PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

NÚCLEO DE EDUCAÇÃO A DISTÂNCIA

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

Osmar Batista de Carvalho Junior

CLASSIFICAÇÃO PARAMETRIZADA DO TRÂNSITO DE CAMINHÕES *EN LASTRE* COM BASE EM REGISTROS DE PASSAGENS

Belo Horizonte

2022

Osmar Batista de Carvalho Junior

CLASSIFICAÇÃO PARAMETRIZADA DO TRÂNSITO DE CAMINHÕES *EN LASTRE* COM BASE EM REGISTROS DE PASSAGENS

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

Belo Horizonte

2022

SUMÁRIO

1. Introdução

1.1. Contextualização

O trabalho é realizado na esfera de atuação da Receita Federal do Brasil – Órgão contratante deste curso de Especialização - que tem, dentre suas competências, a de execução dos serviços de controle aduaneiro e atividades de repressão ao contrabando e descaminho, inclusive contrafação, pirataria, entorpecentes e drogas afins, armas de fogo, lavagem e ocultação de bens, direitos e valores, observada a competência específica de outros órgãos.

A fiscalização do tráfego de mercadorias no contexto de Comércio Exterior, principalmente nas fronteiras, é uma das medidas adotadas no processo de controle aduaneiro, incluindo-se os meios empregados no transporte dessas mercadorias.

O fluxo comercial no modal terrestre é grande – são vários caminhões ingressando no país diariamente; e como não há efetivo de pessoal suficiente, torna-se inviável a fiscalização integral dos caminhões, seja na Zona Primária seja na Secundária[1], e termina-se por haver priorização da fiscalização daqueles que se encontram carregados, sendo que aqueles que ingressam *en lastre* (vazios) nem sempre são submetidos à fiscalização por scanner, o que potencializa o risco de "ingressos irregulares" no país.

Ademais, há de se considerar o equilíbrio que se deve ter entre o controle (fiscalização) e a facilitação do comércio internacional – preocupação constante da Administração Aduaneira. Pois controle rigoroso implicaria alto custo – tempo, longas filas de caminhões, aumento do custo Brasil etc; por outro lado, controle menos rigoroso poderia resultar em alto risco de ingressos irregulares de mercadorias, inclusive armas, drogas etc

[1] A zona primária é constituída pelos portos, aeroportos e pontos de fronteira alfandegados. A zona secundária é o restante do território nacional.

Este trabalho, nesse contexto, foca no controle de entrada de caminhões *en lastre* num determinado ponto de fronteira do território aduaneiro[2], podendo ser aplicado da mesma forma em outros pontos de fronteira terrestre.

Registre-se que não se pretende com este trabalho indicar ou identificar, por qualquer meio, pessoa – seja física ou jurídica, sendo os parâmetros de classificação empregados de forma exclusivamente objetiva com base em histórico de movimentação dos caminhões, sendo expressão desta. Logo, qualquer ocorrência neste sentido, será mera coincidência. Inclusive, devido a isso, nem todos os dados empregados são disponibilizados, ou o são de forma anonimizada.

1.2. O problema proposto

Dado o exposto, fica evidente que o problema reside na impossibilidade de se realizar integralmente a fiscalização de caminhões *en lastre* devido à quantidade de veículos e à falta de recursos humanos, que são necessários para efetivação dessa atividade, vulnerabilizando os controles internos do país.

Com isso, o cerne do que se pretende é viabilizar forma mais eficiente de fiscalização, provendo a essa atividade meios para que se possa realizá-la de forma seletiva, mitigando aquele risco, considerando, num determinado período, os registros de passagens durante o trânsito de caminhões *en lastre* em pontos (rodovias) relativamente próximos do ponto de entrada no país, observando-se principalmente aqueles que saem do país na mesma condição que entraram – *en lastre*.

Para tal, são analisados registros de passagens extraídos de sistemas que usam câmeras LPR[3], juntamente com dados de controle documental (no ingresso e saída), sendo estes obtidos dentro do órgão e de agente concessionário.

Objetiva-se, a partir desses dados, definir *labels* (rótulos) a partir de parâmetros de passagens e de tempo, para que se possa treinar modelo supervisionado capaz de realizar a classificação de cada evento de entrada/trânsito

[2] Território Aduaneiro compreende todo o território nacional, inclusive o mar territorial, as águas territoriais e o espaço aéreo correspondente.
[3] License Plate Recognition - câmeras que podem extrair números e letras das placas de veículos.

de determinado caminhão, considerando a sua placa e aspectos temporais pré-definidos.

Os dados coletados, relativos a um trimestre - Jul à Set/2022 -, referem-se à unidade específica de fronteira, contudo, o objetivo é de aplicação em qualquer unidade cujo modal de transporte seja executado por caminhões. No escopo deste, são tratados apenas dados relativos ao trânsito de caminhões *en lastre* entre os países Brasil e Paraguai.

Nesse processo foi necessário contato com funcionários de alguns setores para entender melhor a sistemática envolvida no controle do fluxo de caminhões que ingressam no país, principalmente os que são objetos deste trabalho – os *en lastre*.

Cabe o registro de que o objeto deste trabalho tem o intuito de, adicionalmente, conformar-se à diretriz do Órgão contratante do curso de que tenha aplicação na atividade laboral. Contudo, há dados que, embora não tenham natureza econômico-fiscal, por prudência são removidos ou anonimizados, mitigando-se qualquer ilação sobre os mesmos, dando, minimamente, natureza reservada a seu conteúdo.

2. Coleta de Dados

Os dados (5 *datasets*), devido ao volume, foram coletados por etapas (várias extrações/arquivos) em bases de quatro sistemas distintos (aqui designados por siglas no escopo deste trabalho, que também identificam os respectivos *datasets*), sendo dois de sistemas de câmeras LPR (PIA/PIABR e SIV) e dois relativos a registros de entrada/saída (com interação humana) a partir de apresentação documental (MIC e SAR).

Deve-se destacar que nos sistemas de LPR há dias sem registros, pois pode ter havido indisponibilidade do sistema; e, além disso, houve coletas adicionais: do último dia do mês de Jun devido à inconsistência na digitação de dados do MIC, sendo verificada também em outros dias dentro do período de análise; e do mês de Out, apenas para verificação de saída dos caminhões, nos casos em que houve entrada nos últimos dias de Set.

2.1 *Dataset* MIC

A partir deste *dataset* é que são identificados os caminhões *en lastre*. O sistema é alimentado a partir de documento onde constam dados de entrada/saída do caminhão. Os dados apresentam inconsistências, pois o sistema não oferece muitas restrições/críticas quanto à entrada de dados. Gerou-se com isso trabalho extra, inclusive em ponto central que é o de definir a data/hora de passagem do caminhão ao ingressar ou sair do país. Essas inconsistências foram identificadas no processamento inicial deste com o *dataset* PIA, sendo necessário visita ao setor e breve entrevista com pelo menos dois digitadores. Dessa forma, constatou-se a problemática existente neste *dataset*.

MIC		
COLUNA	**DESCRIÇÃO**	**TIPO**
DATA_CADASTRO	Data/hora de cadastro no sistema	Data/Hora
DATA_EMISSAO	Data de emissão do documento	Data
DATA_PASSAGEM	Data de passagem na fronteira	Data
DESTINO	Local de destino	Texto
EM_LASTRE	Indicação dessa condição (sem dados)	Texto
EQUIPE	Equipe de trabalho	Texto
MOTORISTA	Nome do motorista	Texto
NUMERO	Número documento identificação	Número
ORIGEM	Local de origem	Texto
PLACA_CARRETA	Placa da carreta	Texto
PLACA_CAVALO	Placa do cavalo	Texto
RESPONSAVEL	Responsável pelo transporte	Texto
TIPO	Entrada ou Saída do país	Texto
TRANSPORTADORA	Nome da transportadora	Texto

2.2 *Dataset* PIA

Em conjunto com o MIC, este *dataset* é que permite identificar a data/hora de passagem na fronteira (entrada ou saída). O sistema é alimentado de forma

automática pelas câmeras LPR, conforme ocorrem as passagens dos caminhões. Os dados foram extraídos em 11 arquivos (5 de entradas e 6 de saídas) devido à limitações do próprio sistema, sendo consolidados em dois *subdatasets* (arquivos), um referente à entrada e outro referente à saída.

PIA		
COLUNA	**DESCRIÇÃO**	**TIPO**
CAMERA	Permite identificar o local da câmera, se entrada/saída e a posição de leitura (indica o tipo de placa)	Texto
DATA_HORA	Data/hora de passagem na fronteira	Data/Hora
ENCAMINHADA	Flag de status do registro	Texto
EQUIPAMENTO	Permite identificar se entrada/saída	Texto
ID	Índice do *dataset*	Texto
PLACA	Placa do caminhão (cavalo/carreta)	Texto (formato)

2.3 *Dataset* PIABR

Dataset que permite complementarmente confirmar os registros de saída que são verificados em PIA, em conjunto com MIC. Só foram coletados dados relativos a Set (e Out para verificação de saída) devido à indisponibilidade nos outros meses. O sistema também é de câmeras LPR, registrando as passagens antes da fronteira. Os dados foram extraídos em sete arquivos devido à limitações do sistema, sendo consolidados em único *dataset*.

PIABR		
COLUNA	**DESCRIÇÃO**	**TIPO**
CAMERA	Permite identificar a faixa de leitura na via e a posição de leitura (indica o tipo de placa)	Texto
DATA_HORA	Data/hora de passagem	Data/Hora
ENCAMINHADA	Flag de status do registro	Texto
EQUIPAMENTO	Permite identificar o local	Texto
ID	Índice do *dataset*	Número
PLACA	Placa do veículo	Texto (formato)

2.4 *Dataset* SAR

Este *dataset* é empregado principalmente para verificar se houve ingresso de caminhão em recinto de despacho aduaneiro para carregamento, possibilitando

inferir que não houve, para um determinado registro de entrada, o evento de saída *en lastre*. O sistema é alimentado com dados digitados (também baseados em documentação) por empresa concessionária prestadora de serviços ao órgão fiscalizador, fazendo, dentre outros, a movimentação e armazenagem de mercadorias. Os dados foram extraídos em dois *datasets*, sendo consolidados em um.

SAR		
COLUNA	**DESCRIÇÃO**	**TIPO**
CNPJ_TRANSPORTADORA	CNPJ transportadora	Texto
CPF	CPF motorist	Texto
DATA_CADASTRO	Data/hora de cadastro no sistema	Data/Hora
DATA_ENTRADA	Data/hora de entrada no recinto	Data/Hora
DATA_PASSAGEM_FRONTEIRA	Data/hora de passagem na fronteira (saída)	Texto
DATA_SAIDA	Data/hora de saída do recinto	Data/Hora
DESTINO	Local de destino	Texto
MIC	Número MIC	Número
MOTORISTA	Nome motorista	Texto
ORIGEM	Local de origem	Texto
PLACA_CARRETA	Placa da carreta	Texto (formato)
PLACA_CAVALO	Placa do cavalo	Texto (formato)
RG	Identificação do motorista	Texto
TEMPO_CAD_ENT	Tempo entre cadastro e entrada recinto	Número
TEMPO_SAIDA_FRONTEIRA	Tempo entre a saída recinto e fronteira	Número
TRANSPORTADORA	Nome da transportadora	Texto

2.5 *Dataset* SIV

Usado para verificação dos registros de passagens em 9 pontos de interesse. O sistema é também alimentado de forma automática pelas câmeras LPR, conforme ocorrem as passagens dos caminhões. Os dados foram extraídos em 220 arquivos devido à limitações do sistema, sendo consolidados inicialmente em nove *subdatasets*. Estes poderiam ter sido consolidados em único arquivo por compartilharem de mesmas colunas, contudo optou-se, devido ao elevado número de registros e certas peculiaridades, por mantê-los separados até que houvesse tratamento inicial dos dados. Cada *subdataset* recebe as seguintes designações

“lógicas e físicas” (sufixos em arquivo *.csv): siv_ecaz, siv_eccv, siv_esmi, siv_mfoz, siv_prot, siv_rcaz, siv_rgua, siv_rsti e siv_vcor.

SIV		
COLUNA	**DESCRIÇÃO**	**TIPO**
DATA_HORA	Data/hora de passagem	Data/Hora
INFORMACOES	Informações diversas	Texto
PLACA	Placa do veículo	Texto (formato)
PONTO	Ponto de monitoramento	Texto
SENTIDO	Sentido do trajeto de acordo com o ponto	Texto

3. Processamento/Tratamento de Dados

São processados, após as devidas consolidações, os cinco *datasets*, apresentando os seguintes números concernentes à verificação de duplicidades (os percentuais são relacionados ao total de cada *dataset*):

DATASET	REGISTROS	DUPLICADOS	%
MIC	28845	12	0,04%
PIA	117022	562	0,48%
PIABR	1735249	138158	7,96%
SAR	70227	0	0,00%
SIV	8460982	24692	0,29%
TOTAL	**10412325**	**163424**	**1,57%**

Os *subdatasets* que compõem o *dataset* a ser consolidado – SIV - apresentam individualmente:

SUBDATASET	REGISTROS	DUPLICADOS	%
siv_ecaz	798398	0	0,00%
siv_eccv	602644	0	0,00%
siv_esmi	1001484	24687	2,47%
siv_mfoz	139551	0	0,00%

siv_prot	112943	0	0,00%
siv_rcaz	1357543	1	0,00%
siv_rgua	1538752	0	0,00%
siv_rsti	2232071	4	0,00%
siv_vcor	677596	0	0,00%
TOTAL	**8460982**	**24692**	**0,29%**

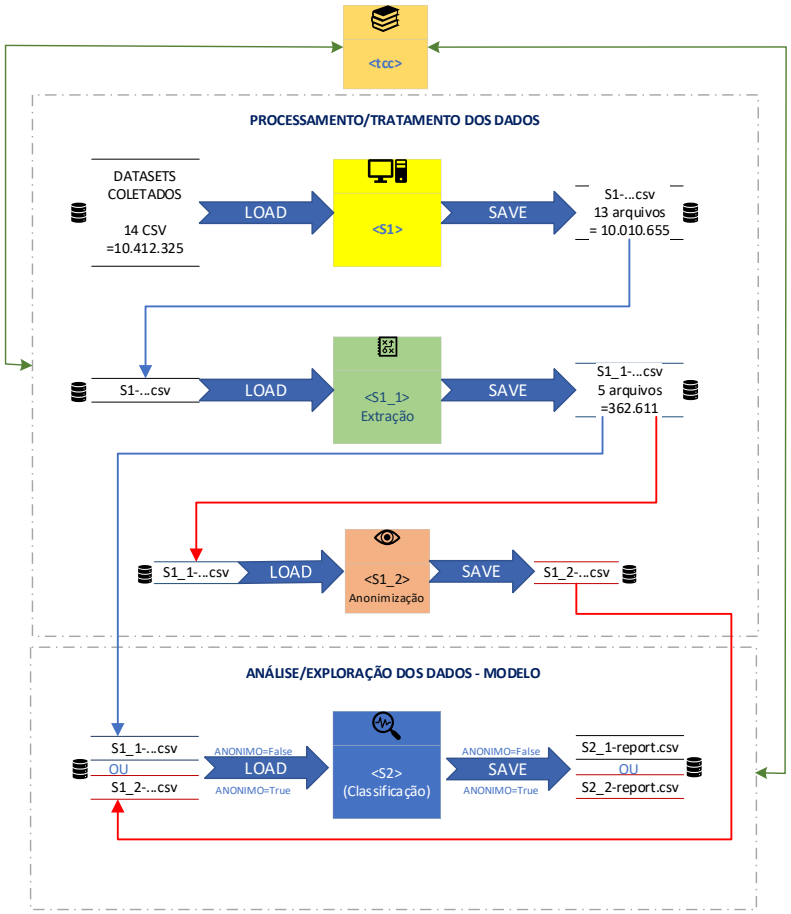
Todos esses registros (em duplicidade e também com placas inválidas) são removidos sem comprometer o objeto do trabalho. Apenas como observação, nota-se que há percentual elevado de registros duplicados e de placas inválidas (leituras erradas do sistema LPR) em PIABR, o que é explicável devido ao fato de que o ponto de leitura é de fato problemático, apresentando instabilidade com certa frequência. Com relação às duplicidades em siv_esmi, houve realmente extração de registros em duplicidade.

Considerando o volume de dados e as finalidades de cada *dataset* no processo, optou-se por tratá-los individualmente, principalmente para reduzir a quantidade de registros, focando naqueles de interesse. E ao final de cada etapa gerou-se um arquivo CSV que seria entrada da etapa seguinte. Com isso maximizou-se as performances em tempo de processamento, facilitando inclusive o desenvolvimento das etapas seguintes.

3.1 *Workflow*

Com intuito de dar uma visão geral, a figura abaixo, mostra a sequência executada em cada etapa do trabalho, iniciando-se com os 14 arquivos coletados – relativos aos *datasets*. As designações com prefixos <S1...S2> referem-se aos arquivos (*IPython Notebook = *.ipynb*) de implementação, sendo o de prefixo <S1> da fase de Processamento/Tratamento dos dados; e o de <S2> da fase de Análise/Exploração dos dados e de definição do Modelo, conforme linhas de fronteira (tracejadas). As linhas azuis e vermelhas indicam o fluxo (entradas e saídas de arquivos CSV) entre os *notebooks*; e a verde a interação dos *notebooks* com um

módulo de funções do TCC. São usados no desenvolvimento o ambiente *Jupyter Notebook* (no início do trabalho) e *JupyterLab.*



Inicialmente são processados os 14 *datasets* (arquivos CSV) que contêm os dados coletados, totalizando 10.412.325 registros. Cada *dataset* é tratado (remoção de registros em duplicidade ou inválidos, correção de registros, etc) individualmente em <S1> (5 *notebooks* – S1-ProcTrataDados-<*dataset*>.ipynb), lembrando que SIV é composto por 9 *subdatasets* e PIA 2, resultando em 13 arquivos CSV (com 10.010.655 registros) com prefixo "S1-"...csv que são usados (carregados) como entrada da "Extração" <S1_1>. Neste (S1_1-ProcTrataDados-Extract.ipynb) a principal finalidade é a de reduzir a quantidade de registros e fazer a concatenação dos *subdatasets* SIV. Para tal, são selecionados nos *datasets*, a partir do *dataset* MIC (que é o *dataset* que informa quais são os caminhões *en lastre*), os registros de interesse cujas placas constam em MIC e vice-versa, fazendo-se a exclusão dos registros em que não houve match, resultando em 5 arquivos CSV (um de cada

dataset - totalizando 362.611 registros) com prefixo “S1_1-“...csv que são usados carregados em <S2> para processamento.

Em <S2> (S2-AnaliseExploraDados-Modelo.ipynb) são verificadas as correspondências entre os registros de MIC x PIA e devidos processamentos subsequentes, lembrando que MIC não tem a hora de passagem, mas apenas a data (que apresenta inconsistência), daí a necessidade de se verificar o exato momento de passagem por meio do *dataset* PIA (dados provenientes de câmera LPR). Na sequência, já com a data/hora de entrada de cada registro, são feitas as verificações de passagens nos demais *datasets*. Ao final, definem-se rótulos para cada registro selecionado, cria-se o modelo, faz-se a validação/teste e geram-se relatórios para subsidiar a seleção para fiscalização – “S2_1-“...csv. No início de <S2> é definido parâmetro (flag) ANONIMO que identifica a execução, conforme a entrada, para <S1_1> (normal) ou <S1_2> (anonimizada). Esse flag tem ainda a finalidade de identificar distintamente o nome do arquivo com o prefixo correspondente (na entrada S1_2* e na saída S2_2*), assim é possível saber, pelo nome, quais são os arquivos anonimizados.

Com relação ao fluxo em vermelho <S1_2> (Anonimização – S1_2-Anonimizacao.ipynb), este tem apenas a função de anonimizar algumas *features* (colunas), contudo permitindo o processamento normal em <S2> tal qual feito para <S1_1>, sendo que neste caso, resultando em 5 arquivos CSV com prefixo/início “S1_2-“...csv. Estes *datasets* anonimizados é que são disponibilizados no escopo deste TCC, podendo ser carregados em <S2> com a execução de *script*. Os demais *scripts* são disponibilizados apenas com o resultado da execução (em formato HTML) – vide apêndice.

É definido também um módulo <tcc> (arquivo tcc_cdbd.py – vide apêndice), importado nos *notebooks* como tcc, onde são implementadas algumas funções utilizadas nas diversas etapas do desenvolvimento. Informações sobre as funções em *docstrings.*

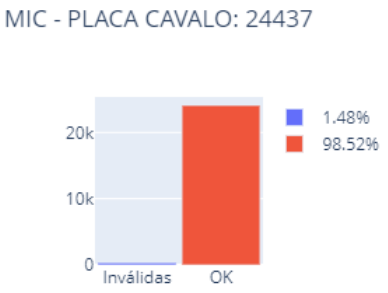
Dada essa contextualização do processo, seguem tratamentos dados especificamente a cada *dataset* conforme etapa e descrição retro.

3.2 Processamento em <S1>

- **MIC:**
  - arquivo de entrada: mic.csv;
  - remoção das colunas DATA_EMISSAO, EM_LASTRE, RESPONSAVEL, TRANSPORTADORA, EQUIPE, MOTORISTA, NUMERO - as três primeiras por não conterem dados e as demais por identificar pessoas, seja física ou jurídica, não sendo usadas no contexto deste TCC;
  - Informações de entrada (após remoção dessas colunas):

	DATA_PASSAGEM	DATA_CADASTRO	ORIGEM	DESTINO	PLACA_CAVALO	PLACA_CARRETA	TIPO
column type	object	object	object	object	object	object	object
count values	28845	28845	28845	28845	28845	28625	28845
unique values	110	19158	1708	2171	7973	7293	2
null values	0	0	0	0	0	220	0
null values (%)	0.0	0.0	0.0	0.0	0.0	0.762697	0.0

  - remoção dos registros em duplicidade;
  - remoção de caracteres especiais das colunas referentes à placas e ORIGEM e DESTINO;
  - remoção dos registros com placas inválidas (apenas do cavalo):



  - remoção dos registros de entrada de Out, pois apenas os de saída é que são usados para verificação deste evento;
  - mudança para tipo *datetime* das colunas referentes à data;
  - atribuição do valor “NO_DATA” para campos nulos em PLACA_CARRETA;

- identificação das placas de carreta inválidas (sem remoção), atribuindo o valor “ERROR”;
- criação das colunas ORIGEM_PAIS e DESTINO_PAIS, cujos dados foram definidos a partir das colunas ORIGEM e DESTINO;
- remoção dos registros em que não foi possível identificar o país de origem/destino ou com dados inconsistentes;
- remoção de registros onde ORIGEM_PAIS e DESTINO_PAIS igual a ‘AR’ ou ‘UR’, ficando apenas ‘BR’ e ‘PY’, que são objeto deste TCC.
- remoção das colunas DATA_CADASTRO, ORIGEM e DESTINO por não serem mais necessárias;
- Principal problema do *dataset*, além de outras inconsistências, é o da data de passagem devido a sua importância no processo de identificação da data de entrada no *dataset* PIA;
- arquivo de saída: S1-ProcTrataDados-mic.csv
- Informações de saída:

	DATA_PASSAGEM	PLACA_CAVALO	PLACA_CARRETA	TIPO	ORIGEM_PAIS	DESTINO_PAIS
column type	datetime64[ns]	object	object	object	object	object
count values	23510	23510	23510	23510	23510	23510
unique values	103	6331	5895	2	2	2
null values	0	0	0	0	0	0
null values (%)	0.0	0.0	0.0	0.0	0.0	0.0

- **PIA:**

  - arquivos de entrada: pia_in.csv, pia_out.csv;
  - remoção das colunas ID e ENCAMINHADA por não serem necessárias;
  - Informações de entrada – 2 *subdatasets* (após remoção dessas colunas):

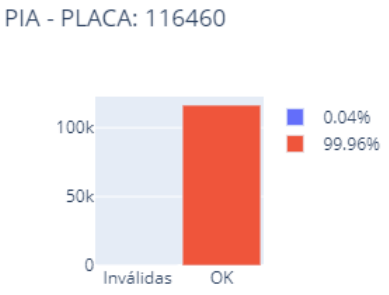
'PIA_IN'

	DATA_HORA	EQUIPAMENTO	CAMERA	PLACA
column type	object	object	object	object
count values	48021	48021	48021	48021
unique values	45085	1	2	22572
null values	0	0	0	0
null values (%)	0.0	0.0	0.0	0.0

'PIA_OUT'

	DATA_HORA	EQUIPAMENTO	CAMERA	PLACA
column type	object	object	object	object
count values	69001	69001	69001	69001
unique values	63389	1	2	27566
null values	0	0	0	0
null values (%)	0.0	0.0	0.0	0.0

- mudança para tipo *datetime* da coluna referente à data;
- criação da coluna TIPO, sendo atribuído o conteúdo ENTRADA ou SAÍDA, conforme o *subdataset* (pia_in ou pia_out);
- concatenação dos *subdatasets* em único *dataset* PIA;
- remoção dos registros em duplicidade;
- remoção de caracteres especiais na coluna PLACA;
- remoção de registros com placas inválidas:



- renomeação das colunas EQUIPAMENTO e CAMERA para PONTO e CAM_POSICAO, respectivamente. As mudanças foram feitas para

conformação com o *dataset* SIV (PONTO) e para possível identificação se a leitura da câmera ocorreu na parte frontal ou traseira do caminhão.

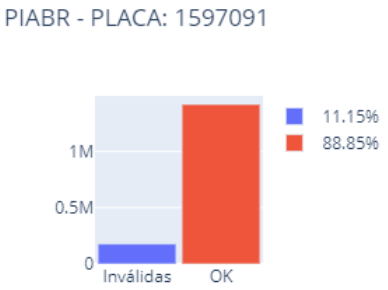
- arquivo de saída: S1-ProcTrataDados-pia.csv
- Informações de saída:

	DATA_HORA	PLACA	TIPO	PONTO	CAM_POSICAO
column type	datetime64[ns]	object	object	object	object
count values	116419	116419	116419	116419	116419
unique values	108305	41664	2	2	2
null values	0	0	0	0	0
null values (%)	0.0	0.0	0.0	0.0	0.0

- **PIABR:**
  - arquivo de entrada: piabr.csv;
  - remoção das colunas ID e ENCAMINHADA por não serem necessárias;
  - Informações de entrada (após remoção dessas colunas):

	DATA_HORA	EQUIPAMENTO	CAMERA	PLACA
column type	object	object	object	object
count values	1735249	1735249	1735249	1735249
unique values	72750	2	4	339169
null values	0	0	0	0
null values (%)	0.0	0.0	0.0	0.0

  - mudança para tipo *datetime* da coluna referente à data;
  - correção de conteúdo da coluna EQUIPAMENTO;
  - criação da coluna TIPO, sendo atribuído o conteúdo SAÍDA;
  - remoção dos registros em duplicidade;
  - remoção de caracteres especiais na coluna PLACA;
  - remoção de registros com placas inválidas;



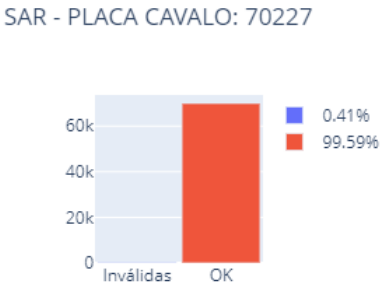
- renomeação das colunas EQUIPAMENTO e CAMERA para PONTO e CAM_POSICAO, respectivamente. As mudanças são feitas para conformação com o *dataset* SIV (PONTO) e para possível identificação se a leitura da câmera ocorreu na parte frontal ou traseira do caminhão.
- arquivo de saída: S1-ProcTrataDados-piabr.csv
- Informações de saída:

	DATA_HORA	PLACA	TIPO	PONTO	CAM_POSICAO
column type	datetime64[ns]	object	object	object	object
count values	1419070	1419070	1419070	1419070	1419070
unique values	72547	279375	1	1	2
null values	0	0	0	0	0
null values (%)	0.0	0.0	0.0	0.0	0.0

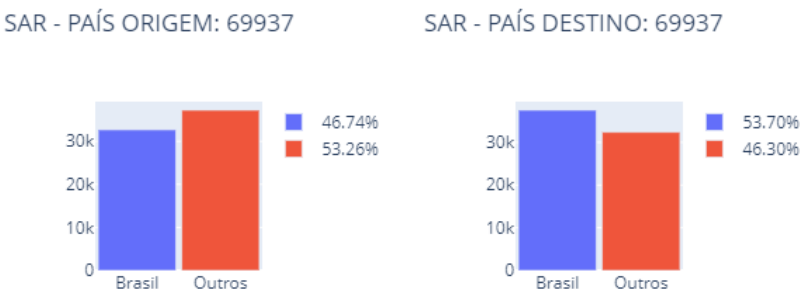
- **SAR:**
  - arquivo de entrada: sar.csv;
  - remoção das colunas TEMPO_CAD_ENT, TEMPO_SAIDA_FRONTEIRA, MIC, MOTORISTA, RG, CPF, TRANSPORTADORA, CNPJ_TRANSPORTADORA – registros que identificam pessoas, além de não serem necessárias no escopo deste TCC;
  - Informações de entrada (após remoção dessas colunas):

	DATA_CADASTRO	DATA_ENTRADA	PLACA_CAVALO	PLACA_CARRETA	ORIGEM	DESTINO	DATA_SAIDA	DATA_PASSAGEM_FRONTEIRA
column type	object	object	object	object	object	object	object	object
count values	70227	70227	70227	66809	70227	70227	70221	32558
unique values	50582	55656	11050	10617	4	3	51280	25240
null values	0	0	0	3418	0	0	6	37669
null values (%)	0.0	0.0	0.0	4.867074	0.0	0.0	0.008544	53.638914

- mudança para tipo *datetime* das colunas referentes à data;
- remoção dos registros em duplicidade;
- remoção de caracteres especiais das colunas referentes à placas;
- remoção dos registros com placas inválidas (apenas do cavalo):



- atribuição do valor “NO_DATA” para campos nulos em PLACA_CARRETA;
- identificação das placas de carreta inválidas (sem remoção), atribuindo o valor ‘ERROR’;
- criação das colunas ORIGEM_PAIS e DESTINO_PAIS, cujos dados são definidos a partir das colunas ORIGEM e DESTINO:



- criação da coluna TIPO cujo dado é definido a partir de ORIGEM_PAIS e DESTINO_PAIS – valores: ‘INTERNO’, ‘SAIDA’, ‘ENTRADA’.

- remoção das colunas ORIGEM, DESTINO e DATA_PASSAGEM_FRONTEIRA por não serem mais necessárias;
- arquivo de saída: S1-ProcTrataDados-sar.csv

- Informações de saída:

	DATA_CADASTRO	DATA_ENTRADA	PLACA_CAVALO	PLACA_CARRETA	TIPO	ORIGEM_PAIS	DATA_SAIDA	DESTINO_PAIS
column type	datetime64[ns]	datetime64[ns]	object	object	object	object	datetime64[ns]	object
count values	69937	69937	69937	69937	69937	69937	69931	69937
unique values	50422	55482	10864	10546	3	3	51103	3
null values	0	0	0	0	0	0	6	0
null values (%)	0.0	0.0	0.0	0.0	0.0	0.0	0.008579	0.0

- **SIV:**

- arquivos de entrada: <*subdataset*>.csv;
- Remoção da coluna INFORMAÇÕES por não ser necessária;
- Informações de entrada – 9 *subdatasets* (após remoção dessa coluna), totalizando 8.460.982 registros:

'SUBDATASET: siv_mfoz'

	PLACA	DATA_HORA	PONTO	SENTIDO
column type	object	datetime64[ns]	object	object
count values	139551	139551	139551	139551
unique values	11155	138112	1	2
null values	0	0	0	0
null values (%)	0.0	0.0	0.0	0.0

'SUBDATASET: siv_rsti'

	PLACA	DATA_HORA	PONTO	SENTIDO
column type	object	datetime64[ns]	object	object
count values	2232071	2232071	2232071	2232071
unique values	735528	1649850	1	2
null values	0	0	0	0
null values (%)	0.0	0.0	0.0	0.0

'SUBDATASET: siv_esmi'

	PLACA	DATA_HORA	PONTO	SENTIDO
column type	object	datetime64[ns]	object	object
count values	1001484	1001484	1001484	1001484
unique values	248853	925296	1	2
null values	0	0	0	0
null values (%)	0.0	0.0	0.0	0.0

'SUBDATASET: siv_rcaz'

	PLACA	DATA_HORA	PONTO	SENTIDO
column type	object	datetime64[ns]	object	object
count values	1357543	1357543	1357543	1357543
unique values	516141	1065282	1	2
null values	0	0	0	0
null values (%)	0.0	0.0	0.0	0.0

'SUBDATASET: siv_ecaz'

	PLACA	DATA_HORA	PONTO	SENTIDO
column type	object	datetime64[ns]	object	object
count values	798398	798398	798398	798398
unique values	233033	762717	1	2
null values	0	0	0	0
null values (%)	0.0	0.0	0.0	0.0

'SUBDATASET: siv_eccv'

	PLACA	DATA_HORA	PONTO	SENTIDO
column type	object	datetime64[ns]	object	object
count values	602644	602644	602644	602644
unique values	174015	583873	1	2
null values	0	0	0	0
null values (%)	0.0	0.0	0.0	0.0

'SUBDATASET: siv_vcor'

	PLACA	DATA_HORA	PONTO	SENTIDO
column type	object	datetime64[ns]	object	object
count values	677596	677596	677596	677596
unique values	173549	652029	1	2
null values	0	0	0	0
null values (%)	0.0	0.0	0.0	0.0

'SUBDATASET: siv_rgua'

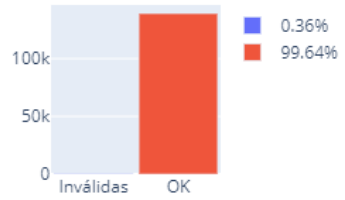
	PLACA	DATA_HORA	PONTO	SENTIDO
column type	object	datetime64[ns]	object	object
count values	1538752	1538752	1538752	1538752
unique values	690204	991610	1	2
null values	0	0	0	0
null values (%)	0.0	0.0	0.0	0.0

'SUBDATASET: siv_prot'

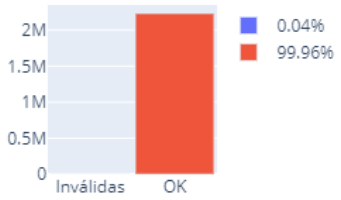
	PLACA	DATA_HORA	PONTO	SENTIDO
column type	object	datetime64[ns]	object	object
count values	112943	112943	112943	112943
unique values	35075	108708	1	1
null values	0	0	0	0
null values (%)	0.0	0.0	0.0	0.0

- remoção dos registros em duplicidade;
- remoção de caracteres especiais na coluna PLACA;
- remoção de registros com placas inválidas:

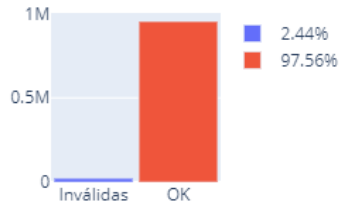
SIV_MFOZ - PLACA: 139551



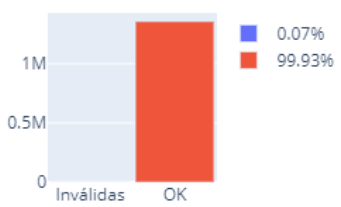
SIV_RSTI - PLACA: 2232067



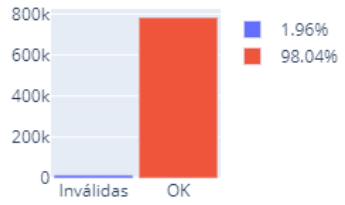
SIV_ESMI - PLACA: 976797



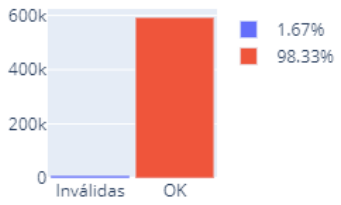
SIV_RCAZ - PLACA: 1357542



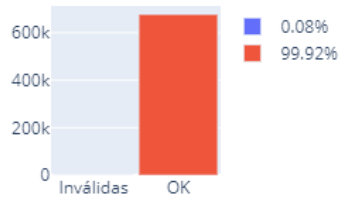
SIV_ECAZ - PLACA: 798398



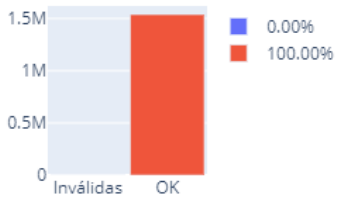
SIV_ECCV - PLACA: 602644

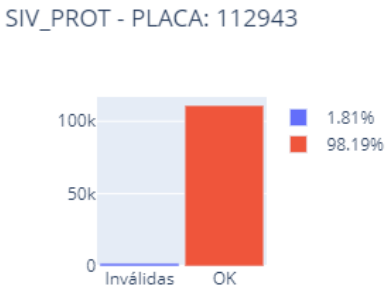


SIV_VCOR - PLACA: 677596



SIV_RGUA - PLACA: 1538752





- arquivos de saída: S1-ProcTrataDados-<*subdataset*>.csv
- Informações de saída (consolidado SIV: 8.381.719 registros):

'SUBDATASET: siv_mfoz'

	DATA_HORA	PLACA	PONTO	SENTIDO
column type	datetime64[ns]	object	object	object
count values	139055	139055	139055	139055
unique values	137626	10967	1	2
null values	0	0	0	0
null values (%)	0.0	0.0	0.0	0.0

'SUBDATASET: siv_rsti'

	DATA_HORA	PLACA	PONTO	SENTIDO
column type	datetime64[ns]	object	object	object
count values	2231156	2231156	2231156	2231156
unique values	1649295	734863	1	2
null values	0	0	0	0
null values (%)	0.0	0.0	0.0	0.0

'SUBDATASET: siv_esmi'

	DATA_HORA	PLACA	PONTO	SENTIDO
column type	datetime64[ns]	object	object	object
count values	952920	952920	952920	952920
unique values	904076	239381	1	2
null values	0	0	0	0
null values (%)	0.0	0.0	0.0	0.0

'SUBDATASET: siv_rcaz'

	DATA_HORA	PLACA	PONTO	SENTIDO
column type	datetime64[ns]	object	object	object
count values	1356551	1356551	1356551	1356551
unique values	1064740	515460	1	2
null values	0	0	0	0
null values (%)	0.0	0.0	0.0	0.0

'SUBDATASET: siv_ecaz'

	DATA_HORA	PLACA	PONTO	SENTIDO
column type	datetime64[ns]	object	object	object
count values	782754	782754	782754	782754
unique values	748676	225231	1	2
null values	0	0	0	0
null values (%)	0.0	0.0	0.0	0.0

'SUBDATASET: siv_eccv'

	DATA_HORA	PLACA	PONTO	SENTIDO
column type	datetime64[ns]	object	object	object
count values	592584	592584	592584	592584
unique values	574520	169082	1	2
null values	0	0	0	0
null values (%)	0.0	0.0	0.0	0.0

'SUBDATASET: siv_vcor'

	DATA_HORA	PLACA	PONTO	SENTIDO
column type	datetime64[ns]	object	object	object
count values	677060	677060	677060	677060
unique values	651545	173434	1	2
null values	0	0	0	0
null values (%)	0.0	0.0	0.0	0.0

'SUBDATASET: siv_rgua'

	DATA_HORA	PLACA	PONTO	SENTIDO
column type	datetime64[ns]	object	object	object
count values	1538735	1538735	1538735	1538735
unique values	991600	690187	1	2
null values	0	0	0	0
null values (%)	0.0	0.0	0.0	0.0

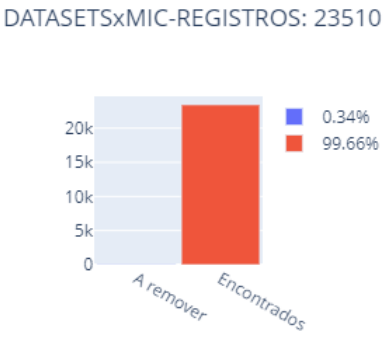
'SUBDATASET: siv_prot'

	DATA_HORA	PLACA	PONTO	SENTIDO
column type	datetime64[ns]	object	object	object
count values	110904	110904	110904	110904
unique values	107419	33692	1	1
null values	0	0	0	0
null values (%)	0.0	0.0	0.0	0.0

3.3 Processamento em <S1_1> - Extração

No tratamento realizado nos dados em <S1_1>, todos os arquivos e informações de entrada originam-se de <S1>. Como resultado, totalizou-se em 362.611 registros remanescentes (decréscimo superior a 96% em relação à <S1>):

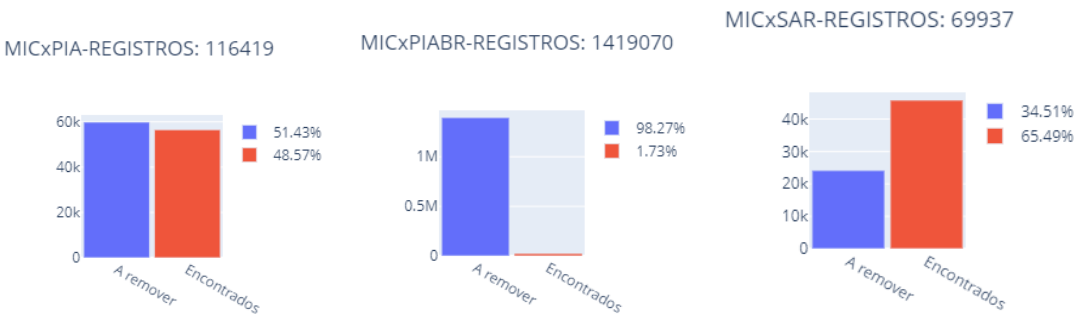
- **MIC:**
  - inicialmente é feita a seleção de todas as placas, sem repetição, constantes em MIC, armazenando-as numa lista usada para verificar a ocorrência dessas placas nos outros *datasets*. As que não são encontradas têm os respectivos registros excluídos (a remoção é feita no segundo *dataset* do par exibido nas figuras apresentadas nos tópicos de cada *dataset* a seguir);
  - ao final, é feita a marcação das placas encontradas e remoção dos registros cujas placas não foram encontradas em MIC com base em seleção idêntica feita nos outros *datasets* (em conjunto):



- arquivos de saída: S1_1-ProcTrataDados-mic.csv
- Informações de saída:

	DATA_PASSAGEM	PLACA_CAVALO	PLACA_CARRETA	TIPO	ORIGEM_PAIS	DESTINO_PAIS
column type	datetime64[ns]	object	object	object	object	object
count values	23430	23430	23430	23430	23430	23430
unique values	103	6281	5843	2	2	2
null values	0	0	0	0	0	0
null values (%)	0.0	0.0	0.0	0.0	0.0	0.0

- **PIA, PIABR e SAR:**
  - marcação das placas encontradas e remoção dos registros cujas placas não são encontradas (no segundo *dataset* do par):



  - arquivos de saída:
    - **PIA**: S1_1-ProcTrataDados-**pia**.csv
    - **PIABR**: S1_1-ProcTrataDados-**piabr**.csv

- **SAR**: S1_1-ProcTrataDados-**sar**.csv

- Informações de saída:

'DATASET: PIA'

	DATA_HORA	PLACA	TIPO	PONTO	CAM_POSICAO
column type	object	object	object	object	object
count values	56542	56542	56542	56542	56542
unique values	55957	9286	2	2	2
null values	0	0	0	0	0
null values (%)	0.0	0.0	0.0	0.0	0.0

'DATASET: PIABR'

	DATA_HORA	PLACA	TIPO	PONTO	CAM_POSICAO
column type	object	object	object	object	object
count values	24500	24500	24500	24500	24500
unique values	14111	7776	1	1	2
null values	0	0	0	0	0
null values (%)	0.0	0.0	0.0	0.0	0.0

'DATASET: SAR'

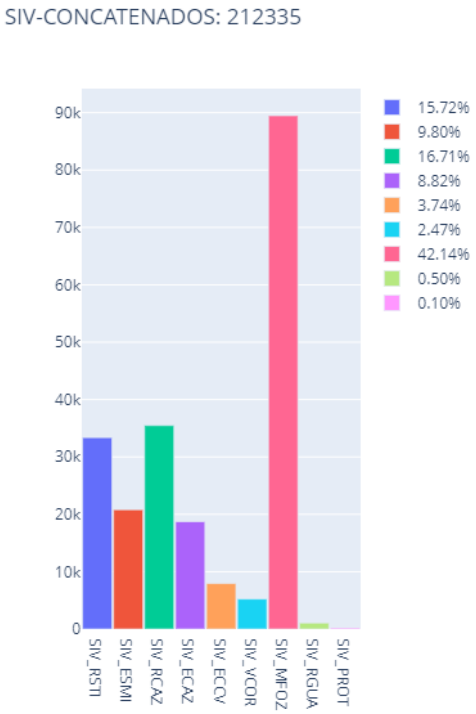
	DATA_ENTRADA	DATA_SAIDA	PLACA_CAVALO	PLACA_CARRETA	TIPO	ORIGEM_PAIS	DESTINO_PAIS
column type	object	object	object	object	object	object	object
count values	45804	45801	45804	45804	45804	45804	45804
unique values	39566	36475	5583	5350	3	3	3
null values	0	3	0	0	0	0	0
null values (%)	0.0	0.00655	0.0	0.0	0.0	0.0	0.0

- **SIV:**

  - marcação das placas encontradas e remoção dos registros cujas placas não são encontradas:

- consolidação dos 9 *subdatasets*, concatenando-os em único *dataset* SIV, sendo o identificador de cada *subdataset* usado como chave (*key)* na concatenação:

```
#concatena os dataframes em <siv>
siv = pd.concat([siv_rsti,siv_esmi,siv_rcaz,siv_ecaz,siv_eccv,
                 siv_vcor,siv_mfoz,siv_rgua,siv_prot],
      keys=['SIV_RSTI','SIV_ESMI','SIV_RCAZ','SIV_ECAZ','SIV_ECCV',
            'SIV_VCOR','SIV_MFOZ','SIV_RGUA','SIV_PROT'])
```



- arquivos de saída: S1_1-ProcTrataDados-siv.csv
- Informações de saída:

	DATA_HORA	PLACA	PONTO	SENTIDO
column type	object	object	object	object
count values	212335	212335	212335	212335
unique values	209281	9878	9	9
null values	0	0	0	0
null values (%)	0.0	0.0	0.0	0.0

3.4 Processamento em <S1_2> - Anonimização

No contexto deste trabalho, embora o seu resultado seja objetivo com a rotulagem feita pelo modelo treinado, tratando de aspectos concernentes ao trânsito de caminhões *en lastre* e abstraindo pessoas envolvidas (vários dados/colunas foram excluídos), por prudência optou-se por anonimizar alguns dados, principalmente as placas. São mantidas as data/hora por serem necessárias em <S2> e também por ser inviável determinar especificamente a qual caminhão refere-

se determinado registro. Contudo, essa anonimização não afeta a predição feita pelo modelo, uma vez que as *features* aplicadas são todas do tipo bool.

Assim, descreve-se o processo de anonimização em <S1_2>, sendo os arquivos e informações de entrada oriundos de <S1_1> (5 *datasets*):

- criação, no *dataset* PIA, da coluna PLACA_BR que especifica se a placa é brasileira ou não. Esta é a única coluna em que é criada em <S2> e que foi preciso criá-la em <S1_2> justamente devido à anonimização da placa. Em <S2>, o parâmetro ANONIMO, se tiver valor *True,* não permite a recriação da referida coluna (PLACA_BR);
- é definida uma função hash_column() onde são codificadas as colunas, conforme o *dataset*, usando-se funções do módulo hashlib:
  - MIC e SAR: PLACA_CAVALO e PLACA_CARRETA; e
  - PIA/PIABR/SIV: PLACA e PONTO;

```
def hash_column(df, column):
    """
    df      - dataset sob o qual haverá aplicação da função hash
    column  - coluna (feature) do dataset <df> a ser anonimizada
    """
    df[column] = df[column].apply(lambda x:hashlib.sha256(x.encode()).hexdigest())
```

- arquivos de saída (5 arquivos com colunas anonimizadas): S1_2-ProcTrataDados-<*dataset*>.csv.

4. Análise e Exploração dos Dados

O processo relativo a esta etapa e a seguinte, desenvolveu-se em <S2> tendo como dados de entrada os arquivos de <S1_1> ou <S1_2>, conforme seja processamento “normal” (para desenvolvimento/TCC e trabalhos futuros) ou anonimizado (para apresentação do TCC), respectivamente.

Os principais aspectos para consecução do objetivo são as identificações de passagens em vários pontos, sendo essencial o de passagem em PIA, pois é este *dataset* que possibilita saber a real data/hora de entrada de determinado caminhão,

considerando principalmente as inconsistências já expressas com relação ao MIC. A partir dessa identificação de registro (match) de MIC em PIA, cálculos de tempo e verificação de saída na mesma condição de entrada do caminhão (*en lastre*) são exequíveis.

4.1 Configuração inicial <S2>

Para consecução do TCC, verificou-se a necessidade de se anonimizar alguns dados em <S1_2>. Diante disso, definiu-se o parâmetro ANONIMO na configuração inicial do *notebook* em <S2>, que tem a finalidade de indicar o arquivo a ser carregado/salvo como entrada/saída de <S2>, quando anonimizado, usando-se os prefixos S1_2 ou S2_2, respectivamente.

Outro ponto verificado na análise foi, dado o contexto, a necessidade de se definir parâmetros de ajuste com relação a alguns tempos envolvidos que subsidiam a criação de *features* relacionadas (com designação parametrizada – que têm como sufixo o tempo definido no parâmetro), permitindo flexibilidade e conformação ao contexto temporal e geográfico. Assim são criados parâmetros (constantes) iniciais de configuração em <S2> com tempos que permitem a definição de valores distintos conforme são modificados, inclusive alterando-se o processo de rotulagem do *dataset*, que é usado para treinar o modelo:

- T_RSTI - tempo (min) de passagem no ponto a partir da entrada país;
- T_ESMI - tempo (min) de passagem no ponto a partir da entrada país;
- T_SAR - tempo (min) de ingresso ponto a partir da entrada país;
- T_BR - tempo (min) de permanência no país desde a entrada.

A título de exemplo, para: T_RSTI=35, T_ESMI=50, T_SAR=240 e T_BR=720 - são criadas as *features*: RSTI35, ESMI50, SAR240 e BR720, respectivamente. Sendo que para cada *feature* parametrizada é definida uma variável correspondente que contém o nome daquela *feature* a ser usada na implementação: f_rsti (=RSTI35), f_esmi (=ESMI50), f_sar (=SAR240) e f_br (=BR720). Cada uma dessas colunas tendo valor *True,* significa que houve registro de passagem no referido ponto em tempo (minutos) menor ou igual ao sufixo numérico.

Outro parâmetro inicial é DIAS, que é empregado para limitar/estabelecer número de dias para encontrar a data seguinte <prox_data>, dessa forma estabelecendo um período <data_hora>; ou para limitar número de dias quando o intervalo entre duas datas for superior a este tempo.

Os valores de parâmetros acima expressos são os usados no escopo deste trabalho, sendo DIAS = 5.

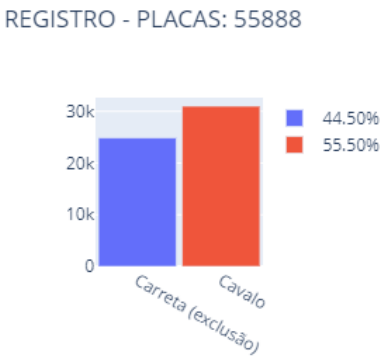
A partir de cópia do *dataset* PIA definiu-se o dataframe (*dataset* lógico) mic_pia (MIC_PIA), aproveitando-o para estruturar o *dataset* a ser processado no modelo. Em sequência, após análise, são sinteticamente elencados os itens implementados, conforme a interação dos *datasets* (e, didaticamente, seguindo estrutura do *notebook* <S2>), sendo todas as alterações feitas em mic_pia. Como regra, há modificação de tipo das *features* relativas à datas para datetime.

4.2 Interações dos *datasets*

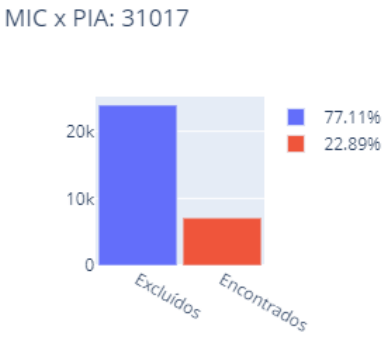
- **PIA x MIC:**
  - objetivo principal: identificar os registros correspondentes nos dois *datasets*, sendo ponto central a captura da data/hora de entrada;
  - criação das *features*:
    - DATA a partir de DATA_HORA;
    - TIPO_PLACA a partir da *feature* CAM_POSICAO – identifica se placa cavalo ou carreta;
    - PLACA_BR a partir do formato da placa (função *mark_plate_origem()*), se o parâmetro ANONIMO for *False*. Se for *True*, já terá sido criada em <S1_2> - identifica se a placa do caminhão é nacional (*True*) ou estrangeira (*False*);
    - MIC_IDX – identifica a correspondência dos registros entre MIC e PIA;

- **MIC_PIA x MIC_PIA:**
  - objetivo principal: verificação da data/hora de saída, atribuindo-a à *feature* DATA_SAIDA;

- criação da *feature* DATA_SAIDA – para registro da data provável de saída, se houver;
- definição das funções:
  - placas_passagens_dh() - usada para limitar as buscas por passagens nos pontos em cada *dataset*, mantendo a correspondência para cada registro de Entrada;
  - get_proxima_data() - usada para estabelecer um período de verificação de passagens conforme ENTRADAS/SAÍDAS no/do Brasil. Retorna a próxima data de passagem conforme a placa e data/hora informada.
- remoção dos registros com conteúdo igual a 'CARRETA' na *feature* TIPO_PLACA. Após análise, verificou-se que este dado, além das várias inconsistências, agregaria pouco neste trabalho, bastando então apenas ter os dados do cavalo:



- remoção dos registros sem dados em MIC_IDX – sem correspondência entre MIC x PIA:



- **PIABR x MIC_PIA:**
  - objetivo principal: verificação da data/hora de passagem em PIABR, atribuindo-a à *feature* de mesma designação em MIC_PIA. Esta *feature* é usada, complementarmente, para verificar se houve a saída do caminhão, considerando determinado período desde a sua entrada;
  - criação da *feature* PIABR para registro da data/hora de passagem.
  - definição da função match_micpiabr_siv() – que atribui à *feature* PIABR as datas/horas de passagem;
  - remoção dos registros da *feature* TIPO cujos valores sejam 'SAIDA', pois já utilizados para atribuição da *feature* DATA_SAIDA;

- **SIV x MIC_PIA:**
  - objetivo principal: verificação da data/hora de passagem em diversos pontos relativos aos *subdatasets* SIV, atribuindo-os às *features* de mesma designação em MIC_PIA;
  - criação das *features* com registro da data/hora de passagem: SIV_MFOZ, SIV_RSTI, SIV_ESMI, SIV_RCAZ, SIV_ECAZ, SIV_ECCV, SIV_VCOR, SIV_PROT, SIV_RGUA;
  - em SIV, remoção das *features* PONTO e de índice (sem nome); e renomeação de coluna sem nome (onde constam valores do parâmetro *key* na concatenação de SIV em <S1_1>) como *DATASET*, mantendo-se com isso referência ao *subdataset* original;
  - atribuição das data/hora de passagem às *features* acima (com designação dos *subdatasets*) com chamada da função match_micpiabr_siv();

- **SAR x MIC_PIA:**
  - objetivo principal: verificação da data/hora de entrada ou saída em SAR, que são usadas para definir se houve carregamento do caminhão;
  - criação das *features* para registro da data/hora de entrada ou saída: SAR_ENTRADA e SAR_SAIDA, que têm atribuídas as data/hora de entrada e saída localizadas em SAR;
  - criação da *feature* CARGA, que é usada para identificar se houve carregamento do caminhão;

4.3 Outras definições

- criação da *feature* SAIDA_VAZIO, que é usada para identificar se o caminhão saiu vazio;
- definição da função flag_tempo_passagem(), que define quatro *features* (colunas) dinamicamente com os prefixos RSTI, ESMI, SAR e BR, sendo estas concatenadas com o tempo em minutos dos parâmetros iniciais T_RSTI, T_ESMI, T_SAR e T_BR, respectivamente. Recebem valor *True* quando o intervalo entre a data/hora de passagem e a data/hora de entrada no país for menor ou igual ao valor (tempo) do respectivo parâmetro T_*;
- criação da *feature* SCANNER_ON, que indica se na entrada do caminhão o equipamento estava em funcionamento, conforme dia da semana: 4ª e 6ª das 18:00 às 05:00 (do dia seguinte); 3ª, 5ª e sábado das 19:20 às 04:20 (do dia seguinte); e domingo das 00:00 às 04:20. Atribuição de valores pela função set_scanner_on():

```
#atribuição Scanner ON nos registros selecionados, conforme períodos
def set_scanner_on(idx):
    for i in idx:
        mic_pia['SCANNER_ON'][mic_pia.index == i] = True
```

```
#4ª 6ª:18:00  às 05:00
idx = mic_pia[(mic_pia.TIPO == 'ENTRADA') &
    (
        (mic_pia.DATA.dt.day_of_week == 2) | #Quarta
        (mic_pia.DATA.dt.day_of_week == 4)   #Sexta
    )
].between_time('18:00','23:59').index
set_scanner_on(idx)

idx = mic_pia[(mic_pia.TIPO == 'ENTRADA') &
    (
        (mic_pia.DATA.dt.day_of_week == 2) |
        (mic_pia.DATA.dt.day_of_week == 4) |
        (mic_pia.DATA.dt.day_of_week == 6)   #Domingo

    )
].between_time('00:00','04:20').index
set_scanner_on(idx)
```

```
#3ª 5ª sáb: 19:20 as 04:20 SCANNER_ON
idx = mic_pia[(mic_pia.TIPO == 'ENTRADA') &
    (
        (mic_pia.DATA.dt.day_of_week == 1) | #Terça
        (mic_pia.DATA.dt.day_of_week == 3) | #Quinta
        (mic_pia.DATA.dt.day_of_week == 5)   #Sáb
    )
].between_time('19:20','23:59').index
set_scanner_on(idx)

idx = mic_pia[(mic_pia.TIPO == 'ENTRADA') &
    (
        (mic_pia.DATA.dt.day_of_week == 3) |
        (mic_pia.DATA.dt.day_of_week == 5)
    )
].between_time('00:00','05:00').index
set_scanner_on(idx)
```

- definiu-se por atribuir *flags* (*True*) às *features* SIV_* onde houvesse registro de passagem, com exceção de SIV_MFOZ que é usada para atribuição da *feature* CARGA.

4.4 Rotulagem – *Target Label*: MOTIVO

Neste tópico é feita a rotulagem dos registros conforme parâmetros de interesse no escopo desse trabalho. São usados rótulos de 0 a 21, sendo que o rótulo 0 significa que não houve MOTIVO (“Não constatado”) para rotular, portanto 22 rótulos. Nesta etapa é definido o *dataset* lógico mic_pia_label a partir de mic_pia (esta definição é mera técnica de desenvolvimento).

- criação da constante ***TARGET*** que se refere ao nome da *feature target*, no caso ‘MOTIVO’;
- criação da *feature* MOTIVO que contém a rotulagem;

- a rotulagem é feita com base nas colunas: 'SCANNER_ON', 'CARGA', 'SAIDA_VAZIO', 'PLACA_BR', **'RSTI35'**, **'ESMI50'**, **'SAR240'**, **'BR720'**, 'SIV_RSTI', 'SIV_ESMI', 'SIV_RCAZ', 'SIV_ECAZ', 'SIV_ECCV', 'SIV_VCOR', 'SIV_PROT', 'SIV_RGUA'. Observe-se que as *features* criadas com uso dos parâmetros T_* (em destaque) encontram-se nesta lista com os respectivos sufixos de tempo (min) definidos anteriormente.
- a descrição dos parâmetros (motivo da rotulagem – relacionado ao caminhão e seu trânsito) é armazenada na variável **parâmetro** (tipo dict), que tem como chave (*key*) o respectivo rótulo. O processo de rotulagem está expresso em seção de mesma designação desta em <S2> (vide apêndice). Como exemplo, definição do rótulo 17 com sua respectiva descrição:

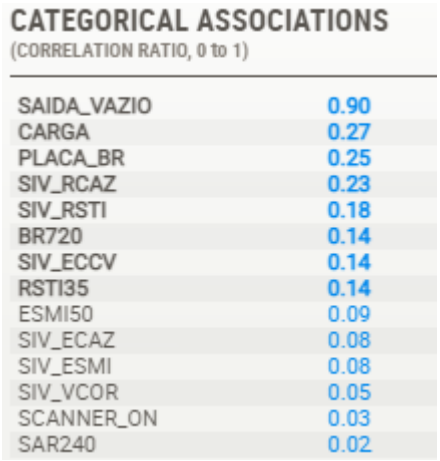
```
#17
parametro[17]='E/S vazio; scanner ON; placa BR; sem passagem pontos BR; mais de '+str(T_BR)+'min no Brasil'

mic_pia_label[TARGET][(mic_pia_label.SCANNER_ON) & (mic_pia_label.SAIDA_VAZIO) & (mic_pia_label.PLACA_BR) &
    (~mic_pia_label[f_br]) & (~mic_pia_label[f_rsti]) & (~mic_pia_label[f_esmi]) &
    ~( (mic_pia_label.SIV_RCAZ) | (mic_pia_label.SIV_ECAZ) | (mic_pia_label.SIV_ECCV) |
     (mic_pia_label.SIV_VCOR) | (mic_pia_label.SIV_ESMI) | (mic_pia_label.SIV_RSTI)) &
   (~mic_pia_label.SIV_PROT) & (~mic_pia_label.SIV_RGUA)
   ] = 17
```

MOTIVO	Parametros
0	Não constatado
1	Passagem Rota Lago/Guaíra: ou saindo vazio ou com placa estrangeira ou sem carga.
2	Entrou vazio; não carregou; placa estrangeira; passagem pontos BR; sem registro saída vazio
3	Entrou vazio; não carregou; placa estrangeira; direto STI/SMI; passagem outros pontos; sem registro de saída vazio
4	Entrou vazio; scanner OFF; placa estrangeira; direto STI/SMI; passagem outros pontos; entrou p/carga após 240min
5	E/S vazio; scanner ON; placa estrangeira; sem passagem pontos BR; mais de 720min no Brasil
6	E/S vazio; scanner ON; placa BR; direto STI/SMI; passagem outros pontos; mais de 720min no Brasil
7	E/S vazio; scanner ON; placa estrangeira; direto STI/SMI; passagem outros pontos; mais de 720min no Brasil
8	E/S vazio; scanner ON; placa BR; passagem pontos BR; mais de 720min no Brasil
9	E/S vazio; scanner OFF; placa estrangeira; passagem pontos BR; mais de 720min no Brasil
10	E/S vazio; scanner ON; placa estrangeira; sem passagem pontos BR; até 720min no Brasil
11	E/S vazio; scanner OFF; placa BR; direto STI/SMI; passagem outros pontos; mais de 720min no Brasil
12	E/S vazio; scanner OFF; placa estrangeira; sem passagem pontos BR; mais de 720min no Brasil
13	E/S vazio; scanner OFF; placa estrangeira; direto STI/SMI; passagem outros pontos; mais de 720min no Brasil
14	E/S vazio; scanner OFF; placa estrangeira; sem passagem pontos BR; até 720min no Brasil
15	E/S vazio; scanner OFF; placa BR; passagem pontos BR; mais de 720min no Brasil
16	E/S vazio; scanner ON; placa estrangeira; direto STI/SMI; sem passagem outros pontos; mais de 720min no Brasil
17	E/S vazio; scanner ON; placa BR; sem passagem pontos BR; mais de 720min no Brasil
18	E/S vazio; scanner ON; placa estrangeira; passagem pontos BR; mais de 720min no Brasil
19	E/S vazio; scanner OFF; placa estrangeira; direto STI/SMI; sem passagem outros pontos; mais de 720min no Brasil
20	E/S vazio; scanner OFF; placa BR; sem passagem pontos BR; mais de 720min no Brasil
21	E/S vazio; scanner OFF; placa BR; sem passagem pontos BR; até 720min no Brasil

(*) E/S – entrada e saída

- uso da biblioteca ***sweetviz*** para verificação das associações entre as *features*, inclusive com a *feature TARGET* - MOTIVO. Esta biblioteca é muito interessante em processo EDA (*Exploratory Data Analysis*), pois possibilita a análise por meio de algumas características resumidas do *dataset*, inclusive de forma visual. Neste caso em particular, só foi usada após a rotulagem, uma vez que devido à natureza dos dados, julga-se que não haveria muito a acrescentar nas etapas iniciais. Os relatórios/gráficos (arquivos HTML) encontram-se listados no apêndice:



CATEGORICAL ASSOCIATIONS
(CORRELATION RATIO, 0 to 1)

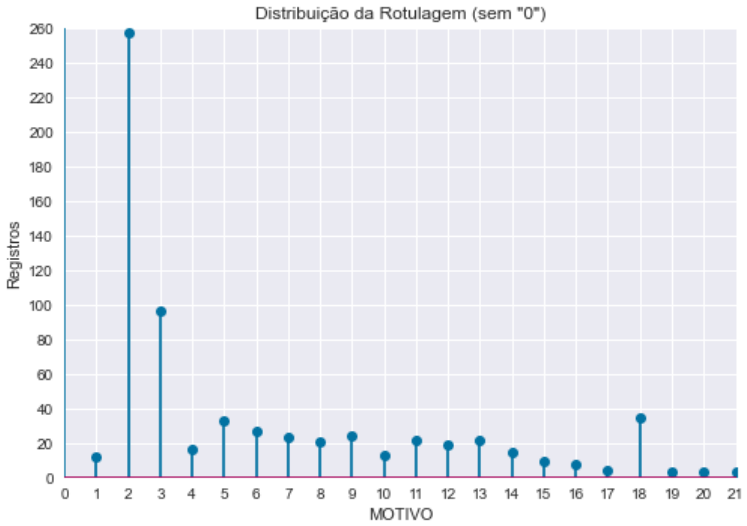
SAIDA_VAZIO	0.90
CARGA	0.27
PLACA_BR	0.25
SIV_RCAZ	0.23
SIV_RSTI	0.18
BR720	0.14
SIV_ECCV	0.14
RSTI35	0.14
ESMI50	0.09
SIV_ECAZ	0.08
SIV_ESMI	0.08
SIV_VCOR	0.05
SCANNER_ON	0.03
SAR240	0.02

4.5 Gráficos sobre as *features*

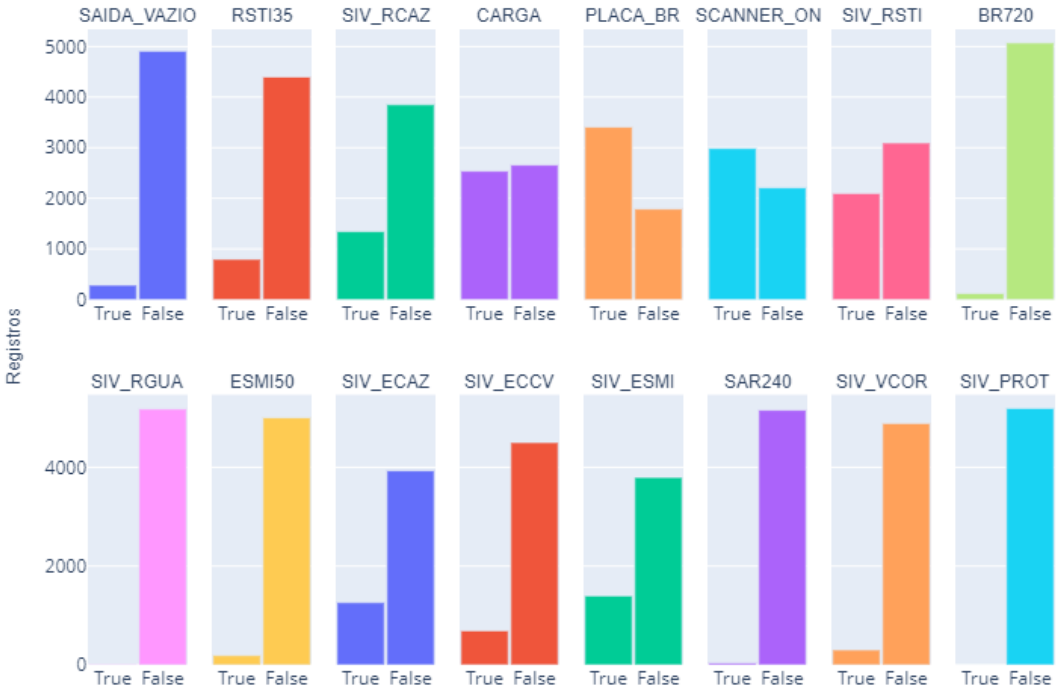
A seguir são apresentados alguns gráficos sobre os dados usados no modelo, considerando as distribuições dos rótulos e/ou o quantitativo de valores em cada *feature*:

- distribuição da rotulagem da base usada para definição do modelo (sem o rótulo 0 que ficou com 4529 registros - 87%; os demais com 665 registros). Deve-se observar o destaque dos rótulos 2 e 3, que têm parâmetros bem parecidos, contudo, com a diferença de que o caminhão estrangeiro passou por determinado ponto imediatamente após ingressar no país. Esses motivos (rótulos), no mínimo, sinalizam para que haja verificação da situação,

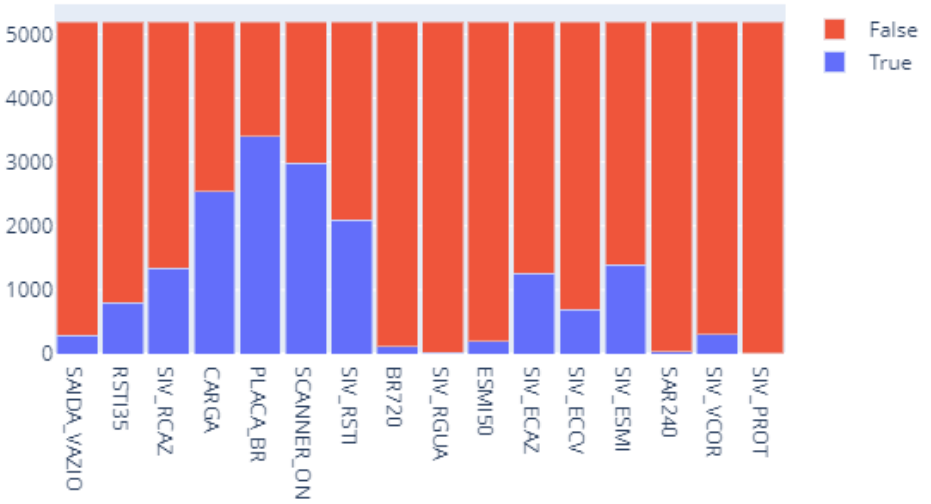
principalmente se houver frequência, podendo ser verificados nos relatórios gerados:



- quantificação das *features* usadas na definição dos rótulos:

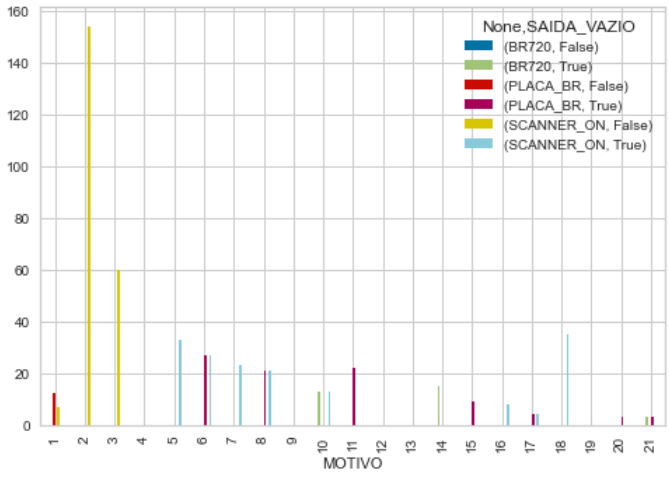


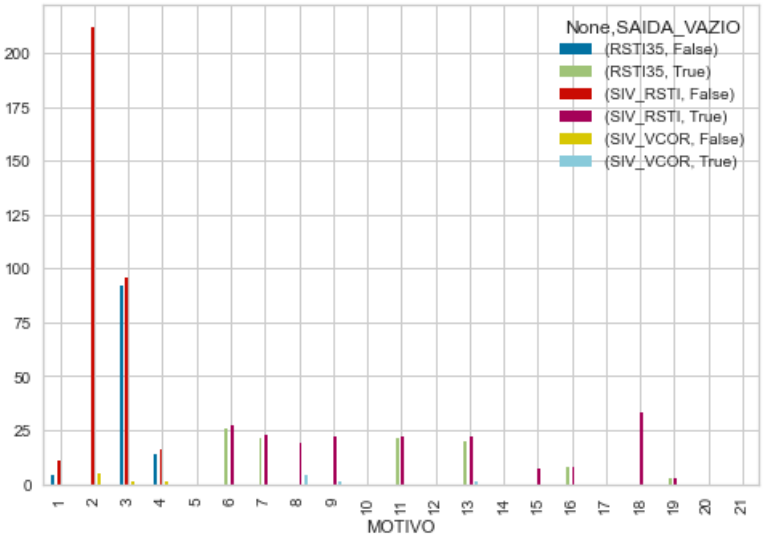
- quantificação das *features* sob outra perspectiva:

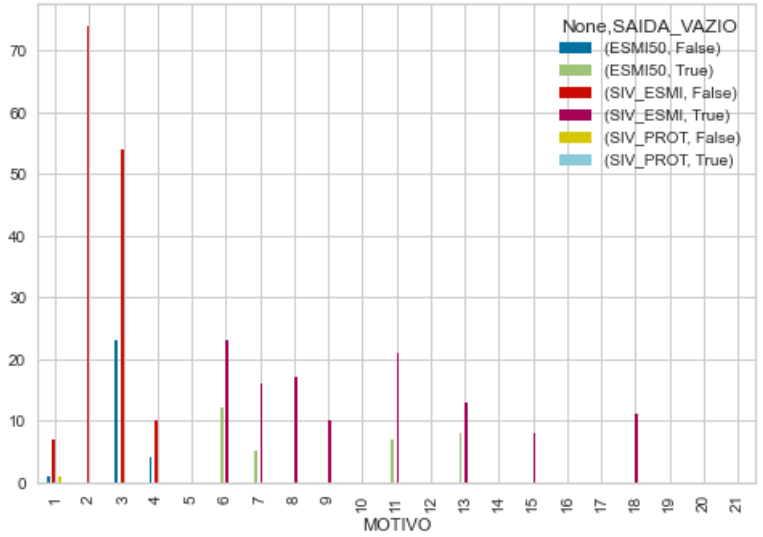


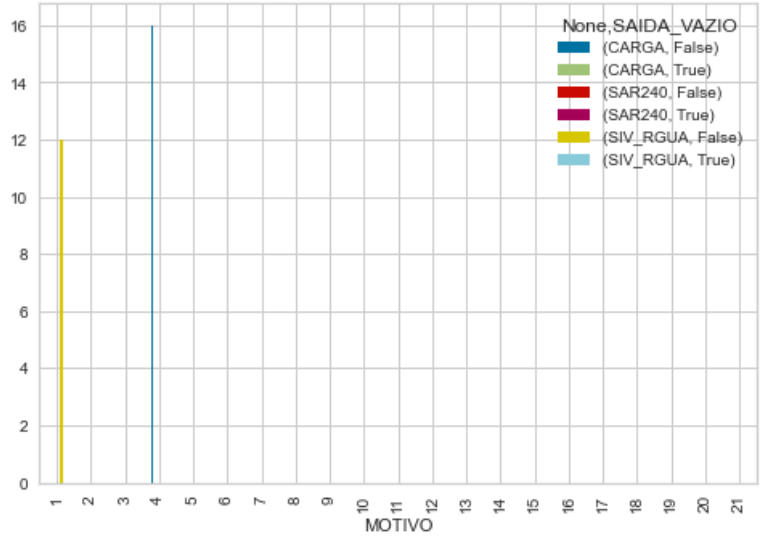
- interação das *features* com SAIDA_VAZIO (**MOTIVO > 0**) – considerando que esta *feature* é a principal para constatação de que houve entrada e saída (E/S) vazio por determinado caminhão, é interessante observar a distribuição das interações na *feature* MOTIVO, que inclusive pode permitir avaliação futura quanto aos parâmetros de tempo (T_*).

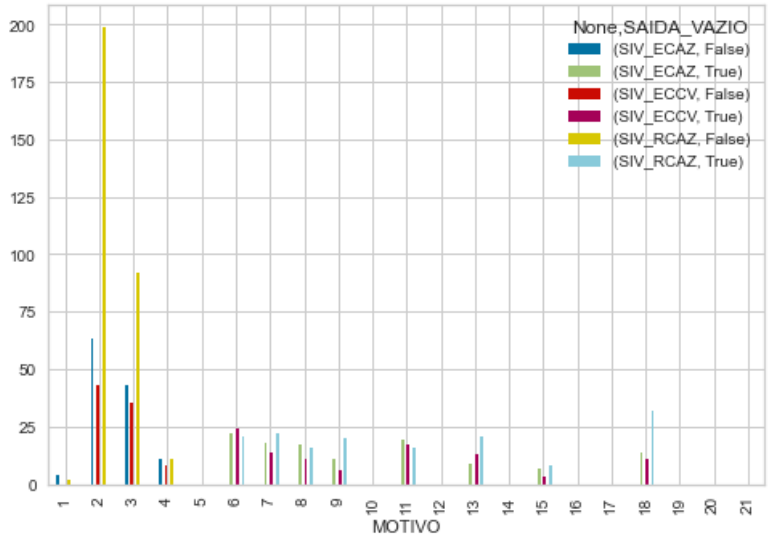
  (A leitura se faz nas tuplas das legendas considerando como *True* as *features* e o valor à direita (*True* ou *False*) se refere à *feature* SAIDA_VAZIO. E a não ocorrência no gráfico significa que as *features* têm valor *False* e SAIDA_VAZIO = *True*, sendo neste caso o total da distribuição do rótulo):



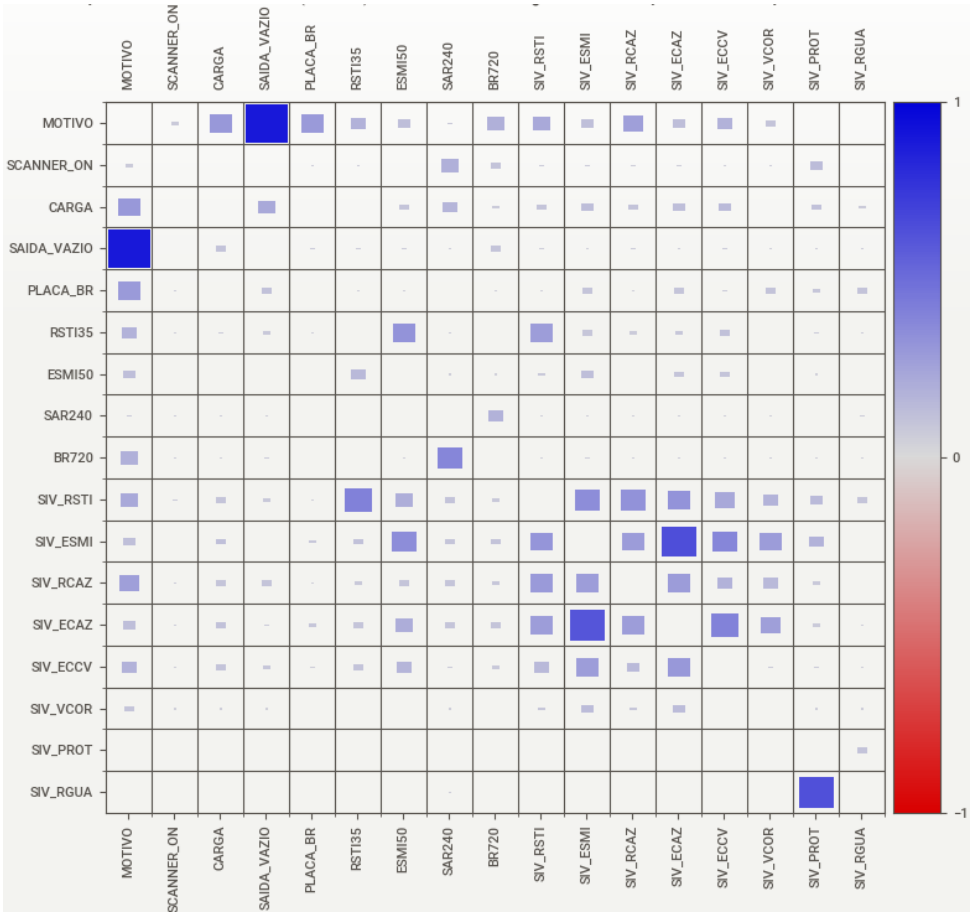

None,SAIDA_VAZIO
(RSTI35, False)
(RSTI35, True)
(SIV_RSTI, False)
(SIV_RSTI, True)
(SIV_VCOR, False)
(SIV_VCOR, True)
MOTIVO


None,SAIDA_VAZIO
(ESMI50, False)
(ESMI50, True)
(SIV_ESMI, False)
(SIV_ESMI, True)
(SIV_PROT, False)
(SIV_PROT, True)
MOTIVO


None,SAIDA_VAZIO
(CARGA, False)
(CARGA, True)
(SAR240, False)
(SAR240, True)
(SIV_RGUA, False)
(SIV_RGUA, True)
MOTIVO



- matriz – sobre toda base - com a associação entre todas as *features* que são empregadas na definição do modelo:



5. Criação de Modelos de *Machine Learning*

O modelo definido é do tipo supervisionado de classificação multiclasse, sendo usados no contexto de ML:

a) do módulo sklearn.model_selection, a função train_test_split() (para divisão do *dataset* em treinamento (train_X/train_y) – 70%; e em teste (test_X/test_y) – 30%. Este não é usado no treinamento do modelo. É salvo em disco (e logicamente excluído – variável test_X, sendo mantido test_y para comparação da predição) para posterior carga e aplicação ao modelo finalizado (também carregado do disco) para teste de predição, simulando o que deve ocorrer no dia-a-dia; e

b) do módulo pycaret.classification as demais funções (para processamento do modelo).

Nesta etapa é definido o *dataset* lógico enlastre a partir de mic_pia_label – pelo mesmo motivo expresso anteriormente. Este *dataset* (que contém todos os registros) é composto pelas seguintes *features* que são empregadas no processo de classificação, sendo a última o *target label*: 'SCANNER_ON', 'CARGA', 'SAIDA_VAZIO', 'PLACA_BR', 'RSTI35', 'ESMI50', 'SAR240', 'BR720', 'SIV_RSTI', 'SIV_ESMI', 'SIV_RCAZ', 'SIV_ECAZ', 'SIV_ECCV', 'SIV_VCOR', 'SIV_PROT', 'SIV_RGUA', **'MOTIVO'**. As *features* 'DATA_HORA' e 'PLACA' são excluídas do processamento do modelo, embora contidas no *dataset*. Deve-se observar o emprego das *features* codificadas (f_*) a partir dos parâmetros iniciais.

```
model_features = ['DATA_HORA', 'PLACA','SCANNER_ON','CARGA','SAIDA_VAZIO','PLACA_BR',f_rsti,
                  f_esmi, f_sar, f_br]+siv_features+[TARGET]
print(model_features)
```

```
['DATA_HORA', 'PLACA', 'SCANNER_ON', 'CARGA', 'SAIDA_VAZIO', 'PLACA_BR', 'RSTI35', 'ESMI50',
'SAR240', 'BR720', 'SIV_RSTI', 'SIV_ESMI', 'SIV_RCAZ', 'SIV_ECAZ', 'SIV_ECCV', 'SIV_VCOR',
'SIV_PROT', 'SIV_RGUA', 'MOTIVO']
```

Como há *labels* com poucos registros associados (de 19 a 21), deve-se ter a atenção de que os dados de treinamento (train_X/train_y) tenham todos os *labels* ou que pelo menos os de teste (test_X/test_y) estejam contidos naqueles. Com isso, como são 22 *labels*, implementou-se verificação visual para identificar se todos

constam nos dados de treinamento, sendo verde: OK; vermelho: faltando algum nos dados de treinamento.

```
#verificação se consta todos labels (rótulos) na base de treino
#a base de teste não deve ter labels ausentes na base de treino.
t_labels = show_df(train_X[TARGET].value_counts().sort_index(),[TARGET,'Total'])

if (t_labels.shape[0] < len(parametro)):
    print("ATENÇÃO: modelo não será treinado com todos Labels - verificar a base de teste!")
    display(t_labels.T.style.highlight_between(color='red'))
else:
    display(t_labels.T.style.highlight_between(color='green'))
```

MOTIVO	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Total	3165	9	176	69	13	20	20	18	17	16	8	19	11	18	11	5	6	2	27	1	2	2

5.1 Definição do Modelo

A primeira função a ser executada no processamento com pycaret é necessariamente setup(), que realiza a inicialização do ambiente de treinamento e cria o pipeline de transformação para preparar os dados para modelagem. Tem apenas dois parâmetros obrigatórios: data (= train_X) e *target*, sendo os demais opcionais (configuração default). Usou-se neste trabalho, além dos obrigatórios, o session_id (para reprodutibilidade – semelhante ao random_state do scikit-learn) e ignore_*features* (para ignorar as *features* DATA_HORA e PLACA durante o treinamento do modelo).

```
# Parâmetros de configuração do treinamento
clf1 = setup(train_X,
             target = TARGET,
             session_id=1,
             ignore_features = ['DATA_HORA','PLACA'],
            )
```

Interessante considerar algumas características (default ou inferidas) do ambiente exibidas após a execução de setup:

	Description	Value
0	session_id	1
1	Target	MOTIVO
2	Target Type	Multiclass
3	Label Encoded	None
4	Original Data	(3635, 19)
5	Missing Values	False
6	Numeric Features	0
7	Categorical Features	16
8	Ordinal Features	False
9	High Cardinality Features	False
10	High Cardinality Method	None
11	Transformed Train Set	(2544, 16)
12	Transformed Test Set	(1091, 16)

- ***Target Type* –** detectado automaticamente como ***Multiclass***, considerando que são 22 *labels* (incluindo 0);
- ***Original Data* –** exibe (3635, 19) que se refere ao **train_X** (70% dos dados – 5194 - para treinamento), sendo 3635 registros (ou instâncias) e 19 *features*;
- ***Transformed Train Set* (TTrS)–** exibe a forma (2544, 16) já transformada. Observe-se que foram excluídas as *features Target* e do parâmetro *ignore_features*, ficando com 16 *features;* e os dados de treinamento ficaram com 2544 (70% de *Original Data* – é o padrão, que pode ser alterado no parâmetro *train_size* da função *setup()*);
- ***Transformed Test Set* (TTeS) –** exibe a forma (1091,16), sendo análogo ao descrito em *Transformed Train Set,* observando-se a diferença relativa de registros.
  - Assim, os dados de treinamento, considerando a separação interna feita pelo *pycaret* sobre *Original Data*, referem-se a 49% ($[0,7]^2$) de todo *dataset*; e os de teste (validação), referem-se a 21% ([0,7*0,3]), totalizando 70%.

Considerando o requisito de comparação de modelos para escolha daquele de melhor desempenho, ao invés de definir modelos individualmente para depois compará-los, usa-se a função compare_models() do pycaret que processa os dados por meio de validação cruzada (VC) com Fold = 10 (default) fornecendo pontuação para avaliação das métricas relativas a cada modelo da biblioteca do pycaret (neste trabalho são 14, conforme abaixo). Como resultado é exibida uma tabela com as médias das métricas de acurácia (Accuracy), Recall, Precision e F1 (algumas métricas e modelo são excluídas da exibição) de todos os modelos. Embora observando também as demais métricas, no escopo deste usa-se a de acurácia.

Com a execução da função compare_models(), selecionando os 5 melhores modelos com base na acurácia e atribuindo-os a best_models, obtêm-se os valores a seguir:

```
#avalia os modelos da biblioteca, exibindo os scores.
#selecionados os 5 melhores (n_select = 5), considerando a acurácia
#excluído: <catboost> devido a erro nas etapas seguintes
best_models = compare_models(n_select = 5, exclude = ['catboost'], verbose=False)
pull()
```

	Model	Accuracy	Recall	Prec.	F1
dt	Decision Tree Classifier	0.9949	0.9386	0.9949	0.9944
et	Extra Trees Classifier	0.9941	0.9144	0.9926	0.9930
rf	Random Forest Classifier	0.9937	0.9049	0.9917	0.9923
xgboost	Extreme Gradient Boosting	0.9937	0.9020	0.9931	0.9929
svm	SVM - Linear Kernel	0.9882	0.8341	0.9848	0.9856
lr	Logistic Regression	0.9878	0.8391	0.9807	0.9838
gbc	Gradient Boosting Classifier	0.9807	0.7412	0.9808	0.9799
knn	K Neighbors Classifier	0.9709	0.6574	0.9577	0.9631
lightgbm	Light Gradient Boosting Machine	0.9524	0.8086	0.9764	0.9592
nb	Naive Bayes	0.9210	0.9248	0.9636	0.9343
ridge	Ridge Classifier	0.8899	0.3038	0.8173	0.8508
lda	Linear Discriminant Analysis	0.8825	0.4207	0.9071	0.8914
ada	Ada Boost Classifier	0.8821	0.1569	0.8076	0.8410
qda	Quadratic Discriminant Analysis	0.0000	0.0000	0.0000	0.0000

Observa-se que o modelo *Decision Tree Classifier* (‘dt’) é o que obteve melhor acurácia dentre os demais modelos. Embora com excelente desempenho, o pycaret possibilita fazer treinamento em um conjunto de modelos, que pode ser útil para melhora e generalização do modelo. Isso é feito com a função blend_models() sobre os dois melhores modelos (aquele mais *Extra Trees Classifier* – ‘et’), que são selecionados na variável top2_models. Essa função faz uma espécie de votação (voting classifier) e é definida como ‘hard’ no parâmetro method, ou seja, utiliza cada predição como 0 ou 1 na votação. Seguindo-se, aplica-se processo de ajuste de hiperparâmetros, usando-se a função tune_model(), obtendo-se as avaliações abaixo:

```
# "Votacao" com os dois melhores modelos dentre os cinco melhores
#Combinação de modelos para melhorar score individual.
top2_models = best_models[0:2] #dt + et
blender_models = blend_models(estimator_list = top2_models, method = 'hard', verbose = False)
pull()
```

```
#tune <best_models> para ajustar hiperparâmetros
#houve pequena melhora no desempenho com ajuste (tune)
t_blender_models = tune_model(blender_models, verbose = False)
pull()
```

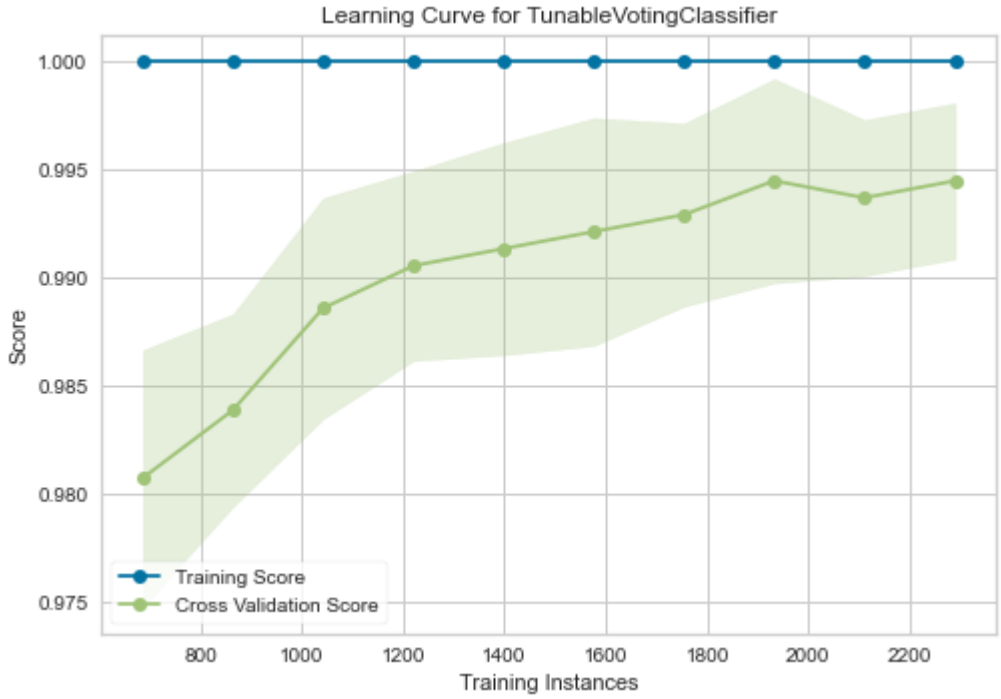
	Accuracy	Recall	Prec.	F1
0	0.9882	0.8925	0.9895	0.9873
1	0.9961	0.9945	0.9961	0.9960
2	0.9961	0.8750	0.9961	0.9961
3	0.9922	0.8235	0.9902	0.9908
4	0.9961	0.9375	0.9941	0.9948
5	0.9961	0.9688	0.9980	0.9961
6	0.9921	0.9615	0.9941	0.9921
7	1.0000	1.0000	1.0000	1.0000
8	0.9921	0.9330	0.9905	0.9909
9	1.0000	1.0000	1.0000	1.0000
Mean	0.9949	0.9386	0.9949	0.9944
SD	0.0035	0.0562	0.0037	0.0039

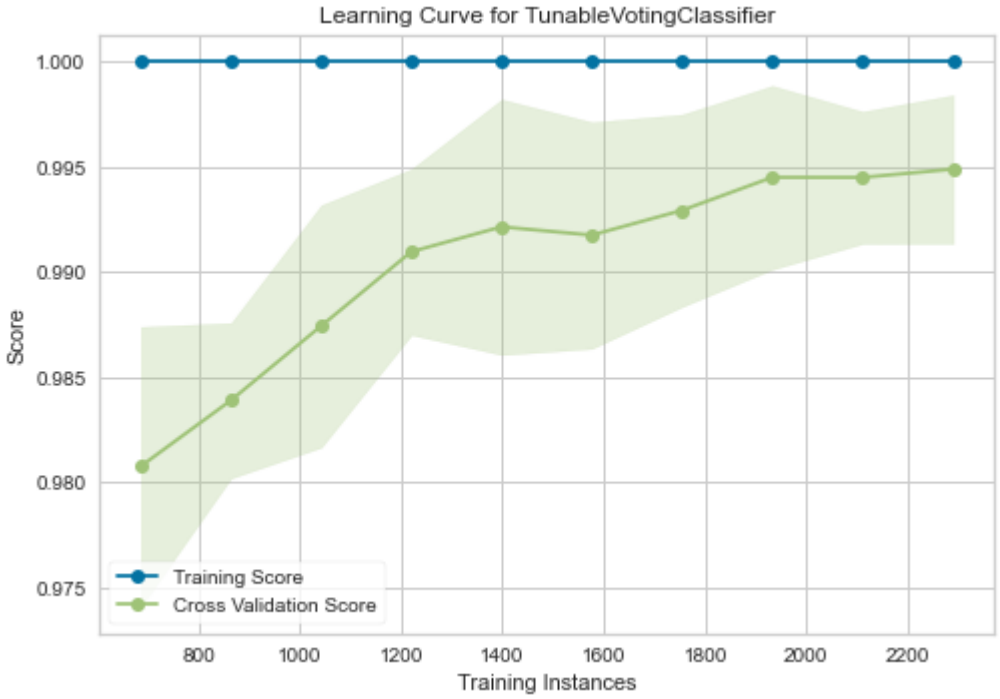
Deve-se observar que a acurácia média corresponde a do modelo ‘dt’ expresso após a execução de compare_models(). Isso é porque esta função exibe

os valores médios das métricas considerando o processamento de todos Fold da VC, que no caso, como já expresso, são 10.

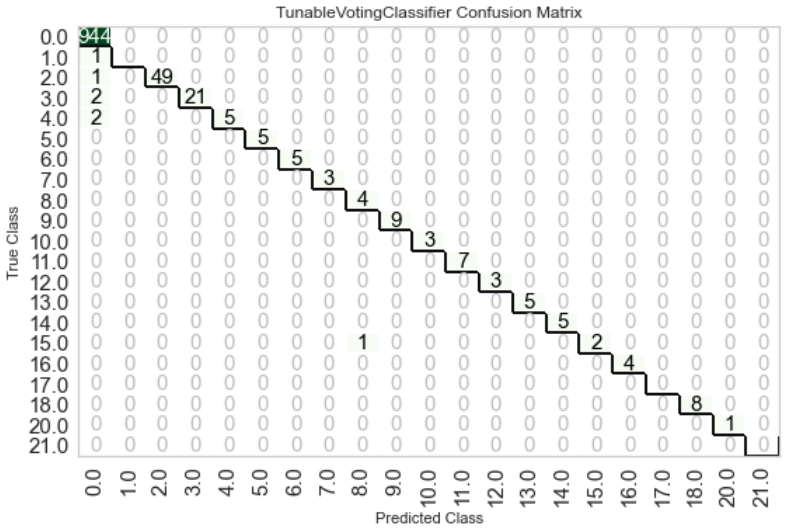
Feitos estes ajustes no modelo, é usada a função plot_model() para exibição de alguns gráficos, como matriz de confusão, curva de aprendizado e de importância das *features*, que permitem a análise de desempenho sob estes aspectos com base no treino feito com o TTrS, inclusive observando as diferenças antes e após os ajustes por tune_model().

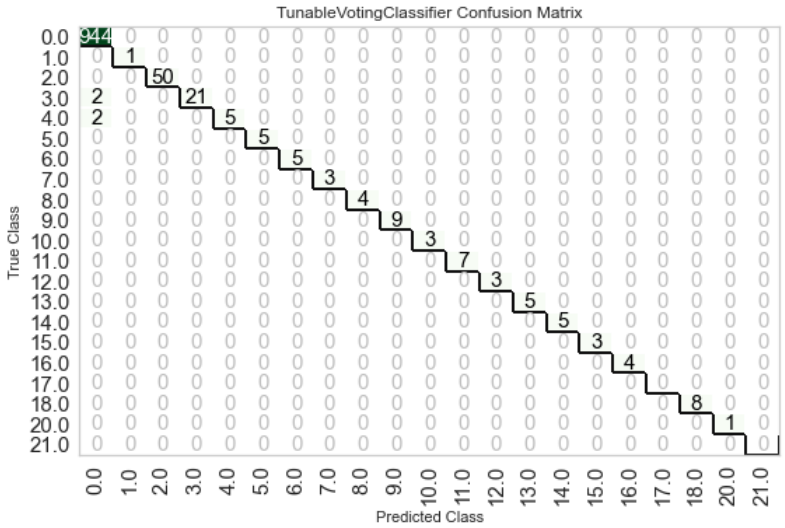
Nos gráficos de curva de aprendizado relativos ao modelo, sendo o primeiro sobre blender_models, que apresenta melhor desempenho praticamente até a primeira metade dos dados de treinamento, e o segundo sobre t_blender_models, que apresenta sutil melhora após a segunda metade, com leve declínio por volta de 1600 registros.



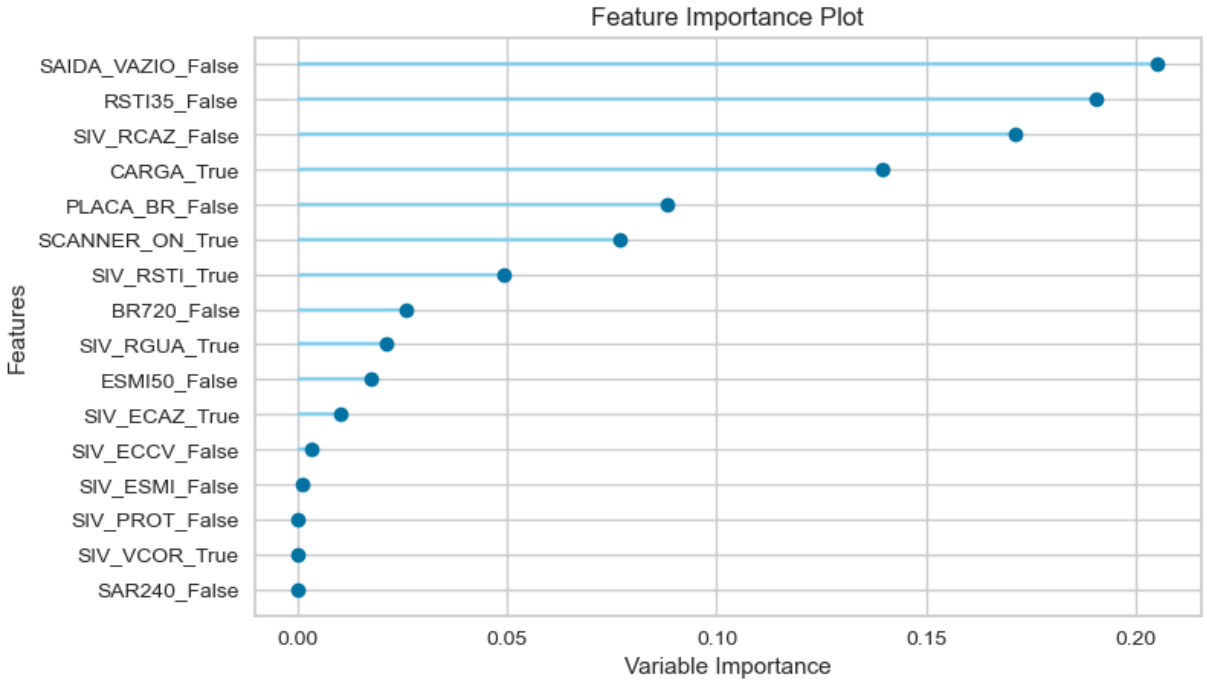


No gráfico de matriz de confusão, mantendo-se a mesma ordem acima (blender_models e t_blender_models), fica concretamente evidente a diferença apresentada na curva de aprendizado, pois se verifica a melhor assertividade na predição do segundo gráfico.





Pelo gráfico de importância das *features*, verifica-se o quanto cada *feature* influencia o treinamento do modelo. Como os registros de passagem têm uma sequência na rodovia a partir do ingresso (refletido na parametrização/rotulagem) ou em alguns há mínima ocorrência, somando-se ao fato de possível indisponibilidade do sistema LPR, isso é refletido na disposição das últimas *features* no gráfico.



Antes da finalização do modelo, considerando as avaliações das métricas anteriormente feitas com TTrS, faz-se a predição (usando a função predict_model() sobre TTeS) com uso de t_blender_models para verificação.

```
#Avaliação da predição sobre Transformed Test Set (TTeS)
predict_model(t_blender_models);
```

	Model	Accuracy	Recall	Prec.	F1
0	Voting Classifier	0.9954	0.9314	0.9947	0.9948

Observe-se que a acurácia sobre TTeS é de 0.9954 contra 0.9949 verificado no resultado anterior (com uso de blender_models). Uma pequena variação para melhor, contudo, não seria significativo mesmo se para baixo. Se houvesse uma grande variação da acurácia neste sentido sobre TTeS poderia indicar overfitting. Feita essa verificação, finaliza-se o modelo com a função finalize_model(), que treina o modelo sobre todo *dataset* (*Original Data*). Em seguida exibe-se os parâmetros de final_models por meio de sua função get_params().

```
# finaliza o modelo - treina com todo conjunto de dados TTrT +TTeT (train_X)
final_models = finalize_model(t_blender_models)
# parâmetros
final_models.get_params()
```

```
{'estimators': [('dt',
   DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                          max_depth=None, max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=1, min_samples_split=2,
                          min_weight_fraction_leaf=0.0, presort='deprecated',
                          random_state=1, splitter='best')),
  ('et',
   ExtraTreesClassifier(bootstrap=False, ccp_alpha=0.0, class_weight=None,
                        criterion='gini', max_depth=None, max_features='auto',
                        max_leaf_nodes=None, max_samples=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=-1,
                        oob_score=False, random_state=1, verbose=0,
                        warm_start=False))],
 'flatten_transform': True,
 'n_jobs': -1,
 'verbose': False,
 'voting': 'hard',
 'weights': [0.52, 0.08],
```

Como etapa final na definição do modelo, usa-se a função save_model() para salvar o modelo junto com o pipeline de transformação (Transformation Pipeline), permitindo com isso que novos dados sejam aplicados ao modelo para predição. Para formação do nome do arquivo a ser salvo, são usados data e tempo dos parâmetros definidos na configuração inicial, com isso, pode-se definir previamente vários modelos com parâmetros distintos, se necessário em produção.

```
#nome do modelo com base na data e nos tempos dos parâmetros iniciais
#ordem: T_RSTI+T_ESMI+T_SAR+T_BR
#extensão do arquivo *.pkl (Pickle File)
hoje = str(dt.date.today())
last_name = 'R'+str(T_RSTI)+'E'+str(T_ESMI)+'S'+str(T_SAR)+'B'+str(T_BR)

model_name = file_path+'S2_1-model_enlastre_'+hoje+'_'+last_name

#Salvando o modelo = file: S2_1_model_enlastre_<?>.pkl
save_model(final_models, model_name = model_name);
```

```
Transformation Pipeline and Model Succesfully Saved
```

5.2 Teste do Modelo

Para teste do modelo com dados inéditos e aproveitando para simular como deve ser o procedimento para predição de dados novos, foi carregado na variável S2_model_enlastre o modelo salvo na seção anterior, usando-se a função load_model() do pycaret.

```
#carregaando o modelo salvo
S2_model_enlastre = load_model(file_path+'S2_1-model_enlastre_2022-02-22_R35E50S240B720')
```

```
Transformation Pipeline and Model Successfully Loaded
```

A predição com os dados inéditos que foram inicialmente separados para teste do modelo (test_X) é realizada na sequência, após carga deste *dataset*, sendo retornadas na variável tx_predictions as predições:

```
#carregamento do dataset de teste - salvo em disco
test_X = pd.read_csv(file_path+start_file+'teste_enlastre.csv', sep=';')\
.set_index('Unnamed: 0')

# #teste do modelo - dataset inédito: <test_X>
tx_predictions = S2_model_enlastre.predict(test_X)
```

Em seguida verificou-se o percentual de acerto das predições usando-se a função score() da variável S2_model_enlastre, obtendo-se ~99,81%, sendo 3 registros com divergência de predição num universo de 1559.

```
#verificação do percentual de acerto da predição
perc_acerto = S2_model_enlastre.score(test_X, test_y)
print("Predição de Teste com {0:.2f}% de acerto!".format(perc_acerto*100))
```

```
Predição de Teste com 99.81% de acerto!
```

Adicionalmente, definiu-se um dataframe (test_pred – colunas: DATA_HORA, PLACA, MOTIVO, MOTIVO_PRED e PARAMETRO) apenas para verificação, sendo gerado relatório (planilha) da predição. E analisando-se as divergências da predição, uma (MOTIVO=11) foi classificada em *label* (MOTIVO_PRED=15) cujos parâmetros são bem próximos, perdendo-se apenas a especificidade, mas considerada aceitável a classificação; os outros dois ficaram sem classificação (MOTIVO_PRED=0), contudo, dado o fim a que se destina este trabalho (fora do escopo do TCC), é melhor que a classificação seja para 0 do que o inverso, o que não significa que não seja posteriormente avaliada, principalmente com mais experimentos neste contexto.

```
#verifica os registros com divergências de predição
test_pred[[TARGET,TARGET_PRED]][test_pred[TARGET] != test_pred[TARGET_PRED]]\
    .style.set_caption('PREDIÇÕES DIVERGENTES - TESTE')
```

PREDIÇÕES DIVERGENTES - TESTE

	MOTIVO	MOTIVO_PRED
3703	4	0
3058	3	0
4603	11	15

6. Apresentação dos Resultados

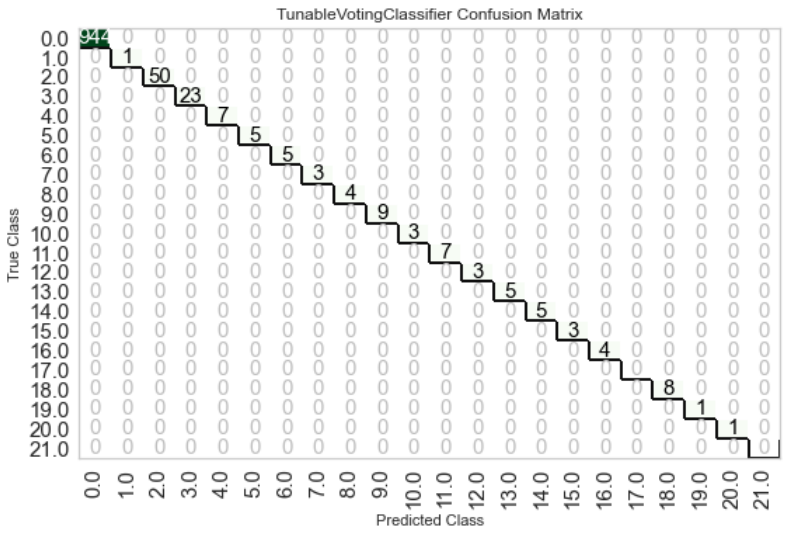
6.1 Modelo de Canvas (Vasandani)

Classificação parametrizada do trânsito de caminhões "*en lastre*" com base em registros de passagens		
Definição do Problema	**Resultados/Previsões**	**Aquisição dos Dados**
Necessidade de prover meios seletivos de fiscalização de caminhões *en lastre*, considerando que a quantidade de veículos é muito grande e somada à falta de pessoal, torna-se inviável esta atividade a contento, potencializando o risco de "ingressos irregulares" no país.	Classificação parametrizada dos registros de passagens dos caminhões *en lastre*. Predição da *target feature* MOTIVO (22 *labels* incluindo o 0) a partir do *dataset* train_X (treinamento e validação).	Os dados são de uso do Órgão contratante do Curso de Especialização e são extraídos de sistemas usados internamente. São obtidos diretamente ou por solicitação.
Modelagem	**Avaliação do Modelo**	**Preparação dos Dados**
Uso de algorítimo de ML supervisionado, sendo processada a classificação por meio da seleção/combinação de modelos, usando-se o módulo *pycaret.classification* que treina e classifica os registros com base em 22 *labels* atribuídos a cada registro conforme parâmetros de rotulagem definidos.	Uso principal da métrica de acurácia para avaliação, com exibição de outras métricas**,** sendo avaliados em fase de treinamento, validação e teste, bem como, ao final sobre todo *dataset* enlastre (train_X + test_X)	Considerando o volume de dados obtidos e de sistemas distintos, somando-se a isto a preocupação quanto a dados sensíveis no contexto de privacidade/sigilo, foram necessários: remoção de vários dados (principalmente os que identificam PF ou PJ); exclusão de registros inválidos, em duplicidade ou desnecessários; correção de dados; criação de registros tanto para processamento quanto para uso do modelo; transformações de tipos de dados.

6.2 Modelo - combinação

Como exposto, após a avaliação das médias das métricas de acurácia para cada modelo processado pelo pycaret, houve a seleção, para definição do modelo, de 2 algorítmos de melhor desempenho (Decision Tree Classifier; e Extra Trees Classifier), dentre 5 selecionados, que foram combinados e então o modelo finalizado (final_models). Essa combinação, além de poder melhorar e generalizar o modelo, pode torná-lo mais robusto com predições de diferentes algoritmos.

Fazendo-se verificação da matriz de confusão sobre o modelo finalizado, observa-se a diferença daquele apresentado sobre t_blender_models anteriormente, demonstrando melhora, considerando que o modelo é finalizado sobre o *dataset* de treinamento (train_X = TTrS + TTeS):



Ademais, confirmaram-se os excelentes valores de acurácia (0.9954) durante a fase de treinamento, pois em aplicação do modelo sobre os dados de teste – test_X – obteve-se taxa de acerto de ~0.9981, que está dentro da margem, considerando-se um desvio padrão médio de 0.0035. E quanto às 3 divergências de classificação, são consideradas dentro do esperado conforme valores apresentados até a finalização do modelo.

A título de experimento, fez-se a predição de toda a base de dados (*dataset* lógico enlastre - 5194 registros) também com uso do modelo salvo – pois, na prática,

o procedimento será este: dados novos sendo submetidos ao modelo salvo (treinado) para se obter as predições.

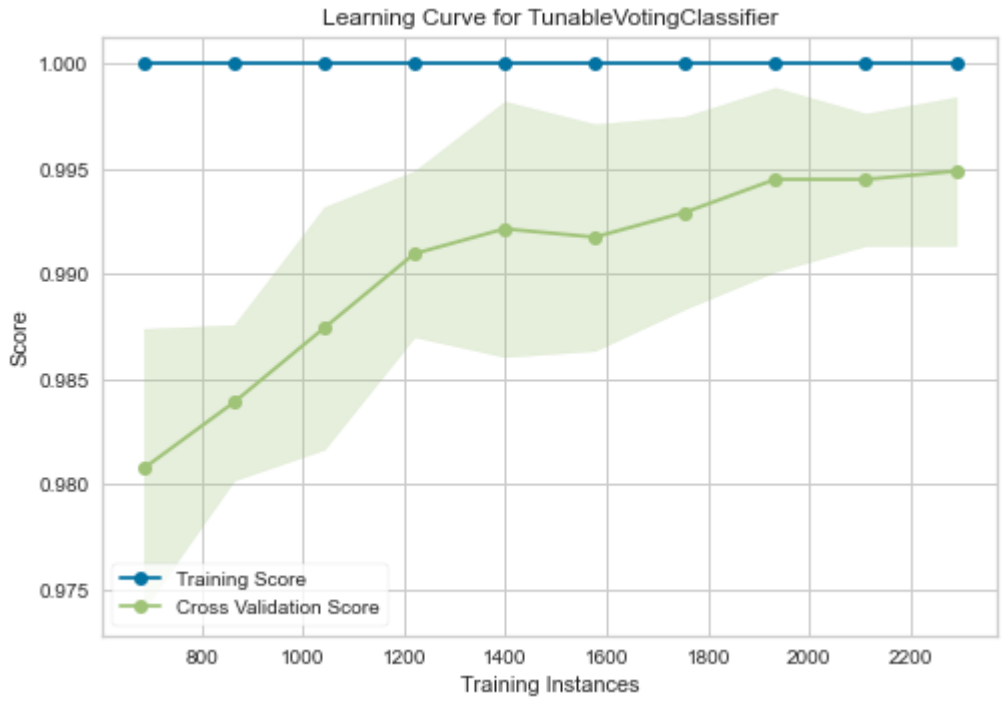
```
#test_all recebe cópia de todo dataset <enlastre>
test_all = enlastre.copy()
y = test_all.pop(TARGET)

# aplicando o modelo carregado - usando toda a base
te_predictions = S2_model_enlastre.predict(test_all)
```

```
#verificação do percentual de acerto da predição
perc_acerto = S2_model_enlastre.score(test_all, y)
print("Predição de Teste de toda base com {0:.2f}% de acerto!".format(perc_acerto*100))
```

```
Predição de Teste de toda base com 99.94% de acerto!
```

Observou-se as mesmas (3) divergências apontadas no teste (sobre test_X), contudo, como são mais dados, houve incremento do percentual de acerto, que está em consonância com o gráfico da curva de aprendizado de t_blender_models, considerando a margem de variabilidade positiva.



Faz-se a observação de que, em avaliação individual com o modelo ‘dt’ (atribuindo-o exclusivamente a top2_models e seguindo com idêntico procedimento), obteve-se o mesmo resultado de acurácia (e demais métricas) daquele com os dois

modelos combinados, após execução de tune_models(), sendo estes mantidos por esse motivo (sem alteração do resultado) e para avaliação quando os parâmetros se tornarem mais complexos, uma vez que a combinação torna o modelo mais robusto com diferentes algoritmos, podendo ser útil a sua generalização.

Dada essa complementar exposição à seção precedente, considera-se atingido o objetivo de classificação seja pela avaliação da acurácia, seja pela predição de fato, fazendo-se a observação de que os parâmetros/*features* empregados foram reduzidos no escopo deste TCC devido a questões de privacidade, contudo, na atividade laboral interna, algumas variáveis não serão removidas (e outras incluídas), tornando o processo mais complexo e eficiente sob o ponto de vista da seletividade. Com isso, provavelmente a acurácia não será tão alta quanto às apresentadas no escopo deste, onde foram processadas pelo modelo apenas *features* booleanas.

6.3 Relatórios (planilhas)

Adicionalmente, são gerados relatórios para subsidiar o trabalho de seleção de quais caminhões devem ser verificados dentro do universo daqueles que foram rotulados. Este é o motivo central de se ter vários rótulos - permitir a análise de conveniência e oportunidade nessa seleção, focando nos motivos de interesse à fiscalização num dado momento, principalmente pela falta de recursos humanos para lidar com crescente demanda.

Optou-se pela geração em planilhas Excel usando-se principalmente as funções do pandas groupby(), pivot_table() e crosstab(), tendo-se como objetivos:

- agrupar os registros por PLACA, MOTIVO e PARAMETRO, exibindo todos os rótulos (MOTIVO) relativos a uma determinada placa com respectivo parâmetro e quantidade de ocorrências:
  - S2_*-report_placa-motivo_descricao.xlsx
- agrupar por PLACA - MOTIVO/MOTIVO – PLACA, permitindo verificar, para cada placa/motivo, a quantidade de ocorrências e de SAIDA_VAZIO dentro de cada mês:

  - S2_*-report_mensal_motivo-placa.xlsx
  - S2_*-report_mensal_placa-motivo.xlsx
- agrupar por PLACA - MOTIVO/MOTIVO – PLACA, permitindo verificar, para cada placa/motivo, os parâmetros usados na rotulagem, tendo-se o cruzamento de valores (booleanos) relativos a cada *feature*, inclusive com a quantidade de ocorrências (parcial) e de registros (total):
  - S2_*-report_motivo-placa_cruzamento.xlsx
  - S2_*-report_placa-motivo_cruzamento.xlsx

7. Links

Link para o vídeo: https://youtu.be/eapsdOTzfE0
Link para o repositório: https://github.com/junior-py/PUCMG-TCC

APÊNDICE

Considerando a natureza dos dados, como já exposto, não são disponibilizados os *datasets* originais do trabalho. E aqueles contidos neste tópico encontram-se com algumas colunas anonimizadas, assim como, alguns arquivos (*scripts*) têm codificação suprimida também para esse fim. Os *scripts* <S1 – S1_2>, originalmente *notebooks*, são disponibilizados em formato HTML **(*)**.

Os arquivos encontram-se organizados no GitHub, com a descrição abaixo:

Escrita/Apresentação do TCC:

- PUCMG-CDBD-TCC-RFB-OsmarBCJunior.pdf - este
- PUCMG-CDBD-TCC-RFB-PRESENT.pptx - síntese

Programação/*Notebooks* – arquivos comentados na seção **3.1 Workflow :**

- **S1**-ProcTrataDados-mic.ipynb (*)
- **S1**-ProcTrataDados-pia.ipynb (*)
- **S1**-ProcTrataDados-piabr.ipynb (*)
- **S1**-ProcTrataDados-siv.ipynb (*)
- **S1**-ProcTrataDados-sar.ipynb (*)
- **S1_1**-ProcTrataDados-Extract.ipynb (*)
- **S1_2**-Anonimizacao.ipynb (*)
- **S2**-AnaliseExploraDados-Modelo.ipynb
- tcc_cdbd.py

Folder: [00-DataSets]

- ***Datasets* anonimizados –** carregados por **<S2>:**
  - **S1_2**-ProcTrataDados-mic.csv
  - **S1_2**-ProcTrataDados-pia.csv
  - **S1_2**-ProcTrataDados-piabr.csv
  - **S1_2**-ProcTrataDados-siv.csv – (compactado *.zip)
  - **S1_2**-ProcTrataDados-sar.csv

- **Modelo salvo –** carregado em **<S2>:**
  - **S2_1**-model_enlastre_2022-02-22_R35E50S240B720.pkl

Folder: [Report]

- **Planilhas** – arquivos comentados na seção **6.3 Relatórios (planilhas)**:
  - **S2_2**-report_placa-motivo_descricao.xlsx
  - **S2_2**-report _motivo-placa_mensal.xlsx
  - **S2_2**-report_placa-motivo_mensal.xlsx
  - **S2_2-**report_motivo-placa_cruzamento.xlsx
  - **S2_2**-report_placa-motivo_cruzamento.xlsx

- **EDA-Relatórios/Gráficos –** uso da biblioteca **Sweetviz:**
  - **S2_1**-eda-report_enlastre.html - todo *dataset*;
  - **S2_1**-eda-report_enlastre-No0Label.html - todo *dataset* (sem label 0);
  - **S2_1**-eda-report_train_test.html - comparação train_X - test_X