "por", "para", "peso", "acessórios",
    "aparelhos", "superior", "inferior", "igual", "outras", "outros", "partes", "misturas", "tipo", "uso", 'de', 'mesmo',
    'seu', 'não', 'para', 'posição', 'é', 'peso', 'quais', 'devem', 'quais', 'constituir', 'principal', 'acessória'])
```

Agora podemos criar as nuvens de palavras:

```
#Pegando a descrição do item/subitem, ou seja, ao nível de adição da declaração
coluna_ncm_exp = df_exp_full['NO_NCM_POR']
#Juntando todas as descrições
all_summary_exp = " ".join(s for s in coluna_ncm_exp)
#Criando a nuvem
wordcloud_exp = WordCloud(stopwords=stopwords,
                           background_color='black', width=1600,
                           height=800).generate(all_summary_exp)

fig, ax = plt.subplots(figsize=(16,8))
ax.imshow(wordcloud_exp, interpolation='bilinear')
ax.set_axis_off()
plt.imshow(wordcloud_exp)
```



Fazendo o mesmo para a importação:



Vemos aqui a predominância, principalmente na importação, de palavras como máquina, ferro, aço e plástico. Isso é esperado. A nuvem de palavras foi construída com base na frequência desses termos nas descrições das adições das mercadorias, muito comuns nas inúmeras NCMs (a nível de subitem da classificação) de partes e peças de máquinas. Em outras palavras, as posições do Sistema Harmonizado que tratam de mercadorias mais elaboradas, como máquinas, possuem mais itens e subitens, subdivisões, que as posições de mercadorias mais simples, como animais e vegetais. Além disso, essas descrições costumam ser mais longas e repetir termos. Não é isso que buscamos nessa análise.

Buscamos uma nuvem de palavras que use como dado da frequência o valor das exportações/importações: as maiores palavras devem as mercadorias com maior valor das operações. Para corrigir isso, precisamos fazer algumas alterações nos *dataframes* (como adiantamos no momento da explicação do tratamento dos dados). Também iremos, além disso, agrupar as mercadorias por posições (e não itens e subitens) para evitar o efeito negativo anteriormente descrito das descrições longas e do grande número de subdivisões nas posições das mercadorias mais elaboradas.

Primeiro vamos realizar o pré-processamento, retirando dígitos, pontuações e agrupando pelo valor FOB:

```
#Definindo uma função para retirar a pontuação dos textos. Isso é necessário pois faremos manualmente o cálculo das frequências
#Definindo todas pontuações, menos o hífen para não quebrar palavras compostas. Além disso, estamos retirando os dígitos
punct = '!"#$%&\`()*+,:;=>?@[\\]^_`{}`~1234567890'

def remove_pontuacao(df, coluna):
    transtab = str.maketrans(dict.fromkeys(punct, '')) 
    df[coluna] = '|'.join(df[coluna].tolist()).translate(transtab).split('|')

] #Agrupando as descrições a nível de posição da NCM e somando o valor FOB
df_exp_words_fob = df_exp_full.groupby(['NO_SH4_POR'])['VL_FOB'].sum().sort_values(ascending=False).to_frame().reset_index()
remove_pontuacao(df_exp_words_fob, 'NO_SH4_POR')
df_exp_words_fob
```

	NO_SH4_POR	VL_FOB
0	Soja mesmo triturada	28564148360
1	Minérios de ferro e seus concentrados incluída...	25789230630
2	Óleos brutos de petróleo ou de minerais betumi...	19613857946
3	Açúcares de cana ou de beterraba e sacarose qu...	8744182963
4	Carnes de animais da espécie bovina congeladas	6679112677
...
1174	Escórias exceto escória de altos-fornos granul...	66
1175	Fibras artificiais descontínuas não cardadas n...	65
1176	Automotoras mesmo para circulação urbana excet...	54
1177	Esboços de chapéus entrançados ou obtidos por ...	48
1178	Lã de madeira farinha de madeira	10

1179 rows × 2 columns

Fazendo o mesmo para a importação:

```
#Agrupando as descrições a nível de posição da NCM e somando o valor FOB
df_imp_words_fob = df_imp_full.groupby(['NO_SH4_POR'])['VL_FOB'].sum().sort_values(ascending=False).to_frame().reset_index()
remove_pontuacao(df_imp_words_fob, 'NO_SH4_POR')
df_imp_words_fob
```

	NO_SH4_POR	VL_FOB
0	Barcos-faróis barcos-bombas dragas guindastes ...	9996860982
1	Óleos de petróleo ou de minerais betuminosos e...	7391434003
2	Aparelhos elétricos para telefonia ou telegraf...	4441757673
3	Circuitos integrados e microconjuntos electrón...	3979809123
4	Tubos flexíveis de metais comuns mesmo com ace...	3839429143
...
1178	Mates de níquel sinters de óxidos de níquel e ...	324
1179	Locomotivas e locotractores de fonte externa d...	270
1180	Mates de cobre cobre de cementação precipitado...	232
1181	Gás de hulha gás de água gás pobre gás de ar e...	96
1182	Metais comuns ou prata folheados ou chapeados ...	26

1183 rows × 2 columns

Agora criamos uma função para separar as linhas e contar as frequências por palavras, obtendo novos resultados:

```
#Definindo um função para realizar o corte das strings das descrições das NCMS e duplicar as linhas
#conforme as frequências do valor FOB
def corta_string(df, column, sep='|', keep=False):
    indexes = list()
    new_values = list()
    df = df.dropna(subset=[column])
    for i, presplit in enumerate(df[column].astype(str)):
        values = presplit.split(sep)
        if keep and len(values) > 1:
            indexes.append(i)
            new_values.append(presplit)
        for value in values:
            indexes.append(i)
            new_values.append(value)
    new_df = df.iloc[indexes, :].copy()
    new_df[column] = new_values
    return new_df
```

Também definimos as *stopwords*, como nas nuvens anteriores. Contudo, como estamos gerando as frequências manualmente em razão do valor FOB, o atributo *stopwords* é ignorado automaticamente pelo método e todo processamento deve ser feito manualmente. Desta forma, os *dataframes* foram alterados, removendo a mesma lista anterior.

```
#Criando um novo dataframe com as palavras cortadas
df_exp_words_fob_2 = corta_string(df_exp_words_fob, 'NO_SH4_POR', sep=' ').reset_index(drop=True)
df_imp_words_fob_2 = corta_string(df_imp_words_fob, 'NO_SH4_POR', sep=' ').reset_index(drop=True)

#Retirando manualmente as StopWords
df_exp_words_fob_2 = df_exp_words_fob_2[~df_exp_words_fob_2.NO_SH4_POR.isin(stopwords)]
df_imp_words_fob_2 = df_imp_words_fob_2[~df_imp_words_fob_2.NO_SH4_POR.isin(stopwords)]

#Exibindo os resultados
display(df_exp_words_fob_2)
display(df_imp_words_fob_2)
```

	NO_SH4_POR	VL_FOB
0	Soja	28564148360
2	triturada	28564148360
3	Minérios	25789230630
5	ferro	25789230630
8	concentrados	25789230630

	NO_SH4_POR	VL_FOB
0	Barcos-faróis	9996860982
1	barcos-bombas	9996860982
2	dragas	9996860982
3	guindastes	9996860982
4	flutuantes	9996860982

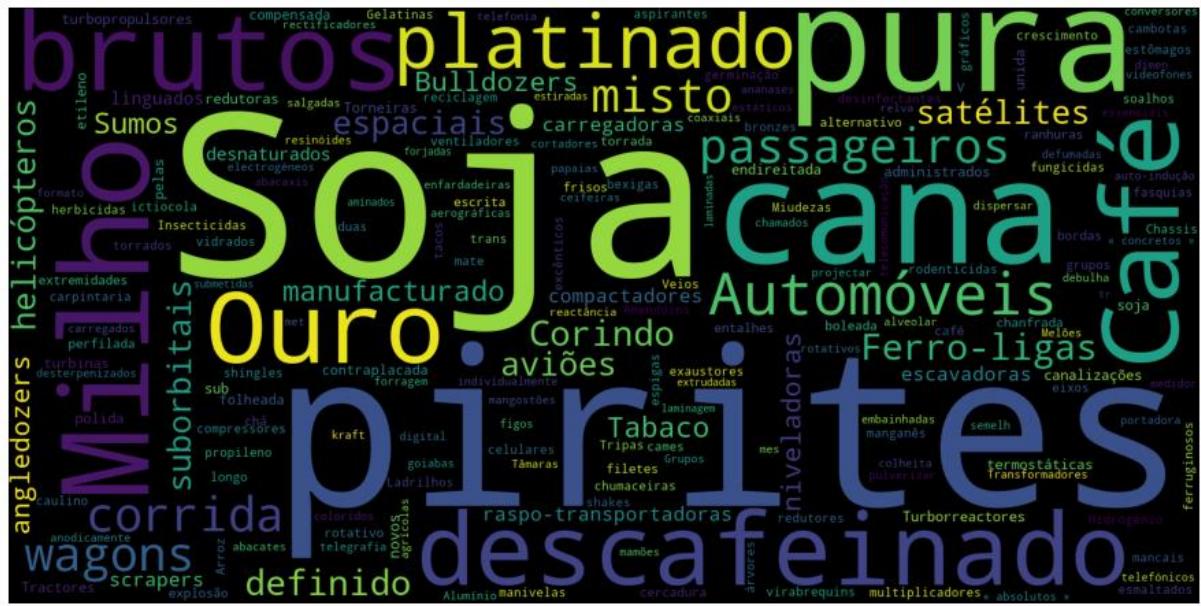
Criando agora as nuvens:

```
data_exp = dict(zip(df_exp_words_fob_2['NO_SH4_POR'].tolist(), df_exp_words_fob_2['VL_FOB'].tolist()))

wordcloud_exp = WordCloud(collocations=False,
                           background_color='black', width=1600,
                           height=800).generate_from_frequencies(data_exp)

fig, ax = plt.subplots(figsize=(16,8))
ax.imshow(wordcloud_exp, interpolation='bilinear')
ax.set_axis_off()

plt.imshow(wordcloud_exp)
```



Repetindo o procedimento para a importação obtemos:



Esse resultado é coerente com o obtido nas análises anteriores. Vemos que, na exportação, a soja é nosso maior produto. O Brasil é o segundo maior produtor de soja do mundo e a China é o seu maior comprador, conforme dados da mídia (<https://www.suinoculturainustrial.com.br/imprensa/importacoes-de-soja-brasileira-pela-china-disparam-51-em-setembro/20201027-085855-y822>), tendo a exportação sido especialmente estimulada em 2020 pela recomposição do rebanho suíno chinês, após ter sido dizimado pela peste suína africana.

Em suma, verificamos a predominância das palavras soja, milho, café, cana e pirites na exportação e das palavras barcos, dragas, guindastes e embarcações na

importação. Isso mostra como o Brasil exporta mais produtos básicos, commodities, e importa produtos de alto valor agregado, industrializados. Isso é um problema. Produtos manufaturados possuem maior valor se comparados a produtos agroindustriais. Isso é reflexo da nossa pobre industrialização e competitividade. A escassez de investimentos em ciência, tecnologia e industrialização mantém o país com sua pauta de exportação de bens básicos, que tendem a diminuir seus preços ao longo do tempo, e mantém a necessidade de importar bens de alto valor agregado que encarecem a cada dia. Isso torna cada vez mais difícil manter nossa balança comercial positiva de forma sustentável e deve servir de norte para a alteração de nossas políticas públicas com urgência.

4.4. Análise Temporal

Vamos agora analisar o valor das operações mês a mês durante o ano de 2020 e verificar qual o efeito da pandemia do novo coronavírus no contexto do comércio internacional. Utilizaremos o gráfico de linha, um tipo de gráfico que exibe informações com uma série de pontos de dados ligados por segmentos de linha reta.

```
#Agrupando por mês
df_tempo_exp = df_exp_full.groupby(['CO_MES'])['VL_FOB'].sum().to_frame().reset_index()
df_tempo_imp = df_imp_full.groupby(['CO_MES'])['VL_FOB'].sum().to_frame().reset_index()

#Plotando o Gráfico
f, ax = plt.subplots(1,1,figsize =(20,10))

ax.plot_date(df_tempo_exp['CO_MES'],y=df_tempo_exp['VL_FOB']/1000000000, color="blue", label="Exportação", linestyle="-", xdate = False)
ax.plot_date(df_tempo_imp['CO_MES'],y=df_tempo_imp['VL_FOB']/1000000000, color="red", label="Importação", linestyle="-", xdate = False)
ax.legend()

plt.xticks(df_tempo_exp['CO_MES'])
plt.xlabel('Mês',size=18)
plt.ylabel('Valor FOB em Bilhões de US$',size=18)
plt.title('Valor Exportações e Importações Brasil 2020', fontdict = {'fontsize': 25})
plt.show()
```



Durante o ano observa-se uma queda das importações (com uma recuperação no final) e um aumento das exportações. A queda das importações pode ser explicada, em parte, pelo impacto da pandemia na economia. Contudo, por minha experiência de trabalho com importação, posso afirmar que parte dessa queda das importações é sazonal e que a demanda das importações costuma ser sempre maior no final do ano. O aumento das exportações pode ser explicado, possivelmente, pela rápida recuperação da economia chinesa e sua demanda de soja para seus rebanhos suíños, entre outros.

Partamos agora para uma análise mais direta dos efeitos da pandemia e de um de seus produtos com maior demanda: máscaras de proteção. Faremos essa análise usando as NCMs 6307.90.10 e 6307.90.90 para as máscaras de proteção de tecido e falso tecido. Observe que essa análise não considera os casos de erro de classificação e nem poderia, pois o *dataset* utilizado não contém as descrições das mercadorias, apenas as classificações. Exportações que utilizaram classificação incorreta e saíram em canal verde de conferência não serão computadas. Além disso, essas classificações podem incluir outras mercadorias presentes na mesma classificação, pois o nível de detalhamento do *dataset* é insuficiente para separar esses casos. Contudo, como queremos observar as tendências de crescimento/diminuição das exportações pelo tempo, esse nível de detalhamento deve se mostrar suficiente.

```
#Definindo as NCMs buscadas
ncms_mask = ['63079010', '63079090']

#Criando um novo dataset com apenas as NCMs desejadas
df_mask_exp = df_exp_full.loc[df_exp_full['CO_NCM'].isin(ncms_mask)]
df_mask_imp = df_imp_full.loc[df_imp_full['CO_NCM'].isin(ncms_mask)]
#Agrupando por mês
df_mask_exp = df_mask_exp.groupby(['CO_MES'])['VL_FOB'].sum().to_frame().reset_index()
df_mask_imp = df_mask_imp.groupby(['CO_MES'])['VL_FOB'].sum().to_frame().reset_index()
#Exibindo os resultados
print('Exportação')
display(df_mask_exp)
print('Importação')
display(df_mask_imp)

#Plotando o Gráfico
f, ax = plt.subplots(1,1,figsize =(20,10))

ax.plot_date(df_mask_exp['CO_MES'],y=df_mask_exp['VL_FOB']/1000000, color="blue", label="Exportação", linestyle="-", xdate = False)

plt.xticks(df_tempo_exp['CO_MES'])
plt.xlabel('Mês',size=18)
plt.ylabel('Valor FOB em Milhões de US$',size=18)
plt.title('Valor Aproximado de Exportações de Máscaras de Proteção Brasil 2020', fontdict = {'fontsize': 25})
plt.show()
```



Verificamos um pico nas exportações de máscaras para o exterior no período de fevereiro e março, se mantendo baixa o resto do ano. Vamos então plotar um outro gráfico, apenas do primeiro semestre do ano, em conjunto com diversos marcos temporais importantes da pandemia para podermos estender nossa análise:

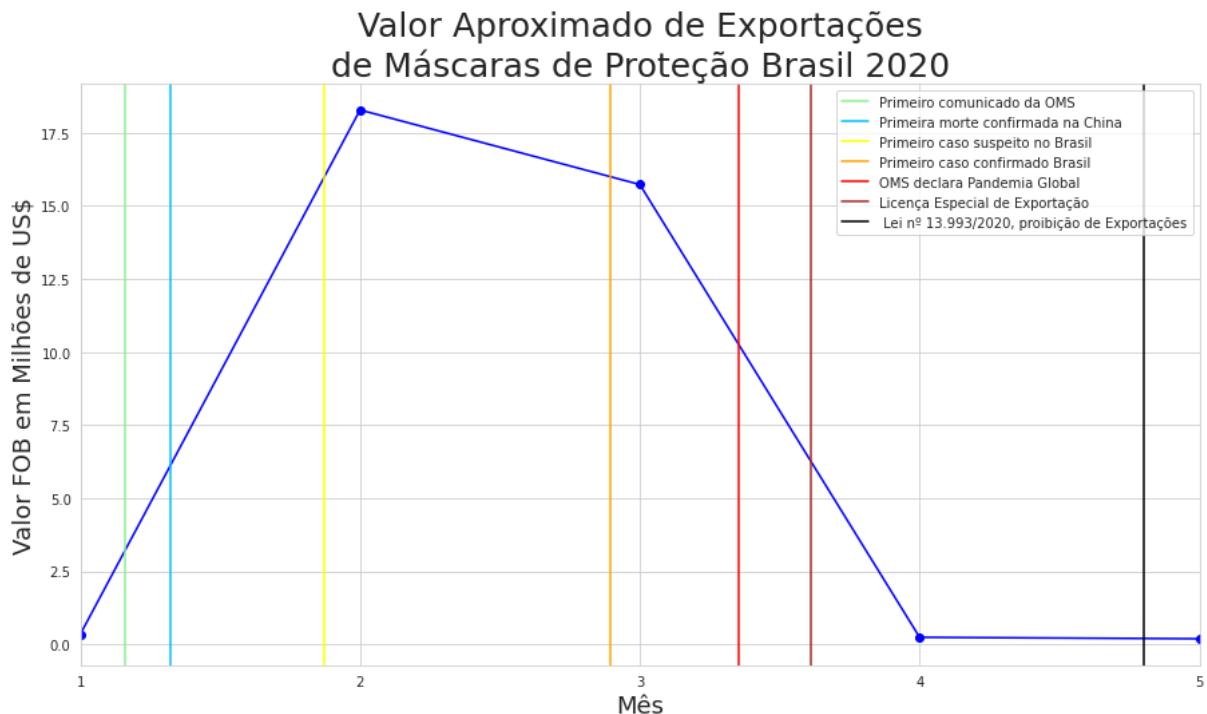
```
#Plotando o gráfico destacando datas importantes entre os meses de janeiro e maio
#Dados de:
#https://coronavirus.saude.gov.br/linha-do-tempo
#https://gauchazh.clicrbs.com.br/saude/noticia/2020/12/linha-do-tempo-veja-a-evolucao-da-covid-19-no-mundo-ao-completar-um-ano-ckjbv0iwx009o019w4kx1h0cd.html
#https://www.fazcomex.com.br/blog/proibicao-das-exportacoes-de-produtos-de-combate-a-covid-19/
#https://www.sanarmed.com/linha-do-tempo-do-coronavirus-no-brasil

f, ax = plt.subplots(1,1,figsize =(15,8))

#Definindo os dados
ax.plot_date(df_mask_exp['CO_MES'],y=df_mask_exp['VL_FOB']/1000000, color="blue", linestyle="-", xdate = False)

#Criando as linhas verticais
plt.axvline(1+5/31, label = 'Primeiro comunicado da OMS', color = 'lightgreen')
plt.axvline(1+10/31, label = 'Primeira morte confirmada na China', color = 'deepskyblue')
plt.axvline(1+27/31, label = 'Primeiro caso suspeito no Brasil', color = 'yellow')
plt.axvline(2+26/29, label = 'Primeiro caso confirmado Brasil', color = 'orange')
plt.axvline(3+11/31, label = 'OMS declara Pandemia Global', color = 'red')
plt.axvline(3+19/31, label = 'Licença Especial de Exportação', color = 'brown')
plt.axvline(4+24/30, label = 'Lei nº 13.993/2020, proibição de Exportações', color = 'black')

#Definindo o local das legendas
plt.legend(loc='upper right');
plt.xticks(df_tempo_exp['CO_MES'])
plt.xlabel('Mês',size=18)
plt.ylabel('Valor FOB em Milhões de US$',size=18)
plt.title('Valor Aproximado de Exportações de Máscaras de Proteção Brasil 2020', fontdict = {'fontsize': 25})
#Definindo o limite do eixo x
plt.xlim([1, 5])
plt.show()
```



O resultado é claro. Mesmo após o primeiro comunicado da OMS, a primeira morte na China, o primeiro caso suspeito e o primeiro caso confirmado no Brasil, as exportações de máscaras se mantiveram em valores elevados. O Brasil não se preparou devidamente, não entendeu a tempo a gravidade da pandemia e, hipnotizado pela alta demanda externa do produto, vendeu rapidamente sua produção de máscaras de proteção para o exterior.

Posteriormente, após a decretação de pandemia global pela OMS e a proibição de Exportações de máscaras no final de abril, já estávamos desabastecidos e tal medida foi, naquele ponto, infelizmente inócuia.

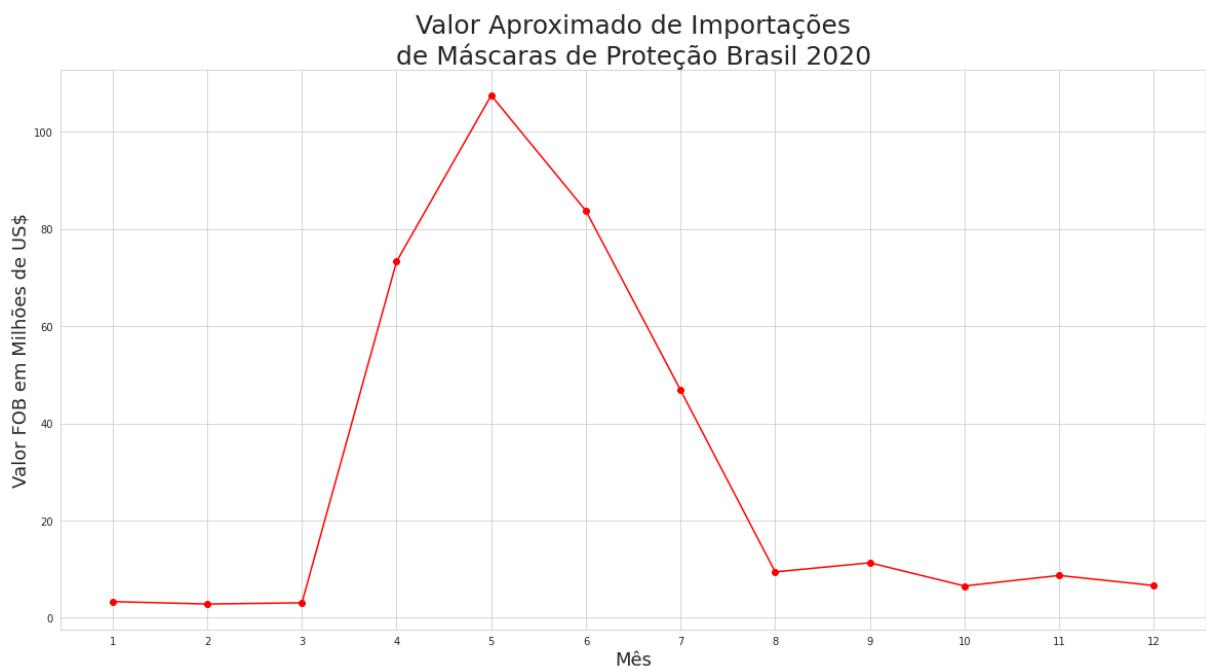
Como o Brasil exportou toda sua produção de máscaras, quando o novo coronavírus se espalhou pelo país, houve falta de máscaras de proteção, álcool gel e outros itens. A falta de máscaras e dos demais equipamentos de proteção individual (EPIs) foi especialmente crítica para os profissionais de saúde, havendo reclamações, protestos e até notícias de postos de saúde com uma única máscara para o uso de todos os profissionais (<https://gauchazh.clicrbs.com.br/coronavirus-servico/noticia/2020/03/falta-da-mascara-n95-gera-protestos-em-porto-alegre-e-divide-opinioes-de-profissionais-da-saude-ck87pcc5o01lr01rz9y6s7kwf.html>).

Com isso, o Brasil se viu obrigado a procurar emergencialmente fora do país modos de suprir a sua demanda de máscaras. Vejamos isso no gráfico a seguir:

```
#Plotando o Gráfico
f, ax = plt.subplots(1,1,figsize =(20,10))

ax.plot_date(df_mask_imp['CO_MES'],y=df_mask_imp['VL_FOB']/1000000, color="red", label="Exportação", linestyle="-", xdate = False)

plt.xticks(df_tempo_exp['CO_MES'])
plt.xlabel('Mês',size=18)
plt.ylabel('Valor FOB em Milhões de US$',size=18)
plt.title('Valor Aproximado de Importações de Máscaras de Proteção Brasil 2020', fontdict = {'fontsize': 25})
plt.show()
```

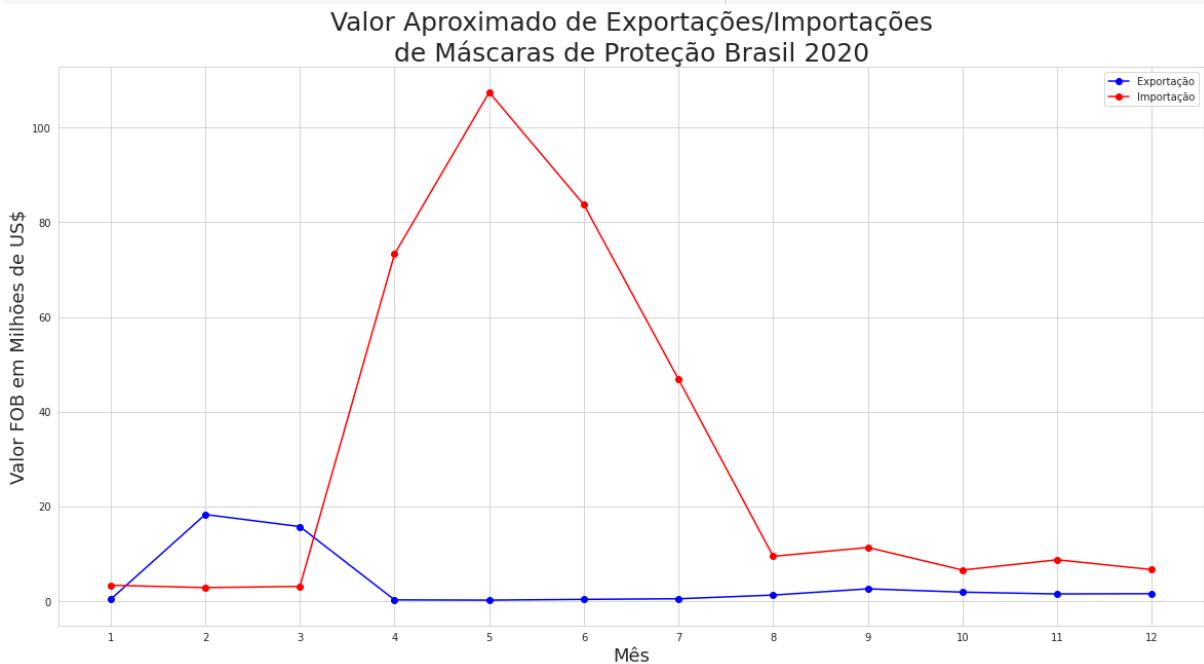


Observa-se que esse gráfico se casa perfeitamente com o outro da exportação de máscaras. Quando, em abril, esgotamos nosso estoque de máscaras pela exportação, houve um aumento súbito das importações. Vejamos os dois gráficos juntos:

```
#Juntando os dois gráficos
f, ax = plt.subplots(1,1,figsize =(20,10))

ax.plot_date(df_mask_exp['CO_MES'],y=df_mask_exp['VL_FOB']/1000000, color="blue", label="Exportação", linestyle="-", xdate = False)
ax.plot_date(df_mask_imp['CO_MES'],y=df_mask_imp['VL_FOB']/1000000, color="red", label="Importação", linestyle="-", xdate = False)
ax.legend()

plt.xticks(df_tempo_exp['CO_MES'])
plt.xlabel('Mês',size=18)
plt.ylabel('Valor FOB em Milhões de US$',size=18)
plt.title('Valor Aproximado de Exportações/Importações de Máscaras de Proteção Brasil 2020', fontdict = {'fontsize': 25})
plt.show()
```



Observa-se que em abril há o mínimo das exportações e começa o pico das importações. Nota-se também a enorme discrepância entre os valores totais de exportações e importações. Qual seria a causa disso? Será que importamos em maior quantidade do que exportamos inicialmente ou será que a discrepância é de valores: vendemos barato e compramos caro? Precisamos então alterar o *dataset* para responder a essa pergunta utilizando a coluna 'QT_ESTAT', que determina a quantidade da mercadoria em sua unidade de medida estatística (normalmente unidades). Para isso, criamos uma nova coluna: Valor FOB em US\$ por unidade, denominada FOB_UN.

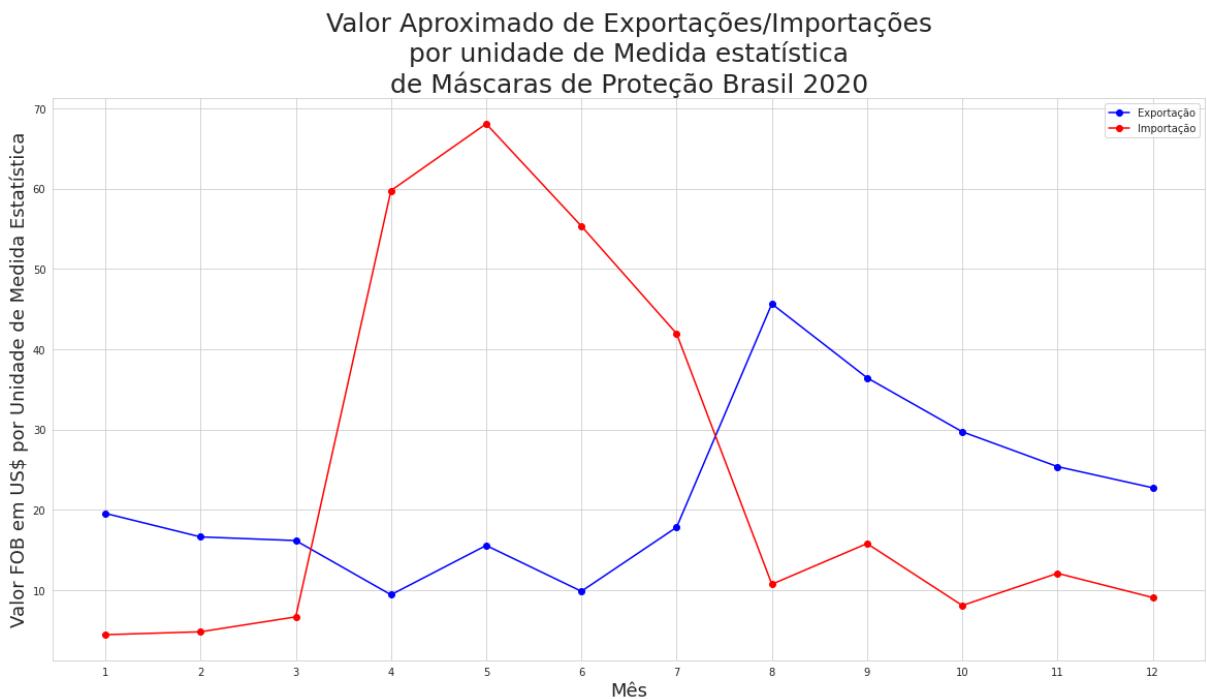
```
#Criando um novo dataset com apenas as NCMs desejadas
df_mask_exp_2 = df_exp_full.loc[df_exp_full['CO_NCM'].isin(ncms_mask)]
df_mask_imp_2 = df_imp_full.loc[df_imp_full['CO_NCM'].isin(ncms_mask)]
#Agrupando por mês
df_mask_exp_2 = df_mask_exp_2.groupby(['CO_MES'])[['VL_FOB','QT_ESTAT']].sum().reset_index()
df_mask_imp_2 = df_mask_imp_2.groupby(['CO_MES'])[['VL_FOB','QT_ESTAT']].sum().reset_index()
#Criando a coluna FOB/UN
df_mask_exp_2['FOB_UN'] = df_mask_exp_2['VL_FOB']/df_mask_exp_2['QT_ESTAT']
df_mask_imp_2['FOB_UN']= df_mask_imp_2['VL_FOB']/df_mask_imp_2['QT_ESTAT']
#Exibindo os resultados
print('Exportação')
display(df_mask_exp_2[['CO_MES','FOB_UN']])
print('Importação')
display(df_mask_imp_2[['CO_MES','FOB_UN']])
```

Plotando o gráfico novamente, mas agora com o eixo vertical em valores relativos à quantidade:

```
#Juntando os dois gráficos
f, ax = plt.subplots(1,1,figsize =(20,10))

ax.plot_date(df_mask_exp_2['CO_MES'],y=df_mask_exp_2['FOB_UN'], color="blue", label="Exportação", linestyle="--", xdate = False)
ax.plot_date(df_mask_imp_2['CO_MES'],y=df_mask_imp_2['FOB_UN'], color="red", label="Importação", linestyle="--", xdate = False)
ax.legend()

plt.xticks(df_tempo_exp['CO_MES'])
plt.xlabel('Mês',size=18)
plt.ylabel('Valor FOB em US$ por Unidade de Medida Estatística',size=18)
plt.title('Valor Aproximado de Exportações/Importações\npor unidade de Medida estatística\nde Máscaras de Proteção Brasil 2020', fontdict = {'fontsize': 25})
plt.show()
```



Como esperávamos, o alto valor das importações referia-se ao alto preço pago pelas máscaras. No pico da importação, em maio, atinge-se quase 70 dólares por unidade de medida estatística enquanto na exportação, nos 3 primeiros meses do ano, o valor não passou de 20 dólares por unidade. Exportamos as máscaras a um valor baixo, para, posteriormente, importar a um valor elevado.

Note que, como destacado anteriormente, trata-se apenas de uma estimativa pela NCM, pois os dados não possuem detalhamento suficiente para separarmos as outras eventuais mercadorias (ainda que poucas) que também podem ser classificadas nessas NCMs.

Então, em resumo, vimos que a Lei nº 13.993/2020, da proibição de Exportações, foi tardia e seu efeito imediato nulo. Contudo, ao longo do tempo, o mesmo não pode ser dito. Com a diminuição de casos do final de 2020, a manutenção da proibição das exportações e a recuperação da indústria nacional, as importações de máscaras também caíram a partir do mês de agosto. Sem essa proibição, é possível que as exportações de máscaras voltassem a subir no final de 2020, causando um novo desabastecimento na segunda onda de 2021, o que seria desastroso.

Na mesma linha, para o combate da pandemia, o governo publicou no primeiro semestre de 2021 duas resoluções concedendo redução temporária da alíquota do Imposto de Importação para diversas mercadorias, de modo a estimular a importação e facilitar o acesso da população a esses itens. A primeira delas foi a RESOLUÇÃO Nº 17, DE 17 DE MARÇO DE 2020, que beneficiou a importação de álcool em gel, EPIs, respiradores, entre outros.

Primeiramente, executamos a raspagem de dados da página web da resolução, obtendo as NCMs beneficiadas. Essa parte do código já foi exposta no tópico de coleta de dados (2. Coleta de Dados). Após isso, criamos um *dataframe* da importação utilizando essas NCMs como filtro e plotamos dois gráficos no tempo, um para valor e outro para quantidade de importações, com uma barra vertical indicando a publicação da resolução.

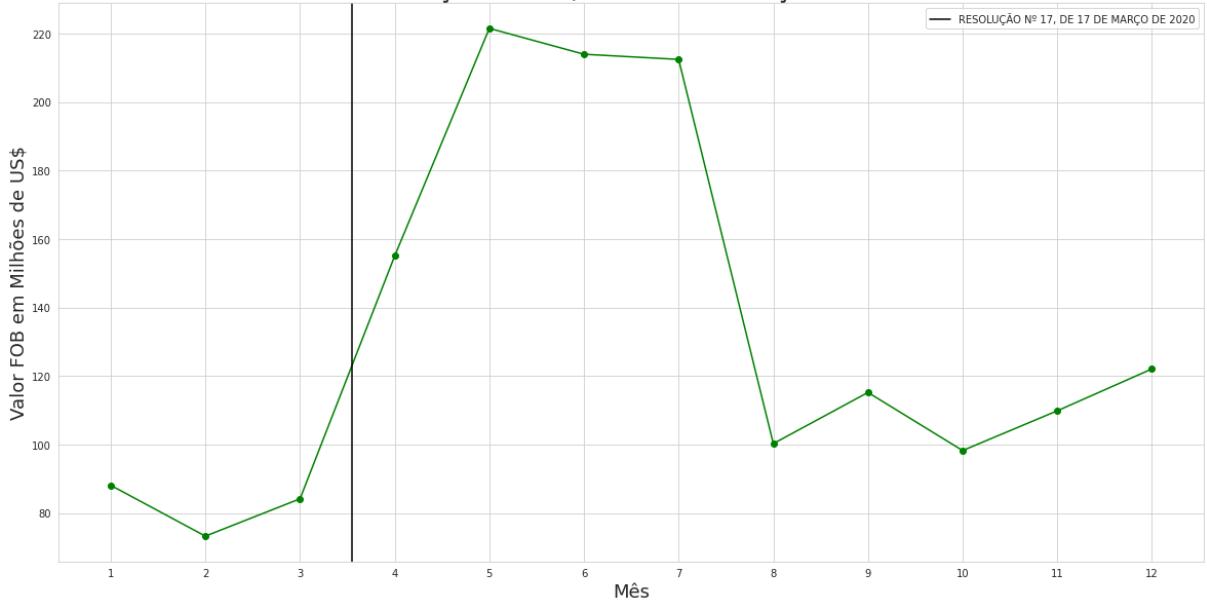
```
#Criando um novo dataset com apenas as NCMs desejadas
df_marco_imp = df_imp_full.loc[df_imp_full['CO_NCM'].isin(ncm_marco)]
#Agrupando por mês
df_marco_fob_imp = df_marco_imp.groupby(['CO_MES'])[['VL_FOB']].sum().to_frame().reset_index()
df_marco_qt_imp = df_marco_imp.groupby(['CO_MES'])[['QT_ESTAT']].sum().to_frame().reset_index()
#Exibindo os resultados
print('Valor Importação')
display(df_marco_fob_imp)
print('Quantidade Importação')
display(df_marco_qt_imp)
```

```
#Plotando o gráfico de Valor das NCMs relacionadas à Resolução de Março
f, ax = plt.subplots(1,1,figsize =(20,10))

ax.plot_date(df_marco_fob_imp['CO_MES'],y=df_marco_fob_imp['VL_FOB']/1000000, color="green", linestyle="-", xdate = False)

#Data da Resolução
plt.axvline(3+17/31, label = 'RESOLUÇÃO Nº 17, DE 17 DE MARÇO DE 2020', color = 'black')
#Definindo o local das legendas
plt.legend(loc='upper right');
plt.xticks(df_tempo_exp['CO_MES'])
plt.xlabel('Mês',size=18)
plt.ylabel('Valor FOB em Milhões de US$',size=18)
plt.title('Variação das Importações das Mercadorias da\nRESOLUÇÃO Nº 17, DE 17 DE MARÇO DE 2020', fontdict = {'fontsize': 25})
plt.show()
```

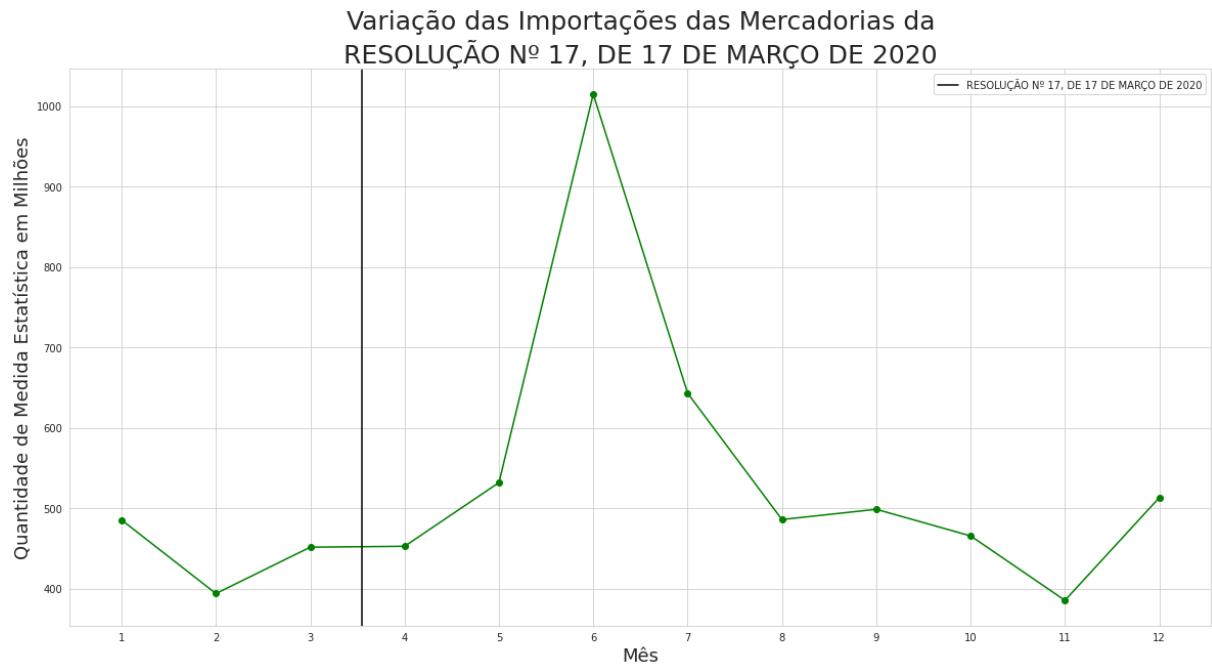
Variação das Importações das Mercadorias da
RESOLUÇÃO Nº 17, DE 17 DE MARÇO DE 2020



```
#Plotando o gráfico de Quantidade das NCMs relacionadas à Resolução de Março
f, ax = plt.subplots(1,1,figsize =(20,10))

ax.plot_date(df_marco_qty_imp['CO_MES'],y=df_marco_qty_imp['QT_ESTAT']/1000000, color="green", linestyle="-", xdate = False)

#Data da Resolução
plt.axvline(3+17/31, label = 'RESOLUÇÃO Nº 17, DE 17 DE MARÇO DE 2020', color = 'black')
#Definindo o local das legendas
plt.legend(loc='upper right');
plt.xticks(df_tempo_exp['CO_MES'])
plt.xlabel('Mês',size=18)
plt.ylabel('Quantidade de Medida Estatística em Milhões',size=18)
plt.title('Variação das Importações das Mercadorias da\nRESOLUÇÃO Nº 17, DE 17 DE MARÇO DE 2020', fontdict = {'fontsize': 25})
plt.show()
```



Possivelmente, a alta demanda, aliada aos benefícios da resolução, serviram de estímulo para as importações das mercadorias beneficiadas. Lembrando sempre que correlação não implica em causalidade: duas coisas correlacionadas não implicam, necessariamente, no fato de uma ser causa da outra.

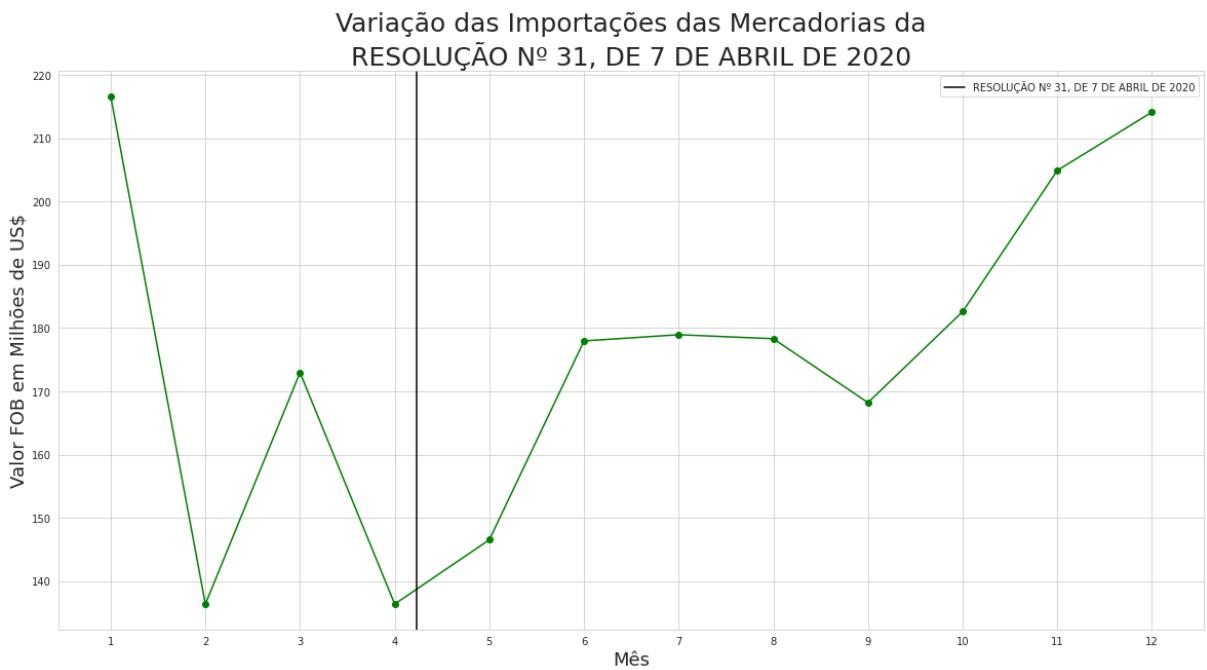
Vejamos agora a segunda resolução, a RESOLUÇÃO Nº 31, DE 7 DE ABRIL DE 2020, que beneficiou importações de vitamina D, Zinco e partes de máquinas, entre outros. Novamente não mostraremos aqui o código que executou a raspagem de dados por já estar no *notebook* e ser semelhante ao código anterior.

```
#Criando um novo dataset com apenas as NCMs desejadas
df_abril_imp = df_imp_full.loc[df_imp_full['CO_NCM'].isin(ncm_marco)]
#Agrupando por mês
df_abril_fob_imp = df_abril_imp.groupby(['CO_MES'])['VL_FOB'].sum().to_frame().reset_index()
df_abril_qt_imp = df_abril_imp.groupby(['CO_MES'])['QT_ESTAT'].sum().to_frame().reset_index()
#Exibindo os resultados
print('Valor Importação')
display(df_abril_fob_imp)
print('Quantidade Importação')
display(df_abril_qt_imp)
```

```
#Plotando o gráfico de Valor das NCMs relacionadas à Resolução de Abril
f, ax = plt.subplots(1,1,figsize =(20,10))

ax.plot_date(df_abril_fob_imp['CO_MES'],y=df_abril_fob_imp['VL_FOB']/1000000, color="green", linestyle="-", xdate = False)

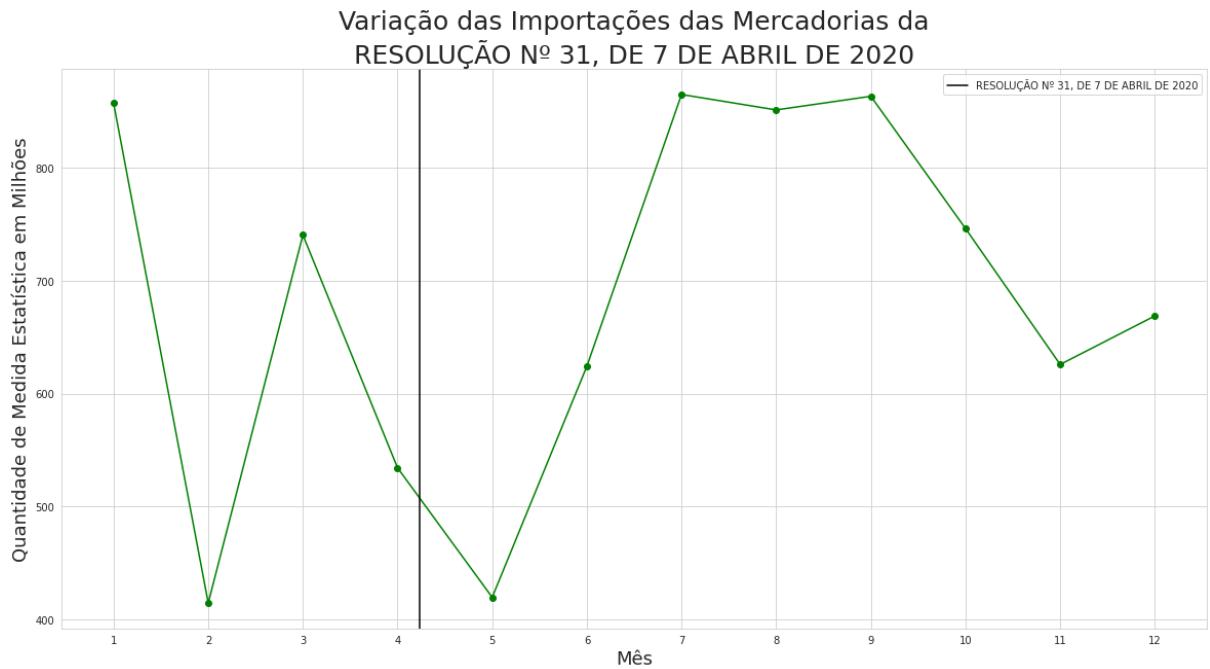
#Data da Resolução
plt.axvline(4+7/30, label = 'RESOLUÇÃO Nº 31, DE 7 DE ABRIL DE 2020', color = 'black')
#Definindo o local das legendas
plt.legend(loc='upper right');
plt.xticks(df_tempo_exp['CO_MES'])
plt.xlabel('Mês',size=18)
plt.ylabel('Valor FOB em Milhões de US$',size=18)
plt.title('Variação das Importações das Mercadorias da\nRESOLUÇÃO Nº 31, DE 7 DE ABRIL DE 2020', fontdict = {'fontsize': 25})
plt.show()
```



```
#Plotando o gráfico de Valor das NCMs relacionadas à Resolução de Abril
f, ax = plt.subplots(1,1,figsize =(20,10))

ax.plot_date(df_abril_qt_imp['CO_MES'],y=df_abril_qt_imp['QT_ESTAT']/1000000, color="green", linestyle="-", xdate = False)

#Data da Resolução
plt.axvline(4+7/30, label = 'RESOLUÇÃO Nº 31, DE 7 DE ABRIL DE 2020', color = 'black')
#Definindo o local das legendas
plt.legend(loc='upper right');
plt.xticks(df_tempo_exp['CO_MES'])
plt.xlabel('Mês',size=18)
plt.ylabel('Quantidade de Medida Estatística em Milhões',size=18)
plt.title('Variação das Importações das Mercadorias da\nRESOLUÇÃO Nº 31, DE 7 DE ABRIL DE 2020', fontdict = {'fontsize': 25})
plt.show()
```



Mesmo que menos visível que no caso da resolução anterior, é possível verificar uma tendência do aumento do valor e da quantidade das importações após a resolução. Sempre importante destacar novamente que correlação não implica causalidade.

4.5. Análise dos Modais de Transporte

Finalmente chegamos à análise dos modais de transporte internacional de mercadorias que servirá, posteriormente, como base para nossos modelos de aprendizado de máquina.

Vamos inicialmente contar a quantidade de operações por modal:

```
#Quantidade de operações por nome da via de transporte
print('Exportação - Quantidade de Operações por Via')
display(df_exp_full['NO_VIA'].value_counts())
print('Importação - Quantidade de Operações por Via')
display(df_imp_full['NO_VIA'].value_counts())

Exportação - Quantidade de Operações por Via
VIA NAO DECLARADA           481680
MARITIMA                      359209
AEREA                         327035
RODOVIARIA                   213851
EM MAOS                       26438
VICINAL FRONTEIRICO          6615
MEIOS PROPRIOS                 2275
FERROVIARIA                   373
CONDUTO/REDE DE TRANSMISSAO    189
FLUVIAL                        72
DUTOS                          52
POR REBOQUE                     29
LACUSTRE                       3
Name: NO_VIA, dtype: int64
Importação - Quantidade de Operações por Via
MARITIMA                      857280
AEREA                         826468
RODOVIARIA                   37001
VIA NAO DECLARADA              26268
ENTRADA/SAIDA FICTA            950
MEIOS PROPRIOS                 361
POSTAL                         115
CONDUTO/REDE DE TRANSMISSAO     33
COURIER                        11
FERROVIARIA                   5
FLUVIAL                        1
Name: NO_VIA, dtype: int64
```

Nota-se a grande quantidade, na exportação, de operações com “VIA NÃO DECLARADA”. Isso já foi discutido no tópico do tratamento dos dados, de modo que iremos simplesmente retirar esses dados do *dataframe*:

```
#Criando um novo dataframe retirando os dados de VIA NAO DECLARADA
df_exp_full_3=df_exp_full[df_exp_full['NO_VIA']!='VIA NAO DECLARADA']
df_imp_full_3=df_imp_full[df_imp_full['NO_VIA']!='VIA NAO DECLARADA']

print('Exportação - Quantidade de Operações por Via')
display(df_exp_full_3['NO_VIA'].value_counts())
print('Importação - Quantidade de Operações por Via')
display(df_imp_full_3['NO_VIA'].value_counts())
```

Exportação - Quantidade de Operações por Via

MARITIMA	359209
AEREA	327035
RODOVIARIA	213851
EM MAOS	26438
VICINAL FRONTEIRICO	6615
MEIOS PROPRIOS	2275
FERROVIARIA	373
CONDUTO/REDE DE TRANSMISSAO	189
FLUVIAL	72
DUTOS	52
POR REBOQUE	29
LACUSTRE	3

Name: NO_VIA, dtype: int64

Importação - Quantidade de Operações por Via

MARITIMA	857280
AEREA	826468
RODOVIARIA	37001
ENTRADA/SAIDA FICTA	950
MEIOS PROPRIOS	361
POSTAL	115
CONDUTO/REDE DE TRANSMISSAO	33
COURIER	11
FERROVIARIA	5
FLUVIAL	1

Name: NO_VIA, dtype: int64

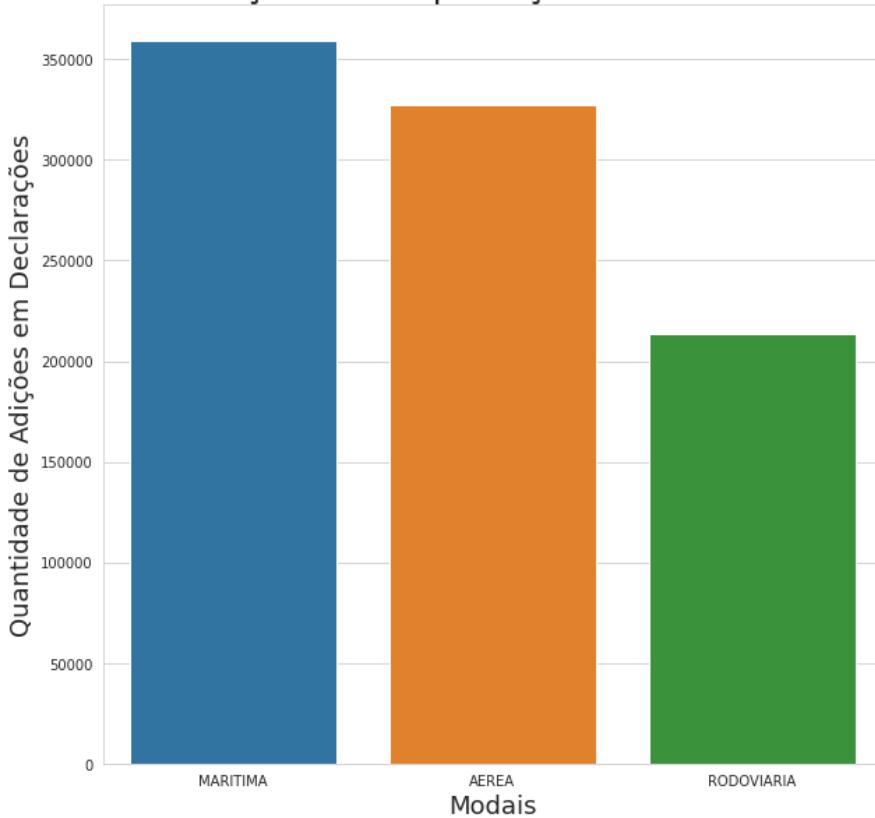
Podemos então selecionar os 3 (três) maiores modais em cada tipo por quantidade de adições (operações):

```
#Armazenando os dados dos maiores para plotar
data_exp = df_exp_full_3['NO_VIA'].value_counts().nlargest(3)
data_imp = df_imp_full_3['NO_VIA'].value_counts().nlargest(3)

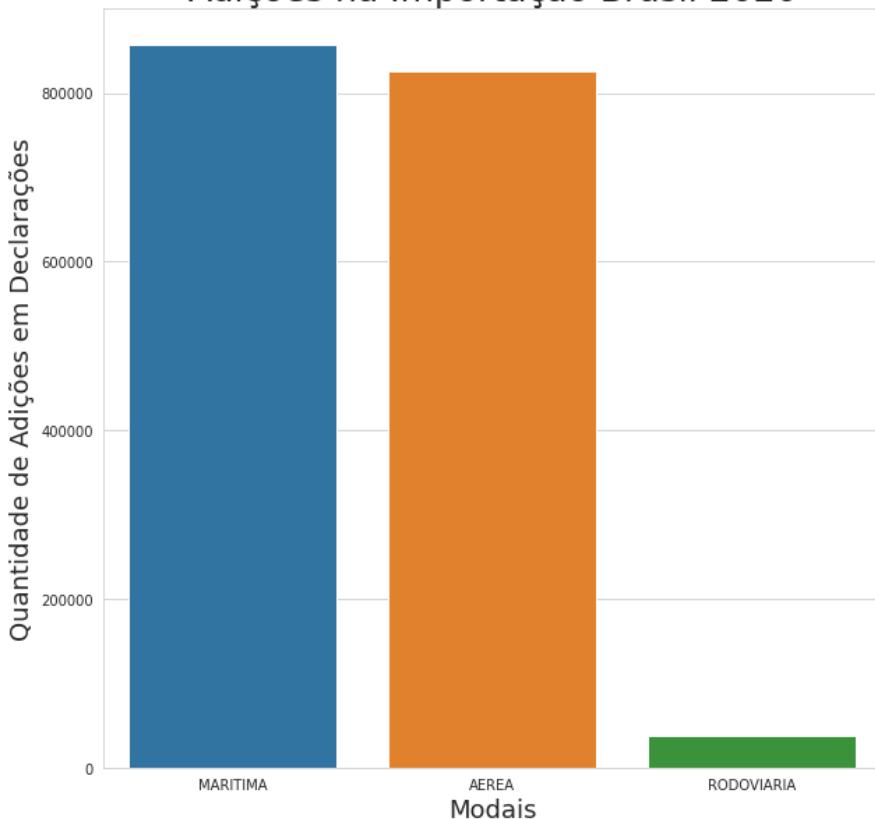
#Plotando o gráfico da exportação
plt.figure(figsize=(10,10))
sns.set_style('whitegrid')
sns.barplot(x=data_exp.index,y=data_exp.values)
plt.xlabel('Modais',size=18)
plt.ylabel('Quantidade de Adições em Declarações',size=18)
plt.title('3 Maiores Modais em Quantidade de\nAdições na Exportação Brasil 2020', fontdict = {'fontsize': 25})

#Plotando o gráfico da importação
plt.figure(figsize=(10,10))
sns.set_style('whitegrid')
sns.barplot(x=data_imp.index,y=data_imp.values)
plt.xlabel('Modais',size=18)
plt.ylabel('Quantidade de Adições em Declarações',size=18)
plt.title('3 Maiores Modais em Quantidade de\nAdições na Importação Brasil 2020', fontdict = {'fontsize': 25})
```

3 Maiores Modais em Quantidade de Adições na Exportação Brasil 2020



3 Maiores Modais em Quantidade de Adições na Importação Brasil 2020



Vemos que tanto na exportação quanto na importação os maiores modais em quantidade de operações são o marítimo, o aéreo e o rodoviário, sendo o rodoviário menos relevante na importação.

Façamos agora as mesmas análises utilizando dessa vez os valores das operações como eixo das ordenadas:

```
#Agrupando e ordenando os dados
df_modais_fob_exp = df_exp_full_3.groupby(['NO_VIA'])['VL_FOB'].sum().sort_values(ascending=False).to_frame().reset_index()
print('Exportação')
display(df_modais_fob_exp)

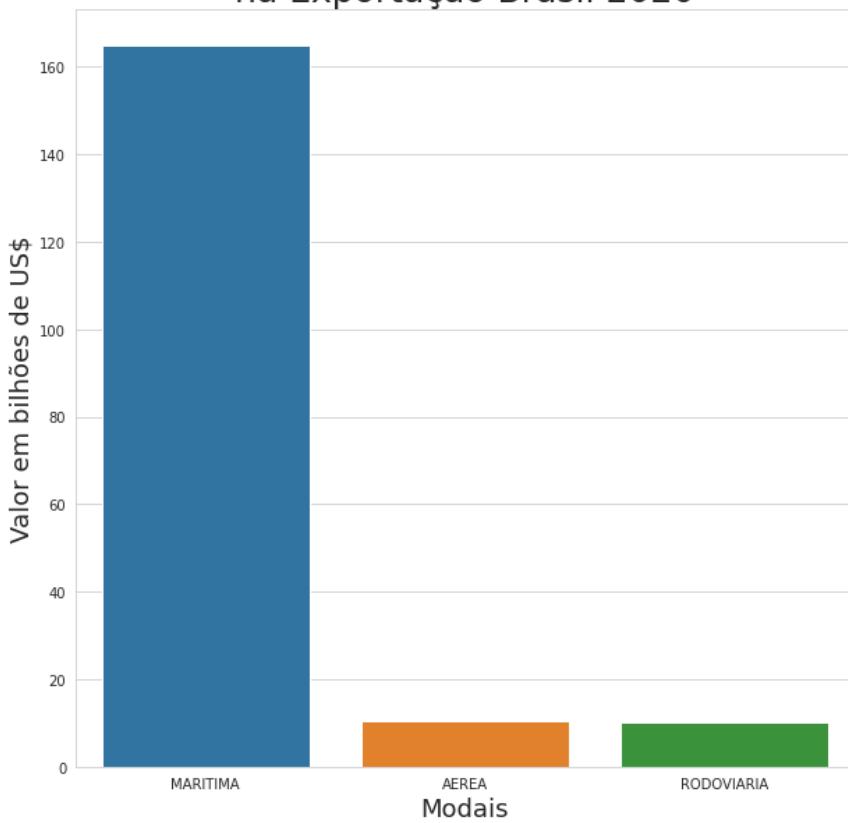
df_modais_fob_imp = df_imp_full_3.groupby(['NO_VIA'])['VL_FOB'].sum().sort_values(ascending=False).to_frame().reset_index()
print('Importação')
display(df_modais_fob_imp)

#Armazenando os dados dos maiores 3 para plotar
data_exp = df_modais_fob_exp.nlargest(3,'VL_FOB')
data_imp = df_modais_fob_imp.nlargest(3,'VL_FOB')

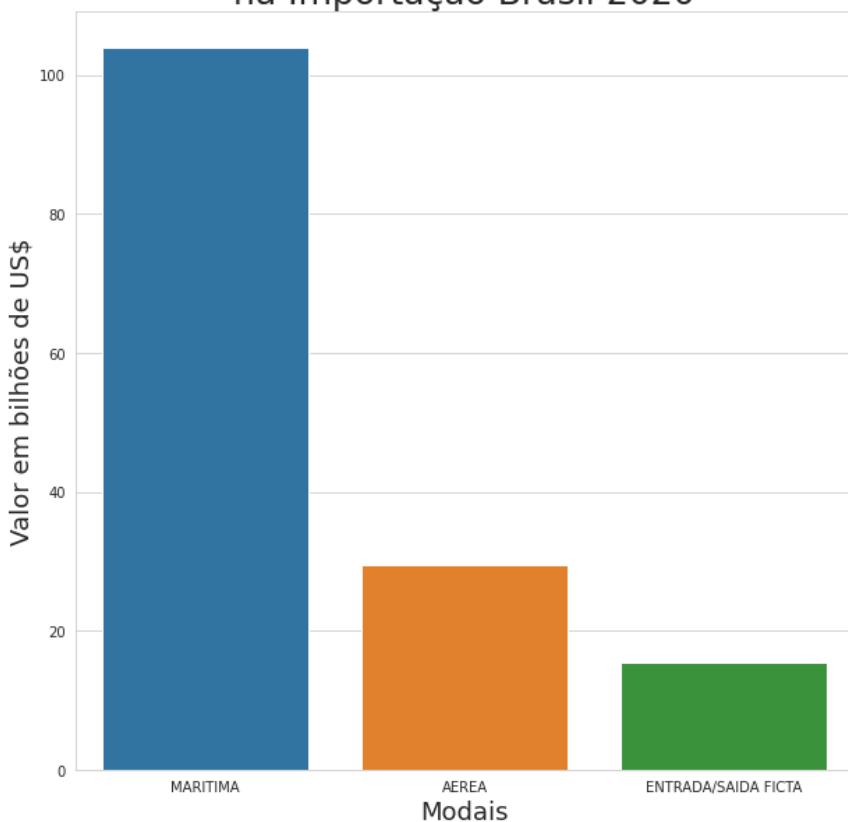
#Plotando o gráfico da exportação
plt.figure(figsize=(10,10))
sns.set_style('whitegrid')
sns.barplot(x=data_exp['NO_VIA'],y=data_exp['VL_FOB']/1000000000)
plt.xlabel('Modais',size=18)
plt.ylabel('Valor em bilhões de US$',size=18)
plt.title('3 Maiores Modais em Valor\nna Exportação Brasil 2020', fontdict = {'fontsize': 25})

#Plotando o gráfico da importação
plt.figure(figsize=(10,10))
sns.set_style('whitegrid')
sns.barplot(x=data_imp['NO_VIA'],y=data_imp['VL_FOB']/1000000000)
plt.xlabel('Modais',size=18)
plt.ylabel('Valor em bilhões de US$',size=18)
plt.title('3 Maiores Modais em Valor\nna Importação Brasil 2020', fontdict = {'fontsize': 25})
```

3 Maiores Modais em Valor na Exportação Brasil 2020



3 Maiores Modais em Valor na Importação Brasil 2020



Nota-se aqui visivelmente que o modal marítimo é dominante em valor total nas operações de comércio exterior brasileira, sobretudo na exportação.

Vejamos agora o comparativo por peso transportado:

```
#Agrupando e ordenando os dados
df_modais_peso_exp = df_exp_full_3.groupby(['NO_VIA'])['KG_LIQUIDO'].sum().sort_values(ascending=False).to_frame().reset_index()
print('Exportação')
display(df_modais_peso_exp)

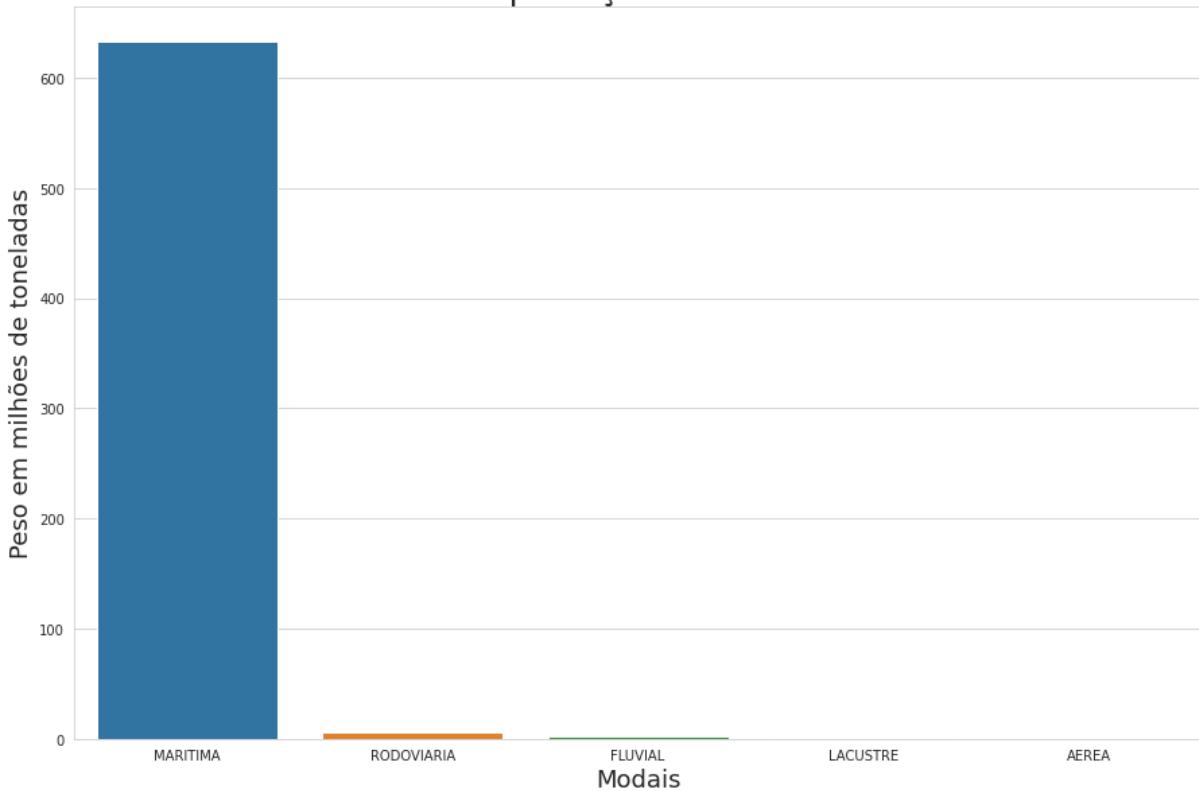
df_modais_peso_imp = df_imp_full_3.groupby(['NO_VIA'])['KG_LIQUIDO'].sum().sort_values(ascending=False).to_frame().reset_index()
print('Importação')
display(df_modais_peso_imp)

#Armazenando os dados dos maiores 5 para plotar
data_exp = df_modais_peso_exp.nlargest(5,'KG_LIQUIDO')
data_imp = df_modais_peso_imp.nlargest(5,'KG_LIQUIDO')

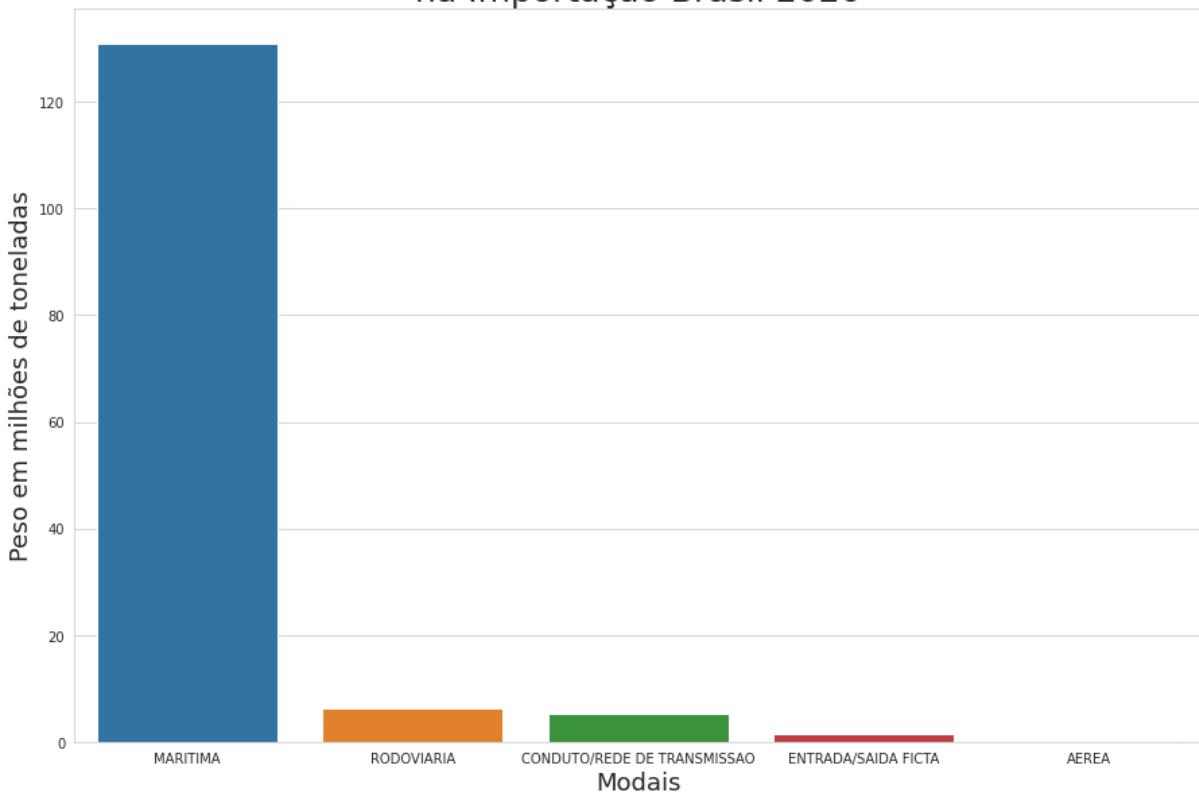
#Plotando o gráfico da exportação
plt.figure(figsize=(15,10))
sns.set_style('whitegrid')
sns.barplot(x=data_exp['NO_VIA'],y=data_exp['KG_LIQUIDO']/1000000000)
plt.xlabel('Modais',size=18)
plt.ylabel('Peso em milhões de toneladas',size=18)
plt.title('5 Maiores Modais em Peso\nna Exportação Brasil 2020', fontdict = {'fontsize': 25})

#Plotando o gráfico da importação
plt.figure(figsize=(15,10))
sns.set_style('whitegrid')
sns.barplot(x=data_imp['NO_VIA'],y=data_imp['KG_LIQUIDO']/1000000000)
plt.xlabel('Modais',size=18)
plt.ylabel('Peso em milhões de toneladas',size=18)
plt.title('5 Maiores Modais em Peso\nna Importação Brasil 2020', fontdict = {'fontsize': 25})
```

5 Maiores Modais em Peso na Exportação Brasil 2020



5 Maiores Modais em Peso na Importação Brasil 2020



Como esperado, o modal marítimo é o modal ideal para transportar grandes tamanhos, pesos e quantidades. A discrepância de pesos é tão grande que a visualização do gráfico fica prejudicada. Por isso, utilizaremos uma escala logarítmica devido ao logaritmo reduzir o tamanho dessa representação dos dados no eixo y, facilitando a visualização.

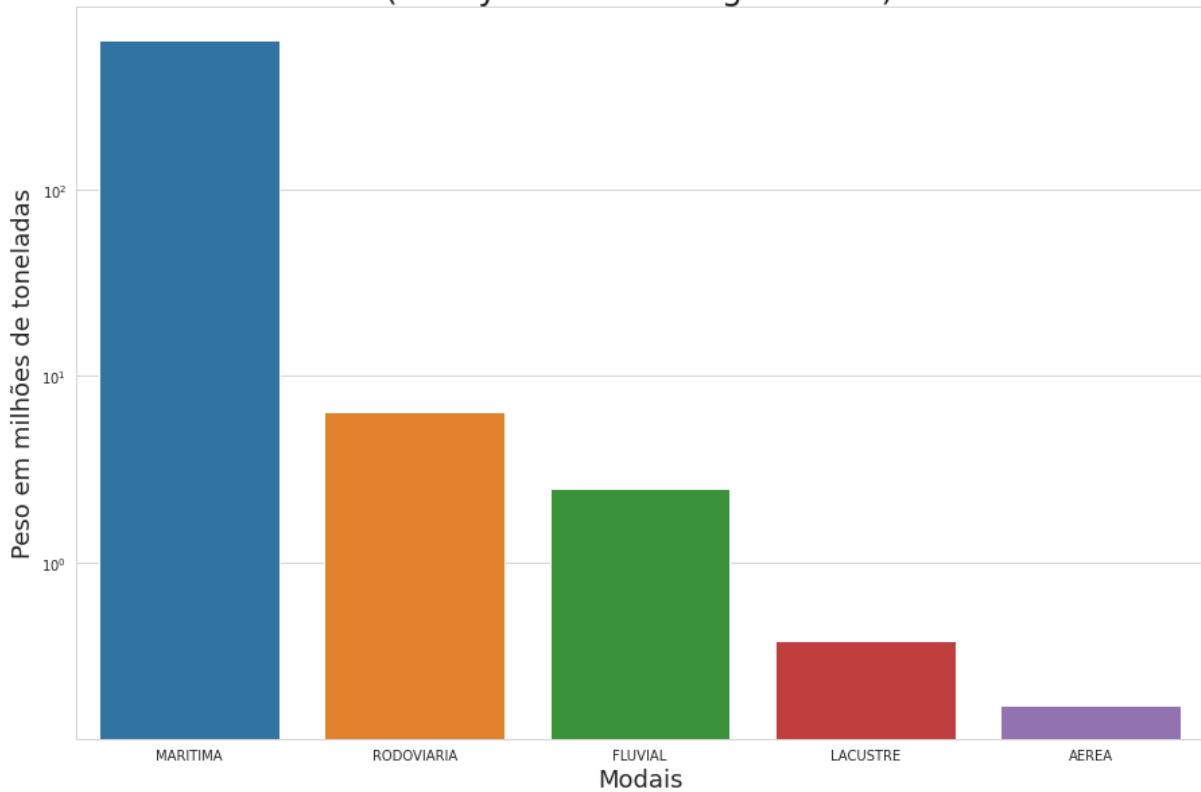
```
#Plotando o eixo y em escala logaritmica

#Armazenando os dados dos maiores 5 para plotar
data_exp = df_modais_peso_exp.nlargest(5,'KG_LIQUIDO')
data_imp = df_modais_peso_imp.nlargest(5,'KG_LIQUIDO')

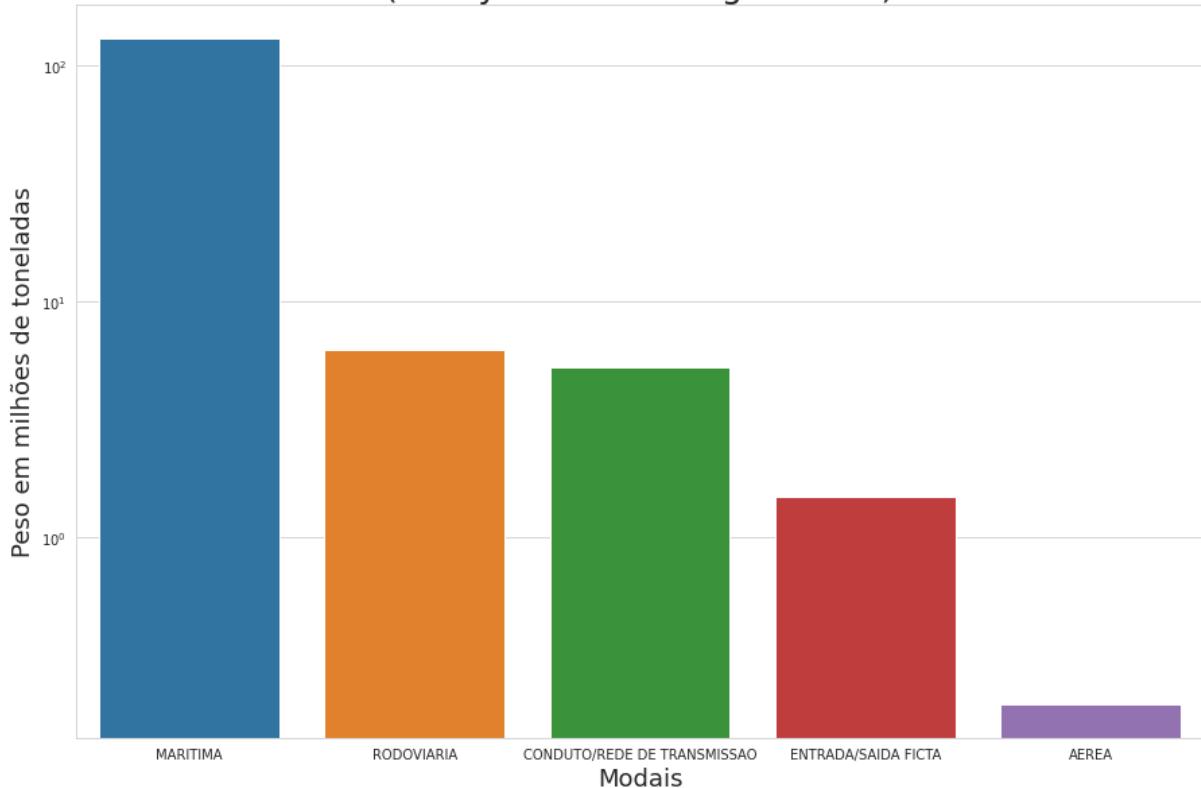
#Plotando o gráfico da exportação
plt.figure(figsize=(15,10))
ax = plt.gca()
ax.set_yscale('log')
sns.set_style('whitegrid')
sns.barplot(x=data_exp['NO_VIA'],y=data_exp['KG_LIQUIDO']/1000000000)
plt.xlabel('Modais',size=18)
plt.ylabel('Peso em milhões de toneladas',size=18)
plt.title('5 Maiores Modais em Peso\nna Exportação Brasil 2020\n(eixo y em escala logarítmica)', fontdict = {'fontsize': 25})

#Plotando o gráfico da importação
plt.figure(figsize=(15,10))
ax = plt.gca()
ax.set_yscale('log')
sns.set_style('whitegrid')
sns.barplot(x=data_imp['NO_VIA'],y=data_imp['KG_LIQUIDO']/1000000000)
plt.xlabel('Modais',size=18)
plt.ylabel('Peso em milhões de toneladas',size=18)
plt.title('5 Maiores Modais em Peso\nna Importação Brasil 2020\n(eixo y em escala logarítmica)', fontdict = {'fontsize': 25})
```

**5 Maiores Modais em Peso
na Exportação Brasil 2020
(eixo y em escala logarítmica)**



**5 Maiores Modais em Peso
na Importação Brasil 2020
(eixo y em escala logarítmica)**



Concluímos então que o modal marítimo é o campeão em valor e peso, devido a sua capacidade de transporte de maiores volumes. A tendência, como explicado no momento da proposição do problema, é que mercadorias maiores, mais baratas e menos urgentes sejam transportadas pelo modal marítimo e menores, mais caras e mais urgentes pelo modal aéreo.

Verifiquemos então o comparativo entre os modais em relação ao valor por peso (FOB/kg). Para isso precisamos criar uma nova coluna (FOB_KG):

```
#Reutilizando os dataframes já construídos utilizando um join
df_modais_fob_peso_exp=df_modais_fob_exp.join(df_modais_peso_exp.set_index('NO_VIA'), on='NO_VIA')
df_modais_fob_peso_imp=df_modais_fob_imp.join(df_modais_peso_imp.set_index('NO_VIA'), on='NO_VIA')

#Criando a nova coluna com a divisão
df_modais_fob_peso_exp['FOB_KG'] = df_modais_fob_peso_exp['VL_FOB']/df_modais_fob_peso_exp['KG_LIQUIDO']
df_modais_fob_peso_imp['FOB_KG'] = df_modais_fob_peso_imp['VL_FOB']/df_modais_fob_peso_imp['KG_LIQUIDO']

#Deletando as colunas desnecessárias
#df_modais_fob_peso_exp.drop(['VL_FOB', 'KG_LIQUIDO'], axis=1, inplace=True)
#df_modais_fob_peso_imp.drop(['VL_FOB', 'KG_LIQUIDO'], axis=1, inplace=True)

print('Exportação')
display(df_modais_fob_peso_exp[['NO_VIA', 'FOB_KG']])

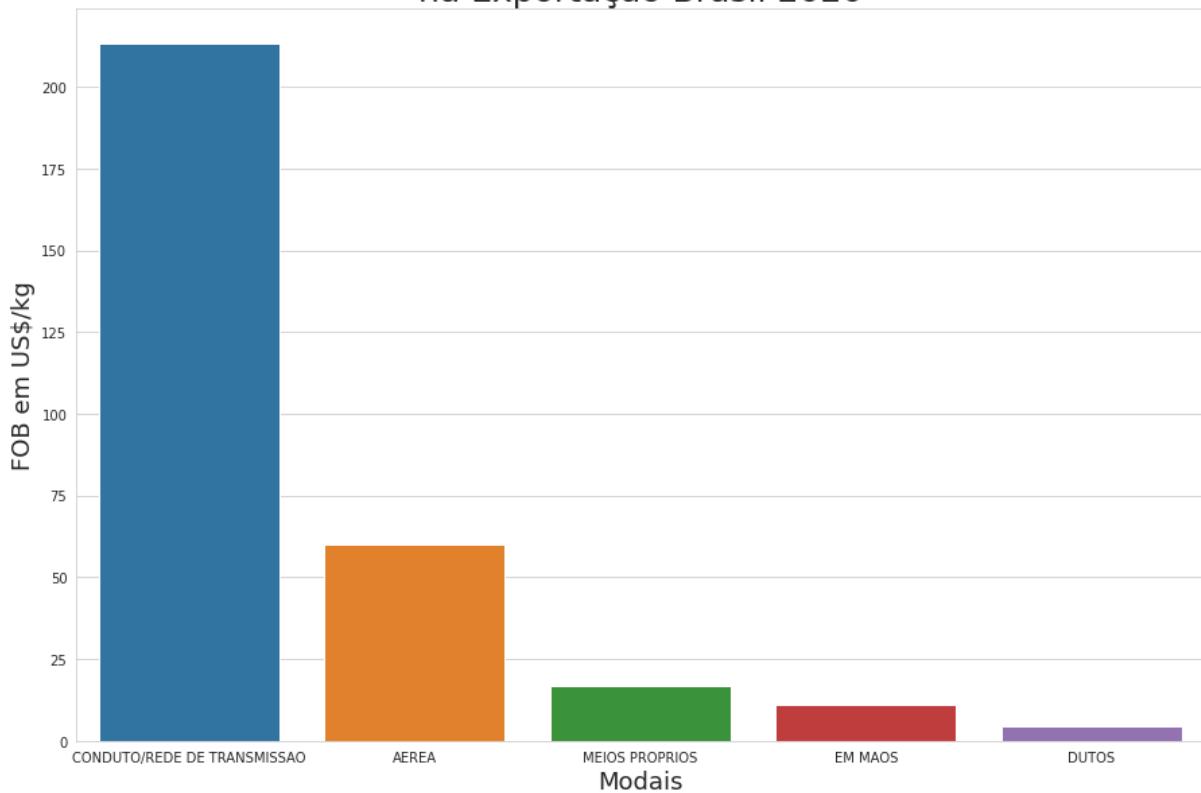
print('Importação')
display(df_modais_fob_peso_imp[['NO_VIA', 'FOB_KG']])

#Armazenando os dados dos maiores 3 para plotar
data_exp = df_modais_fob_peso_exp.nlargest(5,'FOB_KG')
data_imp = df_modais_fob_peso_imp.nlargest(5,'FOB_KG')

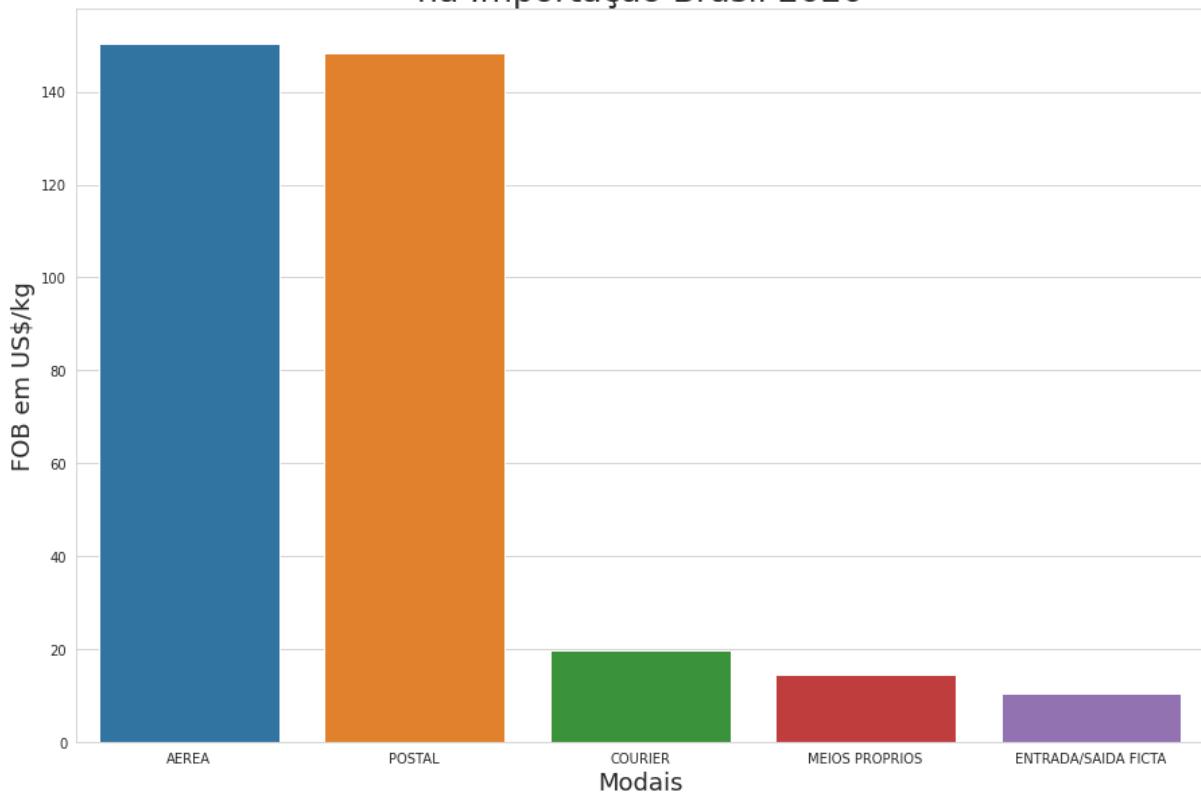
#Plotando o gráfico da exportação
plt.figure(figsize=(15,10))
sns.set_style('whitegrid')
sns.barplot(x=data_exp['NO_VIA'],y=data_exp['FOB_KG'])
plt.xlabel('Modais',size=18)
plt.ylabel('FOB em US$/kg',size=18)
plt.title('5 Maiores Modais em FOB/kg\nna Exportação Brasil 2020', fontdict = {'fontsize': 25})

#Plotando o gráfico da importação
plt.figure(figsize=(15,10))
sns.set_style('whitegrid')
sns.barplot(x=data_imp['NO_VIA'],y=data_imp['FOB_KG'])
plt.xlabel('Modais',size=18)
plt.ylabel('FOB em US$/kg',size=18)
plt.title('5 Maiores Modais em FOB/kg\nna Importação Brasil 2020', fontdict = {'fontsize': 25})
```

5 Maiores Modais em FOB/kg na Exportação Brasil 2020



5 Maiores Modais em FOB/kg na Importação Brasil 2020



Aqui o quadro é outro. Na exportação, os maiores valores por peso são por conduto/rede de transmissão, usada para o transporte de petróleo, gás e energia elétrica. O segundo lugar é do modal aéreo. Na importação os maiores valores são por via aérea (de cargas) e em segundo lugar a postal, que também é, quase em sua totalidade, aérea.

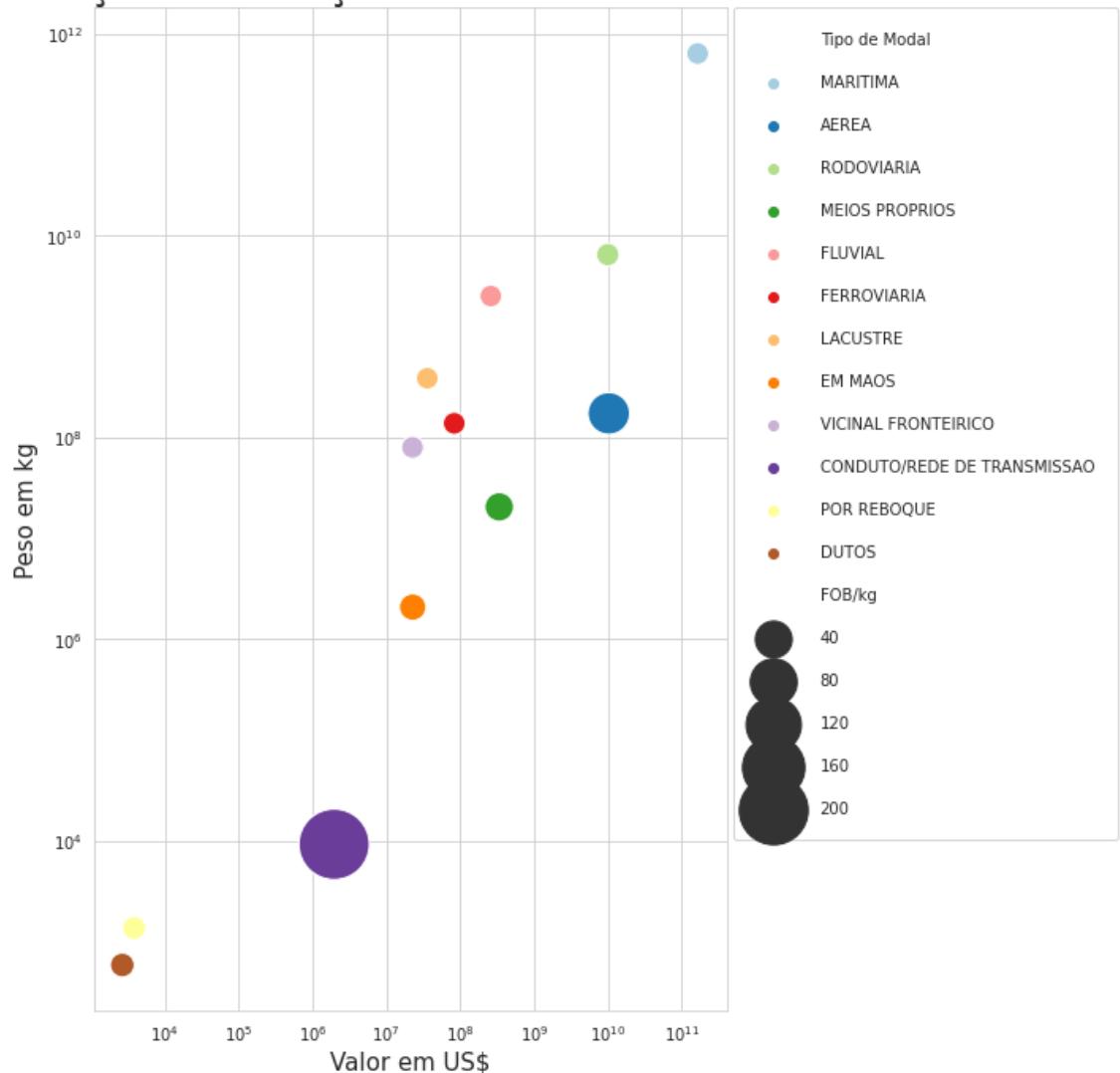
Vamos agora tentar agregar todas essas informações (valor, peso e FOB/kg) em uma única representação gráfica e fazer uma análise combinada. Para tal, utilizaremos o gráfico de dispersão, representação de dados de duas ou mais variáveis organizadas em um gráfico.

```
#Plotando um gráfico agora único com todas informações
f, ax = plt.subplots(figsize=(10, 10))
sns.set_style('whitegrid')
ax.set(xscale="log", yscale="log")
sns.scatterplot(x=df_modais_fob_peso_exp['VL_FOB'], y=df_modais_fob_peso_exp['KG_LIQUIDO'],
                 hue=df_modais_fob_peso_exp['NO_VIA'].rename('Tipo de Modal'),
                 ax=ax, size=df_modais_fob_peso_exp['FOB_KG'].rename('FOB/kg'),
                 sizes=(200, 2000), legend = 'brief', palette = 'Paired')

# Colocar a legenda do lado de fora do gráfico
plt.legend(bbox_to_anchor=(1.01, 1), borderaxespad=0, borderpad=1.5, labelspacing=1.7, handletextpad = 2)

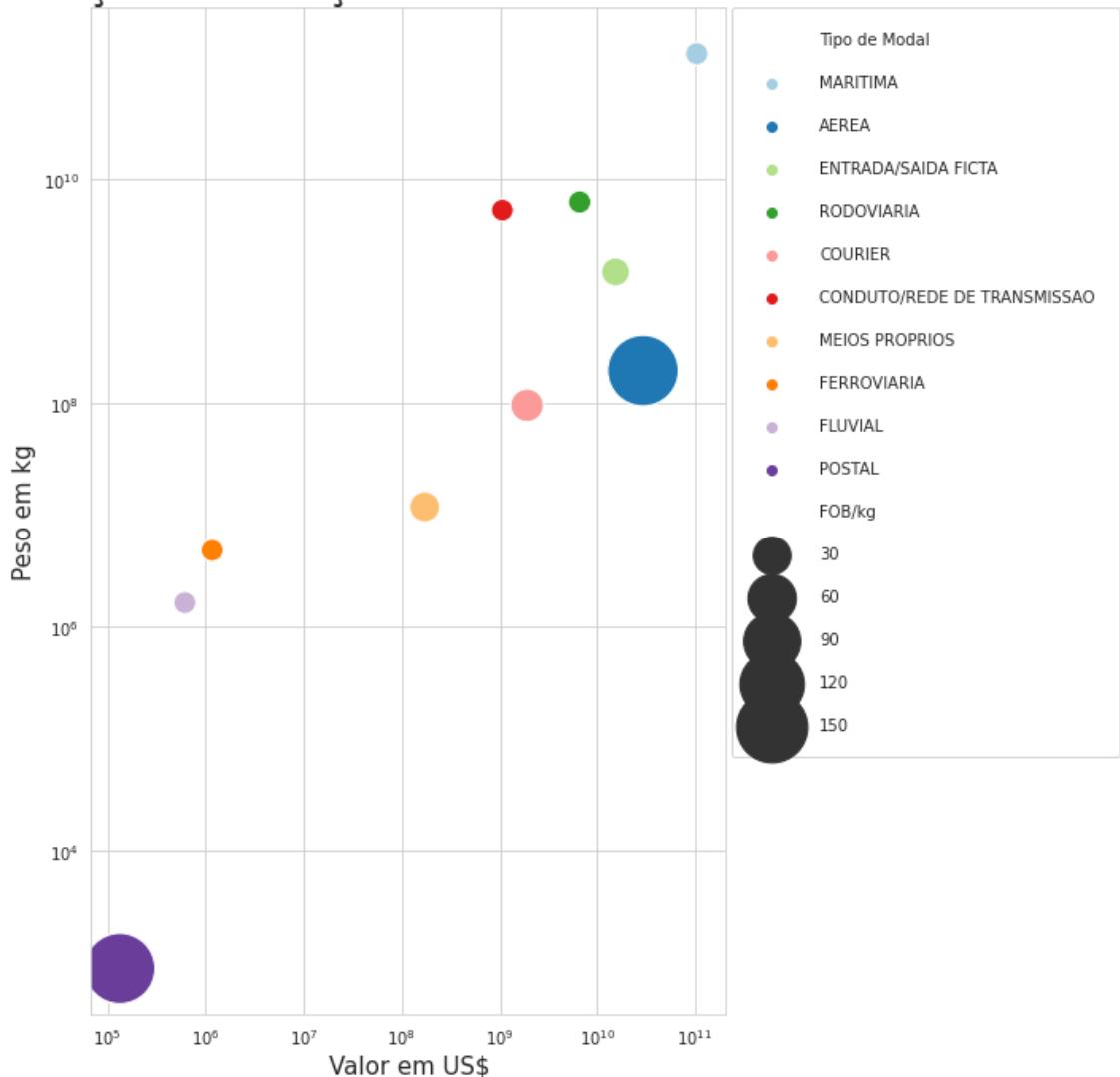
# Definindo os textos
plt.xlabel("Valor em US$", fontsize=15)
plt.ylabel("Peso em kg", fontsize=15)
plt.title("Exportação: Correlação Valor e Peso nos Modais", fontsize=25)
plt.tight_layout()
plt.show()
```

Exportação: Correlação Valor e Peso nos Modais



Fazendo o equivalente para a importação temos o seguinte resultado:

Importação: Correlação Valor e Peso nos Modais



Ambos os eixos estão em escala logarítmica. O tamanho dos círculos representa o FOB/kg. Podemos com essa representação verificar com facilidade todas as informações presentes nos gráficos anteriores e as principais características de cada modal. Note, por exemplo, o isolamento no canto superior direito do círculo que representa o modal marítimo e seu tamanho diminuto, explicado pelos seus altos valores e pesos, mas baixo FOB/kg. Ou do tamanho grande do círculo do modal aéreo, graças ao seu alto FOB/kg.

5. Criação de Modelos de Machine Learning

O aprendizado de máquina (*Machine Learning*) é o estudo de algoritmos de computador que melhoram automaticamente através da experiência e do uso de dados. Pode ser classificado em: a) aprendizado supervisionado, em que o computador é apresentado com exemplos de entradas e suas saídas desejadas, rotuladas, e o objetivo é aprender uma regra geral que mapeia entradas em saídas; b) aprendizado não supervisionado, em que nenhum rótulo é dado ao computador, deixando-o sozinho para encontrar a estrutura em sua entrada. O aprendizado não supervisionado pode ser um objetivo em si (descobrir padrões ocultos nos dados) ou um meio para um fim (aprendizado de recursos); e c) aprendizado por reforço, em que o computador interage com um ambiente dinâmico no qual deve realizar uma determinada tarefa. À medida que navega no espaço do problema, o programa recebe um *feedback*, que ele tenta maximizar.

O presente estudo é um claro problema de classificação supervisionada. Faremos a predição do modal comum, dada as características da operação, e com isso poderemos detectar a utilização de modais estranhos.

Isso tudo já foi explicado anteriormente na proposição do problema, mas não custa enfatizar o que nos motiva nessa empreitada. Por exemplo, por que motivo alguém usaria um modal caro para uma mercadoria barata ou vice-versa? Ou por que motivo alguém usaria um modal diferente de todas as outras importações da mesma mercadoria da mesma origem? Vários motivos legítimos podem existir, como a urgência ou a falta dela ou a compra em conjunto de outras mercadorias com características distintas, pois, como dito anteriormente, só temos nesses *datasets* o nível de detalhamento de adição, não temos informações de identificação das declarações individuais. Ou seja, por mais que uma mercadoria barata esteja em um modal caro, ela pode estar sendo transportada em conjunto com uma mercadoria cara no mesmo voo. Nos dados reais das declarações poderíamos detectar esses casos, mas com a granularidade disponível nesses *datasets* isso não é possível.

Assim como fizemos no momento da importação e análise dos dados, iremos aqui tratar das bibliotecas necessárias.

```
#Importando as bibliotecas específicas de ML
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.model_selection import cross_val_score
from pycaret.classification import *
```

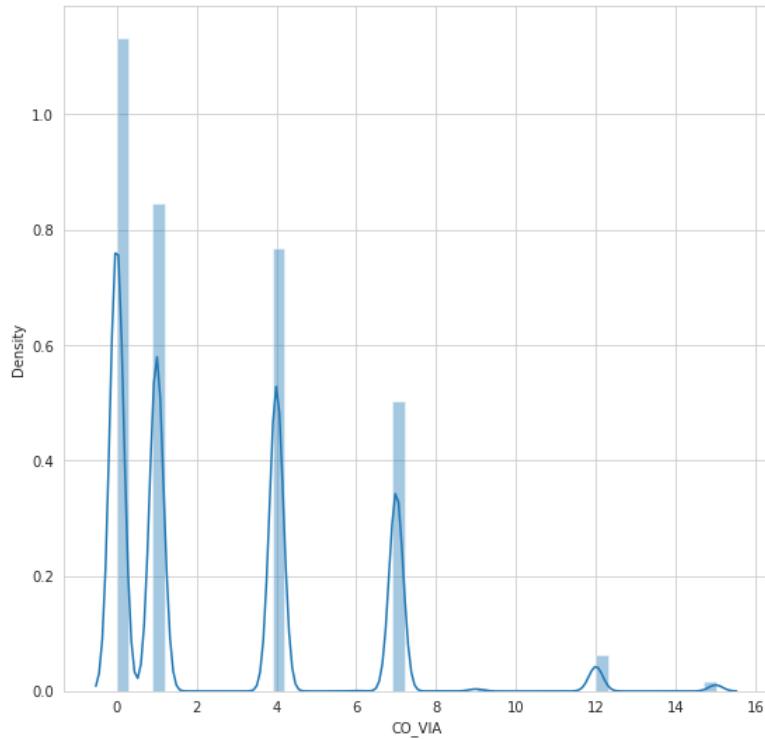
- 1) sklearn – biblioteca de aprendizado de máquina de código aberto para a linguagem de programação Python;
 - a. train_test_split – utilizada para separação da base de dados em treino e teste;
 - b. GaussianNB – utilizada para criação do modelo de Naive Bayes;
 - c. RandomForestClassifier – utilizada para criação do modelo Random Forest;
 - d. confusion_matrix – utilizada para criação da matriz de confusão;
 - e. precision_score – utilizada para a métrica de desempenho de mesmo nome;
 - f. recall_score – utilizada para a métrica de desempenho de mesmo nome;
 - g. f1_score – utilizada para a métrica de desempenho de mesmo nome;
 - h. cross_val_score – utilizada para a validação cruzada.
- 2) pycaret – biblioteca de aprendizado de máquina de pouco código.

Visto isso, vamos primeiramente verificar a distribuição dos dados. Para isso, iremos utilizar o histograma, uma distribuição de frequências.

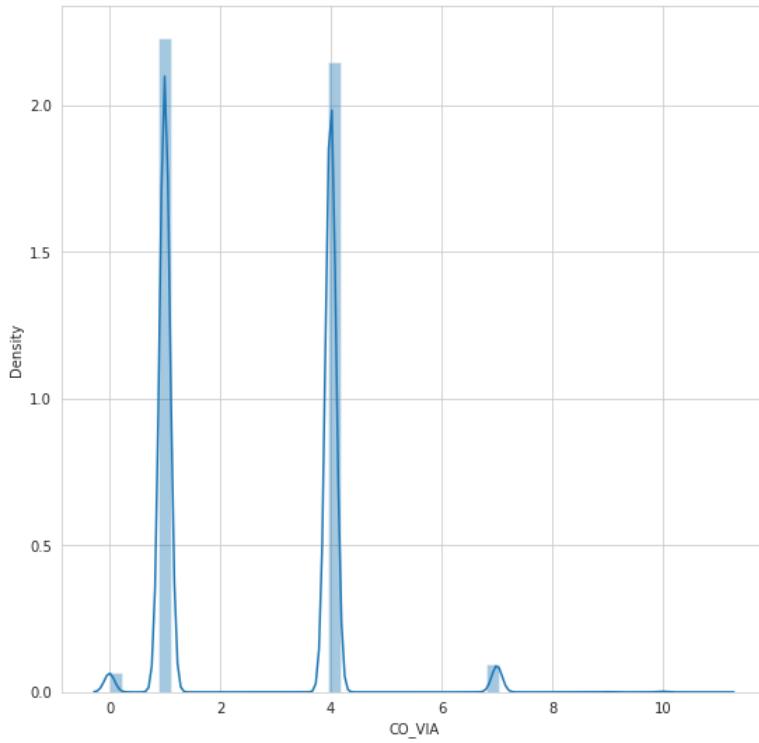
```
#Lendo os dados
df_exp_ml = pd.read_csv('/content/gdrive/MyDrive/data/EXP_2020.csv', sep=';')
df_imp_ml = pd.read_csv('/content/gdrive/MyDrive/data/IMP_2020.csv', sep=';')
df_via = pd.read_csv('/content/gdrive/MyDrive/data/VIA.csv', sep=';')

#Vendo a distribuição de valores
print('Exportação')
display(sns.distplot(df_exp_ml['CO_VIA']))
```

Primeiro para a exportação:



Em seguida para a importação:



Como vemos, há um grande desbalanceamento das classes. Num modelo de classificação, a consequência desse desequilíbrio é que o modelo terá uma tendência a dar muitos “alarmes falsos”, ele irá ter um bom desempenho para as classes majoritárias, mas terá um desempenho ruim para as minoritárias.

Trataremos disso mais adiante, escolhendo apenas os principais modais para cada tipo de operação.

Primeiramente retiramos do *dataframe* três colunas, CO_ANO, SG_UF_NCM e CO_URF. A primeira pois é constante, a segunda e a terceira pois estão diretamente relacionadas à variável alvo: cada estado e cada unidade da Receita Federal estão limitados a determinados modais. Por exemplo: uma unidade da RFB que seja um aeroporto, terá quase que a totalidade de suas operações por meio aéreo. Utilizando esses dados teríamos facilmente um modelo com grande acurácia, mas nenhuma utilidade. Queremos justamente detectar quando uma empresa se utiliza de um modal incomum (e consequentemente de uma rota ou unidade da RFB incomum) para suas operações. Colocando esses dados como fonte de informação, perdemos o propósito do trabalho. Sendo assim, iremos retirar essas colunas e verificar as correlações da variável alvo com as demais variáveis.

```
#Já retirando algumas colunas do df que não serão utilizadas
#Retirando a coluna CO_ANO (constante em todo df) e SG_UF_NCM
#(string diretamente relacionada à variável alvo, o modal) e 'CO_URF'
#(diretamente relacionada à variável alvo, o modal)
df_exp_ml = df_exp_ml.drop(['CO_ANO', 'SG_UF_NCM', 'CO_URF'], axis=1)
df_imp_ml = df_imp_ml.drop(['CO_ANO', 'SG_UF_NCM', 'CO_URF'], axis=1)

#Imprimindo as correlações com a variável alvo
c_exp = df_exp_ml.corr().abs()
c_exp = c_exp['CO_VIA'].sort_values()
print('Exportação')
display(c_exp)

c_imp = df_imp_ml.corr().abs()
c_imp = c_imp['CO_VIA'].sort_values()
print('Importação')
display(c_imp)
```

```

Exportação
QT_ESTAT      0.004651
KG_LIQUIDO    0.006574
VL_FOB        0.014956
CO_PAIS       0.030054
CO_MES         0.036746
CO_UNID       0.049330
CO_NCM         0.261130
CO_VIA         1.000000
Name: CO_VIA, dtype: float64
Importação
QT_ESTAT      0.004817
VL_FOB        0.004997
CO_UNID       0.009324
CO_MES         0.015436
CO_PAIS       0.016794
KG_LIQUIDO    0.018878
CO_NCM         0.068721
CO_VIA         1.000000

```

Quanto maior o valor obtido, maior a interdependência entre duas ou mais variáveis e mais importante é a variável para o modelo. Vemos que não temos uma correção muito forte entre as variáveis preditoras e a variável alvo. O que indica que, como estão, os dados não serão capazes de produzir um modelo com alta acurácia.

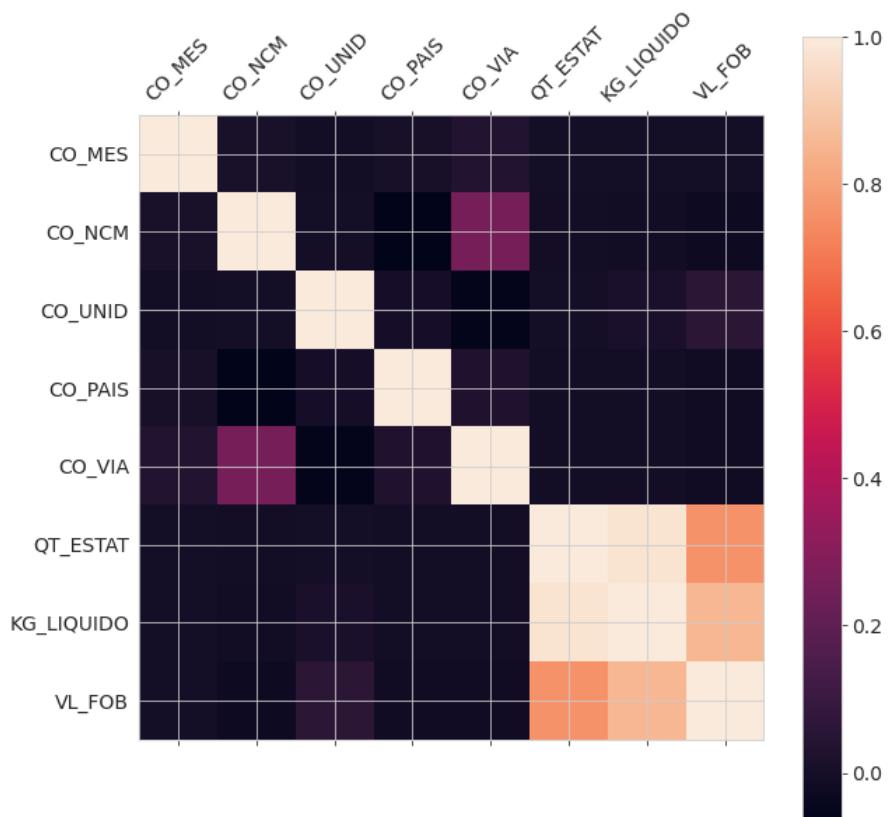
Podemos também plotar a matriz de correlação. Uma matriz de correlação é uma tabela mostrando coeficientes de correlação entre todas as variáveis.

```

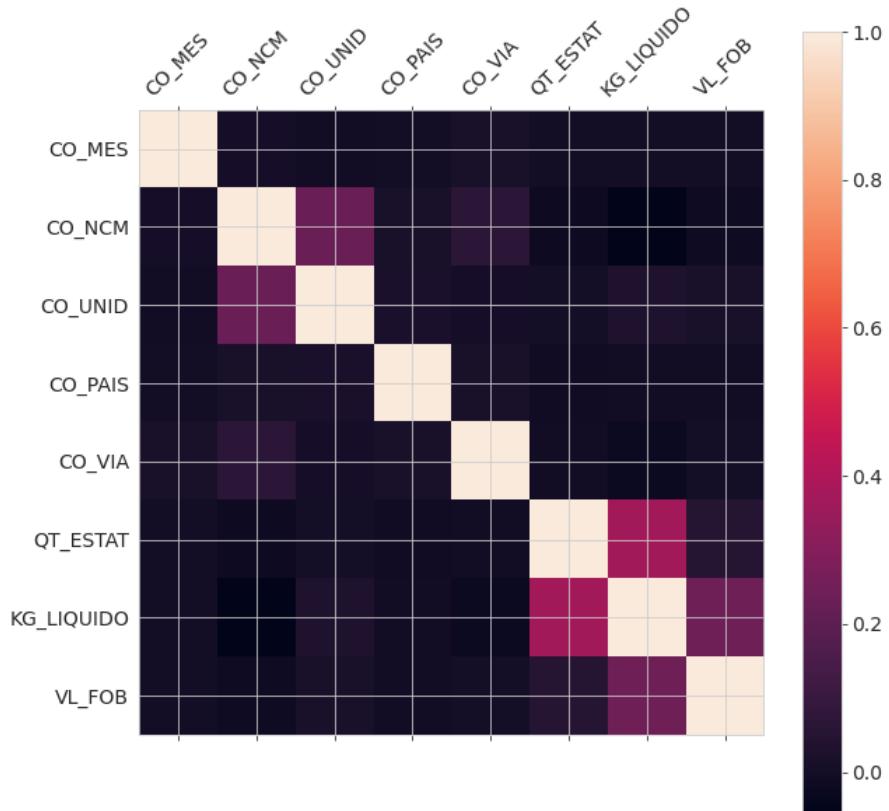
#Matriz de correlação Exportação

f = plt.figure(figsize=(10, 10))
plt.matshow(df_exp_ml.corr(), fignum=f.number)
plt.xticks(range(df_exp_ml.shape[1]), df_exp_ml.columns, fontsize=14, rotation=45)
plt.yticks(range(df_exp_ml.shape[1]), df_exp_ml.columns, fontsize=14)
cb = plt.colorbar()
cb.ax.tick_params(labelsize=14)
#plt.title('Martiz de Correlação Exportação', fontsize=16)

```



Fazendo o mesmo para a importação:



Podemos observar que as variáveis QT_ESTAT, KG_LIQUIDO e VL_FOB tem uma forte correlação, principalmente na exportação, onde a correlação entre

QT_ESTAT e KG_LIQUIDO é quase 1. Isso significa que elas são de algum modo dependentes umas das outras. Com isso, usar as três variáveis contribui muito pouco para o modelo e seus resultados, mas aumenta o custo computacional. Trataremos disso adiante, juntando as variáveis em uma única variável ou eliminando variáveis.

Já começamos nos tópicos anteriores, principalmente no do tratamento de dados, a fazer a chamada *Feature Engineering* (Engenharia de Variáveis), quando unimos *datasets*, verificamos e retiramos linhas e colunas. Precisamos fazer isso pois muitas vezes os dados como estão podem não ser suficientes para projetar um bom modelo de aprendizado de máquina. Por isso, devemos preparar os dados para o uso dos algoritmos na busca de um melhor desempenho dos modelos.

Tendo isso em mente podemos partir para o código em si:

```
#Tratando as variáveis: Feature Engineering + Missing Data
#Esse é um passo iterativo analisando as matrizes anteriores e
#os resultados posteriores na busca do melhor resultado

#Retirando os modais com baixa ocorrência, pegando apenas os maiores e quantidade de operações por modal conforme item 2.5.1. e distribuição de valores
#Exportação: 1 - Marítima, 4 - Aérea, 7 - Rodoviária; Importação: 1 - Marítima, 4 - Aérea
counts = df_exp_ml['CO_VIA'].value_counts()
df_exp2 = df_exp_ml[~df_exp_ml['CO_VIA'].isin(counts[counts < 100000].index)]

counts = df_imp_ml['CO_VIA'].value_counts()
df_imp2 = df_imp_ml[~df_imp_ml['CO_VIA'].isin(counts[counts < 100000].index)]

#Retirando o modal "VIA NAO DECLARADA" de ambos df, conforme já explicado no momento da análise dos dados do item 2.5.1.
df_exp2 = df_exp2[df_exp2['CO_VIA']!=0]
df_imp2 = df_imp2[df_imp2['CO_VIA']!=0]

#Na exportação, tirando eventuais linhas com o peso zero ou extremamente baixo
#e criando a coluna FOB_KG, conforme já explicado no momento da análise dos dados
#Como podemos ver claramente na matriz de correlação da exportação, essas variáveis
#são fortemente correlacionadas entre si. Na importação não são tão correlacionadas
#e após testes verificou-se melhor desempenho com elas separadas
df_exp2=df_exp2[df_exp2['KG_LIQUIDO']>0.0001]
df_exp2['FOB_KG'] = df_exp2[['VL_FOB']]/df_exp2['KG_LIQUIDO']

#Retirando de ambos df as colunas CO_URF (determinante da variável alvo, o modal de transporte)
#CO_UNID (determinante da variável alvo, o modal de transporte), QT_ESTAT (correlacionada com KG_LIQUIDO)
#CO_MES (categórica, pesando para o modelo e com baixíssima correlação e resultado nos modelos após testes)

df_exp2 = df_exp2.drop(['QT_ESTAT', 'CO_UNID', 'CO_MES', 'VL_FOB', 'KG_LIQUIDO'], axis=1)
display(df_exp2)

df_imp2 = df_imp2.drop(['QT_ESTAT', 'CO_UNID', 'CO_MES'], axis=1)
display(df_imp2)
```

Primeiramente retiramos os modais de baixa ocorrência para eliminar o problema de desbalanceamento. Mantivemos os maiores modais em quantidade de operações (acima de 100 mil) para ambos *datasets*. Com isso, ficamos com os seguintes modais: Exportação: 1 - Marítima, 4 - Aérea, 7 - Rodoviária; Importação: 1 - Marítima, 4 – Aérea.

Como a correlação entre KG_LIQUIDO e VALOR_FOB é alta, principalmente na exportação, iremos utilizar a coluna FOB_KG que agrupa as duas informações. Na importação, devido a correlação ser menor, deixaremos as duas colunas. Retiramos de ambos *datasets* a colunas QT_ESTAT.

Por fim, tiramos outras colunas com baixa correlação que, mesmo após testes, não serviram para melhorar o modelo. Observe que esse é um passo iterativo, na busca do melhor modelo possível.

```
#Na exportação, tirando eventuais linhas com o peso zero ou extremamente baixo
#e criando a coluna FOB_KG, conforme já explicado no momento da análise dos dados
#Como podemos ver claramente na matriz de correlação da exportação, essas variáveis
#são fortemente correlacionadas entre si. Na importação não são tão correlacionadas
#e após testes verificou-se melhor desempenho com elas separadas
df_exp2=df_exp2[df_exp2['KG_LIQUIDO']>0.0001]
df_exp2['FOB_KG'] = df_exp2['VL_FOB']/df_exp2['KG_LIQUIDO']

#Retirando de ambos df as colunas
#CO_UNID (dispensável, código da unidade de medida utilizada), QT_ESTAT (correlacionada com KG_LIQUIDO)
#CO_MES (categórica, pesando para o modelo e com baixíssima correlação e resultado nos modelos após testes)

df_exp2 = df_exp2.drop(['QT_ESTAT', 'CO_UNID', 'CO_MES', 'VL_FOB', 'KG_LIQUIDO'], axis=1)
display(df_exp2)

df_imp2 = df_imp2.drop(['QT_ESTAT', 'CO_UNID', 'CO_MES' ], axis=1)
display(df_imp2)

#Tratando as variáveis: Feature Engineering + Missing Data
#Esse é um passo iterativo analisando as matrizes anteriores e
#os resultados posteriores na busca do melhor resultado

#Retirando os modais com baixa ocorrência, pegando apenas os maiores e quantidade de operações
#por modal conforme item 2.5.1. e distribuição de valores
#Exportação: 1 - Marítima, 4 - Aérea, 7 - Rodoviária; Importação: 1 - Marítima, 4 - Aérea
counts = df_exp_ml['CO_VIA'].value_counts()
df_exp2 = df_exp_ml[~df_exp_ml['CO_VIA'].isin(counts[counts < 100000].index)]

counts = df_imp_ml['CO_VIA'].value_counts()
df_imp2 = df_imp_ml[~df_imp_ml['CO_VIA'].isin(counts[counts < 100000].index)]

#Retirando o modal "VIA NAO DECLARADA" de ambos df,
#conforme já explicado no momento da análise dos dados do item 2.5.1.
df_exp2 = df_exp2[df_exp2['CO_VIA']!=0]
df_imp2 = df_imp2[df_imp2['CO_VIA']!=0]
```

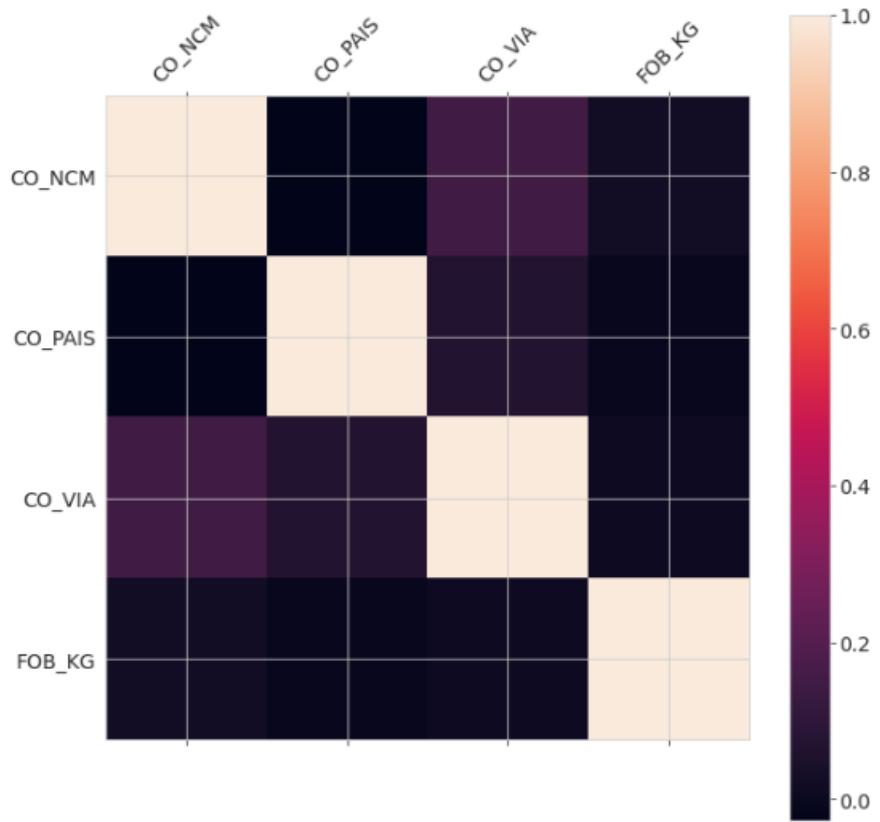
Com os *dataframes* prontos, podemos então verificar os tipos de dados e as matrizes de correlação resultantes:

```
#Verificando os tipos de dados
df_exp2.info()
df_imp2.info()

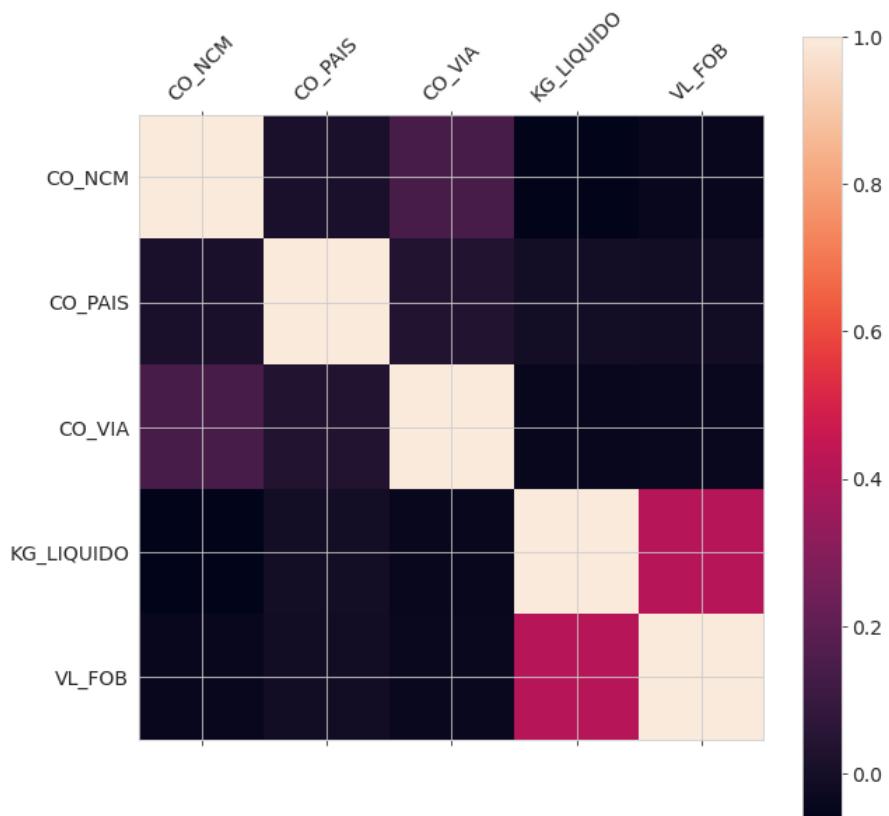
<class 'pandas.core.frame.DataFrame'>
Int64Index: 828732 entries, 0 to 1417817
Data columns (total 4 columns):
 #   Column      Non-Null Count   Dtype  
--- 
 0   CO_NCM      828732 non-null    int64  
 1   CO_PAIS     828732 non-null    int64  
 2   CO_VIA      828732 non-null    int64  
 3   FOB_KG      828732 non-null    float64 
dtypes: float64(1), int64(3)
memory usage: 31.6 MB
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1683748 entries, 0 to 1748492
Data columns (total 5 columns):
 #   Column      Non-Null Count   Dtype  
--- 
 0   CO_NCM      1683748 non-null    int64  
 1   CO_PAIS     1683748 non-null    int64  
 2   CO_VIA      1683748 non-null    int64  
 3   KG_LIQUIDO  1683748 non-null    int64  
 4   VL_FOB      1683748 non-null    int64  
dtypes: int64(5)
memory usage: 77.1 MB
```

```
#Matriz de correlação Exportação

f = plt.figure(figsize=(10, 10))
plt.matshow(df_exp2.corr(), fignum=f.number)
plt.xticks(range(df_exp2.shape[1]), df_exp2.columns, fontsize=14, rotation=45)
plt.yticks(range(df_exp2.shape[1]), df_exp2.columns, fontsize=14)
cb = plt.colorbar()
cb.ax.tick_params(labelsize=14)
#plt.title('Martiz de Correlação Exportação', fontsize=16)
```



E para a importação:



5.1. Naive Bayes (NB) e Random Forest (RF) na Exportação

Iremos nesse momento atacar o problema proposto, utilizando dois modelos muito conhecidos. O primeiro deles é o algoritmo de *Naive Bayes.*, classificador probabilístico baseado no Teorema de Bayes.

Naive Bayes é uma técnica de classificação com uma suposição de independência entre as variáveis preditoras. O classificador *Naive* (do inglês *ingênuo*) *Bayes* assume que a presença de uma característica particular em uma classe não está relacionada com a presença de qualquer outra. É um modelo fácil de construir, leve, rápido e útil em bases com grandes volumes de dados. Para entender o algoritmo, vejamos a seguinte fórmula base:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Onde:

- 1) $P(A | B)$ é a probabilidade posterior da classe (A, alvo) dada preditor (B, atributos).
- 2) $P(A)$ é a probabilidade original da classe.
- 3) $P(B | A)$ é a probabilidade que representa a probabilidade de preditor dada a classe.
- 4) $P(B)$ é a probabilidade original do preditor.

O segundo modelo que utilizaremos é o modelo *Random Forest*, que significa floresta aleatória. Para entendê-lo, primeiro devemos entender o que é uma árvore de decisão. Nesse algoritmo vários pontos de decisão são criados. Estes pontos são os “nós” da árvore e em cada um deles o resultado da decisão será seguir por um caminho (ramo) ou outro. Em resumo, o algoritmo criará uma espécie de fluxograma com “nós” onde uma condição é verificada e, se atendida, o fluxo segue por esse ramo, caso contrário, por outro, sempre levando ao próximo nó, até a finalização da árvore e a decisão final.



Fonte: <https://didatica.tech/como-funciona-o-algoritmo-arvore-de-decisao/>

A Random Forest é um *ensemble*, uma combinação de diferentes modelos para se obter um único resultado. Nesse caso, serão criadas várias árvores de decisão aleatórias e, em problemas de classificação, o resultado que mais vezes for apresentado será o escolhido.

Comparando os dois algoritmos vemos que o *Random Forest* (RF) é mais elaborado, fazendo uso da correlação entre as variáveis ao preço de um maior custo computacional. *Naive Bayes* (NB) tem um tamanho de modelo menor e com crescimento constante em relação aos dados. RF é robusto e dá bons resultados com uma base de treinamento grande e é bom para dados numéricos e categóricos, apesar de lento para treinar. NB é fácil de treinar, pequeno e útil para tarefas textuais mas, por não considerar a correlação entre as variáveis, pode produzir resultados ruins dependendo do problema.

Para criação dos modelos, utilizaremos a biblioteca *scikit-learn*, uma biblioteca de aprendizado de máquina de código aberto para a linguagem de programação Python.

Vamos então primeiro separar a base de dados em treino e testes com o uso do método `train_test_split`:

```
#Separando treino e testes

X = df_exp2.drop('CO_VIA', axis=1)
Y = df_exp2['CO_VIA']

x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=18)
```

Definimos a variável alvo como a coluna CO_VIA e separamos a base de dados com 70% dos dados para treino e 30% para teste. Usamos o random_state 18 para permitir a reproduzibilidade dos resultados.

Criando inicialmente o modelo NB utilizando uma distribuição gaussiana temos o seguinte:

```
model = GaussianNB()
model.fit(x_train, y_train)

y_pred = model.predict(x_test)

print('Acurácia NB')
print((y_test == y_pred).sum()/x_test.shape[0])
```

Acurácia NB
0.4276566647896388

No primeiro comando treinamos o modelo com a base de treinamento para depois fazer as previsões na base de teste. A medida de avaliação de desempenho utilizada para avaliar o modelo foi a acurácia, definida como:

$$Acuracia = \frac{VP + VN}{VP + VN + FP + FN}$$

Onde:

- 1) VP são os verdadeiros positivos
- 2) VN são os verdadeiros negativos
- 3) FP são os falsos positivos
- 4) FN são os falsos negativos

Temos como resultado para o modelo NB uma acurácia baixa de 42,7%.

Vejamos agora o modelo RF:

```

model = RandomForestClassifier(n_estimators=10, random_state=18)
model.fit(x_train, y_train)

y_pred = model.predict(x_test)

print('Acurácia RF')
print((y_test == y_pred).sum()/x_test.shape[0])

Acurácia RF
0.7744911913763978

```

Com o modelo RF temos um resultado bem superior, de 77,4%, mostrando o grande poder preditivo desse modelo.

5.2. Naive Bayes (NB) e Random Forest (RF) na Importação

Todos os comentários da seção anterior se aplicam a essa, de modo que não iremos reproduzi-los novamente. Apenas faremos os comentários quando necessário.

```

#Separando o dataset em treino e teste
X = df_imp2.drop('CO_VIA', axis=1)
Y = df_imp2['CO_VIA']

x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=18)

model = GaussianNB()
model.fit(x_train, y_train)

y_pred = model.predict(x_test)

print('Acurácia NB')
print((y_test == y_pred).sum()/x_test.shape[0])

Acurácia NB
0.6034446919079436

```

Observa-se que o modelo NB obteve um resultado melhor na importação (60,3%) do que havia obtido na exportação. Isso se deve ao fato de só termos 2 possíveis resultados (aéreo e marítimo), e não 3 (aéreo, marítimo e rodoviário), nesse tipo de operação.

```

model = RandomForestClassifier(n_estimators=10, random_state=18)
model.fit(x_train, y_train)

y_pred = model.predict(x_test)

print('Acurácia RF')
print((y_test == y_pred).sum()/x_test.shape[0])

```

Acurácia RF
0.7639693145261074

Novamente temos uma melhora no modelo RF (76,4%). Contudo, a acurácia obtida é inferior à obtida no modelo da exportação (77,4%).

5.3. Demais Modelos na Exportação

Iremos agora buscar formas de melhorar os modelos, ou buscar modelos melhores, tendo como parâmetro a acurácia. Para isso, testaremos diversos outros algoritmos, tentaremos alterar seus parâmetros para obter melhores resultados (o chamado *tunning*) e tentaremos combinar modelos para ver se conseguimos um modelo combinado superior (o chamado *ensembling*).

Para essa empreitada, em vez de ficarmos alterando iterativamente os parâmetros dos modelos criados manualmente nos itens anteriores e criando novos modelos, utilizaremos o *PyCaret*, uma biblioteca criada para facilitar a utilização de algoritmos de *machine learning*. Com ela, iremos comparar, tunar, juntar e assim buscar o melhor algoritmo para nosso problema de classificação de forma rápida.

Inicialmente, assim como fizemos anteriormente, iremos separar a base em teste e treino.

```

#Separando a base entre teste e treino
#O pycaret não vai utilizar esse dataset de test. Estamos diminuindo o dataset de treino por limitações de memória
#Ele vai usar diretamente dentro do dataset de treino um train_test_split com tamanho de teste 0.3 criando sua base de teste
#Então ele não irá, inicialmente, treinar com a mesma base usada nos itens anteriores. Após isso, com o método finalize
#treinaremos com a base de treino (train_df) e poderemos testar nessa base des teste
train_df_exp, test_df_exp = train_test_split(df_exp2, test_size=0.3, random_state=18)

```

Agora temos uma parte muito importante, o chamado setup, no qual definimos diversos parâmetros. O motivo das escolhas realizadas e maiores informações podem ser obtidas nos comentários do notebook e na documentação da biblioteca.

```
#Fazendo o setup, selecionando a variável alvo e outras opções:
#O pycaret já separa o df entre teste e treino com o train_test_split e um test_size de 0.3 dentro do df
#Session_id é um número randômico para podermos posteriormente obter o mesmo resultado
#silent = True não exibe a janela de diálogo com os tipos das variáveis
#Há diversos outros hiperparâmetros para alterarmos e testarmos os resultados
#https://pycaret.org/classification/
clf_exp = setup(train_df_exp, target = 'CO_VIA', session_id=18, experiment_name='teste_exp',
                 silent=True)
```

Interessante observar todos os modelos disponíveis pela biblioteca e que eles são objetos da mesma biblioteca utilizada anteriormente para nossos modelos individuais de NB e RF, *sklearn*. Assim, o que a biblioteca *pycaret* faz é automatizar o uso da biblioteca *sklearn*, poupando tempo e permitindo o desenvolvimento mais rápido.

```
#Exibindo todos os modelos disponíveis
#Observe que são objetos da biblioteca sklearn, assim como os feitos manualmente nos itens anteriores
#com a vantagem de podermos testar, comparar e melhorar eles em boa parte automaticamente
models()
```

ID	Name	Reference	Turbo
lr	Logistic Regression	sklearn.linear_model._logistic.LogisticRegression	True
knn	K Neighbors Classifier	sklearn.neighbors._classification.KNeighborsCl...	True
nb	Naive Bayes	sklearn.naive_bayes.GaussianNB	True
dt	Decision Tree Classifier	sklearn.tree._classes.DecisionTreeClassifier	True
svm	SVM - Linear Kernel	sklearn.linear_model._stochastic_gradient.SGDC...	True
rbfsvm	SVM - Radial Kernel	sklearn.svm._classes.SVC	False
gpc	Gaussian Process Classifier	sklearn.gaussian_process._gpc.GaussianProcessC...	False
mlp	MLP Classifier	sklearn.neural_network._multilayer_perceptron....	False
ridge	Ridge Classifier	sklearn.linear_model._ridge.RidgeClassifier	True
rf	Random Forest Classifier	sklearn.ensemble._forest.RandomForestClassifier	True
qda	Quadratic Discriminant Analysis	sklearn.discriminant_analysis.QuadraticDiscrim...	True
ada	Ada Boost Classifier	sklearn.ensemble._weight_boosting.AdaBoostClas...	True
gbc	Gradient Boosting Classifier	sklearn.ensemble._gb.GradientBoostingClassifier	True
lda	Linear Discriminant Analysis	sklearn.discriminant_analysis.LinearDiscrimina...	True
et	Extra Trees Classifier	sklearn.ensemble._forest.ExtraTreesClassifier	True
lightgbm	Light Gradient Boosting Machine	lightgbm.sklearn.LGBMClassifier	True

Foge ao escopo desse trabalho explicar as minúcias de cada modelo. Comentários pontuais serão feitos posteriormente se necessários.

Testando agora todos os modelos disponíveis para a exportação:

```
#Testando vários modelos de uma vez e exibindo os resultados
#Estamos usando validação cruzada mas com fold = 2 para acelerar
compare_models(fold = 2)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
et	Extra Trees Classifier	0.7705	0.9040	0.7796	0.7695	0.7699	0.6486	0.6486	17.055
lightgbm	Light Gradient Boosting Machine	0.7705	0.9116	0.7872	0.7683	0.7677	0.6510	0.6522	8.300
rf	Random Forest Classifier	0.7691	0.9108	0.7798	0.7674	0.7680	0.6469	0.6471	20.190
gbc	Gradient Boosting Classifier	0.7611	0.9030	0.7778	0.7584	0.7582	0.6364	0.6376	45.065
ada	Ada Boost Classifier	0.7383	0.8604	0.7549	0.7368	0.7356	0.6021	0.6036	6.440
dt	Decision Tree Classifier	0.7250	0.7849	0.7339	0.7251	0.7250	0.5782	0.5783	1.190
knn	K Neighbors Classifier	0.6962	0.8521	0.6995	0.6959	0.6957	0.5354	0.5356	6.170
lda	Linear Discriminant Analysis	0.4644	0.6091	0.4210	0.4962	0.4274	0.1447	0.1536	0.675
ridge	Ridge Classifier	0.4504	0.0000	0.3992	0.4361	0.3889	0.1172	0.1277	0.510
nb	Naive Bayes	0.4257	0.6133	0.3491	0.5339	0.2833	0.0272	0.0812	0.615
qda	Quadratic Discriminant Analysis	0.3720	0.6602	0.4151	0.5850	0.2977	0.1071	0.1420	0.835
lr	Logistic Regression	0.3373	0.4928	0.3333	0.1138	0.1702	0.0000	0.0000	1.960
svm	SVM - Linear Kernel	0.2922	0.0000	0.3335	0.2560	0.1343	0.0002	0.0080	25.795

Observe que mesmo usando o mesmo `random_state = 18` os resultados dos modelos NB e RF diferem dos obtidos anteriormente. Isso ocorre pois a base de treino é diferente da anterior. Dividimos, assim como antes, a base com um `split` de 0.3 para teste mas, posteriormente, a biblioteca dividiu essa base de treino novamente com um `split` default de 0.3. Temos então uma base efetiva de treino de 49%. Posteriormente, iremos utilizar um método para treinar com o restante da base de treino e obter os mesmos tamanhos finais para as bases de treinamento e teste dos modelos anteriores (70% e 30%). Fizemos isso para manter o controle da base de testes para uso posterior nas previsões.

Feito isso, selecionamos dentre esses alguns algoritmos com bom desempenho e com lógicas de funcionamento diversas para tentarmos melhorá-los e, posteriormente, juntá-los na busca de um modelo final superior. Escolhemos os seguintes algoritmos: *Random Forest* (RF), *Light Gradient Boosting Machine* (LIGHTGBM), *AdaBoost* (ADA) e *K-Nearest Neighbors* (KNN). Desses, os três primeiros já são ensembles. Não é o objetivo desse trabalho explicar o funcionamento de cada algoritmo, mas, de modo geral, a ideia dos métodos de ensemble é combinar as previsões de vários estimadores de base construídos com um determinado algoritmo de aprendizagem, a fim de melhorar a generalização/robustez.

```
#Criando os modelos
#Dados os resultados da comparação anterior, estamos criando os melhores 4 modelos individuais
#e queremos depois tentar combinar modelos distintos
#Não pegamos o et e dt por termos já pego o rf e serem todos similares
#e não pegamos o gbc por já termos pego o lighgbm e pelo alto TT (devagar)
#Queremos modelos de tipos diferentes para depois testar a combinação

#Random Forest Classifier
modelo_rf_exp = create_model('rf', fold = 2)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7685	0.9103	0.7794	0.7669	0.7675	0.6461	0.6462
1	0.7697	0.9113	0.7802	0.7679	0.7685	0.6477	0.6479
Mean	0.7691	0.9108	0.7798	0.7674	0.7680	0.6469	0.6471
SD	0.0006	0.0005	0.0004	0.0005	0.0005	0.0008	0.0008

```
#Light Gradient Boosting Machine
modelo_lightgbm_exp = create_model('lightgbm', fold = 2)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7700	0.9113	0.7867	0.7677	0.7672	0.6502	0.6514
1	0.7710	0.9119	0.7877	0.7688	0.7683	0.6518	0.6530
Mean	0.7705	0.9116	0.7872	0.7683	0.7677	0.6510	0.6522
SD	0.0005	0.0003	0.0005	0.0006	0.0005	0.0008	0.0008

```
#Ada Boost Classifier
modelo_ada_exp = create_model('ada', fold = 2)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7375	0.8618	0.7569	0.7379	0.7353	0.6021	0.6040
1	0.7391	0.8591	0.7529	0.7357	0.7359	0.6020	0.6031
Mean	0.7383	0.8604	0.7549	0.7368	0.7356	0.6021	0.6036
SD	0.0008	0.0013	0.0020	0.0011	0.0003	0.0000	0.0004

```
#K Neighbors Classifier
modelo_knn_exp = create_model('knn', fold = 2)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.6964	0.8523	0.6994	0.6960	0.6959	0.5355	0.5357
1	0.6961	0.8519	0.6995	0.6958	0.6955	0.5353	0.5355
Mean	0.6962	0.8521	0.6995	0.6959	0.6957	0.5354	0.5356
SD	0.0001	0.0002	0.0000	0.0001	0.0002	0.0001	0.0001

Depois de obtidos os modelos, iremos tunar seus parâmetros na busca de melhorias. Note que, por estarmos realizando poucas iterações (para acelerar o desenvolvimento), é possível que o modelo tunado tenha desempenho inferior ao modelo original, pois com poucas iterações não é possível testar muitas combinações de parâmetros. Por isso utilizamos o parâmetro `choose_better = true`. Caso o resultado pior, ele retorna o modelo original.

```
#Tunando os hiperparâmetros dos modelos na busca de melhores valores
#choose_better = true: caso o modelo tunado tenha performance pior, retorna o original
modelo_rf_exp = tune_model(modelo_rf_exp, fold = 2, choose_better = True)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7373	0.8877	0.7614	0.7430	0.7341	0.6045	0.6093
1	0.7363	0.8889	0.7593	0.7413	0.7333	0.6027	0.6071
Mean	0.7368	0.8883	0.7603	0.7422	0.7337	0.6036	0.6082
SD	0.0005	0.0006	0.0011	0.0009	0.0004	0.0009	0.0011

```
modelo_lightgbm_exp = tune_model(modelo_lightgbm_exp, fold = 2, choose_better = True)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7197	0.8739	0.7370	0.7134	0.7136	0.5723	0.5743
1	0.7184	0.8735	0.7352	0.7122	0.7122	0.5702	0.5724
Mean	0.7191	0.8737	0.7361	0.7128	0.7129	0.5712	0.5734
SD	0.0006	0.0002	0.0009	0.0006	0.0007	0.0010	0.0010

```
modelo_ada_exp = tune_model(modelo_ada_exp, fold = 2, choose_better = True)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7481	0.8652	0.7677	0.7467	0.7455	0.6176	0.6190
1	0.7471	0.8671	0.7638	0.7432	0.7431	0.6148	0.6163
Mean	0.7476	0.8662	0.7657	0.7449	0.7443	0.6162	0.6177
SD	0.0005	0.0009	0.0020	0.0017	0.0012	0.0014	0.0014

```
modelo_knn_exp = tune_model(modelo_knn_exp, fold = 2, choose_better = True)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7262	0.8852	0.7355	0.7270	0.7253	0.5836	0.5846
1	0.7275	0.8858	0.7368	0.7284	0.7265	0.5857	0.5867
Mean	0.7269	0.8855	0.7361	0.7277	0.7259	0.5846	0.5857
SD	0.0007	0.0003	0.0006	0.0007	0.0006	0.0010	0.0011

Note que os dois primeiros modelos (RF e LIGHTGBM) pioraram com o *tunning* e os modelos ADA e KNN melhoraram.

Podemos então finalizar os modelos (treinando com a base de treino inteira de 70% do *dataframe*) e salvá-los para uso posterior.

```
#Finaliza os modelos treinando com a base inteira, inclusive a usada para teste
modelo_rf_exp = finalize_model(modelo_rf_exp)

modelo_lightgbm_exp = finalize_model(modelo_lightgbm_exp)

modelo_ada_exp = finalize_model(modelo_ada_exp)

modelo_knn_exp = finalize_model(modelo_knn_exp)

#Salva os modelos para uso posterior
save_model(modelo_rf_exp, 'f_rf_exp')
save_model(modelo_lightgbm_exp, 'f_lightgbm_exp')
save_model(modelo_ada_exp, 'f_ada_exp')
save_model(modelo_knn_exp, 'f_knn_exp')
```

Feito isso, iremos criar o nosso *ensemble*. O método de combinação *blend_models* usa um consenso entre estimadores para gerar previsões finais. A ideia por trás da combinação é combinar diferentes algoritmos de aprendizado de máquina e usar o voto da maioria ou as probabilidades médias previstas para prever o resultado pois alguns modelos têm um desempenho melhor e alguns modelos têm um desempenho pior em diferentes subconjuntos dos dados. O método aceita duas opções para o método de junção de modelos: *soft* e *hard*. Nesse caso, usamos o parâmetro *hard*, que usa os rótulos previstos como critério de escolha dos modelos, pois esse método foi capaz de obter melhores resultados nesse *dataframe*.

```
#Criação de ensembles: combinação de modelos pode ser útil para melhoria e generalização do modelo.
#Uma grande vantagem de ensembles é que pode tornar o modelo mais robusto, com previsões de diferentes tipos de algoritmos.
blender_hard_exp = blend_models(estimator_list = [modelo_rf_exp,
                                                    modelo_lightgbm_exp,
                                                    modelo_ada_exp,
                                                    modelo_knn_exp,
                                                   ],
                                   method = 'hard', fold = 5, choose_better = True)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7818	0.0	0.7906	0.7796	0.7800	0.6656	0.6662
1	0.7831	0.0	0.7915	0.7808	0.7807	0.6674	0.6685
2	0.7802	0.0	0.7889	0.7777	0.7775	0.6628	0.6641
3	0.7829	0.0	0.7913	0.7807	0.7807	0.6671	0.6681
4	0.7824	0.0	0.7908	0.7802	0.7801	0.6662	0.6673
Mean	0.7821	0.0	0.7906	0.7798	0.7798	0.6658	0.6668
SD	0.0011	0.0	0.0009	0.0011	0.0012	0.0016	0.0016

Obtemos uma acurácia média de 0,7821, uma melhoria de aproximadamente 1% em relação ao melhor modelo anterior, o LIGHTGBM (*Light Gradient Boosting Machine*) de 0,7705.

Por fim, salvamos o modelo

```
save_model(blender_hard_exp, 'f_blender_hard_exp')
```

5.4. Demais Modelos na Importação

O mesmo procedimento que foi feito na exportação será feito aqui. Comentários pontuais serão feitos quando houver necessidade.

```
#Separando a base entre teste e treino
#O pycaret não vai utilizar esse dataset de test
#Ele vai usar diretamente dentro do dataset de treino um train_test_split com tamanho de teste 0.3 criando sua base de teste
#Então ele não irá, inicialmente, treinar com a mesma base usada nos itens anteriores. Após isso, com o método finalize
#treinaremos com a base de treino (train_df) e poderemos testar nessa base des teste
train_df_imp, test_df_imp = train_test_split(df_imp2, test_size=0.3, random_state=18)

#Fazendo o setup, selecionando a variável alvo e outras opções:
#O pycaret já separa o df entre teste e treino com o train_test_split e um test_size de 0.3 dentro do df
#Session_id é um número randômico para podermos posteriormente obter o mesmo resultado
#silent = True não exibe a janela de diálogo com os tipos das variáveis
#Há diversos outros hiperparâmetros para alterarmos e testarmos os resultados
#https://pycaret.org/classification/
clf_imp = setup(train_df_imp, target = 'CO_VIA', session_id=18, experiment_name='teste_imp',
                 silent=True)
```

```
#Testando vários modelos de uma vez e exibindo os resultados
#Estamos usando validação cruzada mas com fold = 2 para acelerar
compare_models(fold = 2)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
et	Extra Trees Classifier	0.7795	0.8620	0.7900	0.7679	0.7788	0.5590	0.5592	55.900
lightgbm	Light Gradient Boosting Machine	0.7786	0.8645	0.8335	0.7459	0.7873	0.5580	0.5618	6.635
gbc	Gradient Boosting Classifier	0.7732	0.8581	0.8334	0.7386	0.7831	0.5472	0.5516	34.320
rf	Random Forest Classifier	0.7705	0.8573	0.7921	0.7535	0.7723	0.5412	0.5419	62.995
ada	Ada Boost Classifier	0.7678	0.8513	0.8300	0.7329	0.7784	0.5366	0.5412	11.835
lr	Logistic Regression	0.7186	0.7888	0.9576	0.6436	0.7698	0.4416	0.5021	2.270
knn	K Neighbors Classifier	0.7172	0.7855	0.7638	0.6924	0.7263	0.4351	0.4374	14.645
dt	Decision Tree Classifier	0.7097	0.7101	0.7000	0.7066	0.7033	0.4192	0.4193	2.755
qda	Quadratic Discriminant Analysis	0.6107	0.6984	0.9951	0.5583	0.7153	0.2313	0.3554	1.220
nb	Naive Bayes	0.6038	0.6976	0.9955	0.5539	0.7118	0.2180	0.3438	1.170
ridge	Ridge Classifier	0.5687	0.0000	0.6954	0.5482	0.6131	0.1410	0.1461	1.045
lda	Linear Discriminant Analysis	0.5687	0.5888	0.6954	0.5482	0.6131	0.1410	0.1461	1.280
svm	SVM - Linear Kernel	0.5509	0.0000	0.4999	0.7298	0.3550	0.0987	0.1764	41.820

Observando a acurácia obtida para os modelos e a similaridade da lógica de funcionamento de alguns deles, escolhemos dessa vez os mesmo modelos da exportação com um modelo a mais, a *Logistic Regression* (LR), que nesse caso obteve acurácia superior a 70%. Criando os modelos:

```
#Criando os modelos
#Dados os resultados da comparação anterior, estamos criando os 5 melhores modelos individuais
#e queremos depois tentar combinar modelos distintos
#Não pegamos o et e dt por termos já pego o rf e serem todos similares
#e não pegamos o gbc por já termos pego o lightgbm e pelo alto TT (devagar)
#Todos acima de 70% de acurácia e queremos modelos de tipos diferentes para depois testar a combinação

#Random Forest Classifier
modelo_rf_imp = create_model('rf', fold = 2)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7700	0.8568	0.7899	0.7539	0.7715	0.5403	0.5409
1	0.7709	0.8578	0.7943	0.7530	0.7731	0.5421	0.5429
Mean	0.7705	0.8573	0.7921	0.7535	0.7723	0.5412	0.5419
SD	0.0004	0.0005	0.0022	0.0004	0.0008	0.0009	0.0010

```
#Light Gradient Boosting Machine
modelo_lightgbm_imp = create_model('lightgbm', fold = 2)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7787	0.8644	0.8331	0.7461	0.7872	0.5581	0.5618
1	0.7786	0.8647	0.8340	0.7456	0.7873	0.5580	0.5618
Mean	0.7786	0.8645	0.8335	0.7459	0.7873	0.5580	0.5618
SD	0.0000	0.0001	0.0004	0.0003	0.0000	0.0001	0.0000

```
#Ada Boost Classifier
modelo_ada_imp = create_model('ada', fold = 2)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7674	0.8508	0.8273	0.7334	0.7775	0.5356	0.5399
1	0.7683	0.8518	0.8328	0.7324	0.7794	0.5375	0.5425
Mean	0.7678	0.8513	0.8300	0.7329	0.7784	0.5366	0.5412
SD	0.0005	0.0005	0.0027	0.0005	0.0009	0.0010	0.0013

```
] #K Neighbors Classifier
modelo_knn_imp = create_model('knn', fold = 2)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7168	0.7853	0.7630	0.6922	0.7258	0.4344	0.4366
1	0.7175	0.7856	0.7645	0.6926	0.7268	0.4359	0.4382
Mean	0.7172	0.7855	0.7638	0.6924	0.7263	0.4351	0.4374
SD	0.0004	0.0002	0.0008	0.0002	0.0005	0.0008	0.0008

```
#Logistic Regression
modelo_lr_imp = create_model('lr', fold = 2)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7182	0.7883	0.9576	0.6433	0.7696	0.4409	0.5015
1	0.7190	0.7892	0.9577	0.6439	0.7701	0.4424	0.5027
Mean	0.7186	0.7888	0.9576	0.6436	0.7698	0.4416	0.5021
SD	0.0004	0.0004	0.0000	0.0003	0.0002	0.0007	0.0006

Tunando os modelos:

```
#Tunando os hiperparâmetros dos modelos na busca de melhores valores
#choose_better = true: caso o modelo tunado tenha performance pior, retorna o original
modelo_rf_imp = tune_model(modelo_rf_imp, fold = 2, choose_better = True)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7718	0.8575	0.8436	0.7325	0.7842	0.5447	0.5508
1	0.7713	0.8572	0.8493	0.7297	0.7849	0.5438	0.5509
Mean	0.7716	0.8574	0.8465	0.7311	0.7846	0.5442	0.5509
SD	0.0002	0.0001	0.0028	0.0014	0.0004	0.0004	0.0001

```
modelo_lightgbm_imp = tune_model(modelo_lightgbm_imp, fold = 2, choose_better = True)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7824	0.8684	0.8341	0.7508	0.7902	0.5655	0.5689
1	0.7819	0.8688	0.8354	0.7495	0.7901	0.5645	0.5681
Mean	0.7822	0.8686	0.8347	0.7501	0.7902	0.5650	0.5685
SD	0.0003	0.0002	0.0007	0.0006	0.0001	0.0005	0.0004

```
modelo_ada_imp = tune_model(modelo_ada_imp, fold = 2, choose_better = True)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7718	0.8560	0.8370	0.7353	0.7828	0.5446	0.5497
1	0.7724	0.8566	0.8392	0.7351	0.7837	0.5457	0.5510
Mean	0.7721	0.8563	0.8381	0.7352	0.7833	0.5451	0.5503
SD	0.0003	0.0003	0.0011	0.0001	0.0004	0.0006	0.0007

```
modelo_knn_imp = tune_model(modelo_knn_imp, fold = 2, choose_better = True)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7367	0.8167	0.8287	0.6946	0.7557	0.4750	0.4838
1	0.7358	0.8175	0.8293	0.6933	0.7552	0.4733	0.4823
Mean	0.7363	0.8171	0.8290	0.6939	0.7555	0.4742	0.4830
SD	0.0004	0.0004	0.0003	0.0006	0.0002	0.0009	0.0007

```
modelo_lr_imp = tune_model(modelo_lr_imp, fold = 2, choose_better = True)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7182	0.7883	0.9576	0.6433	0.7696	0.4409	0.5015
1	0.7190	0.7892	0.9577	0.6439	0.7701	0.4424	0.5027
Mean	0.7186	0.7888	0.9576	0.6436	0.7698	0.4416	0.5021
SD	0.0004	0.0004	0.0000	0.0003	0.0002	0.0007	0.0006

Observe que, nesse caso, todos os modelos melhoraram ou mantiveram a mesma acurácia após o *tunning*. Finalizando os modelos (treinando com toda a base de treinamento de 70% do *dataframe*) e salvando para uso posterior.

```
#Finaliza os modelos treinando com a base inteira, inclusive a usada para teste
modelo_rf_imp = finalize_model(modelo_rf_imp)
```

```
modelo_lightgbm_imp = finalize_model(modelo_lightgbm_imp)
```

```
modelo_ada_imp = finalize_model(modelo_ada_imp)
```

```
modelo_knn_imp = finalize_model(modelo_knn_imp)
```

```
modelo_lr_imp = finalize_model(modelo_lr_imp)
```

```
#Salva os modelos para uso posterior
save_model(modelo_rf_imp, 'f_rf_imp')
save_model(modelo_lightgbm_imp, 'f_lightgbm_imp')
save_model(modelo_ada_imp, 'f_ada_imp')
save_model(modelo_knn_imp, 'f_knn_imp')
save_model(modelo_lr_imp, 'f_lr_imp')
```

Feito isso, podemos criar o *ensemble*. Nesse caso, usamos o parâmetro *soft* que usa as probabilidades de previsão como critério de escolha dos modelos pois esse método foi capaz de obter melhores resultados nesse *dataframe*:

```
#Criação de ensembles: combinação de modelos pode ser útil para melhoria e generalização do modelo.
#Uma grande vantagem de ensembles é que pode tornar o modelo mais robusto, com previsões de diferentes tipos de algoritmos.
blender_soft_imp = blend_models(estimator_list = [modelo_rf_imp,
                                                    modelo_lightgbm_imp,
                                                    modelo_ada_imp,
                                                    modelo_knn_imp,
                                                    modelo_lr_imp
                                                   ], method = 'soft', fold = 5, choose_better = True)
```

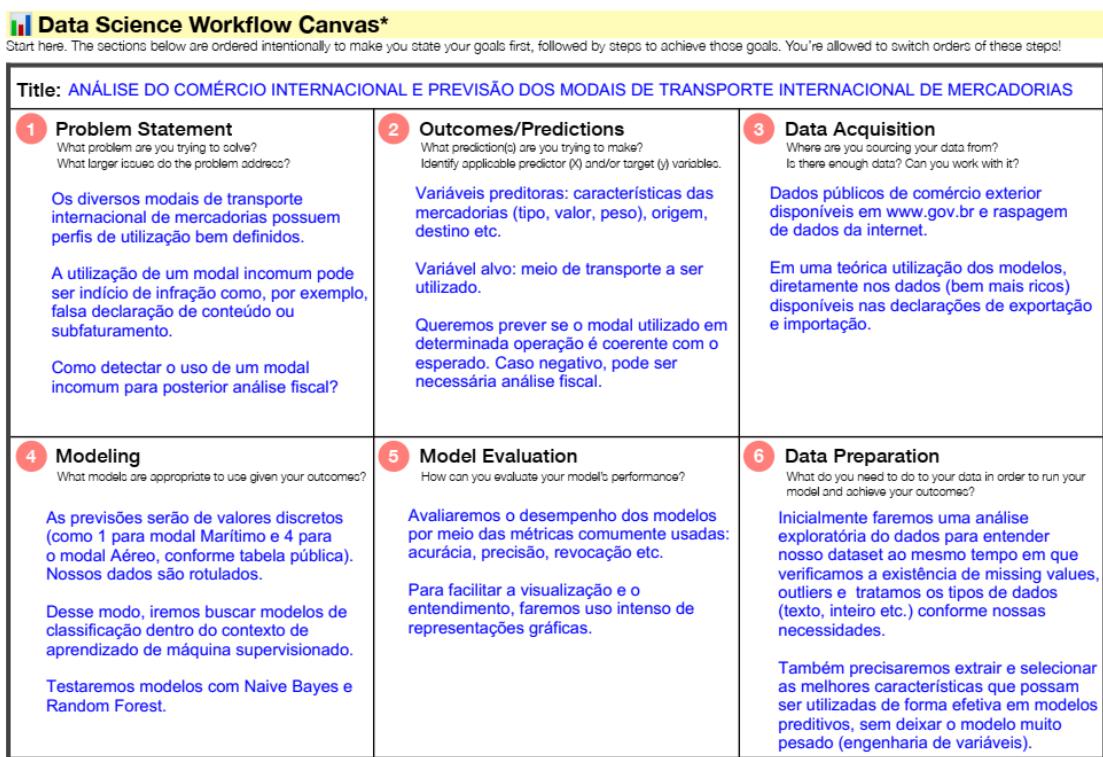
	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7804	0.8719	0.8918	0.7247	0.7996	0.5623	0.5772
1	0.7798	0.8717	0.8919	0.7240	0.7992	0.5612	0.5763
2	0.7803	0.8707	0.8940	0.7239	0.8000	0.5623	0.5778
3	0.7809	0.8727	0.8946	0.7244	0.8005	0.5635	0.5790
4	0.7797	0.8721	0.8932	0.7234	0.7994	0.5611	0.5765
Mean	0.7802	0.8718	0.8931	0.7241	0.7998	0.5621	0.5774
SD	0.0004	0.0007	0.0011	0.0004	0.0005	0.0009	0.0010

Vemos que, nesse caso, diferentemente da importação, o ensemble não conseguiu uma acurácia superior. Em verdade, teve um resultado inferior ao do modelo tunado LIGHTGBM (0,7802 contra 0,7822 do *Light Gradient Boosting Machine*). Por fim, salvamos o modelo:

```
save_model(blender_soft_imp, 'f_blender_soft_imp')
```

6. Apresentação dos Resultados

Para a execução desse trabalho, utilizamos o modelo *Canvas*, proposto por Vasandani (<https://towardsdatascience.com/a-data-science-workflow-canvas-to-kickstart-your-projects-db62556be4d0>) que pode ser visto abaixo. Ao trabalhar em um projeto de ciência de dados, você geralmente não tem um conjunto de instruções para alcançar um resultado predeterminado. Em vez disso, você deve determinar os resultados e as etapas para alcançá-los. É um processo iterativo. Esse fluxo de trabalho de ciência de dados foi projetado com esse processo em mente.



Conceptualized by Jasmine Vasandani using notes from General Assembly's Data Science Immersive. Formatted inspired by Business Model Canvas.

Activation

When you finish filling out the canvas above, now you can begin implementing your data science workflow in roughly this order.

- 1 Problem Statement → 2 Data Acquisition → 3 Data Prep → 4 Modeling → 5 Outcomes/Preds → 6 Model Eval

* Note: This canvas is intended to be used as a starting point for your data science projects. Data science workflows are typically nonlinear.

Na seção anterior já apresentamos parte dos resultados dos modelos, a acurácia obtida. Não era possível ser de outro modo, pois utilizamos a acurácia para servir como guia no desenvolvimento dos modelos.

Agora, com os modelos prontos, podemos verificar as demais medidas de desempenho e outros resultados importantes, assim como realizar previsões com os modelos, simulando a implementação deles. Não iremos prolongar as análises dos modelos NB pelo baixo desempenho obtido.

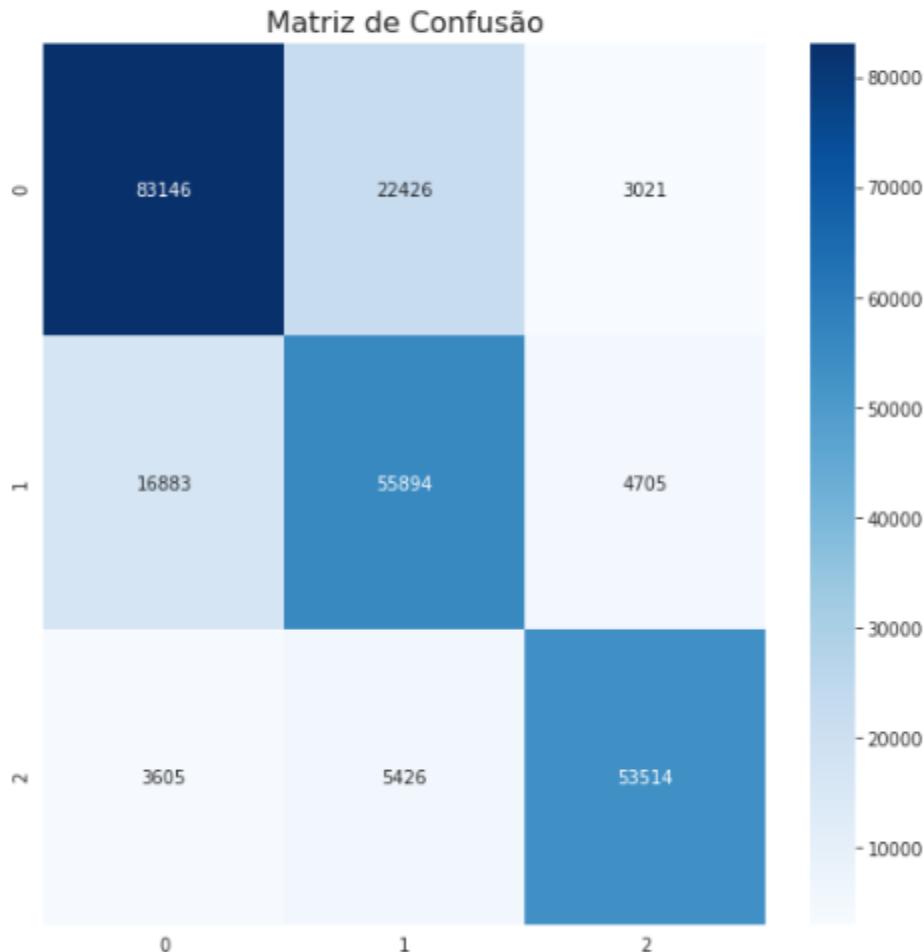
6.1. Random Forest (RF) na Exportação

Podemos plotar a matriz de confusão para o modelo RF. Utilizaremos essa matriz para visualizar a acurácia resultante desse modelo. Por definição, uma matriz de confusão C é tal que $C[i, j]$ é igual ao número de observações conhecidas no grupo i e previsto para estar no grupo j . Em outras palavras, o eixo vertical mostra as classes reais e o horizontal as classes previstas. Na diagonal principal temos os acertos.

Note que aqui temos uma particularidade nas representações gráficas que requer nossa atenção. Na exportação temos as classes 0 (CO_VIA = 1, MARITIMA), 1 (CO_VIA = 4, AEREA) e 2 (CO_VIA = 7, RODOVIÁRIA). Na importação temos as classes 0 (CO_VIA = 1, MARITIMA), 1 (CO_VIA = 4, AEREA).

```
#Plotando a matriz de confusão
cf_matrix = confusion_matrix(y_pred, y_test)
sns.heatmap(cf_matrix, annot=True, fmt='g', cmap='Blues')
plt.title('Matriz de Confusão', fontsize=16)
```

Text(0.5, 1.0, 'Matriz de Confusão')



Não é o objetivo desse trabalho definir todas as medidas de desempenho, principalmente para não tornar esse documento ainda mais longo. Contudo, brevemente, como vimos anteriormente, a acurácia indica uma performance geral do modelo: dentre todas as classificações, quantas o modelo classificou corretamente. A precisão mostra dentre todas as classificações de classe positivo que o modelo fez, quantas estão corretas. A revocação (*recall*) mostra dentre todas as situações de classe positivo como valor esperado, quantas estão corretas e o F1-Score é a média harmônica entre precisão e revocação.

$$Precisao = \frac{VP}{VP + FP}$$

$$Revocacao = \frac{VP}{VP + FN}$$

$$F1 - Score = \frac{2.Precisao \cdot Revocacao}{Precisao + Revocacao}$$

Onde:

- 1) VP são os verdadeiros positivos
- 2) VN são os verdadeiros negativos
- 3) FP são os falsos positivos
- 4) FN são os falsos negativos

```
#Computando outras métricas: Precision, Recall e F1
print('Precisão')
print(precision_score(y_test, y_pred, average='macro'))
print('Revocação')
print(recall_score(y_test, y_pred, average='macro'))
print('F1-score')
print(f1_score(y_test, y_pred, average='macro'))
```

```
Precisão
0.7808849023897412
Revocação
0.7811892706042055
F1-score
0.7805129755269594
```

Também podemos utilizar o método de validação cruzada no modelo, na busca de medida mais confiável sobre a capacidade do modelo de representar o processo gerador dos dados. Novamente, não entraremos em maiores detalhes para prolongar demais a discussão:

```
#Validação Cruzada
scores = cross_val_score(model, x_train, y_train, cv=5, scoring='f1_macro')
score = scores.sum()/5
score
```

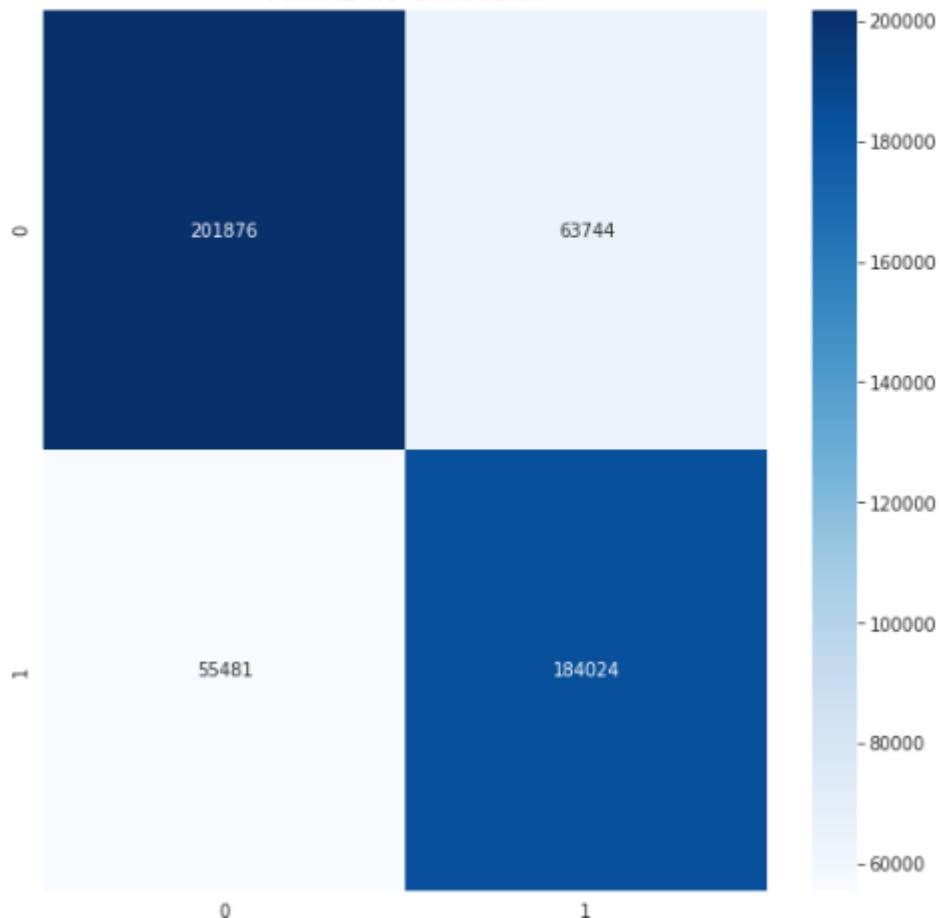
```
0.7765611444072007
```

6.2. Random Forest (RF) na Importação

Assim como no tópico anterior, podemos plotar a matriz de confusão e as demais métricas:

```
#Plotando a matriz de confusão
cf_matrix = confusion_matrix(y_pred, y_test)
sns.heatmap(cf_matrix, annot=True, fmt='g', cmap='Blues')
plt.title('Matriz de Confusão', fontsize=16)
```

Text(0.5, 1.0, 'Matriz de Confusão')
 Matriz de Confusão



```
#Computando outras métricas: Precision, Recall e F1
print('Precisão')
print(precision_score(y_test, y_pred, average='macro'))
print('Revocação')
print(recall_score(y_test, y_pred, average='macro'))
print('F1-score')
print(f1_score(y_test, y_pred, average='macro'))
```

Precisão
 0.7641847311699234
 Revocação
 0.7635735764869855
 F1-score
 0.7636741342824303

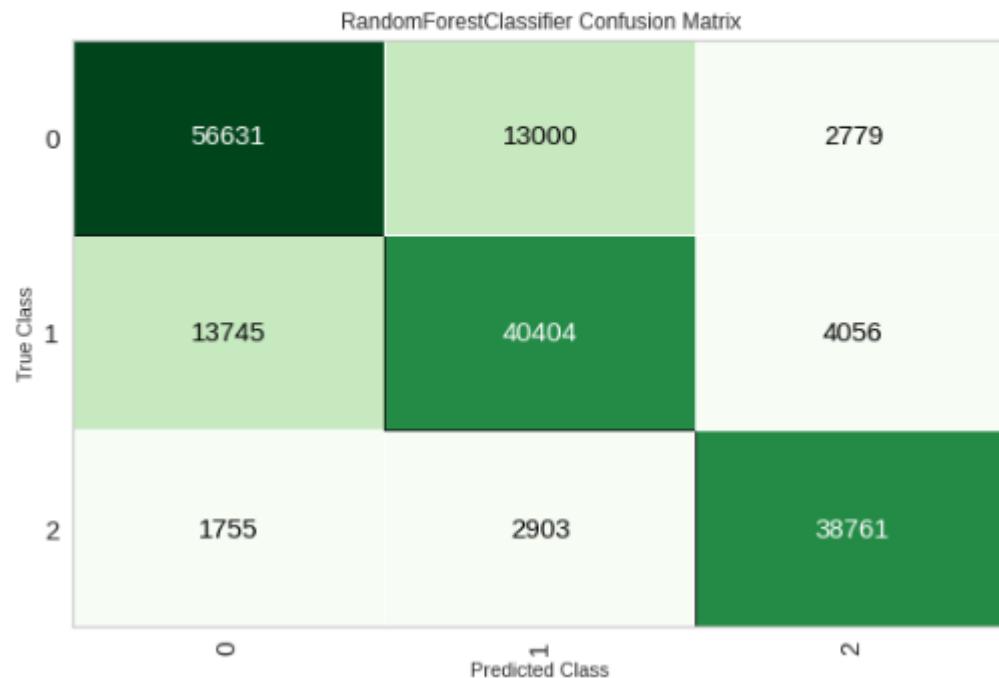
```
#Validação Cruzada
scores = cross_val_score(model, x_train, y_train, cv=5, scoring='f1_macro')
scores.sum()/5
```

0.7603682692511207

6.3. Demais Modelos na Exportação

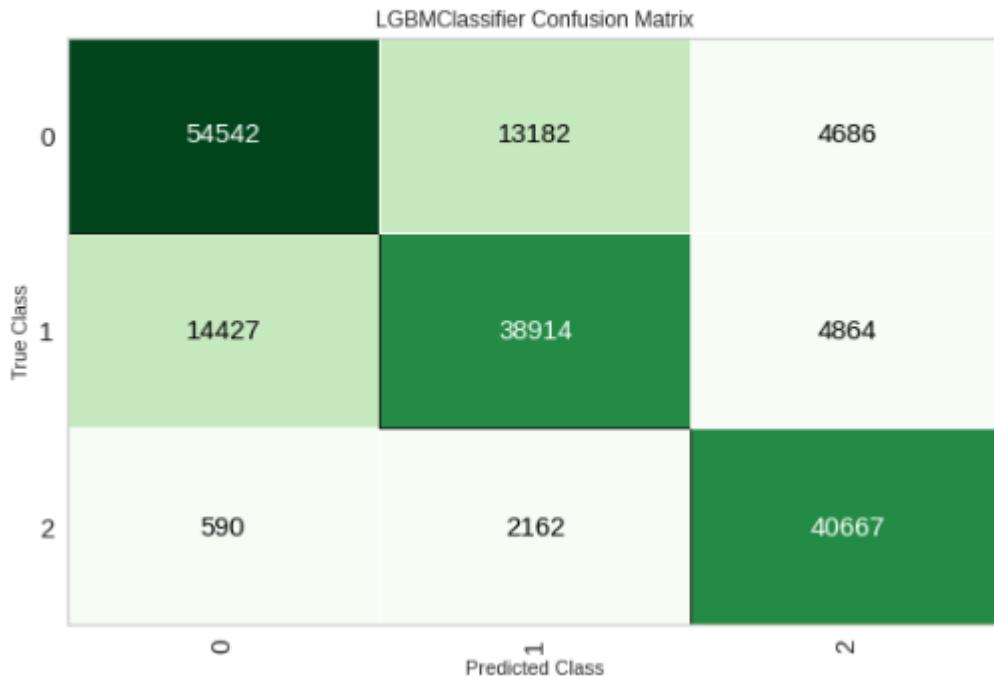
Agora, além da matriz de confusão já utilizada anteriormente, vejamos também para esses modelos o relatório de classificação (representação das principais métricas de classificação por classe) e a importância das variáveis (representação de quais variáveis são mais importantes para prever a variável alvo).

```
#Plota diversos gráficos para análise dos modelos
plot_model(modelo_rf_exp, plot = 'confusion_matrix')
```

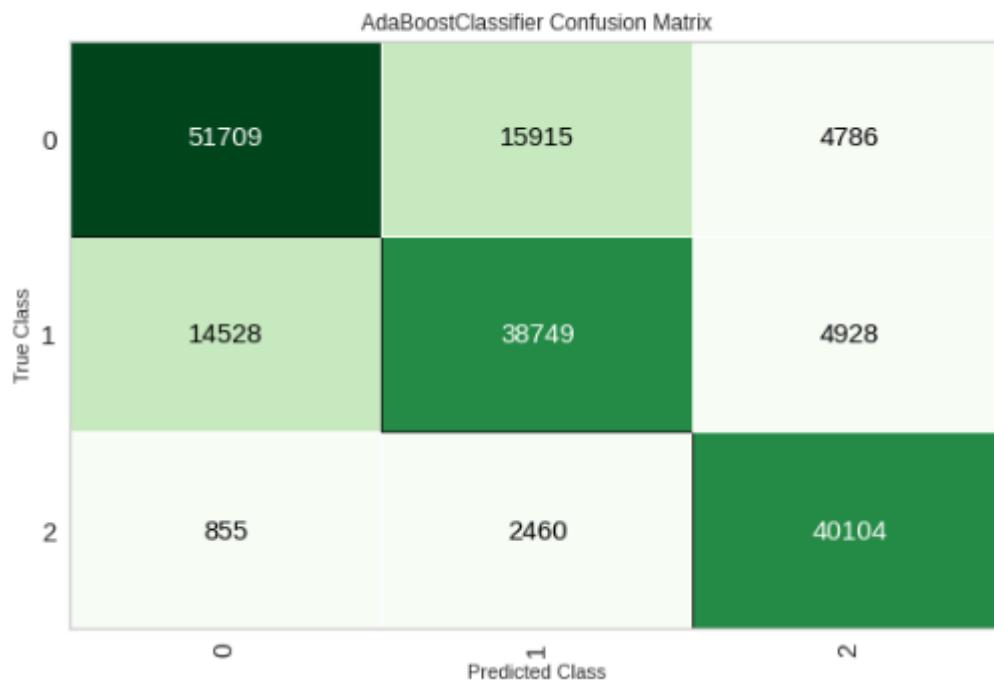


100

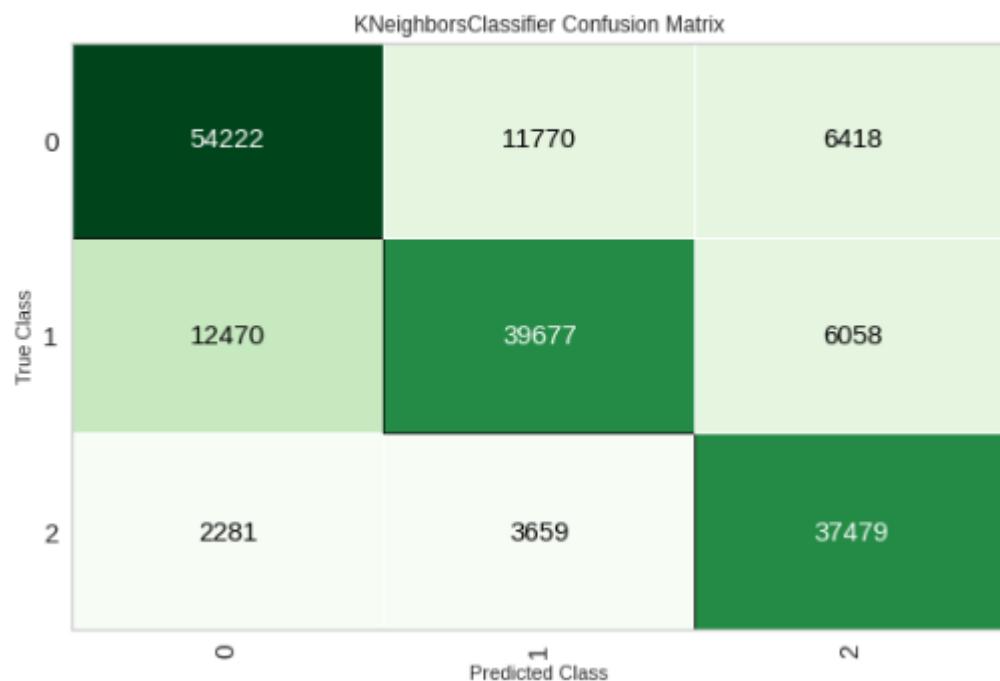
```
plot_model(modelo_lightgbm_exp, plot = 'confusion_matrix')
```



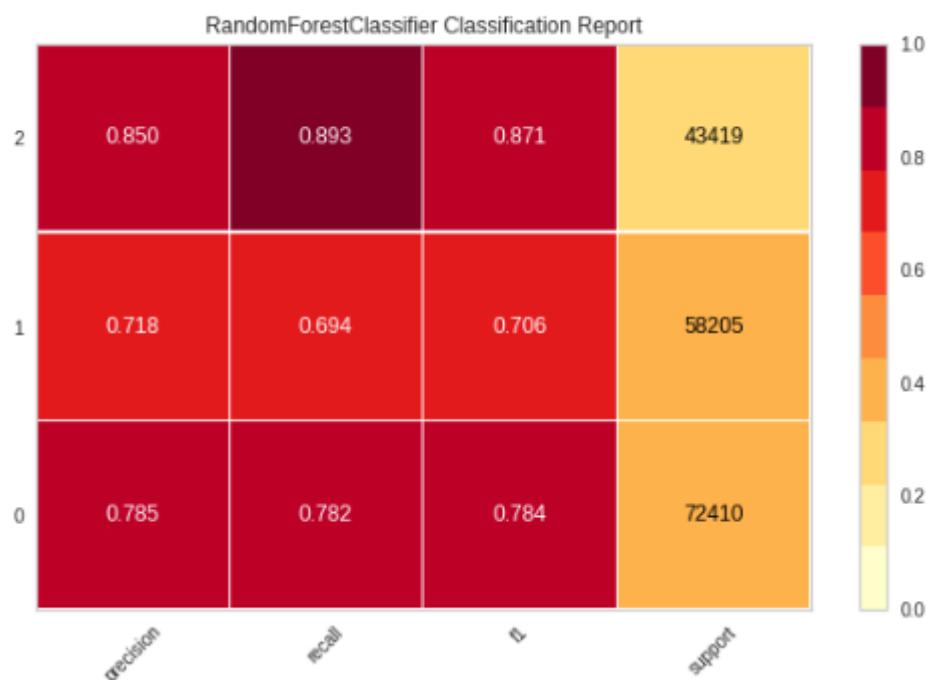
```
plot_model(modelo_ada_exp, plot = 'confusion_matrix')
```



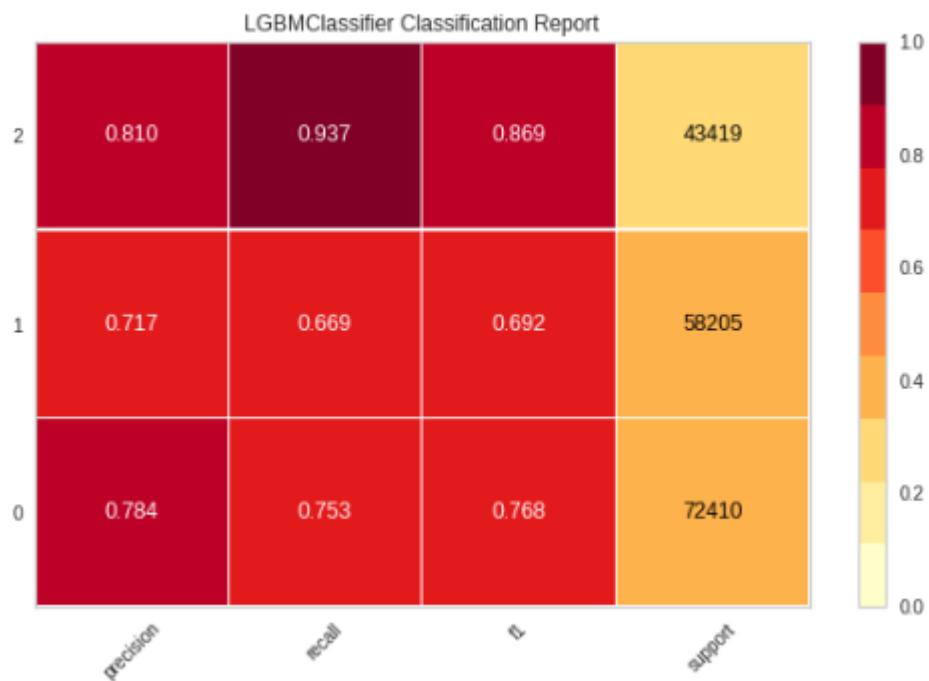
```
plot_model(modelo_knn_exp, plot = 'confusion_matrix')
```



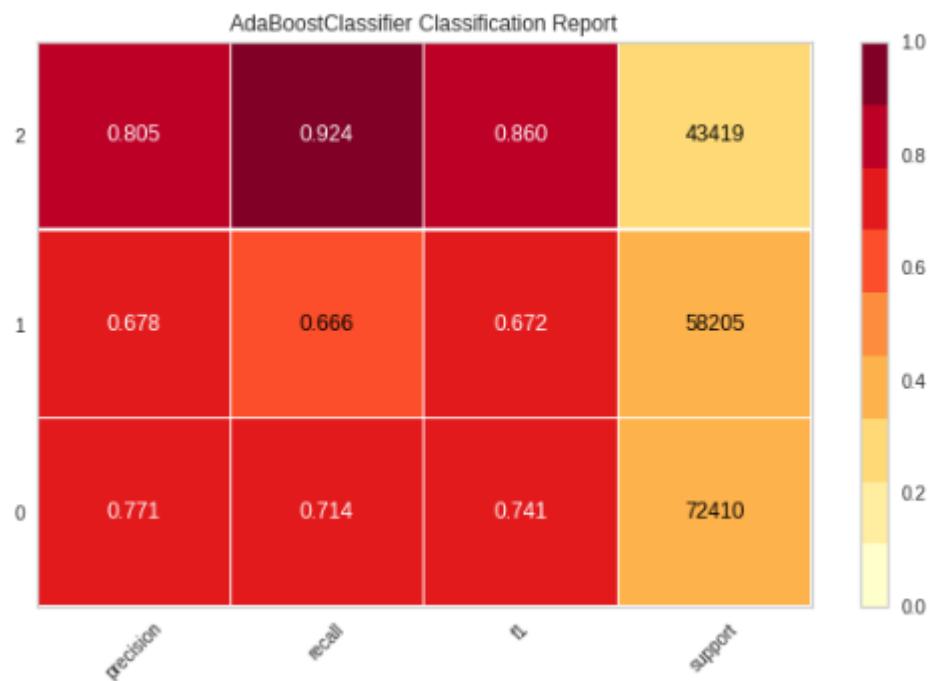
```
plot_model(modelo_rf_exp, plot = 'class_report')
```



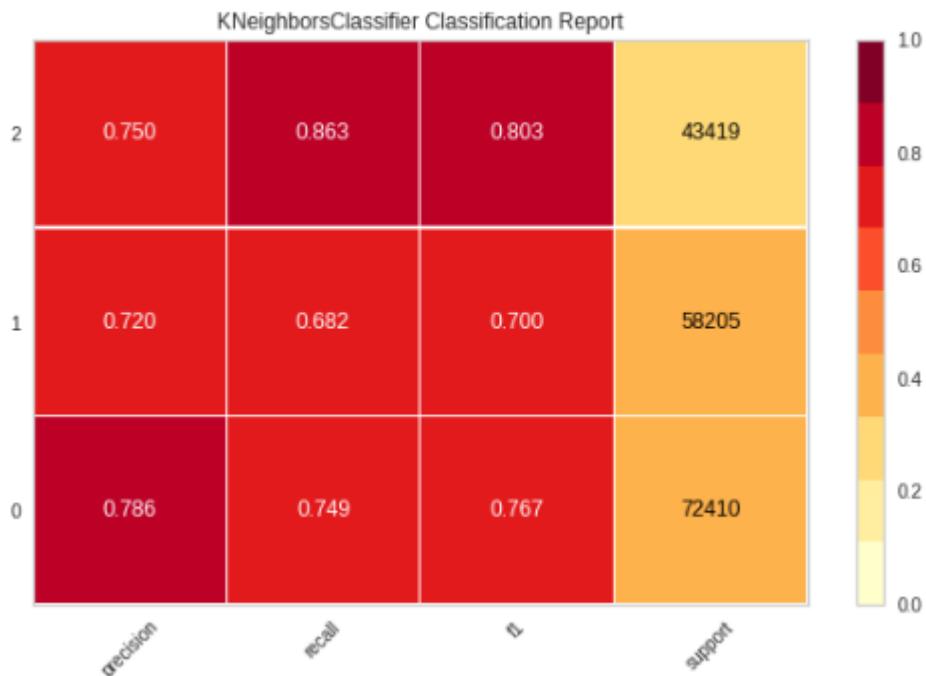
```
plot_model(modelo_lightgbm_exp, plot = 'class_report')
```



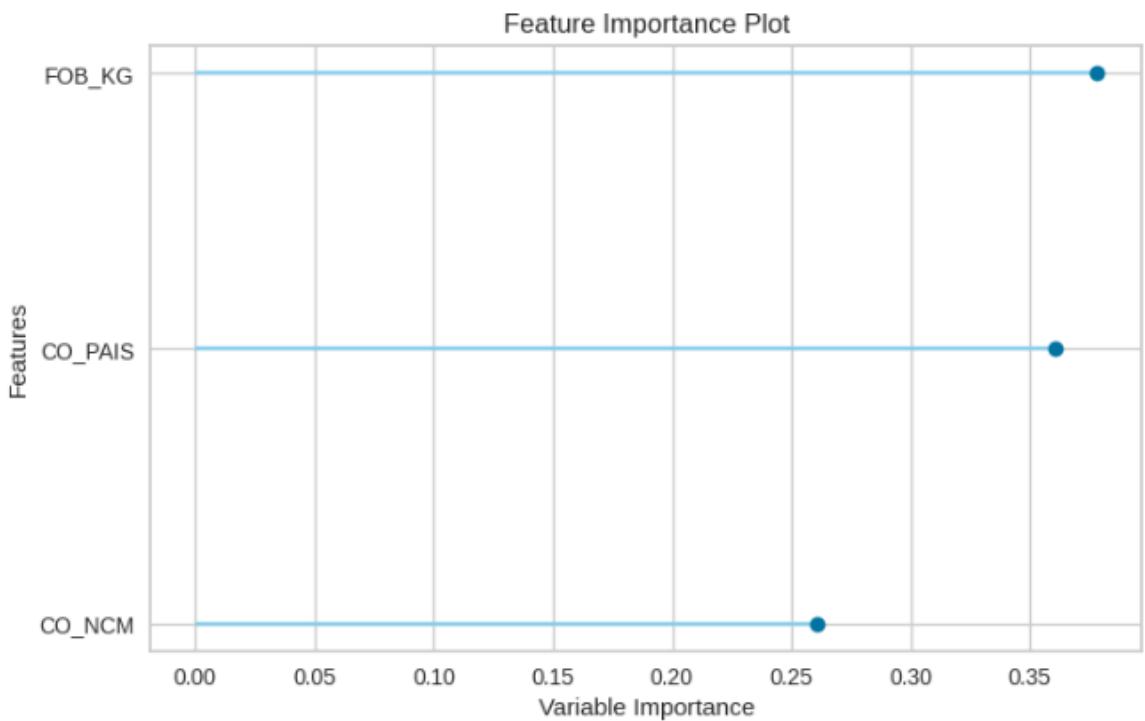
```
plot_model(modelo_ada_exp, plot = 'class_report')
```



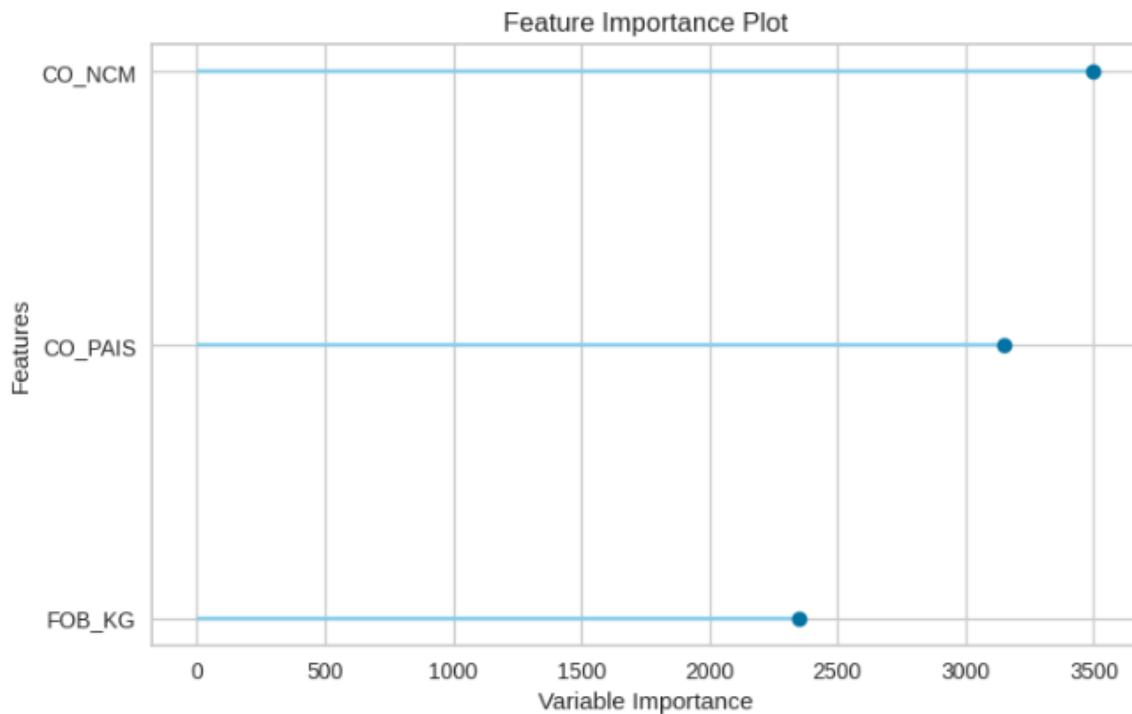
```
plot_model(modelo_knn_exp, plot = 'class_report')
```



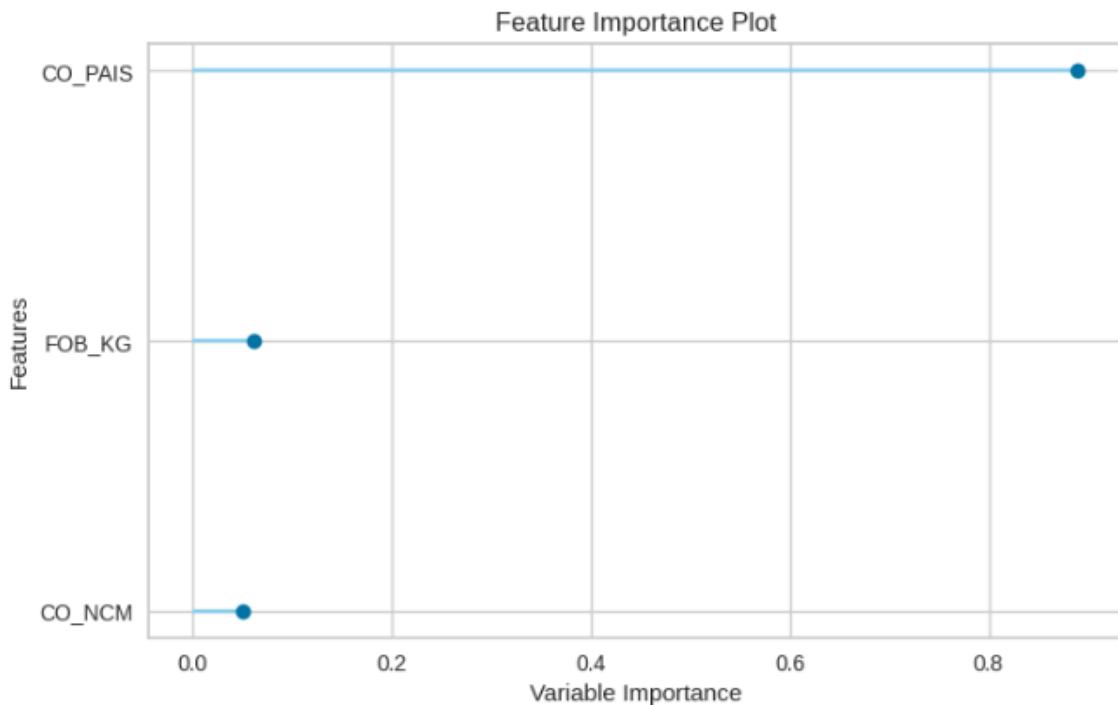
```
plot_model(modelo_rf_exp, plot = 'feature')
```



```
plot_model(modelo_lightgbm_exp, plot = 'feature')
```



```
plot_model(modelo_ada_exp, plot = 'feature')
```



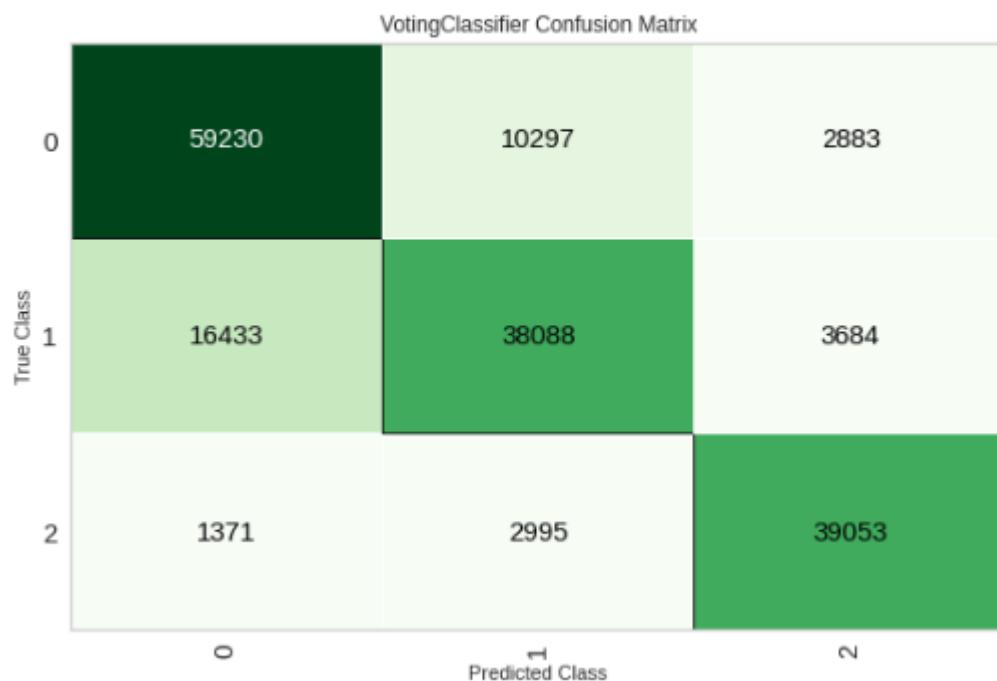
Curioso notar na matriz de confusão como determinados modelos são melhores para prever determinadas classes. Por exemplo, os modelos ADA e LIGHTGBM tem uma revocação maior na classe 2 (CO_VIA = 7, RODOVIARIO). E isso se reflete na matriz de confusão, pois esses modelos são os com o maior

número de verdadeiros positivos para essa classe, em comparação com os modelos RF e KNN. Isso indica que podemos obter alguma melhoria construindo um *ensemble* de modelos. Em contrapartida, se as previsões corretas dos modelos fossem muito próximas/correlacionadas, o efeito da combinação de modelos pode não ser muito relevante.

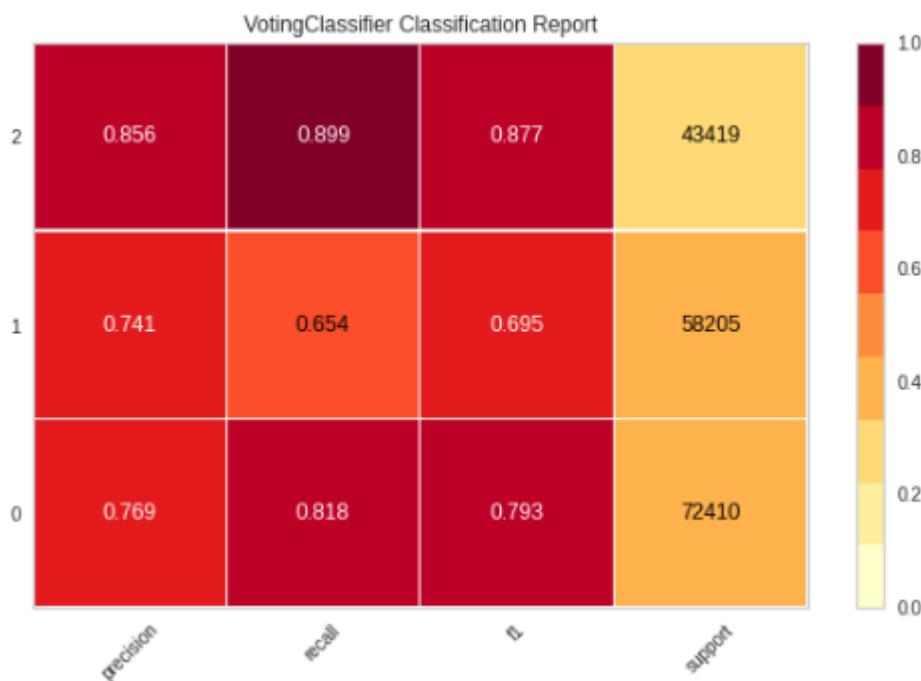
Também é interessante notar como determinados modelos, por terem lógicas completamente distintas para obtenção de seus resultados, dão importância dessemelhante para diferentes variáveis. O modelo RF tem como variáveis mais importantes FOB_KG e CO_PAIS, o LIGHTGBM as variáveis CO_NCM e CO_PAIS enquanto o ADA utiliza quase que exclusivamente a variável CO_PAIS para obter suas previsões. O modelo KNN não permite a visualização da importância das variáveis.

Verificando agora os resultados do *ensemble*:

```
#Mostrando a matriz de confusao
plot_model(blender_hard_exp, plot = 'confusion_matrix')
```



```
plot_model(blender_hard_exp, plot = 'class_report')
```



Finalizada a análise do desempenho dos modelos, vejamos as previsões. Como o modelo com melhor resultado foi o *ensemble*, faremos as previsões com ele:

```
#Usa os modelos nos dados não usados anteriormente, como seria feito no problema real
predictions_bh_exp = predict_model(blender_hard_exp, data = test_df_exp)
predictions_bh_exp
```

	CO_NCM	CO_PAIS	CO_VIA	FOB_KG	Label
213942	84262000	573	1	0.790014	1
100050	73182100	647	4	98.000000	4
354821	33011290	169	1	6.301541	1
480866	84186931	158	4	12.925698	4
1368319	30049099	196	4	124.616883	4

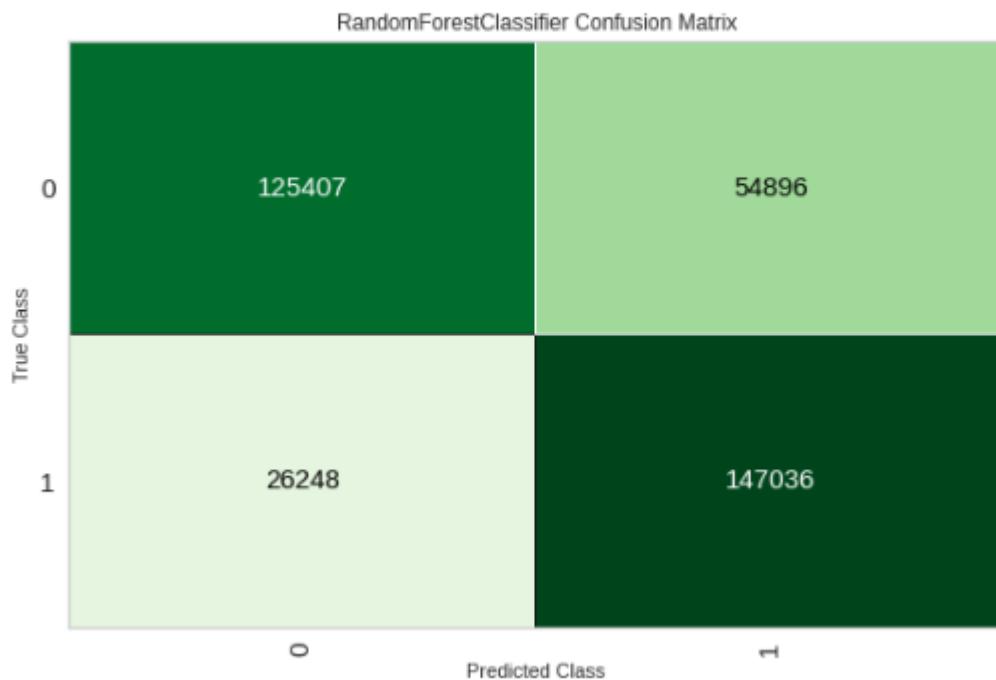
Onde CO_VIA é a classe real e *Label* é a classe prevista. Observe que dessas 5 previsões o resultado obtido foi igual ao esperado.

6.4. Demais Modelos na Importação

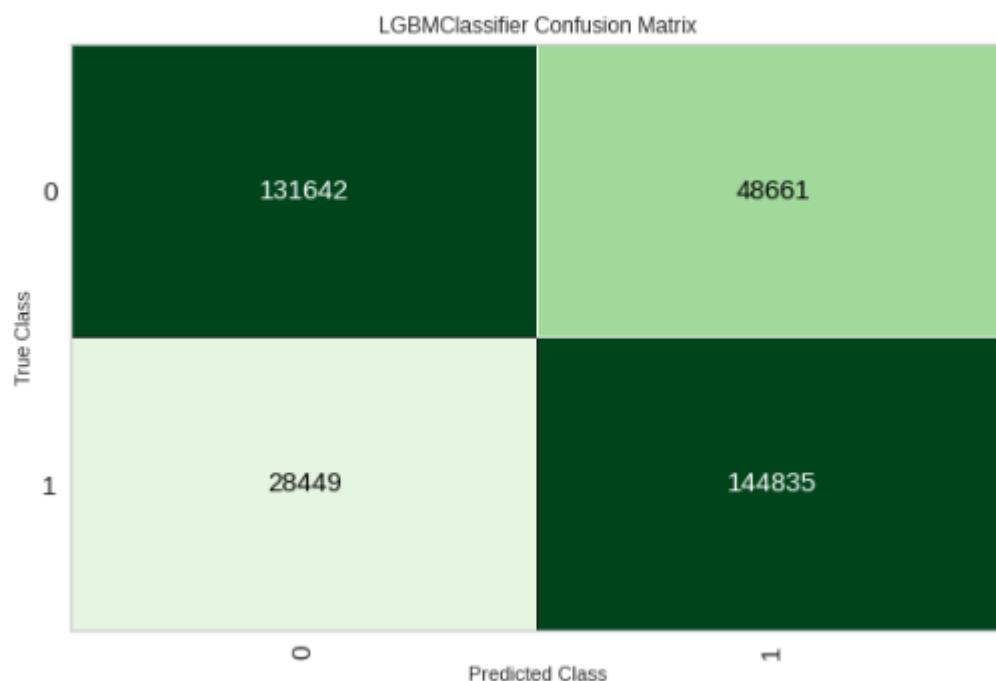
Plotando os mesmos gráficos escolhidos anteriormente, podemos analisar os modelos da importação:

```
#Plota diversos gráficos para análise dos modelos
```

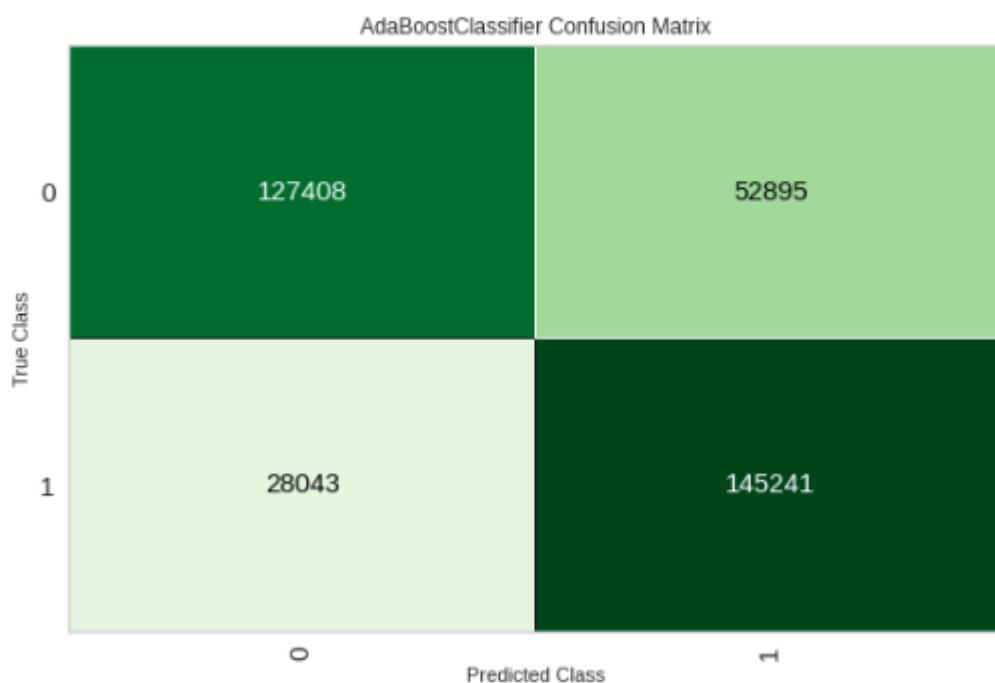
```
plot_model(modelo_rf_imp, plot = 'confusion_matrix')
```



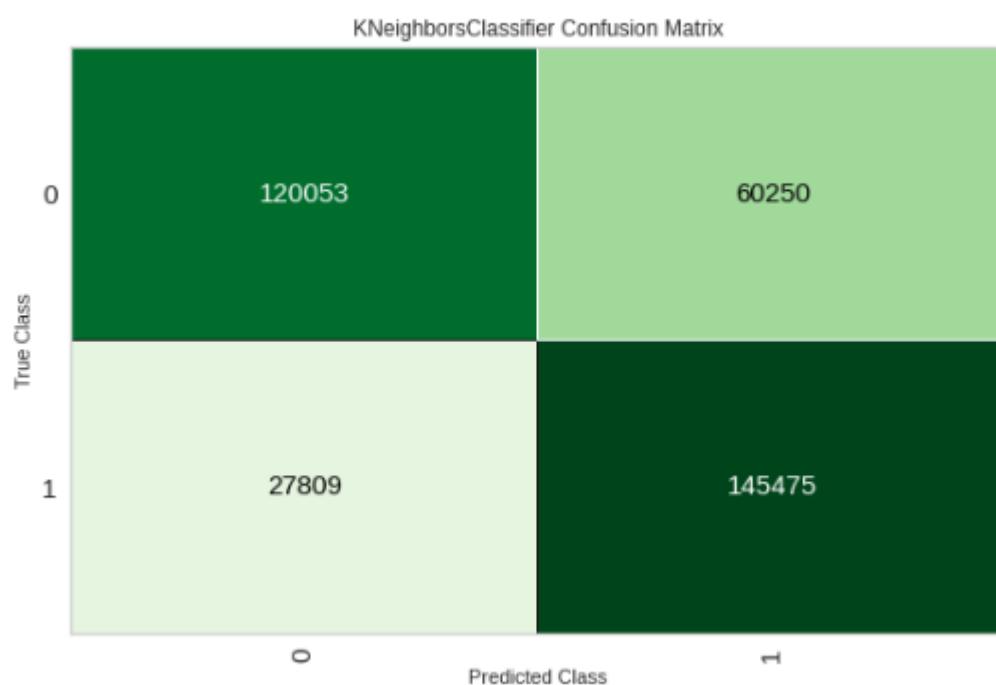
```
plot_model(modelo_lightgbm_imp, plot = 'confusion_matrix')
```



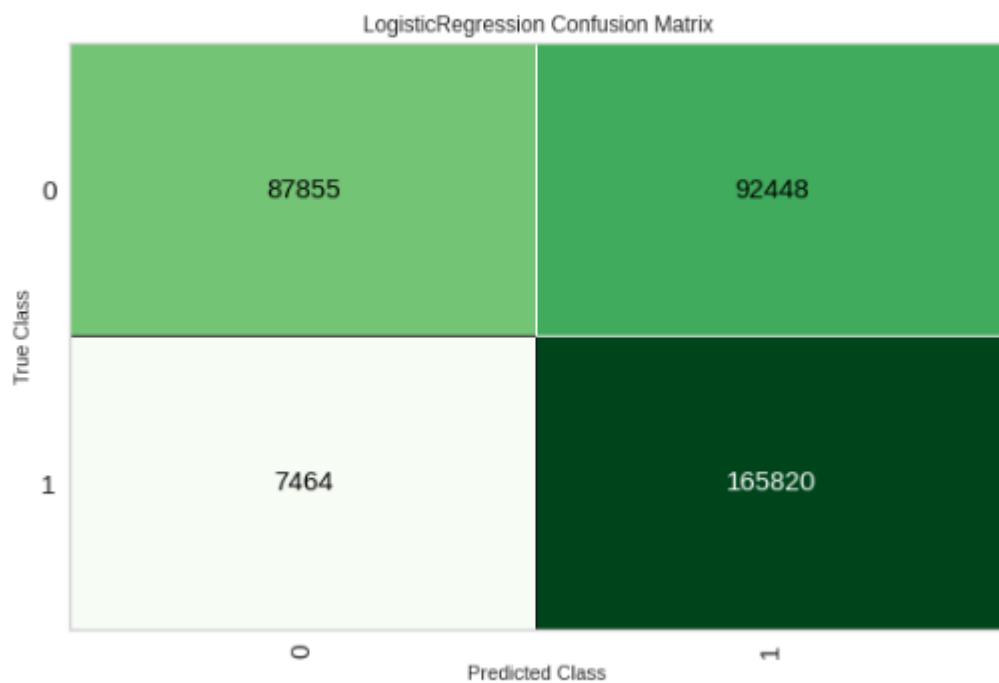
```
plot_model(modelo_ada_imp, plot = 'confusion_matrix')
```



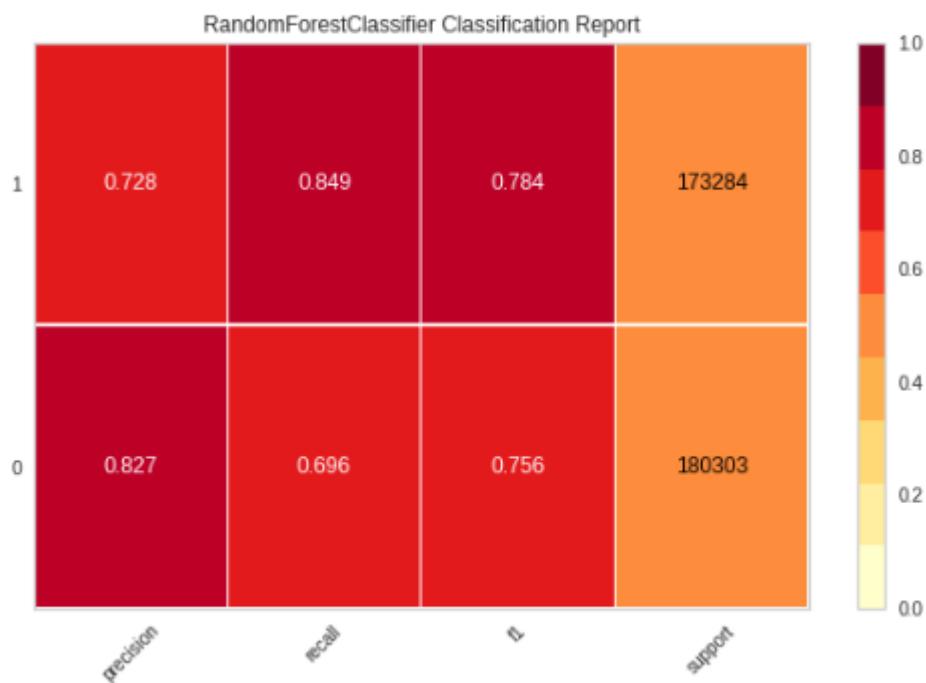
```
plot_model(modelo_knn_imp, plot = 'confusion_matrix')
```



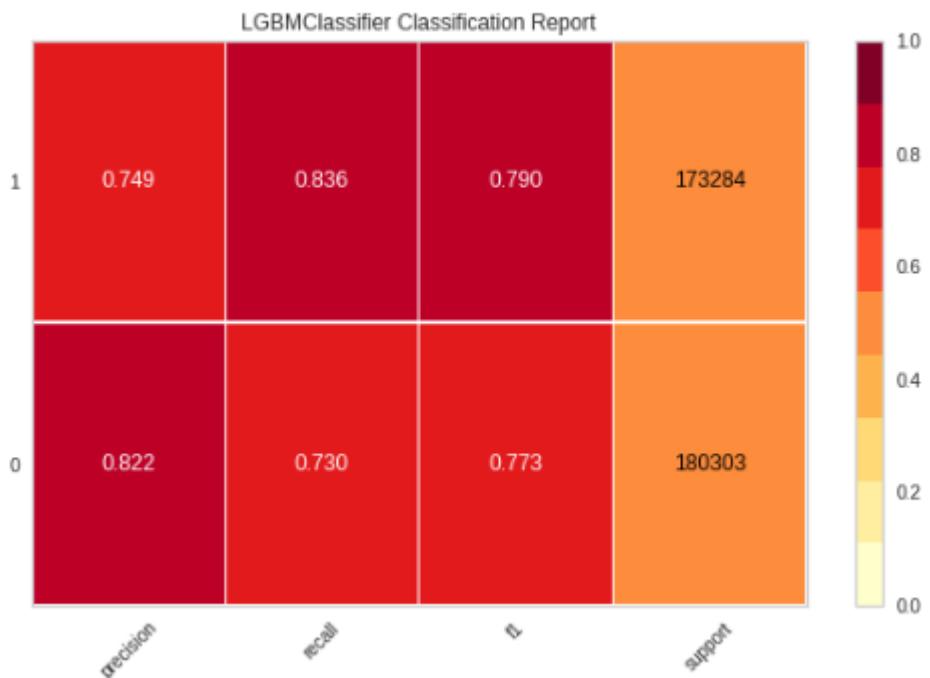
```
plot_model(modelo_lr_imp, plot = 'confusion_matrix')
```



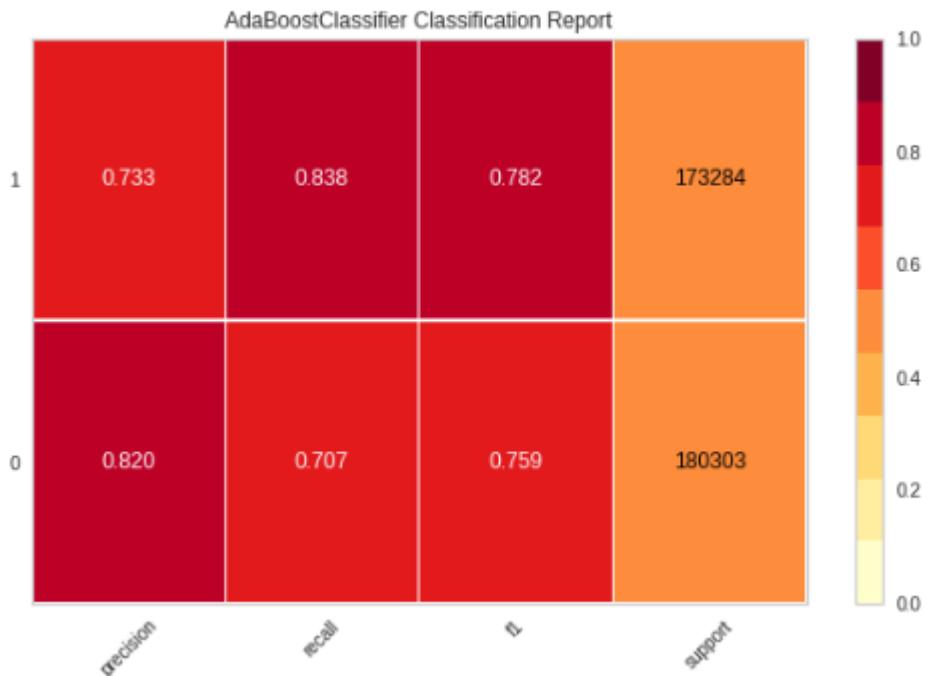
```
plot_model(modelo_rf_imp, plot = 'class_report')
```



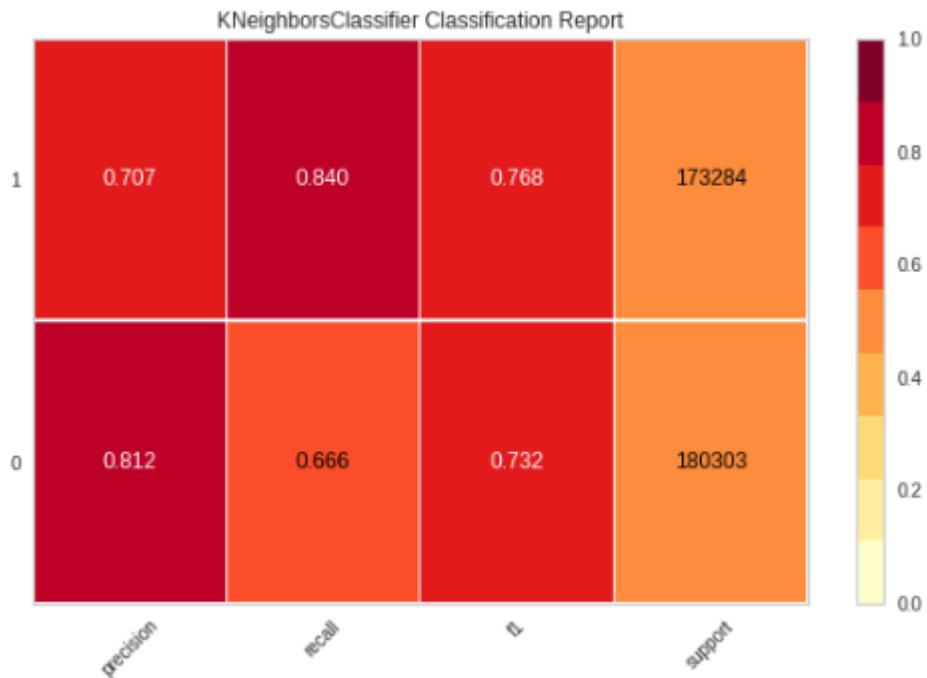
```
plot_model(modelo_lightgbm_imp, plot = 'class_report')
```



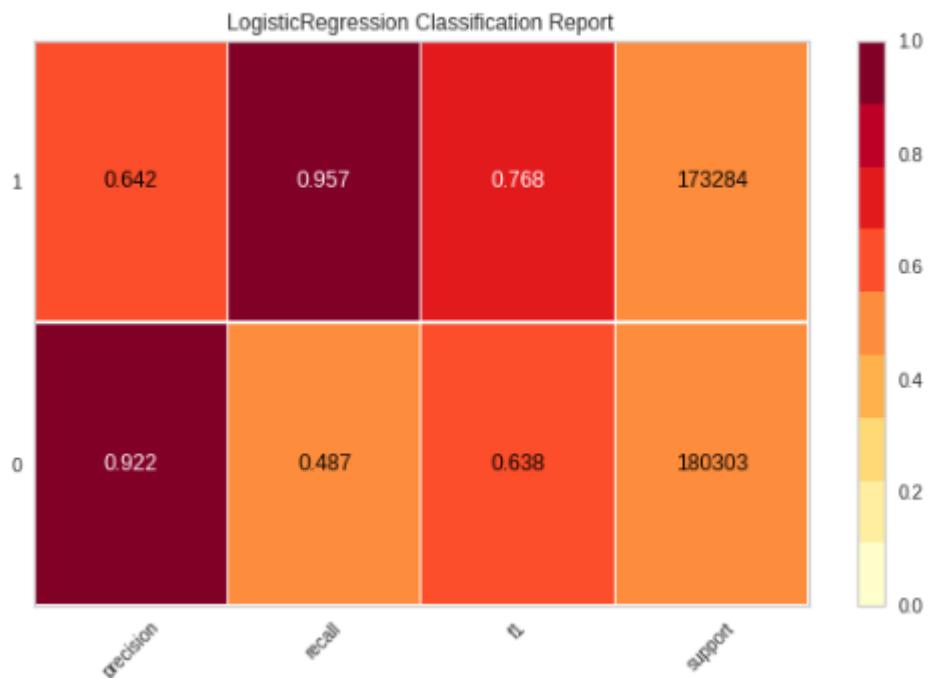
```
plot_model(modelo_ada_imp, plot = 'class_report')
```



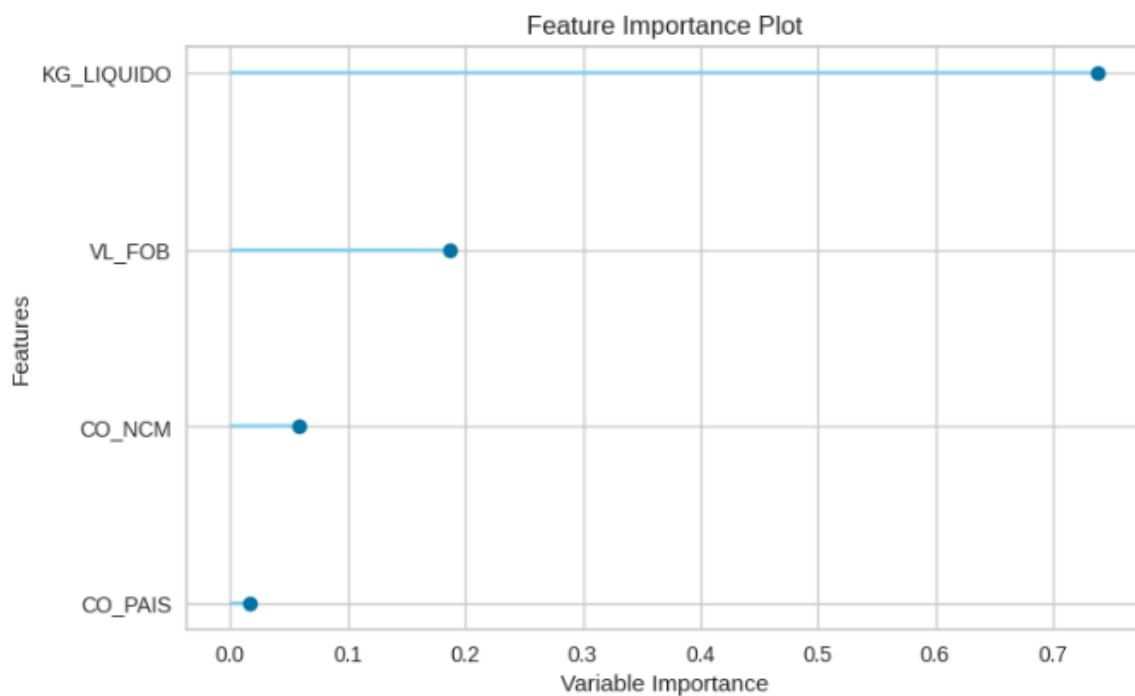
```
plot_model(modelo_knn_imp, plot = 'class_report')
```



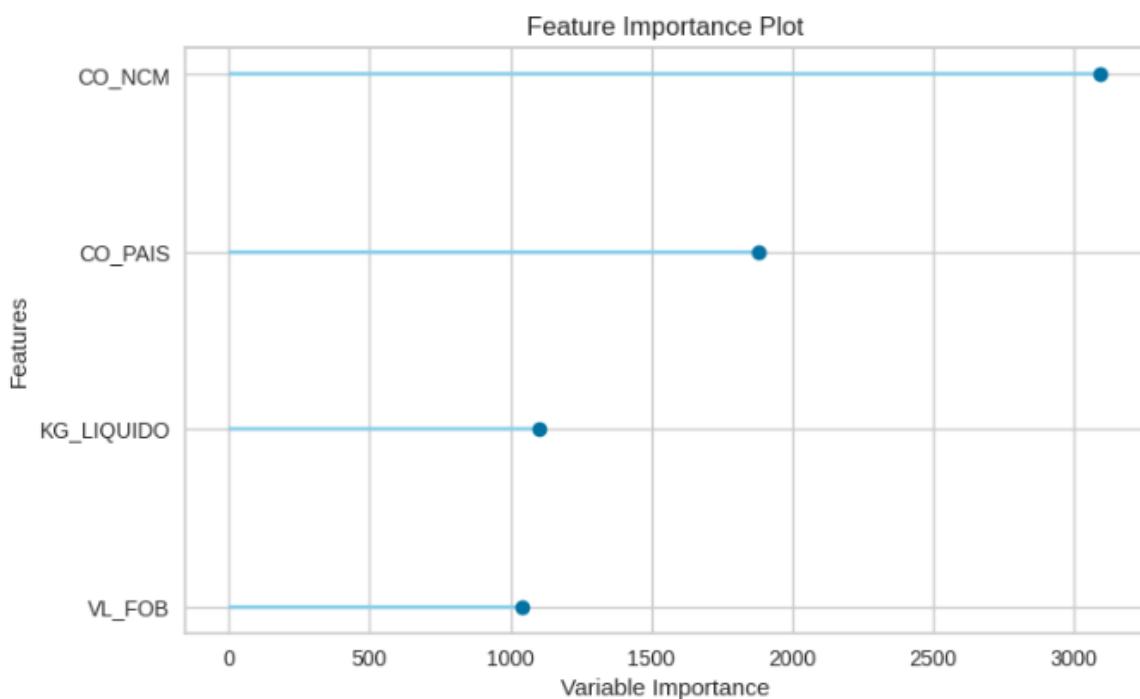
```
plot_model(modelo_lr_imp, plot = 'class_report')
```



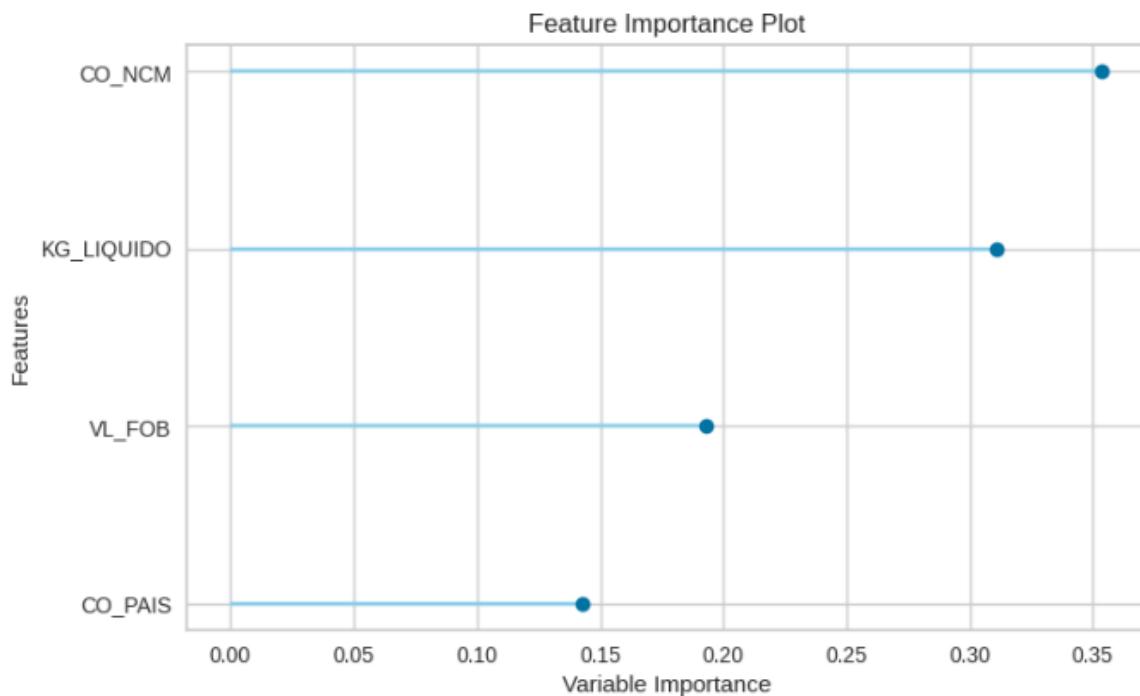
```
plot_model(modelo_rf_imp, plot = 'feature')
```



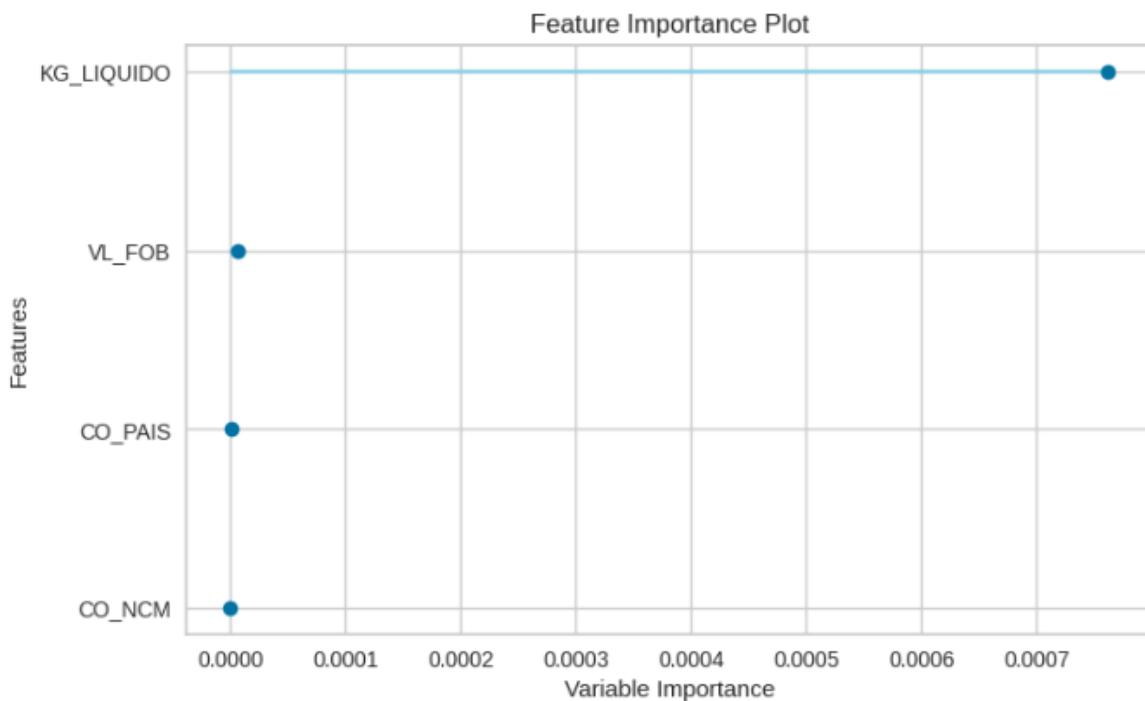
```
plot_model(modelo_lightgbm_imp, plot = 'feature')
```



```
plot_model(modelo_ada_imp, plot = 'feature')
```



```
plot_model(modelo_lr_imp, plot = 'feature')
```

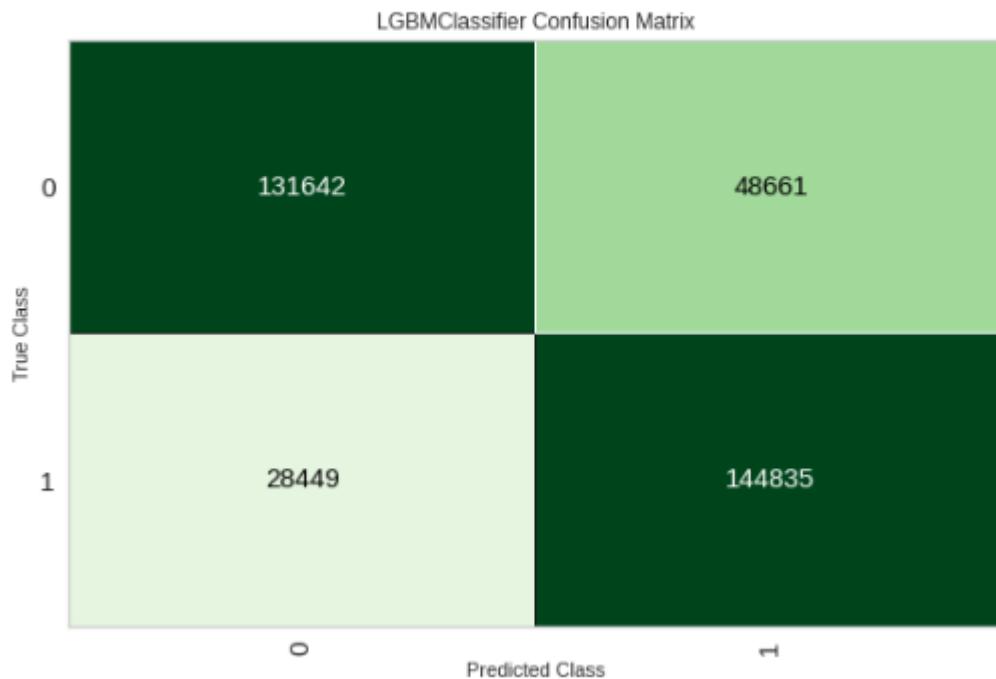


Novamente, podemos notar na matriz de confusão como determinados modelos são melhores para prever determinadas classes. O modelo LR (*Logistic Regression*), por exemplo, tem uma revocação de 95,7% para a classe 1 (CO_VIA = 4 - Aéreo).

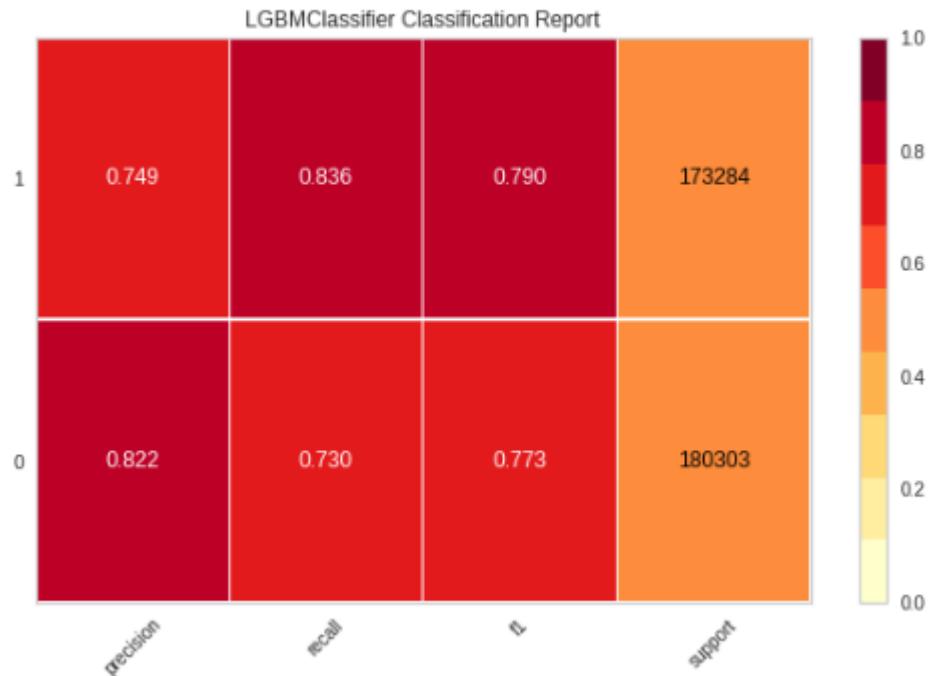
Os modelos RF e LR tem como variáveis mais importantes KG_LIQUIDO e VL_FOB, o modelo LIGHTGBM as variáveis CO_NCM e CO_PAIS enquanto o ADA as variáveis CO_NCM e KG_LIQUIDO. Os demais modelos não permitem a visualização da importância das variáveis.

Verificando agora os resultados do *ensemble*:

```
#Mostrando a matriz de confusao  
plot_model(blender_soft_imp, plot = 'confusion_matrix')
```



```
plot_model(blender_soft_imp, plot = 'class_report')
```



Nota-se que o *ensemble* tem um desempenho um pouco inferior ao do modelo individual com melhor desempenho (LIGHTGBM tunado). Isso também é comum e normalmente envolve um modelo de alto desempenho cujas previsões são pioradas por um ou mais outros modelos de baixo desempenho e o conjunto não é capaz de aproveitar suas contribuições individuais de forma eficaz. Sendo assim, o modelo de melhor desempenho deve ser usado no lugar do *ensemble* criado. Finalizada a análise do desempenho dos modelos, vejamos as previsões. Como o modelo com melhor resultado foi o LIGHTGBM (*Light Gradient Boosting Machine*) tunado, faremos as previsões com ele:

```
predictions_lightgbm_imp = predict_model(modelo_lightgbm_imp, data = test_df_imp)
predictions_lightgbm_imp
```

	CO_NCM	CO_PAIS	CO_VIA	KG_LIQUIDO	VL_FOB	Label	Score
875740	84821010	161	4	6	137	1	0.5485
122066	84133010	249	4	6	428	4	0.7089
1552522	73181600	23	1	2	85	4	0.6676
364523	87083019	791	1	330	4442	1	0.7730
680939	85411099	791	4	20	5819	4	0.7018

Onde CO_VIA é a classe real e $Label$ é a classe prevista. Observe que dessas 5 previsões o resultado obtido foi igual ao esperado em 3 (três) e diferente do esperado em 2 (dois). Nesse caso, essas duas operações podem ser objeto de análise fiscal para verificar se há algum indício de infração.

7. Links

Link para o vídeo: <https://youtu.be/v3fgQOjPCEo>

Link para o repositório:

<https://drive.google.com/drive/folders/1ZO3VWbZqBJ3oITGRLf0bwqoacyxdWfm?usp=sharing>

REFERÊNCIAS

MCKINNEY, William Wesley. Python para análise de Dados. São Paulo: Novatec Editora , 2018.

<http://editor.economia.gov.br/Economia/noticias/2019/06/secretaria-de-comercio-exterior-atualiza-estatisticas-de-2018>

<http://www.camex.gov.br/tarifa-externa-comum-tec/tec-listas-em-vigor>

<https://dclogisticsbrasil.com/6-aspectos-para-avaliar-na-escolha-do-seu-modal-de-transporte/>

<https://didatica.tech/como-funciona-o-algoritmo-arvore-de-decisao/>

<https://didatica.tech/o-que-e-e-como-funciona-o-algoritmo-randomforest/>

<https://gauchazh.clicrbs.com.br/coronavirus-servico/noticia/2020/03/falta-da-mascara-n95-gera-protestos-em-porto-alegre-e-divide-opinioes-de-profissionais-da-saude-ck87pcc5o01lr01rz9y6s7kwf.html>

<https://gauchazh.clicrbs.com.br/coronavirus-servico/noticia/2020/03/falta-de-equipamentos-de-protectao-preocupa-profissionais-da-saude-e-governo-acelera-compras-ck87viii507yu01pq7la80ktd.html>

<https://matplotlib.org/>

<https://machinelearningmastery.com/why-use-ensemble-learning/>

<https://pandas.pydata.org/>

<https://pt.stackoverflow.com/>

https://pt.wikipedia.org/wiki/Com%C3%A9rcio_internacional

<https://pycaret.org/>

<https://receita.economia.gov.br/sobre/institucional/cadeia-de-valor-1/controle-aduaneiro#:~:text=Consiste%20em%20realizar%20o%20controle,de%20medidas%20de%20defesa%20comercial.>

<https://repositorio.enap.gov.br/bitstream/1/3896/1/Rafael%20Cunha%20Alves%20Moreira%20-%20vCorrigidoFinal.pdf>

<https://saude.abril.com.br/medicina/oms-decreta-pandemia-do-novo-coronavirus-saiba-o-que-isso-significa/#:~:text=A%20Organiza%C3%A7%C3%A3o%20Mundial%20da%20Sa%C3%A3o,Ade,quantidade%20de%20pa%C3%ADses%20afetados%20triplicou.>

<https://scikit-learn.org/stable/>

<https://sigmoidal.ai/como-criar-uma-wordcloud-em-python/>

<https://towardsdatascience.com/a-data-science-workflow-canvas-to-kickstart-your-projects-db62556be4d0>

<https://www.analyticsvidhya.com/blog/2020/10/7-feature-engineering-techniques-machine-learning/>

[https://www.comexdobrasil.com/alvo-dos-ataques-de-bolsonaro-a-china-nao-tem-competidor-como-maior-parceiro-comercial-do-brasil/#:~:text=Da%20Reda%C3%A7%C3%A3o%20\(*\)%20Bras%C3%ADa%20%E2%80%93%20A,vendido%20pelo%20pa%C3%ADs%20no%20exterior.](https://www.comexdobrasil.com/alvo-dos-ataques-de-bolsonaro-a-china-nao-tem-competidor-como-maior-parceiro-comercial-do-brasil/#:~:text=Da%20Reda%C3%A7%C3%A3o%20(*)%20Bras%C3%ADa%20%E2%80%93%20A,vendido%20pelo%20pa%C3%ADs%20no%20exterior.)

<https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

<https://www.datageeks.com.br/naive-bayes/>

<https://www.dezyre.com/recipes/drop-out-highly-correlated-features-in-python>

<https://www.domaniconsultoria.com/post/com%C3%A9rcio-exterior-e-o-brasil-mundo#:~:text=A%20import%C3%A2ncia%20do%20com%C3%A9rcio%20exterior,e,m%20que%20alcan%C3%A7am%20maior%20competitividade.>

<https://www.gov.br/produtividade-e-comercio-exterior/pt-br/assuntos/comercio-exterior/estatisticas/base-de-dados-bruta>

<https://www.gov.br/pt-br/noticias/financas-impostos-e-gestao-publica/2021/01/balanca-comercial-fecha-2020-com-superavit-de-us-50-9-bilhoes>

[https://www.gov.br/pt-br/noticias/financas-impostos-e-gestao-publica/2021/01/balanca-comercial-fecha-2020-com-superavit-de-us-50-9-bilhoes\).](https://www.gov.br/pt-br/noticias/financas-impostos-e-gestao-publica/2021/01/balanca-comercial-fecha-2020-com-superavit-de-us-50-9-bilhoes).)

<https://www.legisweb.com.br/noticia/?id=25074>

<https://www.paho.org/pt/covid19>

<https://www.prestex.com.br/blog/modais-de-transporte-de-carga-no-brasil-conheca-os-5-principais/>

<https://www.suinoculturainustrial.com.br/imprensa/importacoes-de-soja-brasileira-pela-china-disparam-51-em-setembro/20201027-085855-y822>

APÊNDICE

Slides do PowerPoint e
Jupyter notebook.