Universidade de São Paulo
Instituto de Matemática e Estatística
Bacharelado em Ciência da Computação

Previsão de consumo nos estados do Brasil

Julia Leite da Silva

Monografia Final

mac 499 — Trabalho de Formatura Supervisionado

Supervisor: Prof. Dr. Marcelo Finger

São Paulo
2022

Dedico este trabalho a minha família,
meu pai e minha mãe que têm iluminado
meu dia e me dado força, alegria e paz.

Agradecimentos

Fight with determination, embrace life and live it with passion. Lose your battles with class and dare to win because the world belongs to those who dare to live. Life is worth too much to be insignificant.
— Charles Chaplin

Queria agradecer, antes de tudo, a Deus por tudo que Ele tem me dado. Uma vez minha mãe me disse que Deus abençoa o trabalho das nossas mãos, que várias vezes onde ela pensou que não daria certo ou não daria conta, ela pedia ajuda e, no final, tudo se endireitava. Ela me disse também que Deus colocava pessoas na vida dela que a ajudavam e orientavam a encontrar um caminho. Fico muito grata de enxergar isso acontecendo na minha vida também.

Agradeço aos meus pais, que têm sido minha força, alegria e lar. Agradeços aos meus amigos, amigas e namorado, por colorir meus dias e encher meu rosto de sorrisos.

Quero também agradecer ao Marcelo Finger, meu orientador e ao Felipe, que me surpreendeu com a parceria e a nessa jornada. Muito ajuda. Agradecer também ao Gabriel, pelo apoio para escrever esse trabalho.

Resumo

Julia Leite da Silva. **Previsão de consumo nos estados do Brasil**. Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2022.

Elemento obrigatório, constituído de uma sequência de frases concisas e objetivas, em forma de texto. Deve apresentar os objetivos, métodos empregados, resultados e conclusões. O resumo deve ser redigido em parágrafo único, conter no máximo 500 palavras e ser seguido dos termos representativos do conteúdo do trabalho (palavras-chave). Deve ser precedido da referência do documento. Texto texto. Texto texto.

Palavras-chave: Palavra-chave1. Palavra-chave2. Palavra-chave3.

Abstract

Julia Leite da Silva. . Capstone Project Report (Bachelor). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2022.

Elemento obrigatório, elaborado com as mesmas características do resumo em língua portuguesa. De acordo com o Regimento da Pós-Graduação da USP (Artigo 99), deve ser redigido em inglês para fins de divulgação. É uma boa ideia usar o sítio www.grammarly.com na preparação de textos em inglês. Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text. Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text.

Keywords: Keyword1. Keyword2. Keyword3.

Lista de Abreviaturas

IA	Inteligência Artifical
ML	*Machine Learning*
IBGE	Instituto Brasileiro de Geografia e Estatística
FGV	Fundação Getúlio Vargas
PIB	Produto Interno Bruto
BACEN	Banco Central do Brasil
IPEA	Instituto de Pesquisa Econômica Aplicada
RMSE	*root-mean-square error*
MAPE	*mean absolute percentual error*
MAE	*mean absolute error*
PNAD	Pesquisa Nacional por Amostra de Domicílios
INCC	Índice Nacional de Custo da Construção Civil
IPCA	Índice Nacional de Preços ao Consumidor Amplo
IGP	Índice Geral de Preços
NFSP	Necessidade de Financiamento do Setor Público
IDH	Índice de Desenvolvimento Humano
SNIC	Sindicato Nacional da Indústria do Cimento
LSTM	*Long Short Term Memory*
GRU	*Gated Recurrent Units*

Lista de Símbolos

ω	Frequência angular
ψ	Função de análise *wavelet*
Ψ	Transformada de Fourier de ψ

Lista de figuras

Lista de tabelas

Lista de programas

Sumário

Introdução

O que prédios, pontes, hidrelétricas e aeroportos têm em comum? Todos são frutos da indútria da contrução civil, um importante componente do investimento brasileiro e, consequentemente, uma das grandes engrenagens responsáveis por movimentar a atividade econômica no Brasil. Em 2021, por exemplo, o Produto Interno Bruto (PIB) desse setor destacou-se com alta de 9,7%, enquanto o PIB do Brasil cresceu 4,6% e o PIB do Agronegócio registrou queda de 0,2%. Dessa forma, o setor da construção figura como importante impulsionador da economia do país e como fonte de renda e de empregos. (Vasconcelos, 2022)

Nesse contexto, o cimento[1] é uma das principais matérias primas da indústria da construção civil. Esse material é um pó com propriedades aglomeramentes que endurece quando é submetido água. Após endurecer, não é mais decomposto, mesmo em contato com a água. Além disso, ao ser misturado com areia, pedra britada, pó-de-pedra e cal resulta na argamassa e no concreto utilizados em construções. (Cimento Portland, 2002)

Motivação

Apesar disso, a falta de um modo bem fundamentado para prever o consumo de cimento nos estados do Brasil é uma demanda entre as empresas cimenteiras. Uma nova fábrica representa para a empresa um alto investimento a longo prazo, uma vez que a contrução exige grande capital financeiro, além de tempo, por demorar anos para finalizar as obras.

Um exemplo é a fábrica construída pela Votorantim Cimentos no Complexo Industrial e Portuário do Pecém, no Ceará, cuja construção demorou 3 anos e custou cerca de 200 milhões de reais, conforme noticiado no portal do Instituto Brasileiro de Mineração (IBRAM, 2021). Além disso, o cimento é um produto que não suporta longas estadias em estoque, visto que a norma brasileira recomenda o uso em até 90 dias após a fabricação – importante, então, que a produção do material esteja fortemente alinhada ao consumo. (Normas Técnicas, 2018)

Assim, um modelo que permitisse prever a demanda por cimento a nível de estados do Brasil poderia auxiliar gestores a tomar melhores decisões e a estruturar a estratégia

[1] O material conhecido na contrução civil como "cimento"é denominado mundialmente como cimento de *portland*. Em 1824, construções com pedra de *portland* eram comuns na Inglaterra, por isso, o inventor do cimento, Joseph Aspdin, ao notar que sua invenção tinha aspecto similar ao material tão difundido na época, optou por registrar a patente como "cimento de *portland*".

de forma mais embasada, de modo a reduzir os riscos do setor. Além disso, poder-se-ia apoiar órgãos governamentais a direcionar ações para mitigar o impacto ambiental da fabricação desse produto, pontuado como um grande emissor de gases de efeito-estufa em escala mundial. (Carvalho, 2008)

Objetivos

Este trabalho, então, propõe-se a aplicar modelos de aprendizado de máquina para determinar qual é mais eficiente para prever a demanda por cimento nos estados do Brasil. Os algoritmos avaliados em ordem crescente de complexidade e robustez são: regressão linear, redes neurais *multi-layer* e redes neurais recorrentes.

Capítulo 1

Fundamentação teórica

1.1 Inteligência artificial

Atualmente, a inteligência artificial (IA) permeia diversos momentos do cotidiano. Um exemplo é a empresa norte-americana de *streaming* Netflix, que utiliza um conjunto de técnicas de inteligência artificial para recomendar conteúdo personalizado aos usuários da plataforma, de acordo com os interesses particulares de cada um. Em particular para a Netflix, não há um modelo ou algoritmo único utilizado para todas as recomendações de conteúdo. Essa tarefa é dividida em subtarefas realizadas por diferentes modelos, de acordo com a atividade a ser realizada e os dados disponíveis. Por exemplo, a escolha de qual vídeo será exibido para o usuário ao logar no perfil da plataforma é executada por um modelo diferente do que o que elenca os vídeos já assistidos que o membro pode continuar a ver.(Steck *et al.*, 2021)

Dessa forma, a empresa proporciona uma experiência única a cada indivíduo que acessa a plataforma. Essa estratégia tem o objetivo de aumentar a satisfação a longo prazo do cliente e garantir a retenção dos membros, uma vez que a plataforma é monetizada com assinaturas mensais. O artigo ainda ressalta que a estratégia de utilizar inteligência artificial para as recomendações tem se refletido ao longo dos anos com uma melhora na taxa de retenção dos membros.

> *"Therefore, the value of a recommender system can be measured by the increase in member retention. Over years of the development of personalization and recommendation technologies, we have been able to repeatedly create meaningful improvements in retention"* (Steck *et al.*, 2021)

Mas afinal, o que é inteligência artificial? O termo, do inglês *artificial intelligence*, foi elaborado por John McCarthy e utilizado oficialmente pela primeira vez em 1956 no seminário de Dartmouth, um *workshop* sobre a área que reuniu os maiores estudiosos do ramo durante dois meses (Russell e Norvig, s.d.). Em 1950, entretanto, Alan Turing já se perguntava se máquinas poderiam pensar e desenvolvia estudos e conceitos no tema que

permanecem relevantes, como o Teste de Turing[1]. O termo "inteligência artificial"pode ser utilizado com várias conotações, uma vez que não apresenta uma definição única e aceita Wang, 2019. Uma possível definição é a ciência e engenharia de construir máquinas inteligentes, em especial programas de computador. Nesse contexto, inteligência é o aspecto computacional da habilidade de atingir objetivos(McCarthy, 2007).

> *"The science and engineering of making intelligent machines, especially intelligent computer programs. (...) Intelligence is the computational part of the ability to achieve goals in the world."* (McCarthy, 2007)

Em 1959, o pioneiro em IA Arthur Samuel[2] descreveu aprendizado de máquina como o campo de estudo que dá aos computadores a habilidade de aprender sem serem especificamente programados. Aprendizado de máquina ou *machine learning*, portanto, compreende sistemas de inteligência artificial capazes de adquirir seu próprio conhecimento ao extrair padrões dos dados brutos (Goodfellow *et al.*, 2016). *Machine learning*, então, configura-se como uma sub-área da inteligência artificial.

Já o aprendizado profundo, ou *deep learning*, é uma categoria específica de algoritmos de *machine learning*. Caracteriza-se por modelos que fazem processamento de dados com neurônios matemáticos de forma a imitar o funcionamento do cérebro humano (DSA, 2022). Nesses algoritmos, a informação é passada de camada em camada até obter a saída. Trata-se, portanto, de redes neurais com várias camadas ocultas de neurônios (Zhang *et al.*, 2021a).

A relação entre inteligência artificial, aprendizado de máquina e aprendizado profundo pode ser vista na imagem 1.1.

[1] Em 1950, Alan Turing propôs o Teste de Turing com a intenção de determinar se uma máquina era inteligente. Esse teste é uma variação do Jogo da Imitação, em que um entrevistador deve fazer perguntas a dois jogadores, um humano e uma máquina, sem qualquer distinção. Ao final, o entrevistador deve descobrir qual dos jogadores é uma máquina e qual é a pessoa. Se a máquina fosse capaz de enganar o entrevistador, seria considerada inteligente.(Turing, 1950)

[2] Arthur Samuel foi um engenheiro e um dos pioneiros em inteligência artificial e desenvolveu um programa que jogava damas com humanos e aprendia com cada jogada dos oponentes. O programa tornava suas jogadas mais assertivas ao calcular as probabilidades de cada jogada e é considerado uma das primeiras aplicações de aprendizado de máquina.

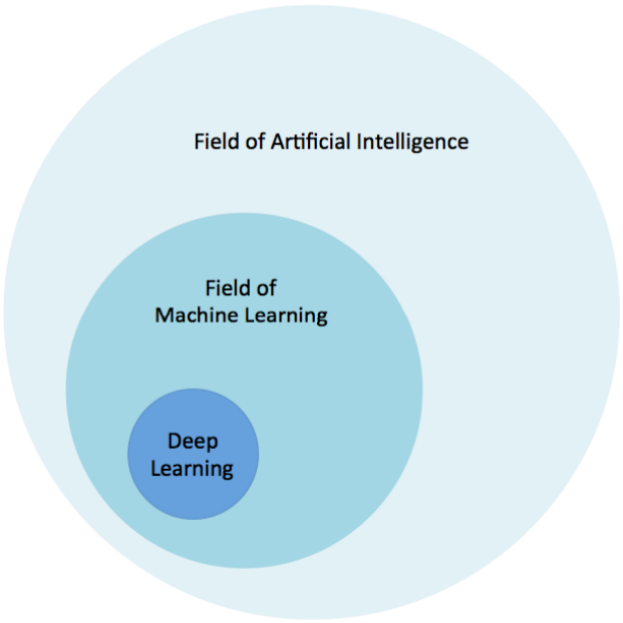


Figura 1.1: *Relação entre inteligência artificial, aprendizado de máquina e aprendizado profundo (Patterson e Gibson, s.d.)*

1.2 Tarefa

Os modelos de *machine learning* podem realizar várias categorias de tarefas, dentre elas a regressão e a classificação, a depender da atividade realizada pelo algoritmo. Neste trabalho, foram utilizados modelos de regressão cujo objetivo é prever um valor real a partir dos dados de entrada (Goodfellow *et al.*, 2016). É possível, então, descrever cada *input* dos modelos como um vetor x com n atributos (*features*)[3] tal que $x \in \mathbb{R}^n, x = \{x_1, x_2, ..., x_n\}$ e o processamento realizado pelos modelos de regressão como $f : \mathbb{R}^n \to \mathbb{R}$.

Os modelos de classificação, por outro lado, têm como objetivo determinar a qual das k categorias disponíveis um *input* pertence (Patterson e Gibson, s.d.). Dessa forma, é utilizada uma função $f : \mathbb{R}^n \to \{1, ..., k\}$, quando $f(x) = y$, o vetor de entrada x foi classificado na categoria y. Um exemplo de tarefa de classificação seria determinar se uma operação com cartão de crédito é fraudulenta ou não. Neste trabalho, contudo, não foi empregado tal tipo de modelo.

Os problemas de aprendizado de máquina também podem ser divididos entre aprendizado não-supervisionado e supervisionado. No primeiro, o modelo recebe um conjunto de dados (*dataset*) não rotulado e aprende propriedades da estrutura dos dados. Um exemplo de aprendizado não supervisionado é a clusterização, que consiste em dividir o conjunto de dados em *clusters* com amostras similares. No último, por sua vez, os dados de entrada estão associados a resultados conhecidos, chamados de *labels* ou rótulos(Goodfellow *et al.*, 2016). Neste trabalho, utiliza-se aprendizado supervisionado para prever o consumo

[3] Neste trabalho, as *features* utilizadas são indicadores econômicos, monetários, sociais e da construção civil descritos na seção 2.2.

de cimento mensal nos estados da União a partir dos dados de entrada e compará-lo com o valor real do consumo e, assim, calcular a precisão do modelo.

1.3 Modelos utilizados

Neste trabalho, utilizaram-se três categorias de modelos de aprendizado de máquina para prever a demanda por cimento: regressão linear, redes neurais *multilayer* e redes recorrentes. Foram testados, também, métodos de pré-processamento de dados (descritos na seção 2.2.3) e diferentes arquiteturas de redes neurais ao alterar o número de camadas, a quantidade de neurônios em cada camada, o tipo de camada, a função de ativação, entre outras configurações, com o objetivo de comparar o desempenho dos modelos e encontrar o que apresenta menor erro na previsão.

1.3.1 Regressão linear

A regressão linear é um modelo de aprendizado de máquina que assume um relacionamento linear entre a variável que será prevista (*target*) e os dados de entrada. Desse modo, seu objetivo é construir uma função que, para cada par[4] (x, y), recebe como entrada o vetor x que corresponde às variáveis de *input*, $x \in \mathbb{R}^k, x = \{x_1, x_2, ..., x_k\}$ e calcula coeficientes $\beta = \{\beta_0, \beta_1, \dots, \beta_k\}$ para cada atributo de x, além da constante β_0. O algoritmo, então, utiliza esses coeficientes para prever um valor para a variável *target*, y, sendo que $y \in \mathbb{R}$ (Hyndman e Athanasopoulos, 2021). Seja, então, $\hat{y}$ o valor previsto pelo modelo para um par (x, y), a função que descreve a regressão linear, então, dada por

$$\hat{y} = f(x) = \beta_0 + \beta_1 x_{1_i} + \beta_2 x_{2_i} + \cdots + \beta_k x_{k_i} \tag{1.1}$$

Logo, é possível contruir uma matriz X em que a linha i corresponde ao vetor x_i dos dados de entrada e cada coluna j representa uma *feature*. Pode-se, além disso, construir a matriz β dos coeficientes associados a cada elemento da matriz X. Assim, o modelo é dado por:

$$y = X\beta + \epsilon \tag{1.2}$$

Na equação 1.2, ϵ é o vetor com o erro associado a cada umas das previsões, tal que $\epsilon_i = y_i - \hat{y}_i = y - (\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k)$. Os coeficientes β são calculados durante o treinamento do modelo utilizando o método do *least squares estimation* que visa minimizar a soma do erro quadrado associado às previsões, como descrito em:

$$\sum_{i=1}^{n} \epsilon_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{1_i} + \beta_2 x_{2_i} + \cdots + \beta_k x_{k_i}))^2 \tag{1.3}$$

[4] No caso deste estudo, o par (x, y) é tal que x representa o valor de cada indicador descrito na seção 2.2 em um estado, mês e ano e y corresponde ao número de toneladas de cimento consumidas por esse estado nessa data.

Os coeficientes $\beta_1, \beta_2, \ldots, \beta_k$ determinam como cada atributo de entrada afeta a previsão. Por exemplo, se o coeficiente da variável x_i for $\beta_i = 0$, essa variável não tem influência no valor previsto pelo modelo. Caso o coeficiente seja positivo, por outro lado, um aumento no valor de x_i resulta em aumento no valor previsto $\hat{y}_i$, já se β_i for negativo, um aumento no valor de x_i se reflete na diminuição no valor de $\hat{y}_i$.

Na imagem 1.2, há um exemplo de um modelo de regressão linear com apenas uma variável:

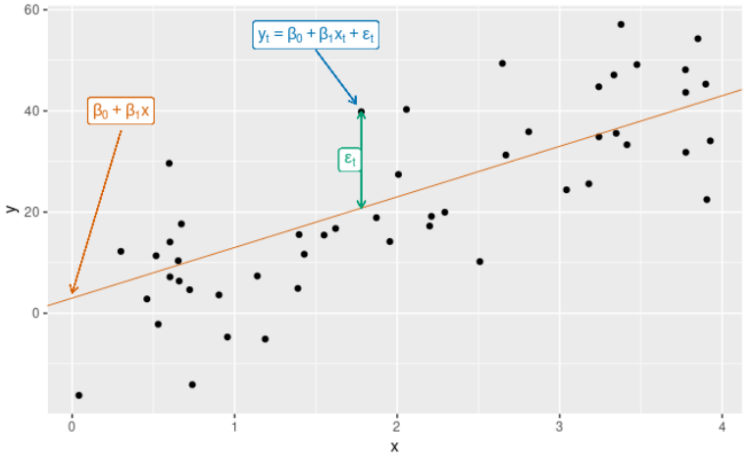


Figura 1.2: *Exemplo de um modelo simples de regressão linear (Hyndman e Athanasopoulos, 2021)*

Na figura 1.2, as observações y_i estão representadas pelos pontos pretos, enquanto a linha em laranja corresponde à previsão realizada pelo modelo. Observa-se que o modelo não prevê com total exatidão os dados observados, há um erro associado a cada previsão, como o destacado em verde na ilustração, comportamento esperado, já que fenômenos previstos são sujeitos a fatores externos e não lineares.

Por se tratar de um modelo mais simples, os resultados obtidos com a regressão linear são utilizados neste trabalho como base para comparar o desempenho de modelos mais robustos.

1.3.2 Redes neurais

Redes neurais são modelos computacionais inspirados no funcionamento do cérebro animal. O cérebro é formado por neurônios que se conectam para transmitir informações sem a necessidade de uma unidade central de controle. Um neurônio biológico, então, é uma célula nervosa que se comunica com outros neurônios por meio de impulsos eletroquímicos. Essa comunicação é denominada sinapse e ocorre apenas se o impulso for forte o bastante para ativar a liberação de químicos na fenda sináptica (DSA, 2022). Um neurônio é composto por vários dendritos, um axônio e um corpo celular, como ilustrado em 1.3.

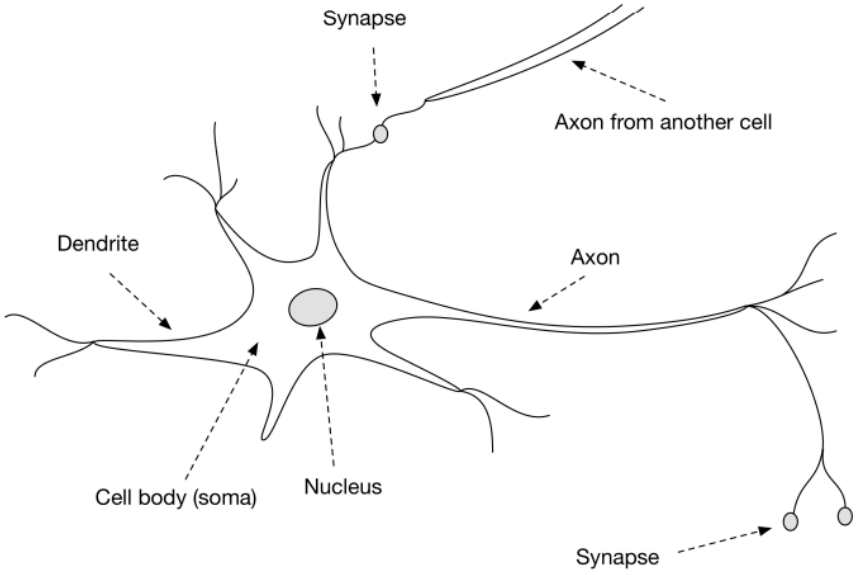


Figura 1.3: *Ilustração de um neurônio biológico (Patterson e Gibson, s.d.)*

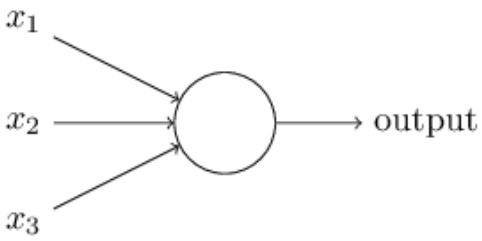
Destaca-se na figura 1.3 a informação chegando a um dendrito do neurônio em destaque por meio de uma sinapse, também está representada a comunicação com outra célula por meio de outra sinapse iniciada no axônio do neurônio em questão.

O processo de propagação da informação nos neurônios envolve as três partes da célula: dendritos, corpo celular e axônio. Os dendritos recebem informações de neurônios vizinhos na forma de impulsos elétricos e são responsáveis por conduzí-las até o corpo celular. Ao chegar ao local, a informação é processada, e novos impulsos são gerados e repassados a outro neurônio por meio do axônio no processo de sinapse (Nicholas Lacascio, 2017). A estrutura e funcionamento dos neurônios biológicos inspiraram os cientistas ao projetarem neurônios artificiais, como os *perceptrons*.

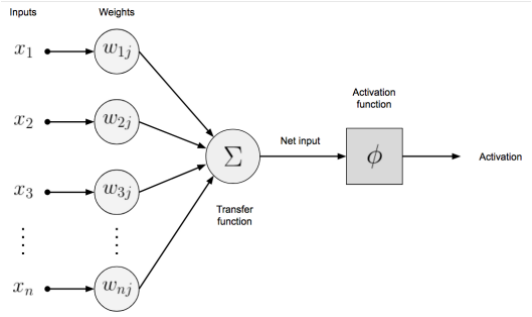
Neurônios artificiais

O *perceptron* foi desenvolvido em 1957 por Frank Rosenblatt, inspirado nos trabalhos de Warren McCulloch e Walter Pitts[5]. Trata-se de um modelo linear de classificação que recebe *n* entradas e produz uma saída binária, como mostrado na ilustração simplificada 1.4a (DSA, 2022). Esse modelo inicial apresentava limitações e passou por evoluções com o passar do tempo. Apesar disso, as redes neurais atualmente utilizam, em geral, outro modelo de neurônio, como o ilustrado em 1.4b.

[5] Em 1943, Warren McCulloch e Walter Pitts apresentaram a primeira ideia de neurônio artificial.(Walter Pitts, 1943)



(a) *Visão simplificada de um neurônio (DSA, 2022)*



(b) *Ilustração da arquitetura de um neurônio artificial (Patterson e Gibson, s.d.)*

A figura 1.4b mostra o funcionamento de um neurônio artificial. O neurônio apresenta n entradas x_i, cada uma associada a um peso w_i, que expressa a importância da respectiva entrada para o valor de saída[6]. O produto escalar entre os pesos e as respectivas entradas passa por uma função de ativação[7] ϕ que determina a saída do neurônio. Além disso, um valor de *bias* ou polarização é adicionado ao produto escalar e possibilita que um neurônio com todas as entradas nulas apresente saída não nula, de modo a aumentar a capacidade de aproximação da rede. (DSA, 2022)

Seja x o vetor das n entradas do neurônio, $x \in \mathbb{R}^n, x = \{x_1, x_2, ..., x_n\}$, e seja w o vetor com os pesos associados a cada entrada, $w \in \mathbb{R}^n, w = \{w_1, w_2, ..., w_n\}$. Além disso, seja b o valor de *bias* e ϕ a função de ativação. Dessa forma, o *output* h de um neurônio é dado por:

$$h_{w,b} = \phi(w \cdot x + b) \tag{1.4}$$

Essa saída é utilizada como uma das entradas dos neurônios na camada seguinte, de modo a formar a estrutura das redes neurais.

Os neurônios, então, formam a unidade que compõe as redes neurais artificiais, como ilustrado na figura 1.5:

[6] O peso atribuído a uma das entradas expressa a influência desta no *output* do nó, de modo similar aos coeficientes calculados para cada atributo da regressão linear, explicada na seção 1.3.1

[7] Um detalhamento sobre as funções de ativação pode ser encontrado na seção 1.3.2

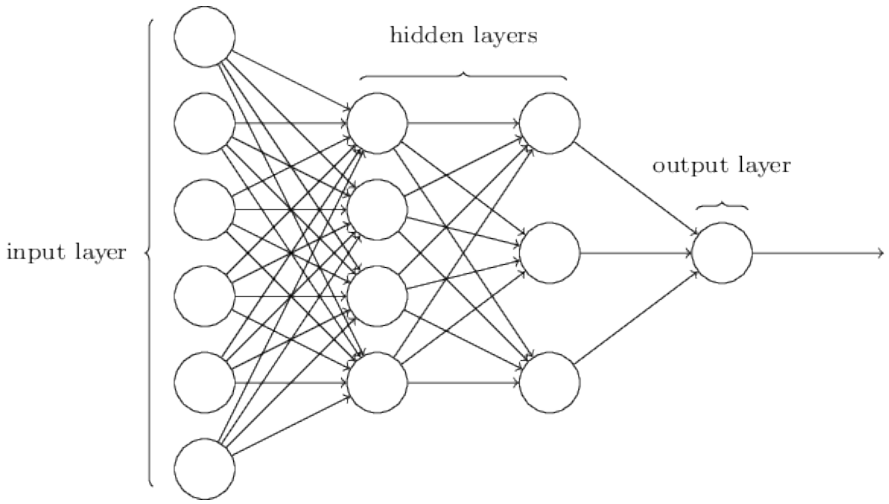


Figura 1.5: *Ilustração de uma rede neural (Nielsen, 2015)*

A imagem 1.5 representa uma simplificação de uma rede neural. Pode-se observar o fluxo da informação, primeiro recebida pelos neurônios de uma camada, então processada nessas unidades e, por fim, repassada à camada seguinte, até obter a saída final da rede. Em particular, é representada uma rede *feed forward*, uma vez que não há conexões entre neurônios de uma mesma camada nem conexões entre uma camada e a anterior. A informação então se propaga apenas no sentido da camada de entrada em direção à final, de saída.

Além disso, em 1.5, é ilustrada uma rede neural *multilayer*. Conforme a figura, a camada mais à esquerda da rede é denomida *input layer*, a camada mais à direita, de *output layer*, e as camadas intermediárias são chamadas *hidden layers*. Por razões históricas, é possível encontrar referências a essas redes como *multilayer perceptron*. Contudo, as redes em geral utilizam neurônios sigmóides ao invés de *perceptrons*.

Função de ativação

A função de ativação determina se um neurônio será ativado, ou seja, se a saída será propagada para a camada seguinte. Enquanto os pesos e o *bias* realizam uma transformação linear nos dados de entrada, a função de ativação aplica uma transformação não linear. Assim, possibilita que a rede neural resolva problemas não lineares e complexos, como reconhecer padrões de escrita (DSA, 2022 e Zhang *et al.*, 2021b).

A função de ativação é um atributo de cada uma das camadas da rede e é escolhida de acordo com a tarefa que será executada. Por exemplo, a função sigmóide é recomendada para problemas de classificação. Neste trabalho, foram utilizadas as funções *rectified linear unit* (ReLU) e *swish*.

Rectified linear unit (ReLU)

A função ReLU, do inglês *rectified linear unit*, é a função de ativação mais popular atualmente, uma vez que apresenta bom desempenho em diferentes tarefas (Patterson e Gibson, s.d.). A função ReLU é dada por:

$$f(x) = max(0, x) \tag{1.5}$$

Por possuir derivada igual a zero ou a uma constante, a ReLU não sofre do problema da dissipação do gradiente[8] como outras funções de ativação, uma vez que a derivada é utilizada nos modelos de *machine learning* para atualizar os pesos e *bias* no treinamento da rede. O gráfico da função ReLU está ilustrado em 1.6.

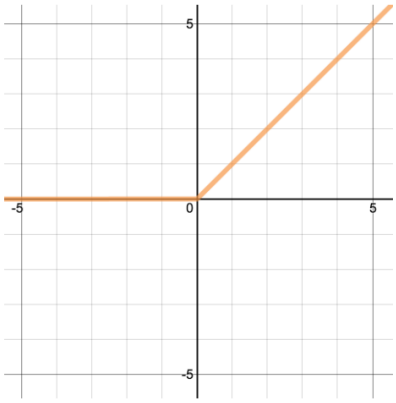


Figura 1.6: *Rectified linear unit (ReLU)* (Patterson e Gibson, s.d.)

Swish

A função *swish* foi proposta por pesquisadores da Google em 2017, com a promessa de apresentar desempenho igual ou superior à ReLU em redes neurais profundas (Ramachandran *et al.*, 2017). A *swish* é uma função não monótona[9] e suave[10], cuja fórmula é dada por:

$$f(x) = x \cdot sigmoid(x) = \frac{x}{1 + e^{-x}} \tag{1.6}$$

O gráfico da função é similar ao da ReLU e pode ser observado em 1.7.

[8] Um detalhamento sobre o problema da dissipação do gradiente será dado na seção 1.3.2.

[9] Uma função definida como $f : I \to \mathbb{R}$ é monótona se é não crescente no intervalo I ou não decrescente em I.

[10] Uma função f é suave se possui derivada de todas as ordens.

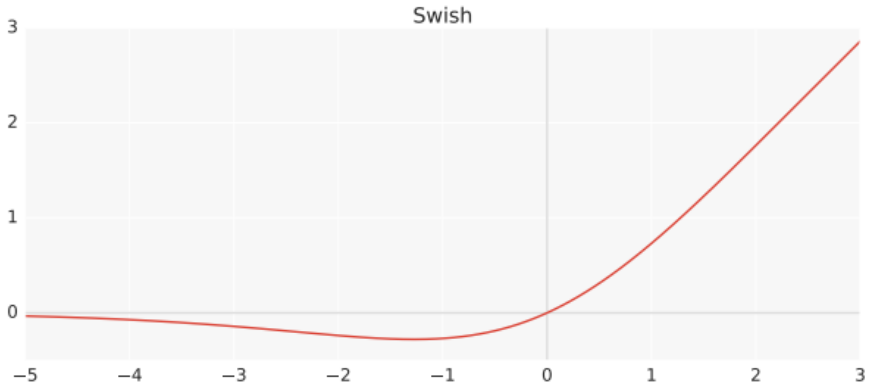


Figura 1.7: *Função swish (Ramachandran et al., 2017)*

Problema da Dissipação do Gradiente

O aprendizado de uma rede neural ocorre ao alterar os valores dos pesos e *bias* dos neurônios de acordo com a função de custo ou *loss function* (Nielsen, 2015). Sejam x, w e b os conjuntos de entradas, pesos e *bias* da rede, respectivamente. Então, a função de custo C é dada por:

$$C(w,b) = \frac{1}{2n} \sum_x ||y(x) - a||^2 \tag{1.7}$$

Na equação acima, $y(x)$ representa o vetor com as respostas que a rede tenta prever, a é o vetor de *output* da rede e n é o número total de entradas de treino. Assim, é possível calcular o gradiente δ^i_j relativo ao j-ésimo neurônio da i-ésima camada:

$$\delta^i_j = \frac{\partial C}{\partial b^i_j} \tag{1.8}$$

Seja δ^i o vetor cujos elementos estão associados aos neurônios da camada i. Como os gradientes definem a alteração nos pesos e *bias* de cada neurônio, δ^i determina a velocidade de aprendizado dessa camada.

O problema da dissipação do gradiente[11] ocorre quando os gradientes das camadas iniciais ficam com valores muito próximos a zero. Nesse caso, os pesos e *bias* não serão atualizados de forma eficiente nessas camadas, de tal forma que o treinamento e aprendizado tornam-se demasiadamente lentos. Esse problema atinge, em especial, modelos de *deep learning* com muitas camadas e redes recorrentes.

1.3.3 Redes neurais recorrentes

Redes recorrentes são uma subclasse das redes neurais e possuem a capacidade de capturar o contexto como diferencial frente às redes *feed forward* tradicionais. Por levar

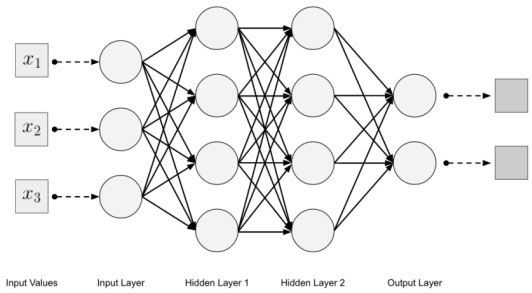
[11] O problema da dissipação do gradiente é conhecido em inglês como *the vanishing gradient problem.*

em consideração o presente e o passado recente ao realizar uma previsão, apresentam bom desempenho na previsão de séries temporais ou sequências. Há várias implementações possíveis de redes recorrentes. Neste trabalho testaram-se as redes LSTM e GRU.

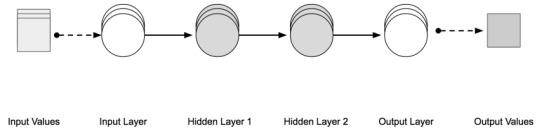
Redes LSTM

As redes *Long Short Term Memory* (LSTM) foram introduzidas em 1997 por Hochreiter e Schmidhuber[12]. Atualmente, são a variação mais utilizada de redes neurais recorrentes, em particular, na classificação de séries temporais e reconhecimento de fala. Redes LSTM apresentam células de memória conectadas cujo conteúdo é modulado pelo *input gate* e o *forget gate*[13]. Por exemplo, se ambos estiverem fechados em um instante, o conteúdo da célula permanecerá o mesmo no instante em questão e no próximo. Essa estrutura permite que a informação seja retida ao longo do tempo e evita o problema do *vanishing gradient* que ocorre com a maioria das redes recorrentes.

As redes neurais LSTM apresentam uma arquitetura diferente das redes tradicionais, uma vez que há ciclos de *feedback* nas conexões entre as células. Para melhor ilustrar essa arquitetura, Patterson e Gibson, s.d. utilizam a visualização *flat* (achatada) das redes neurais, na qual as células de uma mesma camada da rede estão representadas como um único nó. Pode-se observar essa representação em uma rede tradicional *feed forward* na imagem 1.8b e, em 1.8a, a representação tradicional.



(a) *Arquitetura de rede neural feed forward*

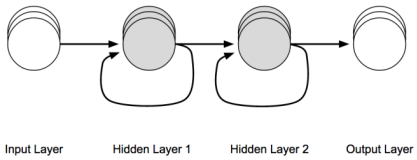


(b) *Visão flat (achatada) da arquitetura de uma rede neural feed forward*

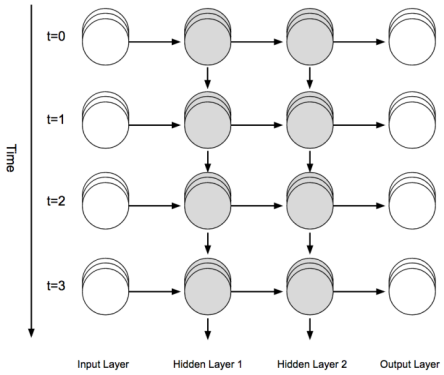
As redes LSTM introduzem o conceito de conexão entre a saída de uma camada oculta da rede neural e a entrada da mesma camada. A partir desse ciclo, obtém-se *outputs* de tempos anteriores como parte da informação que chega ao tempo atual. Na figura 1.9a, essas conexões recorrentes são representadas como as setas que saem de uma célula e atingem a mesma célula, uma vez que se utiliza a representação achatada. A imagem 1.9b, por sua vez, representa a rede LSTM desenrolada atráves do eixo do tempo.

[12] As redes neurais LSTM foram apresentadas no artigo Hochreiter e Schmidhuber, 1997.

[13] Um melhor detalhamento sobre o *input gate* e o *forget gate* será dado mais adiante nesta seção.



(a) *Visão flat (achatada) arquitetura de rede neural LSTM*



(b) *Visão da rede neural LSTM ao longo do tempo*

As unidades que formam cada uma das camadas de uma LSTM são uma variação dos neurônios artificiais clássicos. Essas unidades, representadas em 1.10, permitem que a rede mantenha o estado ao longo do tempo e apresentam conexões vindas da camada anterior e de *outputs* dessa mesma unidade em tempos anteriores.

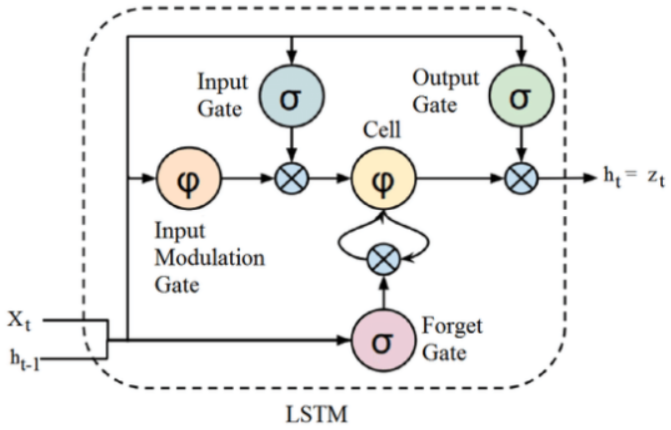


Figura 1.10: *Célula de memória de rede LSTM (DSA, 2022)*

A unidade LSTM recebe duas entradas: a observação no tempo atual e a saída do último estado oculto. Nela, a informação é retida nas células e as manipulações de memória são realizadas nos *gates* (portões). O *forget gate* é responsável por remover informações que não são mais úteis para a célula, ou seja, pelo esquecimento. Essa operação é realizada por meio de multiplicação de matrizes de pesos, adição de parâmetro de viés e aplicação de uma função de ativação que fornece uma saída binária. O *input gate*, por sua vez, adiciona informações úteis ao estado da célula a partir da aplicação de funções sigmóides e tangente hiperbólica, além da multiplicação de vetores. Por fim, o *output gate* extrai informações úteis ao estado da célula atual para apresentar como saída da célula e entrada para a seguinte, com o auxílio de funções tangente hiperbólica e sigmoide, além da multiplicação de vetores. (DSA, 2022)

Redes GRU

As redes *gated recurrent units* são redes recorrentes similares às LSTM e foram introduzidas por (Cho *et al.*, 2014). As unidades das redes LSTM apresentam dois estados passados entre as células: um que carrega memória de longo prazo e outro, de curto prazo. As unidades GRU, por sua vez, apresentam apenas um estado oculto carregado ao longo do tempo, capaz de manter as dependências de curto e longo prazo (DSA, 2022). A estrutura das unidades GRU pode ser vista em 1.11.

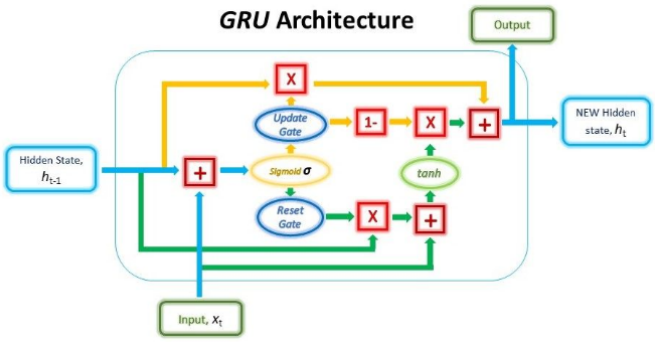


Figura 1.11: *Unidade da rede GRU (DSA, 2022)*

As unidades GRU apresentam portões, assim como as unidades LSTM, para controlar o fluxo de informações. O *reset gate*, ou portão de redefinição, é responsável por controlar quais informações das etapas anteriores serão mantidas na etapa atual, por meio da função sigmoide e multiplicação de vetores. O *update gate*, por sua vez, tem como objetivo determinar quanto das informações armazenadas no estado oculto atual são apenas cópias do estado anterior. (Zhang *et al.*, 2021a)

Por possuírem arquitetura mais simples, as redes GRU apresentam um treinamento mais rápido que as LSTM.

Redes bidirecionais

As redes bidirecionais foram introduzidas em 1997 por Schuster e Paliwal com a proposta de utilizar uma entrada para influenciar ao mesmo tempo as previsões do futuro e do passado (Schuster e Paliwal, 1997). Redes recorrentes, como as LSTM e GRU, analisam uma entrada com base nos eventos passados e presentes. As redes bidirecionais, por outro lado, inovam ao considerar o passado, presente e o futuro ao realizar uma previsão. Por isso, apresentam bom desempenho em atividades como reconhecimento de escrita a mão, reconhecimento de fala, processamento de linguagem natural. (Alla, 2021)

As redes bidirecionais combinam duas redes recorrentes para realizar a previsão. Uma rede analisa a sequência $x^{(i)}$ fornecida no *input* do início x_0 em direção ao final x_i, a outra é treinada no sentido oposto. As redes utilizadas podem ser simples redes recorrentes, redes LSTM ou GRU. Na figura 1.12, pode-se observar o treinamento das redes nos dois sentidos ocorrendo simultaneamente.

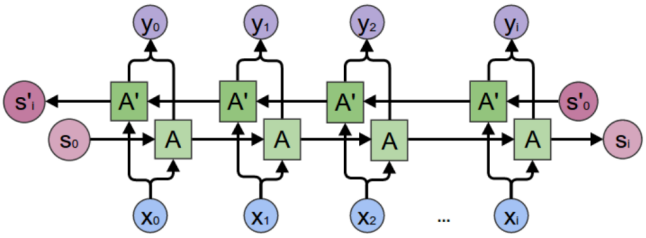


Figura 1.12: *Estrutura de rede bidirecional (Alla, 2021)*

Capítulo 2

Metodologia

Nesta seção é feito um detalhamento dos dados utilizados que abrange as fontes e as estratégias de preparação e de pré-processamento. Também são descritas as tecnologias utilizadas na implementação, as métricas estatísticas usadas para mensurar o desempenho dos modelos e o método de avaliação dos modelos.

2.1 Tecnologias utilizadas

Este projeto foi implementado na linguagem Python e utilizando o ambiente de desenvolvimento interativo fornecido pelo Jupyter. Foram utilizadas as bibliotecas Pandas e NumPy para realizar a preparação dos dados, além da TensorFlow e Scikit-learn com o intuito de treinar e avaliar os modelos. A maior parte dos gráficos foram gerados por meio das bibliotecas Seaborn e Matplotlib, contudo alguns foram contruídos por meio da plataforma Microsoft Excel.

Os códigos utilizados no projeto estão disponíveis em https://github.com/LeiteJu/TCC.

2.2 Dados

Os algoritmos de aprendizado de máquina utilizam dados para fazer previsões. Neste trabalho, optou-se por utilizar dados de 2003 até 2019 para treinamento e avaliação dos modelos, em virtude da disponibilidade dos dados de consumo mensal de cimento. Além disso, adotou-se granularidade de dados mensal e por estado com o intuito de aumentar a quantidade de entradas disponíveis para treinamento e avaliação dos modelos. Essa estratégia de aumentação de dados visa diminuir o risco de *underfitting*, que acontece quando o modelo não é capaz de aprender com os dados e resulta em altos erros nas etapas de treinamento e teste, de acordo com Goodfellow *et al.* (2016).

2.2.1 Fontes

O modelo utiliza dados econômicos, sociais e da construção civil para estimar a demanda por cimento. Na tabela 2.1, são apresentados os dados utilizados, juntamente com a fonte, a granularidade e o período em que estavam disponíveis.

Dado	Fonte	Período disponível	Granularidade
PIB a preços constantes	IBGE[a]	1983 até 2019	anual por estado
PIB a preços de mercado	IBGE[a]	1985 ate 2019	anual por estado
PIB *per capita*	IBGE[a]	1985 até 2019	anual por estado
PIB da construção civil	IBGE[a]	1985 até 2019	anual por estado
Desemprego	IBGE[a]	1991 até 2022	irregular [b]
IPCA	IBGE[c]	1981 até 2021	mensal para o Brasil
INCC	FGV[d]	1980 ate 2021	mensal para o Brasil
IGP	FGV[a]	1944 até 2021	mensal para o Brasil
Taxa Selic	IBGE[e]	1986 até 2022	mensal para o Brasil
NFSP	BACEN[a]	1991 até 2022	mensal para o Brasil
Estoque líquido de capital fixo	IPEA[a]	1947 ate 2019	anual para o Brasil
População	IBGE[f]	1991 até 2021	anual por estado
IDH	IBGE[a]	1991 ate 2017	irregular[g]
Produção mensal de cimento	SNIC[h]	2003 até 2022	mensal por estado
Valor médio do cimento[i]	SNIC[h]	1947 ate 2019	anual para o Brasil
Consumo de cimento em ton.	SNIC[h]	2003 até 2019	mensal por estado

Tabela 2.1: *Indicadores utilizados no trabalho*

[a] Dado retirado do portal do Ipeadata em http://www.ipeadata.gov.br/Default.aspx

[b] Havia dados de 1992 até 2014 com granularidade anual e por estado. A partir de 2012, foram disponibilizados dados mensais a nível de Brasil por conta da Pesquisa Nacional por Amostra de Domicílios (PNAD) Contínua realizada pelo IBGE. Neste trabalho, utilizou-se os dados anuais até 2012 e, após 2012, os dados provenientes da PNAD Contínua.

[c] Dado retirado do IBGE em https://sidra.ibge.gov.br/tabela/1737

[d] Dado obtido a partir do portal da FGV em https://www.debit.com.br/tabelas/tabela-completa-pdf.php?indice=incc

[e] Dado obtido em https://www.debit.com.br/tabelas/tabela-completa.php?indice=selic

[f] Dado obtido do portal Base dos Dados em https://basedosdados.org/dataset/br-ibge-populacao

[g] Os indicadores de IDH (Renda, Longevidade e Educação) estão disponíveis a nível de estados da União em anos de censo do IBGE (1990, 2000, 2010). Há dados, também, de 2014 a 2017 por conta da PNAD Contínua.

[h] Dados retirados do portal http://www.cbicdados.com.br/menu/materiais-de-construcao/cimento

[i] Evolução do valor médio/mediano do cimento Portland 32 em US$/Tonelada

Na tabela, são utilizadas siglas para melhorar a legibilidade, uma descrição das abreviações utilizadas no trabalho pode ser encontrada no início do documento, em Lista de Abreviaturas.

2.2.2 Preparação dos dados

Com o intuito de direcionar a estratégia de preparação de dados, foi realizada uma análise exploratória dos dados de entrada e da variável resposta. Foi identificada nessa análise, uma alta taxa de variação nos atributos, dessa forma, está presente nos dados um grande número de *outliers*. De acordo com Hoaglin (2013) e Tukey (1977), *outliers* são observações discrepantes do restante dos dados que podem interferir no processo de previsão.

Um exemplo de atributo com alta variação é o PIB da contrução civil, que apresenta desvio padrão maior que o valor médio dessa variável. Para ilustrar a distribuição dos pontos de dados desse indicador, utiliza-se um gráfico *boxplot* na figura 2.1.

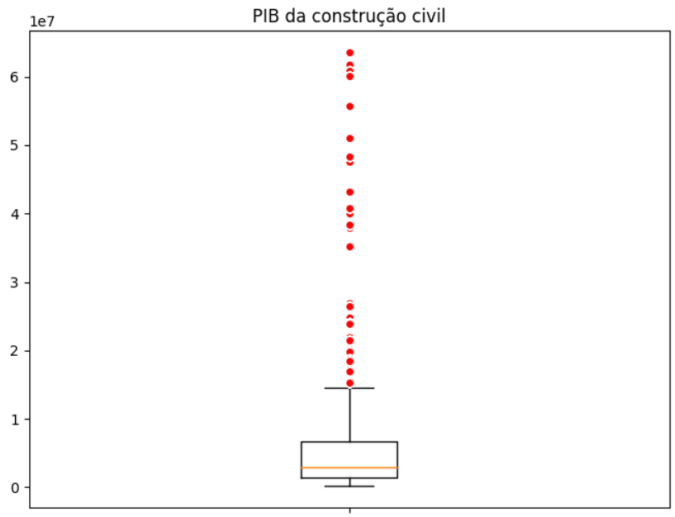


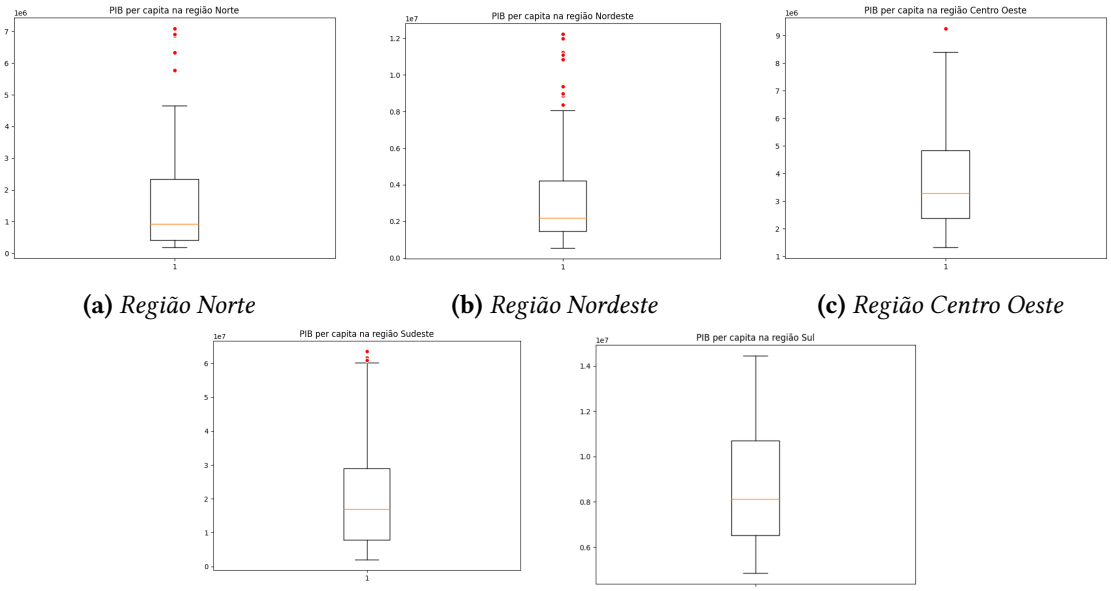
Figura 2.1: *Gráfico boxplot do PIB da construção civil*

Segundo Williamson *et al.* (1989), o *boxplot* é uma técnica estatística utilizada para identificar visualmente padrões nos dados. Na figura acima, a linha em laranja corresponde à mediana [1] dos dados, os limite inferior e o superior do retângulo representam o primeiro e terceiro quartis[2], respectivamente. As observações fora do intervalo dos limites superior e inferior, representadas com círculos vermelhos na figura, são *outliers*. Pode-se observar, então, a alta incidência de dados discrepantes nesse indicador.

Parte do alto volume de *outliers* nos atributos se deve às diferenças econômicas, geográficas e sociais entre os estados do Brasil. A saber, o valor médio do PIB da construção civil no estado de São Paulo é de 48, 96 milhões, enquanto em Santa Catarina é de 7, 1 milhões e em Roraima, de 0, 4 milhões. Aliás, ao analisar o PIB da construção civil nas regiões do país, há uma significativa redução no número de dados discrepantes, como pode-se observar na figura abaixo.

[1] Mediana é valor que fica no meio quando os dados estão ordenados ou a média dos dois valores centrais se o número de pontos de dados for par. (Esselman, 2022)

[2] O primeiro quartil marca a mediana relativa aos valores superiores à mediana destacada na imagem. O terceiro quartil, de maneira análoga, assinala a mediana dos valores inferiores à mediana destacada. Assim, entre o primeiro e terceiro quartis está contida metade dos dados.



(a) *Região Norte* **(b)** *Região Nordeste* **(c)** *Região Centro Oeste*

(d) *Região Sudeste* **(e)** *Região Sul*

Figura 2.2: *Comparação dos gráficos boxplot entre as regiões*

Além disso, analisou-se, a correlação entre as variáveis de entrada com o auxílio de uma matriz de correlação. Foi identificada alta correlação entre os indicadores do PIB do estado, o PIB da construção civil e a população. Além disso, há correlação entre os três indicadores de IDH (Longevidade, Saúde e Renda) e entre o preço do saco de cimento e o preço do kilograma, como pode-se validar na figura 2.3.

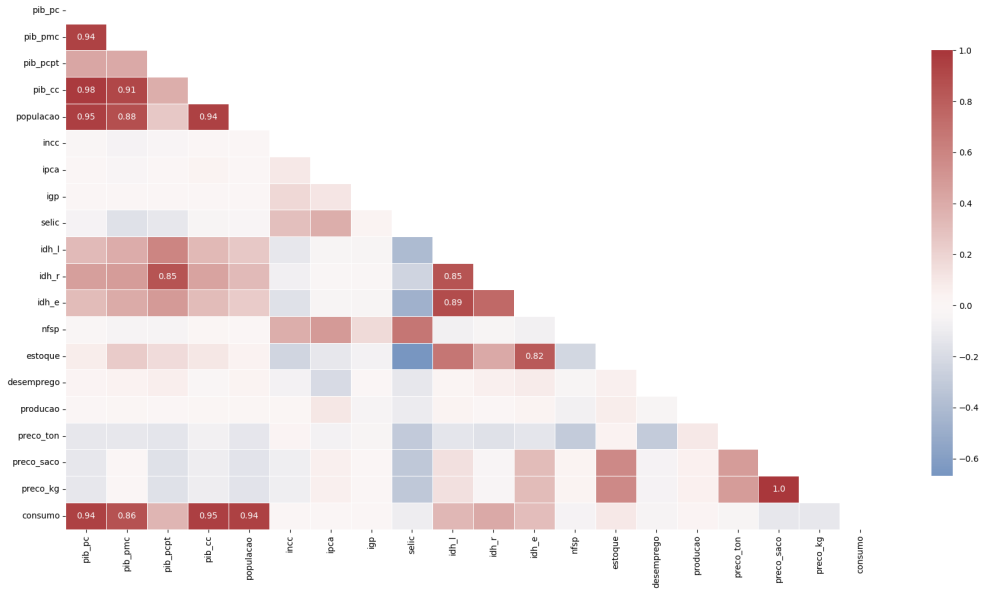


Figura 2.3: *Matriz de correlação*

Foram adotadas estratégias para garantir dados na granularidade mensal e por estado. Caso os indicadores apresentassem granularidade anual, o valor de cada medição foi

dividido por 12 de modo a obter a média mensal utilizada em todos os meses do ano correspondente. Caso a granularidade fosse a nível de Brasil, o valor apresentado foi repetido para todos os estados no mês e ano correspondente. As exceções foram os indicadores de IDH por apresentarem valores apenas em anos específicos, conforme detalhado em 2.1. Nesse caso, quando não havia dados para um determinado ano foi repetido o valor da última medição.

Também foi necessário lidar com dados faltantes, o método utilizado foi repetir o último valor disponível nos dados de entrada para preencher a ocorrência faltante. Contudo, para a produção mensal de cimento e os indicadores de evolução do preço do cimento foi utilizado um valor não presente no intervalo dos dados de entrada (-1) para marcar as ocorrências faltantes como nulo, uma vez que essas variáveis não apresentavam valores mais antigos.

Além disso, garantiu-se que a previsão utiliza dados dos meses anteriores e não do mês alvo da previsão, uma vez que o objetivo do projeto é prever a demanda por cimento em um mês a partir dos dados disponíveis no mês anterior. Por isso, nos dados com granularidade anual, foi realizado um um deslocamento de modo a associar as entradas de um ano ao consumo no ano seguinte. Nos dados mensais, analogamente, deslocou-se as entradas para associá-las ao consumo no mês seguinte.

Por fim, o estado correspondente à medição foi usado como dado de entrada. Como os modelos de inteligência artificial aceitam apenas caracteres numéricos, utilizou-se o método de codificação *one hot* para criar 27 colunas, uma para cada estado, nas quais o valor é um quando a linha possui dados daquele estado e é zero caso contrário.

2.2.3 Pré-processamento de dados

Foram utilizadas técnicas de pré-processamento de dados com o intuito de garantir que as variáveis de entrada estivessem na mesma escala. Esse processo tem a finalidade de permitir a comparação entre variáveis com diferentes unidades e melhorar o funcionamento dos algorimos de aprendizado de máquina, uma vez que se as *features* estiverem em escalas diferentes alguns pesos podem ser atualizados mais rápidos que outros, segundo Raschka (2014). Neste trabalho, foram testadas: normalização, *min-max scaler* e *power transformer*.

Normalização

A normalização garante que as variáveis de entrada estejam em uma escala com as propriedades de uma distribuição normal: média (μ) igual a zero e variância (σ) igual a um. Dessa forma, a operação realizada para aplicar o processo em uma entrada x é

$$x_{norm} = \frac{x - \mu}{\sigma} \tag{2.1}$$

Min-Max scaler

De acordo com Raschka (2014), o *min-max scaler* é uma abordagem alterantiva à normalização e transforma os dados de modo que a escala tenha um intervalo definido,

em geral, entre 0 e 1. A operação aplicada em uma entrada x para é

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{2.2}$$

Sendo x_{norm} a versão normalizada da entrada x, e x_{min} e x_{max} os valor mínimo e máximo assumido por x.

Power transformer

O *power transformer* é outra alternativa para aproximar os dados de uma distribuição Gaussiana. Esse método visa estabilizar a variância e diminuir a assimetria dos dados, por meio da aplicação da transformação de Yeo e Johnson (2000) nas entradas.

2.3 Avaliação de performance

Para comparar a eficiência dos modelos mede-se os erros de cada previsão, ou seja, a distância entre o valor previsto pelo algoritmo e o valor do dado real. Neste trabalho, utilizou-se as seguintes métricas estatísticas para mensurar o desempenho: *mean absolute error* (MAE), *root mean square error* (RMSE) e *mean absolute percentage error* (MAPE). Além disso, foi utilizada a variação percentual (Δ) para avaliar se o modelo tende a subestimar ou superestimar o valor previsto.

2.3.1 *Mean absolute error* (MAE)

De acordo com Hewamalage *et al.* (2022), a *mean absolute error* (MAE) mede o erro absoluto de cada previsão. Seja $\hat{y}_i$ o valor previsto pelo modelo e y_i o valor real, a fórmula da MAE é dada por:

$$MAE = \frac{\sum_{i=1}^{n} |\hat{y}_i - y_i|}{n} \tag{2.3}$$

Essa métrica não leva em consideração a proporção do erro em relação ao valor real, apenas o valor absoluto da diferença, dessa forma não expressa a ordem de grandeza do erro em relação ao valor real.

2.3.2 *Root mean squared error* (RMSE)

A RMSE, sigla para *root mean squared error* é uma métrica de erro semelhante à MAE. Sendo $\hat{y}_i$ o valor previsto pelo modelo para um *output* y_i, a fórmula da RMSE é dada por:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{n}} \tag{2.4}$$

A RMSE é influenciada por erros mais significativos, dessa forma, é mais sensível a *outliers* que a MAE. (Hewamalage *et al.*, 2022)

2.3.3 *Mean absolute percentage error* (MAPE)

Foi utilizada também a *mean absolute percentage error*, para mensurar a magnitude do erro em relação ao tamanho das medições.

A MAPE, *mean absolute percentage error*, calcula a proporção do erro em relação ao valor que o modelo tentou prever. Sendo $\hat{y}_i$ o valor previsto pelo modelo para um *output* y_i, a fórmula da MAPE é dada por:

$$MAPE = \sum_{t=1}^{n} \left| \frac{y_t - \hat{y}_t}{y_t} \right| \tag{2.5}$$

A MAPE proporciona uma visão do tamanho do erro em relação ao valor que se deseja prever.

2.3.4 Variação percentual

A variação percentual, Δ, é utilizado para mensurar se o modelo apresenta tendência de subestimar ou superestimar a variável. Sendo $\hat{y}_i$ o valor previsto pelo modelo para um *output* y_i, a fórmula da variação percentual Δ é dada por:

$$\Delta = \frac{\hat{y}_i - y_i}{y_i} \tag{2.6}$$

2.4 Avaliação do desempenho do modelo

Neste trabalho, separou-se a massa de dados em conjunto de treino com 85% das entradas e teste com 15% do conjunto. Esse método tem o objetivo de avaliar a capacidade de generalização do modelo. Por medir o desempenho em dados desconhecidos pelo modelo, também permite avaliar a ocorrência de *overfitting*[3]. Não foi utilizado embaralhamento de dados, então os modelos foram treinados com dados de janeiro de 2003 até junho de 2017 e testados com dados de julho de 2017 até dezembro de 2019.

[3] O *overfitting* ocorre quando um modelo tem baixa capacidade de generalização, então, apesar de apresentar baixa taxa de erro nos dados de treino, apresenta desempenho ruim em dados não conhecidos.

Capítulo 3

Experimentos

Para cada um dos modelos selecionados foram realizados diferentes experimentos alterando parâmetros com o objetivo de obter um melhor desempenho. Neste seção estão descritos os experimentos realizados de acordo com o modelo.

3.1 Regressao linear

A regressão linear é utilizada neste estudo como base para comparar o desempenho dos outros modelos por ser um modelo mais simples de aprendizado de máquina. Foram testados métodos de normalização e transformação de dados, além da remoção de variáveis de alta correlação conforme descrito na tabela 3.1.

Experimento	Transformação nos dados	Período de dados	Remoção de variáveis
A	nenhum	de 2003 até 2019	não
B	*Standard scaler*	de 2003 até 2019	não
C	*MinMax scaler*	de 2003 até 2019	não
D	*Power Transformer*	de 2003 até 2019	não
E	nenhum	de 2003 até 2019	sim[a]

Tabela 3.1: *Experimentos realizados na regressão linear*

[a] As variáveis removidas foram: PIB *per capita*, INCC, IGP, taxa Selic, IDH Educação e IDH Longevidade, NFSP, preço do saco de cimento e preço da tonelada de cimento

O desempenho do modelo em cada um dos experimentos encontra-se na tabela 3.2:

Experimento	MAE	RMSE	MAPE
A	35148.6	55391.1	0.6
B	35148.6	55391.1	0.6
C	35148.6	55391.1	0.6
D	49073.6	67446.7	1.05
E	**26435.6**	**38970.6**	**0.32**

Tabela 3.2: *Desempenho dos modelos de regressão linear*

Observa-se que não houve alteração na performance ao normalizar os dados com *standard scaler* ou *minmax scaler*, ao utilizar *power transformer*, contudo, houve piora no desempenho. Ressalta-se a grande melhoria nas métricas ao remover variáveis de alta correlação tanto nas métricas de erro quantitativo, como o MAE e RMSE, quanto proporcionalmente ao atingir 32% de erro percentual médio.

Foi realizado o teste de remover variáveis de alta correlação, pois segundo Jiang *et al.*, 2022, esse tipo de *feature* pode prejudicar no modelo por aumentar a variabilidade dos coeficientes em relação à amostra, dessa forma, os coeficientes calculados pelo modelo podem ter um alto grau de variação de uma amostra para outra.

Na seção 4, estão apresentados os resultados e previsões do modelo de regressão linear com melhor desempenho dentre os testados no modelo, o experimento E, sem normalização de dados e removendo atributos com alta correlação.

3.2 Redes *feed forward*

As redes neurais *feed forward* podem ser construídas com várias arquiteturas e configurações. Neste estudo, testou-se alterar a quantidade de camadas da rede, o número neurônios em cada camada, a função de ativação utilizada, além da quantidade de *epoch*[1] utilizada no treinamento.

Os valores testados para cada um dos parâmetros são dados na tabela 3.3:

Parâmetro	Valores
Número de camadas	1,2,3 e 4
Número de neurônios	32,64,128 e 256
Função de ativação	ReLU e *swish*
Epochs	50, 100 e 150
Pré-processamento de dados	normalização, *min-max scaler* e *power transformer*

Tabela 3.3: *Parâmetros testados nas redes feed forward*

Cada experimento corresponde a uma combinação dos parâmetros da tabela 3.3, por exemplo, um modelo de uma camada de 32 neurônios, função de ativação ReLU e 50 *epochs*.

[1] Uma *epoch* é uma passada pelos dados de entrada (Patterson e Gibson, s.d.)

Os resultados dos experimentos foram separados de acordo com o número de camadas na rede para melhor visualização. O desempenho dos melhores modelos com melhor em relação a outros com o mesmo número de camadas é dado na tabela 3.4.

Número de camadas	MAE	RMSE	MAPE
1	51665.4	87660.1	0.49
2	32407.0	55355.1	0.35
3	**25520.8**	**42929.2**	**0.25**
4	24458.9	38424.9	0.30

Tabela 3.4: *Experimentos com melhor desempenho segundo o número de camadas das redes feed forward*

Interessante notar que ao priorizar os erros absolutos (MAE e RMSE), o modelo com quatro camadas apresentou um desempenho ligeiramente melhor que o de três camadas, contudo, possui maior erro percentual. Assim, por mais que os erros absolutos sejam menores, ao levar em consideração o valor que o modelo pretende prever, os erros desse modelo são maiores que os gerados pelo modelo de três camadas.

Neste trabalho, priorizou-se modelos com menor erro percentual (MAPE), então o modelo com melhor desempenho entre as redes *multilayer feed forward* foi o de três camadas. Esse modelo utiliza a função de ativação *swish*, o método de normalizaçõa para pré-processar os dados e foi treinado com 100 *epochs*, além disso a primeira camada oculta possui 256 neurônios, a segunda, 64 e a terceira, 32.

3.3 Redes recorrentes

Neste trabalho, foram utilizadas redes recorrentes LSTM e GRU, para ambas as redes, realizaram-se experimentos alterando os mesmos parâmetros, a saber, número de camadas, número de neurônios, função de ativação, número de *epochs* e método de pré-processamento de dados. Na tabela 3.5 encontra-se a relação dos valores testados para cada parâmetro.

Parâmetro	Valores
Número de camadas	1 e 2
Número de neurônios	32,64,128 e 256
Função de ativação	tangente hiperbólica[a], ReLU e *swish*
Epochs	50, 100 e 150
Pré-processamento de dados	normalização, *min-max scaler* e *power transformer*
time step[b]	3 e 5

Tabela 3.5: *Parâmetros testados nas redes recorrentes*

[a] Por padrão, as redes LSTM e GRU utilizam a tangente hiperbólica (*tanh*) como função de ativação.

[b] O *time step* é o tamanho da sequência que a rede recebe como entrada.

Foram realizados os mesmos experimentos para ambas as redes, assim como para as redes neurais tradicionais, os resultados foram agrupados de acordo com o número de camadas da rede. O desempenho dos modelos com melhores resultados estão na tabela 3.6.

Número de camadas	Rede	MAE	RMSE	MAPE
1	**LSTM**	**22694**	**43918**	**0.19**
1	GRU	26751	52281	0.24
2	LSTM	25002	47565	0.19
2	GRU	24992	50602	0.20

Tabela 3.6: *Experimentos com melhor desempenho segundo o número de camadas das redes recorrentes*

Observa-se que as redes LSTM obtiveram um melhor desempenho, em geral, que as redes GRU e que a rede LSTM mais simples, com uma camada, fez as previsões mais precisas. A rede recorrente com melhor desempenho, então, é uma rede LSTM de uma camada com 32 neurônios, que utiliza a ReLU como função de ativação, com 150 *epochs* e normalização como método de processamento de dados.

3.4 Redes bidirecionais

As redes bidirecionais utilizam duas redes recorrentes para realizar as previsões. Neste estudo, realizaram-se experimentos com redes LSTM e GRU, alterou-se também o número de camadas, a quantidade de neurônios nas camadas, a função de ativação utilizada nas redes, o método de pré-processamento de dados e a quantidade de *epochs*. A relação dos valores testados para cada parâmetro está na tabela 3.7.

Parâmetro	Valores
Número de camadas	1 e 2
Número de neurônios	32, 64, 128 e 256
Função de ativação	tangente hiperbólica[a], ReLU e *swish*
Epochs	50, 100 e 150
Pré-processamento de dados	normalização, *min-max scaler* e *power transformer*
time step[b]	3 e 5

Tabela 3.7: *Parâmetros testados nas redes bidirecionais*

[a] Por padrão, as redes LSTM e GRU utilizam a tangente hiperbólica (*tanh*) como função de ativação.

[b] O *time step* é o tamanho da sequência que a rede recebe como entrada.

O resultado dos modelos foi agrupado de acordo com o número de camadas da rede e com a categoria de rede recorrente escolhida (LSTM ou GRU). A relação dos modelos com melhor desempenho é dada na tabela 3.8.

Número de camadas	Rede	MAE	RMSE	MAPE
1	**LSTM**	**19185**	**33942**	**0.17**
1	GRU	19952	33979	0.18
2	LSTM	22964	38688	0.20
2	GRU	20993	35817	0.20

Tabela 3.8: *Experimentos com melhor desempenho das redes bidirecionais*

Observa-se que, o modelo com melhor desempenho das redes bidirecionais utiliza uma camada de rede LSTM de 2 neurônios para fazer as previsões, além disso, o experimento foi realizado com 150 *epochs* e com o método de pré-processamento de dados *min-max scaler*.

Capítulo 4

Resultados

É apresentada nessa seção uma comparação entre as previsões realizadas pelos modelos testados neste trabalho. Foram realizados experimentos com diferentes combinações de parâmetros, detalhadas no capítulo 3, nesta seção é utilizado o modelo que apresentou melhor desempenho de cada uma das classes de algoritmos testadas: regressão linear, redes *multilayer feed forward*, redes recorrentes e redes bidirecionais.

É possível comparar o desempenho dos modelos a partir da tabela 4.1.

Modelo	MAE	RMSE	MAPE
Regressão linear	26435	38970	0.32
Rede *multilayer feed forward*	25520.8	42929.2	0.25
Rede recorrente	22694	43918	0.19
Rede bidirecional	19185	33942	0.17

Tabela 4.1: *Desempenho dos modelos de regressão linear*

Na análise da tabela acima, nota-se que o desempenho nas previsões melhora à medida que se aumenta a robustez do modelo.

Observa-se uma significativa melhora de 13% da regressão linear para os modelos de redes recorrentes, reflexo da capacidade de reconhecer sequências e contexto. A melhora também se reflete nos erros absolutos calculados pela MAE, contudo a RMSE apresentou piora em relação à regressão linear, provavelmente por conta de *outliers*, uma vez que a RMSE é mais sensível a valores discrepantes.

As redes bidirecionais apresentaram melhor desempenho em comparação com os outros modelos, em virtude da arquitetura mais complexa com duas redes recorrentes. Há redução de 2% da MAPE em relação às redes recorrentes e de 15% em relação à regressão linear. Esse comportamento se reflete, também, nos indicadores de erro absoluto que são os menores entre as classes de algoritmos testados.

Será utilizado o estado de São Paulo para comparar as previsões realizadas pelo modelo e o valor real do consumo, a escolha se deu por se tratar do maior consumidor de cimento

no Brasil. A evolução do consumo mensal de cimento em São Paulo pode ser vista na figura 4.1.

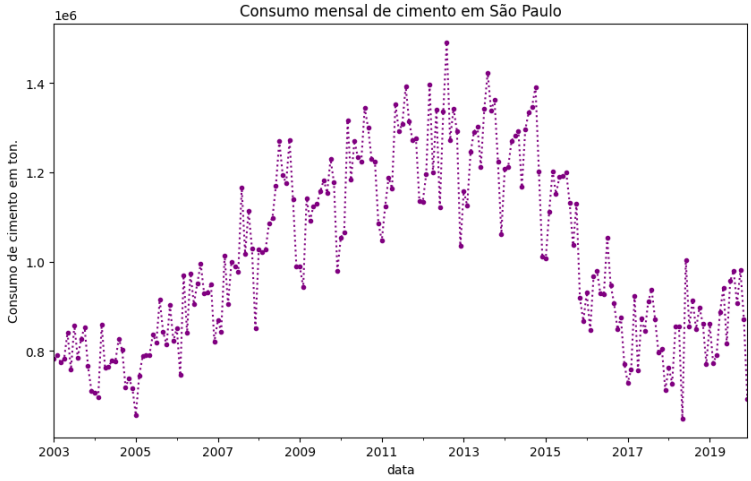


Figura 4.1: *Evolução do consumo mensal de cimento em São Paulo*

Na imagem acima, os círculos representam as medições mensais do consumo de cimento em São Paulo, observa-se que a demanda por cimento no estado vinha em tendência de queda desde 2013 e passou a apresentar alta de 2018 em diante.

4.1 Regressão linear

Capítulo 5

Conclusão

A partir da análise do desempenho dos três modelos estudados, observou-se o ganho ao utilizar as redes neurais recorrentes, uma vez que, por levarem em consideração informação histórica, apresentam melhor performance frente à regressão linear e às redes *feedforward*.

Referências

[DSA 2022] Data Science Academy. *Deep Learning Book*. disponível em: https://www.deeplearningbook.com.br. Acesso em: 17 Dezembro. 2022. 2022 (citado nas pgs. 4, 7–10, 14, 15).

[Alla 2021] Samhita Alla. *Advanced Recurrent Neural Networks: Bidirectional RNNs*. 2021. url: https://blog.paperspace.com/bidirectional-rnn-keras/ (acesso em 28/01/2023) (citado nas pgs. 15, 16).

[IBRAM 2021] IBRAM Mineração do Brasil. *Votorantim Cimentos inicia a operação de nova fábrica em Pecém com lançamento de produto no mercado cearense*. 2021. url: https://ibram.org.br/noticia/votorantim-cimentos-inicia-a-operacao-de-nova-fabrica-em-pecem-com-lancamento-de-produto-no-mercado-cearense/ (citado na pg. 1).

[Carvalho 2008] Maria Beatriz Maury de Carvalho. "Impactos e Conflitos da Produção de Cimento no Distrito Federal". Diss. de mestr. Universidade de Brasília, 2008 (citado na pg. 2).

[Cho *et al.* 2014] Kyunghyun Cho *et al. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. 2014. doi: 10.48550/ARXIV.1406.1078. url: https://arxiv.org/abs/1406.1078 (citado na pg. 15).

[Cimento Portland 2002] Associação Brasileira de Cimento Portland. "Guia básico de utilização do cimento portland". 2002 (citado na pg. 1).

[Desai 2019] Rashi Desai. *Top 10 Python Libraries for Data Science*. 2019. url: https://towardsdatascience.com/top-10-python-libraries-for-data-science-cd82294ec266 (acesso em 28/01/2023).

[Esselman 2022] Amy Esselman. "What is a boxplot?" 2022. url: https://www.storytellingwithdata.com/blog/what-is-a-boxplot (citado na pg. 19).

[Goodfellow *et al.* 2016] Ian Goodfellow, Yoshua Bengio e Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016 (citado nas pgs. 4, 5, 17).

[Hewamalage *et al.* 2022] Hansika Hewamalage, Klaus Ackermann e Christoph Bergmeir. *Forecast Evaluation for Data Scientists: Common Pitfalls and Best Practices*. 2022. doi: 10.48550/ARXIV.2203.10716. url: %5Curl%7Bhttps://arxiv.org/abs/2203.10716%7D (citado nas pgs. 22, 23).

[Hoaglin 2013] David C. Hoaglin. "Volume 16: how to detect and handle outliers". Em: 2013 (citado na pg. 19).

[Hochreiter e Schmidhuber 1997] Sepp Hochreiter e Jürgen Schmidhuber. "Long short-term memory". Em: *Neural computation* 9 (dez. de 1997), pgs. 1735–80. doi: 10.1162/neco.1997.9.8.1735 (citado na pg. 13).

[Hyndman e Athanasopoulos 2021] Rob J Hyndman e George Athanasopoulos. *Forecasting: principles and practice*. 3ª ed. OTexts.com/fpp3. Accessed on 17 December 2022. OTexts, 2021 (citado nas pgs. 6, 7).

[Jiang *et al.* 2022] Hao-sheng Jiang, Jia-li Chen e Chong-qi Zhang. *Constructing K−optimal designs for different Scheffé models*. 2022. doi: 10.48550/ARXIV.2210.07922. url: https://arxiv.org/abs/2210.07922 (citado na pg. 26).

[Julio da Motta Singer 2022] Pedro Alberto Morettin e Julio da Motta Singer. *Estatística e ciência de dados*. 2022.

[Lima Peressim 2021] Felipe de Lima Peressim. "Previsão da resistência à compressão do cimento". Diss. de mestr. Instituto de Matemática e Estatístican da Universidade de São Paulo, 2021.

[McCarthy 2007] John McCarthy. "What is artificial intelligence?" Em: (nov. de 2007). url: http://www-formal.stanford.edu/jmc/whatisai.pdf (citado na pg. 4).

[Nicholas Lacascio 2017] Nikhil Buduma e Nicholas Lacascio. *Fundamentals of Deep Learning*. O'Reilly Media, Inc, 2017 (citado na pg. 8).

[Nielsen 2015] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015 (citado nas pgs. 10, 12).

[Normas Técnicas 2018] Associação Brasileira de Normas Técnicas. "Cimento portland — requisitos". Em: 2018 (citado na pg. 1).

[Patterson e Gibson s.d.] Josh Patterson e Adam Gibson. *Deep Learning: A practitioner's approach* (citado nas pgs. 5, 8–11, 13, 26).

[Poole *et al.* 1998] David Poole, Alan Mackworth e Randy Goebel. *Computational Intelligence: A Logical Approach*. Oxford University Press, 1998.

[Ramachandran *et al.* 2017] Prajit Ramachandran, Barret Zoph e Quoc V. Le. *Searching for Activation Functions*. 2017. doi: 10.48550/ARXIV.1710.05941. url: https://arxiv.org/abs/1710.05941 (citado nas pgs. 11, 12).

[S. Raschka 2014] Sebastian Raschka. “About feature scaling and normalization - and the effect of standardization for machine learning algorithms”. 2014. URL: https://sebastianraschka.com/Articles/2014_about_feature_scaling.html.

[Raschka 2014] Sebastian Raschka. *About Feature Scaling and Normalization – and the effect of standardization for machine learning algorithms)*. Jul. de 2014. URL: https://sebastianraschka.com/Articles/2014_about_feature_scaling.html (acesso em 27/01/2023) (citado na pg. 21).

[Russell e Norvig s.d.] Stuart J. Russell e Peter Norvig. *Artificial Intelligence: A Modern Approach, third edition* (citado na pg. 3).

[Saxena 2021] Shipra Saxena. *Introduction to Gated Recurrent Unit (GRU)*. 2021. URL: https://www.analyticsvidhya.com/blog/2021/03/introduction-to-gated-recurrent-unit-gru/ (acesso em 27/01/2023).

[Schuster e Paliwal 1997] M. Schuster e K.K. Paliwal. “Bidirectional recurrent neural networks”. Em: *IEEE Transactions on Signal Processing* 45.11 (1997), pgs. 2673–2681. DOI: 10.1109/78.650093 (citado na pg. 15).

[Silva Barboza 2016] Lucas da Silva Barboza. “Estudo sobre o impacto da redução do consumo de cimento no comportamento mecânico do concreto autoadensável”. Diss. de mestr. Universidade Federal de São Carlos, 2016.

[Siqueira 2021] Daniel Siqueira. *Melhorando a análise com o Boxplot*. 2021. URL: https://www.alura.com.br/artigos/melhorando-a-analise-com-o-boxplot (acesso em 29/01/2023).

[SNIC 2021] SNIC. “Relatório nacional 2021 sindicato nacional da indústria do cimento”. 2021.

[Steck *et al.* 2021] Harald Steck *et al.* “Deep learning for recommender systems: a netflix case study”. Em: *AI Magazine* 42.3 (nov. de 2021), pgs. 7–18. DOI: 10.1609/aimag.v42i3.18140. URL: https://ojs.aaai.org/index.php/aimagazine/article/view/18140 (citado na pg. 3).

[Tukey 1977] John W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977 (citado na pg. 19).

[Turing 1950] A. M. Turing. “Computing machinery and intelligence”. Em: *Mind* 59.236 (1950) (citado na pg. 4).

[Vasconcelos 2022] Ieda Vasconcelos. *Informativo Econômico PIB*. 2022. URL: https://cbic.org.br/wp-content/uploads/2022/03/informativo-economico-pib-04-marco-2022.pdf (citado na pg. 1).

[Walter Pitts 1943] Warren McCulloch e Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". Em: *Bulletin of Mathematical Biology* 5 (1943) (citado na pg. 8).

[Wang 2019] Pei Wang. "On defining artificial intelligence". Em: *Journal of Artificial General Intelligence* 10.2 (2019), pgs. 1–37 (citado na pg. 4).

[Williamson *et al.* 1989] D Williamson, RA Parker e Juliette Kendrick. "The box plot: a simple visual method to interpret data". Em: *Annals of internal medicine* 110 (jul. de 1989), pgs. 916–21. doi: 10.1059/0003-4819-110-11-916 (citado na pg. 19).

[Yeo e Johnson 2000] In-Kwon Yeo e Richard A. Johnson. "A new family of power transformations to improve normality or symmetry". Em: *Biometrika* 87.4 (dez. de 2000), pgs. 954–959. issn: 0006-3444. doi: 10.1093/biomet/87.4.954. eprint: https://academic.oup.com/biomet/article-pdf/87/4/954/633221/870954.pdf. url: https://doi.org/10.1093/biomet/87.4.954 (citado na pg. 22).

[Zhang *et al.* 2021a] Aston Zhang, Zachary C. Lipton, Mu Li e Alexander J. Smola. "Dive into deep learning". Em: *arXiv preprint arXiv:2106.11342* (2021) (citado nas pgs. 4, 15).

[Zhang *et al.* 2021b] Aston Zhang, Zachary C. Lipton, Mu Li e Alexander J. Smola. "Dive into deep learning". Em: *arXiv preprint arXiv:2106.11342* (2021) (citado na pg. 10).