PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Inteligência Artificial e Aprendizado de Máquina

Felipe Israel Corrêa

INTELIGÊNCIA ARTIFICIAL PARA PREDIÇÃO DE ACIDENTES EM RODOVIAS FEDERAIS

Januária
Outubro de 2022

Felipe Israel Corrêa

INTELIGÊNCIA ARTIFICIAL PARA PREDIÇÃO DE ACIDENTES EM RODOVIAS FEDERAIS

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Inteligência Artificial e Aprendizado de Máquina, como requisito parcial à obtenção do título de *Especialista*.

Januária

Outubro de 2022

SUMÁRIO

1. Introdução

No Brasil em 18 de novembro de 2011, foi sancionada a lei nº 12.527 que trata do acesso à informação dos dados gerados pela União, Estados, Distrito Federal e Municípios (PLANALTO, 2022). Em resumo essa lei trata da disponibilização das informações geradas pelos órgãos públicos a qualquer cidadão.

Essa iniciativa por parte do governo fez com que inúmeros projetos para análise dessas informações começassem a aparecer. Um desses projetos, e um dos mais conhecidos é o Serenata de Amor que tem por objetivo, conforme definem-se, fiscalizar gastos públicos e compartilhar as informações de forma acessível a qualquer pessoa (SERENATA DE AMOR, 2022).

Dessa forma como esse e tantos outros projetos conseguem gerar conhecimento por meio de uma quantidade tão grande de dados e que, na grande maioria, são impossíveis de serem analisados mentalmente?

É para esse tipo de problema que o uso de Aprendizado de Máquina, técnica de computação baseada em Inteligência Artificial passou a ser utilizada. Desenvolvida a partir da década de 70 esse campo de estudo foi se popularizando com o aumento da capacidade de processamento computacional e é utilizado nas mais diversas áreas, seja para predição, agrupamento, reconhecimento de padrões, etc.

Este estudo pretende utilizar-se de dados abertos, referentes a acidentes em rodovias federais. Processando-os por meio de técnicas de aprendizado de máquina pretende-se gerar uma predição quanto a possíveis ocorrências, circunstâncias e locais mais prováveis de acidentes. Dessa forma órgãos responsáveis pelo atendimento imediato a possíveis vítimas, podem se programar e ajustar quantidade de insumos necessários para fornecer atendimento adequado e de qualidade.

2. Descrição do Problema e da Solução Proposta

De acordo com dados do Departamento Nacional de Trânsito (Denatran) o Brasil chegou em 2019 a uma frota de 45,4 milhões de automóveis, ou seja, 1 veículo a cada 4,4 habitantes (DENATRAN, 2022).

Observado esse número, outro que é bastante expressivo, se refere a quantidade de acidentes de trânsito registrados em rodovias federais, que de acordo com o painel CNT de Acidentes Rodoviários, disponibilizado pela Confederação

Nacional do Transporte (CNT), são em média mais 50 mil acidentes com vítimas por ano. Só em 2021 foram 52762 acidentes com vítimas, sendo 5391 fatais (CNT, 2022).

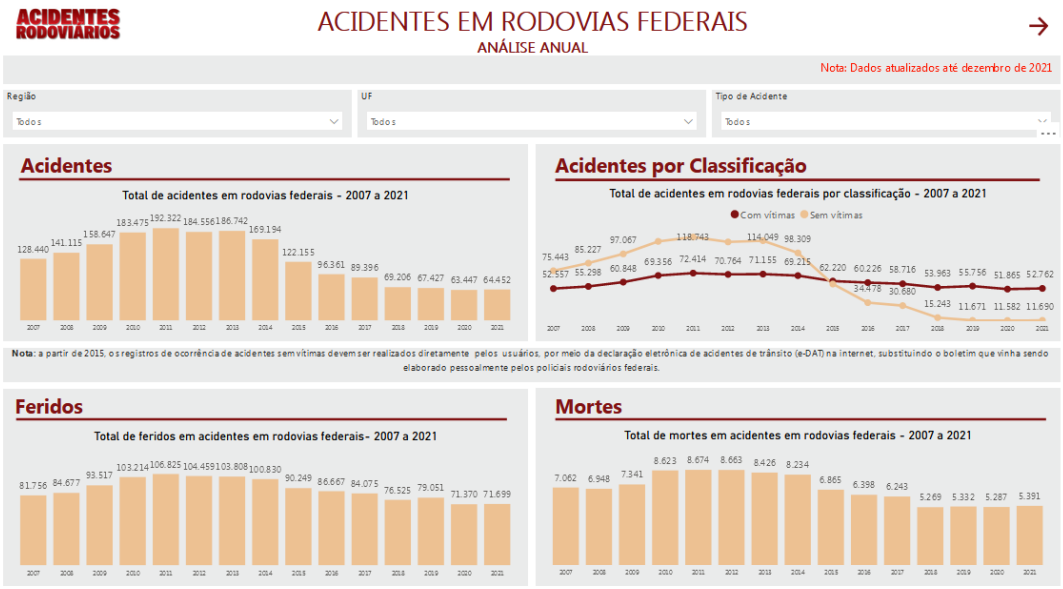

FIGURA 1: Acidentes em rodovias federais
Fonte: https://www.cnt.org.br/painel-acidente

Dado esse contexto o projeto se utiliza das bases públicas de ocorrências disponibilizadas pela Polícia Rodoviária Federal (PRF) com objetivo de criar uma aplicação que preveja a quantidade de acidentes para um determinado trecho da rodovia. Com esses dados, as equipes da própria PRF, Bombeiros e SAMU podem se preparar previamente para atender as chamadas aumentando assim as chances de uma mortalidade menor.

Para a criação da aplicação serão gerados 2 modelos de aprendizado supervisionado de máquina resolvendo um problema de regressão. Um deles utilizará o algoritmo de árvore de regressão enquanto o outro fará uso de uma rede neural. Ao final serão comparados e o que apresentar o melhor resultado será escolhido, mas caso ocorra um empate, ou seja, uma diferença pouco significativa no resultado, o de menor complexidade será selecionado.

3. Canvas Analítico

Questão	É possível prever a localização de onde ocorrem mais acidentes em rodovias federais dependendo do dia e do clima?

Bases de Dados	Para a resposta à questão buscar os dados de ocorrências de acidentes em rodovias federais e que constem o dia, quantidade de vítimas, clima e localização.
Heurísticas	No processamento dos dados efetuar uma checagem mais rigorosa nos dados que, a princípio, ajudarão na resposta à questão. São os mesmos elencados no passo Bases de Dados.
Validação	A entrega deve ser apresentada da maneira mais visual e didática. Ao observarem os resultados os usuários devem gastar o mínimo de tempo para entenderem a proposta.
Implementação	Para implementação utilizar as seguintes bibliotecas, todas para linguagem Python: • Scikit-learn (árvore de regressão) • TensorFlow (rede neural) • Pandas (manipulação de dados) • Searborn (visualização de dados) • Statsmodels (estatística)
Resultados	Preencher após Análise e Exploração de Dados
Próximos Passos	Preencher após entrega dos resultados

4. Coleta de Dados

Para a elaboração do projeto, foram utilizados os dados públicos de ocorrência por acidente, no período de 2007 a 2021, fornecidas pelo site da PRF (PRF, 2022).

As bases em formato CSV estão separadas por ano, totalizando 15 bases, e compactadas no formato rar (de 2007 a 2016)* e no formato zip (de 2017 a 2021), abrangendo todas as ocorrências sucedidas em rodovias federais.

O dicionário de dados foi preenchido com as descrições disponibilizadas pelo própria PRF (PRF, 2022).

Nome do dataset: datatran<ano>.csv **Descrição: Ocorrência por acidente** **Link:**https://www.gov.br/prf/pt-br/acesso-a-informacao/dados-abertos/dados-abertos-acidentes		
Nome do Atributo	**Descrição**	**Tipo**
id	Variável com valores numéricos, representando o identificador do acidente	Inteiro
data_inversa	Data da ocorrência	Texto
dia_semana	Dia da semana da ocorrência: segunda, terça, etc	Texto
horario	Horário da ocorrência	Texto
uf	Unidade da Federação: MG, PE, DF, etc	Texto
br	Variável com valores numéricos representando o identificador da BR do acidente	Inteiro
km	Identificação do quilômetro onde ocorreu o acidente, com valor mínimo de 0,1 Km e com a casa decimal separada por ponto	Decimal

municipio	Nome do município de ocorrência do acidente	Texto
causa_acidente	Identificação da causa presumível do acidente: falta de atenção, velocidade incompatível	Texto
tipo_acidente	Identificação do tipo de acidente: colisão frontal, saída de pista, etc	Texto
classificação_acidente	Classificação quanto à gravidade do acidente: sem vítimas, com vítimas feridas, com vítimas fatais ou ignorado	Texto
fase_dia	Fase do dia no momento do acidente: amanhecer, pleno dia, etc	Texto
sentido_via	Sentida da via considerando o ponto de colisão: crescente ou decrescente	Texto
condição_meteorologica	Condição meteorológica no momento do acidente: céu claro, chuva, vento, etc	Texto
tipo_pista	Tipo da pista considerando a quantidade de faixas: dupla, simples ou múltipla	Texto
tracado_via	Descrição do traçado da via: reta, curva ou cruzamento	Texto

uso_solo	Descrição sobre as características do local do acidente: Urbano=Sim ou Rural=Não	Texto
ano	Ano da ocorrência	Inteiro
pessoas	Total de pessoas envolvidas na ocorrência	Inteiro
mortos	Total de pessoas mortas envolvidas na ocorrência	Inteiro
feridos_leves	Total de pessoas com ferimentos leves envolvidas na ocorrência	Inteiro
feridos_graves	Total de pessoas com ferimentos graves envolvidas na ocorrência	Inteiro
ilesos	Total de pessoas ilesas envolvidas na ocorrência	Inteiro
ignorados	Total de pessoas envolvidas na ocorrência e que não se soube o estado físico	Inteiro
feridos	Total de pessoas feridas envolvidas na ocorrência, total de feridos leves e graves	Inteiro
veiculos	Total de veículos envolvidos na ocorrência	Inteiro
latitude	Latitude do local do acidente	Decimal
longitude	Longitude do local do acidente	Decimal
regional	-	Texto
delegacia	-	Texto

uop	-	Texto

*Observações: De 2007 a 2016 não constam nas bases os campos de latitude, longitude, regional, delegacia e uop.

5. Processamento/Tratamento de Dados

O tratamento dos dados das bases foi desenvolvido utilizando-se a linguagem *Python* e as bibliotecas Pandas (manipulação de dados), *Seaborn* e *Missingno* (visualização de dados).

Conforme apontado na parte 4 desse documento, as bases de acidentes são disponibilizadas por ano de ocorrência, portanto o primeiro passo para tratamento foi concatená-las em um único conjunto de dados, conforme apresentado no código abaixo:

```
# Datasets concatenation

datasets = [f'/content/drive/MyDrive/PosPuc/TCC/datasets/acidentes_ocorrencia/datatran{i}.csv' for i in range(2007, 2022)]
df_geral = pd.DataFrame()

for data in datasets:
    df_temp = pd.read_csv(data, sep=';', encoding='utf-8')
    df_geral = pd.concat([df_geral, df_temp], ignore_index=True)
```

FIGURA 2: Concatenação das bases
Fonte: O Autor

Após a concatenação o conjunto de dados resultante ficou com 1910652 linhas e 31 atributos (colunas). Conforme a imagem de saúde da base abaixo, nota-se uma grande quantidade de dados faltantes, principalmente por serem colunas, conforme observação na parte 4 desse documento, passaram a ser catalogadas a partir de 2017:

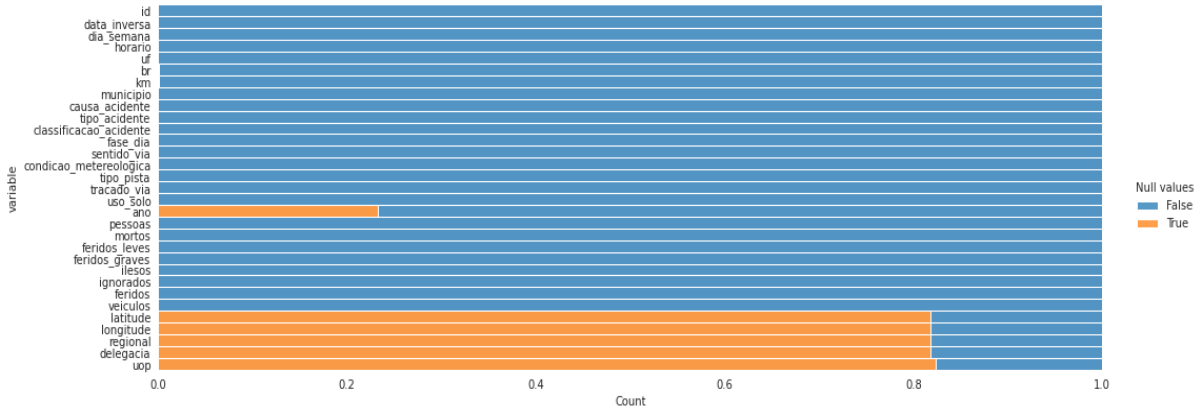

FIGURA 3: Porcentagem de dados nulos por variável
Fonte: O Autor

Depois de realizada uma análise visual dos atributos da base, principalmente os de tipo texto, foram elencados alguns tratamentos necessários:

- **Codificação dos dados:**
  - Conversão do atributo id de inteiro para texto, pois se trata de um dado qualitativo que somente expressa o número da ocorrência. Essa variável será utilizada para contagem da quantidade de ocorrências.
  - Criação de um novo atributo chamado km_intervalo composto pelo intervalo de 50km do atributo km. Exemplo: se o valor de km é igual a 67, ele deve pertencer ao km_intervalo 50 – 100. O objetivo desse novo atributo visa estabelecer uma quantidade menor de *dummy variables* a ser utilizada pelo modelo de aprendizado de máquina.
  - Converter o atributo br de inteiro para texto por se tratar de uma variável qualitativa.
  - Truncar o valor da variável km para buscar melhorar a distribuição dos valores e utilizá-lo para criação de um novo.
  - Ajustar o campo data_inversa para aaaa-mm-dd*.
  - Remover o texto “-feira” do atributo dia_semana e maiusculizar o restante da palavra*.
  - Os atributos latitude e longitude estavam usando o separador decimal ‘,’ então foi necessária a substituição para o separador decimal ‘.’ que é o utilizado pela linguagem de programação que também foram convertidos para tipo decimal.
- **Tratamento de dados ausentes:**
  - Preencher os dados nulos dos atributos regional, delegacia e uop com a adição do texto UNKNOWN.
  - Preencher valores nulos para evitar erro, remover a acentuação e maiusculizar os atributos dia_semana, causa_acidente, tipo_acidente, classificação_acidente, fase_dia, condição_metereologica, tipo_pista, tracado_via e uso_solo*.
  - Preencher os campos nulos da variável ano, com o ano respectivo e converter para o tipo inteiro*.
  - Os mesmos atributos latitude e longitude tiveram os seus valores nulos preenchidos pela média da própria variável latitude ou longitude vinda

do agrupamento dos atributos br e km_interval, conforme o código abaixo. Essa tratativa visa a utilização futura na Análise e Exploração dos Dados.

```
# Fill latitude and longitude
df_temp = (
    df
    .groupby(['br', 'km_intervalo'], as_index=False)
    .agg({'longitude': 'mean', 'latitude': 'mean'})
).rename(columns={'longitude': 'temp_long', 'latitude': 'temp_lati'}, inplace=True)

# Joining tables
df = df.merge(df_temp, on=['br', 'km_intervalo'], how='left')

df = (
    df
    .assing(longitude = df.apply(lambda x: x.temp_long if (np.isnan(x.longitude) and not np.isnan(x.temp_long)) else 0, axis=1))
    .assing(latitude = df.apply(lambda x: x.temp_lati if (np.isnan(x.latitude) and not np.isnan(x.temp_lati)) else 0, axis=1))
)

# Remove unnecessary columns
df.drop(columns=['temp_long', 'temp_lati'], inplace=True)
```

FIGURA 4: Tratamento atributos latitude e longitude.
Fonte: O Autor

- **Dados duplicados:**
  - Ao analisar o campo id, observou-se a ocorrência de duplicidade, portanto foi necessária a remoção da duplicação levando em consideração o ano, pois existem números de id iguais em anos diferentes. O código abaixo apresenta relata ocorrência de 35 casos do tipo e executa a remoção deles.

```
# Check ID duplicity
id_duplicity = df.groupby(['id', 'ano'], as_index=False).agg({'id': ['first', 'count'], 'ano': 'first'})
ids_duplicity = id_duplicity[id_duplicity[('id', 'count')] > 1][('id', 'first')].values.tolist()
print(f' Was found: {len(ids_duplicity)} duplicate ids')

# Remove duplicity
df.drop_duplicates(subset=['id', 'ano'], keep='first', inplace=True)
```

```
 Was found: 35 duplicate ids
```

FIGURA 5: Duplicidade atributo id.
Fonte: O Autor

- **Adição de atributos:**
  - Adição do atributo mes para possibilidade de utilização como atributo no modelo de aprendizado de máquina.

Finalizado o tratamento do conjunto de dados com um total de 1910617 linhas e 33 atributos.

Foram adicionadas imagens somente dos tratamentos que apresentaram uma maior complexidade para serem escritos, mas todo o código e análises estão

disponíveis para consulta e reprodução em https://github.com/ficorrea/courses/tree/main/pos_graducacao_puc/TCC/project.

*Esses tratamentos serviram para estabelecer um padrão para os atributos em todo o conjunto de dados.

6. Análise e Exploração dos Dados

Essa etapa é iniciada com a observação da quantidade de ocorrências por determinados períodos. Nas imagens abaixo temos as ocorrências por ano e por mês:

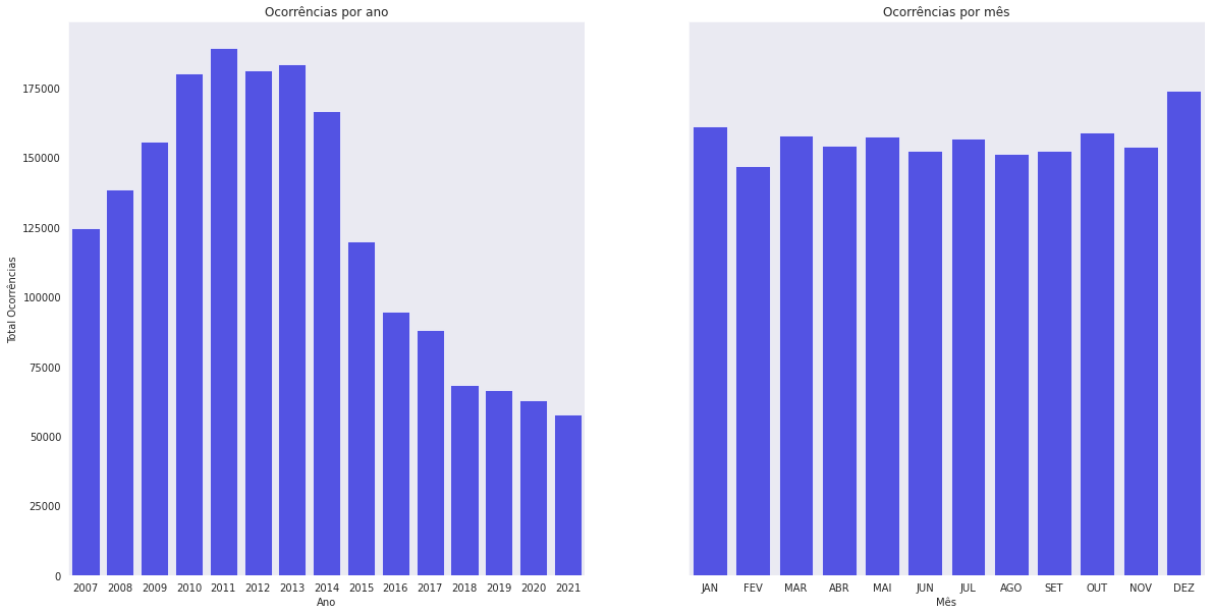

FIGURA 5: Ocorrências por ano e mês.
Fonte: O Autor

Nessa primeira observação vê-se claramente uma queda nas ocorrências iniciada no ano de 2014 e que se manteve até 2021. Em relação aos meses não existe uma diferença considerável entre eles.

Observando as ocorrências por dia de semana, vê-se que a terça-feira é o dia com a menor quantidade:

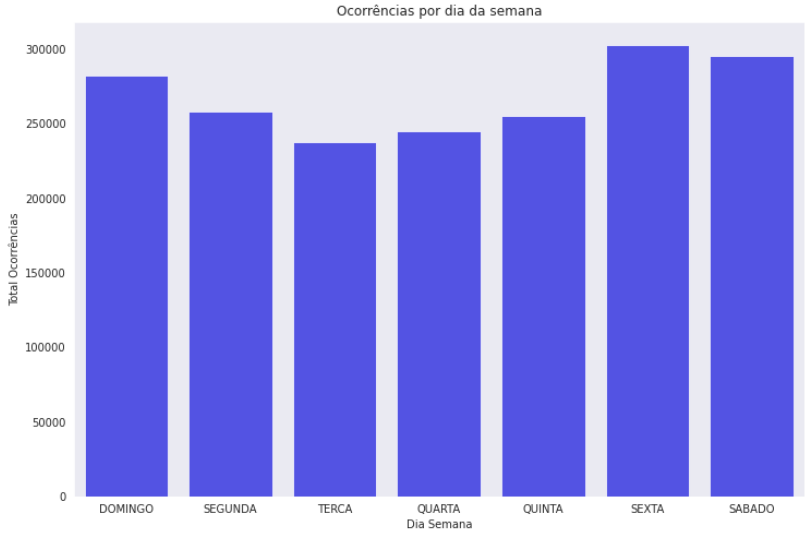

FIGURA 6: Ocorrências por dia de semana.
Fonte: O Autor

A partir da observação das ocorrências por Unidade Federal (UF), vide gráfico abaixo, é que algumas percepções começaram a aparecer.

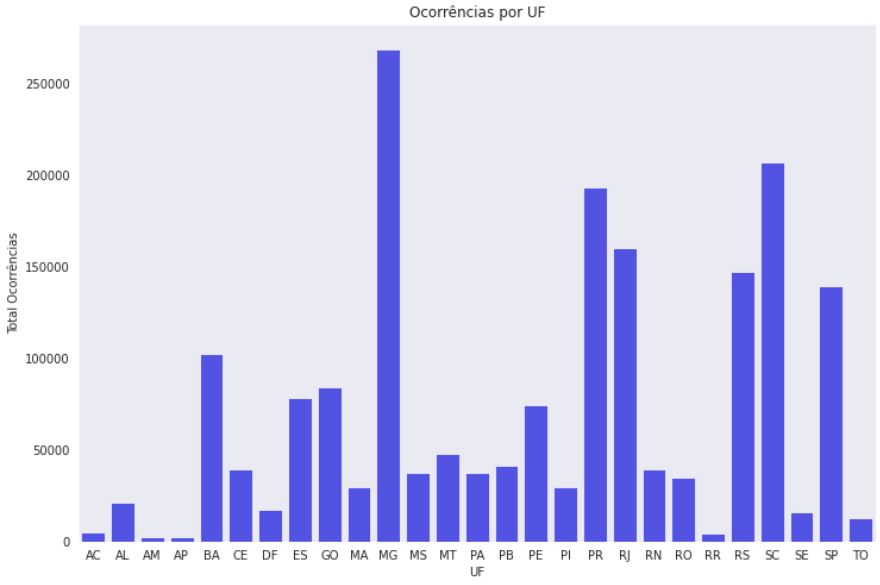

FIGURA 7: Ocorrências por UF.
Fonte: O Autor

Por esse gráfico vimos que apenas 6 estados, sendo eles MG, PR, RJ, RS, SC e SP, são responsáveis por praticamente 60% das ocorrências.

Seguindo por essa linha e observando a quantidade de ocorrências por rodovia pode-se verificar uma predominância nas BR101 e BR116:

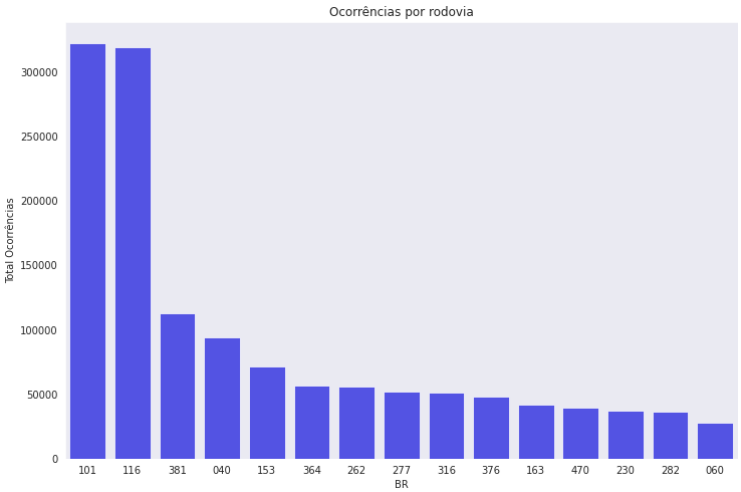

FIGURA 8: Ocorrências por rodovia.*
Fonte: O Autor

*Nesse gráfico foram plotadas as 15 primeiras rodovias ordenadas pela quantidade de ocorrências.

Pelos dois últimos gráficos analisados e com o intuito de evitar o viés de aprendizado do modelo para os estados e rodovias com maior número de ocorrências, serão eliminados os estados e rodovias com menor quantidades.

No gráfico abaixo vê-se que filtrando a base pelas ocorrências nas BR101 e BR116, três estados, RJ, SC e SP aparecem com destaque com uma quantidade muito superior aos demais:

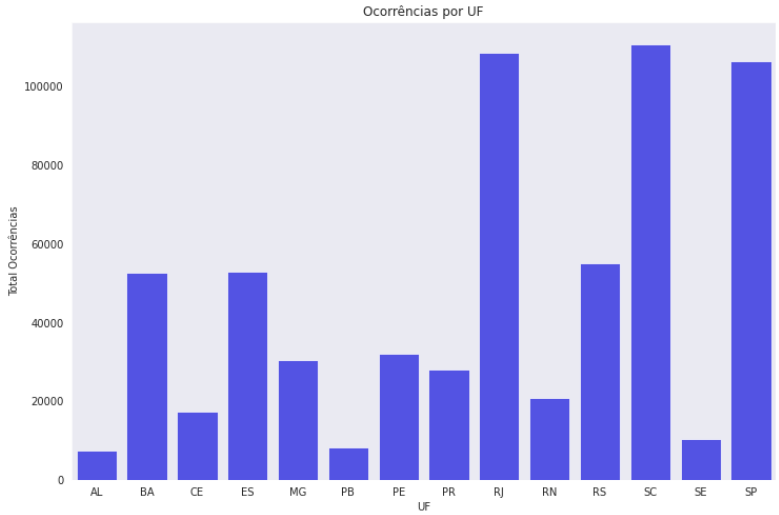

FIGURA 9: Ocorrências por UF.
Fonte: O Autor

Na imagem abaixo temos as ocorrências nas BR101 e BR116 dada a condição meteorológica:

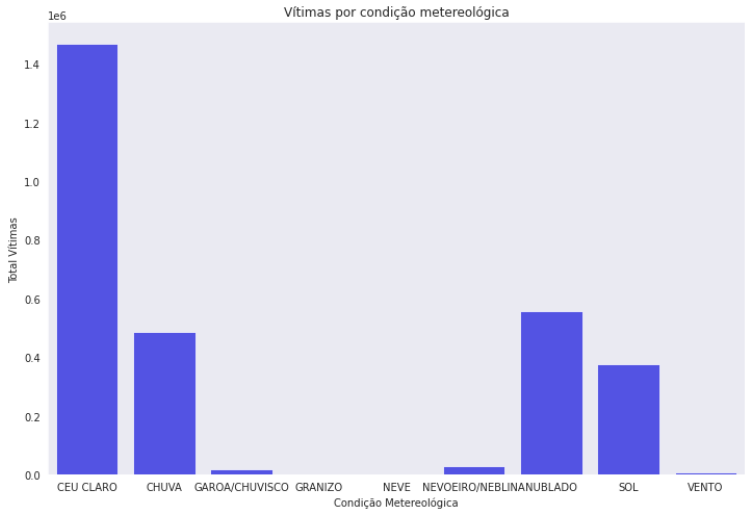

FIGURA 10: Ocorrências por condição meteorológica.
Fonte: O Autor

Como pode-se observar existem tipos de condição meteorológica com um número praticamente desprezível de ocorrências, por isso um tratamento foi feito para agrupá-las em quatro grandes grupos:

Grupo Principal	Tipos de condição metereológica
CEU CLARO	CEU CLARO
CHUVA	CHUVA, GAROA/CHUVISCO, GRANIZO, NEVE
SOL	SOL
NUBLADO	NUBLADO, NEVOEIRO/NEBLINA, VENTO

Após o agrupamento dos dados de condição meteorológica foi gerado o gráfico abaixo por quantidade de ocorrências por rodovia e condição metereológica:

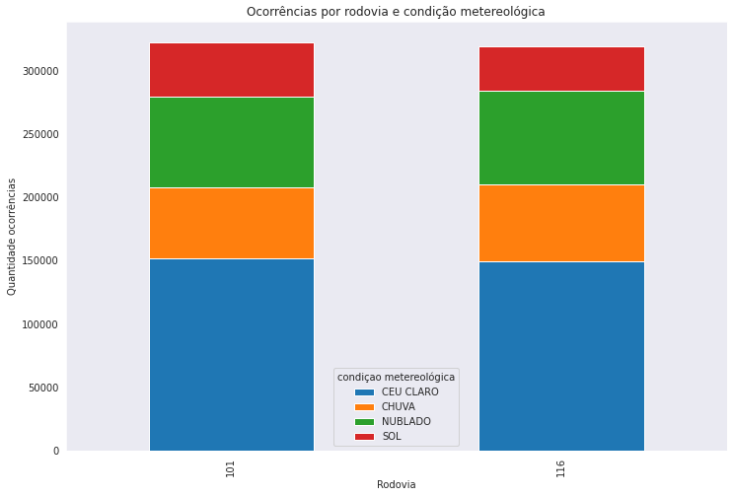

FIGURA 11: Ocorrências por rodovia e condição meteorológica.
Fonte: O Autor

Na imagem seguinte observa-se a ocorrência por intervalo km, filtradas pelas 20 maiores ofensoras:

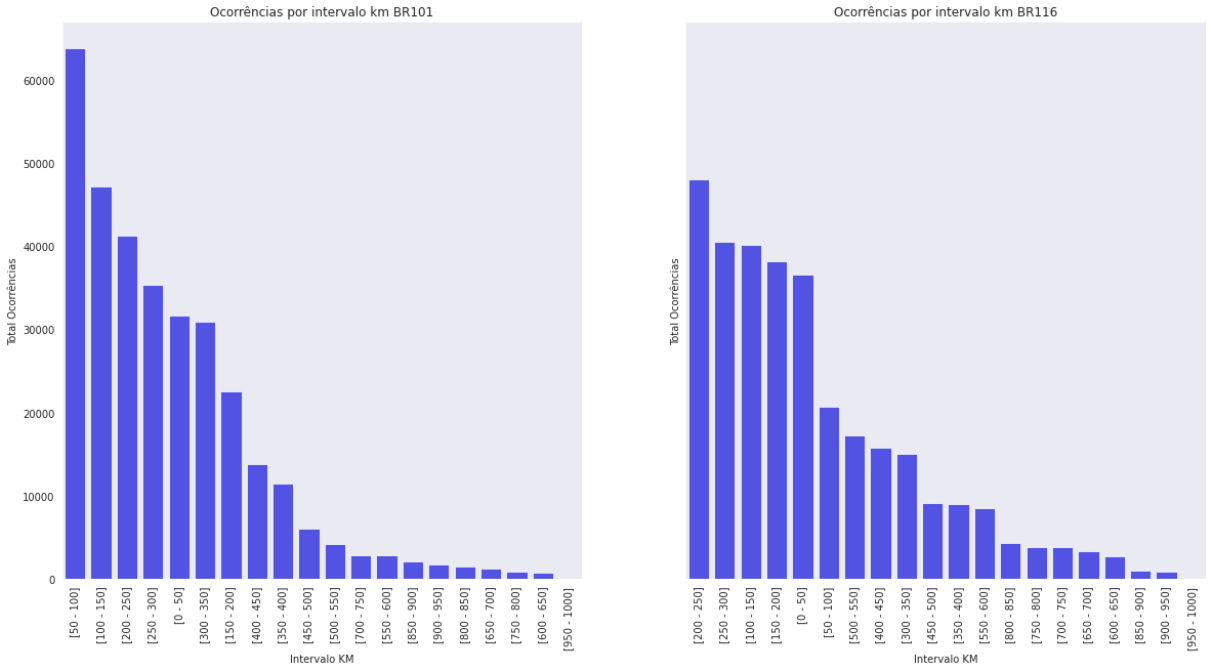

FIGURA 12: Ocorrências por rodovia e condição meteorológica.
Fonte: O Autor

Essa última imagem apresenta a correlação entre as variáveis:

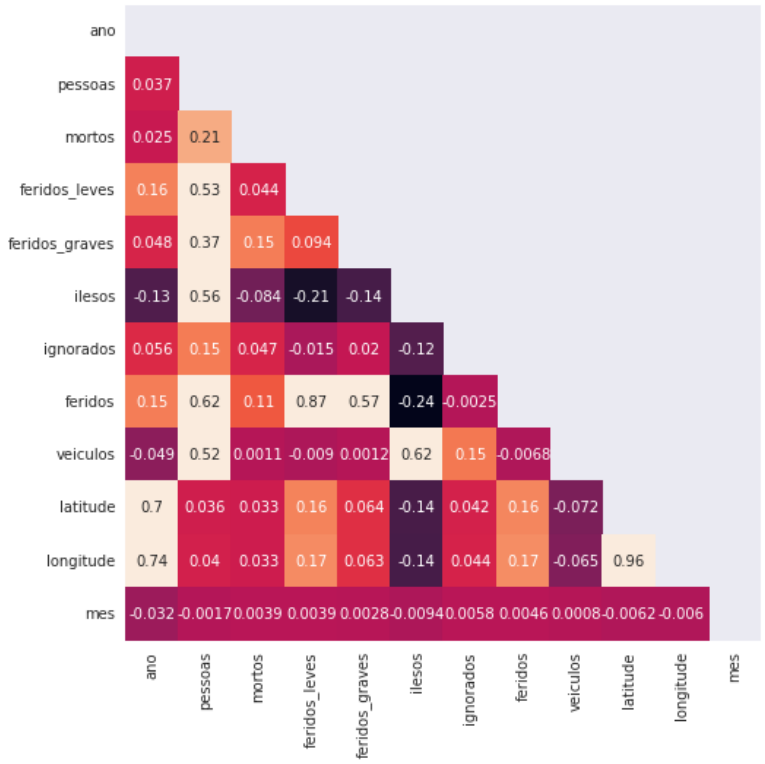

FIGURA 13: Correlação entre as variáveis.
Fonte: O Autor

Observa-se uma correlação positiva mais significativa, coeficiente >= 0.7, entre as variáveis feridos/feridos_leves, latitude/ano, longitude/ano e latitude/longitude, não ocorrendo nenhum caso de correlação negativa mais significativa, coeficiente <= -0.7. A correlação entre feridos/feridos_leves pode ser explicada pela primeira se tratar da soma das variáveis feridos_leves e feridos_graves e porque o atributo feridos_leves apresentar, quase sempre, um valor diferente de 0 o que não ocorre com feridos_graves. As correlações entre latitude/ano, longitude/ano e latitude/longitude se tratam de correlações espúrias, ou seja, não apresentam um sentido real para existirem.

Por fim analisando as demais variáveis define-se os seguintes pontos abaixo para a versão final do conjunto de dados e que serão utilizados na próxima etapa:

- Conterá somente os estados de RJ, SP e SC;
- Conterá as ocorrências nas rodovias BR101 e BR116, além dos 15 maiores intervalos de km em números de ocorrência;
- Os campos id, data_inversa, horário, km, município, causa_acidente, classificação_acidente, fase_dia, sentido_via, condição_meteorologica,

tipo_pista, tracado_via, uso_solo, regional, delegacia, upo serão removidos por não acrescentarem informação relevante;

- Por se tratar de uma proposta que preveja a quantidade de pessoas envolvidas no acidente, independentemente de ser vítima fatal, os campos mortos, feridos_leves, feridos_graves, ilesos e ignorados serão removidos;

7. Preparação dos Dados para os Modelos de Aprendizado de Máquina

Com o conjunto de dados gerado após a análise e exploração dos dados a preparação para uso nos modelos de aprendizado de máquina começa com o agrupamento das variáveis independentes afim de somar a quantidade de pessoas envolvidas em ocorrências acometidas em situações iguais, ou seja, mesmo dia_semana, uf, br, tipo_acidente_agg, fase_dia_agg, km_intervalo, mes e condiçao_metereologica_agg. Esse agrupamento gerou um conjunto de dados com 81407 linhas.

Finalizado o agrupamento verificou-se a porcentagem de casos de cada uma das variáveis independentes no conjunto de dados. Dessa verificação a variável tipo_acidente_agg apresentou valores muito baixos para os tipos INCENDIO, DANOS, DERRAMAMENTO, ENGAVETAMENTO e EVENTOS, conforme imagem:

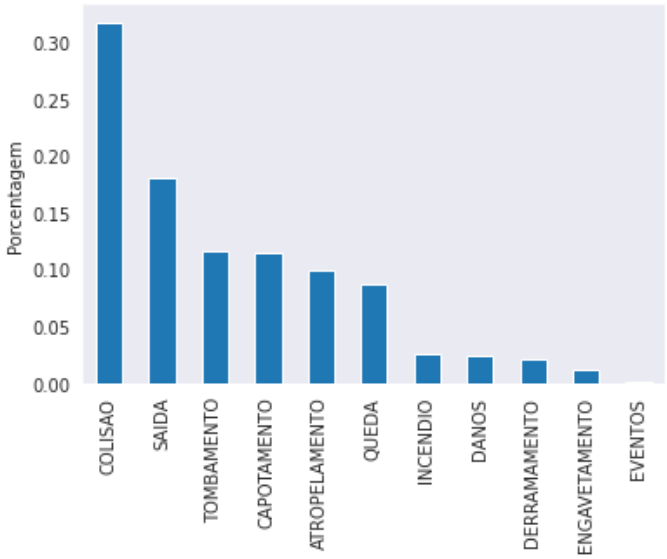

FIGURA 14: Tipo acidente antes do agrupamento.
Fonte: O Autor

Para diminuir a granulidade desse tipo de variável, esses tipos de acidentes relacionados foram classificados como OUTROS, conforme imagem:

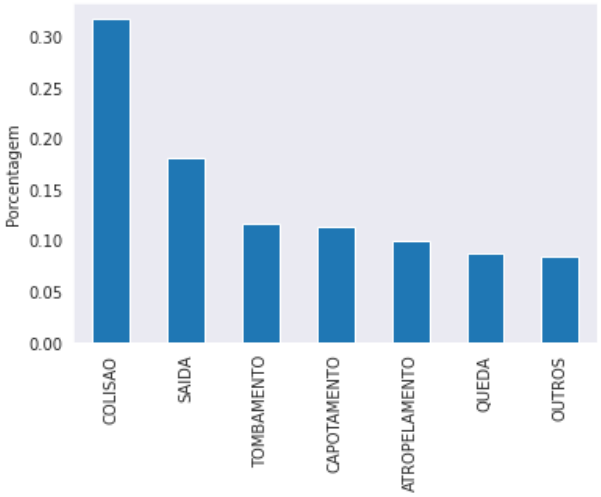

FIGURA 15: Tipo acidente após agrupamento.
Fonte: O Autor

Por fim o conjunto de dados foi randomicamente dividido em treinamento (train), teste (test) e validação (valid), com as respectivas proporções 75%, 15% e 10% de acordo com o código apresentado na próxima imagem:

```
# Split base in train, test, validation
df_group.reset_index(drop=True, inplace=True)

from sklearn.model_selection import train_test_split
x_train, x_temp, y_train, y_temp = train_test_split(df_group.iloc[:, 0:-1], df_group.iloc[:, -1], test_size=0.25)
x_test, x_valid, y_test, y_valid = train_test_split(x_temp, y_temp, test_size=0.4)
```

FIGURA 16: Divisão do conjunto de dados em treinamento, teste e validação.
Fonte: O Autor

8. Aplicação de Modelos de Aprendizado de Máquina

No projeto foram criados 2 modelos de aprendizado de máquina para comparação da performance. Um deles faz uso de árvores de decisão e é encontrado na biblioteca scikit-learn pelo nome de RandomForestRegressor o outro foi desenvolvido usando o framework Tensorflow para criação de redes neurais.

Nas imagens abaixo observam-se as implementações dos 2 modelos:

```
# Encoding variables
from sklearn.preprocessing import OneHotEncoder

ohe = OneHotEncoder(handle_unknown='ignore').fit(df_train)
x_train = ohe.transform(df_train).toarray()
x_test = ohe.transform(df_test).toarray()
```

FIGURA 17: *Encoding* das variáveis qualitativas com o método de One-Hot Encoding.
Fonte: O Autor

```python
# Using grid to test hyperparameters
from sklearn.ensemble import RandomForestRegressor

grid_params = {
    'n_estimators': [50, 100, 200, 300],
    'criterion': ['absolute_error'],
    'max_depth': [10, 15, 20, 25, 30],
    'max_features': ['auto'],
    'random_state': [42]
}

from sklearn.model_selection import GridSearchCV
gr_search_cv = GridSearchCV(estimator=RandomForestRegressor(), param_grid=grid_params).fit(x_train, y_train.values.ravel())
```

FIGURA 18: Implementação do modelo RandomForestRegressor com *grid search* para busca de melhores parâmetros.
Fonte: O Autor

```python
def built_model(layers_size, layers, optimizer, activation):

    hidden = []

    hidden.append(Input(shape=(x_train.shape[1],)))
    for _ in range(layers):
        hidden.append(Dense(layers_size, activation=activation))
        hidden.append(Dropout(0.2))

    hidden += [Dense(layers_size // 2, activation=activation), Dropout(0.2), Dense(1)]

    model = Sequential(hidden)

    model.compile(
        loss='mse',
        optimizer=optimizer,
        metrics=[tf.keras.metrics.MeanAbsoluteError()])

    return model
```

FIGURA 19: Implementação do modelo de rede neural.
Fonte: O Autor

```python
# Grid to search bests hyperparameters
results = []

optimizers = [Adam()]
layers_size = [32, 64, 128]
layers = [2, 3, 4]
epochs = [100, 200, 300]
activation = ['relu', 'gelu']

tf.random.set_seed(42)

for opt in optimizers:
    for lsize in layers_size:
        for l in layers:
            for act in activation:
                model = built_model(lsize, l, opt, act)
                for eph in epochs:
                    model.fit(x_train, y_train.values.ravel(), epochs=eph, verbose=False)
                    results.append(
                        {'optimizer': opt.get_config()['name'],
                         'layers_size': lsize,
                         'layers': l,
                         'epochs': eph,
                         'activation': act,
                         'loss': model.history.history['loss'][-1],
                         'mae': model.history.history['mean_absolute_error'][-1]})
```

FIGURA 20: Implementação do *grid search* ou *hyperparameter tuning* para o modelo de rede neural.
Fonte: O Autor

Como observado nas imagens acima os modelos foram testados com diversos parâmetros e os melhores resultados podem ser observados nas tabelas abaixo. Os resultados completos se encontram disponíveis no github do projeto.

Resultados do modelo de árvore de regressão:

estimators	depth	r2_score	mae	mse
50	30	0.981384558	3.286271395	45.20499538
100	30	0.982329396	3.272144378	44.50408434
200	30	0.982688766	3.262037917	44.44696988
300	30	0.98276498	3.264557639	44.37059085

Onde estimators é a quantidade de árvores geradas, depth é a profundidade máxima das árvores, r2_score (coeficiente de determinação) indica o quanto uma variável dependente pode ser explicada pela variável independente e idealmente deve ser mais próximo de 1, mae (erro absoluto médio) que é o valor absoluto da diferença entre o valor real e o valor estimado e deve ser o menor possível e mse (erro quandrático médio) que fornece a diferença quadrática entre o valor real e valor estimado, assim como mae deve ser o menor possível.

Resultados do modelo de rede neural:

optimizer	layers_size	layers	epochs	activation	loss (mse)	mae
Adam	32	3	300	gelu	45.58987045	3.47167754
Adam	64	3	300	gelu	32.56781006	3.09441685
Adam	128	2	300	gelu	22.86741066	2.72772049

Onde optimizer é o otimizador utilizado, layers_size é a quantidade de neurônios em cada camada escondida, layers total de camadas escondidas, activation é a função de ativação utilizada. Para a rede neural foi feita uma tentativa de utilização do otimizador de gradiente descendente SGD, mas os resultados de mostraram bastante ruins e podem ser vistos no github do projeto.

9. Discussão dos Resultados

Com os resultados das métricas dos modelos observada no capítulo anterior definiu-se que seria utilizado o que apresentou os melhores valores. Portanto foi escolhido o modelo de rede neural com 2 camadas escondidas e 128 neurônios em cada uma delas.

Para verificar a performance do modelo em termos de generalização ele foi submetido a uma validação cruzada (cross-validation). Além da validação cruzada, variou-se a quantidade de épocas pois queria analisar se um número maior ou menor poderia influenciar na generalização. Abaixo pode se observar a implementação do teste:

```
# Using cross validation nested to check best epoch size and activation

# Para utilização do grid search do scikit-learn necessário uso
# do KerasRegressor do scikeras.wrappers

from scikeras.wrappers import KerasRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_absolute_error, make_scorer

sc = 'neg_mean_absolute_error'

params = {'random_state': [42],
          'epochs': [100, 200, 300]}

model_params = {
    'loss': 'mse',
    'optimizer': Adam(),
    'metrics': [tf.keras.metrics.MeanAbsoluteError()],
    'verbose': False,
    'batch_size': 100
}

model = KerasRegressor(built_model, **model_params)
gs = GridSearchCV(model, param_grid=params, cv=5, scoring=sc, refit=True, n_jobs=-1)
gs.fit(x_train, y_train.values.ravel())
```

FIGURA 21: Implementação da validação cruzada usando grid search.
Fonte: O Autor

Na tabela abaixo tem-se os resultados das métricas e observa-se que o modelo de 100 épocas apresentou o melhor valor na métrica de mean_test_score, que representa a média total de todos os resultados finais de erro absoluto médio além de possuírem um valor de desvio-padrão, de acordo com a coluna std_test_score, que não é discrepante dos demais.

param_epochs	mean_test_score	std_test_score	rank_test_score
100	-2.987680483	0.0288213972	1

200	-3.010177422	0.02836043636	2
300	-3.037857771	0.01943810293	3

As demais colunas que compõem a saída do grid search podem ser verificadas no github do projeto.

Para verificação de overffiting do modelo foi feito um teste com os dados de teste que foram previamente separados conforme consta no capítulo 7. Os resultados de MAE (erro absoluto médio), MDAE (mediana do erro absoluto) e R^2 (coeficiente de determinação) entre o resultado predito e o resultado real comprovam a eficácia e generalização do modelo para além dos dados de treinamento descartando assim a ocorrência de overffiting. Abaixo a tabela com os resultados das métricas e um gráfico com as 200 primeiras predições comparadas com os 200 primeiros valores da variável dependente:

MAE	MDAE	R^2
3.0502662922082333	1.3667738437652588	0.8984063782940498

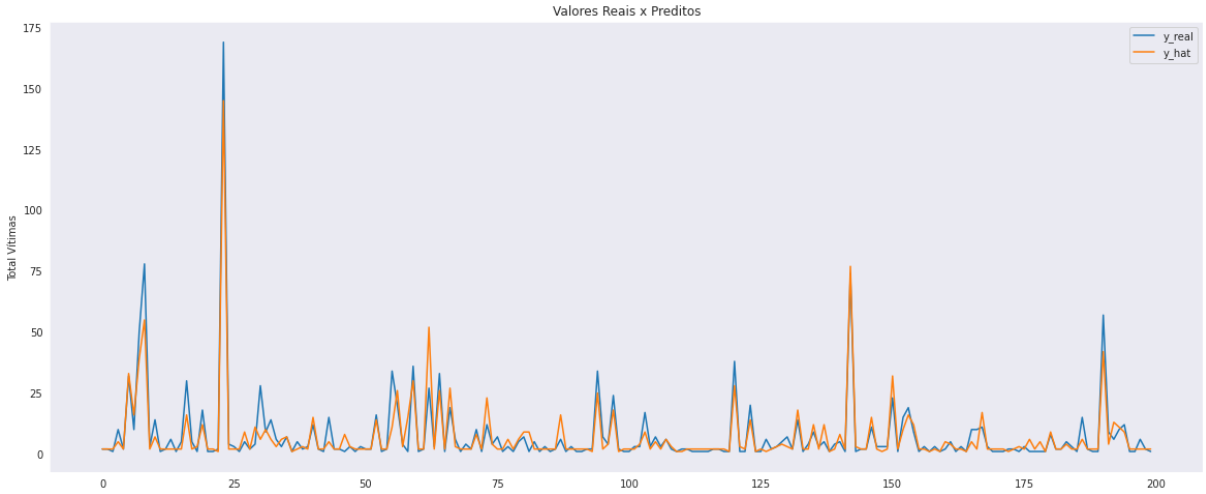

FIGURA 22: 200 primeiros valores reais versus valores preditos pelo modelo com dados de teste.
Fonte: O Autor

Por fim a eficácia do modelo pode ser verificada em mais um teste dessa vez com os dados de validação previamente separados conforme capítulo 7 e que demostra valores de MAE, MDAE e R^2 bastante satisfatórios além do gráfico com as 200 primeiras predições comparadas com os 200 primeiros valores da variável

dependente e que demostram, assim como nas predições com os dados de teste, que o modelo segue performando de forma bastante satisfatória.

MAE	MDAE	R^2
2.975739397005138	1.3571653366088867	0.8964060156354174

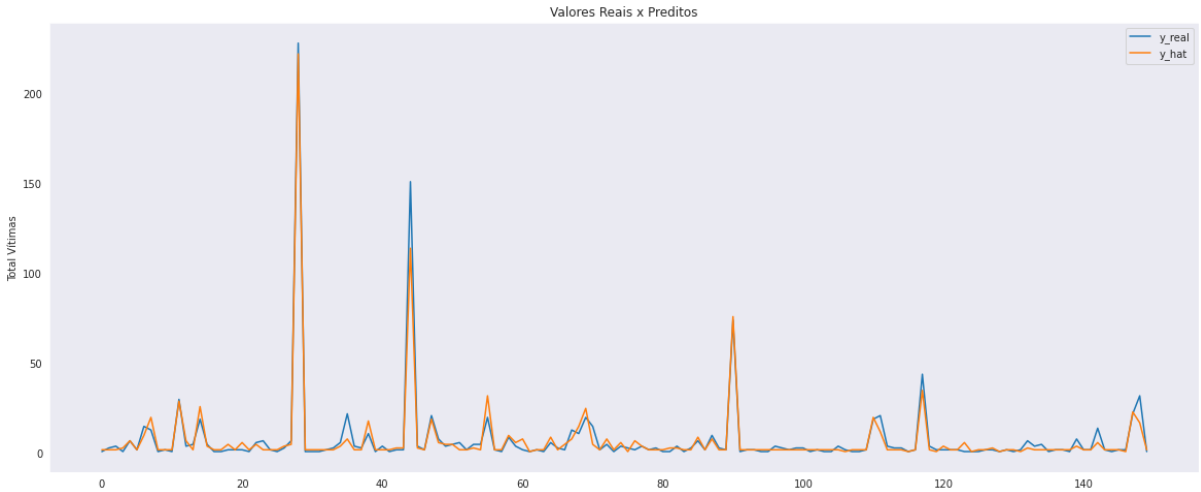

FIGURA 23: 200 primeiros valores reais versus valores preditos pelo modelo com dados de validação.
Fonte: O Autor

10. Conclusão

De acordo com o Denatran, em 2019, o Brasil chegou a um contingente de 45,4 milhões de veículos além de, segundo a CNT, ocorrerem em média 50 mil acidentes com vítimas por ano em rodovias federais.

Esse trabalho se propôs, por meio das bases de dados de acidentes obtidas no site da PRF, criar um modelo que pudesse, dada as variáveis de dia da semana, unidade federal, rodovia, tipo de acidente, fase do dia, km da rodovia, mês e condição meteorológica, prever a quantidade de vítimas que podem estar envolvidas num acidente. Essa informação seria utilizada por órgãos como SAMU, Bombeiros e a própria PRF para se prepararem pro atendimento aos trechos que poderão ter mais ocorrências.

Após todo o tratamento e análise dos dados a solução desenvolvida se fez com o uso de uma rede neural que apresentou um R^2 ou coeficiente de determinação entre as predições fornecidas pelo modelo e os valores e reais, tanto para os dados de teste quanto de validação, de quase 90%, indicando a boa generalização.

Para uma abordagem futura será interessante a inclusão de outras rodovias o que requer um esforço bastante grande para balanceamento dos dados, além da criação de uma interface web que permita indicar os locais de ocorrência de forma mais amigável.

11. Links

Github do projeto: https://github.com/ficorrea/courses/tree/main/pos_graducacao_puc/TCC/project.

12. Referências

PLANALTO. Disponível em http://www.planalto.gov.br/ccivil_03/_ato2011-2014/2011/lei/l12527.htm, acesso em 26 de março de 2022.

SERENATA DE AMOR. Disponível em https://serenata.ai/, acesso em 10 de abril de 2022.

DENATRAN. Disponível em http://www.and.org.br/brasil-ja-tem-1-carro-a-cada-4-habitantes-diz-denatran/, acesso 20 de março de 2022.

CNT. Disponível em https://www.cnt.org.br/painel-acidente, acesso em 20 de março de 2022.

PRF. Disponível em https://www.gov.br/prf/pt-br/acesso-a-informacao/dados-abertos/dados-abertos-acidentes, acesso em 26 de fevereiro de 2022.

PRF. Disponível em https://www.gov.br/prf/pt-br/acesso-a-informacao/dados-abertos/dicionario-acidentes. acesso em 26 de fevereiro de 2022.