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

ANDRÉ LEONARDO DE LA CORTE

APLICAÇÃO DE TÉCNICAS DE MACHINE LEARNING PARA PREDIÇÃO DE TEMPO DE DESEMBARAÇO ADUANEIRO

Belo Horizonte
2021

André Leonardo de La Corte

APLICAÇÃO DE TÉCNICAS DE MACHINE LEARNING PARA PREDIÇÃO DE TEMPO DE DESEMBARAÇO ADUANEIRO

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

Belo Horizonte

2021

SUMÁRIO

1. Introdução

O presente trabalho visa analisar a aplicação de técnicas de machine learning na predição do tempo de despacho aduaneiro de importação na modalidade marítima, utilizando dados públicos disponibilizados pela Secretaria Especial da Receita Federal do Brasil.

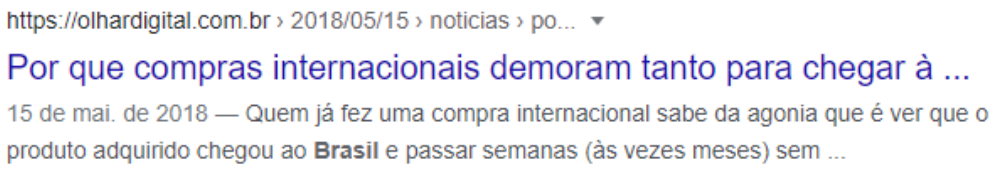
O estudo está dividido em 6 partes, sendo que esta introdução apresenta a contextualização e importância do tema proposto. Em seguida será apresentada a coleta e tratamento dos dados. Ao final serão apresentados os modelos de machine learning utilizados e seus resultados.

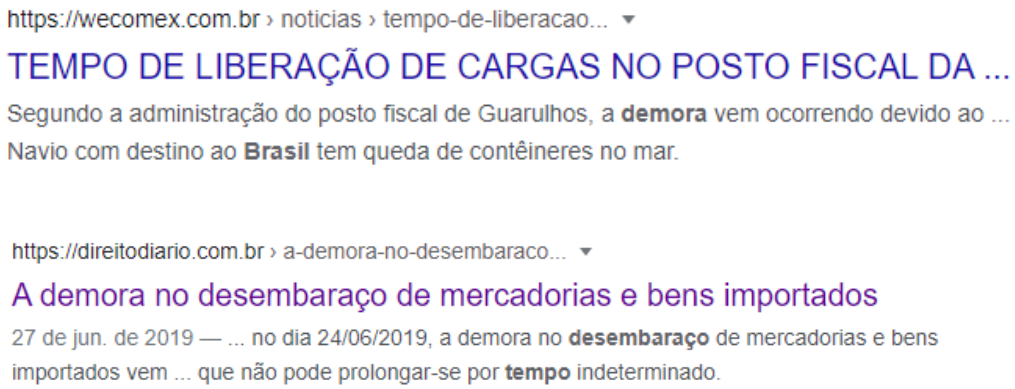
1.1. Contextualização

O comércio internacional desempenha importante papel no desenvolvimento econômico de todos os países. O crescente fluxo comercial entre as nações gera uma crescente necessidade de aperfeiçoamento dos processos de liberação de mercadorias nas fronteiras internacionais, por meio da simplificação e harmonização das práticas aduaneiras e das demais agências de controles fronteiriços, com garantia de fluxos comerciais mais seguros, mais previsíveis e mais rápidos.

O desembaraço aduaneiro de importação é uma etapa, sem dúvidas, muito importante do processo de importação, a qual depende muitas vezes da interação de contribuinte, Receita Federal do Brasil (RFB) e demais órgão anuentes, e que é responsável por boa parte do tempo deste processo.

Trata-se, portanto, de um tema estratégico no desenvolvimento nacional e que é objeto constante de críticas na mídia.





A relevância do assunto, no entanto, ultrapassa a fronteira nacional e de sua discussão internacional fora criado o Acordo de Facilitação de Comércio (AFC) no âmbito da Organização Mundial do Comercio (OMC). Tal, acordo, do qual o Brasil é signatário, possui um artigo dedicado à liberação e despacho aduaneiro de bens, que, em seu item 6, define o estabelecimento e publicação do tempo médio de liberação.

A Organização Mundial da Aduanas (OMA) desenvolveu uma metodologia para o estudo do tempo de liberação de cargas a ser aplicada pelos países interessados, sobretudo os signatários do AFC.

O Time Release Study (TRS) é, portanto, uma ferramenta desenvolvida pela OMA para medição da eficiência operacional dos mais relevantes procedimentos conduzidos pela aduana, pelos órgãos anuentes e pelos intervenientes do setor privado nos processos de importação, exportação e trânsito aduaneiro de mercadorias. O TRS objetiva apurar os tempos para a liberação de mercadorias desde sua chegada até sua efetiva saída da área sob controle aduaneiro, apontando possíveis medidas corretivas e de aprimoramento de performance dos participantes do processo.

A RFB publicou seu Estudo TRS em 2020, utilizando dados de declarações de importações dos meses de junho e julho do ano de 2019. Junto com o estudo foram disponibilizados diversos conjuntos de dados na forma de planilhas excel e libreoffice.

Ainda para ilustrar como o tema é importante para o órgão, o Grau de Fluidez de Despacho de Importação, ou seja, e o percentual de declarações de importação desembaraçadas em menos de 24 horas, é um dos indicadores de Eficiência e Produtividade constantes da Portaria RFB nº 31, de 18 de janeiro de 2017.

1.2. Sobre o desembaraço Aduaneiro

O artigo 237 da Constituição Federal de 1988 dá ao Ministério da Economia a responsabilidade pela administração aduaneira no Brasil . O Ministério da Fazenda, no artigo primeiro do Anexo I da Portaria MF nº 403 definiu a competência da RFB para a execução das atividades de controle aduaneiro. A RFB, órgão do Ministério da Economia, também é responsável pela administração tributária.

De acordo com o Decreto-lei nº 37, de 1966, toda mercadoria procedente do exterior deverá ser submetida a despacho aduaneiro, o qual será processado com base em declaração de importação (DI) apresentada à repartição aduaneira. O despacho aduaneiro de importação está disciplinado na Instrução Normativa da Receita Federal nº 680(BRASIL, 2006), a qual estabelece que a DI deve ser registrada pelo importador no sistema SISCOMEX.

O Sistema Integrado de Comércio Exterior – SISCOMEX, instituído pelo Decreto nº10660, é um sistema informatizado responsável por integrar as atividades de registro, acompanhamento e controle das operações de comércio exterior, através de um fluxo único e automatizado de informações. No SISCOMEX, a DI é identificada pela sua numeração automática única, que é sequencial e nacional, reiniciada a cada ano.

O processo do controle aduaneiro da importação inicia com a chegada da mercadoria ao País e pode ser dividido em três fases ou etapas: pré-despacho, despacho aduaneiro e pós-despacho. Na fase pré-despacho, o controle logístico da carga é realizado pela RFB de forma automática, com uso de sistemas próprios para o controle da carga conforme o modal (marítimo, aéreo e rodoviário). Na fase do despacho aduaneiro, é realizada a conferência documental e/ou física da mercadoria, que é então liberada para integrar a economia nacional. Na fase pós despacho, os importadores fiscalizados e as suas DI são revisadas.

Como definido no Regulamento Aduaneiro, o despacho aduaneiro de importação é o procedimento mediante o qual é verificada a exatidão dos dados declarados pelo importador em relação à mercadoria importada, aos documentos apresentados e à legislação específica. A Instrução Normativa SRF nº 680 disciplina que o despacho de importação inicia na data do registro da respectiva DI e é concluído com o

desembaraço aduaneiro, ato pelo qual é registrada a conclusão da conferência aduaneira e liberada a mercadoria ao importador.

Uma vez registradas, as DIs são submetidas a um procedimento de parametrização, que consiste na sua seleção para canais de conferência aduaneira, de acordo com a estratégia de gestão de riscos da RFB. A seleção para canal de conferência aduaneira é efetuada por intermédio do SISCOMEX e leva em consideração, entre outros, os seguintes elementos: o perfil e a habitualidade do importador, o volume e o valor da importação.

O canal verde implica a dispensa de conferência aduaneira, sendo a DI desembaraçada automaticamente; as DI selecionadas para o canal amarelo submetem-se ao exame documental; as DI selecionadas para o canal vermelho passam pelo exame documental e conferência física das mercadorias; as DI selecionadas para o canal cinza, em caráter excepcional, para procedimento fiscal diferenciado.

Para as DI submetidas aos canais amarelo, vermelho ou cinza, o importador deverá anexar no SISCOMEX os documentos digitalizados relativos ao despacho, por exemplo, o conhecimento de transporte, a fatura comercial e o *packing list*. Este procedimento de anexação ou entrega dos documentos é também conhecido como recepção. Anteriormente, esse era um procedimento manual, em que a fiscalização aduaneira registrava no sistema SISCOMEX a recepção da documentação entregue.

1.3. Sobre a Licença de Importação

A Licença de Importação (LI) é um documento por meio do qual o Governo autoriza a importação realizada por uma empresa ou pessoa física, mediante verificação do cumprimento de normas legais e administrativas. Ela é necessária quando a importação que se pretende realizar está sujeita à anuência de um ou mais órgão anuentes (como DECEX, ANVISA, MAPA, INMETRO, etc).

A LI é obtida para cada Adição de uma DI, ou conjunto de bens que compartilham a mesma classificação fiscal, e, portanto, pode haver mais de uma LI para cada DI.

1.4. Sobre o Trânsito Aduaneiro

O Trânsito Aduaneiro é um regime especial que permite o transporte de mercadoria, sob controle aduaneiro, de um ponto a outro do território aduaneiro, com suspensão do pagamento de tributos.

O regime subsiste do local de origem ao local de destino e desde o momento do desembaraço para trânsito aduaneiro pela unidade de origem até o momento em que a unidade de destino conclui o trânsito aduaneiro.

No caso do processo de importação, isso significa que a unidade de entrada da carga não será a mesma a realizar o despacho de importação.

1.5. O problema proposto

O presente projeto tem como finalidade avaliar se com determinadas variáveis de baixa complexidade é possível se estabelecer um modelo de aprendizagem de máquina para realizar a predição do tempo entre o registro e o desembaraço de DIs da modalidade marítima.

Why

A predição do tempo de desembaraço traz diversos benefícios como a possibilidade de ajustes na alocação da força de trabalho das unidades aduaneiras e a melhoria das métricas de desempenho e produtividade. Já pelo lado do contribuinte permite uma racionalização da logística envolvida na operação.

Who

Os dados analisados neste projeto pertencem à Secretaria Especial da Receita Federal do Brasil, um órgão do Ministérios da Fazenda. Trata-se de dados da base de DIs registradas no SISCOMEX - Sistema Integrado de Comércio Exterior, um instrumento administrativo que integra as atividades de registro, acompanhamento e controle das operações de comércio exterior.

What

O objetivo do projeto é realizar a exploração dos dados básicos extraídos das DIs direcionadas para canal verde, amarelo e vermelho a fim de observar possível

impacto no tempo de desembaraço, e a partir desses dados construir modelos de Aprendizagem de Máquina Supervisionados e comparar seus resultados.

Where

Os dados obtidos abrangem todo o território nacional, englobando DIs da modalidade “Consumo” no modal marítimo.

When

O período analisado abrange DIs registradas nos meses de junho e julho do ano de 2019.

2. Coleta de Dados

Os dados obtidos para o presente estudo tiveram origem no estudo TRS da Receita Federal do Brasil, e estão disponíveis para download na página da RFB na internet.

Trata-se de um conjunto de planilhas eletrônicas para cada modal de transporte. O presente trabalho aborda os dados do modal marítimo.

O relacionamento entre os datasets é feito pelo *id* das declarações selecionadas, uma chave artificial numérica.

O primeiro conjunto foi adotado como principal e possui a maior quantidade de utilizados. Apresenta 146.937 registros, cada um representando uma DI e 24 colunas.

O segundo conjunto apresenta dados relativos às LIs, contendo 150.489 registros, cada um representando LI de uma Adição pertencente a uma DI, e 41 colunas.

O terceiro conjunto conta informações gerais de uma DI, porém apresenta a data do Trânsito Aduaneiro, caso tenha ocorrido. Apresenta 184.742 registros, representando uma DI cada e conta com 20 colunas.

Abaixo temos a descrição sumária dos campos utilizados em cada conjunto, bem como seus tipos:

Conjunto 1 – Dataset Declarações de Importação

Nome da coluna/campo	Descrição	Tipo
ID DI	Identificador único para uma DI	Numérico Inteiro
TIPO CE	Indica o tipo de conhecimento de transporte utilizado na operação de importação	Texto
OEA	Indicador de empresa importadora credenciada no programa Operador Econômico Autorizado (OEA)	Texto

MODALIDADE DESPACHO	Define a forma como foi efetuado o despacho de importação.	Texto
UL PRESENCA	Código da Unidade Local onde foi efetuada a presença da carga no processo de importação.	Numérico Inteiro
DESCRICAO UL	Código da Unidade Local onde foi efetuada a presença da carga no processo de importação.	Texto
RA PRESENCA	Código do Recinto Aduaneiro onde foi efetuada a presença de carga no processo de importação	Numérico Inteiro
DESCRICAO RA	Nome do Recinto Aduaneiro onde foi efetuada a presença de carga no processo de importação	Texto
TIPO DECLARACAO IMPORTACAO	Define o regime de importação de acordo com a origem ou destino da mercadoria.	Texto
CANAL	Indica o canal de conferência selecionado para a declaração. Amarelo indica exame documental e Vermelho indica exame documental e físico.	Texto
DATA ATRACACAO	Data da atracação da embarcação no porto de destino	Data e Hora
DATA PRESENCA	Data em que ocorreu o evento presença de carga no Recinto Aduaneiro	Data e Hora
DT REGISTRO DI	Data em que foi efetuado o registro da DI	Data e Hora
DT_SELECAO	Data em que a DI foi selecionada para canal	Data e Hora
DT_RECEPCAO	Data em que o contribuinte efetuou a entrega dos documentos comprobatórios da importação	Data e Hora
DT_DISTRIBUICAO	Data em que a DI foi distribuída para procedimento de fiscalização	Data e Hora

HOSRAS_EXIG	Quantidade de horas que o contribuinte levou para o atendimento das exigências fiscais	Numérico Decimal
DT DESEMBARACO DI	Data em que a DI foi efetivamente desembaraçada	Data e Hora
DT ENTREGA	Data em que a carga proveniente do exterior foi entregue ao importador em território nacional	Data e Hora

Conjunto 2 – Dataset Licenciamento de Importação

Nome da coluna/campo	Descrição	Tipo
ID LI	Identificador único para uma licença de importação	Número Inteiro
ID DI	Identificador único para uma DI	Número Inteiro
ANUENTE	Nome do primeiro órgão anuente da licença de importação	Texto
ANUENTE2	Nome do segundo órgão anuente da licença de importação, caso haja mais de um	Texto
ANUENTE6	Nome do terceiro órgão anuente da licença de importação, caso haja mais de dois	Texto
ANUENTE10	Nome do quarto órgão anuente da licença de importação, caso haja mais de três	Texto
DT REGISTRO DI	Data em que foi efetuado o registro da DI	Data e Hora
DATA DEFERIMENTO	Data em que houve deferimento da Licença de Importação	Texto

Conjunto 3 – Dataset Trânsito Aduaneiro

Nome da coluna/campo	Descrição	Tipo
ID DI	Identificador único para uma DI	Número Inteiro
DATA TRANSITO	Data em que teve início o procedimento de trânsito aduaneiro	Texto

3. Processamento/Tratamento de Dados

3.1. Dados Declaração de Importação

O primeiro conjunto de dados, que chamaremos de dataset_di, contém informações em nível de DI, e será usado como dataset principal neste trabalho.

Sobre este dataset serão efetuadas várias operações para a definição de variáveis quantitativas e qualitativas para utilização nos modelos de machine learning, processo conhecido como *feature engineering*.

3.1.1. Quantidade Horas Despacho

A quantidade de horas do Despacho Aduaneiro (QTDE HORAS DESPACHO) será a variável alvo (label) do presente estudo, e representa o tempo bruto, em horas, no qual a DI permaneceu sob procedimento fiscal.

Não pertencendo ao conjunto original do dataset, será obtida através da diferença entre a data de desembaraço (DT DESEMBARACO DI) e a data de registro (DT REGISTRO DI) da DI.

```
In [11]: df_di['QTDE HORAS DESPACHO'] = (df_di['DT DESEMBARACO DI'] - df_di['DT REGISTRO DI'])/pd.Timedelta(hours=1)

In [12]: df_di[['DT DESEMBARACO DI','DT REGISTRO DI','QTDE HORAS DESPACHO']].head()
```

Out[12]:

	DT DESEMBARACO DI	DT REGISTRO DI	QTDE HORAS DESPACHO
0	2019-06-21 17:40:40	2019-06-21 11:28:42	6.199444
1	2019-07-19 12:10:46	2019-07-18 14:49:35	21.353056
2	2019-06-03 18:10:45	2019-06-03 09:22:05	8.811111
3	2019-07-19 18:10:47	2019-07-19 12:58:42	5.201389
4	2019-07-24 10:34:06	2019-07-24 10:34:06	0.000000

Figura 1 - Criação da variável alvo QTDE HORAS DESPACHO

3.1.2. Quantidade Horas Presença de Carga

A quantidade de horas para a presença de carga (QTDE HORAS PRESENCA) representa a quantidade de horas entre a atracação da embarcação e a armazenagem da carga nos recintos aduaneiros. Serve como um indicador da complexidade logística envolvida na operação de importação, pois pode depender do tipo, quantidade e características da carga, bem como das instalações portuárias, e que pode influenciar

em outras etapas do processo de importação, como o a vistoria da carga em caso de canal de conferência, no qual a carga terá que ser movimentada.

Da mesma forma que a variável anterior, será obtida pela diferença entre a data da atracação (DATA ATRACACAO) e a data da presença de carga (DATA PRESENÇA), representando o tempo bruto para a movimentação da carga.

```
In [13]: df_di['QTDE HORAS PRESENCA'] = (df_di['DATA PRESENCA'] - df_di['DATA ATRACACAO'])/pd.Timedelta(hours=1)

In [14]: df_di[['DATA ATRACACAO','DATA PRESENCA','QTDE HORAS PRESENCA']].head()
```

Out[14]:

	DATA ATRACACAO	DATA PRESENCA	QTDE HORAS PRESENCA
0	2019-06-19 08:07:00	2019-06-19 20:19:00	12.200000
1	2019-07-07 15:08:00	2019-07-17 16:45:00	241.616667
2	2019-05-28 10:07:00	2019-05-30 02:16:00	40.150000
3	2019-07-14 18:27:00	2019-07-16 04:38:00	34.183333
4	2019-07-23 13:08:00	2019-07-23 22:02:00	8.900000

Figura 2- Criação da variável Quantidade de Horas para a Presença de Carga

3.1.3. Quantidade Horas Distribuição

A quantidade de horas para a distribuição representa o tempo bruto, em horas, para que uma DI direcionada para canal de conferência amarelo ou vermelho leva para ser distribuída para um Auditor-Fiscal iniciar o procedimento de fiscalização.

É obtido pela diferença entre a data de registro (DT REGISTRO DI) e a data de distribuição (DT_DISTRI) e independe de qualquer ação do importador.

```
In [15]: df_di['QTDE HORAS DISTRIBUICAO'] = (df_di['DT_DISTRI'] - df_di['DT REGISTRO DI'])/pd.Timedelta(hours=1)

In [16]: df_di[['DT REGISTRO DI','DT_DISTRI','QTDE HORAS DISTRIBUICAO']][df_di['QTDE HORAS DISTRIBUICAO']>0].head()
```

Out[16]:

	DT REGISTRO DI	DT_DISTRI	QTDE HORAS DISTRIBUICAO
46	2019-05-14 12:16:38	2019-05-16 11:13:00	46.939444
53	2019-06-18 11:14:00	2019-06-21 10:03:00	70.816667
104	2019-03-12 18:51:34	2019-03-14 16:21:00	45.490556
122	2019-05-29 17:13:57	2019-05-30 11:25:00	18.184167
306	2019-05-17 13:25:18	2019-05-23 09:41:00	140.261667

Figura 3 - Criação da variável Quantidade de Horas para a Distribuição

3.1.4. Quantidade Horas Recepção

A quantidade de horas para a recepção de documentos representa o tempo que importador leva para anexar os documentos instrutivos do Despacho Aduaneiro de Importação em uma DI direcionada para canal de conferência.

É obtida pela diferença entre a data de registro da DI (DATA REGISTRO DI) e a data da recepção dos documentos instrutivos no sistema (DT RECEPCAO), e depende exclusivamente da ação do importador.

```
In [17]: df_di['QTDE HORAS RECEPCAO'] = (df_di['DT_RECEPCAO'] - df_di['DT REGISTRO DI'])/pd.Timedelta(hours=1)

In [18]: df_di[['DT REGISTRO DI','DT_RECEPCAO','QTDE HORAS RECEPCAO']][df_di['QTDE HORAS RECEPCAO']>0].head()
```

Out[18]:

	DT REGISTRO DI	DT_RECEPCAO	QTDE HORAS RECEPCAO
46	2019-05-14 12:16:38	2019-05-15 12:09:01	23.873056
53	2019-06-18 11:14:00	2019-06-19 15:38:03	28.400833
104	2019-03-12 18:51:34	2019-03-13 16:26:19	21.579167
122	2019-05-29 17:13:57	2019-05-30 09:57:14	16.721389
306	2019-05-17 13:25:18	2019-05-22 13:16:24	119.851667

Figura 4 - Criação da variável Quantidade de Horas para Recepção Documentos

3.1.5. Dia da Semana

Variável categórica que representa do dia da semana em que a DI foi registrada. Pode afetar o tempo de despacho pois geralmente o direcionamento de declarações para canal de conferência não ocorre em dias não úteis. Afeta, portanto, as declarações dos três canais diferente das variáveis anteriores.

```
In [19]: df_di['DIA SEMANA'] = df_di['DT REGISTRO DI'].dt.weekday

In [20]: dias = {0:'SEGUNDA',1:'TERCA',2:'QUARTA',3:'QUINTA',4:'SEXTA',5:'SABADO',6:'DOMINGO'}
         df_di['DIA SEMANA'] = df_di['DIA SEMANA'].apply(lambda x: dias[x])

In [21]: df_di[['DT REGISTRO DI','DIA SEMANA']].head()
```

Out[21]:

	DT REGISTRO DI	DIA SEMANA
0	2019-06-21 11:28:42	SEXTA
1	2019-07-18 14:49:35	QUINTA
2	2019-06-03 09:22:05	SEGUNDA
3	2019-07-19 12:58:42	SEXTA
4	2019-07-24 10:34:06	QUARTA

Figura 5 - Criação da variável Dia da Semana

3.1.6. Unidade

Variável categórica que representa a Unidade Local da RFB que processou a importação. Obtida pela concatenação do código da unidade (UL PRESENCA) com o nome da unidade (DESCRICAO UL).

```
In [22]: df_di['UNIDADE'] = df_di['UL PRESENCA'].astype(str)+" - "+df_di['DESCRICAO UL']

In [23]: df_di['UNIDADE'].head()

Out[23]: 0                          927800 - ITAJAI
         1                817800 - PORTO DE SANTOS
         2                817800 - PORTO DE SANTOS
         3                817800 - PORTO DE SANTOS
         4    927700 - PORTO DE SAO FRANCISCO DO SUL
         Name: UNIDADE, dtype: object
```

Figura 6 - Criação da variável Unidade

3.1.7. Recinto

Variável categórica que representa o Recinto Aduaneiro responsável pela armazenagem da carga durante o processo de importação. Obtida pela concatenação do código do recinto (RA PRESENCA) com o nome do recinto (DESCRICAO RA).

```
In [17]: df_di['RECINTO'] = df_di['RA PRESENCA'].astype(str)+" - "+df_di['DESCRICAO RA']
         df_di['RECINTO'].head()

Out[17]: 0    9101401 - INST.PORT.MAR.ALF.USO PRIVATIVO MIST...
         1    8931319 - INST.PORT.MAR.ALF-USO PUBL.CIA.BANDE...
         2             8931339 - ECOPORTO SANTOS S.A. (PáTIO 2)
         3    8931319 - INST.PORT.MAR.ALF-USO PUBL.CIA.BANDE...
         4           9981403 - ITAPOÁ TERMINAIS PORTUÁRIOS S/A
         Name: RECINTO, dtype: object
```

Figura 7 - Criação da variável Recinto

3.1.8. Limpeza

Ao final iremos fazer a remoção das variáveis que não serão utilizadas pelos modelos de machine learning.

```
In [19]: df_di.drop(['UL PRESENCA','DESCRICAO UL', 'RA PRESENCA', 'DESCRICAO RA', 'DATA ATRACACAO','DATA PRESENCA', 'REG_LI',
                    'DEFERIMENTO_LI', 'ANUÊNCIA?', 'DT REGISTRO DI', 'DT_SELECAO', 'DT_RECEPCAO', 'DT_DISTRI', 'DT DESEMBARACO DI',
                    'DT ENTREGA', 'ATRAC - ENTR', 'LOG'],axis='columns', inplace=True)
         df_di.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 146937 entries, 0 to 146936
         Data columns (total 14 columns):
          #   Column                      Non-Null Count   Dtype
         ---  ------                      --------------   -----
          0   ID DI                       146937 non-null  int64
          1   TIPO CE                     146937 non-null  object
          2   OEA                         5052 non-null    object
          3   MODALIDADE DESPACHO         146937 non-null  object
          4   TIPO DECLARACAO IMPORTACAO  146937 non-null  object
          5   CANAL                       146937 non-null  object
          6   HORAS_EXIG                  2747 non-null    float64
          7   QTDE HORAS DESPACHO         146937 non-null  float64
          8   QTDE HORAS PRESENCA         146937 non-null  float64
          9   QTDE HORAS DISTRIBUICAO     3813 non-null    float64
          10  QTDE HORAS RECEPCAO         3813 non-null    float64
          11  DIA SEMANA                  146937 non-null  object
          12  UNIDADE                     146937 non-null  object
          13  RECINTO                     146937 non-null  object
         dtypes: float64(5), int64(1), object(8)
         memory usage: 15.7+ MB
```

Figura 8 - Descarte das colunas não utilizadas

3.2. Dados Licenciamento de Importação

O segundo conjunto de dados, que chamaremos de dataset_li, contém informações em nível de Adição de uma DI e, portanto, pode conter mais de uma linha para cada DI.

A Adição representa um conjunto de itens de uma DI que possuem mesma classificação fiscal e tratamento tributário, e, portanto, podem ser agrupadas para facilitar o preenchimento da declaração. A LI deve fazer referência a todos os itens da mesma adição

3.2.1. Orgãos Anuentes

Cada Licença de Importação (LI) pode conter mais de um Órgão Anuente e cada DI pode conter mais de uma LI. Este dataset apresenta até 4 órgãos para uma mesma LI, representados por valores do tipo string contidos nas colunas ANUENTE, ANUENTE2, ANUENTE6 e ANUENTE10. Os valores possíveis são: DECEX, ANVISA, MAPA, INMETRO, ANP, IBAMA, DFPC, DPF, MCT, CNEN, CNPQ e BB.

A intenção foi transformar um atributo categórico de uma LI em um atributo quantitativo de uma DI. Isso foi feito somando a quantidade de vezes que cada órgão anuente aparece em LIs de uma mesma DI.

Foi obtido um novo dataset transformando as colunas supracitadas em variáveis *dummy*, num processo conhecido como *one-hot encoding*. Cada órgão de cada campo anterior terá sua própria coluna e terá um valor numérico binário atribuído de acordo com a presença ou não na LI.

```
In [25]: dum_df = pd.get_dummies(df_li, columns=['ANUENTE','ANUENTE2','ANUENTE6','ANUENTE10'], prefix=['A1','A2','A3','A4'] )
```

Figura 9 - Criação de variáveis dummy

Em seguida serão agrupadas as colunas de cada órgão para cada prefixo anteriormente atribuído somando seus valores para cada DI, e em seguida é adicionada a coluna com o valor correspondente usando como chave o “ID DI”. Desta forma, será replicada em todas as linhas os valores por DI.

```
In [26]: def criaListaColunas(df, str):
             lista = []
             if ('A1_'+str in df.columns):
                 lista.append('A1_'+str)
             if('A2_'+str in df.columns):
                 lista.append('A2_'+str)
             if('A3_'+str in df.columns):
                 lista.append('A3_'+str)
             if('A4_'+str in df.columns):
                 lista.append('A4_'+str)
             return lista

In [27]: orgaos = df_li['ANUENTE'].unique()
         for orgao in orgaos:
             colunas = criaListaColunas(dum_df, orgao)
             dfgroup = dum_df.groupby('ID DI')[colunas].sum()
             serie = dfgroup.sum(axis=1)
             serie.name = 'QTDE LI '+ orgao
             df_li = df_li.join(serie, on='ID DI')
```

Figura 10 - Agrupamento por orgão e por DI

3.2.2. Deferimento LI após registro da DI

Variável quantitativa que representa a quantidade de LI que foram deferidas após o registro da DI. Sua obtenção se deu pela comparação entre as datas e posterior agrupamento por “ID DI”.

```
In [38]: df_li['DATA DEFERIMENTO'] = pd.to_datetime(df_li['DATA DEFERIMENTO'])

In [39]: df_li['LI DEF POS REGISTRO'] = np.where(df_li['DT REGISTRO DI']<df_li['DATA DEFERIMENTO'],1,0)
```

Agrupamos as quantidades de LIs deferidas após o registro por DI

```
In [40]: agrupado = df_li.groupby('ID DI')['LI DEF POS REGISTRO'].sum()
         agrupado.name = 'QTDE LI DEF POS REGISTRO'
         df_li = df_li.join(agrupado, on='ID DI')
```

Figura 11 - Criação da variável Deferimento da LI posterior ao registro da DI

3.2.3. Limpeza

Procede-se a remoção das colunas que não serão utilizadas na construção dos modelos de machine learning e a remoção das linhas duplicadas utilizando como critério o “ID DI”, deixando uma linha por DI.

```
In [36]: df_li.drop(['ID LI','TIPO CONHECIMENTO', 'MODALIDADE DESPACHO',
                     'UL PRESENCA', 'DESCRICAO UL', 'RA PRESENCA',
                     'TIPO DECLARACAO IMPORTACAO', 'DESCRICAO RA', 'CANAL', 'DATA EMISSAO',
                     'DATA ATRACACAO', 'DATA TRANSITO', 'DATA PRESENCA', 'DATA DEFERIMENTO',
                     'DT REGISTRO DI', 'DT DESEMBARACO DI', 'DT ENTREGA', 'DT REG', 'HR REG',
                     'DATA/HORA REGISTRO', 'antes  pres?', 'DT SIT', 'HR SIT', 'SIT', 'QTD',
                     'ANUENTE', 'SITUAÇÃO', 'DATA', 'HORA', 'DATA HORA REG', 'ANUENTE2',
                     'SITUAÇÃO3', 'antes depois 1', 'DATA4', 'HORA5', 'ANUENTE6',
                     'SITUAÇÃO7', 'ANUENTE10', 'SITUAÇÃO11', 'REG-DEF (4)','LI DEF POS REGISTRO'],axis='columns', inplace=True)

In [37]: df_li['QTDE LI DECEX'] = df_li['QTDE LI DECEX'].astype('int64')

In [38]: df_li.drop_duplicates('ID DI',keep='first', inplace=True)
         df_li.info()

         <class 'pandas.core.frame.DataFrame'>
         Int64Index: 63642 entries, 0 to 150488
         Data columns (total 14 columns):
          #   Column                   Non-Null Count  Dtype
         ---  ------                   --------------  -----
          0   ID DI                    63642 non-null  int64
          1   QTDE LI DECEX            63642 non-null  int64
          2   QTDE LI INMETRO          63642 non-null  int64
          3   QTDE LI ANVISA           63642 non-null  int64
          4   QTDE LI IBAMA            63642 non-null  int64
          5   QTDE LI MAPA             63642 non-null  int64
          6   QTDE LI MCT              63642 non-null  int64
          7   QTDE LI ANP              63642 non-null  int64
          8   QTDE LI DPF              63642 non-null  int64
          9   QTDE LI CNEN             63642 non-null  int64
          10  QTDE LI CNPQ             63642 non-null  int64
          11  QTDE LI DFPC             63642 non-null  int64
          12  QTDE LI BB               63642 non-null  int64
          13  QTDE LI DEF POS REGISTRO 63642 non-null  int32
         dtypes: int32(1), int64(13)
         memory usage: 7.0 MB
```

Figura 12 - Limpeza de dados

3.3. Dados Trânsito Aduaneiro

O terceiro conjunto de dados, que chamaremos de dataset_tran, contém informações em nível de DI e possui apenas uma variável de interesse.

3.3.1. Trânsito Aduaneiro

Variável categórica binária que indica se houve operação de trânsito aduaneiro no processo de importação. Será obtida pela conversão da variável Data de Início do Trânsito Aduaneiro (DATA TRANSITO).

```
In [41]: df_tran['TRANSITO'] = np.where(~df_tran['DATA TRANSITO'].str.contains('<N/D>',na=False),1,0)
         df_tran['TRANSITO'].describe()

Out[41]: count    184742.000000
         mean          0.103344
         std           0.304409
         min           0.000000
         25%           0.000000
         50%           0.000000
         75%           0.000000
         max           1.000000
         Name: TRANSITO, dtype: float64
```

Figura 13 - Criação da Variável Trânsito Aduaneiro

3.3.2. Limpeza

Procede-se a remoção das colunas que não serão utilizadas na construção dos modelos de machine learning.

```
In [45]: df_tran.drop(['TIPO CONHECIMENTO', 'MODALIDADE DESPACHO', 'UL PRESENCA',
                       'DESCRICAO UL', 'RA PRESENCA', 'TIPO DECLARACAO IMPORTACAO',
                       'DESCRICAO RA', 'CANAL', 'DATA ATRACACAO', 'DATA TRANSITO',
                       'DATA PRESENCA', 'DATA DEFERIMENTO', 'ANUÊNCIA', 'ANTES DEPOIS',
                       'DATA EMISSAO', 'DT REGISTRO DI', 'ANTES DEPOIS2', 'DT DESEMBARACO DI',
                       'DT ENTREGA'], axis='columns', inplace=True)
```

Figura 14 - Limpeza de dados

3.4. União dos Datasets

Os datasets descritos foram unidos utilizando como critério o identificador único de cada DI.

```
In [48]: df = pd.merge(df_di,df_li,how='left',left_on='ID DI',right_on='ID DI')
         df = pd.merge(df,df_tran,how='left',left_on='ID DI',right_on='ID DI')
         df.head(10)
```

Figura 15 - União dos 3 datasets

Ao final tem-se o dataset final com as 27 variáveis a seguir:

Coluna	Nome	Contagem Valores Não Nulos	Tipo
0	ID DI	146.937	Inteiro
1	TIPO CE	146.937	Texto
2	OEA	5.052	Texto
3	MODALIDADE DESPACHO	146.937	Texto
4	TIPO DECLARACAO IMPORTACAO	146.937	Texto
5	CANAL	146.937	Texto
6	HORAS_EXIG	2.747	Decimal
7	QTDE HORAS DESPACHO	146.937	Decimal
8	QTDE HORAS PRESENCA	146.937	Decimal
9	QTDE HORAS DISTRIBUICAO	3.813	Decimal
10	QTDE HORAS RECEPCAO	3.813	Decimal

11	DIA SEMANA	146.937	Texto
12	UNIDADE	146.937	Texto
13	RECINTO	146.937	Texto
14	QTDE LI DECEX	59.663	Decimal
15	QTDE LI INMETRO	59.663	Decimal
16	QTDE LI ANVISA	59.663	Decimal
17	QTDE LI IBAMA	59.663	Decimal
18	QTDE LI MAPA	59.663	Decimal
19	QTDE LI MCT	59.663	Decimal
20	QTDE LI ANP	59.663	Decimal
21	QTDE LI DPF	59.663	Decimal
22	QTDE LI CNEN	59.663	Decimal
23	QTDE LI CNPQ	59.663	Decimal
24	QTDE LI DFPC	59.663	Decimal
25	QTDE LI BB	59.663	Decimal
26	QTDE LI DEF POS REGISTRO	59.663	Decimal
27	TRANSITO	146.937	Inteiro

3.5. Tratamento de Dados Ausentes

Como visto na tabela acima, existem diversas colunas com valores nulos que precisam ser tratados para uma melhor performance dos modelos de machine learning. Abaixo temos a representação gráfica dos valores ausentes.

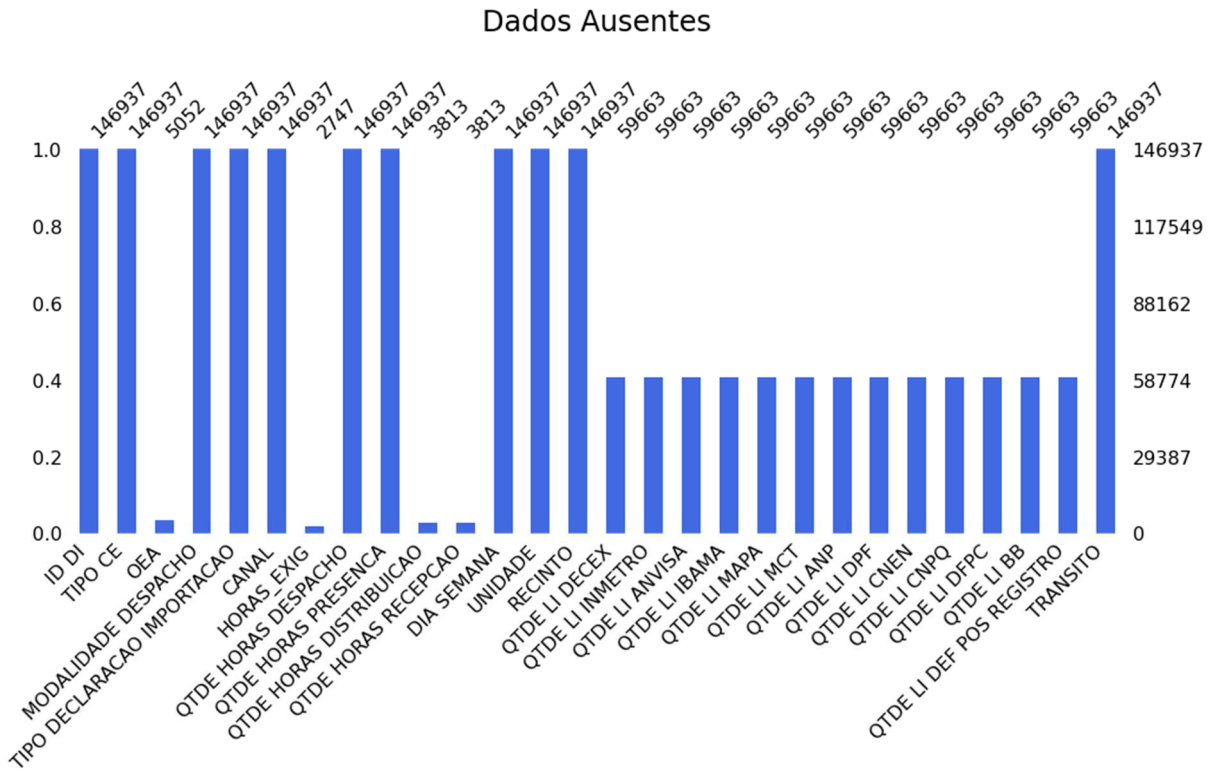


Figura 16- Representação gráfica dos dados ausentes

3.5.1. OEA

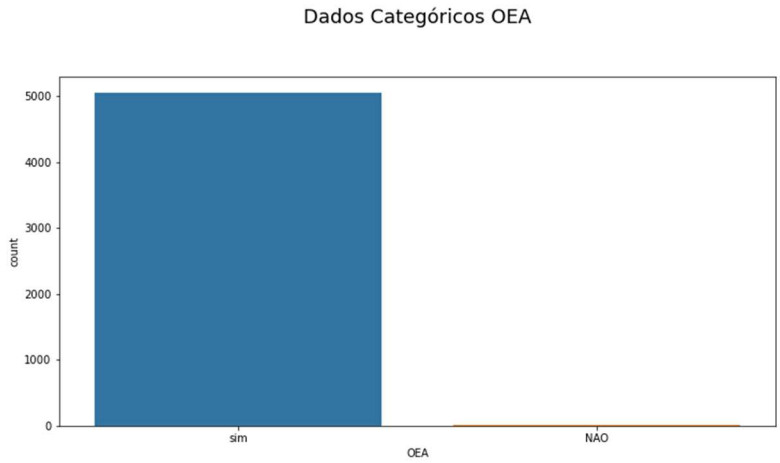


Figura 17 - Distribuição OEA inicial

Nota-se que foram poucos os valores “NÃO” encontrados no domínio. Ocorre que as empresas certificadas OEA são a exceção e não a regra nos processos de importação, de forma que é seguro assumirmos valores “negativos” para os dados ausentes. Faremos a imputação de valores “NÃO”, portanto, aos dados ausentes.

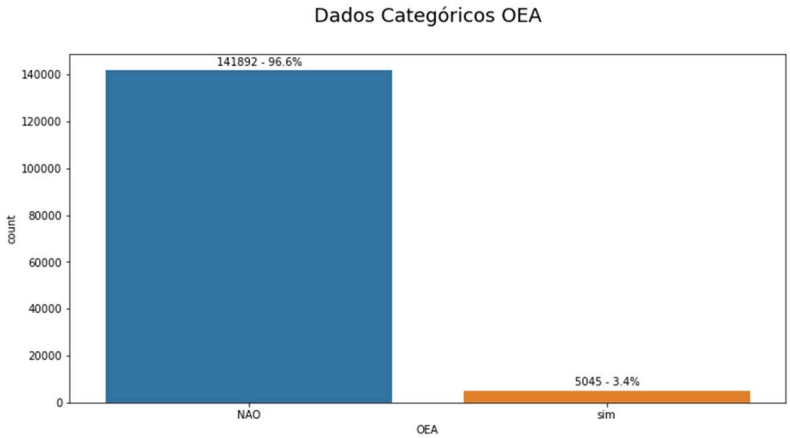


Figura 18 -Distribuição OEA após tratamento de dados ausentes

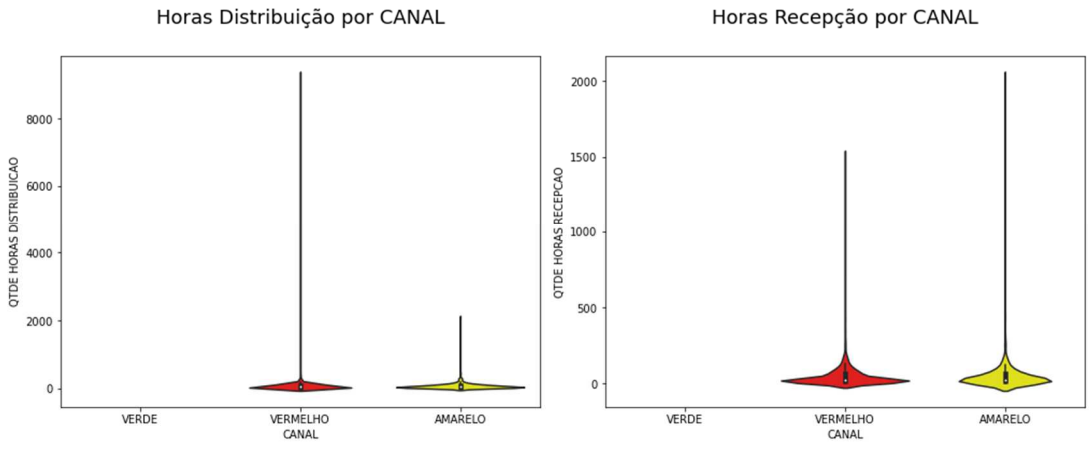
3.5.2. Quantidades oriundas do dataset LI

Nem todas as DIs estão sujeitas a licenciamento, de forma que a quantidade dessas declarações provenientes do dataset LI é substancialmente menor do que do dataset das declarações de importação. O dataset LI contém colunas com quantidades de LI para cada órgão anuente e a quantidade de LIs emitidas posteriormente ao registro da DI.

Portanto, para todas as colunas provenientes daquele dataset imputaremos o valor 0 (zero) aos dados ausentes.

3.5.3. Quantidade de Horas Distribuição e Quantidade de Horas Recepção

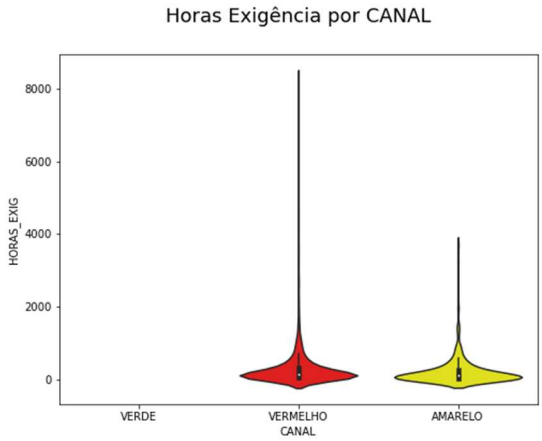
Apenas declarações direcionadas para canal de conferência são distribuídas e têm a recepção de seus documentos.



Portanto as demais declarações receberão o valor 0 aos dados ausentes.

3.5.4. Horas Exigência

Novamente apenas declarações direcionadas para canal de conferência estão sujeitas a alguma exigência fiscal, porém nem todas serão submetidas. Isso explica a quantidade de registros inferior ao item anterior.



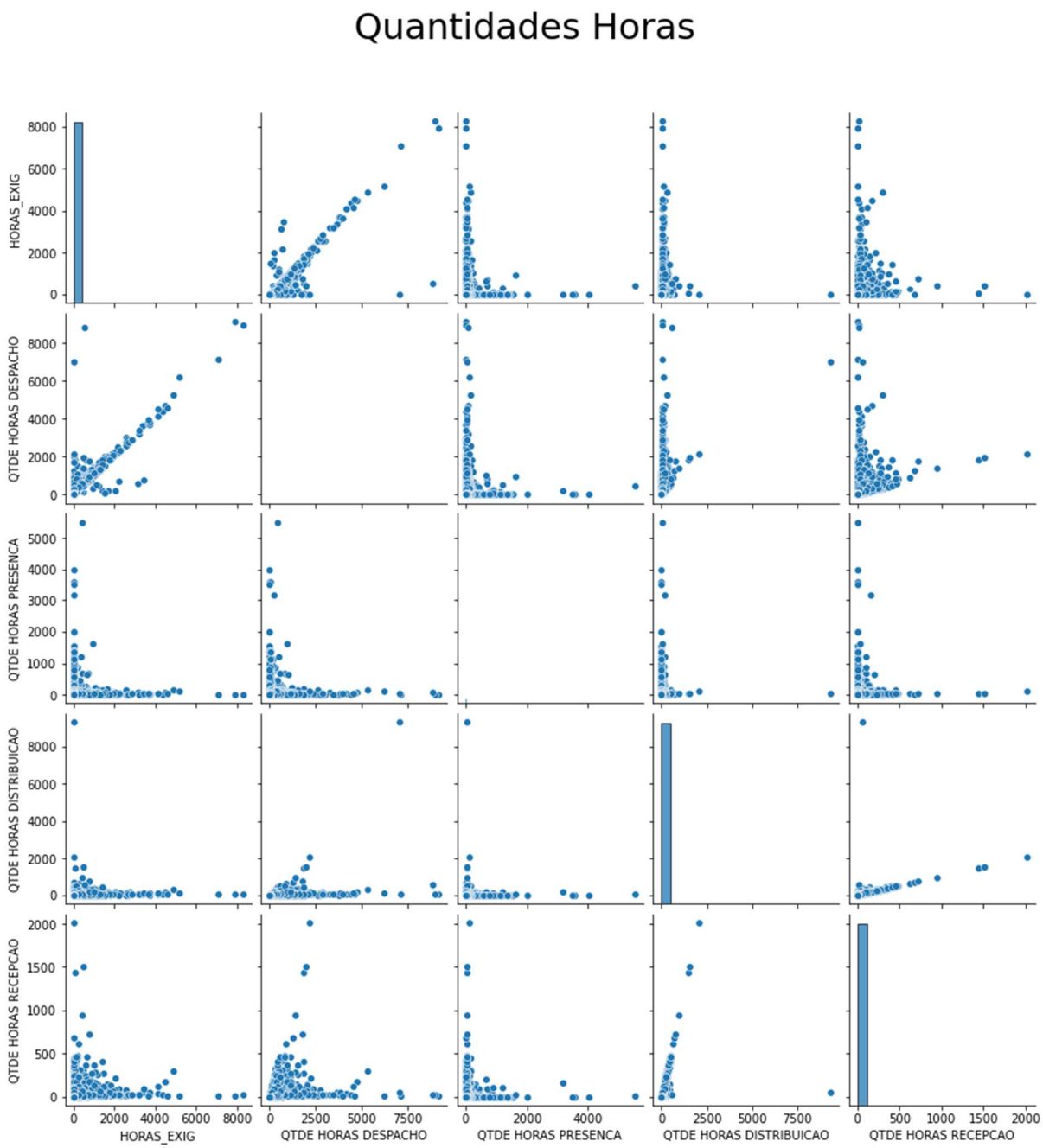
Para os valores ausentes será imputado o valor 0.

Concluída esta etapa, agora temos o dataset final, que será utilizado para a construção dos modelos, contendo 146.937 linhas e 28 colunas.

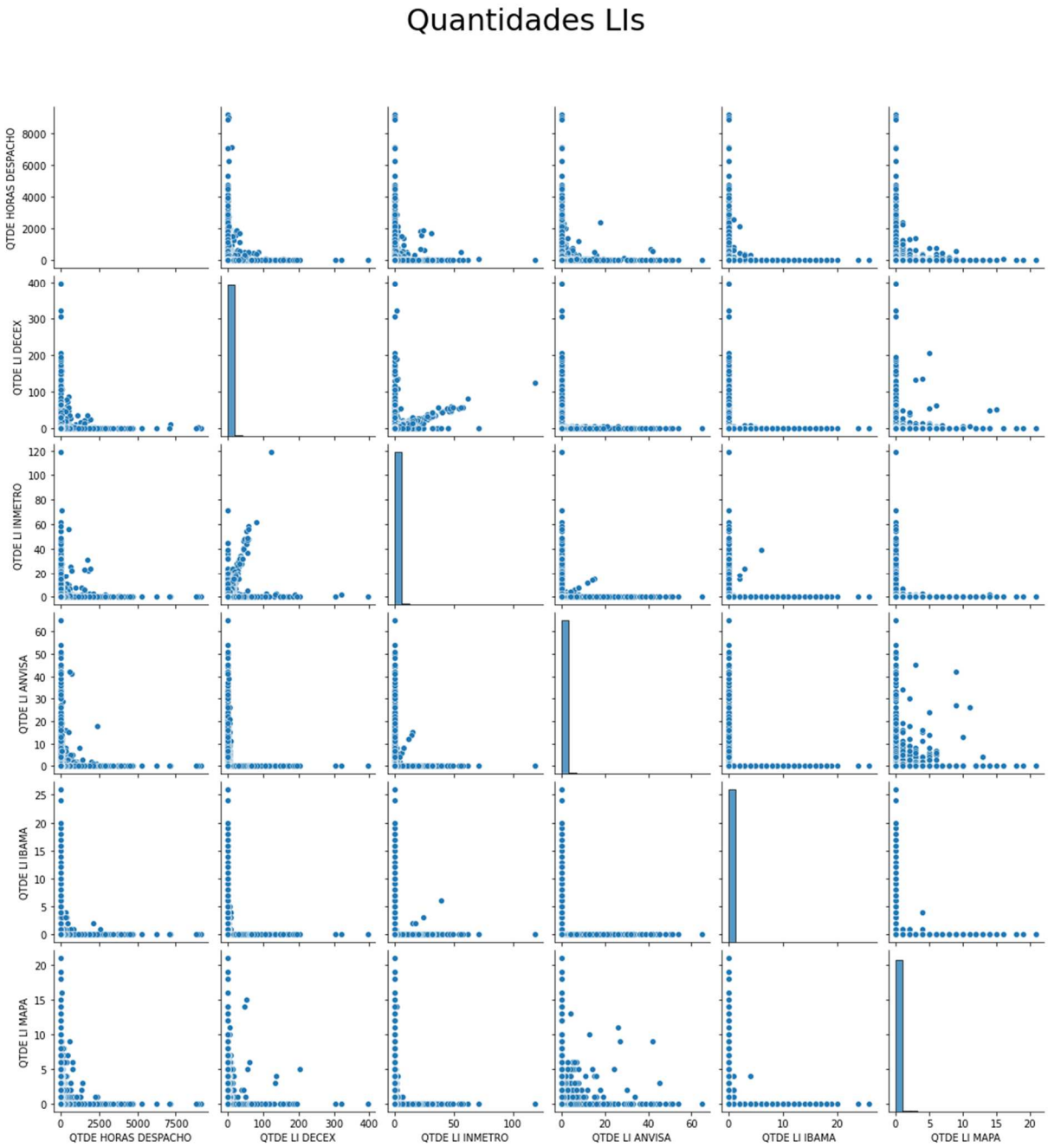
4. Análise e Exploração dos Dados

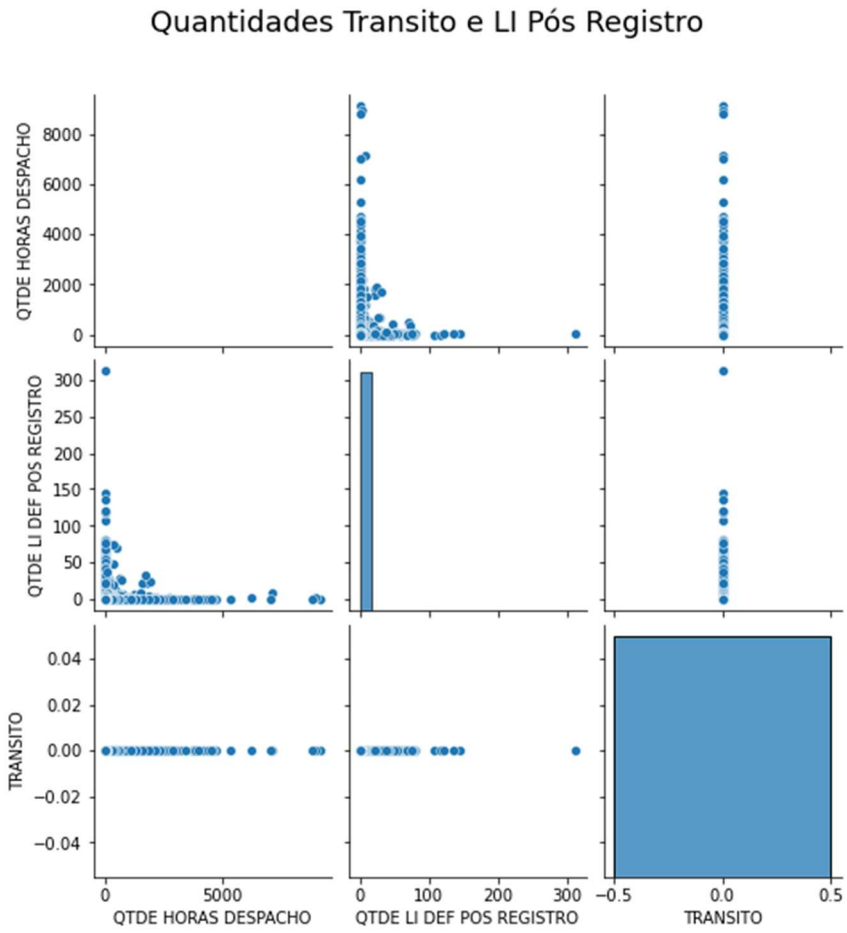
4.1. Variáveis Quantitativas

A análise exploratória dos dados será iniciada com a plotagem das distribuições das variáveis quantitativas, buscando a visualização de correlação entre as variáveis, sobretudo sobre a variável alvo, bem como uma inspeção visual dos dados obtidos.



A correlação mais clara neste conjunto está entre as horas de exigência e a quantidade de horas para o despacho. As demais não apresentam uma clara correlação.

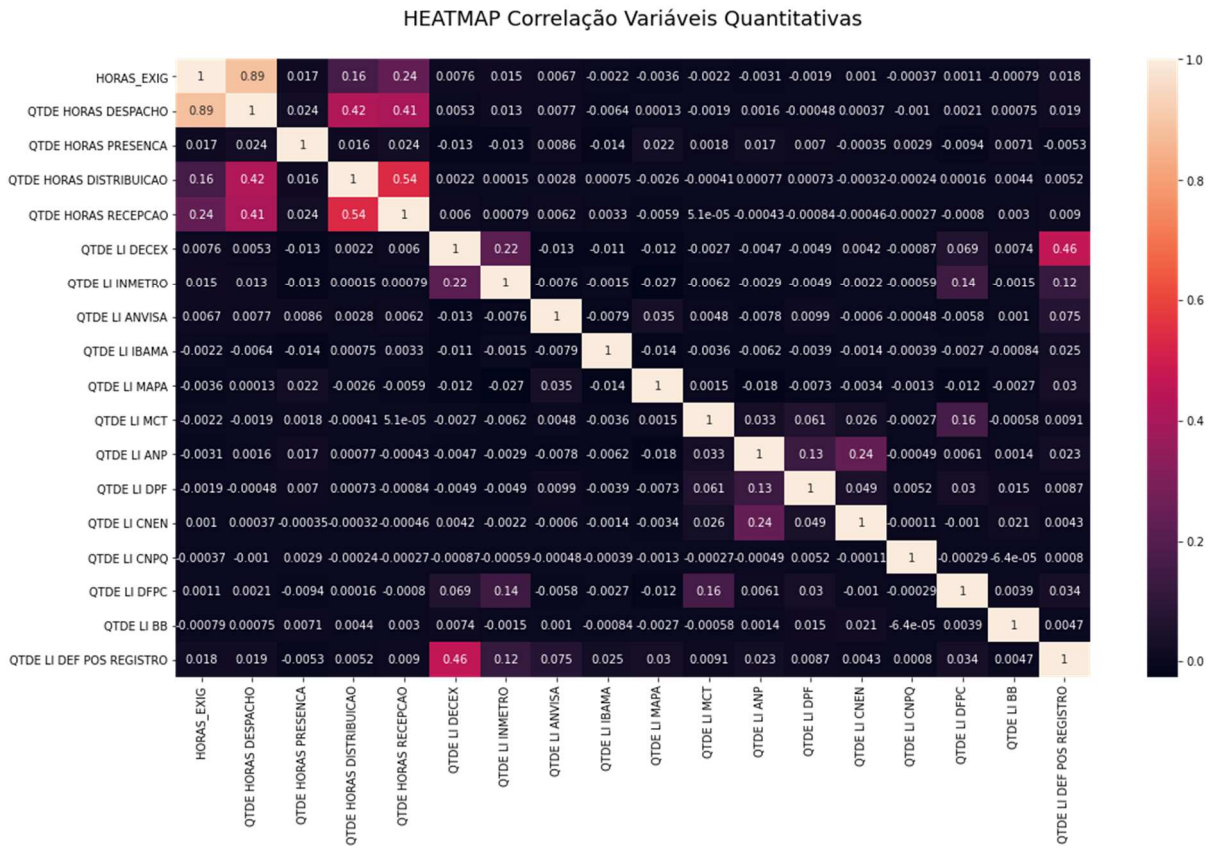




No último conjunto é possível detectar que a variável TRANSITO possui valor constante igual a 0, demonstrando que as declarações submetidas a trânsito aduaneiro foram descartadas do estudo da RFB.

Coluna1	HORAS_EXIG	QTDE HORAS DESPACHO	QTDE HORAS PRESENCA	QTDE HORAS DISTRIBUICAO	QTDE HORAS RECEPCAO	QTDE LI DECEX	QTDE LI INMETRO	QTDE LI ANVI	QTDE LI IBAM	QTDE LI MAPA	QTDE LI MCT	QTDE LI ANP	QTDE LI DPF	QTDE LI CNEN	QTDE LI CNPC	QTDE LI DFPC	QTDE LI BB	QTDE LI DEF POS REGISTRO
count	146937,0000000	146937,0000000	146937,0000000	146937,0000000	146937,0000000	146937,0000000	146937,0000000	146937,0000000	146937,0000000	146937,0000000	146937,0000000	146937,0000000	146937,0000000	146937,0000000	146937,0000000	146937,0000000	146937,0000000	146937,0000000
mean	5,4517580	22,4676420	32,0672820	1,7890400	1,1823310	0,7231870	0,1455730	0,0952110	0,0210160	0,0988720	0,0040090	0,0240440	0,0045120	0,0005990	0,0003270	0,0077310	0,0002380	0,2037540
std	80,4977330	98,9758240	45,8094540	29,3219120	13,9206870	4,0432380	1,1643780	0,8581710	0,2934970	0,4328170	0,0805190	0,2717890	0,0829900	0,0313000	0,0592590	0,1468630	0,0203740	1,9076570
min	0,0000000	0,0000000	0,0000000	0,0000000	-0,0002780	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000
25%	0,0000000	4,7494440	12,4500000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000
50%	0,0000000	8,0755560	24,0500000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000
75%	0,0000000	19,7919440	43,6833330	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000
max	8276,0000000	9164,2441670	5516,5500000	9315,1391670	2015,2622220	397,0000000	119,0000000	65,0000000	26,0000000	21,0000000	10,0000000	22,0000000	11,0000000	4,0000000	17,0000000	20,0000000	5,0000000	312,0000000

Figura 19 - Características estatísticas das variáveis quantitativas



Há pouca correlação entre as variáveis quantitativas, sendo a maior entre as variáveis relativas a quantidades de horas, sobretudo entre HORAS_EXIG e QTDE HORAS DESPACHO, porém não chega a ser tão alta a ponto de ser necessário o descarte.

Nota-se haver uma correlação baixa de algumas variáveis com o alvo, aqui considerada abaixo de 0,01, sobretudo nas quantidades de LIs extraídas, que poderão ser agrupadas em uma etapa futura.

Outro dado interessante é a correlação de aproximadamente 0,5 entre a quantidade de LIs emitidas pela DECEX e a quantidade de LIs obtidas posteriormente ao registro da DI.

4.2. Variáveis Categóricas

4.2.1. Cardinalidade

Iniciaremos a análise das variáveis categóricas aferindo a cardinalidade.

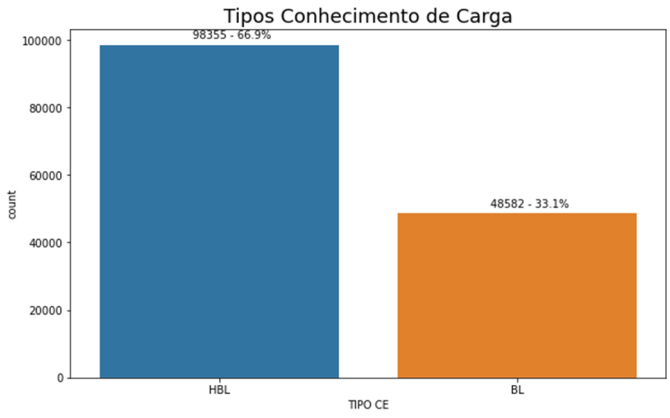
Nome Variável	Quantidade de Valores Únicos
TIPO CE	2
OEA	2
MODALIDADE DESPACHO	1
TIPO DECLARACAO IMPORTACAO	1
CANAL	3
DIA SEMANA	7
UNIDADE	30
RECINTO	123
TRANSITO	1

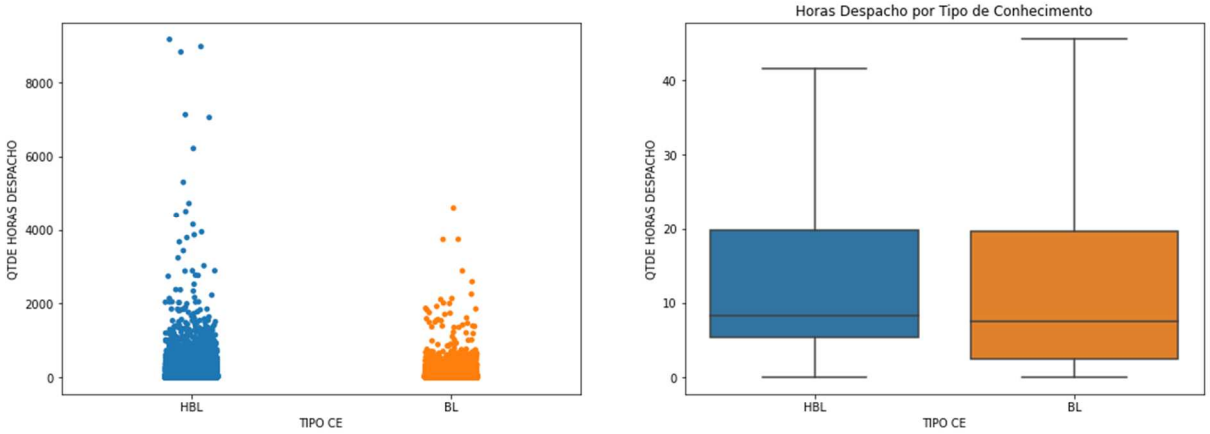
Verifica-se que há 3 variáveis com valores únicos em todo o conjuntos de dados. São elas MODALIDADE DESPACHO, TIPO DECLARACAO IMPORTACAO e TRANSITO.

Tendo em vista que não podem contribuir com a criação de modelos de machine learning, iremos descartar as três variáveis.

4.2.2. Tipo CE

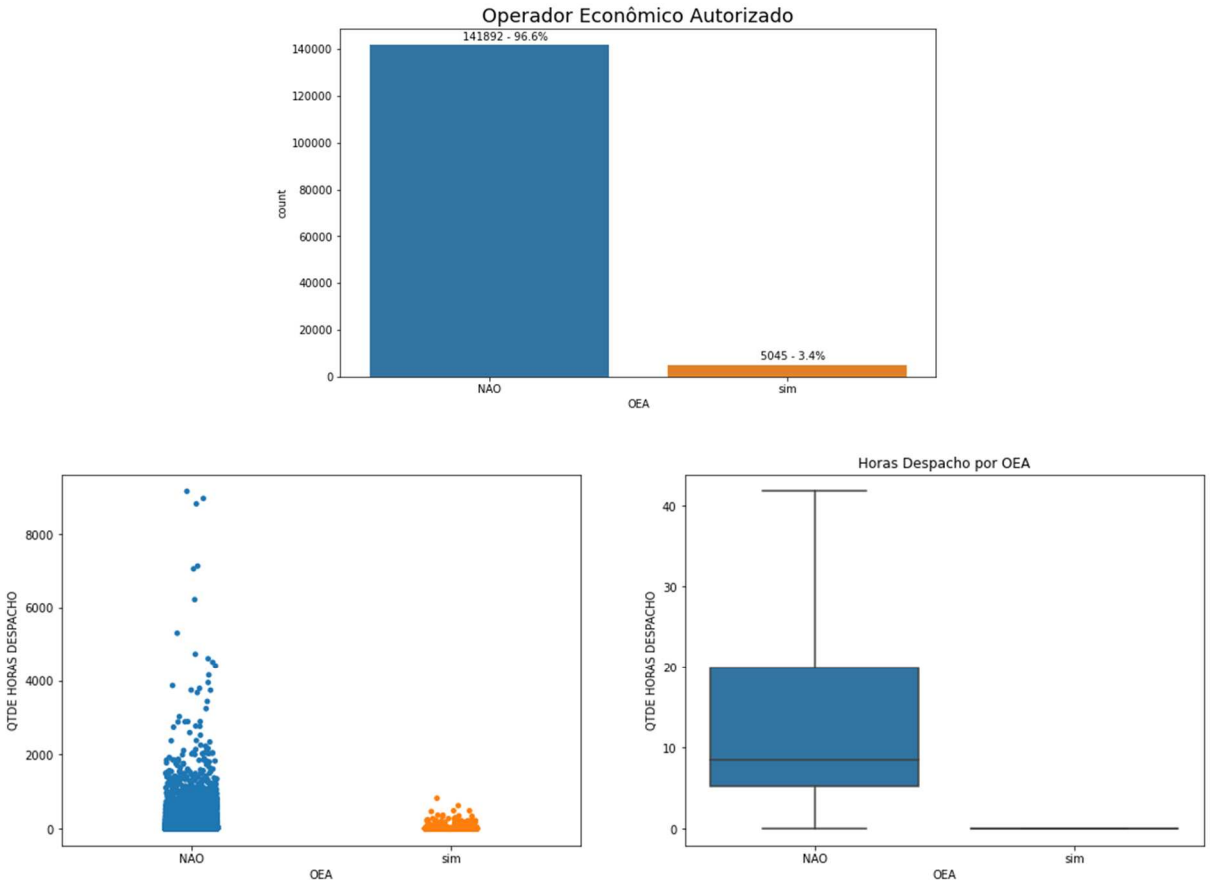
O TIPO CE é uma variável que indica o se o Conhecimento de embarque é do tipo BL (Bill of Lading), de emissão do armador ou do tipo HBL (House Bill of Lading) emitido por um agente de cargas.





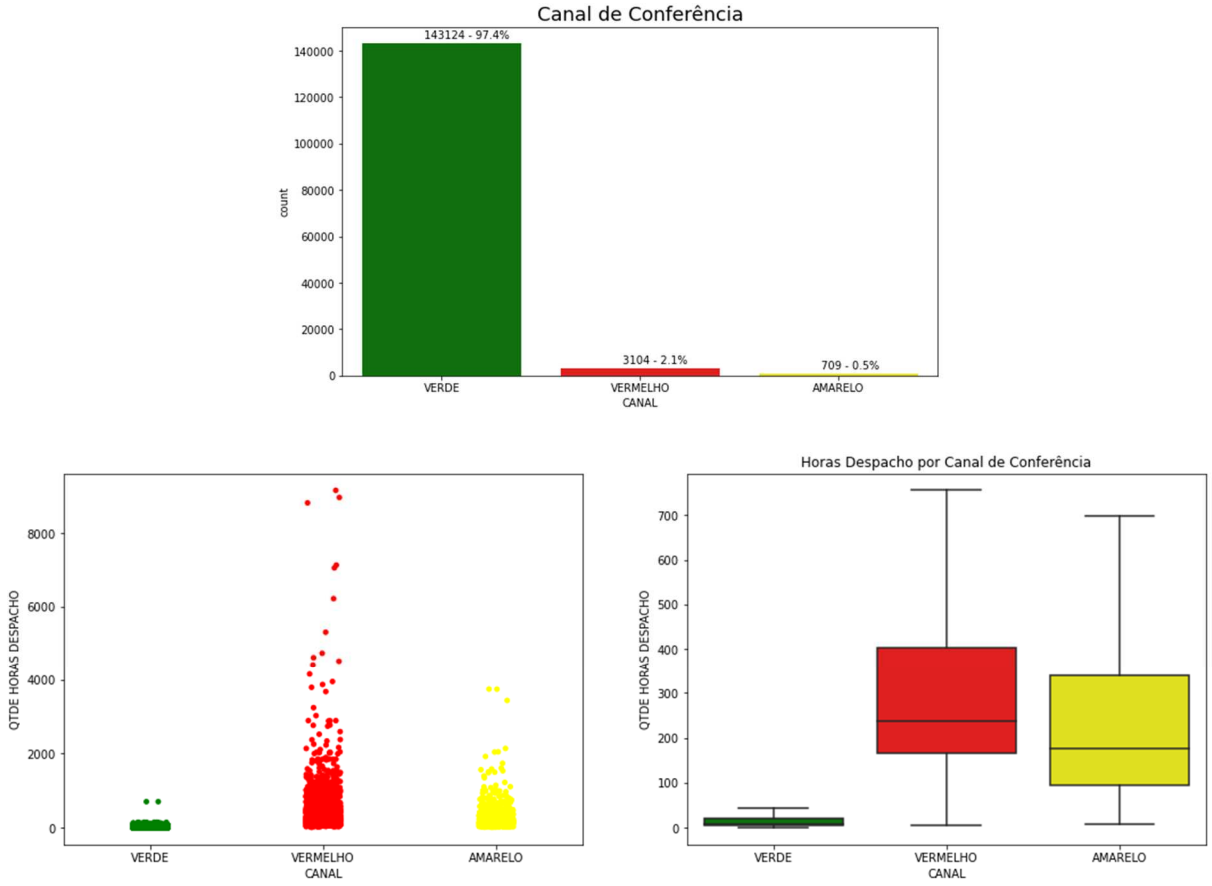
Não se verifica uma diferença significativa nas horas de despacho para a variação de tipo de conhecimento.

4.2.3. OEA



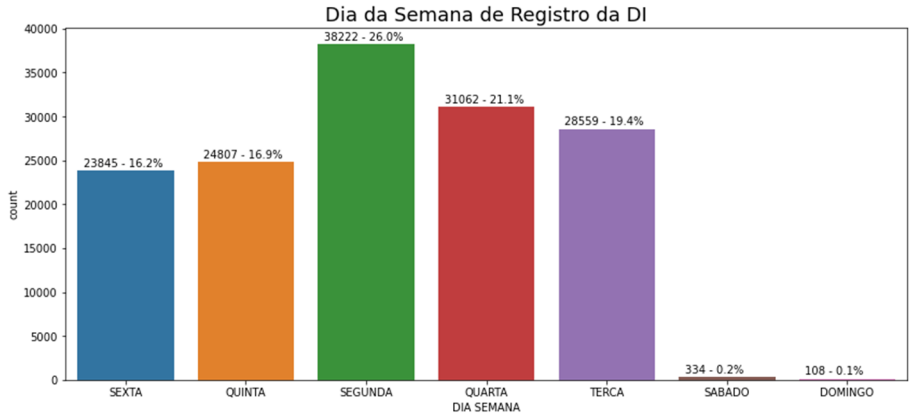
Para empresas certificadas OEA, a quantidade de horas dispendidas no despacho aduaneiro é significativamente melhor e deve contribuir para a formação do modelo.

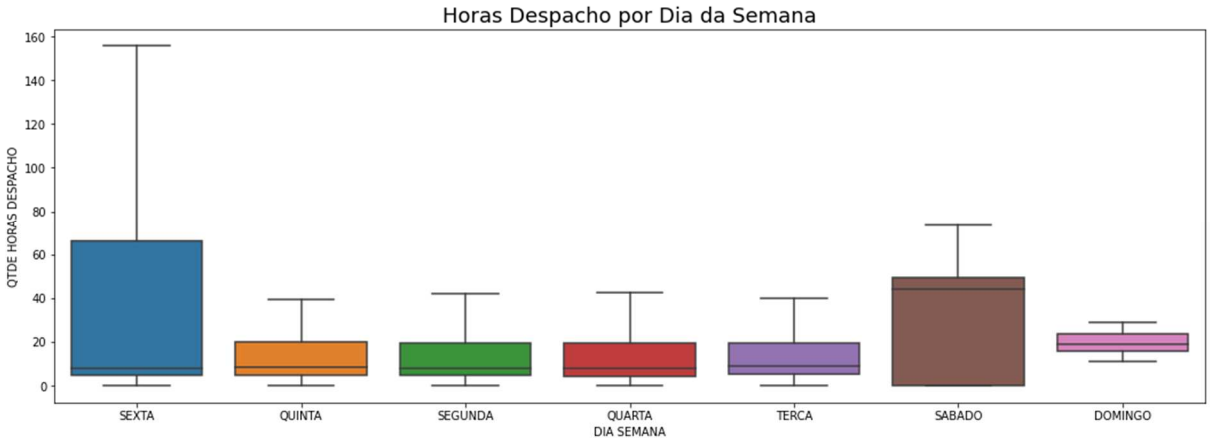
4.2.4. Canal



O canal de conferência aduaneira também exerce um papel importante em relação à variável alvo, porém menos de 3% das declarações são submetidas a canal amarelo ou vermelho.

4.2.5. Dia da Semana





Para a variável dia da semana parece haver uma grande semelhança na distribuição das horas durante a semana e uma boa diferença para a sexta-feira e sábado, em que há uma maior dispersão. Será particularmente útil para as DIs direcionadas a canal verde.

5. Criação de Modelos de Machine Learning

Tratando-se de uma tarefa de previsão de uma variável contínua, e considerando que o conjunto de dados contém rótulos, utilizaremos técnicas de regressão, que são modelos de aprendizagem de máquina supervisionados.

Para a utilização de modelos da biblioteca scikit learn, é necessário a conversão de variáveis categóricas em numéricas.

Para as variáveis de baixa cardinalidade utilizaremos a conversão One-Hot, utilizando recurso da biblioteca pandas.

```
In [112]: dum_df = pd.get_dummies(df, columns=['TIPO CE','OEA','CANAL','DIA SEMANA'], prefix=['TIPO CE','OEA','CANAL','DIA SEMANA'] )
```

Já para as variáveis de alta cardinalidade, usaremos a técnica de Label Encoding para não resultarmos em muitas colunas, que poderia acarretar maior custo computacional.

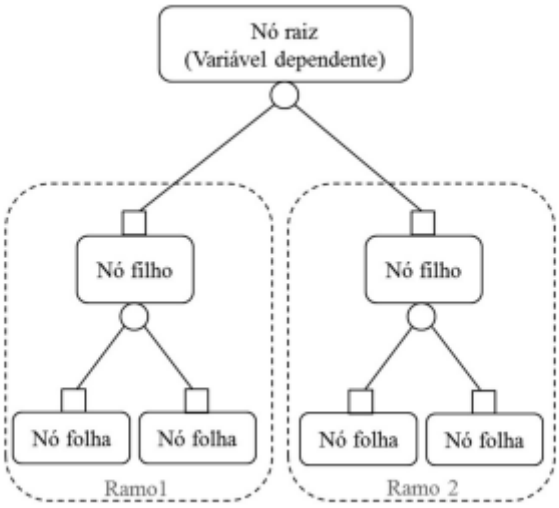
```
In [113]: from sklearn.preprocessing import LabelEncoder
          le = LabelEncoder()
          dum_df['UNIDADE_ENC'] = le.fit_transform(dum_df['UNIDADE'])
          dum_df['RECINTO_ENC'] = le.fit_transform(dum_df['RECINTO'])
```

Por fim separaremos o conjunto de dados resultante em um conjunto de treino e outro de teste, usando 70% para o conjunto de treino.

5.1. Decision Tree Regressor

A árvore de Decisão é um tipo de algoritmo de aprendizagem de máquina supervisionado que se baseia na ideia de divisão dos dados em grupos homogêneos. Uma árvore de decisão é chamada de Árvore de Classificação se a variável resposta for categórica, ou Árvore de Regressão, se numérica (TACONELI, 2008)

O processo de indução de árvores é iniciado por meio de uma amostra, denominada nó raiz, que é dividida em subamostras, denominadas nós filhos ou nós intermediários. Essas subamostras quando subdivididas são chamadas de nós pais, pois geram nós filhos. Quando uma subamostra não puder mais ser subdivida segundo algum critério de parada, é então denominada de nó final ou nó folha. Esse processo é dito recursivo devido a cada subamostra gerar novas subamostras.



Foi utilizada a implementação padrão do pacote scikit learn e a avaliação do modelo usando o coeficiente de determinação, ou R^2.

```
In [117]: from sklearn.tree import DecisionTreeRegressor

In [118]: dtr = DecisionTreeRegressor(random_state=42)
          dtr.fit(x_train,y_train)

Out[118]: DecisionTreeRegressor(random_state=42)

In [119]: dtr.score(x_test,y_test)

Out[119]: 0.5802525489150927
```

A técnica de Grid Search, ou busca em grade, foi utilizada para o aprimoramento dos hiperparâmetros do modelo, testando alguns valores para a profundidade máxima da árvore e a quantidade mínima de amostras para a separação, chegando aos valores de 8 e 3 respectivamente.

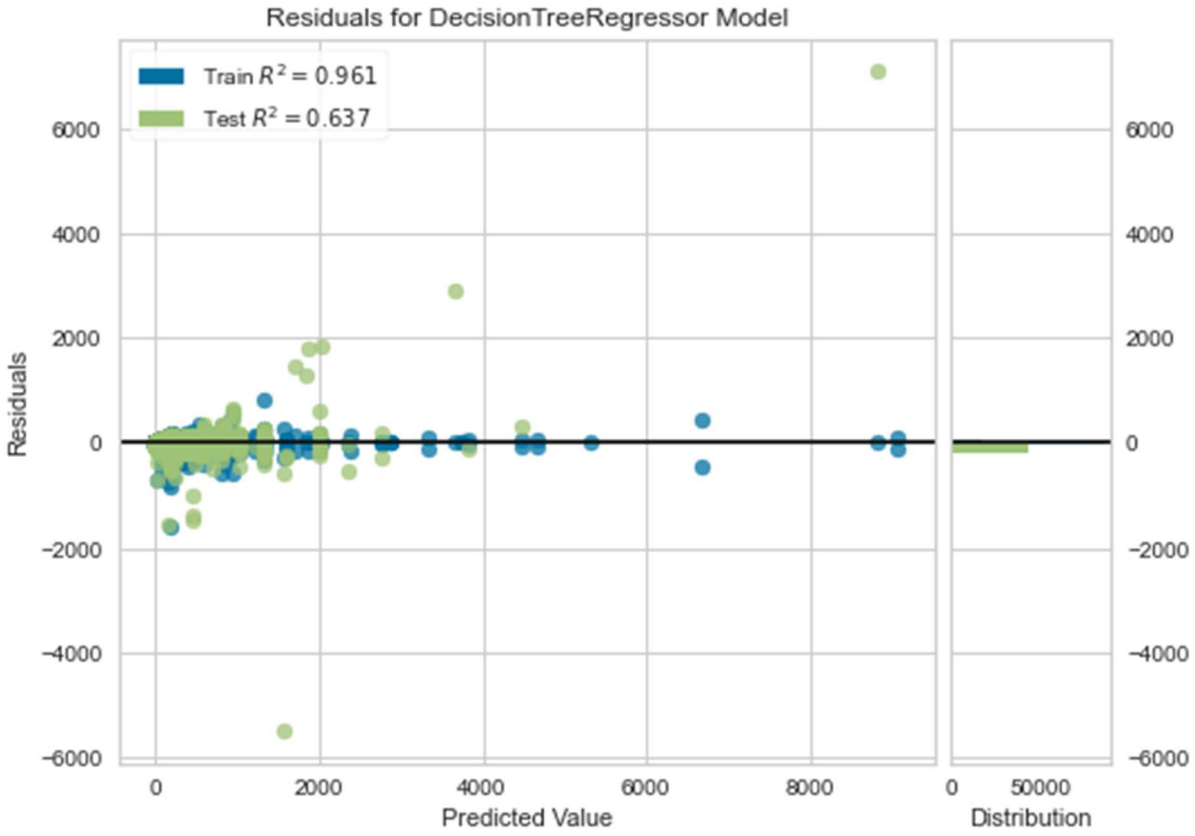
```
In [120]: dtr1 = DecisionTreeRegressor(random_state=42)

In [121]: parametros_dtr={'max_depth':[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15],
                          'min_samples_split':[2,3,4,5,6,7,8,9,10],
                         }

In [122]: grid_dtr = GridSearchCV(estimator=dtr1, param_grid=parametros_dtr,cv=10,n_jobs=-1)

In [123]: grid_dtr.fit(x_train,y_train)

Out[123]: GridSearchCV(cv=10, estimator=DecisionTreeRegressor(random_state=42), n_jobs=-1,
                       param_grid={'max_depth': [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
                                                 13, 14, 15],
                                   'min_samples_split': [2, 3, 4, 5, 6, 7, 8, 9, 10]})

In [124]: grid_dtr.best_params_

Out[124]: {'max_depth': 8, 'min_samples_split': 3}
```

Um modelo final foi criado utilizando os hiperparâmetros encontrados, e resultando em um novo coeficiente de determinação.

```
In [126]: dtr_final = DecisionTreeRegressor(random_state=42,max_depth=8, min_samples_split=3)
          dtr_final.fit(x_train,y_train)
          dtr_final.score(x_test,y_test)

Out[126]: 0.6372698394284206
```

O ajuste efetuado foi suficiente para um ganho significativo no coeficiente de determinação que passou de 0.58 para 0.63, enquanto o erro absoluto médio chegou a 11,73.

Abaixo o gráfico que representa os resíduos:



5.2. RandomForest Regressor

Para entender o algoritmo RandomForest, precisamos primeiramente conhecer os métodos ensemble, dos quais ele faz parte. Estes métodos são construídos da mesma forma que algoritmos mais básicos, como regressão linear, árvore de decisão ou KNN, por exemplo, mas possuem uma característica principal que os diferenciam, a combinação de diferentes modelos para se obter um único resultado. Essa

característica torna esses algoritmos mais robustos e complexos, levando a um maior custo computacional que costuma ser acompanhando de melhores resultados.

Normalmente na criação de um modelo, escolhemos o algoritmo que apresenta o melhor desempenho para os dados em questão. Podemos testar diferentes configurações deste algoritmo escolhido, gerando assim diferentes modelos, mas no fim do processo de machine learning, escolhemos apenas um. Com um método ensemble serão criados vários modelos diferentes a partir de um algoritmo, mas não escolheremos apenas um para utilização final, e sim todos.

No algoritmo RandomForest serão criadas várias árvores de decisão distintas, pois tanto na seleção das amostras, quanto na seleção das variáveis, o processo acontece de maneira aleatória.

A “floresta” que ele cria é uma combinação (ensemble) de árvores de decisão, na maioria dos casos treinados com o método de bagging. A ideia principal do método de bagging é que a combinação dos modelos de aprendizado aumenta o resultado geral.

```
In [18]: from sklearn.ensemble import RandomForestRegressor

In [19]: rfr = RandomForestRegressor(random_state=42, n_estimators=1000)
         rfr.fit(x_train,y_train)

Out[19]: RandomForestRegressor(n_estimators=1000, random_state=42)

In [20]: rfr.score(x_test,y_test)

Out[20]: 0.7511521928533736
```

Novamente a técnica de Grid Search para o aprimoramento dos hiperparâmetros do modelo foi utilizada, culminando em valores para a profundidade máxima da árvore de 15, quantidade mínima de amostras para a separação de 10, quantidade mínima de amostras para folhas de 2, e o parâmetro bootstrap para verdadeiro.

```
In [53]: rfr1 = RandomForestRegressor(random_state=42, n_estimators=100)

In [54]: parametros_rfr = {
                    'max_depth':[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15],
                    'min_samples_split':[2,3,4,5,6,7,8,9,10],
                    'min_samples_leaf':[1,2,4,6,8,10],
                    'bootstrap': [False, True]
                   }

In [55]: grid_rfr = GridSearchCV(estimator=rfr1, param_grid=parametros_rfr,cv=10,n_jobs=-1)

In [56]: grid_rfr.fit(x_test,y_test)

Out[56]: GridSearchCV(cv=10, estimator=RandomForestRegressor(random_state=42), n_jobs=-1,
             param_grid={'bootstrap': [False, True],
                         'max_depth': [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
                                       13, 14, 15],
                         'min_samples_leaf': [1, 2, 4, 6, 8, 10],
                         'min_samples_split': [2, 3, 4, 5, 6, 7, 8, 9, 10]})

In [57]: grid_rfr.best_params_

Out[57]: {'bootstrap': True,
          'max_depth': 15,
          'min_samples_leaf': 2,
          'min_samples_split': 10}
```

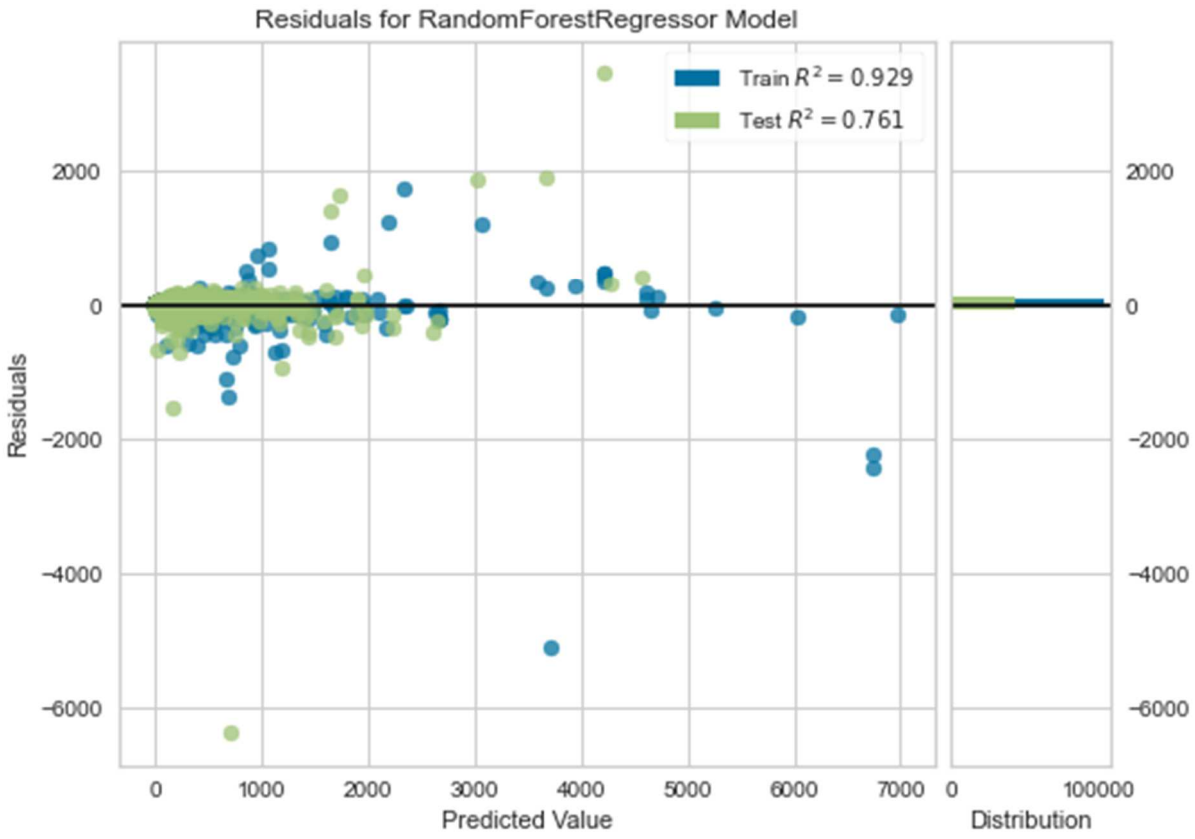
O modelo ideal então foi criado com os hiperparâmetros encontrados e observou-se uma melhora no coeficiente de determinação.

```
In [21]: rfr_final = RandomForestRegressor(random_state=42, n_estimators=1000, bootstrap=True, max_depth=15,min_samples_leaf=2,min_samples
         rfr_final.fit(x_train,y_train)
         rfr_final.score(x_test,y_test)

Out[21]: 0.7610136332844397
```

Com o modelo final foi possível obter um coeficiente de determinação de 0.76, enquanto o erro absoluto médio foi de 10.76.

```
In [23]: metrics.r2_score(y_test,pred_rfr_final)

Out[23]: 0.7610136332844397

In [24]: metrics.mean_absolute_error(y_test,pred_rfr_final)

Out[24]: 10.762704448186174
```

Nota-se que em comparação com o modelo de Árvore de Decisão, há uma substancial melhora na performance, que pode ser observada também no gráfico de resíduos.



5.3. CatBoost Regressor

O GradienBoost é uma técnica de aprendizado de máquina poderosa que atinge resultados de ponta em uma variedade de tarefas práticas. Por vários anos, permaneceu como o principal método para problemas de aprendizagem com características heterogêneas, dados ruidosos e dependências complexas: pesquisa na web, sistemas de recomendação, previsão do tempo e muitos outros [2, 15, 17, 18]. É apoiado por fortes resultados teóricos que explicam como preditores fortes podem ser construídos pela combinação iterativa de modelos mais fracos (preditores de base) por meio de um procedimento ganancioso que corresponde à descida do gradiente em um espaço de função.

CatBoost usa uma estratégia eficiente que reduz o overfitting e permite usar todo o conjunto de dados para treinamento. Realiza uma permutação aleatória do conjunto de dados e para cada exemplo é calculado o valor médio do rótulo para o exemplo com o mesmo valor de categoria colocado antes do dado na permutação.

Seja σ = (σ1, ..., σ n) a permutação, então $x_{\sigma_{p,k}}$ é substituído por $\frac{\sum_{j=1}^{p-1}\left[x_{\sigma_{j,k}}=x_{\sigma_{p,k}}\right]Y_{\sigma_j}+a\cdot P}{\sum_{j=1}^{p-1}\left[x_{\sigma_{j,k}}=x_{\sigma_{p,k}}\right]+a}$ onde também adicionamos um valor anterior P e um parâmetro a> 0, que é o peso do anterior. Adicionar a priori é uma prática comum e ajuda a reduzir o ruído obtido nas categorias de baixa frequência [3]. Para tarefas de regressão, a técnica padrão para cálculo prévio é pegar o valor médio do rótulo no conjunto de dados.

O CatBoost é um algoritmo recente, open source, desenvolvido pela empresa russa Yandex. Produz resultados mais rápidos e precisos do que outros algoritmos que utilizam a mesma técnica de Gradient Bosting como LightGBM, XGBoost e H2O, pode usar GPU para processamento e possui suporte para variáveis categóricas, não necessitando do encoding.

Portanto, para este caso foi realizada uma nova separação do conjunto de dados, com as variáveis categóricas em seu estado original.

```
In [28]: cbr_X = df.drop(['QTDE HORAS DESPACHO','ID DI'], axis=1)
         cbr_y = df['QTDE HORAS DESPACHO']
         cbr_X_train, cbr_X_validation, cbr_y_train, cbr_y_validation = train_test_split(cbr_X, cbr_y, train_size=0.75, random_state=42)
```

O modelo foi criado tendo como parâmetro uma lista com as colunas que contêm as variáveis categóricas para que o algoritmo identifique e faça o correto tratamento dos dados. Juntamente com os dados de treinamento são passados como argumento os dados de validação para medição da performance pelo algoritmo.

```
In [98]: cbr = CatBoostRegressor()

In [101]: cbr.fit(
              cbr_X_train, cbr_y_train,
              cat_features=[ 0,  1,  2, 7, 8, 9 ],
              eval_set=(cbr_X_validation, cbr_y_validation),
              plot=False
          );

Learning rate set to 0.112063
0:      learn: 87.4123114       test: 102.6006562       best: 102.6006562 (0)   total: 139ms    remaining: 2m 18s
1:      learn: 81.0682895       test: 97.8117981        best: 97.8117981 (1)    total: 273ms    remaining: 2m 16s
2:      learn: 75.3247790       test: 92.5430672        best: 92.5430672 (2)    total: 399ms    remaining: 2m 12s
3:      learn: 70.4173225       test: 88.4707967        best: 88.4707967 (3)    total: 526ms    remaining: 2m 10s
4:      learn: 65.9903418       test: 85.4566220        best: 85.4566220 (4)    total: 650ms    remaining: 2m 9s
5:      learn: 62.0943718       test: 81.8775675        best: 81.8775675 (5)    total: 768ms    remaining: 2m 7s
6:      learn: 58.5337308       test: 79.0356351        best: 79.0356351 (6)    total: 886ms    remaining: 2m 5s
7:      learn: 55.4691525       test: 76.3446371        best: 76.3446371 (7)    total: 1.02s    remaining: 2m 6s
8:      learn: 52.5693316       test: 73.2386197        best: 73.2386197 (8)    total: 1.13s    remaining: 2m 4s
9:      learn: 50.1958459       test: 71.0148639        best: 71.0148639 (9)    total: 1.26s    remaining: 2m 4s
10:     learn: 48.0860772       test: 69.6999381        best: 69.6999381 (10)   total: 1.38s    remaining: 2m 4s
11:     learn: 46.0816551       test: 68.3152025        best: 68.3152025 (11)   total: 1.47s    remaining: 2m
12:     learn: 44.2647626       test: 66.6402503        best: 66.6402503 (12)   total: 1.61s    remaining: 2m 2s
13:     learn: 42.6386253       test: 64.7811708        best: 64.7811708 (13)   total: 1.74s    remaining: 2m 2s
14:     learn: 41.2263191       test: 63.1328086        best: 63.1328086 (14)   total: 1.85s    remaining: 2m 1s
15:     learn: 39.9086981       test: 61.6582044        best: 61.6582044 (15)   total: 1.98s    remaining: 2m 2s
16:     learn: 38.8671690       test: 61.0744286        best: 61.0744286 (16)   total: 2.09s    remaining: 2m 1s
17:     learn: 37.9286058       test: 60.0727405        best: 60.0727405 (17)   total: 2.18s    remaining: 1m 59s

In [102]: cbr.score(cbr_X_validation,cbr_y_validation)
Out[102]: 0.824712028615664
```

Para a busca em grade a biblioteca Scikit Learn, usada nas implementações anteriores, exige o encoding das variáveis categóricas. Para tirar proveito das vantagens oferecidas pelo algoritmo, uma implementação simples em python foi utilizada, percorrendo várias combinações de hiperparâmetros, buscando os melhores valores de R2 e MAE.

```
In [118]: grid = {'learning_rate': [0.03, 0.1, 0.2, 0.3, 0.4],
                  'depth': [4, 6, 10],
                  'l2_leaf_reg': [1, 3, 5, 7, 9]}

In [120]: for l in grid['learning_rate']:
              for d in grid['depth']:
                  for lr in grid['l2_leaf_reg']:
                      cbr_grid = CatBoostRegressor(learning_rate=l,depth=d,l2_leaf_reg=lr)
                      cbr_grid.fit(cbr_X_train, cbr_y_train,cat_features=[ 0,  1,  2, 7, 8, 9 ],eval_set=(cbr_X_validation, cbr_y_validatic
                      score = cbr_grid.score(cbr_X_validation,cbr_y_validation)
                      cbr_y_validation_predict = cbr_grid.predict(cbr_X_validation)
                      mae = metrics.mean_absolute_error(cbr_y_validation,cbr_y_validation_predict)
                      print('learning_rate: ',l,' - depth: ',d,' - l2_leaf_reg: ',lr,' - score: ',score,' - MAE: ',mae)

Custom logger is already specified. Specify more than one logger at same time is not thread safe.

learning_rate:  0.03  - depth:  4  - l2_leaf_reg:  1  - score:  0.8215847206783067  - MAE:  11.594492251865976
learning_rate:  0.03  - depth:  4  - l2_leaf_reg:  3  - score:  0.8239176789329612  - MAE:  11.666054908528483
learning_rate:  0.03  - depth:  4  - l2_leaf_reg:  5  - score:  0.8253581475259278  - MAE:  11.722789543443042
learning_rate:  0.03  - depth:  4  - l2_leaf_reg:  7  - score:  0.8287644711577871  - MAE:  11.880227899210519
learning_rate:  0.03  - depth:  4  - l2_leaf_reg:  9  - score:  0.8084635874197975  - MAE:  12.426329709880498
learning_rate:  0.03  - depth:  6  - l2_leaf_reg:  1  - score:  0.8008439840411603  - MAE:  11.536816110441809
learning_rate:  0.03  - depth:  6  - l2_leaf_reg:  3  - score:  0.8229405561849895  - MAE:  11.573424888389722
learning_rate:  0.03  - depth:  6  - l2_leaf_reg:  5  - score:  0.7918737222209755  - MAE:  11.717126251644181
learning_rate:  0.03  - depth:  6  - l2_leaf_reg:  7  - score:  0.78370849819398  - MAE:  11.806208160157505
learning_rate:  0.03  - depth:  6  - l2_leaf_reg:  9  - score:  0.7817450598936169  - MAE:  12.139755114933665
learning_rate:  0.03  - depth:  10  - l2_leaf_reg:  1  - score:  0.6928254849023932  - MAE:  11.60070570050595
learning_rate:  0.03  - depth:  10  - l2_leaf_reg:  3  - score:  0.6804251860499053  - MAE:  11.632309693778767
learning_rate:  0.03  - depth:  10  - l2_leaf_reg:  5  - score:  0.6847442786715252  - MAE:  11.70397325737843
learning_rate:  0.03  - depth:  10  - l2_leaf_reg:  7  - score:  0.6766633665263102  - MAE:  11.778672982223972
learning_rate:  0.03  - depth:  10  - l2_leaf_reg:  9  - score:  0.6910275080663226  - MAE:  11.934439107104554
learning_rate:  0.1  - depth:  4  - l2_leaf_reg:  1  - score:  0.8265618002115682  - MAE:  11.552923214138248
learning_rate:  0.1  - depth:  4  - l2_leaf_reg:  3  - score:  0.8314697832894434  - MAE:  11.592168676070495
learning_rate:  0.1  - depth:  4  - l2_leaf_reg:  5  - score:  0.8247002095113374  - MAE:  11.674122383944034
learning_rate:  0.1  - depth:  4  - l2_leaf_reg:  7  - score:  0.8176293919587236  - MAE:  12.1526723763038
learning_rate:  0.1  - depth:  4  - l2_leaf_reg:  9  - score:  0.8066484496843058  - MAE:  12.658536055650272
learning_rate:  0.1  - depth:  6  - l2_leaf_reg:  1  - score:  0.821511528586987  - MAE:  11.496616819596351
```

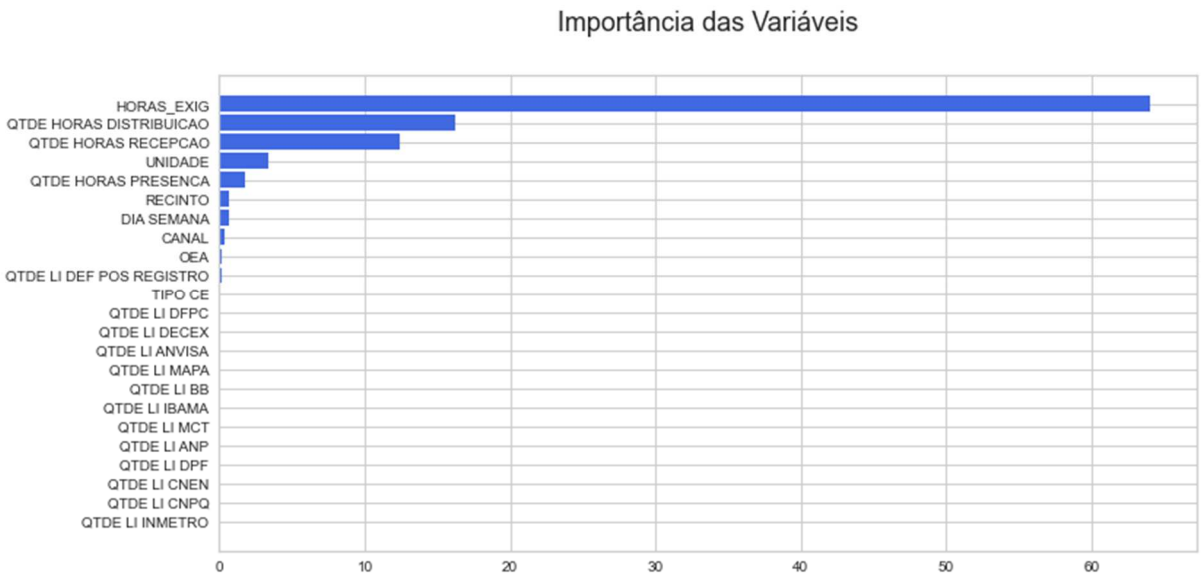
O modelo final com os hiperparâmetros encontrados com a técnica acima foi ligeiramente superior ao original, chegando a um coeficiente de determinação de 0.83.

```
In [38]: cbr_final = CatBoostRegressor(learning_rate=0.1,depth=4,l2_leaf_reg=3)

In [39]: cbr_final.fit(
             cbr_X_train, cbr_y_train,
             cat_features=[ 0,  1,  2, 7, 8, 9  ],
             eval_set=(cbr_X_validation, cbr_y_validation),
             plot=False,
             verbose=False
         );

In [40]: cbr_final.score(cbr_X_validation, cbr_y_validation)
Out[40]: 0.8314697832894434

In [41]: cbr_y_validation_predict = cbr_final.predict(cbr_X_validation)
```

Resultados

```
In [42]: metrics.r2_score(cbr_y_validation,cbr_y_validation_predict)
Out[42]: 0.8314697832894434

In [43]: metrics.mean_absolute_error(cbr_y_validation,cbr_y_validation_predict)
Out[43]: 11.592168676070495
```

Outra vantagem do algoritmo é que possui uma função que revela a contribuição de cada variável para o modelo, ou *feature importance*, que pode ser usado na seleção das variáveis.



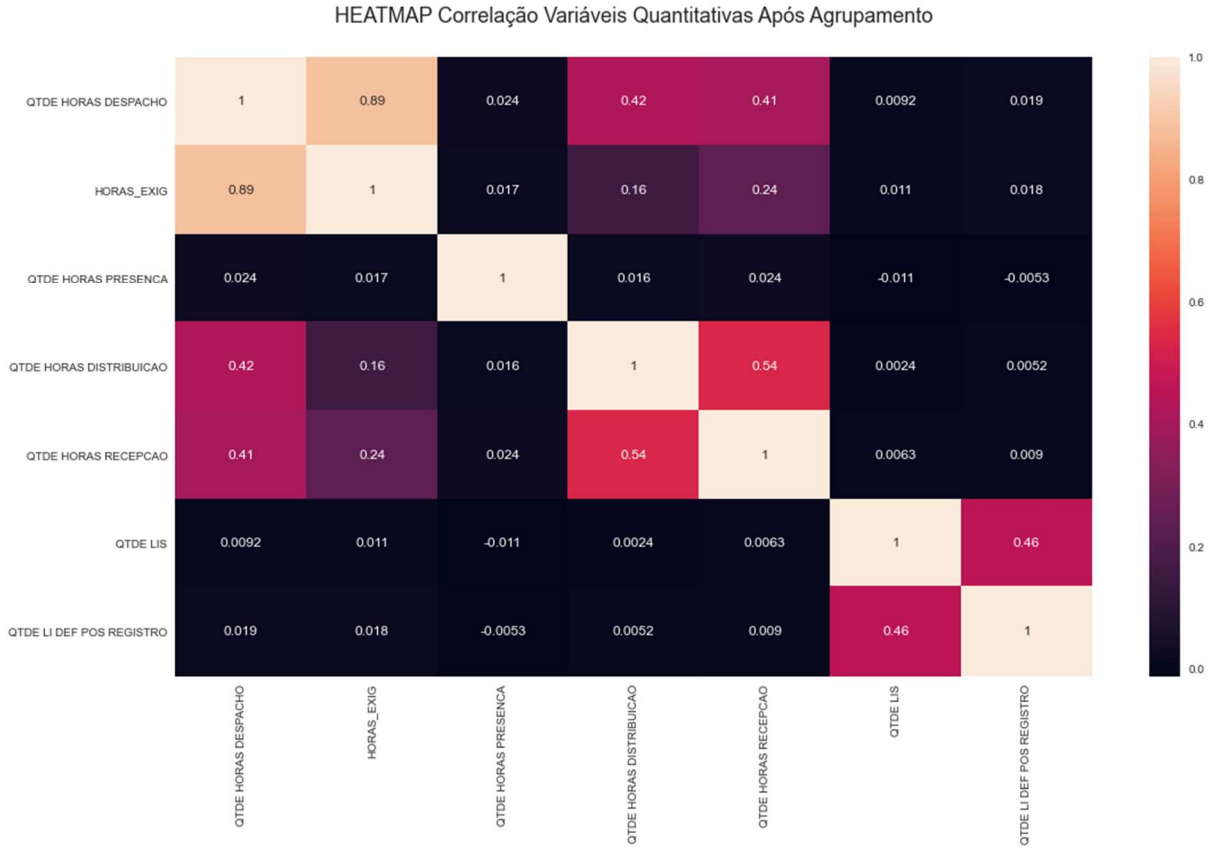
O modelo obtido demonstra uma pouca contribuição das quantidades de LI. Uma possível estratégia seria combinar seus valores como uma única coluna. (Já se havia constatado essa possibilidade na etapa de Análise e Exploração de Dados)

```
In [9]: df2['QTDE LIS']=df2['QTDE LI DECEX']+df2['QTDE LI INMETRO']+df2['QTDE LI ANVISA']+df2['QTDE LI IBAMA']+df2['QTDE LI MAPA']+df2['Q

In [13]: df2[['QTDE LIS','QTDE HORAS DESPACHO']].corr()
```

Out[13]:

	QTDE LIS	QTDE HORAS DESPACHO
QTDE LIS	1.000000	0.009164
QTDE HORAS DESPACHO	0.009164	1.000000

Observa-se que a correlação entre a nova variável e a variável alvo é bem mais significativa que as originais individualmente.



Criou-se um novo conjunto de treino e teste com o ajuste nas variáveis utilizadas, desta vez utilizando a ferramenta do próprio algoritmo.

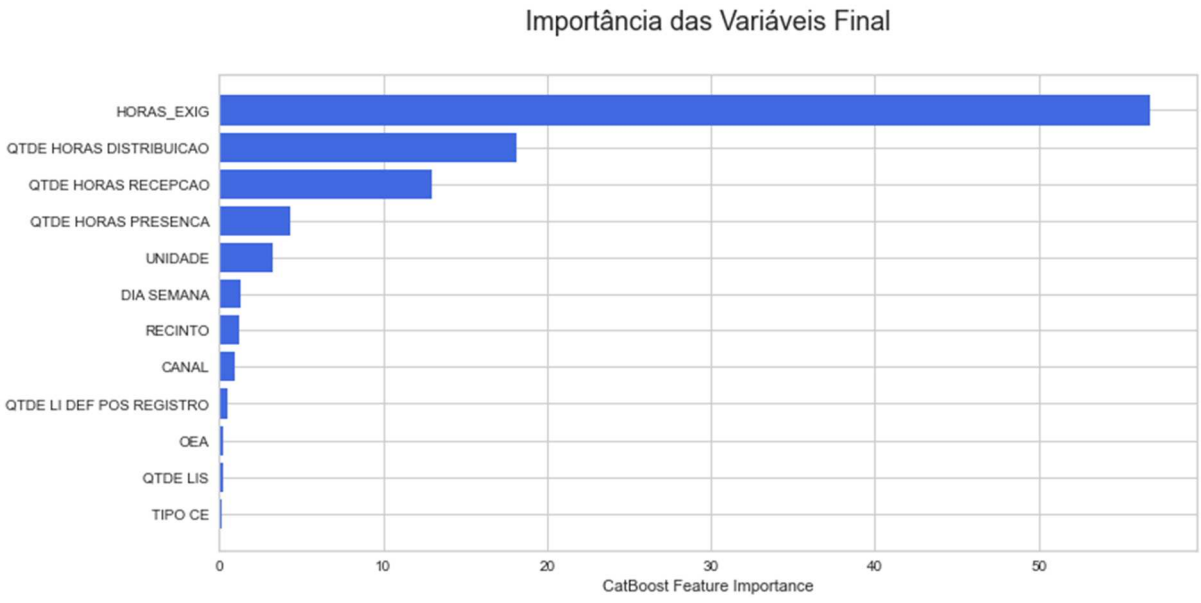
```
In [30]: cbr_X = df2.drop(['QTDE HORAS DESPACHO','ID DI'], axis=1)
         cbr_y = df2['QTDE HORAS DESPACHO']
         cbr_X_train, cbr_X_validation, cbr_y_train, cbr_y_validation = train_test_split(cbr_X, cbr_y, random_state=42)

In [31]: variaveis = ['TIPO CE', 'OEA', 'CANAL', 'HORAS_EXIG',
                '''QTDE HORAS PRESENCA', 'QTDE HORAS DISTRIBUICAO', 'QTDE HORAS RECEPCAO',
                'DIA SEMANA', 'UNIDADE', 'RECINTO','QTDE LIS' , 'QTDE LI DEF POS REGISTRO']
         variaveis_cat = ['TIPO CE','OEA','CANAL','DIA SEMANA', 'UNIDADE', 'RECINTO']

In [32]: train_pool = Pool(cbr_X_train[variaveis], cbr_y_train, cat_features=variaveis_cat)
         test_pool = Pool(cbr_X_validation[variaveis], cbr_y_validation, cat_features=variaveis_cat)
```

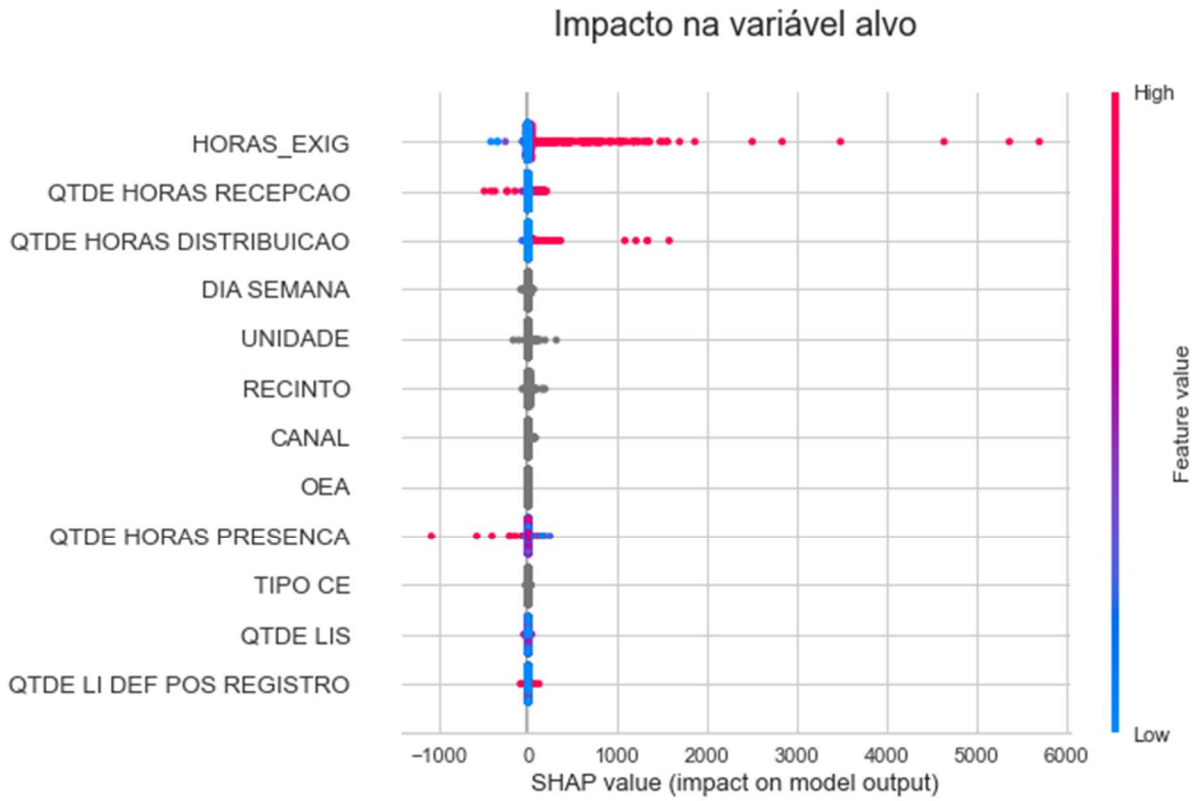
O modelo criado com a implementação padrão obteve resultados bem próximos do modelo anterior, porém a eliminação de 12 colunas do dataframe melhorou o custo computacional para sua execução.

```
In [33]: model = CatBoostRegressor()

In [34]: model.fit(train_pool, eval_set=test_pool, verbose=False)
Out[34]: <catboost.core.CatBoostRegressor at 0x2211e67f580>

In [35]: model.score(test_pool)
Out[35]: 0.8075264985001753

In [36]: cbr_y_validation_predict = model.predict(cbr_X_validation)

In [37]: skmetrics.mean_absolute_error(cbr_y_validation,cbr_y_validation_predict)
Out[37]: 11.691337507393127
```

Por fim, utilizando a busca em grade foi possível obter um modelo ainda melhor adaptado:

```
In [40]: model_grid = CatBoostRegressor()
         grid = {'iterations': [100, 150, 200],
                 'learning_rate': [0.03, 0.1, 0.2, 0.3, 0.4],
                 'depth': [4, 6, 10],
                 'l2_leaf_reg': [1, 3, 5, 7, 9]}
         model_grid.grid_search(grid, train_pool,cv=10,verbose=False)

Custom logger is already specified. Specify more than one logger at same time is not thread safe.

0:      learn: 93.4954762       test: 102.7918652       best: 102.7918652 (0)   total: 20.9ms   remaining: 2.07s
1:      learn: 91.5110766       test: 100.5986417       best: 100.5986417 (1)   total: 44.8ms   remaining: 2.19s
2:      learn: 89.6344230       test: 98.3969502        best: 98.3969502 (2)    total: 65.5ms   remaining: 2.12s
3:      learn: 87.7769597       test: 96.6400265        best: 96.6400265 (3)    total: 83.9ms   remaining: 2.01s
4:      learn: 86.0307328       test: 94.5007988        best: 94.5007988 (4)    total: 102ms    remaining: 1.95s
5:      learn: 84.3437070       test: 92.6120875        best: 92.6120875 (5)    total: 120ms    remaining: 1.88s
6:      learn: 82.7104942       test: 90.5691459        best: 90.5691459 (6)    total: 140ms    remaining: 1.86s
7:      learn: 81.0857965       test: 88.7705986        best: 88.7705986 (7)    total: 160ms    remaining: 1.84s
8:      learn: 79.5273052       test: 86.8314230        best: 86.8314230 (8)    total: 178ms    remaining: 1.8s
9:      learn: 78.0535636       test: 84.9674178        best: 84.9674178 (9)    total: 196ms    remaining: 1.76s
10:     learn: 76.6359145       test: 83.0476960        best: 83.0476960 (10)   total: 213ms    remaining: 1.72s
11:     learn: 75.2261255       test: 81.3315088        best: 81.3315088 (11)   total: 232ms    remaining: 1.7s
12:     learn: 73.8987040       test: 79.6152864        best: 79.6152864 (12)   total: 253ms    remaining: 1.69s
13:     learn: 72.5674835       test: 78.2795879        best: 78.2795879 (13)   total: 271ms    remaining: 1.66s
14:     learn: 71.2799720       test: 76.5157016        best: 76.5157016 (14)   total: 288ms    remaining: 1.63s
15:     learn: 70.0492711       test: 75.0009121        best: 75.0009121 (15)   total: 306ms    remaining: 1.6s
16:     learn: 68.8665652       test: 73.6778776        best: 73.6778776 (16)   total: 325ms    remaining: 1.59s
17:     learn: 67.7521174       test: 72.1250972        best: 72.1250972 (17)   total: 344ms    remaining: 1.57s

In [41]: pred = model_grid.predict(cbr_X_validation)

In [42]: model_grid.score(test_pool)
Out[42]: 0.8117256808001351

In [43]: skmetrics.mean_absolute_error(cbr_y_validation,pred)
Out[43]: 11.54454988878532
```





Nota-se que o modelo agora aproveita-se de todas as variáveis disponíveis, indicando que o procedimento obteve o resultado pretendido.

Por fim, compara-se os resultados previstos com os alvos no dataset de validação. Em um modelo perfeito os pontos estariam alinhados em um ângulo de 45º.

6. Apresentação dos Resultados

Os resultados foram obtidos utilizando duas métricas bastante comuns em modelos de regressão: O Coeficiente de Determinação (R^2) e o Erro Absoluto Médio (MAE).

O coeficiente de determinação é uma medida de ajuste de um modelo estatístico linear generalizado. Desta forma quanto maior o R^2, mais explicativo é o modelo. Por exemplo, um $R^2 = 0,8234$ significa que o modelo linear explica 82,34% da variância da variável alvo a partir das demais variáveis.

Já Erro Absoluto Médio é uma medida de erros entre observações pareadas que expressam o mesmo fenômeno, portanto expressa o erro de previsão médio absoluto de um modelo. Neste caso, como está na mesma unidade do alvo (horas), quanto menor seu valor, melhor é a avaliação do modelo.

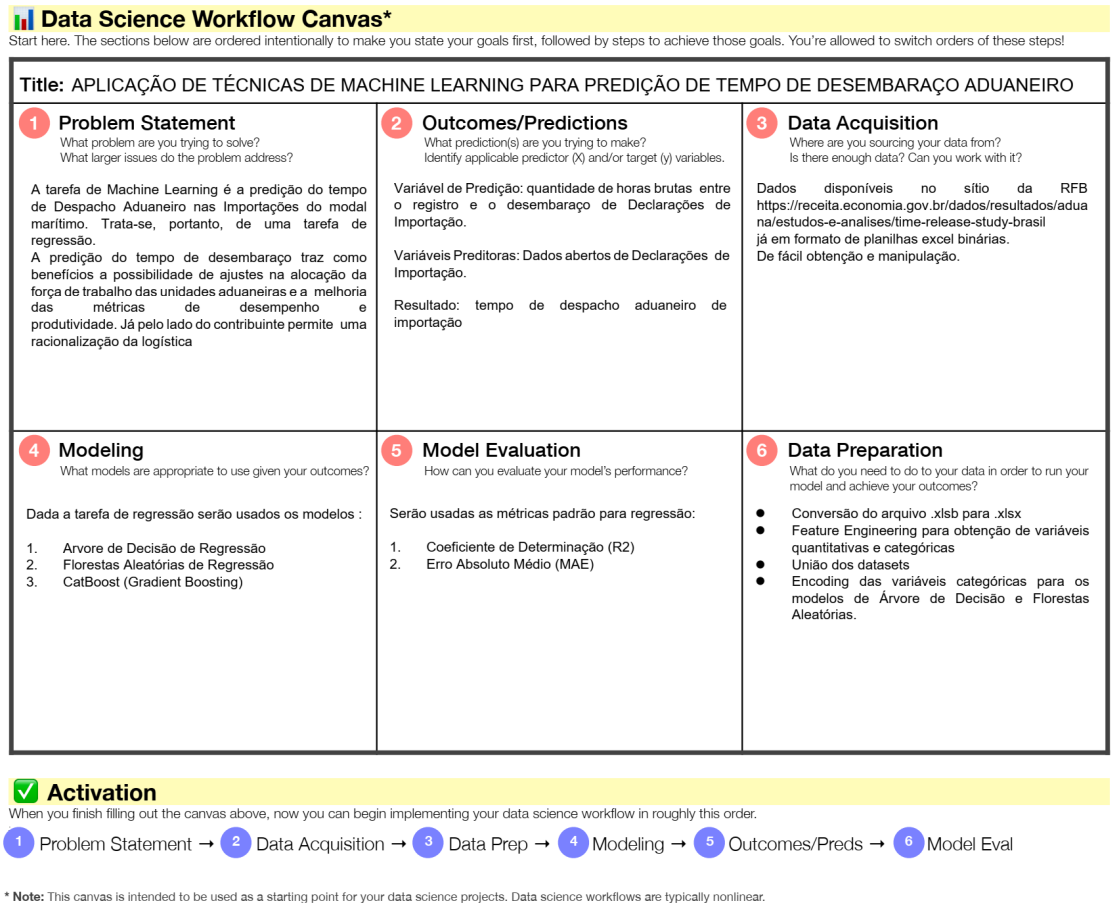
Modelo	Coeficiente de Determinação (R^2)	Erro Absoluto Médio
Arvore de Decisão	0.5802	11.1954
Arvore de Decisão com busca em grade	0,6372	11.7339
Floresta Aleatória	0,7511	10,0451
Floresta Aleatória com busca em grade	0,7610	10,7627
CatBoost	0,8247	11,5652
CatBoost com busca em grade	0,8314	11,5921
CatBoost com seleção de variáveis	0,8075	11,6913
CatBoost com seleção de variáveis e busca em grade	0,8117	11,5445

Observa-se nos resultados que o modelo que melhor se ajusta é o CatBoost com busca em grade, enquanto a Floresta Aleatória apresenta o menor erro médio absoluto.

Considerando as variáveis disponíveis é possível considerar um erro médio em torno de 11 horas bastante razoável. Há de se considerar também, que a instituição

RFB possui uma gama enorme de dados e informações que incluem mercadorias, operações comerciais e operadores, mas que por motivo de sigilo fiscal não puderam ser aproveitados neste estudo. Desta forma, é bastante factível que se obtenha modelos com melhores resultados longe desta restrição.

É de se concluir com os resultados obtidos, que as técnicas de *machine learning* podem produzir modelos plenamente aplicáveis para que a RFB continue melhorando sua eficiência, aperfeiçoando seus processos de trabalho e entregue valor para a sociedade.



Data Science Workflow Canvas*

Start here. The sections below are ordered intentionally to make you state your goals first, followed by steps to achieve those goals. You're allowed to switch orders of these steps!

Title: APLICAÇÃO DE TÉCNICAS DE MACHINE LEARNING PARA PREDIÇÃO DE TEMPO DE DESEMBARAÇO ADUANEIRO

1 Problem Statement
What problem are you trying to solve?
What larger issues do the problem address?

A tarefa de Machine Learning é a predição do tempo de Despacho Aduaneiro nas Importações do modal marítimo. Trata-se, portanto, de uma tarefa de regressão.
A predição do tempo de desembaraço traz como benefícios a possibilidade de ajustes na alocação da força de trabalho das unidades aduaneiras e a melhoria das métricas de desempenho e produtividade. Já pelo lado do contribuinte permite uma racionalização da logística

2 Outcomes/Predictions
What prediction(s) are you trying to make?
Identify applicable predictor (X) and/or target (y) variables.

Variável de Predição: quantidade de horas brutas entre o registro e o desembaraço de Declarações de Importação.

Variáveis Preditoras: Dados abertos de Declarações de Importação.

Resultado: tempo de despacho aduaneiro de importação

3 Data Acquisition
Where are you sourcing your data from?
Is there enough data? Can you work with it?

Dados disponíveis no sítio da RFB https://receita.economia.gov.br/dados/resultados/aduana/estudos-e-analises/time-release-study-brasil
já em formato de planilhas excel binárias.
De fácil obtenção e manipulação.

4 Modeling
What models are appropriate to use given your outcomes?

Dada a tarefa de regressão serão usados os modelos :

1. Arvore de Decisão de Regressão
2. Florestas Aleatórias de Regressão
3. CatBoost (Gradient Boosting)

5 Model Evaluation
How can you evaluate your model's performance?

Serão usadas as métricas padrão para regressão:

1. Coeficiente de Determinação (R2)
2. Erro Absoluto Médio (MAE)

6 Data Preparation
What do you need to do to your data in order to run your model and achieve your outcomes?

- Conversão do arquivo .xlsb para .xlsx
- Feature Engineering para obtenção de variáveis quantitativas e categóricas
- União dos datasets
- Encoding das variáveis categóricas para os modelos de Árvore de Decisão e Florestas Aleatórias.

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.

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

7. Links

Link para o vídeo: https://youtu.be/NlVT-dH80LA

Link para o repositório: https://github.com/alacorte/tccpucmg

REFERÊNCIAS

B. CESTNIK et al. **Estimating probabilities: a crucial task in machine learning**. In ECAI, volume 90, pages 147–149, 1990.

D. Micci-Barreca. **A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems**. ACM SIGKDD Explorations Newsletter, 3(1):27–32, 2001.

RFB. Receita Federal do Brasil. Acordo de Facilitação de Comércio. Disponível em <https://www.gov.br/receitafederal/pt-br/assuntos/aduana-e-comercio-exterior/importacao-e-exportacao/oea/arquivos-e-imagens/arquivos/AcordodeFacilitaodoComrcio-traduzido.pdf>. Acesso em: 26 de jun. de 2021

RFB. Receita Federal do Brasil. Balanço Aduaneiro. Disponível em: <https://receita.economia.gov.br/dados/resultados/aduana/arquivos-e-imagens/BalanoAduaneiroAno2019COANA.pdf>. Acesso em: 20 de jun. de 2021.

WHO. World Customs Organization. Guide to measure the time required for release of goods. Disponível em: <http://www.wcoomd.org/-/media/wco/public/global/pdf/topics/facilitation/instruments-and-tools/tools/time-release-study/time_release-_study.pdf?db=web>. Acesso em: 20 de jun. de 2021.

APÊNDICE

Programação/Scripts

```
#!/usr/bin/env python
# coding: utf-8

# ### Imports Necessários

# In[87]:


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pandas_profiling
import seaborn as sns
import missingno as msno
import shap
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn import metrics as skmetrics
from yellowbrick.regressor import ResidualsPlot
from yellowbrick.regressor import PredictionError
from yellowbrick.contrib.wrapper import wrap


# ## Tabela de Conteúdo:
# * [2 - COLETA DE DADOS](#coleta)
#     * [2.1 Dataset Declarações com Declarações de Importação](#col-di)
#     * [2.2 Dataset Declarações com Licenciamento de Importação](#col-li)
#     * [2.3 Dataset Transito Aduaneiro](#col-tran)
# * [3 - PROCESSAMENTO E TRATAMENTO DE DADOS](#proc)
#     * [3.1 Dados Declaração de Importação](#proc-di)
#     * [3.2 Dataset Dados LI](#proc-li)
#     * [3.3 Dataset Dados Transito](#proc-tran)
#     * [3.4 União dos Datasets](#proc-join)
#     * [3.5 Tratamento Dados Ausentes](#proc-miss)
# * [4 - ANÁLISE E EXPLORAÇÃO DOS DADOS](#aed)
#     * [4.1 - Variáveis Categóricas](#aed-cat)
#         * [4.1.1 - Aferição da Cardinalidade](#aed-card)
#         * [4.1.2 - TIPO CE](#aed-ce)
#         * [4.1.3 - OEA](#aed-oea)
#         * [4.1.4 - CANAL](#aed-canal)
#         * [4.1.5 - DIA SEMANA](#aed-dia)
#         * [4.1.6 - UNIDADE](#aed-unidade)
```

```
#         * [4.1.7 - RECINTO](#aed-recinto)
#     * [4.2 - Variáveis Quantitativas](#aed-quant)
# * [5 - CRIAÇÃO DE MODELOS DE MACHINE LEARNING](#mod)
#     * [5.1 - Decision Tree Regressor ](#mod-dtree)
#     * [5.2 - RandomForest Regressor](#mod-rfor)
#     * [5.3 - CatBoost Regressor](#mod-catb)
#

# # 2 - COLETA DE DADOS <a class="anchor" id="coleta"></a>

# #### Time Release Study - Brasil
# A Secretaria Especial da Receita Federal do Brasil (RFB), em parce-
ria com a Secex, Anvisa e Mapa, realizou o primeiro Estudo de Tempos de Libe-
ração de Cargas, desenvolvido conforme a metodologia da Organização Mun-
dial das Aduanas (OMA), Time Release Study. O estudo repre-
senta um marco na Administração Aduaneira Brasileira na medida em que são ofe-
recidas informações relevantes para todo o público de comércio exte-
rior, tanto brasileiro como internacional, ampliando a transparência e enga-
jando os diversos atores do processo em busca de melhorias.
# A iniciativa decorre de medida prevista no Acordo de Facilitação de comér-
cio (AFC), da Organização Mundial de Comércio (OMC), do qual o Brasil é signa-
tário, e visa prover maior transparência nas informações relativas ao comér-
cio exterior.
# Os tempos medidos compreendem o processo integral da importa-
ção, ou seja, desde a chegada do veículo transportador até a en-
trega da carga ao importador, envolvendo todas as unidades nos modais aé-
reo (foram 21 unidades) e marítimo (22 unidades no total) e as duas princi-
pais do modal rodoviário, que juntas responderam por cerca de 46% da movimen-
tação do modal.
# A realização do estudo contou com apoio do Grupo Banco Mundial, da Organiza-
ção Mundial de Aduanas e do Fundo do Reino Unido para a Prosperidade.
# > O estudo completo você encontra [aqui](https://receita.economia.gov.br/da-
dos/resultados/aduana/estudos-e-analises/time-release-study-brasil).
# ___
#
# Os datasets utilizados estão disponibilizada no sitio da RFB na internet.
# > Voce pode encontrar os datasets [aqui](https://receita.economia.gov.br/da-
dos/resultados/aduana/estudos-e-analises/TRS2020_Martimopblico.xlsb).

# ## 2.1 Dataset Declarações com Declarações de Importação <a class="an-
chor" id="col-di"></a>
# O segundo dataset contém informações acerca das Declaraçõess de Importa-
ção, doravante DI, individualizado por declaração.

# In[5]:


# Carregar o arquivo com a base de dados modal maritmo
```

```
df_di = pd.read_excel("base/TRS2020_Martimopblico.xlsx",sheet_name='Da-
dos com exclusões')
df_di.info()


# In[6]:


df_di.head()


# ## 2.2 Dataset Declarações com Licenciamento de Importação <a class="an-
chor" id="col-li"></a>
# O segundo dataset contém informações acerca dos Licenciamentos de Importa-
ção, doravante LI, individualizado por licença.

# In[7]:


# Carregar o arquivo com a base de dados modal maritmo
df_li = pd.read_excel("base/TRS2020_Martimopblico.xlsx",sheet_name='Da-
dos com LI')
df_li.info()


# In[8]:


df_li.head()


# ## 2.3 Dataset Transito Aduaneiro <a class="anchor" id="col-tran"></a>
# O terceiro dataset contém informações gerais da DI e Declarações de Tran-
sito Aduaneiro associadas.

# In[9]:


# Carregar o arquivo com a base de dados modal maritmo
df_tran = pd.read_excel("base/TRS2020_Martimopblico.xlsx",sheet_name='Da-
dos brutos')
df_tran.info()


# In[10]:


df_tran.head()
```

```
# # 3 - PROCESSAMENTO E TRATAMENTO DE DADOS <a class="anchor" id="proc"></a>

# ## 3.1 Dados Declaração de Importação <a class="anchor" id="proc-di"></a>

# ### 3.1.1 - QTDE HORAS DESPACHO - Variável Alvo (LABEL)
# Representa a Quantidade de horas entre o registro da declaração de importação e seu desembaraço, representando o tempo bruto em que a declaração permaneceu sob procedimento fiscal. Tratando-se da diferença entre duas variáveis do tipo datetime (timedelta), divide-se a diferença pela unidade que se pretende obter.

# In[11]:


df_di['QTDE HORAS DESPACHO'] = (df_di['DT DESEMBARACO DI'] - df_di['DT REGISTRO DI'])/pd.Timedelta(hours=1)
df_di[['DT DESEMBARACO DI','DT REGISTRO DI','QTDE HORAS DESPACHO']].head()


# ### 3.1.2 - QTDE HORAS PRESENCA
# Representa a Quantidade de horas entre o atracação e presença de carga no recinto, representando o tempo bruto para movimentação da carga.

# In[12]:


df_di['QTDE HORAS PRESENCA'] = (df_di['DATA PRESENCA'] - df_di['DATA ATRACACAO'])/pd.Timedelta(hours=1)
df_di[['DATA ATRACACAO','DATA PRESENCA','QTDE HORAS PRESENCA']].head()


# ### 3.1.3 - QTDE HORAS DISTRIBUIÇÃO
# Representa a Quantidade de horas entre o registro de DI e a distribuição para o fiscal responsável.

# In[13]:


df_di['QTDE HORAS DISTRIBUICAO'] = (df_di['DT_DISTRI'] - df_di['DT REGISTRO DI'])/pd.Timedelta(hours=1)
df_di[['DT REGISTRO DI','DT_DISTRI','QTDE HORAS DISTRIBUICAO']][df_di['QTDE HORAS DISTRIBUICAO']>0].head()


# ### 3.1.4 - QTDE HORAS RECEPCAO
# Representa a Quantidade de horas entre o registro da declaração e a entrega dos documentos instrutivos para as dis selecionadas.
```

```
# In[14]:


df_di['QTDE HORAS RECEPCAO'] = (df_di['DT_RECEPCAO'] - df_di['DT REGIS-
TRO DI'])/pd.Timedelta(hours=1)
df_di[['DT REGISTRO DI','DT_RECEPCAO','QTDE HORAS RECEPCAO']][df_di['QTDE HO-
RAS RECEPCAO']>0].head()


# ### 3.1.5 - DIA DA SEMANA
# Variável categórica que representa o dia da semana do Registro da DI

# In[15]:


df_di['DIA SEMANA'] = df_di['DT REGISTRO DI'].dt.weekday
dias = {0:'SEGUNDA',1:'TERCA',2:'QUARTA',3:'QUINTA',4:'SEXTA',5:'SABA-
DO',6:'DOMINGO'}
df_di['DIA SEMANA'] = df_di['DIA SEMANA'].apply(lambda x: dias[x])
df_di[['DT REGISTRO DI','DIA SEMANA']].head()


# ### 3.1.6 UNIDADE
# Concatenação de UL PRESENCA e DESCRICAO UL

# In[16]:


df_di['UNIDADE'] = df_di['UL PRESENCA'].astype(str)+" - "+df_di['DESCRI-
CAO UL']
df_di['UNIDADE'].head()


# ### 3.1.7 RECINTO
# Concatenação de RA PRESENCA e DESCRICAO RA

# In[17]:


df_di['RECINTO'] = df_di['RA PRESENCA'].astype(str)+" - "+df_di['DESCRI-
CAO RA']
df_di['RECINTO'].head()


# ### 3.1.8 LIMPEZA

# In[18]:


df_di.columns
```

```
# In[19]:


df_di.drop(['UL PRESENCA','DESCRICAO UL', 'RA PRESENCA', 'DESCRI-
CAO RA', 'DATA ATRACACAO','DATA PRESENCA', 'REG_LI',
            'DEFERIMENTO_LI', 'ANUÊNCIA?', 'DT REGISTRO DI', 'DT_SELE-
CAO', 'DT_RECEPCAO', 'DT_DISTRI', 'DT DESEMBARACO DI',
            'DT ENTREGA', 'ATRAC - ENTR', 'LOG'],axis='columns', inplace=True)
df_di.info()


# In[20]:


df_di.head()


# ## 3.2 Dataset Dados LI <a class="anchor" id="proc-li"></a>

# ### 3.2.1 - ORGAOS ANUENTES
# O dataset apresenta uma linha para cada Licença de Importação e até 4 or-
gãos anuentes para cada LI. A intenção é agrupar as LIs por quanti-
dade e por orgão anuente presente em cada delcaração de importação.

# In[21]:


df_li['ANUENTE'].value_counts()


# In[22]:


df_li['ANUENTE2'].value_counts()


# In[23]:


df_li['ANUENTE6'].value_counts()


# In[24]:


df_li['ANUENTE10'].value_counts()


# In[25]:
```

```
dum_df = pd.get_dummies(df_li, columns=['ANUENTE','ANUENTE2','ANUENTE6','ANU-
ENTE10'], prefix=['A1','A2','A3','A4'] )


# In[26]:


def criaListaColunas(df, str):
    lista = []
    if ('A1_'+str in df.columns):
        lista.append('A1_'+str)
    if('A2_'+str in df.columns):
        lista.append('A2_'+str)
    if('A3_'+str in df.columns):
        lista.append('A3_'+str)
    if('A4_'+str in df.columns):
        lista.append('A4_'+str)
    return lista


# In[27]:


orgaos = df_li['ANUENTE'].unique()
for orgao in orgaos:
    colunas = criaListaColunas(dum_df, orgao)
    dfgroup = dum_df.groupby('ID DI')[colunas].sum()
    serie = dfgroup.sum(axis=1)
    serie.name = 'QTDE LI '+ orgao
    df_li = df_li.join(serie, on='ID DI')


# In[28]:


# Removemos o dados usado para a agregações para liberar memória
del dum_df
del dfgroup
del serie


# ### 3.2.2 - Deferimento LI após registro da DI
# Ainda sobre as licenças de importação, criaremos uma coluna que indi-
que se alguma licença foi obtida após o registro da declarção de importação

# In[31]:
```

```
df_li['DATA DEFERIMENTO'].describe()


# Conversão para o tipo datetime a fim de permitir operções entre as datas

# In[32]:


df_li['DATA DEFERIMENTO'] = pd.to_datetime(df_li['DATA DEFERIMENTO'])


# In[33]:


df_li['LI DEF POS REGISTRO'] = np.where(df_li['DT REGIS-
TRO DI']<df_li['DATA DEFERIMENTO'],1,0)


# Agrupamos as quantidades de LIs deferidas após o registro por DI

# In[34]:


agrupado = df_li.groupby('ID DI')['LI DEF POS REGISTRO'].sum()
agrupado.name = 'QTDE LI DEF POS REGISTRO'
df_li = df_li.join(agrupado, on='ID DI')


# ### 3.2.3 - Limpeza Dados
# Remoção colunas não utilizadas e agregação dos dados à nível de DI ('ID DI')

# In[35]:


df_li.columns


# In[36]:


df_li.drop(['ID LI','TIPO CONHECIMENTO', 'MODALIDADE DESPACHO',
       'UL PRESENCA', 'DESCRICAO UL', 'RA PRESENCA',
       'TIPO DECLARACAO IMPORTACAO', 'DESCRICAO RA', 'CANAL', 'DATA EMISSAO',
       'DATA ATRACACAO', 'DATA TRANSITO', 'DATA PRESENCA', 'DATA DEFERIMENTO',
       'DT REGISTRO DI', 'DT DESEMBARACO DI', 'DT ENTRE-
GA', 'DT REG', 'HR REG',
       'DATA/HORA REGISTRO', 'antes  pres?', 'DT SIT', 'HR SIT', 'SIT', 'QTD',
       'ANUENTE', 'SITUAÇÃO', 'DATA', 'HORA', 'DATA HORA REG', 'ANUENTE2',
       'SITUAÇÃO3', 'antes depois 1', 'DATA4', 'HORA5', 'ANUENTE6',
```

```
       'SITUAÇÃO7', 'ANUENTE10', 'SITUAÇÃO11', 'REG-DEF (4)','LI DEF POS RE-
GISTRO'],axis='columns', inplace=True)


# In[37]:


df_li['QTDE LI DECEX'] = df_li['QTDE LI DECEX'].astype('int64')


# In[38]:


df_li.drop_duplicates('ID DI',keep='first', inplace=True)
df_li.info()


# In[39]:


df_li.head()


# ## 3.3 Dataset Dados Transito <a class="anchor" id="proc-tran"></a>

# ### 3.3.1 Transito Aduaneiro
# Indicador se houve transito aduaneiro na importação.

# In[40]:


df_tran['DATA TRANSITO'].describe()


# In[41]:


df_tran['TRANSITO'] = np.where(~df_tran['DATA TRANSITO'].str.conta-
ins('<N/D>',na=False),1,0)
df_tran['TRANSITO'].describe()


# In[43]:


df_tran['TRANSITO'].sum()


# ### 3.3.2 - Limpeza Dados
# Remoção colunas não utilizadas
```

```
# In[44]:


df_tran.columns


# In[45]:


df_tran.drop(['TIPO CONHECIMENTO', 'MODALIDADE DESPACHO', 'UL PRESENCA',
       'DESCRICAO UL', 'RA PRESENCA', 'TIPO DECLARACAO IMPORTACAO',
       'DESCRICAO RA', 'CANAL', 'DATA ATRACACAO', 'DATA TRANSITO',
       'DATA PRESENCA', 'DATA DEFERIMENTO', 'ANUÊNCIA', 'ANTES DEPOIS',
       'DATA EMISSAO', 'DT REGISTRO DI', 'ANTES DEPOIS2', 'DT DESEMBARACO DI',
       'DT ENTREGA'], axis='columns', inplace=True)
df_tran.info()


# In[56]:


### SERVIÇO REMOVER DO CODIGO PARA ENTREGA

#df_di.to_pickle("base/df_di.pkl")
#df_li.to_pickle("base/df_li.pkl")
#df_tran.to_pickle("base/df_tran.pkl")

df_di = pd.read_pickle("base/df_di.pkl")
df_li = pd.read_pickle("base/df_li.pkl")
df_tran = pd.read_pickle("base/df_tran.pkl")


# ## 3.4 União dos Datasets <a class="anchor" id="proc-join"></a>

# União datasets Declarações de Importação e Licenças de Importação

# In[48]:


df = pd.merge(df_di,df_li,how='left',left_on='ID DI',right_on='ID DI')
df.head(10)


# In[49]:


print('linhas inicio: ',df_di.shape[0])
print('linhas LI: ',df_li.shape[0])
print('linhas final: ',df.shape[0])
```

```
# União com dataset Transito

# In[50]:


df = pd.merge(df,df_tran,how='left',left_on='ID DI',right_on='ID DI')


# In[51]:


print('linhas transito: ',df_tran.shape[0])
print('linhas final: ',df.shape[0])


# In[78]:


### SERVIÇO REMOVER DO CODIGO PARA ENTREGA

#df.to_pickle("base/df.pkl")

df = pd.read_pickle("base/df.pkl")


# In[80]:


df.info()


# ## 3.5 Tratamento Dados Ausentes <a class="anchor" id="proc-miss"></a>

# In[57]:


msno.bar(df, color='royalblue',figsize=(16, 6))
plt.title("Dados Ausentes",fontsize = 24,pad=100)
plt.savefig('imagens/missing.png',bbox_inches='tight')


# ### 3.5.1 OEA
# As DIs com dados ausentes será considerada "NÃO OEA"

# In[58]:


df['OEA'].describe()
```

```
# In[59]:


plt.subplots(figsize=(12,6))
sns.countplot(x=df['OEA'])
plt.title("Dados Categóricos OEA",fontsize = 18,pad=30)
plt.savefig('imagens/missing-OEA-inicio.png',bbox_inches='tight')


# Imputando dados

# In[61]:


df['OEA'].fillna("NAO", inplace=True)


# Resultado Final

# In[62]:


total = len(df['OEA'])
plt.subplots(figsize=(12,6))
ax = sns.countplot(x=df['OEA'])
plt.title("Dados Categóricos OEA",fontsize = 18,pad=30)
for p in ax.patches:
    ax.annotate('{} - {:.1f}%'.format(p.get_height(),100*p.get_height()/to-
tal), (p.get_x()+0.31, p.get_height()+2000))
plt.savefig('imagens/missing-OEA-final.png',bbox_inches='tight')


# ### 3.5.2 Quantidades Dataset LI
# Nem todas as declarações de importação estão sujeitas a licenciamento de im-
portação, de forma que a quantidade dessas declarações provenientes do da-
taset LI é substancialmente menor do que do dataset das declarações de impor-
tação.
# O dataset LI contém colunas com quantidades de LI para cada orgão anu-
ente e a quantidade de LIs emitidas posteriormente ao registro da DI.
# Portanto, para todas as colunas provenientes daquele dataset imputare-
mos o valor 0 (zero) aos dados ausentes.

# In[63]:


df.fillna({'QTDE LI DECEX':0, 'QTDE LI INMETRO':0, 'QTDE LI ANVI-
SA':0, 'QTDE LI IBAMA':0, 'QTDE LI MAPA':0,
           'QTDE LI MCT':0, 'QTDE LI ANP':0, 'QTDE LI DPF':0,'QTDE LI CNEN':0,
 'QTDE LI CNPQ':0, 'QTDE LI DFPC':0,
           'QTDE LI BB':0, 'QTDE LI DEF POS REGISTRO':0}, inplace=True)
```

```
# ### 3.5.3 QUANTIDADE HORAS DISTRIBUIÇÃO e QUANTIDADE HORAS RECEPCAO
# Apenas declarações direcionadas para canal de conferência são distribuidas e têm a recepção de seus documentos. Portanto as demais declarações receberão o valor 0 aos dados ausentes.

# In[64]:


fig, ax1 = plt.subplots(figsize=(8,6))
cores = {"AMARELO": "yellow", "VERMELHO": "red", "VERDE":"green"}
sns.violinplot(x='CANAL', y='QTDE HORAS DISTRIBUICAO', data=df, palette=cores, showfliers = True, ax=ax1)
plt.title("Horas Distribuição por CANAL",fontsize = 18,pad=30 )
plt.savefig('imagens/missing-horasDist-inicial.png',bbox_inches='tight')


# In[66]:


fig, ax1 = plt.subplots(figsize=(8,6))
cores = {"AMARELO": "yellow", "VERMELHO": "red", "VERDE":"green"}
sns.violinplot(x='CANAL', y='QTDE HORAS RECEPCAO', data=df, palette=cores, showfliers = True, ax=ax1)
plt.title("Horas Recepção por CANAL", fontsize = 18,pad=30)
plt.savefig('imagens/missing-horasRecep-inicial.png',bbox_inches='tight')


# In[67]:


df.fillna({'QTDE HORAS RECEPCAO':0, 'QTDE HORAS DISTRIBUICAO':0}, inplace=True)


# ### 3.5.4 HORAS EXIGENCIA
# Novamente apenas declarações direcionadas para canal de conferência estão sujeitas a alguma exigência fiscal, porém nem todas serão submetidas. Isso explica a quantidade de registros inferior ao item anterior.
# Para os valores ausentes será imputado o valor 0.

# In[68]:


fig, ax1 = plt.subplots(figsize=(8,6))
cores = {"AMARELO": "yellow", "VERMELHO": "red", "VERDE":"green"}
sns.violinplot(x='CANAL', y='HORAS_EXIG', data=df, palette=cores, showfliers = True, ax=ax1)
plt.title("Horas Exigência por CANAL", fontsize = 18,pad=30)
```

```
plt.savefig('imagens/missing-horasExig-inicial.png',bbox_inches='tight')


# In[69]:


df['HORAS_EXIG'].fillna(0, inplace=True)


# In[2]:


### SERVIÇO REMOVER DO CODIGO PARA ENTREGA

#df.to_pickle("base/df_sem_ausentes.pkl")

df = pd.read_pickle("base/df_sem_ausentes.pkl")


# In[4]:


df.head()


# # 4 - ANÁLISE E EXPLORAÇÃO DOS DADOS <a class="anchor" id="aed"></a>

# In[3]:


# Profile do Dataframe
profile = pandas_profiling.ProfileReport(df, title='Profi-
ling TCC TRS',html={'style':{'full_width':True}})
profile.to_file(output_file="reports/tcc_df_report.html")
#profile.to_notebook_iframe() #mostra no jupyter


# In[4]:


df.columns


# In[72]:


g = sns.pairplot(df.drop(['ID DI','QTDE LI DECEX', 'QTDE LI INME-
TRO', 'QTDE LI ANVISA', 'QTDE LI IBAMA', 'QTDE LI MAPA',
    'QTDE LI MCT', 'QTDE LI ANP', 'QTDE LI DPF','QTDE LI CNEN', 'QTDE LI CNPQ'
, 'QTDE LI DFPC', 'QTDE LI BB',
```

```
       'QTDE LI DEF POS REGISTRO', 'TRANSITO'],axis='columns'))
g.fig.suptitle("Quantidades Horas",fontsize=30, y=1.08)
plt.savefig('imagens/pairplot-qtdHoras.png',bbox_inches='tight')


# In[75]:


g = sns.pairplot(df.drop(['ID DI', 'TIPO CE', 'OEA', 'MODALIDADE DESPACHO',
       'TIPO DECLARACAO IMPORTACAO', 'CANAL', 'HORAS_EXIG',
       'QTDE HORAS PRESENCA', 'QTDE HORAS DISTRIBUICAO',
       'QTDE HORAS RECEPCAO', 'DIA SEMANA', 'UNIDADE', 'RECINTO',
        'QTDE LI MCT', 'QTDE LI ANP', 'QTDE LI DPF',
       'QTDE LI CNEN', 'QTDE LI CNPQ', 'QTDE LI DFPC', 'QTDE LI BB',
       'QTDE LI DEF POS REGISTRO', 'TRANSITO'],axis='columns'))
g.fig.suptitle("Quantidades LIs",fontsize=30, y=1.08)
plt.savefig('imagens/pairplot-qtdLIs.png',bbox_inches='tight')


# In[77]:


g = sns.pairplot(df.drop(['ID DI', 'TIPO CE', 'OEA', 'MODALIDADE DESPACHO',
       'TIPO DECLARACAO IMPORTACAO', 'CANAL', 'HORAS_EXIG',
       'QTDE HORAS PRESENCA', 'QTDE HORAS DISTRIBUICAO',
       'QTDE HORAS RECEPCAO', 'DIA SEMANA', 'UNIDADE', 'RECINTO',
       'QTDE LI DECEX', 'QTDE LI INMETRO', 'QTDE LI ANVISA', 'QTDE LI IBAMA',
       'QTDE LI MAPA', 'QTDE LI MCT', 'QTDE LI ANP', 'QTDE LI DPF',
       'QTDE LI CNEN', 'QTDE LI CNPQ', 'QTDE LI DFPC', 'QTDE LI BB'],axis='co-
lumns'))
g.fig.suptitle("Quantidades Transito e LI Pós Registro",fontsize=18, y=1.08)
plt.savefig('imagens/pairplot-qtdTran.png',bbox_inches='tight')


# ## 4.1 - Variáveis Categóricas <a class="anchor" id="aed-cat"></a>

# Coluna|Nome|Tipo
# :-----:|:-----|:-------
#  1  | TIPO CE                   | 146937 non-null  object <br>
#  2  | OEA                       |146937 non-null  object <br>
#  3  | MODALIDADE DESPACHO       | 146937 non-null  object <br>
#  4  | TIPO DECLARACAO IMPORTACAO | 146937 non-null  object <br>
#  5  | CANAL                     | 146937 non-null  object <br>
#  11 | DIA SEMANA                | 146937 non-null  object <br>
#  12 | UNIDADE                   | 146937 non-null  object <br>
#  13 | RECINTO                   | 146937 non-null  object <br>
#  27 | TRANSITO                  | 146937 non-null  int32 <br>

# In[78]:
```

```
categoricas = df[['TIPO CE','OEA','MODALIDADE DESPACHO','TIPO DECLARACAO IM-
PORTACAO','CANAL','DIA SEMANA','UNIDADE','RECINTO','TRANSITO']]


# ### 4.1.1 - Aferição da Cardinalidade <a class="anchor" id="aed-card"></a>

# In[79]:


categoricas.nunique()


# Verificamos que há três varíáveis com valores constantes, e desta forma tor-
nam-se irrelevantes para os modelos

# In[80]:


categoricas['MODALIDADE DESPACHO'].value_counts()


# In[81]:


categoricas['TIPO DECLARACAO IMPORTACAO'].value_counts()


# In[82]:


categoricas.TRANSITO.value_counts()


# Descartando as colunas 'MODALIDADE DESPACHO', 'TIPO DECLARACAO IMPORTA-
CAO' e 'TRANSITO'

# In[83]:


df.drop([ 'MODALIDADE DESPACHO', 'TIPO DECLARACAO IMPORTACAO' , 'TRANSI-
TO'],axis='columns',inplace=True)


# ### 4.1.2 TIPO CE <a class="anchor" id="aed-ce"></a>

# O TIPO CE é uma variável que indica o se o Conhecimento de embar-
que é do tipo BL (Bill of Lading), de emissão do arma-
dor ou to do tipo HBL (House Bill of Lading) emitido por um agente de caragas.
```

```
# In[84]:


df['TIPO CE'].describe()


# In[88]:


total = len(df['TIPO CE'])*1.
fig, ax =plt.subplots(figsize=(10,6))
sns.countplot(x=df['TIPO CE'])
plt.title("Tipos Conhecimento de Carga", fontsize=18 )
for p in ax.patches:
    ax.annotate('{} - {:.1f}%'.format(p.get_height(),100*p.get_height()/to-
tal), (p.get_x()+0.31, p.get_height()+2000))
plt.savefig('imagens/tipoCE.png',bbox_inches='tight')


# In[89]:


fig, (ax1,ax2) = plt.subplots(1,2,sharey=False, figsize=(18,6))
plt.title("Horas Despacho por Tipo de Conhecimento")
sns.stripplot(x='TIPO CE', y='QTDE HORAS DESPACHO', data=df,  ax=ax1)
sns.boxplot(x='TIPO CE', y='QTDE HORAS DESPACHO', data=df, showfli-
ers = False, ax=ax2)
plt.savefig('imagens/tipoCE2.png',bbox_inches='tight')


# In[17]:


sns.displot(data=df[df['QTDE HORAS DESPACHO']<40], x="QTDE HORAS DESPA-
CHO", hue="TIPO CE", kind="kde")


# ### 4.1.3 OEA <a class="anchor" id="aed-oea"></a>

# Variável indica se importador possui certificação OEA (Operador Econô-
mico Autorizado)

# In[18]:


df['OEA'].describe()


# In[90]:
```

```
total = len(df['OEA'])*1.
fig, ax =plt.subplots(figsize=(10,6))
sns.countplot(x=df['OEA'])
plt.title("Operador Econômico Autorizado",fontsize=18)
for p in ax.patches:
    ax.annotate('{} - {:.1f}%'.format(p.get_height(),100*p.get_height()/to-
tal), (p.get_x()+0.31, p.get_height()+2000))
plt.savefig('imagens/OEA.png',bbox_inches='tight')


# In[91]:


fig, (ax1,ax2) = plt.subplots(1,2,sharey=False, figsize=(18,6))
plt.title("Horas Despacho por OEA")
sns.stripplot(x='OEA', y='QTDE HORAS DESPACHO', data=df,  ax=ax1)
sns.boxplot(x='OEA', y='QTDE HORAS DESPACHO', data=df, showfli-
ers = False, ax=ax2)
plt.savefig('imagens/OEA2.png',bbox_inches='tight')


# In[21]:


sns.displot(data=df[df['QTDE HORAS DESPACHO']<40], x="QTDE HORAS DESPA-
CHO", hue="OEA", kind="kde")


# ### 4.1.4 CANAL <a class="anchor" id="aed-canal"></a>

# Representa o canal de conferência a que a declaração de importação foi sub-
metida.

# In[22]:


df['CANAL'].describe()


# In[92]:


total = len(df['CANAL'])
cores = {"VERDE": "green", "VERMELHO": "red","AMARELO":"yellow"}
fig, ax =plt.subplots(figsize=(10,6))
sns.countplot(x=df['CANAL'], palette=cores)
plt.title("Canal de Conferência", fontsize=18)
for p in ax.patches:
```

```
    ax.annotate('{} - {:.1f}%'.format(p.get_height(),100*p.get_height()/to-
tal), (p.get_x()+0.31, p.get_height()+2000))
plt.savefig('imagens/canal.png',bbox_inches='tight')


# In[97]:


fig, (ax1,ax2) = plt.subplots(1,2,sharey=False, figsize=(18,6))
plt.title("Horas Despacho por Canal de Conferência")
cores = {"VERDE": "green", "VERMELHO": "red","AMARELO":"yellow"}
sns.stripplot(x='CANAL', y='QTDE HORAS DESPACHO',palette=co-
res, data=df,  ax=ax1)
sns.boxplot(x='CANAL', y='QTDE HORAS DESPACHO', data=df, palette=cores,sho-
wfliers = False, ax=ax2)
plt.savefig('imagens/canal2.png',bbox_inches='tight')


# In[25]:


cores = {"VERDE": "green", "VERMELHO": "red","AMARELO":"yellow"}
sns.displot(data=df[(df['QTDE HORAS DESPACHO']<400) & (df['CANAL']=='VER-
DE')], x="QTDE HORAS DESPACHO",palette=cores ,hue="CANAL", kind="kde")


# In[26]:


sns.displot(data=df[(df['QTDE HORAS DESPACHO']<400) & (df['CANAL']!='VER-
DE')], x="QTDE HORAS DESPACHO",palette=cores ,hue="CANAL", kind="kde")


# ### 4.1.5 DIA SEMANA <a class="anchor" id="aed-dia"></a>

# Representa o dia da semana em que foi registrada a declaração de importação

# In[27]:


df['DIA SEMANA'].describe()


# In[94]:


total = len(df['DIA SEMANA'])
fig, ax =plt.subplots(figsize=(14,6))
sns.countplot(x=df['DIA SEMANA'])
plt.title("Dia da Semana de Registro da DI",fontsize=18)
```

```
for p in ax.patches:
    ax.annotate('{} - {:.1f}%'.format(p.get_height(),100*p.get_height()/to-
tal), (p.get_x()+.05, p.get_height()+500))
plt.savefig('imagens/semana.png',bbox_inches='tight')


# In[95]:


fig, ax = plt.subplots(figsize=(18,6))
plt.title("Horas Despacho por Dia da Semana")
sns.stripplot(x='DIA SEMANA', y='QTDE HORAS DESPACHO', data=df)


# In[96]:


fig, ax = plt.subplots(figsize=(18,6))
plt.title("Horas Despacho por Dia da Semana", fontsize=18)
sns.boxplot(x='DIA SEMANA', y='QTDE HORAS DESPACHO', data=df, showfli-
ers = False)
plt.savefig('imagens/semana2.png',bbox_inches='tight')


# In[31]:


sns.displot(data=df[df['QTDE HORAS DESPACHO']<100], x="QTDE HORAS DESPA-
CHO" ,hue="DIA SEMANA", kind="kde",height=6,aspect=2 )


# ### 4.1.6 UNIDADE <a class="anchor" id="aed-unidade"></a>

# In[32]:


df['UNIDADE'].describe()


# In[33]:


df['UNIDADE'].value_counts()


# ### 4.1.7 RECINTO <a class="anchor" id="aed-recinto"></a>

# In[34]:
```

```
df['RECINTO'].describe()


# In[35]:


df['RECINTO'].value_counts()


# ## 4.2 Variáveis Quantitativas <a class="anchor" id="aed-quant"></a>

# In[105]:


df.columns


# In[106]:


quantitativas = df[['HORAS_EXIG', 'QTDE HORAS DESPACHO','QTDE HORAS PRESEN-
CA', 'QTDE HORAS DISTRIBUICAO',
                    'QTDE HORAS RECEPCAO','QTDE LI DECEX', 'QTDE LI INME-
TRO','QTDE LI ANVISA', 'QTDE LI IBAMA',
                    'QTDE LI MA-
PA', 'QTDE LI MCT','QTDE LI ANP', 'QTDE LI DPF', 'QTDE LI CNEN', 'QTDE LI CNPQ
',
                    'QTDE LI DFPC', 'QTDE LI BB', 'QTDE LI DEF POS REGISTRO']]


# In[107]:


quantitativas.describe()


# In[108]:


quantitativas.median()


# In[109]:


quantitativas.mode()


# In[110]:
```

```
quantitativas.corr()


# In[111]:


fig, ax = plt.subplots(figsize=(18,10))
sns.heatmap(quantitativas.corr(), annot=True)
plt.title("HEATMAP Correlação Variáveis Quantitativas", fontsize = 18,pad=30)
plt.savefig('imagens/heatmap_corr.png',bbox_inches='tight')


# In[45]:


### SERVIÇO REMOVER DO CODIGO PARA ENTREGA

#df.to_pickle("base/df_pronto_modelos.pkl")

df = pd.read_pickle("base/df_pronto_modelos.pkl")


# # 5 - CRIAÇÃO DE MODELOS DE MACHINE LEARNING <a class="anchor" id="mod"></a>

# Conversão One-Hot para variáveis categóricas de baixa cardinalidade

# In[46]:


dum_df = pd.get_dummies(df, columns=['TIPO CE','OEA','CANAL','DIA SEMA-
NA'], prefix=['TIPO CE','OEA','CANAL','DIA SEMANA'] )


# Label Encoding para variáveis categóricas de alta cardinalidade

# In[47]:


from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
dum_df['UNIDADE_ENC'] = le.fit_transform(dum_df['UNIDADE'])
dum_df['RECINTO_ENC'] = le.fit_transform(dum_df['RECINTO'])


# In[48]:


dum_df.head()
```

```
# In[49]:


dum_df.drop(['UNIDADE','RECINTO'],axis='columns',inplace=True)


# Separação do dataframe em 'treino' e 'teste'

# In[50]:


x_train, x_test, y_train, y_test = train_test_split(dum_df.drop(['QTDE HO-
RAS DESPACHO','ID DI'], axis=1), dum_df['QTDE HORAS DESPA-
CHO'],test_size=0.3,random_state=93)


# ## 5.1 - Decision Tree Regressor <a class="anchor" id="mod-dtree"></a>

# In[51]:


from sklearn.tree import DecisionTreeRegressor


# In[52]:


dtr = DecisionTreeRegressor(random_state=42)
dtr.fit(x_train,y_train)


# In[53]:


dtr.score(x_test,y_test)


# In[55]:


pred_dtr = dtr.predict(x_test)


# In[57]:


skmetrics.mean_absolute_error(y_test,pred_dtr)


# Busca de melhores hiperparâmetros
```

```
# In[58]:


dtr1 = DecisionTreeRegressor(random_state=42)


# In[59]:


parametros_dtr={'max_depth':[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15],
                'min_samples_split':[2,3,4,5,6,7,8,9,10],
               }


# In[60]:


grid_dtr = GridSearchCV(estimator=dtr1, param_grid=parame-
tros_dtr,cv=10,n_jobs=-1)


# In[123]:


grid_dtr.fit(x_train,y_train)


# In[124]:


grid_dtr.best_params_


# In[125]:


grid_dtr.best_score_


# Modelo Final

# In[126]:


dtr_final = DecisionTreeRegressor(random_state=42,max_depth=8, min_sam-
ples_split=3)
dtr_final.fit(x_train,y_train)
dtr_final.score(x_test,y_test)


# In[128]:
```

```
pred_dtr_final = dtr_final.predict(x_test)


# Resultados

# In[129]:


metrics.mean_absolute_error(y_test,pred_dtr_final)


# In[135]:


fig, ax = plt.subplots(figsize=(8,6))
rpv = ResidualsPlot(dtr_final)
rpv.fit(x_train,y_train)
rpv.score(x_test,y_test)
rpv.poof()
fig.savefig("imagens/residuos_dtr.png",bbox_inches='tight')


# ## 5.2 - RandomForest Regressor <a class="anchor" id="mod-rfor"></a>

# In[61]:


from sklearn.ensemble import RandomForestRegressor


# In[62]:


rfr = RandomForestRegressor(random_state=42, n_estimators=1000)
rfr.fit(x_train,y_train)


# In[63]:


rfr.score(x_test,y_test)


# In[64]:


pred_rfr = rfr.predict(x_test)
```

```
# In[65]:


skmetrics.mean_absolute_error(y_test,pred_rfr)


# Busca de melhores hiperparâmetros

# In[53]:


rfr1 = RandomForestRegressor(random_state=42, n_estimators=100)


# In[54]:


parametros_rfr = {
                  'max_depth':[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15],
                  'min_samples_split':[2,3,4,5,6,7,8,9,10],
                  'min_samples_leaf':[1,2,4,6,8,10],
                  'bootstrap': [False, True]
                 }


# In[55]:


grid_rfr = GridSearchCV(estimator=rfr1, param_grid=parame-
tros_rfr,cv=10,n_jobs=-1)


# In[56]:


grid_rfr.fit(x_test,y_test)


# In[57]:


grid_rfr.best_params_


# In[58]:


grid_rfr.best_score_


# Modelo Final
```

```
# In[21]:


rfr_final = RandomForestRegressor(random_state=42, n_estimators=1000, boo-
tstrap=True, max_depth=15,min_samples_leaf=2,min_samples_split=10)
rfr_final.fit(x_train,y_train)
rfr_final.score(x_test,y_test)


# Resultados

# In[22]:


pred_rfr_final = rfr_final.predict(x_test)


# In[23]:


metrics.r2_score(y_test,pred_rfr_final)


# In[24]:


metrics.mean_absolute_error(y_test,pred_rfr_final)


# In[26]:


fig, ax = plt.subplots(figsize=(8,6))
rpv = ResidualsPlot(rfr_final)
rpv.fit(x_train,y_train)
rpv.score(x_test,y_test)
rpv.poof()
fig.savefig("imagens/residuos_rfr.png",bbox_inches='tight')


# ## 5.3 - CatBoost Regressor <a class="anchor" id="mod-catb"></a>

# In[27]:


from catboost import CatBoostRegressor, Pool, metrics, cv


# In[15]:
```

```
cbr_X = df.drop(['QTDE HORAS DESPACHO','ID DI'], axis=1)
cbr_y = df['QTDE HORAS DESPACHO']
cbr_X_train, cbr_X_validation, cbr_y_train, cbr_y_valida-
tion = train_test_split(cbr_X, cbr_y, train_size=0.75, random_state=42)


# In[16]:


cbr_X.info()


# In[29]:


cbr = CatBoostRegressor()


# In[30]:


cbr.fit(
    cbr_X_train, cbr_y_train,
    cat_features=[ 0,  1,  2, 7, 8, 9 ],
    eval_set=(cbr_X_validation, cbr_y_validation),
    plot=False
);


# In[31]:


cbr.score(cbr_X_validation,cbr_y_validation)


# In[35]:


pred_cbr_final = cbr.predict(cbr_X_validation)


# In[37]:


metrics.mean_absolute_error(cbr_y_validation,pred_cbr_final)


# GRID SEARCH
```

```
# In[118]:


grid = {'learning_rate': [0.03, 0.1, 0.2, 0.3, 0.4],
        'depth': [4, 6, 10],
        'l2_leaf_reg': [1, 3, 5, 7, 9]}


# In[120]:


for l in grid['learning_rate']:
    for d in grid['depth']:
        for lr in grid['l2_leaf_reg']:
            cbr_grid = CatBoostRegressor(learning_rate=l,de-
pth=d,l2_leaf_reg=lr)
            cbr_grid.fit(cbr_X_train, cbr_y_train,cat_featu-
res=[ 0,  1,  2, 7, 8, 9 ],eval_set=(cbr_X_validation, cbr_y_valida-
tion),plot=False,verbose=False)
            score = cbr_grid.score(cbr_X_validation,cbr_y_validation)
            cbr_y_validation_predict = cbr_grid.predict(cbr_X_validation)
            mae = metrics.mean_absolute_error(cbr_y_validation,cbr_y_valida-
tion_predict)
            print('learning_rate: ',l,' - de-
pth: ',d,' - l2_leaf_reg: ',lr,' - score: ',score,' - MAE: ',mae)


# Modelo Final

# In[17]:


cbr_final = CatBoostRegressor(learning_rate=0.1,depth=4,l2_leaf_reg=3)


# In[18]:


cbr_final.fit(
    cbr_X_train, cbr_y_train,
    cat_features=[ 0,  1,  2, 7, 8, 9  ],
    eval_set=(cbr_X_validation, cbr_y_validation),
    plot=False,
    verbose=False
);


# In[19]:
```

```
cbr_final.score(cbr_X_validation, cbr_y_validation)


# In[20]:


cbr_y_validation_predict = cbr_final.predict(cbr_X_validation)


# Resultados

# In[23]:


skmetrics.r2_score(cbr_y_validation,cbr_y_validation_predict)


# In[24]:


skmetrics.mean_absolute_error(cbr_y_validation,cbr_y_validation_predict)


# In[25]:


plt.subplots(figsize=(12,6))
sorted_feature_importance = cbr_final.feature_importances_.argsort()
plt.barh(cbr_X.columns[sorted_feature_importance],
        cbr_final.feature_importances_[sorted_feature_importance],
        color='royalblue')
plt.xlabel("CatBoost Feature Importance")
plt.title("Importância das Variáveis", fontsize = 18,pad=30)
plt.savefig('imagens/feature_importance.png',bbox_inches='tight')


# # Usando a <i>feature Importance</i> na seleção de variáveis

# In[6]:


from catboost import CatBoostRegressor, Pool, metrics, cv


# In[7]:


df2 = df.copy()


# Agrupamento de variáveis com as Quanitdades de LIs
```

```
# In[9]:


df2['QTDE LIS']=df2['QTDE LI DECEX']+df2['QTDE LI INMETRO']+df2['QTDE LI ANVI-
SA']+df2['QTDE LI IBAMA']+df2['QTDE LI MA-
PA']+df2['QTDE LI MCT']+df2['QTDE LI ANP']+df2['QTDE LI DPF']+df2['QTDE LI CNE
N']+df2['QTDE LI CNPQ']+df2['QTDE LI DFPC']+df2['QTDE LI BB']


# In[13]:


df2[['QTDE LIS','QTDE HORAS DESPACHO']].corr()


# In[28]:


fig, ax = plt.subplots(figsize=(18,10))
sns.heatmap(df2[['QTDE HORAS DESPACHO','HORAS_EXIG', 'QTDE HORAS PRESEN-
CA', 'QTDE HORAS DISTRIBUICAO', 'QTDE HORAS RECEPCAO','RECIN-
TO','QTDE LIS' , 'QTDE LI DEF POS REGISTRO']].corr(), annot=True)
plt.title("HEATMAP Correlação Variáveis Quantitativas Após Agrupamento", font-
size = 18,pad=30)
plt.savefig('imagens/heatmap_corr_agrupa.png',bbox_inches='tight')


# Criação do modelo com a implementação padrão

# In[30]:


cbr_X = df2.drop(['QTDE HORAS DESPACHO','ID DI'], axis=1)
cbr_y = df2['QTDE HORAS DESPACHO']
cbr_X_train, cbr_X_validation, cbr_y_train, cbr_y_valida-
tion = train_test_split(cbr_X, cbr_y, random_state=42)


# In[31]:


variaveis = ['TIPO CE', 'OEA', 'CANAL', 'HORAS_EXIG',
       'QTDE HORAS PRESENCA', 'QTDE HORAS DISTRIBUICAO', 'QTDE HORAS RE-
CEPCAO',
       'DIA SEMANA', 'UNIDADE', 'RECINTO','QTDE LIS' , 'QTDE LI DEF POS REGIS-
TRO']
variaveis_cat = ['TIPO CE','OEA','CANAL','DIA SEMANA', 'UNIDADE', 'RECINTO']
```

```
# In[32]:


train_pool = Pool(cbr_X_train[variaveis], cbr_y_train, cat_features=varia-
veis_cat)
test_pool = Pool(cbr_X_validation[variaveis], cbr_y_validation, cat_featu-
res=variaveis_cat)


# In[33]:


model = CatBoostRegressor()


# In[34]:


model.fit(train_pool, eval_set=test_pool, verbose=False)


# In[35]:


model.score(test_pool)


# In[36]:


cbr_y_validation_predict = model.predict(cbr_X_validation)


# In[37]:


skmetrics.mean_absolute_error(cbr_y_validation,cbr_y_validation_predict)


# Busca Hiperparâmetros

# In[40]:


model_grid = CatBoostRegressor()
grid = {'iterations': [100, 150, 200],
        'learning_rate': [0.03, 0.1, 0.2, 0.3, 0.4],
        'depth': [4, 6, 10],
        'l2_leaf_reg': [1, 3, 5, 7, 9]}
model_grid.grid_search(grid, train_pool,cv=10,verbose=False)
```

```
# In[41]:


pred = model_grid.predict(cbr_X_validation)


# In[42]:


model_grid.score(test_pool)


# In[43]:


skmetrics.mean_absolute_error(cbr_y_validation,pred)


# In[72]:


plt.subplots(figsize=(12,6))
sorted_feature_importance = model_grid.feature_importances_.argsort()
plt.barh(cbr_X[variaveis].columns[sorted_feature_importance],
        model_grid.feature_importances_[sorted_feature_importance],
        color='royalblue')
plt.xlabel("CatBoost Feature Importance")
plt.title("Importância das Variáveis Final", fontsize = 18,pad=30)
plt.savefig('imagens/feature_importance_final.png',bbox_inches='tight')


# In[78]:


explainer = shap.TreeExplainer(model_grid)
shap_values = explainer.shap_values(cbr_X_validation[variaveis])


# In[86]:


fig, ax = plt.subplots(figsize=(12,6))
shap.summary_plot(shap_values, cbr_X_validation[variaveis],show=False )
plt.title("Impacto na variável alvo", fontsize = 18,pad=30)
plt.savefig('imagens/impacto.png',bbox_inches='tight')


# In[ ]:
```

```
# In[22]:


model_grid.eval_metrics(test_pool,['R2','MAE'], plot=True, )


# In[42]:


final = cbr_y_validation.to_frame()


# In[43]:


final['PREVISÃO'] = pred


# In[44]:


final.head()


# In[92]:


fig, ax = plt.subplots(figsize=(8,8))
model = wrap(model_grid)
pev = PredictionError(model)
pev.fit(cbr_X_train[variaveis], cbr_y_train)
pev.score(cbr_X_validation[variaveis], cbr_y_validation)
plt.title("Erros na predição", fontsize = 18,pad=30)
plt.savefig('imagens/erro.png',bbox_inches='tight')


# In[93]:


import types
def imports():
    for name, val in globals().items():
        if isinstance(val, types.ModuleType):
            yield val.__name__
list(imports())


# In[94]:
```

```
import pkg_resources
import types
def get_imports():
    for name, val in globals().items():
        if isinstance(val, types.ModuleType):
            # Split ensures you get root package,
            # not just imported function
            name = val.__name__.split(".")[0]

        elif isinstance(val, type):
            name = val.__module__.split(".")[0]

        # Some packages are weird and have different
        # imported names vs. system/pip names. Unfortunately,
        # there is no systematic way to get pip names from
        # a package's imported name. You'll have to add
        # exceptions to this list manually!
        poorly_named_packages = {
            "PIL": "Pillow",
            "sklearn": "scikit-learn"
        }
        if name in poorly_named_packages.keys():
            name = poorly_named_packages[name]

        yield name
imports = list(set(get_imports()))

# The only way I found to get the version of the root package
# from only the name of the package is to cross-check the names
# of installed packages vs. imported packages
requirements = []
for m in pkg_resources.working_set:
    if m.project_name in imports and m.project_name!="pip":
        requirements.append((m.project_name, m.version))

for r in requirements:
    print("{}=={}".format(*r))


# In[95]:


pip freeze


# In[96]:
```

```
from platform import python_version

print(python_version())


# In[ ]:
```

Tabelas

Ambiente de desenvolvimento

Biblioteca usada	Versão
Catboost	0.26
Matplotlib	3.3.2
Missingno	0.4.2
Numpy	1.19.5
Pandas	1.1.3
Pandas_profiling	3.0.0
Python	3.8.5
Scikit Learn	0.23.2
Seaborn	0.11.1
Shap	0.39.0
Yellowbrick	1.3.post1