

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

**Wagner Evangelista de Abreu**

**USO DE REDES NEURAS RECORRENTES NA PREVISÃO DE DEMANDA DO  
TRANSPORTE RODOVIÁRIO INTERESTADUAL DE PASSAGEIROS SOB O  
REGIME DE FRETAMENTO COM DESTINO A APARECIDA/SP**

Belo Horizonte  
Junho de 2022

**Wagner Evangelista de Abreu**

**A UTILIZAÇÃO DE REDES NEURAIS RECORRENTES NA PREVISÃO DE  
DEMANDA DO TRANSPORTE RODOVIÁRIO INTERESTADUAL DE  
PASSAGEIROS SOB O REGIME DE FRETAMENTO PARA APARECIDA/SP**

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*.

Belo Horizonte

Junho de 2022

## SUMÁRIO

1. Introdução.....	4
2. Descrição do Problema e da Solução Proposta .....	4
3. Canvas Analítico .....	10
4. Coleta de Dados .....	10
5. Processamento/Tratamento de Dados .....	12
6. Análise e Exploração dos Dados .....	19
7. Preparação dos Dados para os Modelos de Aprendizado de Máquina .....	19
8. Aplicação de Modelos de Aprendizado de Máquina .....	19
9. Discussão dos Resultados.....	19
10. Conclusão .....	19
11. Links .....	19
12. Referências .....	19

## 1. Introdução

A previsão da demanda por transporte de passageiros é de importância significativa para o planejamento da infraestrutura rodoviária e das cidades, da fiscalização e monitoramento das viagens geradas, no caso do poder público, bem como para a adequação da oferta, aumento da qualidade do serviço e redução dos custos de operação, no caso do setor privado e mercado.

Algumas vezes, pode-se obter a demanda pelo serviço de transporte como uma série temporal com determinada periodicidade (diária, semanal, mensal etc.).

Existem modelos que têm sido aplicados com sucesso em previsões de valores de séries temporais. No campo da Aprendizagem de Máquina, o uso de Redes Neurais Artificiais (ANN – *Artificial Neural Networks*, em inglês), especificamente, de Redes Neurais Recorrentes (RNN – *Recurrent Neural Networks*, em inglês), têm sido úteis na execução deste tipo de tarefa, visto que tais redes podem processar de forma eficiente dados sequenciais.

Assim, neste trabalho em particular, foram utilizados modelos de RNN para estimar o volume de passageiros transportados, em um período de referência, via transporte rodoviário interestadual sob regime de fretamento para a localidade de Aparecida/SP, no período de 2007 a 2019.

Como forma de comparação foram utilizadas três arquiteturas a saber: *Vanilla RNN* (*Recurrent Neural Networks*, em inglês), LSTM (*Long Short Term Memory*, em inglês) e GRU (*Gated Recurrent Unit*, em inglês). Além disso, buscou-se enriquecer os dados com as dimensões de feriados nacionais, dias da semana e visitas dos Papas àquela localidade, de modo a obter séries multivariadas.

## 2. Descrição do Problema e da Solução Proposta

O transporte rodoviário interestadual de passageiros sob regime de fretamento “é o serviço prestado à pessoa ou a um grupo de pessoas, em circuito fechado, com emissão de nota fiscal e lista de pessoas transportadas, por viagem, com prévia autorização ou licença da Agência Nacional de Transportes Terrestres – ANTT” (Decreto 2521/98). O circuito fechado, por sua vez, é definido como “uma viagem de um grupo de passageiros com motivação comum que parte em um veículo de local de origem a um ou mais locais de destino e, após percorrer todo o itinerário, observado os tempos de permanência estabelecidos (...), este grupo de passageiros retorna ao

*local de origem no mesmo veículo que efetuou o transporte na viagem de ida” (Res. ANTT 4.777/15).*

Para regiões turísticas ou que possuem atrativos comerciais, conhecer o volume de passageiros transportados por período, seja rotineiramente ou em feriados e datas comemorativas, é bastante importante, considerando-se a necessidade de planejamento da infraestrutura e dos serviços para o atendimento às atividades ligadas ao turismo, como hospedagem e serviços de alimentação, por exemplo (ORTEGA et al., 2013).

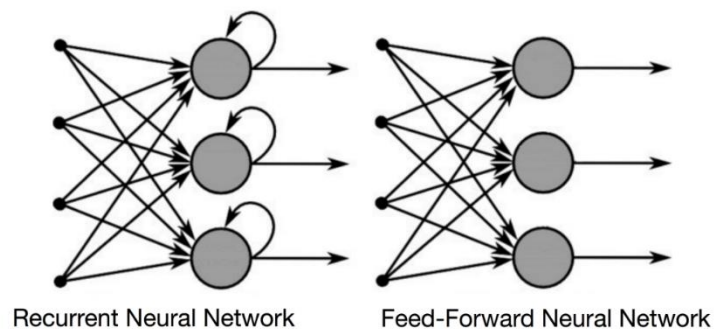
Um caso interessante de se estudar é o do município de Aparecida, São Paulo, região turística, conhecida nacionalmente por abrigar o Santuário Nacional de Nossa Senhora Aparecida. De acordo com CARMO e VALENTE (2017), no período de 1998 a 2014, esta localidade teria recebido anualmente, aproximadamente, 12 milhões de pessoas entre peregrinos católicos e turistas. Quando se compara com os dados de viagens fretadas da ANTT no período de 2007 a 2014, verifica-se que foram transportados anualmente, com destino a essa localidade, uma média de 1,4 milhões de passageiros, o que corresponde a algo em torno de 11,4% do total de visitantes recebidos. Trata-se de uma quantia razoável, principalmente se se considerar que estas viagens são realizadas entre Unidades da Federação, excluindo-se o próprio estado de São Paulo. Tendo, portanto, grande impacto na infraestrutura viária e serviços derivados ou associados às atividades turísticas, evidenciando-se, dessa forma, a importância da previsão do deslocamento desse quantitativo de pessoas.

Como tentativa de se obter alguma estimativa do número de pessoas transportadas, pode-se obter uma série temporal, com periodicidade diária, no mínimo, visto que a ANTT fornece os dados desagregados, individualizados por licença de viagem emitida. Estes dados por sua vez, podem ser agregados em outras unidades temporais como a semana ou o mês.

É sabido que existem modelos estatísticos aplicados com sucesso em realizar previsões de valores de séries temporais. Da mesma forma, modelos baseados em aprendizagem de máquina, como as redes neurais artificiais, por exemplo, também têm sido eficientes na realização deste tipo de previsão, o que pode ser realizado por meio da utilização das Redes Neurais Recorrentes (*Recurrent Neural*

*Networks*- RNN, em inglês), as quais possuem a capacidade de processar entradas de dados sequenciais tais como séries temporais.

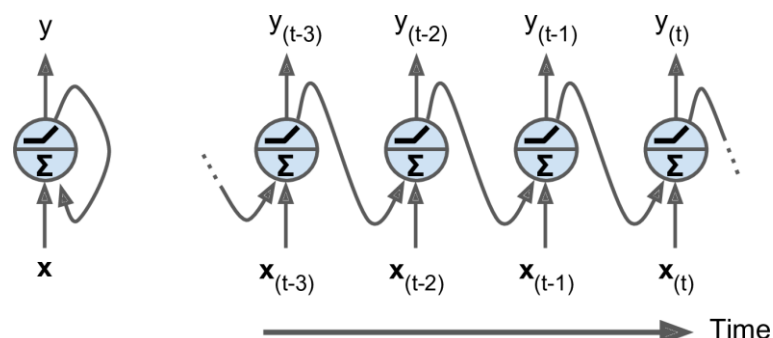
De acordo com Rumelhart et al. (1986a, apud GOODFELLOW, et al., 2016, p. 367), as redes neurais recorrentes (RNN) “são uma família de redes neurais para processamento de dados sequenciais”, apresentando propriedades interessantes como performance robusta na predição e a habilidade de capturar dependências temporais de longo prazo (CHE et al., 2018). Uma RNN simples é semelhante a uma rede neural *feedforward*, com a diferença que a rede recorrente também recebe conexões *backward*. (GERÓN, 2021). A Figura 1 ilustra simplificadaamente as diferenças entre estes dois tipos de redes.



**Figura 1 - Rede Neural Recorrente e Rede Neural Feed-Forward.**

**Fonte: Data Science Academy, 2022.**

A Figura 2 mostra uma RNN “estendida”, o que significa apenas que a rede é mostrada em diferentes sequências de processamento onde os vetores de entrada em  $t-1$  precede àquele em  $t$ .



**Figura 2 – Rede Neural Recorrente “estendida”.**

**Fonte: GERÓN, 2021.**

Matematicamente, pode-se escrever a saída de uma camada recorrente para uma única instância de uma RNN como (GERÓN, 2021):

### Equação 1 – Cálculos RNN Simples

$$\mathbf{y}_{(t)} = \phi(\mathbf{W}_x^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_y^T \cdot \mathbf{y}_{(t-1)} + \mathbf{b})$$

onde:

- $\mathbf{y}_{(t)}$ : saída da camada no tempo ou passo  $t$ ;
- $\phi$ : função de ativação;
- $\mathbf{W}_x$  e  $\mathbf{W}_y$ : matrizes de peso;
- $\mathbf{x}_{(t)}$ : entrada no tempo ou passo  $t$ ; e
- $\mathbf{y}_{(t-1)}$ : saída no passo ou intervalo de anterior ( $t - 1$ ).

Os parâmetros  $\mathbf{W}_x$  e  $\mathbf{W}_y$  são os mesmos para todos os passos do processo, representando tais pesos a memória da rede. Entretanto, para redes neurais profundas, em sequências muito longas, a memória da rede pode ser prejudicada pelo problema do desaparecimento ou explosão do gradiente quando da execução do processo de *back propagation*.

Além das RNN Simples, pode-se utilizar arquiteturas que proporcionam memórias de longo prazo. São estas: as LSTM e as GRU mencionadas anteriormente. A arquitetura LSTM é mostrada na Figura 3. Nela, observa-se que a célula da rede é composta essencialmente por quatro camadas totalmente conectadas diferentes, como segue:

- camada principal: recebe as entradas  $\mathbf{x}_{(t)}$  e  $\mathbf{h}_{(t-1)}$  e gera a saída  $\mathbf{g}_{(t)}$  como resultado da função *tanh*, cuja saída encontra-se no intervalo de -1 e 1.
- *forget gate*,  $\mathbf{f}_{(t)}$ : controla as partes do estado de longo que devem ser deletadas;
- *input gate*,  $\mathbf{i}_{(t)}$ : controla quais partes de  $\mathbf{g}_{(t)}$  serão adicionadas ao estado de longo prazo; e
- *output gate*,  $\mathbf{o}_{(t)}$ : “controla quais partes do estado de longo prazo devem ser lidas e geradas neste intervalo de tempo, tanto para  $\mathbf{h}_{(t)}$  como para  $\mathbf{y}_{(t)}$ ” (GÉRON, 2021).

As saídas  $\mathbf{f}_{(t)}$ ,  $\mathbf{i}_{(t)}$ ,  $\mathbf{o}_{(t)}$  são valores de 0 a 1 modulados pela função sigmóide.

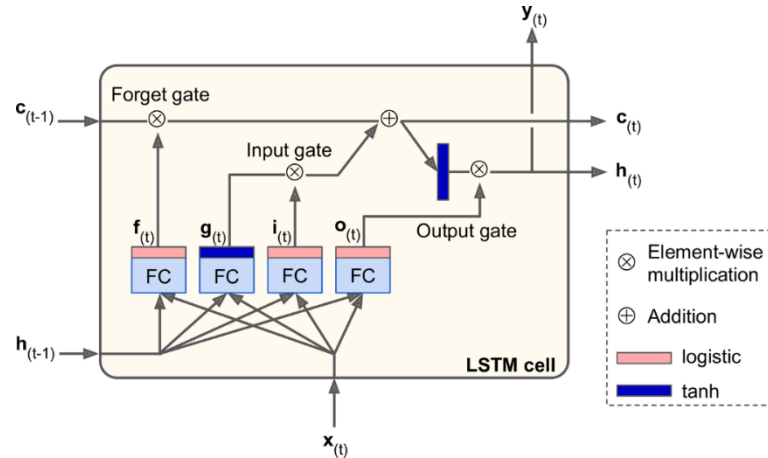


Figura 3 – Arquitetura LSTM

Fonte: GÉRON, 2018.

As equações para que descrevem os processos acima são as mostradas a seguir:

#### Equação 2 – Cálculos LSTM

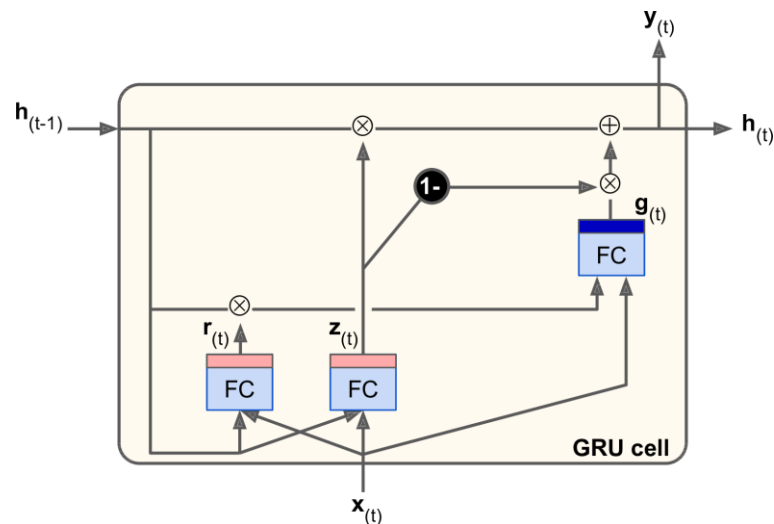
$$\begin{aligned}
 \mathbf{i}_{(t)} &= \sigma(\mathbf{W}_{xi}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hi}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_i) \\
 \mathbf{f}_{(t)} &= \sigma(\mathbf{W}_{xf}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hf}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_f) \\
 \mathbf{o}_{(t)} &= \sigma(\mathbf{W}_{xo}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{ho}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_o) \\
 \mathbf{g}_{(t)} &= \tanh(\mathbf{W}_{xg}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hg}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_g) \\
 \mathbf{c}_{(t)} &= \mathbf{f}_{(t)} \otimes \mathbf{c}_{(t-1)} + \mathbf{i}_{(t)} \otimes \mathbf{g}_{(t)} \\
 \mathbf{y}_{(t)} &= \mathbf{h}_{(t)} = \mathbf{o}_{(t)} \otimes \tanh(\mathbf{c}_{(t)})
 \end{aligned}$$

onde:

- $\mathbf{W}_{xi}, \mathbf{W}_{xf}, \mathbf{W}_{xo}, \mathbf{W}_{xg}$ : são matrizes de peso de cada camada para a entrada  $\mathbf{x}_{(t)}$ ;
- $\mathbf{W}_{hi}, \mathbf{W}_{hf}, \mathbf{W}_{ho}, \mathbf{W}_{hg}$ : são matrizes de peso de cada camada para a entrada  $\mathbf{h}_{(t-1)}$ ; e
- $\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_o, \mathbf{b}_g$ : são vieses para cada camada.

A arquitetura da célula GRU, por sua vez, é uma versão simplificada da célula LSTM. Dentre as modificações está a incorporação dos vetores de estado  $\mathbf{x}_{(t)}$  e  $\mathbf{h}_{(t)}$  em um único vetor  $\mathbf{h}_{(t)}$ . Além disso, um único controlador  $\mathbf{z}_{(t)}$  controla tanto o *forget gate* quanto o *input gate*. Outra modificação é que não há o *output gate*, sendo introduzido o controlador  $\mathbf{r}_{(t)}$  (*reset gate*) que controla qual parte do estado anterior deve ser exibida para a camada principal  $\mathbf{g}_{(t)}$  (Figura 4).





**Figura 4 – Célula GRU.**

**Fonte: GÉRON, 2018.**

A Equação 3 mostra como são realizados os cálculos em cada controlador da célula GRU.

**Equação 3 – Cálculo da célula GRU.**








$$\begin{aligned}
 \mathbf{z}_{(t)} &= \sigma(\mathbf{W}_{xz}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hz}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_z) \\
 \mathbf{r}_{(t)} &= \sigma(\mathbf{W}_{xr}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hr}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_r) \\
 \mathbf{g}_{(t)} &= \tanh(\mathbf{W}_{xg}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hg}^T \cdot (\mathbf{r}_{(t)} \otimes \mathbf{h}_{(t-1)}) + \mathbf{b}_g) \\
 \mathbf{h}_{(t)} &= \mathbf{z}_{(t)} \otimes \mathbf{h}_{(t-1)} + (1 - \mathbf{z}_{(t)}) \otimes \mathbf{g}_{(t)}
 \end{aligned}$$

Embora ainda apresente problemas em aprender padrões de longo prazo em sequências de 100 intervalos de tempo ou mais, ambas as células, LSTM e GRU, conseguem lidar com sequências bem mais longas do que RNNs simples (GÉRON, 2021). Vale ressaltar ainda que existem outras variações das arquiteturas mostradas acima que não serão abordadas no escopo deste trabalho.

### 3. Canvas Analítico

## Software Analytics Canvas

Project: Redes Neurais Recorrentes Previsão de Demanda de TRIP Fretamento

<div></div> <h2>1. Question</h2> <p><i>What is it that we want to know about the software / processes / usage / organization / etc.?</i></p> <p>Na previsão do número de passageiros transportados para Aparecida/SP por meio de serviço rodoviário interestadual fretado, para quais métodos de Redes Neurais Recorrentes (RNN Simples, LSTM e GRU) e para quais periodicidades (diária, semanal e mensal) obtém-se melhor performance quando se utiliza como função de perda o Erro Quadrático Médio (MSE)?</p>	<div></div> <h2>2. Data Sources</h2> <p><i>Which data can possibly answer our question? What information do we need?</i></p> <p>Os dados utilizados foram obtidos do portal de dados abertos do Governo Federal (<a href="https://dados.gov.br/">https://dados.gov.br/</a>), denominado Licenças de Viagem Nacional – Serviço Fretado, no período de janeiro de 2007 a dezembro de 2019. Nesse conjunto são fornecidos o número de transportados passageiros por viagem licenciada pela ANTT.</p>	<div></div> <h2>3. Heuristics</h2> <p><i>Which assumptions do we want to make to simplify the answer to our question?</i></p> <ul style="list-style-type: none"><li>• O quantitativo de passageiros transportados depende de variáveis temporais.</li><li>• A data de referência é dada pela data média entre o início e o término da viagem.</li><li>• Os feriados influenciam no volume de passageiros transportados.</li><li>• O dia da semana também pode influenciar no volume de passageiros transportados por dia.</li><li>• No caso de Aparecida/SP, a visita de um Papa à localidade também afeta o volume de pessoas transportadas.</li><li>• Não foram considerados outros fatores como variáveis, climáticas ou econômicas, entre outras.</li></ul>	<div></div> <h2>4. Validation</h2> <p><i>What results do we expect from our analysis, how are they reviewed and presented in an understandable way?</i></p> <ul style="list-style-type: none"><li>• Podese esperar ao final do processo obter uma série temporal ou resultado isolado com a previsão do número de passageiros transportados em uma determinado dia, semana ou mês do ano;</li><li>• Utilizar como métrica de avaliação o Erro Quadrático Médio;</li><li>• Comparar os resultados obtidos nas arquiteturas de redes utilizadas.</li></ul>
<div></div> <h2>5. Implementation</h2> <p><i>How can we implement the analysis step by step and in a comprehensible way?</i></p> <ol style="list-style-type: none"><li>1. Consolidação dos dados em um único arquivo;</li><li>2. Agregação dos dados período(dia, semana ou mês), somando-se o total de passageiros transportados de modo a se obter uma série temporal.</li><li>3. Enriquecimento da base com o acréscimo dos campos de dia da semana, feriados e visita do Papa.</li><li>4. Preparação dos dados de passageiros transportados, normalizando-os em um intervalo entre 0 e 1, por meio da função <i>MinMaxScaler</i> do pacote <i>Sklearn</i>;</li><li>5. Reescalonamento dos dados de forma a se obter colunas com os valores de passageiros defasados de uma determinado período de tempo.</li><li>6. Divisão dos dados em base de treinamento e de teste;</li><li>7. Construção de um modelo RNN (simples, LSTM e GRU) com dropout, função de perda MSE e otimizador ADAM padrão.</li><li>8. Avaliação do modelo por meio das métricas RMSE, MSE, MAE, MAPE, nos conjuntos de teste e treino.</li><li>9. Predição e Visualização;</li><li>10. Armazenamento do modelo para utilização posterior em produção.</li></ol>	<div></div> <h2>6. Results</h2> <p><i>What are the main insights from our analysis?</i></p> <ul style="list-style-type: none"><li>•Avaliar a influência de outras fatores, incluindo outras variáveis ao modelo como, por exemplo, variáveis econômicas e socioculturais;</li><li>•Modificar as arquiteturas das redes utilizadas, testando se a profundidade influencia positivamente na melhora de performance do modelo;</li><li>•Realizar o ajuste fino de hiperparâmetros do modelo.</li></ul>	<div></div> <h2>7. Next Steps</h2> <p><i>What follow-up actions can we derive from the findings? Who or what do we need to address next?</i></p> <ul style="list-style-type: none"><li>•Avaliar a influência no modelo de outros fatores tais como variáveis econômicas, climáticas ou socioculturais;</li><li>•Modificar as arquiteturas das redes utilizadas, testando se a profundidade da rede influencia positivamente na performance do modelo;</li><li>•Realizar o ajuste fino de hiperparâmetros do modelo.</li></ul>	

Software Analytics Canvas v1.0 designed by Markus Harrer. Visit <https://www.feststelltaste.de/software-analytics-canvas/> for more information. CC BY-SA 4.0

### 4. Coleta de Dados

Os dados de **Licenças de Viagem Nacional – Serviço Fretado** utilizados nesse trabalho foram obtidos na página de Dados Abertos do Governo Federal, em 1º de março de 2022. Os dados podem ser baixados na página citada ou via Github, pelos seguintes endereços:

- [www.dados.gov.br/](http://www.dados.gov.br/);
- <https://dados.gov.br/dataset/licencas-de-viagem-nacional-servico-fretado>
- <https://github.com/weabreu/projeto-tcc-ml-ia/blob/main/dataset/>;
- [https://github.com/weabreu/projeto-tcc-ml-ia/blob/main/dataset/licencas\\_de\\_viagens\\_desagregadas.zip](https://github.com/weabreu/projeto-tcc-ml-ia/blob/main/dataset/licencas_de_viagens_desagregadas.zip); e
- [https://github.com/weabreu/projeto-tcc-ml-ia/blob/main/dataset/licencas\\_de\\_viagens\\_agregadas.zip](https://github.com/weabreu/projeto-tcc-ml-ia/blob/main/dataset/licencas_de_viagens_agregadas.zip).

Por meio dos dados coletados, foi possível construir um histórico da demanda por esse tipo de serviço, com uma periodicidade definida e a partir daí, obter estimativas de passageiros transportados para localidade foco.

À época, era possível obter, neste repositório, dados de licenças de viagens emitidas pela ANTT referentes ao período de janeiro de 2006 a fevereiro de 2022, onde constavam as informações descritas no Quadro 1.

<b>Nome do dataset:</b> Licenças de Viagens Nacionais - Serviço Fretado <b>Descrição:</b> Licenças de viagem emitidas para a prestação do serviço de transporte rodoviário interestadual de passageiros sob regime de fretamento <b>Link:</b> <a href="https://dados.gov.br/">https://dados.gov.br/</a>		
Nome do Atributo	Descrição	Tipo
data_inicio_viagem	Data de início da licença de viagem.	Data
data_fim_viagem	Data final da licença de viagem.	Data
cnpj	Número da autorizatória no cadastro nacional de pessoa jurídica.	Numérico
razao_social	Razão social da empresa autorizada.	Alfanumérico
placa	Placa de identificação do veículo.	Alfanumérico
numero_licenca_viagem	Número da licença de viagem emitida.	Numérico
município_origem	Nome do município de origem da licença de viagem.	Texto
uf_município_origem	Unidade federativa de origem da viagem.	Texto
município_destino	Nome do município mais distante de destino da viagem.	Texto
uf_município_destino	Unidade federativa mais distante de destino da viagem.	Texto
total_passageiros	Quantidade de passageiros transportados.	Numérico
data_fim_viagem	Data final da licença de viagem.	Data

**Quadro 1 – Licenças de Viagem Nacionais emitidas pela ANTT – 2007 a 2019.**

A partir dos dados disponíveis é possível obter séries temporais de viagens executadas e passageiros transportados por meio da agregação das informações das licenças de viagens, cuja unidade temporal mínima é a diária, permitindo assim análises em vários níveis temporais (diária, semanal, mensal, trimestral etc.).

Como mencionado anteriormente, trata-se de uma base de viagens interestaduais de viagens nacionais, ou seja, viagens entre municípios de Unidades da Federação distintas, limitadas ao território nacional. Pode-se obter também vários níveis de

agregação (municipal, estadual, regional ou nacional), além de poder ser segmentado por origem ou destino da viagem.

Já os feriados considerados foram aqueles de caráter nacional decretados oficialmente pelo Governo do Brasil e obtidos por meio do site

<<https://www.anbima.com.br>>, em 31 de março de 2022. Ressalta-se nesse cenário importância do feriado de Nossa Senhora Aparecida, quando muitos romeiros rotineiramente se deslocam para localidade para acompanhar as missas e festividades locais.

<b>Nome do dataset:</b> outros_atributos <b>Descrição:</b> Feriados nacionais decretados oficialmente pelo Governo do Brasil, feriado de Nossa Senhora Aparecida e visitas dos Papas à localidade de Aparecida/SP. <b>Link:</b> <a href="https://github.com/weabreu/projeto-tcc-ml-ia/blob/main/dataset/outros_atributos.csv">https://github.com/weabreu/projeto-tcc-ml-ia/blob/main/dataset/outros_atributos.csv</a>		
Nome do Atributo	Descrição	Tipo
data_feriado	Data de ocorrência do Feriado	Data
feriado	Indicador de feriado. Valores: 0 – Não houve feriado; e 1 – Feriado.	Inteiro
doze_outubro	Indicador do Feriado de Nossa Senhora Aparecida	Inteiro
papa_aparecida	Indica se houve visita do Papa em Aparecida: 0 ou 1.	Inteiro

**Quadro 2 – Feriados nacionais oficiais.**

Para utilização desta base, foi realizada a junção (*join*) com a tabela de viagens por meio da coluna data. Quando consolidado a série é agregada por semana ou mês, ou qualquer nível acima do diário, foram somados o quantitativo de feriados compreendido no período de agregação.

Além disso, por se tratar de um lugar de proeminente turismo religioso, foi acrescentada uma base que indica em que data houve visitas dos Papas em Aparecida, visto que a presença das referidas figuras religiosas na localidade foi capaz de mobilizar a movimentação de quantitativos consideráveis de pessoas (DE SOUZA, 2020).

## 5. Processamento/Tratamento de Dados

Para realização deste trabalho, foi utilizado um computador com sistema operacional Windows 11, processador Intel Core i5, memória RAM de 32GB e GPU de 4GB. Os

*scripts* foram escritos em Python 3.9.7, sendo executados localmente em ambiente virtual do Anaconda 2.1.2, utilizando-se bibliotecas *Tensorflow 2.6.0* , *Keras 2.4.3*, *Numpy 1.21.5*, *Pandas 1.4.1*, *Scikit-learn 1.0.2*, dentre outras ferramentas nativas da linguagem.

Os dados obtidos na página de dados abertos do Governo Federal estão segmentados por ano e alguns por ano e mês. Assim, num primeiro momento é necessário realizar a consolidação dos dados em uma única base de dados, como mostrado na Figura 5. Para isso, foram utilizadas as bibliotecas *OS* e *Pandas*. A primeira para varrer a pasta continente e obter os caminhos dos arquivos. Já a segunda para carregar e aglutinando os dados em um único *dataframe* que é armazenado em um arquivo CSV. O tamanho do arquivo ao final do processo foi de, aproximadamente, 503MB, com um total de 4.180.058 de linhas e 11 colunas.

```

1  # Procedimento para realizar a mesclagem dos arquivos
2  def merge_files(
3      input_folder,
4      output_folder,
5      output_file='output_',
6      skiprows=0):
7      # Mesclar os arquivos e exporta para uma pasta local
8      import pandas as pd
9      from os import listdir
10     from os.path import isfile, join
11
12     # obtendo a lista de arquivos na pasta
13     files = [f for f in listdir(input_folder) if isfile(join(input_folder, f))]
14     df_to_export = pd.DataFrame()
15
16     # Varre todos arquivos da pasta
17     for i in range(len(files)):
18         complete_file_path = input_folder + files[i]
19         # carregando o arquivo
20         df = pd.read_csv(
21             complete_file_path,
22             skiprows=skiprows, delimiter=';',
23             encoding="latin-1",
24             error_bad_lines = False
25         )
26         # Apaga colunas Unnamed
27         df.drop(
28             df.columns[df.columns.str.contains(
29                 'Unnamed', case=False)],
30             axis=1, inplace=True)
31
32         # mesclando os arquivos
33         df_to_export = pd.concat([df_to_export, df], ignore_index=True)
34
35     df_to_export.drop_duplicates(keep = 'last', inplace = True)
36     # Exportando arquivo
37     df_to_export.to_csv(
38         output_folder + output_file + '.csv', sep=';',
39         index=None,
40         header=True,
41         encoding="utf-8"
42     )

```

**Figura 5 – Consolidação dos arquivos.**

Realizada a consolidação, executa-se um pipeline (Figura 6) que realiza as seguintes tarefas: carregamento dos dados; filtragem do município de destino; agrupamento dos dados por dia; preenchimento de dados faltantes; e enriquecimento dos dados com novos atributos e obtenção da série final para processamento. As funções que compõem este pipeline são explicadas a seguir.

```

1 # Periodicidade da série ('M' - mensal; 'W' - semanal; e 'D' - diária)
2 periodicidade = 'M'
3
4 # Campos de referência das localidades:
5 # - municipio: municipio_origem, municipiodestino; e
6 # - uf: uf_municipio_origem, uf_municipio_destino.
7 campos_cidades = {'cidade': 'municipiodestino', 'uf': 'uf_municipio_destino'}
8
9 # Cidade e UF de referência
10 cidade_uf = {'cidade': 'APARECIDA', 'uf': 'SP'}
11
12 # Data de referência (data_inicio_viagem; data_fim_viagem)
13 data_ref = 'data_fim_viagem'
14
15 # Caminho completo do arquivo
16 arquivo = '../dataset/raw/data_fret.csv'
17 data = (
18     load_dataset(arquivo) # carregamento dos dados
19     .pipe(filter_local, campos_cidades, cidade_uf) # filtragem do local de referência
20     .pipe(groupby_day, data_ref) # agrupamento pela data de referência
21     .pipe(fill_missing_dates) # preenchimento das datas faltantes
22     .pipe(get_series, periodicidade) # obtém a série temporal final
23 )

```

**Figura 6 – Pipeline de tarefas a serem executadas.**

Os dados armazenados em um arquivo CSV, delimitado por “;” é lido e carregado em *dataframe* da biblioteca Pandas. O parâmetro *dayfirst* defino o formato da data para DD/MM e *parse\_date* transforma as colunas da lista em *date\_column* no formato *datetime*.

```

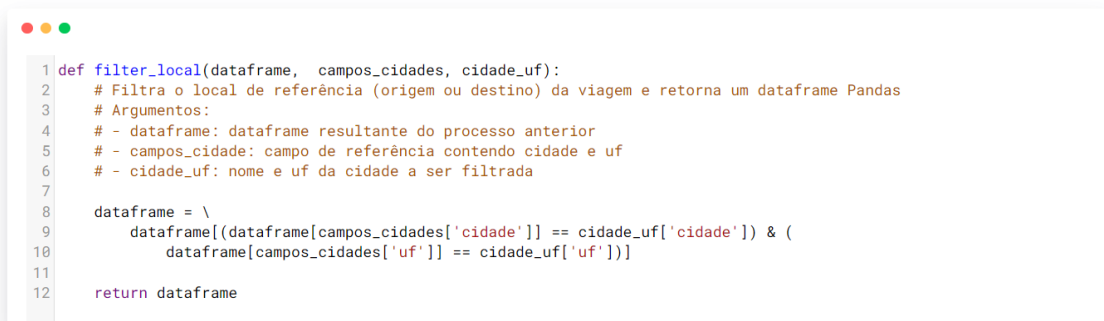
1 def load_dataset(file_path):
2     # Carrega o dataset e retorna um dataframe Pandas
3     # Argumento:
4     # - file_path: caminho do arquivo a ser carregado
5
6     date_column = ['data_inicio_viagem', 'data_fim_viagem']
7     dataframe = pd.read_csv(
8         file_path, engine='python', delimiter=';',
9         parse_dates=date_column, dayfirst=True)
10    dataframe.drop_duplicates(inplace=True)
11
12    return dataframe

```

**Figura 7 – Carregamento dos dados.**

O foco principal do programa é prever o quantitativo de pessoas que se movimentaram para a cidade de destino por meio dos serviços de ônibus fretados. Então, faz-se

necessário filtrar os dados pelo município e Unidade da Federação de destino, modificando-se o *dataframe* original.



```

1 def filter_local(dataframe, campos_cidades, cidade_uf):
2     # Filtra o local de referência (origem ou destino) da viagem e retorna um dataframe Pandas
3     # Argumentos:
4     # - dataframe: dataframe resultante do processo anterior
5     # - campos_cidade: campo de referência contendo cidade e uf
6     # - cidade_uf: nome e uf da cidade a ser filtrada
7
8     dataframe = \
9         dataframe[(dataframe[campos_cidades['cidade']] == cidade_uf['cidade']) & (
10             dataframe[campos_cidades['uf']] == cidade_uf['uf'])]
11
12     return dataframe

```

**Figura 8 – Filtro de município de destino.**

Como os dados estão registrados por viagem, para se obter a série temporal, foi necessário, em primeiro lugar, agrupar os dados por data. Antes disso, alterou-se a data de referência, já que se almejava obter o quantitativo de pessoas na localidade de destino em determinado dia. Como a viagem é em circuito fechado, o campo **data\_fim\_viagem** não representa a data no local de destino, mas sim a data em que o ônibus chega novamente à origem. Isto posto, pode-se supor que, no meio do intervalo entre a data de início e a data de término, os passageiros transportados ainda estivessem no município de destino da viagem. Assim, adotou-se como data de referência a data média entre essas duas. A partir dessa convenção, a série foi obtida somando-se os passageiros transportados em todas as viagens naquele na data de referência (Figura 9). Para preencher os valores de datas que poderiam estar faltando no intervalo, chamou-se a função **interpolate**, utilizando-se o método “time” que interpola um valor em um determinado intervalo de tempo em séries diárias ou de maior resolução (Pandas, 2022).



```

def groupby_day(dataframe, date_input):
    # Agrupa os dados por dia de acordo com o campo de data de final de referencia e
    # retorna um dataframe
    # Argumentos:
    # - dataframe: dataframe resultante do processo anterior
    # - date_input: data de entrada (data_inicio_viagem, data_fim_viagem)

    # Se a date_input for igual a data_inicio_viagem a serie é obtida a série com base na origem da viagem

    # se a date_input for posterior a data_inicio_viagem, a série obtida baseia-se no destino da viagem

    # Transforma data_inicio_viagem e date_input em datetime
    dataframe['data_inicio_viagem'] = pd.to_datetime(
        dataframe['data_inicio_viagem'], infer_datetime_format=True)
    dataframe[date_input] = pd.to_datetime(
        dataframe[date_input], infer_datetime_format=True)

    # Obtém a data de referência
    dataframe['dias'] = \
        (dataframe[date_input] - dataframe['data_inicio_viagem']).dt.days

    # Obtém a data no local de referencia
    dataframe['dias'] = dataframe['dias'].apply(lambda x: math.ceil(x/2))
    dataframe['data_ref'] = dataframe['data_inicio_viagem'] + \
        pd.to_timedelta(dataframe['dias'], unit='d')

    # Agrupa a série pela data de referência
    dataframe = dataframe.groupby(
        'data_ref', group_keys=False)['total_passageiros'].sum().reset_index()
    dataframe['index'] = dataframe['data_ref']
    dataframe = dataframe.set_index(['index'])

    # Preenche a série diária com as datas faltantes no
    # intervalo de datas fornecidas pelos dados
    agg = {'total_passageiros': 'sum'}
    dataframe = dataframe.resample('D', on='data_ref').agg(agg)
    dataframe['total_passageiros'][dataframe['total_passageiros']==0] = np.nan
    dataframe = dataframe.interpolate(method='time').astype(int)

    return dataframe

```

**Figura 9 – Agrupamento por período.**

Antes de prosseguir, foi verificado se havia datas faltantes no intervalo de tempo considerado. Para tal, ordenou-se as datas em ordem crescente, sendo calculada a diferença em dias entre duas datas consecutivas, percorrendo-se todo o *dataset*. Se houvesse uma diferença superior a 1 entre duas datas, isto significou que estava faltando uma data na série. Particularmente, não foram constatadas as faltantes, visto que a função retornou uma lista vazia, como mostrado na Figura 10.

```

def checking_missing_dates(dataframe):
    # Checa se estão faltando datas no intervalo em estudo
    # e retorna um dataframe
    # Argumentos:
    # - dataframe: dataframe resultante do processo anterior

    date_series = pd.to_datetime(
        dataframe.index.to_series(), infer_datetime_format=True)

    # Ordena as datas em ordem crescente
    date_series = date_series.sort_values(ascending=True)
    # Verifica se a diferença entre uma data e a seguinte é superior a 1
    # - se sim então está faltando uma data no intervalo
    # - se não, então não está faltando data
    date_series_missing = date_series[date_series.diff().dt.days > 1]

    print(f'Datas Faltantes\nShape: {date_series_missing.shape}')
    print(date_series_missing)
    del date_series_missing

    return dataframe

```

**Figura 10 – Checagem de datas faltantes.**

Quando se fala de turismo e viagens por ônibus, o dia da semana que a viagem ocorre importa, assim como fato de a data da viagem ser um feriado ou não. Dessa forma, foram acrescentados à série, uma variável *dummy* para indicar se a data se trata de um feriado, sendo 1 para ocorrência de feriado e 0 para ausência. Este atributo, por sua vez, fora armazenado no arquivo *feriados\_nacionais.csv*.

```

1 def including_new_features(dataframe):
2     # Inclui os atributos de feriado disponíveis no arquivo feriados_nacionais.csv e
3     # retorna um dataframe Pandas
4     # Argumentos:
5     # - dataframe: dataframe resultante do processo anterior
6
7     # Carrega o arquivo contendo as datas com os feriados nacionais no período
8     df_feriados = pd.read_csv(
9         '../../../dataset/raw/feriados_nacionais.csv',
10        engine='python',
11        delimiter=';')
12     dataframe['data_ref'] = dataframe.index.to_series()
13
14     # Transforma data_feriado em datetime
15     df_feriados['data_feriado'] = pd.to_datetime(
16         df_feriados['data_feriado'], format='%d/%m/%Y')
17
18     # Mescla os dataframes a partir da data de feriado
19     dataframe = pd.merge(
20         dataframe,
21         df_feriados,
22         left_index=True, right_on='data_feriado').drop(columns=['data_feriado'])
23     del df_feriados
24     dataframe = dataframe.set_index(['data_ref'])
25
26     return dataframe

```

**Figura 11 – Incluindo novos atributos.**

Finalmente, a série final é obtida de acordo com a periodicidade especificada (diária, semanal ou mensal), sendo carregada em um *dataframe*. A série mensal foi indexada

pelo campo derivado **ano\_mes**. Enquanto, as séries diárias e semanais foram indexadas pelas datas de referência. No caso da série diária, como uma forma de enriquecimento da base de dados, foi acrescentado o dia da semana (domingo, segunda-feira etc.), visto que o volume de passageiros transportados pode ser maior ou menor se o dia da semana for uma segunda-feira ou sexta-feira, por exemplo.

```

1 def get_series(dataframe, periodicidade='D'):
2     # Obtém a série temporal enriquecida e agregada por uma periodicidade e
3     # retorna um dataframe Pandas
4     # Argumentos:
5     # - dataframe: dataframe resultante do processo anterior
6     # - periodicidade: 'D' - diária (default), 'W' - semanal, 'M' - mensal
7
8     # Acrescenta uma coluna com uma nova feição: se a data é feriado ou não
9     dataframe = including_new_features(dataframe)
10
11    # Transforma a coluna de data de referência para o formato datetime
12    dataframe['data_ref'] = pd.to_datetime(
13        dataframe.index.to_series(), infer_datetime_format=True)
14
15    # Define os atributos de agregação
16    agg = {'total_passageiros': 'sum', 'feriado': 'sum'}
17    # Obtém o dataframe agregado de acordo com a periodicidade definida
18    dataframe = dataframe.resample(periodicidade).agg(agg)
19
20    # Se a periodicidade for diária, transforma o índice em uma coluna datetime e
21    # obtém o dia da semana (0-domingo, 1-segunda-feira, 2-terça-feira, 3-quarta-feira,
22    # 4-quinta-feira, 5-sexta-feira, 6-sábado)
23    if periodicidade == 'D':
24        date_series = pd.to_datetime(
25            dataframe.index.to_series(), infer_datetime_format=True)
26        dataframe['dia_semana'] = date_series.dt.dayofweek
27
28
29    # Se periodicidade é mensal, obtém a série configurando o índice como ano/mês
30    if periodicidade == 'M':
31        dataframe = dataframe.set_index(dataframe.index.strftime('%Y/%m'))
32
33    return dataframe

```

Figura 12 – Obtendo a série temporal final.

## 6. Análise e Exploração dos Dados

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

## 8. Aplicação de Modelos de Aprendizado de Máquina

## 9. Discussão dos Resultados

## 10. Conclusão

## 11. Links

## 12. Referências

CARMO, H. de O. e VALENTE, T. C. O. **Características dos atendimentos a romeiros no Santuário de Nossa Senhora Aparecida, São Paulo, 2011- 2014**. jul-set 2017.

BRASIL. **Decreto nº 2.521, de 20 de março de 1998**. Dispõe sobre a exploração, mediante permissão e autorização, de serviços de transporte rodoviário interestadual

e internacional de passageiros e dá outras providências. 1998. Disponível em: [http://www.planalto.gov.br/ccivil\\_03/decreto/d2521.htm](http://www.planalto.gov.br/ccivil_03/decreto/d2521.htm). Acesso em: 01 mai. 2022

\_\_\_\_\_. **Portal Brasileiro de Dados Abertos**. <https://dados.gov.br/dataset/licencas-de-viagem-nacional-servico-fretado>. Acesso em: 01 mai. 2022.

Agência Nacional de Transportes Terrestres (ANTT). **Resolução nº 4.777 de 6 de julho de 2015**. Dispõe sobre a regulamentação da prestação do serviço de transporte rodoviário coletivo interestadual e internacional de passageiros realizado em regime de fretamento. Disponível em: [antt.gov.br](http://antt.gov.br). Acesso em: 01 mar. 2022.

BRASIL.

ORTEGA, I. M., JDID, L., BRITO, M. R., SALLES, M. R. **Turismo religioso em Aparecida do Norte, SP: infraestrutura de hospedagem do ponto de vista do visitante**. Revista de Investigación em turismo y desarrollo local 6.14. 1-22. 2013.

Zhengping CHE, Sanjay PURUSHOTHAM, Kyunghyun CHO, David SONTAG & Yan LIU. **Recurrent Neural Networks for Multivariate Time Series with Missing Values**. Disponível em: NATURE. <https://www.nature.com/articles/s41598-018-24271-9.pdf>. Acesso em: 17 abr. 2018.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. MIT Press, 2016. Disponível em: <https://www.deeplearningbook.org>. Acesso em: 10 jan. 2022.

DATA SCIENCE ACADEMY. **Deep Learning Book**, 2022. Disponível em: <https://www.deeplearningbook.com.br/>. Acesso em: 10 jan. 2022.

LIU, S., YANG, N., LI, M. AND ZHOU, M. **A recursive recurrent neural network for statistical machine translation**. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics. vol 1. 2014. p. 1491-1500.

HOCHREITER, S. e SCHMIDHUBER, J. **Long short-term memory**. Neural computation, vol. 9(8), 1997. p.1735-1780.

GÉRON, A. **Neural networks and deep learning**. Chapter 4. Recurrent Neural Networks. 2018. Disponível em: <https://www.oreilly.com/library/view/neural-networks-and/9781492037354/ch04.html>. Acesso em: 22 mai. 2022.

\_\_\_\_\_. **Mãos à Obra: Aprendizado de Máquina com Scikit-Learn & TensorFlow**. Rio de Janeiro - Alta Books, 2021.

GREFF K, SRIVASTAVA RK, KOUTNÍK J, STEUNEBRINK BR, SCHMIDHUBER J. **LSTM: A search space odyssey**. IEEE transactions on neural networks and learning systems. 2016.

YU Z, NIU Z, TANG W, WU Q. **Deep learning for daily peak load forecasting—a novel gated recurrent neural network combining dynamic time warping.** IEEE Access. vol. 7. 2019. p.17184-17194.

OLAH, C. **Understanding LSTM Networks.** 2015. Disponível em: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>. Acesso em: 21 mai. 2022.

LV W, LV Y, OUYANG Q, REN Y. **A Bus Passenger Flow Prediction Model Fused with Point-of-Interest Data Based on Extreme Gradient Boosting.** Applied Sciences. vol. 12 (3). 2022.

**Pandas.** Disponível em: <https://pandas.pydata.org/>. Acesso em: 27 mai.2022.