Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

# Modelos para Previsão Tributária Utilizando Redes Neurais LSTM

**Arthur de Freitas Dornelas**

JUIZ DE FORA
AGOSTO, 2022

# Modelos para Previsão Tributária Utilizando Redes Neurais LSTM

Arthur de Freitas Dornelas

Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Orientador: Luciana Conceição Dias Campos
Coorientador: Karla Tereza Figueiredo Leite

JUIZ DE FORA
AGOSTO, 2022

# Modelos para Previsão Tributária Utilizando Redes Neurais LSTM

Arthur de Freitas Dornelas

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Luciana Conceição Dias Campos
Doutora em Engenharia Elétrica

Karla Tereza Figueiredo Leite
Doutora em Engenharia Elétrica

Luciano Jerez Chaves
Doutor em Ciência da Computação

Heder Soares Bernardino
Doutor em Modelagem Computacional

JUIZ DE FORA
11 DE AGOSTO, 2022

*A minha família que sempre foi a base de tudo que realizei.*

*Aos meus amigos que sempre estiveram comigo nessa jornada.*

*As minhas orientadoras por serem uma constante fonte de motivação e incentivo ao longo de todo o projeto.*

# Resumo

O ICMS (Imposto sobre Circulação de Mercadorias e Prestação de Serviços de Transporte Interestadual e Intermunicipal e de Comunicação) é um dos principais impostos arrecadados pelos estados brasileiros, sendo seu valor importante na gestão e planejamento do governo, em especial para o estado do Rio de Janeiro, que se apresenta em crise econômica e desde o ano 2020 está em Regime de Recuperação Fiscal, necessitando de uma constante atualização da previsão de seus valores de receita e gastos. Devido às incertezas e mudanças externas e internas no estado carioca, a previsão desse valor coletado possui característica de não-linearidade, sendo necessário a aplicação de modelos não lineares que possam considerar essas mudanças nos valores arrecadados ao longo do tempo. Por conseguinte, o trabalho aqui descrito visa utilizar modelos de Redes Neurais Recorrentes *Long Short-Term Memory* (LSTM) e comparar as abordagens *Multivariate Multi-step* e *Univariate Multi-step*, na tentativa de gerar uma previsão anual da arrecadação tributária do estado superior à de outras abordagens, podendo ser utilizados como parâmetros para a tomada de decisões das autoridades governamentais.

**Palavras-chave:** ICMS, Machine Learning, Redes Neurais, Séries Temporais, Long Short-Term Memory, Multivariate, Multi-step.

# Abstract

ICMS (Imposto sobre Circulação de Mercadorias e Prestação de Serviços de Transporte Interestadual e Intermunicipal e de Comunicação) is one of the main taxes collected by Brazilian states, being its important value in the management and planning of the government, especially for the state of Rio de Janeiro, that presents itself in economic crisis and since the year 2020 is in a Tax Recovery Regime. Due to uncertainties and external and internal changes in the state, the forecast of this collected value has a characteristic of non-linearity, being necessary the application of non-linear models that can consider these changes in the amounts collected over time. Therefore, the work described here aims to use models of Recurrent Neural Networks *Long Short-Term Memory* (LSTM) and compare the approaches *Multivariate Multi-step* and *Univariate Multi-step*, in an attempt to generate an annual forecast of the state's tax collection that is superior to other approaches, and can be used as parameters for decision-making by government authorities.

**Keywords:** ICMS, Machine Learning, Neural Networks, Time Series, Long Short-Term Memory, Multivariate, Multi-step.

# Conteúdo

# Lista de Figuras

# Lista de Tabelas

# Lista de Abreviações

| | |
|---|---|
| DCC | Departamento de Ciência da Computução |
| UFJF | Universidade Federal de Juiz de Fora |
| ICMS | Imposto sobre Circulação de Mercadorias e Serviços |
| PIB | Produto Interno Bruto |
| LSTM | Long Short-Term Memory |
| MTL | Multitask Learning |
| ADAM | Adaptive Moment Estimation |
| SGD | Stochastic Gradient Descent |
| SARIMA | Seasonal Autoregressive Integrated Moving Average |
| ARIMA | Autoregressive Integrated Moving Average |
| VAR | Vector Autoregression |

# 1 Introdução

A organização e planejamento nas contas públicas é essencial para o bom funcionamento do estado. A Lei de Responsabilidade Fiscal [Lei Complementar nº 101, de 04/05/2000] (PLANALTO, 2000) propõe parâmetros e condições para os gastos públicos dos estados e municípios com o objetivo de que seja possível evitar situações econômicas indesejadas que possam tornar as entidades não funcionais financeiramente, além de proporcionar a herança administrativa para as próximas gestões. Caso ocorram infrações quanto à lei, sanções e penalidades podem ser aplicadas, como multas, cassação de mandatos e até reclusão dos responsáveis.

Para realizar o planejamento fiscal, de acordo com o Governo, deve-se estimar as receitas, que são os recursos financeiros arrecadados que englobam impostos, taxas, contribuições, entre outros, e as despesas, que são todos os gastos realizados, seja na aquisição de produtos, serviços, obras ou compras, sendo eles fixados para garantir que não ocorra mais gasto do que arrecadação. Todo o planejamento é considerado complexo e tem diversas variáveis, e de acordo com o Portal da Transparência, envolve várias etapas, com destaque a aprovação da Lei do Plano Plurianual (PPA), da Lei de Diretrizes Orçamentárias (LDO) e da Lei Orçamentária Anual (LOA).

Em questão das receitas, o Imposto sobre Circulação de Mercadorias e Prestação de Serviços de Transporte Interestadual e Intermunicipal e de Comunicação (ICMS), que foi regulamentado pela Lei Kandir (PLANALTO, 1996), é um tributo estadual que incide na circulação de produtos e serviços tributáveis nas cidades, estados ou de pessoas jurídicas para pessoas físicas. É o principal meio de arrecadação de tributos do estado, sendo parte principal no montante arrecadado das receitas e consequentemente de extrema importância para o planejamento e organização da gestão (SEFAZ-RJ, 2022).

## 1.1 Motivação

Em 2020, o estado do Rio de Janeiro entrou no Regime de Recuperação Fiscal [Resolução CSRRF nº33 2020] (SEFAZ-RJ, 2020), possibilitando o estado que está em um momento de grave crise financeira a reajustar suas contas na busca de tornar-se novamente sustentável. Uma das necessidades desse regime é a constante atualização dos dados de receitas e despesas do estado, de forma que seja possível acompanhar a evolução da recuperação.

Por conseguinte, possuir uma forma melhor de realizar a previsão da arrecadação do ICMS, através da análise das séries temporais, com uma menor taxa de erro é de suma importância para o estado, sendo providencial para a tomada de decisões e ações que têm impacto na vida dos contribuintes e na organização da gestão.

Devido a mudança contínua do ambiente externo e interno do estado, existe uma não-linearidade no problema, o que invalida uma série de modelos estatísticos-matemáticos que são propostos para problemas que tem critério de linearidade. Portanto, a utilização desses modelos não-lineares de *Machine Learning*, como as Redes Neurais Artificiais, podem apresentar resultados mais eficientes pois esses modelos conseguem capturar as mudanças não lineares dos dados analisados.

## 1.2 Objetivos

Este trabalho tem como objetivo principal ajustar modelos de previsão utilizando séries temporais para realizar a previsão anual do valor arrecadado do ICMS no estado do Rio de Janeiro.

Em segundo plano, o trabalho visa treinar e avaliar modelos de Redes Neurais LSTM (*Long Short-Term Memory*) com apenas uma variável (*Univariate*) e no contexto de múltiplas variáveis (*Multivariate*). Os melhores resultados obtidos de cada modelo são comparados com o atual método de previsão utilizado pelo governo do estado, sendo possível avaliar a capacidade das soluções utilizando Redes Neurais LSTM em relação às outras abordagens, e consequentemente, gerando uma forma de melhorar as previsões e informações disponíveis para os órgãos governamentais.

## 1.3 Organização do Trabalho

O trabalho está organizado da seguinte forma, o Capítulo 2 oferece referencial teórico para o conteúdo abordado pelo trabalho, o Capítulo 3 apresenta a metodologia que foi aplicada para a parte prática, o Capítulo 4 mostra o estudo de caso para o problema de previsão anual do ICMS, com os cenários de teste e resultados. Por fim o Capítulo 5 apresenta a conclusão e a sugestão de trabalhos futuros.

# 2 Fundamentação Teórica

Este capítulo apresenta os conceitos fundamentais para o melhor entendimento do trabalho e das técnicas utilizadas em sua proposta e estudo de casos. Inicia-se com a apresentação do conceito de Redes Neurais Artificiais, os tipos de arquitetura, treinamento e aprendizado. Após isso será explicado o conceito de LSTM e suas classificações quanto a dados de entrada e horizontes de previsões. Por fim, será apresentado o conceito de ICMS, em específico no estado do Rio de Janeiro, e como é feito a previsão atualmente.

## 2.1 Redes Neurais Artificiais

Rede Neural Artificial (RNA) tem como objetivo modelar o funcionamento do cérebro humano de forma digital, na tentativa de atingir a capacidade de processamento paralelo, não linear e altamente complexo do órgão animal (HAYKIN, 1999).

### 2.1.1 Neurônio Artificial

Nas redes neurais artificiais utiliza-se a interconexão dos chamados neurônios artificiais, que são inspirados nos neurônios biológicos, em que o esquema simplificado é apresentado na Figura 2.1. De acordo com Zsolt (2006), a célula do sistema nervoso é responsável por transmitir os impulsos nervosos ao cérebro. Ela é formada: pelo corpo celular que acomoda o núcleo e as organelas celular; os dendritos, por onde os impulsos nervosos são recebidos, ou seja a entrada; o axônio, que é único e tem a função de conduzir os impulsos elétricos.

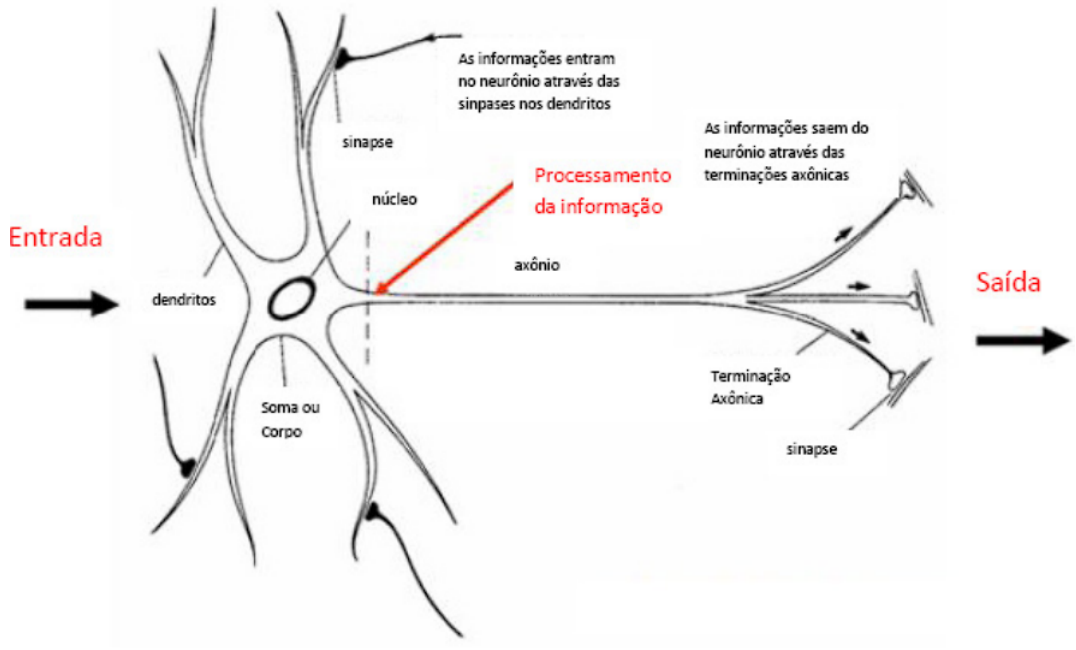


Figura 2.1: Modelo simplificado de neurônio biológico (CAMPOS, 2010).

O cérebro humano possui cerca de 86 bilhões de neurônios e a interligação entre eles é efetuada pela sinapse, que é a saída do axônio e que se conecta ao dendrito de outro neurônio. Essa conexão vai se formando até atingir uma gigantesca rede de impulsos elétricos que geram tráfego de informação. São nessas ligações que acredita-se que ocorra o aprendizado.

A tradução do neurônio biológico para o neurônio artificial pode ser observada na Figura 2.2. Na figura é possível observar que a adaptação dos dendritos do neurônio biológico para o neurônio artificial é o conjunto de entradas $x_1, \ldots, x_m$, responsáveis por receber os dados. As entradas são ponderadas pelos chamados pesos sinápticos, na imagem representados por $w_{i,1}, \ldots, w_{i,m}$, que são os incumbidos de guardar conhecimento por experiência. O corpo celular ou núcleo é apresentado pelo $\sum$ e tem a função de ser o processador interno, fornecendo como resultado o $net_i$; a tradução do axônio para o meio digital é a saída $y_i$.

De acordo com Haykin (2001), para modificar a influência do valor da combinação linear das entradas, utiliza-se o bias, que é representado por $\theta_i$. A função de ativação, representada por $\varphi$, é responsável por limitar a amplitude da saída de um neurônio e introduzir não linearidade ao modelo, de acordo com Barron (1993). A saída do neurônio é observada na equação 2.1, onde $i$ refere-se ao neurônio.

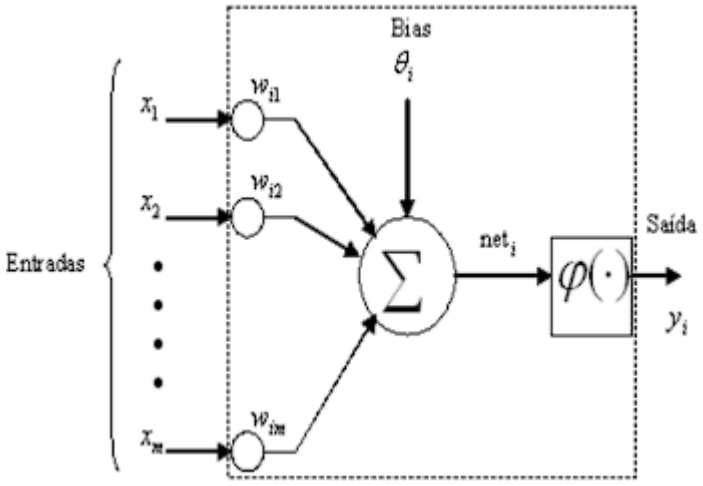


Figura 2.2: Esquema de um neurônio artificial (CAMPOS, 2010).

$$y_i = \varphi(net_i) = \varphi(\sum_{j=1}^{m} x_j \times w_{i,j} + \theta_i) \tag{2.1}$$

Entre seus principais poderes temos a capacidade de ser paralelamente distribuído, devido a organização em camadas e sua capacidade de generalização, que indica que ela consegue produzir saídas razoáveis para entradas não executadas em seu treinamento (HAYKIN, 1999).

### 2.1.2 Funções de ativação

Algumas funções de ativação são:

1. **Função limiar**

$$y_i = \begin{cases} 1 & net_i \geq 0 \\ 0 & net_i < 0 \end{cases} \tag{2.2}$$

Segundo Haykin (2001), nessa função, a saída do neurônio será 1 quando o valor é maior ou igual a 0, e 0 quando o valor é negativo.

2. **Função linear**

$$y_i = \varphi(net_i) = net_i \tag{2.3}$$

Utilizada quando a saída do neurônio não tem um valor limite específico, portanto pode ser de qualquer valor.

3. **Função sigmóide**

$$y_i = \frac{1}{1 + \epsilon^{-\alpha \cdot net_i}} \tag{2.4}$$

onde $\alpha$ é um valor real.

Ela assume sempre valores positivos e segundo Haykin (2001), ao contrário da função limiar que assume valores de 0 ou 1, a função sigmóide assume um intervalo contínuo de valores entre 0 e 1. Se for o caso de utilizar um intervalo entre -1 e 1, a equação será dada pela função tangente hiperbólica, mostrada na equação 2.5.

$$y_i = tanh(\alpha \cdot net_i) \tag{2.5}$$

4. **Função Relu**

$$y_i = max(0, net_i) \tag{2.6}$$

Abreviação de unidade linear retificada, de acordo com Nwankpa et al. (2018), a função assume o valor 0 para todas as entradas negativas e o próprio valor para as entradas positivas. Portanto, é uma função computacionalmente leve em relação a sigmóide, já que não é necessário realizar operações matemáticas.

### 2.1.3 Topologia de Redes Neurais

A arquitetura de uma rede neural é a estrutura ao qual seus neurônios estão dispostos, de acordo suas camadas e a direção em que ocorre a projeção das saídas. A escolha dela é muito importante para garantir uma boa aprendizagem e generalização, além da adequação ao problema e necessidades em relação ao tempo de execução (PRUDENCIO, 2002).

1. **Redes Neurais Não Recorrentes**

   É a rede neural que é estritamente alimentada adiante ou acíclica. Elas não possuem ligações entre os neurônios de mesma camada, camadas anteriores ou camadas que não são subsequentes. Portanto ela possui apenas uma direção, sendo os neurônios produtos das camadas anteriores ao mesmo. Ela pode ser também nomeada como camada única, onde só existe a camada de saída dos neurônios, como mostra a Figura 2.3, ou múltiplas camadas, que possui uma ou mais camadas ocultas, ou seja, camadas intermediárias entre os neurônios de entrada e saída, de acordo com a Figura 2.4 (HAYKIN, 2001).

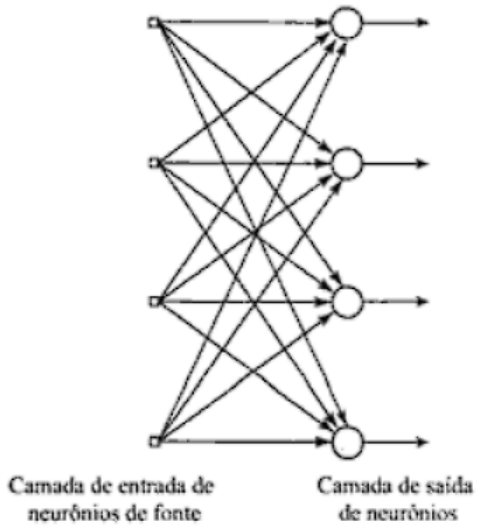


Figura 2.3: Exemplo de rede neural não recorrente de camada única (HAYKIN, 2001).

2. **Redes Neurais Recorrentes**

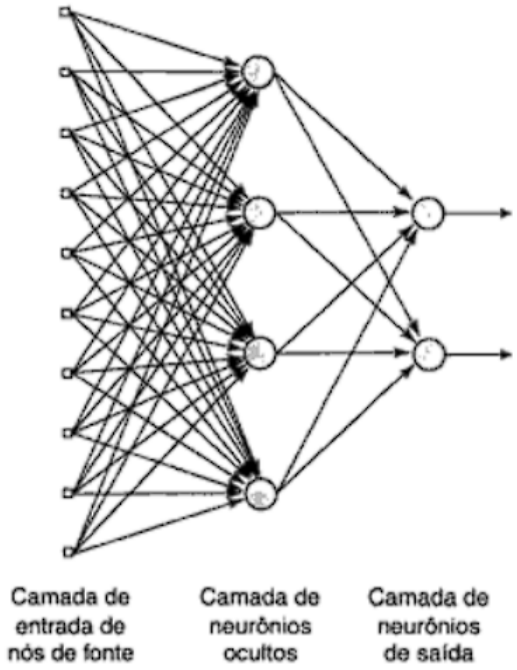


Figura 2.4: Exemplo de rede neural não recorrente de múltiplas camadas (HAYKIN, 2001).

São as redes neurais que possuem realimentação entre neurônios de mesma camada ou camadas não subsequentes. Como mostra a Figura 2.5, essa rede possui maior complexidade e isso causa uma melhora na capacidade de aprendizagem, porém piora o desempenho da mesma, sendo os valores de saída das camadas dependentes de suas entradas e as saídas anteriores (HAYKIN, 2001).

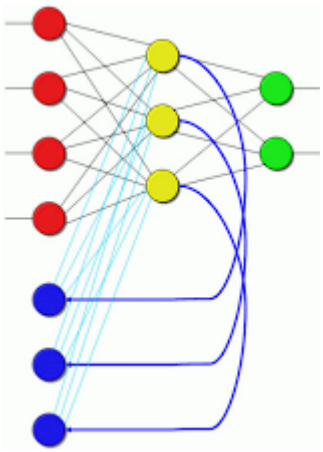

Figura 2.5: Exemplo de rede neural recorrente (CAMPOS, 2010).

### 2.1.4 Tipos de Treinamento

Como o aprendizado das Redes Neurais Artificias é dependente dos dados de entrada, é necessário realizar a escolha correta do conjunto que será disposto para treinamento e tipo de treinamento realizado.

Dessa forma, de acordo com Zsolt (2006), os tipos de aprendizado podem ser separados em dois:

1. **Treinamento Supervisionado:** É disponibilizado para a rede na etapa de treinamento o conjunto de dados de entrada e as saídas desejadas para esse conjunto. Assim é possível comparar a saída desejada com a saída prevista pela rede. Essa comparação é calculada através da diferença entre elas, que é chamada de erro. O erro é utilizado pelo algoritmo de aprendizado para a alteração dos pesos sinápticos e propagação para a rede, na tentativa de diminuí-lo e atingir uma maior assertividade do modelo (ZSOLT, 2006).

2. **Treinamento Não Supervisionado:** É disponibilizado para a rede apenas os dados de entrada. Portanto, não é possível calcular o erro entre a saída prevista e a saída desejada. O treinamento é feito através de análises estáticas dos padrões das entradas, de forma que seja possível gerar agrupamentos de semelhanças, diferenças ou associações como saída do modelo (ZSOLT, 2006).

### 2.1.5 Aprendizado

Para a aprendizagem da rede neural é utilizado o algoritmo de treinamento chamado de *backpropagation* ou retropropagação de erro. Esse algoritmo consiste no cálculo do erro de saída da rede e a retropropagação do mesmo, atualizando os pesos sinápticos das camadas, a partir da saída em direção à entrada, através da técnica de busca do gradiente decrescente dos erros, que é um algoritmo que tem como estratégia a tentativa de minimização da função dos erros (HAYKIN, 2001).

O algoritmo de backpropagation consiste em duas etapas principais. A primeira etapa consiste na propagação da entrada da rede neural por suas camadas até a produção da saída, em que os pesos são fixos. A partir do resultado da camada de saída, o erro

é calculado através da Equação 2.7, em que resposta do neurônio $i$, representada por $y_i$, subtraí a saída deseja para a entrada, representada por $\tilde{y}_i$, gerando assim o erro $e_i$, sendo $n$ o padrão de dados do treinamento proposto (CAMPOS, 2010).

$$e_i(n) = y_i(n) - \tilde{y}_i(n) \tag{2.7}$$

Após isso, ocorre a etapa do *backpropagation*, ou retropropagação do erro, em que o erro encontrado na Equação 2.7 é retropropagado pela rede, ocorrendo os ajustes dos pesos sinápticos, onde o novo peso é calculado de acordo com a Equação Simplificada 2.8. O processo ocorre em cada época (ciclo de treinamento) até que a rede neural tenha atingido um critério de parada.

$$novo\_peso = peso - taxa\_de\_aprendizagem * gradiente \tag{2.8}$$

Durante o treinamento da rede neural é importante evitar o chamado *overfitting*, que consiste no momento em que a rede neural deixa de ser capaz de realizar a generalização pelo fato das saídas serem adequadas apenas aos dados do treinamento, e não mais a qualquer conjunto de entrada, mesmo que não estivesse presente no grupo de treino (HAYKIN, 2001).

Portanto, separamos os dados em três conjuntos distintos de forma que seja possível abranger o interesse de generalização da rede neural. Esses conjuntos são formados pelos dados de treinamento, que em geral é o maior conjunto e será ele que a rede usará para atualizar os pesos sinápticos. Em segundo temos os dados de validação, que serão usados para avaliar a capacidade de generalização da rede durante a fase de treinamento. E, por último, temos os dados de teste, que são utilizados para avaliar o modelo após o fim do treinamento da rede.

A partir dessa divisão do conjunto de dados, podemos usar um dos critérios de parada mais utilizados, que é o *Early Stopping*, método ao qual utiliza-se o conjunto de validação ao final de cada época de treinamento para gerar um erro de validação. Esse erro é analisado e caso ele comece a sofrer um desvio padrão (disperção dos dados em relação a média) pode ser um bom indicativo que a rede está perdendo a capacidade

de generalizar e consequentemente ela deve parar o treinamento (HAYKIN, 2001). Esse comportamento pode ser identificado na Figura 2.6.


Figura 2.6: Análise dos erros de validação e treinamento ocasionando um ponto de parada antecipado durante o treinamento da rede neural (HAYKIN, 2001).

### 2.1.6 Aplicação em Séries Temporais

Para a utilização de redes neurais de treinamento supervisionado com séries temporais, é necessário utilizar o processo de janelas deslizantes para organizar as entradas e saídas desejadas.

Como temos uma dependência temporal de épocas passadas que têm influência no futuro, é necessário para prever um instante $t$ as informações acerca dos instantes passados a $t$. Dessa forma, o processo consiste em sempre existir um número pré-definido de instantes passados que serão utilizados para prever um instante $t$ futuro. No momento que o instante t é previsto, ocorre o descarte do primeiro instante utilizado no conjunto dos passados de $t$ e o instante $t$ é adicionado a esse conjunto, de forma que agora seja possível prever o instante $t + 1$.

A Figura 2.7 descreve esse processo e pode-se observar que nesse caso a janela

deslizante possui um intervalo de três instantes passados para a previsão de um instante futuro.

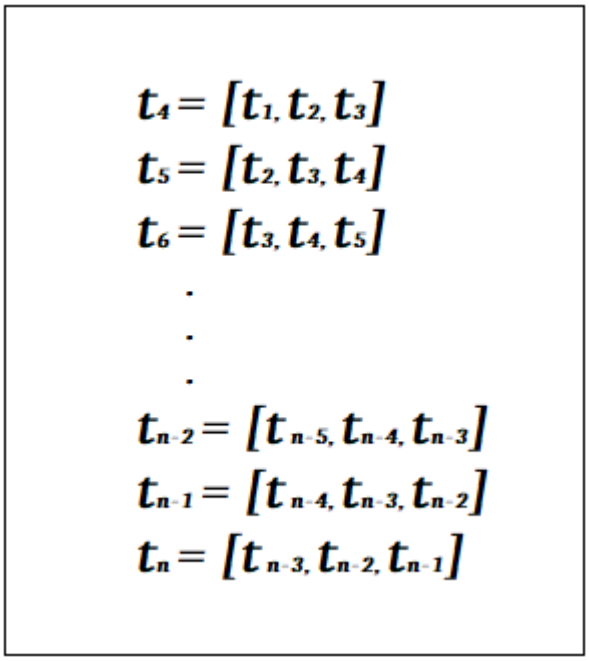


Figura 2.7: Exemplo de janela deslizante. Elaborado pelo autor.

## 2.2 Rede Long Short-Term Memory (LSTM)

A rede neural recorrente tradicional possui um problema em relação às memórias de longo prazo. Em longos intervalos de tempo o custo é muito elevado em questão de tempo e aprendizado. Dessa forma, Hochreiter e Schmidhuber (1997) propuseram o modelo arquitetural para redes neurais recorrentes chamado de Long Short-Term Memory, pautada em um apropriado aprendizado a partir de gradiente, com o propósito de conseguir acessar essas memórias longas.

Na rede recorrente tradicional, o algoritmo de backpropagation, responsável pela alteração dos pesos e consequentemente pelo aprendizado, sofre o "problema do gradiente de fuga". Assim, o gradiente de longo prazo que é usado para atualizar os pesos começa a diminuir, e se esse valor se torna muito pequeno, sua interferência no aprendizado também torna-se muito baixa. Portanto, as informações relacionadas a esses períodos distantes são perdidos, gerando uma alta dependência aos valores mais atuais (HOCHREITER; SCHMIDHUBER, 1997).

Por conseguinte, a LSTM tem como propósito melhorar esse tipo de aprendizado de longo prazo através da utilização de células de memória e portões.

### 2.2.1 Arquitetura

A arquitetura da célula de memória funciona como o caminho para toda a sequência que a informação será processada. Ela apresenta três portões, que são pequenas redes neurais que decidem quais informações terão progresso durante o fluxo na célula. Esses portões são conhecidos como *forget gate*, *input gate* e *output gate* (Figura 2.8).

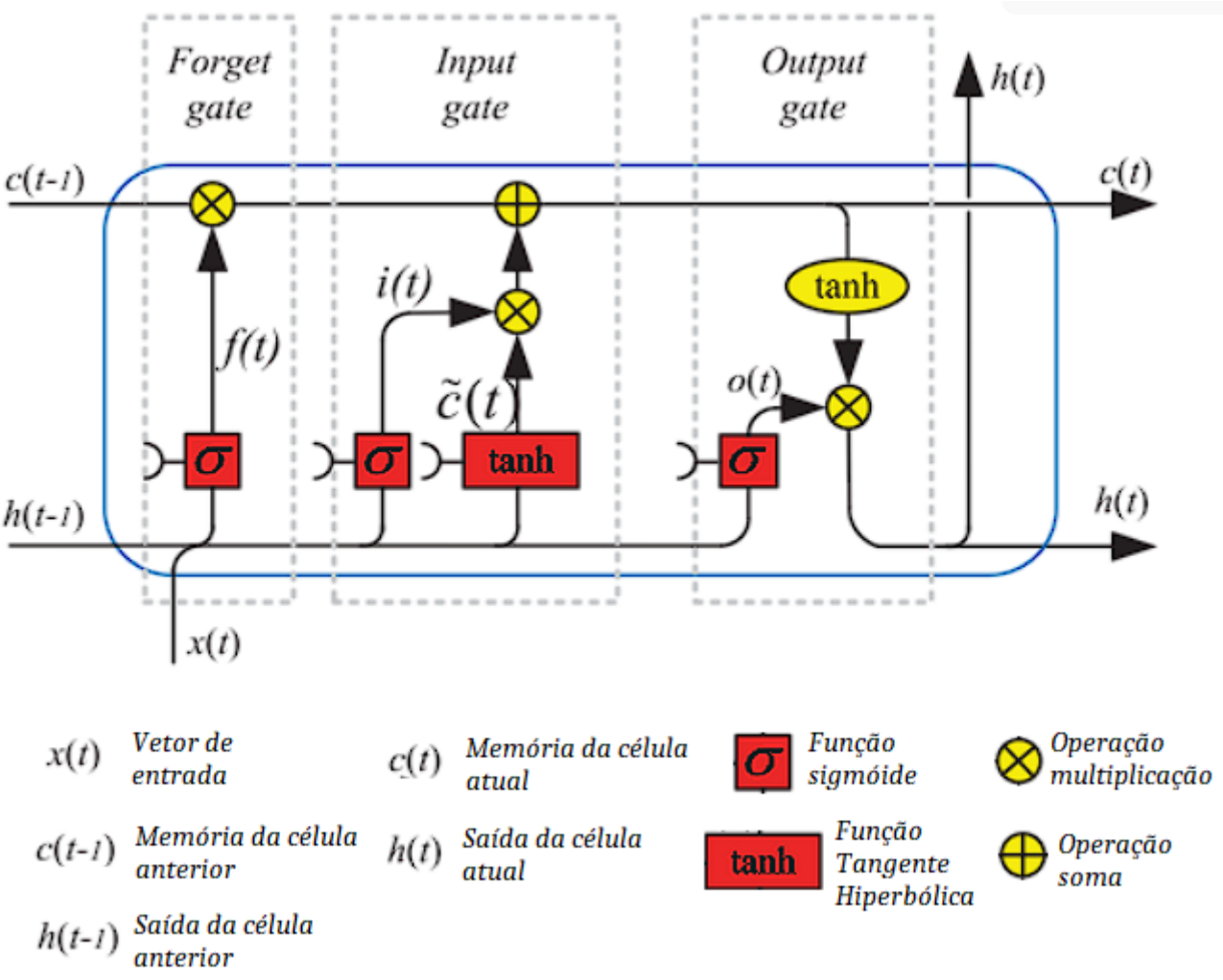


Figura 2.8: Arquitetura célula LSTM. Adaptado de Yu et al. (2019).

O *forget gate* é responsável por avaliar quais informações serão esquecidas ou armazenadas. Recebe como entrada o parâmetro x(t), que é a entrada corrente e h(t-1), que é a saída do estado anterior e o conjunto é passado por uma função sigmóide,

que tem como retorno representado por f(t), sendo um número entre zero e um, em que quanto mais perto de zero significa esquecimento e quanto mais perto de um significa armazenamento.

Já o *input gate* é responsável pela decisão de atualização da nova informação que será armazenada. Novamente, passamos primeiramente os parâmetros x(t) e h(t-1) por uma função sigmóide. Essa etapa tem como importância a decisão de qual a importância do valor na atualização, sendo o resultado da função representado por i(t) e entre zero e um, onde zero significa baixa importância e um significa alta importância. Paralelamente, os parâmetros x(t) e h(t-1) também são passados por uma função tangente hiperbólica, gerando um novo vetor $\tilde{c}(t)$ com possíveis valores de atualização. Fazemos por fim a multiplicação das saídas das duas funções, com finalidade de que o resultado da sigmóide indique qual a importância da informação presente no c(t), que é o resultado da tangente hiperbólica.

Por último temos o *output gate* que é responsável pela geração da saída. Primeiro são passados novamente os parâmetros x(t) e h(t-1) por uma função sigmóide, gerando o(t), cujo resultado é multiplicado pelo resultado da função tangente hiperbólica da soma das informações geradas pelos dois primeiros portões, que é o valor c(t). O produto será representado por h(t), que é o estado escondido. Já o valor de c(t) será o novo valor da célula. Basicamente esse portão é responsável por indicar quais os parâmetros de entrada são mais importantes para a saída.

Em síntese, o *forget gate* é responsável por classificar a informação necessária dos passos anteriores, o *input gate* por decidir qual informação do passo atual deve ser agregada, e o *output gate* tem o papel de selecionar qual o próximo estado escondido. Dessa forma, várias células são conectadas gerando toda a forma de aprendizado da rede.

As equações a seguir mostram os cálculos feitos em cada etapa, onde $x_t$ é o vetor de entrada, $h_{t-1}$ é a saída da célula anterior e $c_{t-1}$ é a memória da célula anterior:

1. **Forget gate**

$$f_t = \sigma(W_f \times h_{t-1} + W_f \times x_t + b_f) \tag{2.9}$$

onde:

$W_f$ = matriz de peso de $f$.

$b_f$ = *bias* de $f$.

$\sigma$ = função sigmóide.

2. **Input gate**

$$i_t = \sigma(W_i \times h_{t-1} + W_i \times x_t + b_i) \tag{2.10}$$

onde:

$W_i$ = matriz de peso de $i$.

$b_i$ = *bias* de $i$.

$\sigma$ = função sigmóide.

$$\tilde{c}_t = tanh(W_{\tilde{c}} \times h_{t-1} + W_{\tilde{c}} \times x_t + b_{\tilde{c}}) \tag{2.11}$$

onde:

$W_{\tilde{c}}$ = matriz de peso de $\tilde{c}$.

$b_{\tilde{c}}$ = *bias* de $\tilde{c}$.

$tanh$ = função tangente hiperbólica.

3. **Atualização célula**

$$c_t = f_t \times c_{t-1} + i_t \times \tilde{c}_t \tag{2.12}$$

4. **Output gate**

$$o_t = \sigma(W_o \times h_{t-1} + W_o \times x_t + b_o) \tag{2.13}$$

onde:

$W_o$ = matriz de peso de $o$.

$b_o$ = *bias* de $o$.

$\sigma$ = função sigmóide.

$$h_t = o_t.tanh(c_t) \tag{2.14}$$

onde:

$tanh$ = função tangente hiperbólica.

### 2.2.2 *Multivariate* x *Univariate*

A rede neural LSTM pode ser separada em duas classificações quanto aos dados inseridos na camada de entrada.

1. ***Univariate*:** Apenas uma variável é inserida na entrada. Todo o aprendizado e análise depende do comportamento da variável e somente dela. O modelo possui a vantagem de não precisar de nenhuma verificação quanto a relacionamento com outras variáveis. Ela pode ter uma ou múltiplas saídas e múltiplas entradas da mesma variável (GUILLÉN-NAVARRO et al., 2020).

   A Figura 2.9 mostra a representação de um modelo LSTM *Univariate*. É possível identificar que a entrada $x_t$ é de apenas uma variável, e a saída obtida $y_t$ e a memória da célula $c_t$ são também entradas para os próximos neurônios LSTM, sendo $t$ um instante.

2. ***Multivariate*:** Duas ou mais variáveis diferentes são inseridas na entrada. Apresenta dependência de relacionamento entre as entradas, sendo importante selecionar

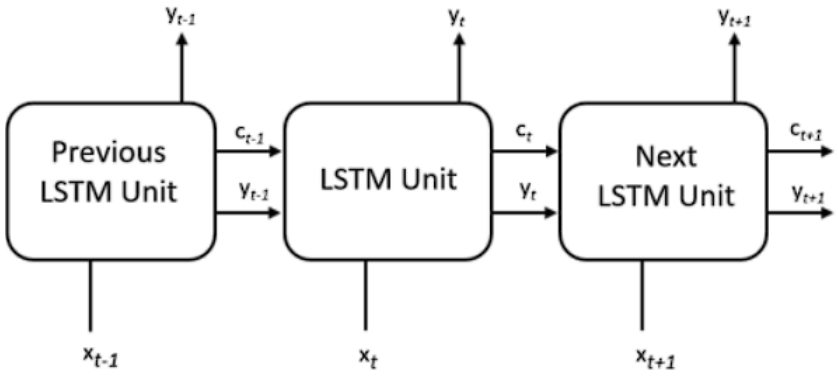


Figura 2.9: Arquitetura LSTM *Univariate* (GUILLÉN-NAVARRO et al., 2020).

variáveis que tenham uma correlação para garantir um melhor aprendizado. Portanto, é necessário um estudo sobre as características de cada uma e o relacionamento entre elas. Pode ter números diferentes de entradas para cada variável, além de uma ou múltiplas saídas (GUILLÉN-NAVARRO et al., 2020).

A Figura 2.10 descreve o funcionamento dos neurônios LSTM nesse modelo. É possível observar que $x_{1_t}, x_{2_t}, x_{n_t}$ representa a entrada de múltiplas variáveis, sendo $n$ o número de variáveis diferentes. Também apresenta a saída única de $y_t$ e a memória $c_t$ que são entradas para o próximo neurônio LSTM, sendo $t$ um instante.

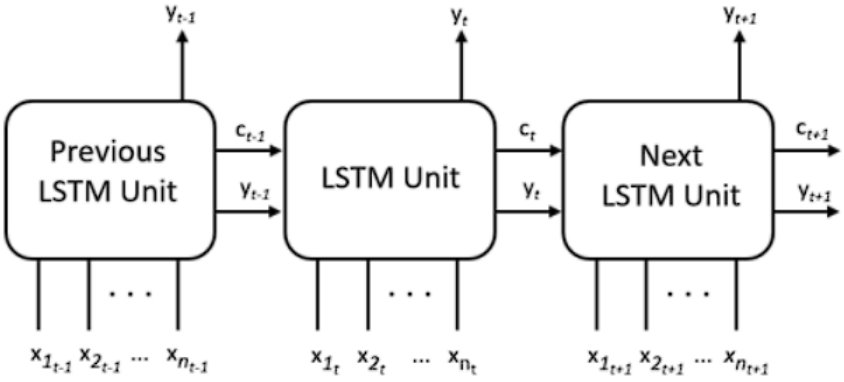


Figura 2.10: Arquitetura LSTM *Multivariate* (GUILLÉN-NAVARRO et al., 2020)
.

Para utilizar múltiplas variáveis na saída, recomenda-se a utilização de outras abordagens em conjunto para garantir o melhor aprendizado e atualização dos pesos sinápticos. De acordo com Zhang e Yang (2017), o Multitask Learning (MTL) é indicado para realizar o aprendizado de mais de uma variável e ainda apresentar uma

saída separada para cada variável. A técnica combina camadas de aprendizado em conjunto para aproveitar as correlações entre as variáveis e ainda utilizar camadas separadas para garantir o melhor desempenho para cada uma.

### 2.2.3 *Multi-step* x *Single-step*

O horizonte de previsão realizada pela LSTM em séries temporais pode ser divida em dois tipos:

1. ***Single-step*:** Apenas a previsão de um instante $t$ é realizada. Essa previsão segue o padrão de intervalo e é sucessor ao último instante disponível na entrada (CHEVILLON, 2007).

   A Figura 2.11 ilustra como é o funcionamento das previsões *single-step*. É possível observar as $n$ entradas sendo inseridas na camada LSTM e a previsão do instante $n + 1$, que é o sucessor do último instante da entrada e é único.

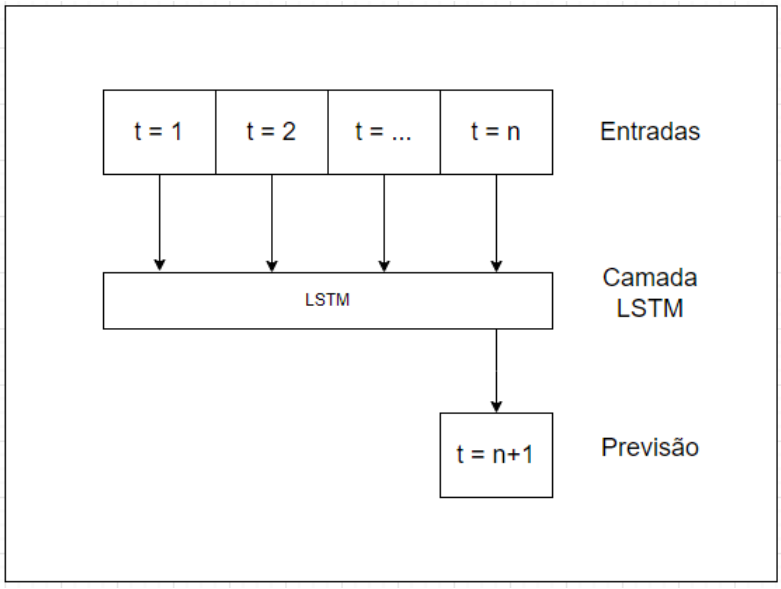


Figura 2.11: Arquitetura LSTM *Single-step*. Elaborado pelo autor.

2. ***Multi-step*:** Podem ser geradas previsões sucessivas de mais de um instante. O instante do primeiro dado previsto é sucessor do último instante da entrada, e o horizonte ou número de instantes previstos é ilimitado (CHEVILLON, 2007).

   O multi-step também pode ser separado em dois tipos:

(a) **Direto**: A primeira é a forma direta, que é representada na Figura 2.12, que consiste em não adicionar valores que foram previstos nas entradas e a previsão dos $k$ instantes são feitas utilizando um mesmo conjunto de entrada. A Figura 2.12 mostra o conceito de uma LSTM com previsão Multi-step. É possível observar as $n$ entradas inseridas na camada LSTM e as $k$ saídas, sendo a primeira $n+1$, sucessiva ao último instante da entrada $n$.

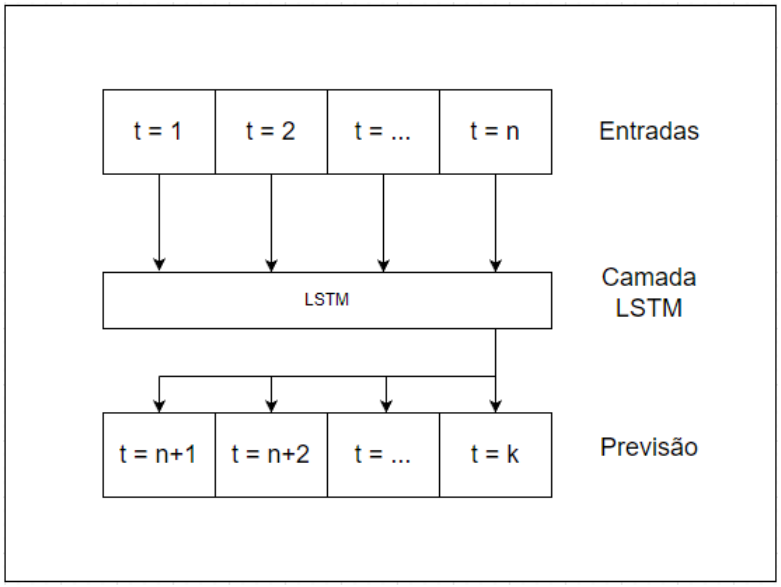


Figura 2.12: Arquitetura LSTM *Multi-step*. Elaborado pelo autor.

(b) **Auto-regressivo**: A outra versão é a auto regressiva, que na verdade realiza vários processos *Single-step* de previsão de um instante $t$ futuro e atualiza esse valor nos dados de entrada, retirando o mais antigo e refazendo o processo de previsão até que os $k$ instantes possam ser previstos (CHEVILLON, 2007).

Para instantes futuros não tão distantes a forma direta da abordagem é satisfatória e pode possuir resultados melhores que a abordagem auto regressiva, porém para horizontes mais afastados a auto regressiva pode ser superior. Importante ressaltar que independente da abordagem quanto mais distante dos instantes de entrada reais, maior será o erro (CHEVILLON, 2007).

## 2.3 Trabalhos Relacionados

Devido a importância do tema na organização dos estados, temos uma sequência de trabalhos realizados utilizando o conceito de redes neurais aplicados em séries temporais na tentativa de atingir previsões melhores.

Silva e Figueiredo (2020) realizou o trabalho de previsão do ICMS do estado do Rio de Janeiro, com a utilização de variáveis endógenas e o conceito de Machine Learning chamado de Long Short-Term Memory(LSTM), comparando com abordagens de Rede Neural Multi-Layer Perceptron (MLP). O trabalho descreve a utilização de previsão de múltiplos passos auto-regressivo e janelamento deslizante de 12 meses, além da comparação entre tratamento de dados utilizando apenas normalização e diferenciação e a realização de manipulações matemáticas. Como conclusão, a utilização de rede LSTM foi superior a MLP, com a utilização de apenas normalização e diferenciação no pré-processamento dos conjuntos.

Castanho (2011) utilizou os dados de ICMS do estado do Espírito Santo de 2000 até 2009 na tentativa de prever a arrecadação do ano 2010. Foi utilizado o modelo Holt-Winters unido a metodologia de Box-Jenkins e modelo econométrico misto com regressão múltipla e apenas uma equação comportamental. Foram analisados as diferentes varáveis e setores que têm impacto no tributo e influenciam na arrecadação. Foi utilizado o erro percentual absoluto médio (MAPE) para decidir qual modelo apresentou o melhor desempenho, chegando ao resultado de que para previsões de curto prazo os modelos de Holt-Winters foram superiores, para médio prazo os melhores modelos foram de Box-Jenkins, enquanto os modelos econométricos são melhores adaptados a longo prazo.

Pessoa, Coronel e Lima (2011) escolheram a arrecadação do período de 1998 até 2011 no estado de Minas Gerais e foi realizada a comparação entre os modelos ARIMA e ARFIMA para a previsão do período. O modelo ARIMA (1, 0, 1) apresentou resultados superiores considerando as métricas de Raiz Quadrada do Erro Quadrado Médio de Previsão (RQEMP), Erro Absoluto Médio de Previsão (EAMP) e Coeficiente de Desigualdade de Theiler (CDT). Já o modelo ARFIMA (1, 0.36, 1) foi superior em relação ao Erro Absoluto Médio Percentual de Previsão (EAMPP). Como conclusão, os dois modelos apresentaram capacidade de realizar a previsão e atingir bons resultados para a tomada

de decisão dos orgãos públicos.

Scheffer, Souza e Zanini (2014) basearam seu trabalho na previsão do ICMS do estado do Rio Grande do Sul do período de 1998 até 2014. Utilizaram como base para previsão os modelos Box-Jenkins com a implementação de modelos ARIMA, chegando no melhor resultando sendo um modelo ajustado SARIMA, em que a base da análise foi de resíduo e testes de significância dos parâmetros, sendo comparado com diversos modelos concorrentes.

Dessa forma, vemos que existe a necessidade por trabalhos que tem como objetivo a previsão do tributo, porém temos muitas técnicas novas de redes neurais e machine learning que ainda podem ser aplicadas e comparadas, visando atingir resultados melhores que possam auxiliar os governantes e sociedade.

## 2.4 ICMS-RJ

O Imposto sobre Circulação de Mercadorias e Serviços foi criado no ano de 1996 pela Lei Complementar 87/1996 (PLANALTO, 1996), que tem o nome de "Lei Kandir". É um tributo de competência estadual, em que o estado tem plenos poderes na instituição e cobrança do mesmo.

De acordo com o IBGE (2021), o estado do Rio de Janeiro é uma das 27 unidades federativas do Brasil, com uma população estimada de 17,6 milhões, sendo o terceiro estado mais populoso da federação .

De acordo com o SEFAZ-RJ (2022), o ICMS no estado carioca é o maior meio de arrecadação de tributos e o valor arrecadado é aplicado em diversas áreas da administração pública. Devido a isso, para o planejamento orçamentário do estado, a previsão do mesmo é um aspecto muito importante para a gestão.

Essa previsão anual é dever da Secretaria de Estado de Fazenda do Rio de Janeiro (SEFAZ-RJ), e é uma tarefa complexa, devido à própria grandeza do imposto e dos diversos fatores, incluindo externos que afetam sua arrecadação.

O gráfico na Figura 2.13 mostra a evolução do valor arrecadado de ICMS no decorrer dos anos. É possível visualizar que esse valor é não linear e possui muitas variações que dificultam a previsão do mesmo.

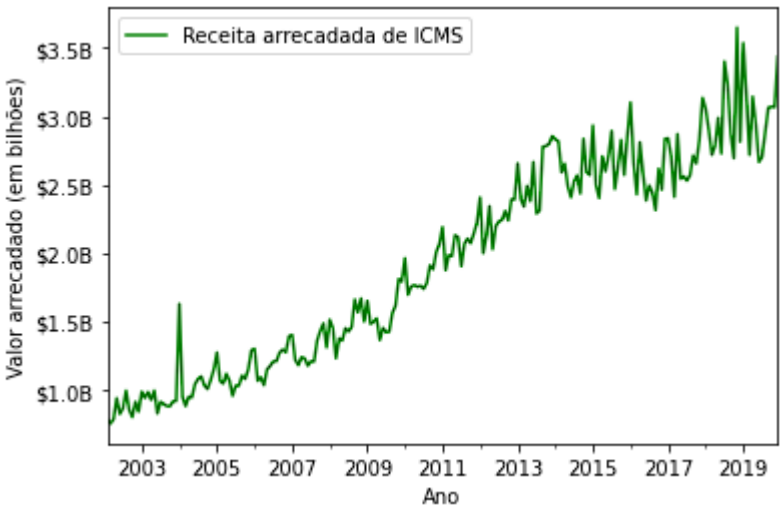


Figura 2.13: Gráfico da arrecadação de ICMS no período de 2002 até 2019. Elaborado pelo autor.

Devido a isso, o Sefaz-RJ utiliza de modelos estatísticos para realizar a previsão. Os modelos econométricos utilizados atualmente são o Autorregressivo Integrado de Médias Móveis Sazonais (SARIMA), que é um extensão do modelo Autoregressivo Integrado de Médias Móveis (ARIMA), onde o SARIMA explicita a parte sazonal, e o modelo Auto Regressão Vetorial (VAR), que considera as correlações lineares entre as séries temporais, sendo possível a presença de mais de uma variável no processo (SILVA; FIGUEIREDO, 2020).

Apesar de serem modelos conceituados em previsões, eles podem apresentar erros que podem ser minimizados utilizando outras técnicas e, consequentemente, gerar um melhor informativo quanto à arrecadação do imposto.

# 3 Metodologia

Neste capítulo é descrita a metodologia utilizada nos modelos de Redes Neurais com o objetivo de fazer a previsão da arrecadação do ICMS no estado do Rio de Janeiro, com o uso de variáveis endógenas e exógenas.

A arquitetura utilizada foi a rede recorrente do tipo LSTM, devido ao bom desempenho e resultados obtidos na previsão de séries temporais de impostos com variáveis endógenas (SILVA; FIGUEIREDO, 2020).

O modelo que utiliza apenas um tipo de variável de interesse tanto para entrada quanto saída é chamado de Univariate. A rede LSTM também suporta a adição de mais variáveis de interesse, seja para a saída ou entrada, sendo esse modelo chamado de Multivariate.

Esse trabalho visa comparar as duas abordagens e analisar qual apresenta um desempenho superior. Ambas têm o objetivo de realizar a previsão do ICMS ao longo de um período estipulado portanto a saída das redes será a mesma, apenas referente a previsão do ICMS anual.

## 3.1 Coleta de dados

Os dados de todas as variáveis utilizadas, para treinamento, validação e teste, foram retirados do período entre 2002 e 2019, sendo adquiridas nos seguintes locais:

ICMS: Foi utilizado nas redes Univariate e Multivariate. Retirado do site SEFAZ-RJ[1].

PIB (Produto Interno Bruto): Foi utilizado na rede Multivariate. Retirado do site IpeaData [2].

Consumo de Energia Elétrica Comercial na Região Sudeste: Foi utilizado na rede Multivariate. Retirado do site Empresa de Pesquisa Energética [3].

[1] http://www.fazenda.rj.gov.br/sefaz/
[2] http://www.ipeadata.gov.br/
[3] http://www.epe.gov.br/

Consumo de Derivados de Petróleo: Foi utilizado na análise das variáveis. Retirado do site da Agência Nacional do Petróleo, Gás Natural e Biocombustíveis [4].

Índice Volume de Vendas no Varejo: Foi utilizado na análise das variáveis. Retirado do site IpeaData[5].

Dívida Líquida do Setor Público: Foi utilizado na análise das variáveis. Retirado do site Portal da Transparência do Governo do Rio de Janeiro [6].

## 3.2 Pré-processamento dos Dados

Esta seção descreve todos os processos iniciais de tratamento dos dados, de forma que garanta a melhor qualidade e aprendizado das redes neurais utilizadas.

### 3.2.1 Conjunto de Dados

Em primeiro lugar a divisão da base de dados foi realizada no modelo *holdout* [7], sendo separados em conjuntos de treino, validação e teste. Cada um apresenta um tamanho diferente, de forma que possam ser adequados para cada uma das etapas.

O conjunto de treino, responsável por ajustar os pesos na etapa de treinamento, é composto pelos dados do período de janeiro de 2002 até dezembro de 2016.

O conjunto de validação tem como objetivo fazer o *Early Stopping* e evitar o *overfitting*, portanto é utilizado o período de janeiro de 2017 até dezembro de 2018.

Já o conjunto de teste, que irá avaliar o desempenho dos modelos e será a base da comparação, é composto pelo período de janeiro de 2019 a dezembro de 2019.

### 3.2.2 Manipulação dos Dados de Entrada

Como manipulação dos dados de entrada, para as redes neurais Univariate e Multivariate foram aplicadas a normalização dos dados, que consiste em transformar os dados em valores normalizados, em torno de um intervalo fixo. Esse intervalo foi o [0,1] e foi utilizado

[4]http://www.gov.br/anp/
[5]http://www.ipeadata.gov.br/
[6]http://www.fazenda.rj.gov.br/transparencia/
[7]separação do conjunto de dados em dados de treinamento, validação e teste.

a função MinMaxScaler, do pacote preprocessing, da biblioteca *Scikit-learn* [8]. A Equação 3.1 mostra o cálculo para alteração dos valores.

$$y = \frac{(x - x_{min})(l_{max} - l_{min})}{x_{max} - x_{min}} + l_{min} \tag{3.1}$$

onde:

y = valor normalizado;

x = valor a ser normalizado;

$x_{max}$ = valor máximo da série;

$x_{min}$ = valor mínimo da série;

$l_{max}$ = limite máximo do intervalo de normalização;

$l_{min}$ = limite mínimo do intervalo de normalização;

A normalização é importante para evitar tendências nos ajustes de pesos, gerando melhor capacidade de generalização, que é o objetivo da rede neural.

## 3.3 Escolha das variáveis exógenas

Como explicado na seção 2.4, a série temporal do ICMS é afetada por dados externos e a utilização de outras variáveis na predição das séries futuras tem sua importância.

Dessa forma, foram avaliados diversos dados de produtos e serviços que possuíam a cobrança de ICMS e selecionados os que apresentavam uma maior correlação.

Para realizar esse processo foi utilizado a função de correlação da biblioteca Pandas[9], em que utilizando o método de correlação de Pearson, onde o coeficiente de Pearson gerado possui um valor de -1 a 1. Quanto mais próximo de -1 menor é a relação entre as variáveis e quanto mais próximo de 1 maior é a relação das mesmas (PEARSON, 1896). Sendo assim é possível quantificar a semelhança entre as variáveis externas com a variável de interesse que é a arrecadação do ICMS.

Gerando um mapa de calor com o auxílio da biblioteca *Seaborn* [10], é possível

[8] https://scikit-learn.org/
[9] https://pandas.pydata.org/
[10] https://seaborn.pydata.org/

visualizar a correlação na Figura 3.1. Como é possível observar no mapa, ao analisar a primeira coluna, temos a correlação entre a variável de interesse, que é o valor mensal arrecadado de ICMS, com as variáveis exógenas selecionadas, em que suas numerações estão indicadas ao final do nome no eixo x. Pode-se considerar também a presença da legenda à direita, que indica qual o valor do coeficiente de correlação de Pearson, através da cor indicada.

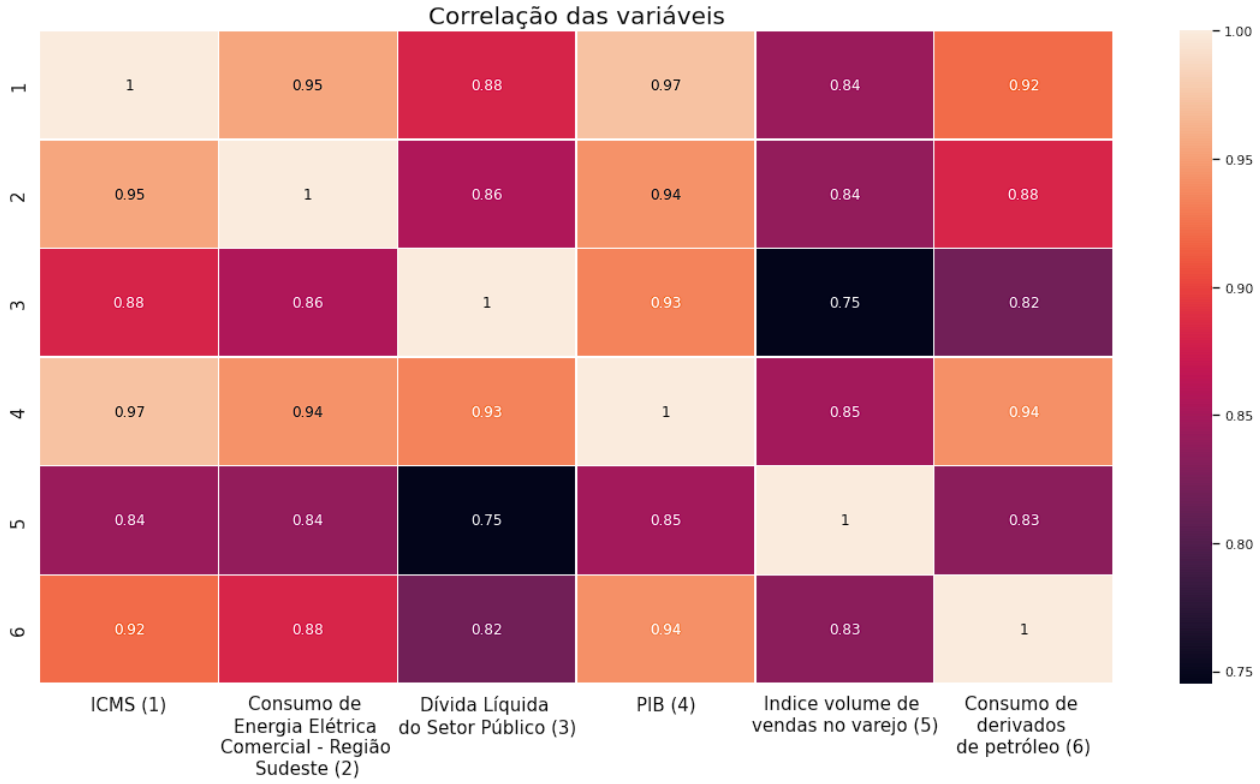


Figura 3.1: Mapa de calor relativo a correlação das variáveis. Para efeito visual os indicativos do eixo y estão marcados no eixo x. Elaborado pelo autor.

Sendo assim, é possível concluir que as variáveis de PIB e Consumo de Energia Elétrica Comercial da Região Sudeste possuem alto grau de correlação com o valor mensal arrecadado de ICMS, chegando a níveis de coeficientes maiores que 0.95. Já as variáveis de Consumo de derivados de petróleo e Dívida Líquida do Setor Público possuem coeficiente entre 0.85 e 0.95. Por último temos o coeficiente do Índice do volume de vendas no varejo que fica em torno de 0.80

Após essa verificação, as variáveis de PIB e Consumo de Energia Elétrica Comercial da Região Sudeste foram selecionadas para serem entradas da rede neural Multivari-

ate, por questão de simplificação nas análises e por possuírem um alto grau de correlação.

Os gráficos da série temporal do ICMS, PIB e Consumo de Energia Elétrica Comercial da Região Sudeste, estão respectivamente nas Figuras 3.2, 3.3 e 3.4

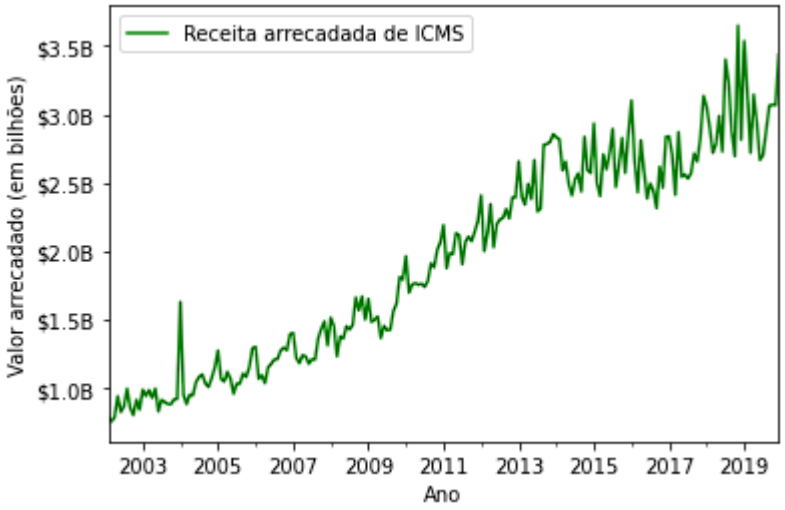


Figura 3.2: Gráfico da arrecadação de ICMS em bilhões (R$). Elaborado pelo autor.

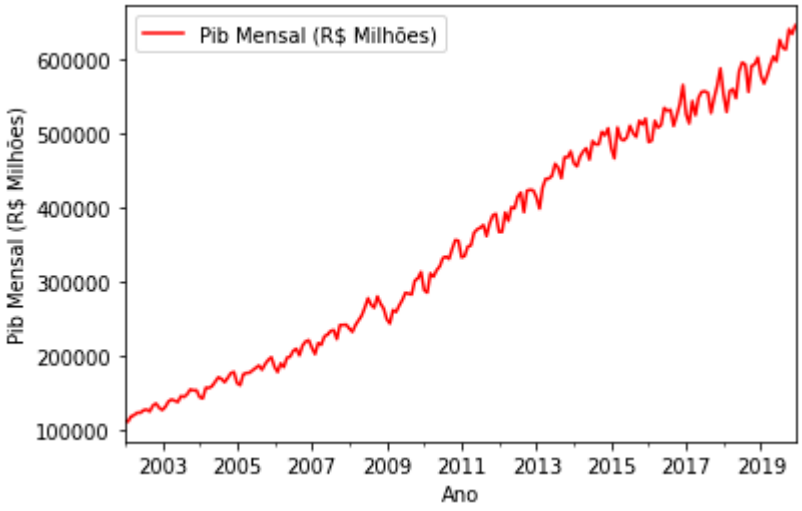


Figura 3.3: Gráfico do Pib Mensal em milhões (R$). Elaborado pelo autor.

Dessa forma, podemos observar nas figuras que o crescimento das três segue um padrão semelhante e alguns altos e baixos tem impacto direto na arrecadação do ICMS, o que indica a alta correlação entre as variáveis.

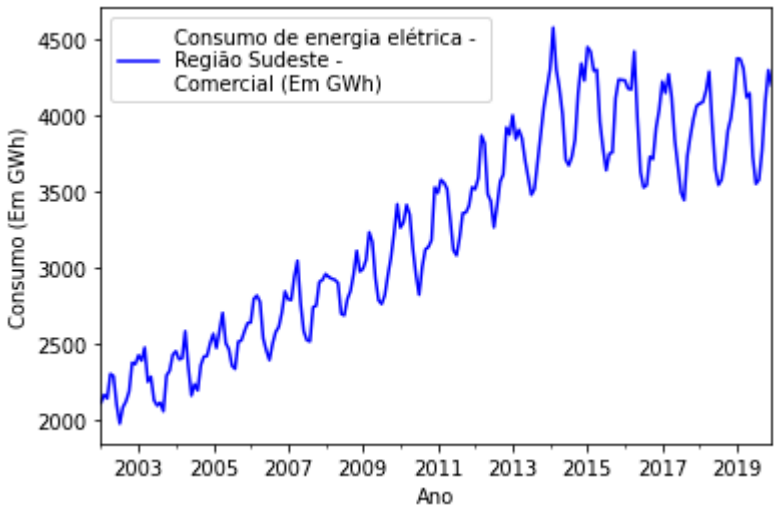


Figura 3.4: Gráfico do Consumo de Energia Elétrica Comercial da Região Sudeste em GWh. Elaborado pelo autor.

## 3.4 Janela deslizante

Para o problema, e como forma de adaptar para a arquitetura *Multi-step*, o processo de janela deslizante descrito na seção 2.1.6, será utilizado um janelamento de intervalo igual a 24, onde para os 12 meses da previsão, o conjunto de observações passadas será composto pelos 24 meses anteriores.

- Para o modelo *Univariate*, o janelamento para a previsão será dado pela equação 3.2:

$$t_{n+11}, t_{n+10}, ..., t_n = [t_{n-24}, t_{n-23}, ..., t_{n-12}, t_{n-11}, ..., t_{n-2}, t_{n-1}] \tag{3.2}$$

  onde todos os instantes $t$ são valores mensais da variável ICMS.

- Para o modelo *Multivariate* , o janelamento para a previsão será dado pela equação 3.3:

$$\begin{aligned} t_{ICMS_{n+11}}, t_{ICMS_{n+10}}, ..., t_{ICMS_n} = [[t_{ICMS_{n-24}}, t_{PIB_{n-24}}, t_{CE_{n-24}}], \\ [t_{ICMS_{n-23}}, t_{PIB_{n-23}}, t_{CE_{n-23}}], ..., [t_{ICMS_{n-12}}, t_{PIB_{n-12}}, t_{CE_{n-12}}], \\ [t_{ICMS_{n-11}}, t_{PIB_{n-11}}, t_{CE_{n-11}}], ..., [t_{ICMS_{n-1}}, t_{PIB_{n-1}}, t_{CE_{n-1}}]] \end{aligned} \tag{3.3}$$

  onde para cada mês temos um conjunto de três varíaveis em cada instante, sendo

elas ICMS, PIB e CE (Consumo de Energia Elétrica Comercial da Região Sudeste).

## 3.5 Estrutura da Rede LSTM

Os modelos implementados foram feitos a partir da biblioteca *Keras* [11] e *TensorFlow* [12] e implementados na linguagem de programação Python [13].

Para a realização da rede neural LSTM Univariate, foi utilizado como referência o melhor resultado obtido no trabalho de Silva e Figueiredo (2020), sendo modificada a estrutura para se adequar ao objetivo de previsão anual. A estrutura da camada de entrada receberá 24 registros, que correspondem aos 24 meses anteriores ao ano de previsão e 12 registros de saída, que será o valor preditivo dos meses do ano alvo.

A Figura 2.9 da seção 2.2.2 descreve o funcionamento da LSTM Univariate e pode ser complementada com a Figura 2.12, localizado na subseção 2.2.3, que possui número de entradas igual a 24 e um número de saídas de previsão igual a 12.

Para a rede neural LSTM Multivariate foram utilizadas três variáveis, o valor endógeno do ICMS, o PIB e Consumo de Energia Elétrica Comercial da Região Sudeste como variáveis exógenas, tem-se portanto um total de 72 entradas, ou seja, três conjuntos de 24 dados passados, um para cada variável, além de 12 saídas, referentes aos valores endógenos previstos de arrecadação do imposto em cada mês durante um ano.

O funcionamento pode ser visto na Figura 3.5, onde temos 24 conjuntos de entrada, de $n-1$ até $n-24$, onde cada conjunto é composto por três instantes, um para cada variável, sendo elas ICMS, PIB e CE (Consumo de Energia Elétrica Comercial da Região Sudeste). As saídas apresentam apenas valores mensais do ICMS e totalizam 12 registros, de $n$ até $n-11$.

[11] https://keras.io/
[12] https://www.tensorflow.org/
[13] https://www.python.org/

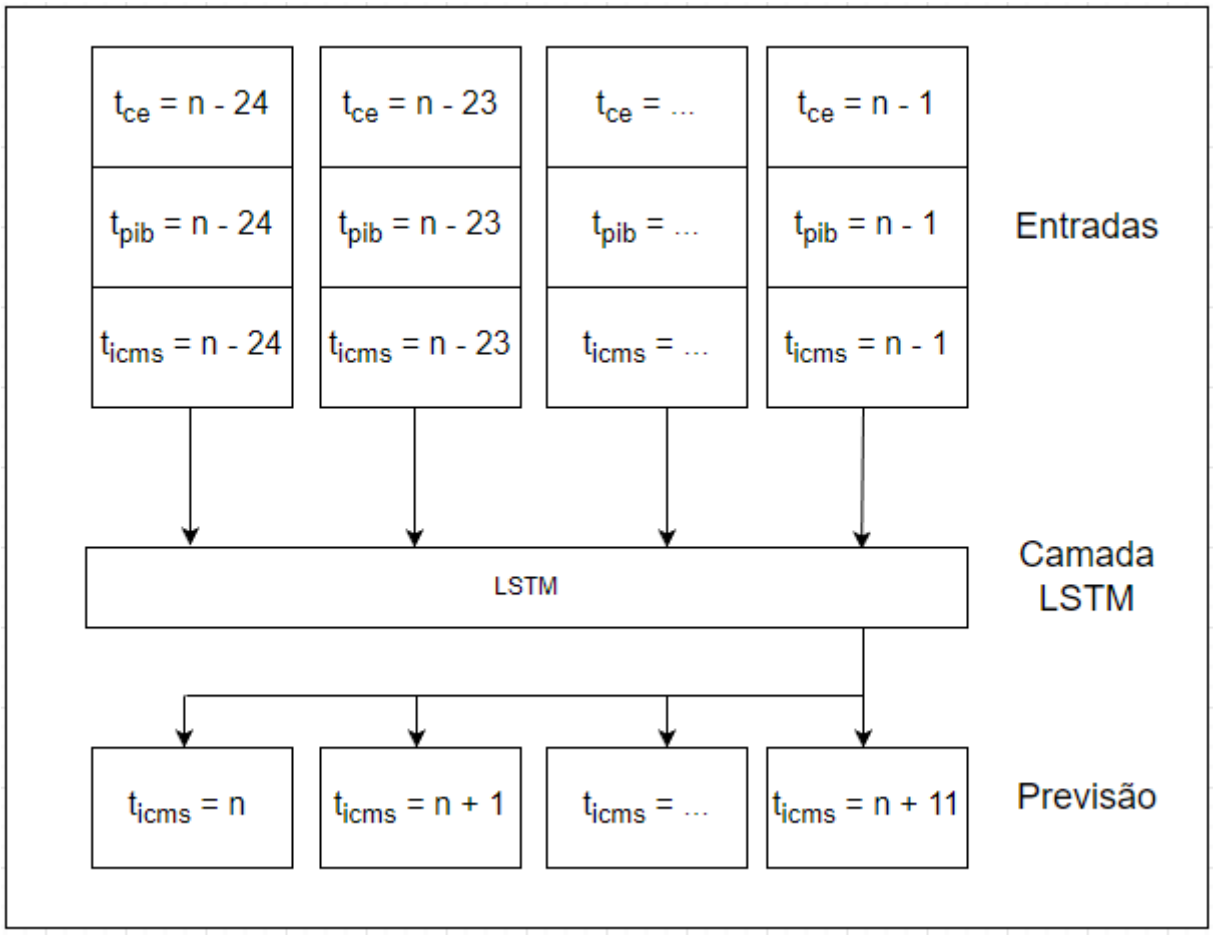


Figura 3.5: Arquitetura da rede neural LSTM *Multivariate Multistep*. Elaborado pelo autor.

## 3.6 Avaliação

Para a avaliação e análise do melhor modelo e parâmetros foi utilizado três métricas de erros que são comuns na comparação de redes neurais.

Os erros são Erro Quadrático Médio (Mean Squared Error - MSE), Raiz do Erro Médio Quadrático (Root Mean Square Error - RMSE) e Média Percentual do Erro Absoluto (Mean Absolute Percentage Error - MAPE), em que as respectivas equações são 3.4, 3.5 e 3.6:

$$MSE = \frac{1}{n} * \sum_{t=1}^{n} (y_t - \hat{y}_t)^2 \tag{3.4}$$

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n} * \sum_{t=1}^{n} (y_t - \hat{y}_t)^2} \tag{3.5}$$

$$MAPE = \frac{1}{n} * \sum_{t=1}^{n} \frac{(y_t - \hat{y}_t)}{y_t} * 100 \tag{3.6}$$

e que:

n = número de observações;

$y_t$ = valor real;

$\hat{y}_t$ = valor previsto;

Para o treinamento, será selecionado o erro MSE para ajustes de pesos. Na etapa de validação, para garantir o *Early Stopping*, descrito na subseção 2.1.4, o valor do erro MSE será considerado, com um acumulado de 100 erros para garantir que não esteja ocorrendo *overfitting* e terminar o treinamento no momento correto. Os erros de RMSE e MAPE são utilizados para análise da eficácia do modelo na validação. Na etapa de teste as três métricas aplicadas a cada modelo mostrará qual possui melhor desempenho e resultado, sendo considerado o melhor modelo.

O treinamento de cada modelo foi repetido 10 vezes, de forma que as métricas finais representam a média dos 10 modelos produzidos, sendo a métrica MAPE escolhida para definir o melhor modelo.

# 4 Estudo de Casos

## 4.1 Arquiteturas dos Modelos de Rede Neural

### 4.1.1 LSTM Univariate

Os parâmetros da LSTM para previsão com variáveis endógenas foram baseados no determinado pelo melhor modelo do trabalho de Silva e Figueiredo (2020), com algumas alterações. Como tamanho de batch foi utilizado o valor 32, o que significa que foram analisados 32 registros para concluir a mudança dos pesos. O número de camadas utilizadas foram sempre duas, com uma camada LSTM e outra uma camada *Dense* (camada oculta) com 12 neurônios, referentes a quantidade dos meses de um ano previsto de saída.

Nos modelos foram experimentados diferentes configurações do número de neurônios na primeira camada LSTM, o algoritmo otimizador de aprendizagem e a função de ativação dos neurônios.

O número de neurônios na primeira camada escolhidos foram os valores de 400, 500, 550, 600 ou 700.

Os algoritmos otimizadores utilizados foram o Adaptive Moment Estimation (ADAM), com a taxa de aprendizado de 0.001, e o Stochastic Gradient Descent (SGD) com a taxa de aprendizado de 0.01. Ambos foram implementados através do framework Keras[14].

As funções de ativação utilizadas foram a Sigmóide e a Relu.

### 4.1.2 LSTM Multivariate

Os parâmetros tamanho de batch e números de camadas são os mesmos do modelo *Univariate*. Os algoritmos otimizadores, funções de ativação e quantidades de neurônios nas camadas também é semelhante, sendo a diferença relacionada ao número de variáveis utilizadas.

[14]https://keras.io/

O número de neurônios na primeira camada foram os valores de 400, 500, 550, 600 ou 700.

Os algoritmos otimizadores utilizados foram o Adaptive Moment Estimation (ADAM), com a taxa de aprendizado de 0.001, e o Stochastic Gradient Descent (SGD) com a taxa de aprendizado de 0.01. Ambos foram implementados através do framework Keras.

As funções de ativação foram a Sigmóide e a Relu.

As variáveis exógenas são PIB e Consumo de Energia Elétrica Comercial na Região Sudeste, que são usadas na entrada junto com o ICMS.

## 4.2 Cenários de Testes

A Tabela 4.1 mostra os cenários de testes para a LSTM Univariate (com apenas variáveis endógenas):

Tabela 4.1: Cenários de teste para LSTM Univariate. Elaborado pelo autor.

| Cenário | Otimizador | Função de Ativação | Número de Neurônios |
|---|---|---|---|
| 1 | Adam | Relu | 400, 500, 550, 600, 700 |
| 2 | | Sigmóide | |
| 3 | Sgd | Relu | |
| 4 | | Sigmóide | |

Os cenários de testes para os modelos de LSTM Multivariate estão organizados de acordo com a Tabela 4.2, onde as arquiteturas *Multivariate* diferem-se em relação ao número de variáveis, onde é possível comparar o quanto a adição de mais variáveis pode ser benéfica para o modelo.

Tabela 4.2: Cenários de teste para LSTM Multivariate. Elaborado pelo autor.

| Cenário | Otimizador | Função de Ativação | Número de Neurônios | Variáveis |
|---|---|---|---|---|
| 5 | Adam | Relu | 400, 500, 550, 600, 700 | ICMS e PIB |
| 6 | | Sigmóide | | |
| 7 | Sgd | Relu | | |
| 8 | | Sigmóide | | |
| 9 | Adam | Relu | 400, 500, 550, 600, 700 | ICMS, PIB e Consumo de Energia Elétrica Comercial - Região Sudeste |
| 10 | | Sigmóide | | |
| 11 | Sgd | Relu | | |
| 12 | | Sigmóide | | |

## 4.3 Resultados dos Experimentos

Para cada cenário disponível foram executados 10 treinamentos de cada modelo, de forma que os valores das métricas de erro analisadas são as médias das execuções.

Para a avaliação das comparações entre os modelos LSTM são sempre consideradas as métricas em relação ao conjunto de validação, que contempla os meses de janeiro de 2017 até dezembro de 2018. Já o conjunto de teste, que contempla os meses de janeiro de 2019 até dezembro de 2019 foi utilizado na comparação final dos melhores modelos e os valores reais de arrecadação do imposto no ano. O melhor modelo selecionado de cada cenário será aquele que atingir menor valor de erro MAPE médio.

### 4.3.1 Cenário 1: Modelo LSTM Univariate, algoritmo de otimização Adam e função de ativação Relu

Cenário da arquitetura LSTM Univariate, com a utilização do algoritmo de otimização Adam e função de ativação Relu.

Na Tabela 4.3 estão dispostos os erros médios RMSE, MSE e MAPE das 10 execuções, sendo todos do conjunto de validação. O modelo apresenta 5 configurações de neurônios na camada LSTM, sendo a outra camada Dense com 12 neurônios.

Tabela 4.3: Erros médios de validação RMSE, MSE e MAPE do cenário de teste 1. Elaborado pelo autor.

| Erro Médio de Validação | | | |
|---|---|---|---|
| Número de Neurônios | RMSE (R$) | MSE (R$) | MAPE (%) |
| 400 | 247.517.209,2 | 61.264.803.791.057.700 | 7.07 |
| 500 | 245.559.707,0 | 60.299.570.879.817.300 | 7.04 |
| **550** | **241.717.815,7** | **58.427.502.439.170.000** | **6.95** |
| 600 | 241.757.162,3 | 58.446.536.875.232.400 | 6.98 |
| 700 | 245.459.015,3 | 60.250.191.640.815.200 | 6.96 |

Ao analisar os resultados, temos que o melhor modelo é aquele com menor erro de MAPE médio. Portanto, em negrito está em destaque o melhor caso, onde o número de neurônios na camada LSTM é 550 e o erro de MAPE foi de 6.95%.

### 4.3.2 Cenário 2: Modelo LSTM Univariate, algoritmo de otimização Adam e função de ativação Sigmóide

Cenário da arquitetura LSTM Univariate, com a utilização do algoritmo de otimização Adam e função de ativação Sigmóide.

Na Tabela 4.4 estão dispostos os erros médios RMSE, MSE e MAPE das 10 execuções, sendo todos do conjunto de validação. O modelo apresenta 5 configurações de neurônios na camada LSTM, sendo a outra camada Dense com 12 neurônios.

Tabela 4.4: Erros médios de validação RMSE, MSE e MAPE do cenário de teste 2. Elaborado pelo autor.

| Erro Médio de Validação | | | |
|---|---|---|---|
| Número de Neurônios | RMSE (R$) | MSE (R$) | MAPE (%) |
| 400 | 269005586,5 | 72.364.005.594.759.100 | 7.29 |
| 500 | 276922889,5 | 76.686.345.576.330.000 | 7.36 |
| 550 | 275929737,9 | 76.137.220.263.641.000 | 7.31 |
| **6**00 | **274878072,9** | **75.558.095.463.383.000** | **7.18** |
| 700 | 283754538,7 | 81.243.601.371.136.000 | 7.64 |

Ao analisar os resultados, temos que o melhor modelo é aquele com menor erro de MAPE médio. Portanto, em negrito está em destaque o melhor caso, onde o número de neurônios na camada LSTM é 600 e o erro de MAPE foi de 7.18%.

### 4.3.3 Cenário 3: Modelo LSTM Univariate, algoritmo de otimização Sgd e função de ativação Relu

Cenário da arquitetura LSTM Univariate, com a utilização do algoritmo de otimização Adam e função de ativação Sigmóide.

Na Tabela 4.5 estão dispostos os erros médios RMSE, MSE e MAPE das 10 execuções, sendo todos do conjunto de validação. O modelo apresenta 5 configurações de neurônios na camada LSTM, sendo a outra camada Dense com 12 neurônios.

Tabela 4.5: Erros médios de validação RMSE, MSE e MAPE do cenário de teste 3. Elaborado pelo autor.

| Erro Médio de Validação | | | |
|---|---|---|---|
| Número de Neurônios | RMSE (R$) | MSE (R$) | MAPE (%) |
| 400 | 262704538,4 | 69.013.718.932.494.700 | 6.94 |
| 500 | 259761386,3 | 67.476.273.970.859.200 | 7.06 |
| **550** | **263604312,0** | **69.487.257.974.734.800** | **6.88** |
| 600 | 259382658,5 | 67.279.550.295.310.300 | 7.04 |
| 700 | 260096435,7 | 67.650.910.347.591.600 | 6.99 |

Ao analisar os resultados, temos que o melhor modelo é aquele com menor erro de MAPE médio. Portanto, em negrito está em destaque o melhor caso, onde o número de neurônios na camada LSTM é 550 e o erro de MAPE foi de 6.88%.

### 4.3.4 Cenário 4: Modelo LSTM Univariate, algoritmo de otimização Sgd e função de ativação Sigmóide

Cenário da arquitetura LSTM Univariate, com a utilização do algoritmo de otimização Sgd e função de ativação Sigmóide.

Na Tabela 4.6 estão dispostos os erros médios RMSE, MSE e MAPE das 10 execuções, sendo todos do conjunto de validação. O modelo apresenta 5 configurações de neurônios na camada LSTM, sendo a outra camada Dense com 12 neurônios.

Tabela 4.6: Erros médios de validação RMSE, MSE e MAPE do cenário de teste 4. Elaborado pelo auto.r

| Erro Médio de Validação | | | |
|---|---|---|---|
| Número de Neurônios | RMSE (R$) | MSE (R$) | MAPE (%) |
| 400 | 522926061,7 | 341.487.417.430.324.000 | 12.34 |
| **500** | **475224467,8** | **261.644.361.085.747.000** | **10.84** |
| 550 | 608456554,2 | 402.925.604.326.657.000 | 15.46 |
| 600 | 486779215,3 | 270.574.862.532.503.000 | 10.88 |
| 700 | 604231619,7 | 382.186.057.657.797.000 | 15.35 |

Ao analisar os resultados, temos que o melhor modelo é aquele com menor erro de MAPE médio. Portanto, em negrito está em destaque o melhor caso, onde o número de neurônios na camada LSTM é 500 e o erro de MAPE foi de 10.84%.

### 4.3.5 Cenário 5: Modelo LSTM Multivariate, algoritmo de otimização Adam, função de ativação Relu e 2 variáveis

Cenário da arquitetura LSTM Multivariate, com a utilização do algoritmo de otimização Adam, função de ativação Relu e 2 variáveis, sendo elas ICMS e PIB.

Na Tabela 4.7 estão dispostos os erros médios RMSE, MSE e MAPE das 10 execuções, sendo todos do conjunto de validação. O modelo apresenta 5 configurações de neurônios na camada LSTM, sendo a outra camada Dense com 12 neurônios.

Tabela 4.7: Erros médios de validação RMSE, MSE e MAPE do cenário de teste 5. Elaborado pelo autor.

| Erro Médio de Validação | | | |
|---|---|---|---|
| Número de Neurônios | RMSE (R$) | MSE (R$) | MAPE (%) |
| 400 | 243125330,9 | 59.109.926.524.605.800 | 6.84 |
| 500 | 245348639,7 | 60.195.954.999.929.500 | 6.90 |
| 550 | 243995230,3 | 59.533.672.428.175.400 | 6.96 |
| **600** | **245596041,5** | **60.317.478.178.919.800** | **6.51** |
| 700 | 244010097,8 | 59.540.958.524.997.400 | 6.92 |

Ao analisar os resultados, temos que o melhor modelo é aquele com menor erro de MAPE médio. Portanto, em negrito está em destaque o melhor caso, onde o número de neurônios na camada LSTM é 600 e o erro de MAPE foi de 6.51%.

### 4.3.6 Cenário 6: Modelo LSTM Multivariate, algoritmo de otimização Adam, função de ativação Sigmóide e 2 variáveis

Cenário da arquitetura LSTM Multivariate, com a utilização do algoritmo de otimização Adam, função de ativação Sigmóide e 2 variáveis, sendo elas ICMS e PIB.

Na Tabela 4.8 estão dispostos os erros médios RMSE, MSE e MAPE das 10 execuções, sendo todos do conjunto de validação. O modelo apresenta 5 configurações de neurônios na camada LSTM, sendo a outra camada Dense com 12 neurônios.

Tabela 4.8: Erros médios de validação RMSE, MSE e MAPE do cenário de teste 6. Elaborado pelo autor.

| Erro Médio de Validação | | | |
|---|---|---|---|
| Número de Neurônios | RMSE (R$) | MSE (R$) | MAPE (%) |
| **400** | **242963963,5** | **59.055.436.342.158.400** | **6.17** |
| 500 | 248262488,1 | 61.634.263.013.125.000 | 6.53 |
| 550 | 249618015,9 | 62.309.153.860.837.400 | 6.61 |
| 600 | 248195554,0 | 61.601.033.049.284.600 | 6.49 |
| 700 | 250796253,6 | 62.898.760.841.095.800 | 6.58 |

Ao analisar os resultados, temos que o melhor modelo é aquele com menor erro de MAPE médio. Portanto, em negrito está em destaque o melhor caso, onde o número de neurônios na camada LSTM é 400 e o erro de MAPE foi de 6.17%.

### 4.3.7 Cenário 7: Modelo LSTM Multivariate, algoritmo de otimização Sgd, função de ativação Relu e 2 variáveis

Cenário da arquitetura LSTM Multivariate, com a utilização do algoritmo de otimização Sgd, função de ativação Relu e 2 variáveis, sendo elas ICMS e PIB.

Na Tabela 4.9 estão dispostos os erros médios RMSE, MSE e MAPE das 10 execuções, sendo todos do conjunto de validação. O modelo apresenta 5 configurações de neurônios na camada LSTM, sendo a outra camada Dense com 12 neurônios.

Tabela 4.9: Erros médios de validação RMSE, MSE e MAPE do cenário de teste 7. Elaborado pelo autor.

| Erro Médio de Validação | | | |
|---|---|---|---|
| Número de Neurônios | RMSE (R$) | MSE (R$) | MAPE (%) |
| 400 | 248855484,2 | 61.929.052.110.102.500 | 6.90 |
| 500 | 250146746,4 | 62.573.394.752.678.800 | 6.90 |
| **550** | **249637477,0** | **62.318.870.072.976.500** | **6.88** |
| 600 | 247953161,1 | 61.480.770.567.418.100 | 6.94 |
| 700 | 248210414,6 | 61.608.409.940.352.200 | 6.89 |

Ao analisar os resultados, temos que o melhor modelo é aquele com menor erro de MAPE médio. Portanto, em negrito está em destaque o melhor caso, onde o número de neurônios na camada LSTM é 550 e o erro de MAPE foi de 6.88%.

### 4.3.8 Cenário 8: Modelo LSTM Multivariate, algoritmo de otimização Sgd, função de ativação Sigmóide e 2 variáveis

Cenário da arquitetura LSTM Multivariate, com a utilização do algoritmo de otimização Sgd, função de ativação Sigmóide e 2 variáveis, sendo elas ICMS e PIB.

Na Tabela 4.10 estão dispostos os erros médios RMSE, MSE e MAPE das 10 execuções, sendo todos do conjunto de validação. O modelo apresenta 5 configurações de neurônios na camada LSTM, sendo a outra camada Dense com 12 neurônios.

Tabela 4.10: Erros médios de validação RMSE, MSE e MAPE do cenário de teste 8. Elaborado pelo autor.

| Erro Médio de Validação | | | |
|---|---|---|---|
| Número de Neurônios | RMSE (R$) | MSE (R$) | MAPE (%) |
| 400 | 255231288,4 | 65.143.010.596.678.300 | 6.42 |
| 500 | 253629773,1 | 64.328.061.790.329.300 | 6.43 |
| 550 | 258464329,1 | 66.803.809.414.796.600 | 6.42 |
| 600 | 250657167,6 | 62.829.015.667.973.900 | 6.36 |
| **700** | **251766933,7** | **63.386.588.900.985.500** | **6.34** |

Ao analisar os resultados, temos que o melhor modelo é aquele com menor erro de MAPE médio. Portanto, em negrito está em destaque o melhor caso, onde o número de neurônios na camada LSTM é 700 e o erro de MAPE foi de 6.34%.

### 4.3.9 Cenário 9: Modelo LSTM Multivariate, algoritmo de otimização Adam, função de ativação Relu e 3 variáveis

Cenário da arquitetura LSTM Multivariate, com a utilização do algoritmo de otimização Adam, função de ativação Relu e 3 variáveis, sendo elas ICMS, PIB e Consumo de Energia Elétrica Comercial da Região Sudeste.

Na Tabela 4.11 estão dispostos os erros médios RMSE, MSE e MAPE das 10 execuções, sendo todos do conjunto de validação. O modelo apresenta 5 configurações de neurônios na camada LSTM, sendo a outra camada Dense com 12 neurônios.

Tabela 4.11: Erros médios de validação RMSE, MSE e MAPE do cenário de teste 9. Elaborado pelo autor.

| Erro Médio de Validação | | | |
|---|---|---|---|
| Número de Neurônios | RMSE (R$) | MSE (R$) | MAPE (%) |
| 400 | 248703527,5 | 61.853.444.569.213.600 | 6.97 |
| **500** | **247022589,7** | **61.020.205.447.003.100** | **6.17** |
| 550 | 257179458,0 | 66.141.333.716.747.300 | 6.59 |
| 600 | 248353433,7 | 61.679.428.018.345.300 | 6.35 |
| 700 | 252994047,4 | 64.007.062.880.153.900 | 6.58 |

Ao analisar os resultados, temos que o melhor modelo é aquele com menor erro de MAPE médio. Portanto, em negrito está em destaque o melhor caso, onde o número de neurônios na camada LSTM é 500 e o erro de MAPE foi de 6.17%.

### 4.3.10 Cenário 10: Modelo LSTM Multivariate, algoritmo de otimização Adam, função de ativação Sigmóide e 3 variáveis

Cenário da arquitetura LSTM Multivariate, com a utilização do algoritmo de otimização Adam, função de ativação Sigmóide e 3 variáveis, sendo elas ICMS, PIB e Consumo de Energia Elétrica Comercial da Região Sudeste.

Na Tabela 4.12 estão dispostos os erros médios RMSE, MSE e MAPE das 10 execuções, sendo todos do conjunto de validação. O modelo apresenta 5 configurações de neurônios na camada LSTM, sendo a outra camada Dense com 12 neurônios.

Tabela 4.12: Erros médios de validação RMSE, MSE e MAPE do cenário de teste 10. Elaborado pelo autor.

| Erro Médio de Validação | | | |
|---|---|---|---|
| Número de Neurônios | RMSE (R$) | MSE (R$) | MAPE (%) |
| 400 | 220307515,7 | 48.535.606.294.635.800 | 5.30 |
| **500** | **218900301,8** | **47.917.358.001.517.100** | **5.21** |
| 550 | 222697988,4 | 49.594.394.877.378.600 | 5.36 |
| 600 | 223821995,2 | 50.096.295.498.421.900 | 5.35 |
| 700 | 221050843,2 | 48.863.481.783.223.600 | 5.30 |

Ao analisar os resultados, temos que o melhor modelo é aquele com menor erro de MAPE médio. Portanto, em negrito está em destaque o melhor caso, onde o número de neurônios na camada LSTM é 500 e o erro de MAPE foi de 5.21%.

### 4.3.11 Cenário 11: Modelo LSTM Multivariate, algoritmo de otimização Sgd, função de ativação Relu e 3 variáveis

Cenário da arquitetura LSTM Multivariate, com a utilização do algoritmo de otimização Sgd, função de ativação Relu e 3 variáveis, sendo elas ICMS, PIB e Consumo de Energia Elétrica Comercial da Região Sudeste.

Na Tabela 4.13 estão dispostos os erros médios RMSE, MSE e MAPE das 10 execuções, sendo todos do conjunto de validação. O modelo apresenta 5 configurações de neurônios na camada LSTM, sendo a outra camada Dense com 12 neurônios.

Tabela 4.13: Erros médios de validação RMSE, MSE e MAPE do cenário de teste 11. Elaborado pelo autor.

| Erro Médio de Validação | | | |
|---|---|---|---|
| Número de Neurônios | RMSE (R$) | MSE (R$) | MAPE (%) |
| 400 | 280556801,5 | 78.719.241.599.699.600 | 6.69 |
| 500 | 264627979,2 | 70.028.259.455.294.400 | 6.67 |
| 550 | 267089640,8 | 71.337.566.371.762.700 | 6.66 |
| **600** | **271123065,1** | **73.514.457.961.830.000** | **6.63** |
| 700 | 254860748,0 | 64.955.818.771.338.300 | 6.83 |

Ao analisar os resultados, temos que o melhor modelo é aquele com menor erro de MAPE médio. Portanto, em negrito está em destaque o melhor caso, onde o número de neurônios na camada LSTM é 600 e o erro de MAPE foi de 6.63%.

### 4.3.12 Cenário 12: Modelo LSTM Multivariate, algoritmo de otimização Sgd, função de ativação Sigmóide e 3 variáveis

Cenário da arquitetura LSTM Multivariate, com a utilização do algoritmo de otimização Sgd, função de ativação Sigmóide e 3 variáveis, sendo elas ICMS, PIB e Consumo de Energia Elétrica Comercial da Região Sudeste.

Na Tabela 4.14 estão dispostos os erros médios RMSE, MSE e MAPE das 10 execuções, sendo todos do conjunto de validação. O modelo apresenta 5 configurações de neurônios na camada LSTM, sendo a outra camada Dense com 12 neurônios.

Ao analisar os resultados, temos que o melhor modelo é aquele com menor erro de MAPE médio. Portanto, em negrito está em destaque o melhor caso, onde o número

Tabela 4.14: Erros médios de validação RMSE, MSE e MAPE do cenário de teste 12. Elaborado pelo autor.

| Erro Médio de Validação | | | |
|---|---|---|---|
| Número de Neurônios | RMSE (R$) | MSE (R$) | MAPE (%) |
| 400 | 270031270,9 | 72.918.179.398.211.700 | 6.41 |
| 500 | 263078333,4 | 69.211.459.287.237.200 | 6.32 |
| 550 | 258721333,5 | 66.946.933.123.238.300 | 6.35 |
| 600 | 266609766,5 | 71.080.940.920.904.600 | 6.51 |
| **700** | **251939538,6** | **63.510.630.497.212.300** | **6.24** |

de neurônios na camada LSTM é 700 e o erro de MAPE foi de 6.24%.

## 4.4 Melhor modelo de LSTM Univariate

Analisando todos os cenários de experimentos da arquitetura de LSTM Univariate, a Tabela 4.15 mostra os melhores resultados de cada cenário.

Tabela 4.15: Comparação dos resultados dos cenários de experimento da arquitetura LSTM Univariate. Elaborado pelo autor.

| Erro Médio de Validação | | | | |
|---|---|---|---|---|
| Cenário | Número de Neurônios | Otimizador | Ativação | MAPE (%) |
| 1 | 550 | Adam | Relu | 6.95 |
| 2 | 600 | Adam | Sigmóide | 7.18 |
| **3** | **550** | **Sgd** | **Relu** | **6.88** |
| 4 | 500 | Sgd | Sigmóide | 10.84 |

Como é possível observar, o menor erro de MAPE foi o do cenário 3, que consiste em um rede LSTM Univariate utilizando algoritmo otimizador Sgd, função de ativação Relu e com 550 neurônios na camada LSTM.

Com erro de MAPE igual a 6.88%, erro RMSE igual a R$ 263.604.312,00 e erro MSE igual a R$ 69.487.257.974.734.800,00, a configuração completa do melhor modelo de rede neural LSTM Univariate é:

- 24 dados passados do ICMS como entrada.
- 2 camadas intermediárias, com uma camada LSTM de 550 neurônios e uma camada Dense com 12 neurônios.

- Algoritmo otimizador SGD.
- Função de ativação Relu.

## 4.5 Melhor modelo de LSTM Multivariate

Analisando todos os cenários de experimentos da arquitetura de LSTM Multivariate, a Tabela 4.16 mostra os melhores resultados de cada cenário.

Tabela 4.16: Comparação dos resultados dos cenários de experimento da arquitetura LSTM Multivariate. Elaborado pelo autor.

| Erro Médio de Validação | | | | | |
|---|---|---|---|---|---|
| Cenário | Número de Variáveis | Número de Neurônios | Otimizador | Ativação | MAPE (%) |
| 5 | 2 | 600 | Adam | Relu | 6.51 |
| **6** | **2** | **400** | **Adam** | **Sigmóide** | **6.17** |
| 7 | 2 | 550 | Sgd | Relu | 6.88 |
| 8 | 2 | 700 | Sgd | Sigmóide | 6.34 |
| 9 | 3 | 500 | Adam | Relu | 6.17 |
| **10** | **3** | **500** | **Adam** | **Sigmóide** | **5.21** |
| 11 | 3 | 600 | Sgd | Relu | 6.63 |
| 12 | 3 | 700 | Sgd | Sigmóide | 6.24 |

No cenário 6 pode-se observar o melhor cenário utilizando apenas 2 variáveis, sendo elas ICMS e PIB. Com erro de MAPE igual a 6.17%, erro RMSE igual a R$ 242.963.963,50 e erro MSE igual a R$ 59.055.436.342.158.400,00, a configuração completa do melhor modelo de rede neural LSTM Multivariate utilizando apenas 2 variáveis é:

- 72 dados passados das variáveis como entrada, sendo 24 registros para cada variável
- 2 camadas intermediárias, com uma camada LSTM de 400 neurônios e uma camada Dense com 12 neurônios.
- Algoritmo otimizador Adam.
- Função de ativação Sigmóide.
- 2 variáveis - ICMS e PIB.

Como é possível observar, o menor erro de MAPE foi o do cenário 10, que consiste em um rede LSTM Multivariate utilizando algoritmo otimizador Adam, função de ativação

Sigmóide, com 500 neurônios na camada LSTM e 3 variáveis, sendo elas ICMS, PIB e Consumo de Energia Elétrica Comercial da Região Sudeste.

Com erro de MAPE igual a 5.21%, erro RMSE igual a R$ 218.900.301,80 e erro MSE igual a R$ 47.917.358.001.517.100,00, a configuração completa do melhor modelo de rede neural LSTM Multivariate é:

- 72 dados passados das variáveis como entrada, sendo 24 registros para cada variável
- 2 camadas intermediárias, com uma camada LSTM de 500 neurônios e uma camada Dense com 12 neurônios.
- Algoritmo otimizador Adam.
- Função de ativação Sigmóide.
- 3 variáveis - ICMS, PIB e Consumo de Energia Elétrica Comercial da Região Sudeste.

## 4.6 Comparação entre os Melhores Modelos

Após a execução dos cenários de teste e comparação dos melhores modelos de cada arquitetura, podemos chegar como resultado a Tabela 4.17 a seguir:

Tabela 4.17: Comparação dos resultados dos melhores cenários de experimento das arquiteturas LSTM. Elaborado pelo autor.

| Erro Médio de Validação | | | | | |
|---|---|---|---|---|---|
| Modelo | Número de Variáveis | Número de Neurônios | Otimizador | Função de Ativação | MAPE (%) |
| LSTM Univariate | 1 | 550 | Sgd | Relu | 6.78 |
| LSTM Multivariate | 2 | 400 | Adam | Sigmóide | 6.17 |
| LSTM Multivariate | 3 | 500 | Adam | Sigmóide | 5.21 |

Portanto, podemos concluir que a LSTM *Multivariate* com a utilização de 3 variáveis, sendo elas o ICMS, PIB e Consumo de Energia Elétrica Comercial da Região Sudeste, conseguiu o melhor desempenho quanto a métrica de erro MAPE, com 5.21%, e por conseguinte, foi o melhor modelo encontrado.

Seguido dela temos a LSTM *Multivariate* com a utilização de 2 variáveis, sendo elas o ICMS e o PIB, com erro de MAPE de 6.17%.

E por último, temos o modelo da LSTM *Univariate* com a utilização de 1 variável, que é o valor endógeno do ICMS. Ela atingiu em seu melhor caso o erro de MAPE de 6.78%.

Dessa forma, conclui-se que a utilização de variáveis externas com alta correlação no modelo LSTM é o mais benéfico na previsão anual do valor do ICMS do estado do Rio de Janeiro, sendo melhor até 2 variáveis exógenas do que 1.

## 4.7 Avaliação dos Melhores Modelos

Como comparativo, foi gerado o erro relativo das previsões para os anos de 2017 e 2018, que é o disponível pelo SEFAZ-RJ (2022) e foi analisado o desempenho dos melhores modelos LSTM *Univariate*, LSTM *Multivariate* com 2 variáveis, LSTM *Multivariate* com 3 variáveis e a previsão feita pela SEFAZ-RJ.

O erro relativo é o valor absoluto da diferença do valor previsto pelo valor real, com o resultado sendo divido pelo valor real.

A Tabela 4.18 mostra essa comparação, onde é possível concluir que as arquiteturas LSTM possuíram melhor desempenho que a atual previsão feita pela SEFAZ-RJ. Além disso, é possível analisar que o comportamento dos experimentos se manteve, sendo a LSTM *Multivariate* com as variáveis ICMS, PIB e Consumo de Energia Elétrica Comercial na Região Sudeste o melhor desempenho entre as redes neurais artificiais analisadas.

Tabela 4.18: Comparação dos resultados dos melhores cenários de experimento das arquiteturas LSTM com a previsão do SEFAZ-RJ, para os anos de 2017 e 2018. Elaborado pelo autor.

| | | Ano 2017 | | Ano 2018 | |
|---|---|---|---|---|---|
| Modelo | Número de Variáveis | Valor (R$ mil) | Erro Relativo | Valor (R$ mil) | Erro Relativo |
| Valor Real | - | 32.362.495,90 | - | 35.836.058,09 | - |
| SEFAZ-RJ | - | 35.287.454,00 | 9.04 % | 33.746.760,00 | 5.83 % |
| LSTM Univariate | 1 | 31.975.661,56 | 1.19 % | 34.653.904,89 | 3.29 % |
| LSTM Multivariate | 2 | 32.514.114,30 | 0.46 % | 34.962.612,54 | 2.43 % |
| LSTM Multivariate | 3 | 32.495.584,51 | 0.41 % | 35.242.139,26 | 1.65 % |

É possível identificar a previsão para o ano de 2019 através do gráfico da Figura 4.1, onde observa-se que a LSTM *Multivariate* se aproxima mais do comportamento da série original no ano em relaçaõ a LSTM *Univariate*.

Por conseguinte, nota-se um comportamento mais acertivo dos dois modelos nos meses inicias e que se perde ao decorrer do ano, o que era um comportamento esperado da abordagem *Multi-step* utilizada.

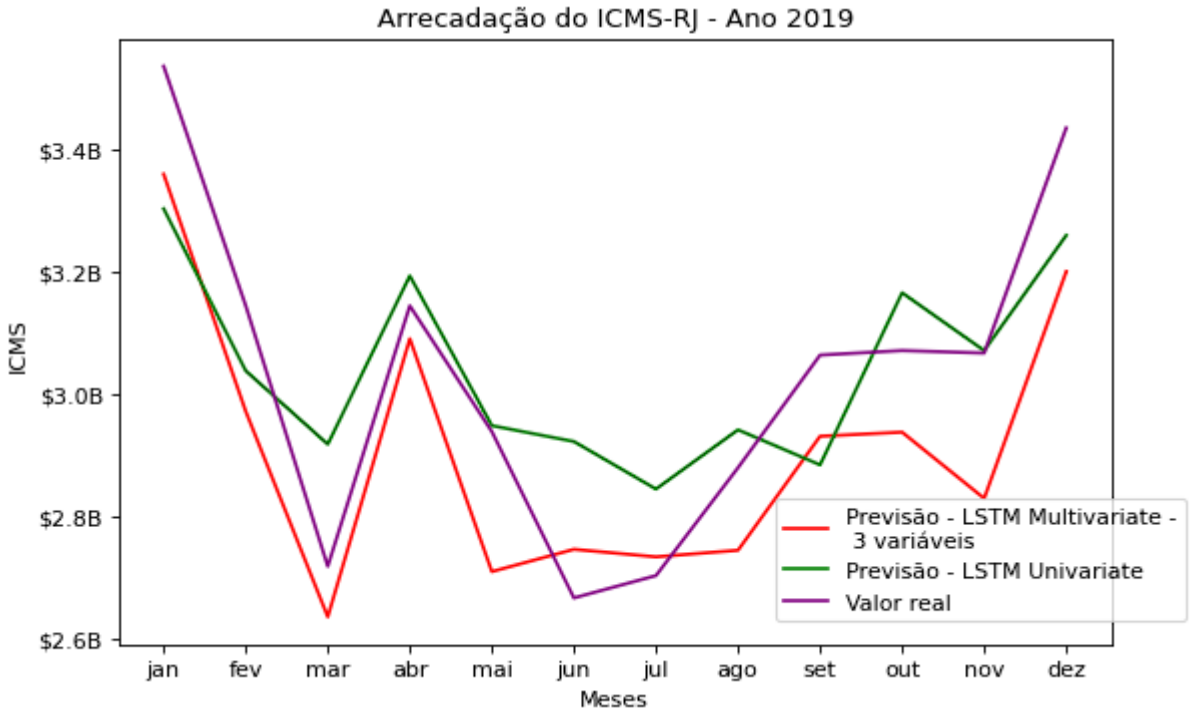


Figura 4.1: Gráfico com a arrecadação real do ICMS no ano de 2019 e as previsões da LSTM Multivariate e Univariate. Elaborado pelo autor.

# 5 Conclusão e Trabalhos Futuros

## 5.1 Conclusão

O objetivo deste trabalho foi desenvolver uma rede neural artificial para previsão de séries temporais com o modelo Long Short-Term Memory, que é uma alternativa para a resolução do problema de memória de longo prazo das redes recorrentes tradicionais, através da implementação de três portões e células de memória.

O problema aplicado foi a necessidade de previsão anual do valor arrecadado do ICMS no estado do Rio de Janeiro. Como o estado está em situação de crise econômica e atualmente faz parte do Regime de Recuperação Fiscal, o planejamento fiscal é primordial, sendo necessária a previsão anual do valor do ICMS para a Lei Orçamentária Anual (LOA). A Secretaria de Estado de Fazenda do Rio de Janeiro (SEFAZ-RJ) já realiza a previsão anual, porém utilizando modelos estatísticos matemáticos.

De acordo com o trabalho de Silva e Figueiredo (2020), os dados utilizados foram do período de janeiro de 2002 até dezembro de 2016 para a etapa de treinamento, janeiro de 2017 até dezembro de 2018 para a validação e janeiro de 2019 até dezembro de 2019 para o teste.

Sendo assim, foram desenvolvidos dois modelos distintos de LSTM na tentativa de comparar a previsão anual do imposto entre eles e com o atual meio de previsão do estado.

O modelo LSTM *Univariate Multi-step* foi utilizado com sua única variável endógena sendo o valor arrecadado do ICMS e atingiu resultados superiores aos encontrados no atual modelo utilizado pelo estado carioca.

Já o modelo LSTM *Multivariate Multi-step* foi separado de duas formas, uma utilizando apenas duas variáveis, que foram o ICMS e o PIB, e um com três variáveis, utilizando o ICMS, PIB e Consumo de Energia Elétrica Comercial da Região Sudeste. O modelo com duas variáveis apresentou melhor desempenho nos experimento que o modelo *Univariate*, e por conseguinte, também superou o atual modo de previsão do SEFAZ-

RJ. O modelo com três variáveis foi o que apresentou melhor acurácia entre todos nos experimentos e também superou a previsão do orgão governamental carioca.

É importante ressaltar que a presença de mais dados da série temporal pode favorecer ainda mais o desempenho da rede neural. Foi feito a tentativa de adicionar mais camadas LSTM e camadas Dense, porém não atingiram resultados superiores. Também foi feita a tentativa de utilizar o modelo *Multi-step* auto regressivo, que atualiza os dados de previsão nas entradas das próximas previsões, porém não foi possível chegar a resultados satisfatórios com essa abordagem para a previsão anual. Foi elaborado também durante o trabalho um modelo de Multi-task Learning para as arquiteturas *Multivariate*, porém a necessidade de apenas prever os valores do ICMS e os resultados do modelo não ultrapassarem o modelo *Multivariate* implementado, mostram que é uma abordagem complexa para o cenário utilizado e que não gerou retornos compatíveis.

Portanto, concluímos que o modelo LSTM *Multivariate Multi-step* é mais eficiente na previsão das séries temporais anuais do ICMS no estado do Rio de Janeiro e pode ser utilizado pelos orgãos públicos nas previsões do imposto.

## 5.2 Trabalhos Futuros

Em trabalhos futuros, podem ser implementados a utilização de mais variáveis exógenas de alta correlação, na tentativa de atingir melhor desempenho e acertividade, incluindo também números diferentes de horizontes de entrada e até de saída.

Caso a previsão dos valores das variáveis além do ICMS também seja necessário, indica-se a utilização da abordagem *Multi-task Learning*, que pode gerar o aprendizado em compartilhado das variáveis ao mesmo tempo que as saídas sejam separadas.

Outra abordagem indicada é a de utilização de outras técnicas de redes neurais artificias, como a *Gated Recurrent Unit* (GRU) (CHO et al., 2014), sendo realizadas de forma separada ou em conjunto com as camadas LSTM.

# Bibliografia

BARRON, A. R. Universal aproximation bounds for superpositions of a sigmoidal function. In: *IEEE Transactions on Information Theory*. [S.l.]: journal, 1993. p. 930–945.

CAMPOS, L. Modelo estocástico periódico baseado em redes neurais. 2010.

CASTANHO, B. Modelos para previsão de receitas tributárias: O ICMS do Estado do Espírito Santo. Universidade Federal do Espírito Santo, 2011.

CHEVILLON, G. Direct multi-step estimation and forecasting. Journal of Economic Surveys 21(4), p. 746–785, 2007.

CHO, K. et al. *Learning phrase representations using RNN encoder–decoder for statistical machine translation*. 2014. 187-253 p. Disponível em: ⟨https://arxiv.org/pdf/1406.1078v3.pdf⟩. Acesso em: 21 jul. 2022.

GOVERNO. *Receitas e Despesas*. Disponível em: ⟨https://www.gov.br/economia/pt-br/acesso-a-informacao/receitas-e-despesas⟩. Acesso em: 04 mar. 2022.

GUILLÉN-NAVARRO, M. et al. A decision support system for water optimization in anti-frost techniques by sprinklers. 2020.

HAYKIN, S. In: *Neural Networks: A Comprehensive Foundation*. 2nd. ed. Patparganj, India: Prentice-Hall, 1999.

HAYKIN, S. In: *Redes Neurais: Princípios e Prática*. Porto Alegre, RS: Bookman, 2001.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. Neural Computation Anais, 1997.

IBGE. *Cidades e Estados*. 2021. Disponível em: ⟨https://www.ibge.gov.br/cidades-e-estados⟩. Acesso em: 21 jul. 2022.

NWANKPA, C. et al. Activation functions: Comparison of trends in practice and research for deep learning. arXiv, 2018. Disponível em: ⟨https://arxiv.org/abs/1811.03378⟩. Acesso em: 21 jul. 2022.

PEARSON, K. Mathematical contributions to the theory of evolution III. Philos Trans R Soc Lond Ser B Biol Sci, p. 187–253, 1896.

PESSOA, F.; CORONEL, D.; LIMA, J. Previsão de Arrecadação de ICMS para o Estado de Minas Gerais: Uma Comparação entre modelos Arima e Arfima. Revista Brasileira de Gestão e Desenvolvimento Regional, 2011. Disponível em: ⟨https://rbgdr.net/revista/index.php/rbgdr/article/view/1020⟩. Acesso em: 21 jul. 2022.

PLANALTO. *Lei Complementar Nº 87, de 13 de maio de 1996. Institui o ICMS*. 1996. Disponível em: ⟨http://www.planalto.gov.br/ccivil_03/leis/LCP/Lcp87.htm⟩. Acesso em: 04 mar. 2022.

PLANALTO. *Lei Complementar Nº 101, de 04 de maio de 2000. Estabelece normas de finanças públicas voltadas para a responsabilidade na gestão fiscal*. 2000. Disponível em: ⟨http://www.planalto.gov.br/ccivil_03/leis/lcp/lcp101.htm⟩. Acesso em: 04 mar. 2022.

PRUDENCIO, R. Projeto híbrido de redes neurais. Universidade Federal de Pernambuco, Recife, PE, 2002.

SCHEFFER, D.; SOUZA, A.; ZANINI, R. Utilização de Modelos Arima Para Previsão da Arrecadação de ICMS do Estado do Rio Grande do Sul. 2014.

SEFAZ-RJ. *Resolução nº 33 do Conselho de Supervisão do Regime de Recuperação Fiscal do Estado do Rio de Janeiro*. 2020. Disponível em: ⟨http://www.fazenda.rj.gov.br/sefaz/content/conn/UCMServer/path/Contribution%20Folders/transparencia/RecuperacaoFiscal/docs/item%207/Resolucoes/SEI_ME%20-%209673492%20-%20Resoluc%cc%a7a%cc%83o%2033-2020.pdf?lve⟩. Acesso em: 22 jul. 2022.

SEFAZ-RJ. *Transparência da Receita Estadual*. 2022. Disponível em: ⟨https://portal.fazenda.rj.gov.br/transparencia-da-receita-estadual/⟩. Acesso em: 21 jul. 2022.

SILVA, P. F.; FIGUEIREDO, K. Aprendizado profundo aplicado na previsão de receita tributária utilizando variáveis endógenas. ENIAC, 2020. Disponível em: ⟨https://sol.sbc.org.br/index.php/eniac/article/view/12147⟩. Acesso em: 21 jul. 2022.

TRANSPARÊNCIA, P. *Orçamento público*. Disponível em: ⟨https://www.portaltransparencia.gov.br/entenda-a-gestao-publica/orcamento-publico#:~:text=O%20processo%20de%20planejamento%20envolve,Lei%20Or%C3%A7ament%C3%A1ria%20Anual%20(LOA).⟩ Acesso em: 04 mar. 2022.

YU, Y. et al. A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. arXiv, 2019. Disponível em: ⟨https://arxiv.org/abs/1706.05098⟩. Acesso em: 21 jul. 2022.

ZHANG, Y.; YANG, Q. An overview of multi-task learning. 2017.

ZSOLT, L. In: *Redes Neurais Artificias: Fundamentos e Aplicações*. 4. ed. [S.l.]: Livraria da Física, 2006.