UNIVERSIDADE FEDERAL DE SÃO CARLOS — *CAMPUS* SOROCABA
CENTRO DE CIÊNCIAS E TECNOLOGIAS PARA A SUSTENTABILIDADE
DEPARTAMENTO DE FÍSICA, QUÍMICA E MATEMÁTICA

Victor Hugo Werneck Nordi

Previsões de preço de criptomoedas utilizando algoritmos de Aprendizado de Máquina

Sorocaba

Outubro, 2023

Victor Hugo Werneck Nordi

Previsões de preço de criptomoedas utilizando algoritmos de Aprendizado de Máquina

Trabalho de Conclusão de Curso apresentado ao curso de Licenciatura Plena em Matemática da Universidade Federal de São Carlos, *campus* Sorocaba, para obtenção do título de Licenciado em Matemática. Sorocaba, 10 de outubro de 2023.

Orientador: Prof. Dr. Renato Fernandes Cantão

Sorocaba

Outubro, 2023

Werneck Nordi, Victor Hugo

Previsões de preço de criptomoedas utilizando algoritmos de Aprendizado de Máquina / Victor Hugo Werneck Nordi -- 2023.
114f.

TCC (Graduação) - Universidade Federal de São Carlos, campus Sorocaba, Sorocaba
Orientador (a): Renato Fernandes Cantão
Banca Examinadora: Antonio Luis Venezuela, Deisemara Ferreira
Bibliografia

1. Criptomoedas. 2. Aprendizado de Máquina. I. Werneck Nordi, Victor Hugo. II. Título.

Ficha catalográfica desenvolvida pela Secretaria Geral de Informática (SIn)

DADOS FORNECIDOS PELO AUTOR

Bibliotecário responsável: Maria Aparecida de Lourdes Mariano - CRB/8 6979

FUNDAÇÃO UNIVERSIDADE FEDERAL DE SÃO CARLOS

COORDENAÇÃO DO CURSO DE LICENCIATURA EM MATEMÁTICA DE SOROCABA - CCML-So/CCTS
Rod. João Leme dos Santos km 110 - SP-264, s/n - Bairro Itinga, Sorocaba/SP, CEP 18052-780
Telefone: (15) 32298874 - http://www.ufscar.br

DP-TCC-FA nº 10/2023/CCML-So/CCTS

Graduação: Defesa Pública de Trabalho de Conclusão de Curso

Folha Aprovação (GDP-TCC-FA)

FOLHA DE APROVAÇÃO

VICTOR HUGO WERNECK NORDI

PREVISÕES DE PREÇO DE CRIPTOMOEDAS UTILIZANDO ALGORITMOS DE APRENDIZADO DE MÁQUINA

Trabalho de Conclusão de Curso

Universidade Federal de São Carlos – Campus Sorocaba

Sorocaba, 10 de outubro de 2023

ASSINATURAS E CIÊNCIAS

Cargo/Função	Nome Completo
Orientador	Prof. Dr. Renato Fernandes Cantão
Membro da Banca 1	Prof. Dr. Antonio Luis Venezuela
Membro da Banca 2	Profa. Dra. Deisemara Ferreira

Documento assinado eletronicamente por **Deisemara Ferreira**, **Docente**, em 11/10/2023, às 15:28, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Antonio Luis Venezuela**, **Docente**, em 11/10/2023, às 17:06, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Renato Fernandes Cantao**, **Docente**, em 16/10/2023, às 08:46, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Modelo de Documento: Grad: Defesa TCC: Folha Aprovação, versão de 02/Agosto/2019

Dedico este trabalho a todos que me apoiaram nessa jornada que foi minha graduação. A minha mãe, Lenita, que sempre esteve nesse processo e me apoiou, meus irmãos, Matheus e Miguel, que deixaram mais leves esses anos de dedicação.

Em especial, dedico também este trabalho ao meu colega de classe, Gustavo Bobato, que faleceu em 2021 devido ao COVID. Nossas memórias me ajudaram a chegar até aqui.

Agradecimentos

Agradeço este trabalho ao meu professor Renato Fernandes Cantão pela sua atenção e conselhos dedicados ao longo da minha graduação.

Agradeço também a minha família, que serviu de suporte em toda minha graduação.

Agradeço profundamente aos meus amigos Leonardo Flores, por, desde que me conheço por gente, estar comigo para me fazer rir e feliz, Lívia Tudela, pelo companheirismo e conexão, Isabelle Castro, pelo entusiasmo, Thayná Dourado, pela maturidade e ensinamentos, Lucas França, pelas risadas e broncas, João Pedro Delgado, pela sensatez, Isadora Tonscheis, por todo conhecimento e diversão, Amanda Teixeira, pelas risadas e confiança, Gabriel Melo, pela visão de mundo e por me tornar melhor, Jessica Ruy, pelos momentos incríveis, Pablo Fabiano, pelo carinho, Rafael Fontarce, pelo convívio e inspiração, Karina Tsuda, pela irmandade e (quase) me fazer gostar de chá, Julia Soares, pela lealdade e por ser um exemplo de pessoa, Ana Laura Marra, pelos conselhos e acolhimento e Vanessa Baker, pela energia e por ser uma irmã.

Agradeço especialmente, por fim, à Luciana Couto, que esteve comigo nos piores e melhores momentos da minha graduação, e tem uma parcela enorme na pessoa que me tornei. Uma pessoa que tem todo amor do mundo para dar e sou muito grato por tê-lo recebido.

“Não existe limão tão azedo que você não consiga fazer algo parecido com uma limonada.”
(This Is Us)

Resumo

As criptomoedas são moedas digitais descentralizadas que surgiram com a proposta de substituir os pagamentos tradicionais. Um aspecto que gera desconfiança em investidores é sua volatilidade característica. Com o propósito de analisar essa volatilidade, utiliza-se de inteligência computacional baseada em algoritmos de *Aprendizado de Máquina* com o intuito de prever valores dessas criptomoedas. Nesse sentido, neste trabalho propõem-se analisar a aplicação dos algoritmos Máquina de Vetores de Suporte (do Inglês, *Support Vector Machine*) e Memória de Curto Prazo Longo (do Inglês, *Long Short-Term memory*) na previsão de valores de algumas criptomoedas selecionadas, prevendo seus preços no futuro. Toda a implementação será feita na linguagem de programação *Python* e os resultados serão avaliados com as métricas de desempenho: erro quadrático médio, erro absoluto médio, erro máximo e R-quadrado (R^2). Espera-se, assim, avaliar a possibilidade de predições precisas — dentro das métricas de avaliação — ou se a alta volatilidade e aleatoriedade invalidam o uso desses algoritmos.

Palavras-chave: Criptomoedas. Python. Aprendizado de Máquina. Redes Neurais Recorrentes. *Support Vector Machines*. *Long Short-Term memory*.

Abstract

Cryptourrencies are decentralized digital currencies created with the intent of replacing traditional forms of payment. Their high volatility is one of the aspects that causes distrust among investors. Artificial Intelligence based on *Machine Learning* algorithms can be an invaluable tool to analyse this characteristic, helping to predict prices. In this sense, this project's goal is to use algorithms implemented in *Python* — *Support Vector Machine* and *Long Short-Term memory* — alongside convenient performance metrics such as mean squared error, mean absolute error, maximum error and R^2 to assess their price prediction capabilities in face of such randomness.

Keywords: Cryptocurrency. Python. Machine Learning. Recurrent Neural Networks. Support Vector Machines. Long Short-Term memory.

Lista de Figuras

Lista de Tabelas

Conteúdo

1 Introdução

1.1 Objetivos

O objetivo nesse trabalho é prever preços das criptomoedas *Bitcoin* (BTC), *Ethereum* (ETH), *Cardano* (ADA), e *Binance Coin* (BNB) utilizando os algoritmos de Aprendizado de Máquina: *Long Short-term memory* (LSTM) com redes neurais recorrentes e *Support Vector Machines* (SVM) implementados na linguagem de programação Python. Estes procedimentos serão estabelecidos no intuito de fornecer subsídios para tomada de decisão de compra e venda de criptomoedas. A qualidade dos ajustes dos algoritmos será avaliada através das métricas de desempenho usuais. Para que o objetivo principal seja alcançado, os seguintes objetivos específicos devem ser contemplados:

i. Coleta dos dados.

ii. Pré-processamento dos dados: limpeza, tratamento de *outliers* e imputação.

iii. Divisão dos dados em conjuntos de treinamento e teste.

iv. Estudo e validação dos principais algoritmos no contexto de movimentação de preços e tendência.

v. Definição das métricas adequadas para avaliação de desempenho dos algoritmos.

vi. Discussão dos resultados obtidos.

1.2 Motivação

A motivação desse trabalho veio do interesse tanto do orientador quanto do orientado em explorar algoritmos de Aprendizado de Máquina para previsão de preços com um conjunto de dados que representasse séries temporais, além das implementações matemáticas envolvidas nesse processo. Ademais, o orientado tem grande interesse no mercado das criptomoedas e trabalhar essas duas temáticas foi uma grande parte da motivação desse trabalho.

1.3 Criptomoedas

As criptomoedas surgiram como uma alternativa digital e descentralizada, isto é, sem um órgão mediador, como bancos, de garantir a integridade e segurança de transações financeiras. Desde 2010, as atividades no mercado de criptomoedas cresceram substancialmente, mostrando-se influente nesse ramo. Assim, a fim de compreender suas funcionalidades, torna-se necessário a

apresentação de conceitos como *blockchain* e *token*, fundamentais para sua existência (BUNJAKU; GJORGIEVA-TRAJKOVSKA; MITEVA-KACARSKI, 2017).

1.3.1 *Blockchain*

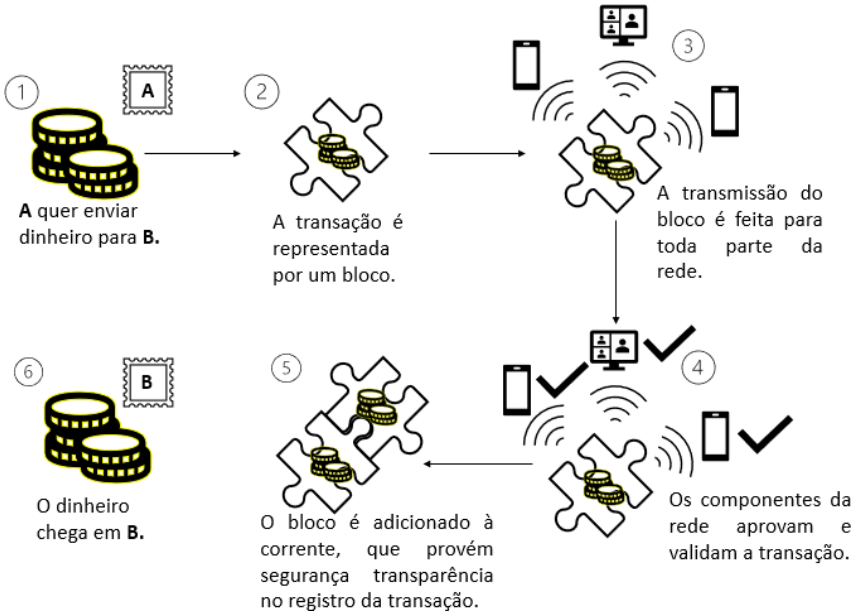
Blockchain é o nome dado a um sistema transparente de câmbio de dinheiro que vem transformando a forma como o mundo dos negócios se conduz (SARMAH, 2018). Essa tecnologia é, essencialmente, um conjunto de dados distribuídos que registra todas as transações e eventos digitais que foram executadas e compartilhadas entre seus participantes. Cada transação no livro-razão (registros) público é verificada pela maioria dos participantes nesse sistema. O objetivo principal da *blockchain* é conter registros verificáveis de cada uma das transações feitas dentro da rede. As vantagens dessa tecnologia ultrapassam os problemas regulatórios e desafios técnicos quando executada por um mediador, como bancos. Um ponto chave para o funcionamento da *blockchain* é o uso de *contratos inteligentes* (do Inglês, *smart contracts*), que são definidos como programas de computadores que executam termos de um contrato quando condições preestabelecidas são cumpridas por todas as partes envolvidas de forma transparente (CROSBY et al., 2016). A *blockchain* é responsável por armazenar em um banco de dados descentralizado, distribuído e imutável, registros em uma rede *peer-to-peer*, isto é, uma arquitetura de redes de computadores onde estes funcionam tanto como servidor (compartilhando serviços e dados) ou como cliente. Os protocolos dessa tecnologia são implementados em código aberto e livremente distribuídos, garantindo integridade e sem a necessidade de confiar em terceiros para execução das transações. Uma vez que todas as alterações na rede são acessíveis, essa pode ser considerada transparente, somada ao fato de que os registros não podem ser revertidos (MOREIRA, 2019).

A *blockchain* é dividida em subconjuntos de dados denominados blocos. Ao final de cada bloco, há um compilado que resume todo seu conteúdo, além do encaixe para o próximo bloco na cadeia. Se ocorrer alguma mudança nesse processo, o bloco não irá se encaixar à primeira linha do próximo e, havendo essa inconsistência, o bloco é descartado e substituído pelo original. Assim, os dados na *blockchain* são efetivamente imutáveis e rastreáveis. Os compilados final e inicial de cada bloco devem se encaixar (isto é, serem compatíveis), e esses são gerados através de uma função criptografada *hash*. Uma função *hash* é um algoritmo que mapeia diferentes estruturas de dados de tamanho variável a um conjunto único de endereços (HOROWITZ; SARTAJ, 1984; HÄRDLE; HARVEY; REULE, 2020).

A *blockchain* é diferenciada de um conjunto de dados ordinário e distribuído devido sua estrutura única, que conecta-se linearmente com as outras peças e blocos da rede. O nome vem de como as transações são armazenadas, que são em blocos conectados, formando uma cadeia. Quanto maior o número de transações, maior a cadeia de blocos, que se conectam através de uma rede de regras que devem ser validadas pelos seus participantes (GUPTA, 2020). Conforme citado anteriormente, a conexão se dá através de uma função criptografada e aleatória, sendo que qualquer tentativa de mudança no histórico da *blockchain* acarretará numa ruptura desse bloco e

num encaixe inválido, já que cada bloco se conecta unicamente com outro (HÄRDLE; HARVEY; REULE, 2020).

Figura 1 – Esquema ilustrativo de uma transação na *blockchain*.



Fonte: Adaptado de Crosby et al. (2015).

Na Figura 1, o processo de *A* enviar dinheiro para *B* pode ser esquematizado da seguinte forma: essa transação, é inicialmente colocada num bloco dentro da rede e compartilhada, a fim de que todos os participantes validem essa ação. Posteriormente, com a transação aprovada através da transmissão desse bloco aos servidores (clientes), o bloco é adicionado aos outros, tendo assim, seu registro transparente e imutável dentro da *blockchain*. Por fim, *B* recebe o dinheiro da transação.

1.3.2 *Tokens*

Token é uma aplicação dentro do ramo das *blockchains*, como as criptomoedas. Trata-se de uma representação digital de um bem, que pode ser transacionado dentro da *blockchain* entre seus participantes. A criação de um *token* é resultado da criação em cima da *blockchain*, programável e utilizado através de um projeto. A diferenciação entre criptomoedas e *tokens* é que, enquanto as primeiras têm sua própria cadeia de blocos ou livro-razão distribuído, o segundo é criado em alguma cadeia preexistente. O nome dado ao processo de converter o pertencimento de bens digitais, ou seja, *token*, é chamado de *tokenização* (do Inglês, *tokenization*). Assim, os *tokens* têm o benefício de alta liquidez[1], programabilidade e imutabilidade de proprietário (ANGELO; SALZER, 2020).

No âmbito de transações, os próprios *tokens* podem funcionar como moeda, sendo, muitas vezes, atribuídos de forma local em seu projeto, como os *dApps*, que são aplicativos que funcionam dentro da *blockchain*, ou seja, de forma decentralizada, com código aberto e que operam de forma

[1] Alta liquidez significa que o resgate é permitido a qualquer momento.

independente de autoridades centrais. Dentro destes, é possível a utilização de *tokens* para transações, com o intuito de comprar recursos oferecidos pelos contratos da rede. Por exemplo, o jogo de *NFTs*[2] *Axie Infinity* tem o seu próprio *token* de governança interno *AXS* e o seu *site*, que permite a interação dos usuários na plataforma para jogar e obter recompensas de forma descentralizada (TRUNG THANH NGUYEN, 2018). Nesse aspecto, os *tokens* são utilizados de forma contratual em seu projeto para construção do seu ecossistema e comunidade (ANGELO; SALZER, 2020).

Segundo Schueffel, Groeneweg e Rico (2019), escritores da "Enciclopédia Cripto" (do Inglês, *The Crypto Encyclopedia*),

> *Tokens* são criptoativos criados e contabilizados no sistema de tecnologia de livro-razão distribuída e representam um bem, direito de uso ou a unidade de valor emitida por uma organização. Eles são, tipicamente, emitidos através de uma oferta inicial ou venda privada[3]. Os *tokens* são construídos em cima da *blockchain* (SCHUEFFEL; GROENEWEG; RICO, 2019).

A tradução mais próxima de *token* para o português é "ficha", que representa um registro digital de uma certa quantidade inventariada na *blockchain*. Ao investidor, uma das vantagens de se obter determinado *token* é ter a possibilidade de ter acesso ao projeto antes do lançamento (BEZERRA; OLIVEIRA; SANTOS, 2020). Os investidores podem comprar os *tokens* de um projeto antes do seu lançamento, a um preço fixo e em grande quantidade, através de uma liquidez pré-determinada pelos desenvolvedores. Assim, quando o projeto for lançado para o público, o *token* sofrerá as variações do mercado alavancando o preço, gerando lucro aos primeiros investidores. Por exemplo, o pré-lançamento da *NFT Bored Ape*[4] custou 0,08 ETH (*token* da *Ethereum*), cerca de duzentos dólares à época. No início de 2022, o *token* mais barato da coleção custava cerca de 84 ETH, avaliado em cerca de duzentos mil dólares. A valorização acontece através do engajamento da comunidade (FINZER; HOLLANDER, 2017).

1.4 Criptomoedas neste trabalho

Criptomoedas são ativos digitais designados para mediar criptograficamente transações financeiras e verificá-las, de forma segura, sem o intermédio e centralização de alguma instituição física — um banco, por exemplo. Elas se diferenciam através do tipo de rede empregada, normalmente descentralizada (HÄRDLE; HARVEY; REULE, 2020). Segundo Abadi e Brunnermeier (2018), não há livro-razão, termo da contabilidade que diz respeito à forma de registro para acompanhamento de todas as transações e como essas são organizadas, que supere simultaneamente as qualidades ideais de qualquer sistema de registro de correção de preço, a saber a eficiência e a descentralização. Para suprir essa demanda, foi criada a *blockchain* (em tradução livre do Inglês, *cadeia de blocos*),

[2] *Token* não fungíveis, espécie de certificado digital como garantia da autenticidade de um arquivo, como personagens e objetos no jogo.

[3] Destinada a pessoas que estão numa *whitelist*, ou seja, selecionadas para poderem obter o *token* primeiramente, antes de disponibilidade pública.

[4] Maior coleção de artes digitalizadas na rede da *Ethereum*, que celebridades como Neymar, Eminem e Post Malone investiram.

que pode ser considerada como uma estrutura de conjunto de dados e registros compartilhada e distribuída, diferentemente do livro-razão do mercado financeiro, que é centralizado.

1.4.1 Bitcoin – BTC

As transações comerciais tradicionais apresentam algumas limitações claras, como o tempo de transação, a quantidade mínima, validação de terceiros (intermédio), possibilidades de fraudes e falta de transparência, entre outros. A fim de atenuar essas complexidades, em 2009, Satoshi Nakamoto (pseudônimo) lançou a primeira moeda digital: o *Bitcoin*. Pela primeira vez, é criada uma moeda sem uma entidade reguladora centralizada — ninguém controla a moeda, a não ser aqueles que a possuem. Diferentemente das moedas tradicionais, o *Bitcoin* não pode ser impresso, mas sim "minerado" e verificado nas transações computacionalmente, desvinculando a dependência de um intermédio bancário, por exemplo. Por outro lado, a rede de circulação do *Bitcoin* é aberta e todas as informações e transações são registradas com a *blockchain*, a fim de denotar transparência no processo. O objetivo de sua criação é romper a necessidade de intermediadores e ser utilizada, de forma mais prática, em escopo global (GUPTA, 2020). Na Figura 2 pode-se apreciar a variação de preço do *Bitcoin* entre o final de 2014 e o início de 2022.

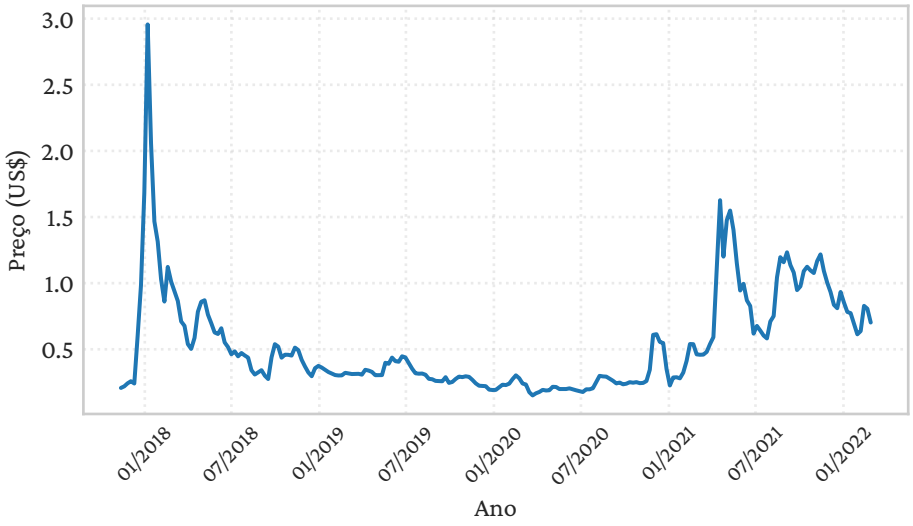
Figura 2 – Variação de preço do *Bitcoin* de Setembro de 2014 até Fevereiro de 2022.

Fonte: Autoral.

1.4.2 Ripple – XRP

O *Bitcoin* foi a primeira criptomoeda criada visando assegurar as transações e padronizar uma moeda digital de pagamento. Posteriormente, Chris Larsen e Jed McCaleb criaram um protocolo de pagamento distribuído, projetado para suportar *tokens* (como moedas físicas e ouro, por exemplo) para pagamentos seguros, instantâneos e com baixas taxas. Os criadores objetivam que bancos e outras instituições adotem o sistema e substituam sua forma nativa de transação pela *blockchain*.

Figura 3 – Variação de preço da XRP de Novembro de 2017 até Fevereiro de 2022.



Fonte: Autoral.

Esse protocolo chama-se *Ripple*, que possui a moeda nativa XRP (DAVID SCHWARTS; BRITTO, 2014).

Há investidores que contrapõem a utilização da XRP, uma vez que diverge do propósito de descentralização pela maior parte das criptomoedas. Em dezembro de 2020, a *Ripple* recebeu seu primeiro processo da Comissão de Valores Mobiliários dos Estados Unidos (SEC), que alegaram a responsabilidade de levantar uma receita de cerca de um bilhão de dólares em títulos não regulados. A SEC procura sua deslistagem desde então. Notícias relacionadas ao processo interferem diretamente no preço do criptoativo, mostrando-se instável desde então. Todavia, caso o processo seja ganho pela *Ripple*, a proposta pode se manter promissora (VIEIRA, 2021).

A *Ripple* foca em pagamentos comerciais em diversos países e interbancários, com o cruzamento de moedas e liquidação de fundos. O protocolo visa se distanciar da dependência direta do participantes com o banco e propõe uma abordagem mais dinâmica e flexível. A abordagem se dá através de um caminho que os fundos percorrem do vendedor, atrelado a uma moeda qualquer, ao receptor desses, através da participação de diversas instituições que oferecem serviços para essa moeda. No caso, a própria criptomoeda do protocolo, XRP, intenciona funcionar como uma forma padronizada de financiar as taxas de transações entre instituições financeiras (BANK, 2018). A Figura 3 mostra a evolução do preço da XRP do final de 2017 até o início de 2022.

Conforme a Figura 3, em janeiro de 2018 a XRP alcançou seu pico, no valor de 3.84 USD (dólares), valor nunca mais atingindo desde então.

1.4.3 Ether – ETH

Em julho de 2015, foi lançado, pelo russo Vitalik Buterin, o protocolo da *Ethereum*, com a criptomoeda *Ether*, considerada a segunda maior atualmente em relação à capitalização de mercado, que

diz respeito ao montante comercializado. O russo analisou as deficiências do *Bitcoin* e desenvolveu uma tecnologia superior, permitindo que desenvolvedores construam, na *blockchain* da *Ethereum*, aplicativos descentralizados, denominados *dApps*, *tokens*, *NFT*, mercado digital[5], entre outros. As transações de *Bitcoin* levam cerca de 60 vezes mais tempo para serem executadas do que as que acontecem na rede da *Ethereum* (FERREIRA, F. L., 2017).

A plataforma se envolve em contratos inteligentes (em Inglês, *smart contracts*), que têm como finalidade a validação contratual, estabelecida entre duas partes (em Inglês, *peer-to-peer*, em que um programa vincula seus *tokens* ao cumprimento de determinadas obrigações). Os desenvolvedores da *Ethereum* visam o lançamento da versão 2.0 do protocolo, tornando-se, segundo esses, o computador mundial, uma vez que, com essa rede, os servidores serão formados por uma cadeia de blocos e validados por voluntários ao redor do mundo, descentralizando a internet (**ethereu**; GUPTA, 2020). Na Figura 4, ilustra-se a variação do preço do *Ether* do final de 2017 ao início de 2022.

1.4.4 Binance Coin – BNB

Desde seu lançamento em 2018, a *Binance* é considerada a maior bolsa global de criptomoedas. Trata-se de uma plataforma de *exchange*, ou seja, transações para compra e venda de criptomoedas de forma segura e sem taxa de depósito com moedas tradicionais. Além do suporte para transações de mais de 250 criptomoedas na plataforma, há o *token* próprio, a *Binance Coin* (BNB), que executa funções transacionais no ecossistema, como a emissão de *tokens*, transferência de ativos e pagamento de taxas.

As taxas no protocolo da *Binance*, aplicadas nas transações entre diferentes carteiras e na assinatura dos contratos, são consideravelmente inferiores em relação aos supracitados. A plataforma também possibilita a conversão de valores pequenos de criptomoedas em seu *token*, o que a diferencia das outras plataformas, que exige uma quantidade mínima para transação (EXCHANGE, 2017). A Figura 5 ilustra a variação do preço da *Binance Coin* do final de 2017 até o início de 2022.

1.4.5 Cardano – ADA

Lançada em 2015, por Charles Hoskinson, com a empresa *Input Output Hong Kong* por trás do desenvolvimento do protocolo, a intenção da *Cardano* é criar uma *blockchain* com um desempenho superior ao da *Ethereum*, com transações mais rápidas e menores taxas. A *blockchain* da *Cardano* foi desenvolvida do zero, com a criptomoeda ADA como *token* base para realização de transações e a realização de contratos inteligentes (FERREIRA, R., 2018).

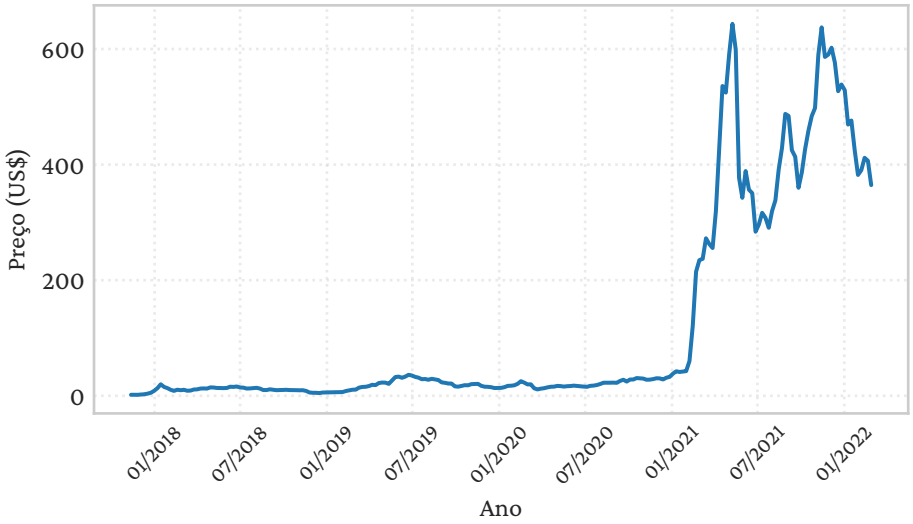
Uma das utilidades da rede da *Cardano* é voltada à comunidade científica, com a proposta de que periódicos, artigos e revistas sejam publicadas e registradas no ecossistema, assegurando ao detentor a publicação digital. A *blockchain* da *Cardano* apresenta duas camadas: a de liquidação e

[5] Do Inglês *marketplace*, funciona como uma loja de objetos e personagens digitais para compra e venda através de um determinado *token*.

Figura 4 – Variação de preço do ETH de Novembro de 2017 até Fevereiro de 2022.

Fonte: Autoral.

Figura 5 – Variação de preço da BNB de Novembro de 2017 até Fevereiro de 2022.



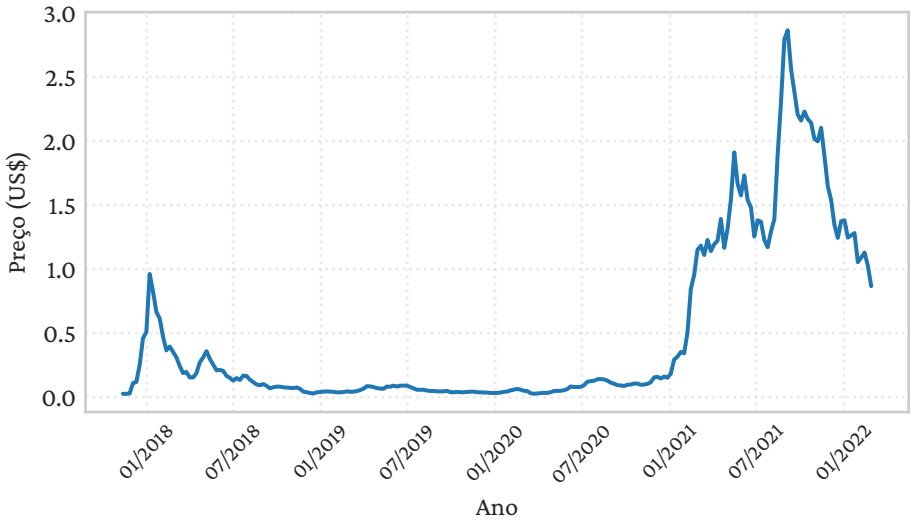
Fonte: Autoral.

a de computação. A primeira diz respeito à permissão dos usuários enviarem e receberem o *token* ADA entre carteiras, como a *Ethereum*. A segunda permite que os usuários participem e criem os contratos inteligentes dentro do ecossistema (FERREIRA, R., 2018). Na Figura 6, pode-se verificar a variação do preço da ADA do final de 2017 até o início de 2022.

1.5 Aprendizado de Máquina

Em 1959, Arthur Lee Samuel, pioneiro no ramo de jogos para computadores e inteligência artificial, citou pela primeira vez o termo *Aprendizado de Máquina* para prever movimentos no jogo

Figura 6 – Variação de preço da ADA de Novembro de 2017 até Fevereiro de 2022.



Fonte: Autoral.

de damas (SAMUEL, 1959). Nesse sentido, a utilização do Aprendizado de Máquina fora relacionada com o reconhecimento e classificação de padrões. Posteriormente, em 1981, foram utilizadas técnicas de aprendizado com redes neurais capazes de reconhecer 40 diferentes caracteres escritos (BLUM, 2000). Com essa mudança não somente na concepção do que é Aprendizado de Máquina, mas também nas suas potenciais aplicações, questiona-se se, com o passar do tempo, as máquinas poderão ser capazes de exercer as funcionalidades de capacidades humanas de forma geral.

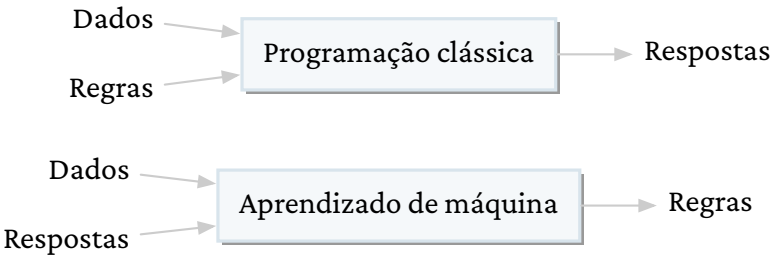
Segundo Lorena e Leon Ferreira de Carvalho (2007), o Aprendizado de Máquina faz uso da indução, princípio de inferência que possibilita a obtenção de conclusões genéricas a partir de entradas fornecidas ao algoritmo. Essa indução pode ocorrer basicamente de duas formas: supervisionada ou não supervisionada (mais sobre isso à frente).

O Aprendizado de Máquina é considerado um subcampo da *Inteligência Artificial*, uma vez que este envolve a capacidade do computador de realizar tarefas complexas através de algoritmos treinados para que a máquina realize alguma determinada ação sem ser programada de forma explícita para fazê-la. Assim sendo, o algoritmo procura "criar sentido" para dados pré-existentes, encontrando um padrão pela aproximação de uma função que será posteriormente usada para previsão a partir de dados novos. Alguns algoritmos são capazes de aprender com os próprios erros, trazendo uma sofisticação que os torna mais eficazes nas previsões (HURWITZ; KIRSCH, 2018). Em síntese, há dois pilares que definem o Aprendizado de Máquina: *classificar dados* sob modelos desenvolvidos e *fazer predições* baseadas nestas classificações (BLUM, 2000).

Numa definição mais voltada ao campo da engenharia, diz-se que um programa de computador aprende, em relação a uma experiência E e tarefa T, quando uma performance P aperfeiçoa E realizando T. Trata-se da habilidade do computador aprender e fazer sem ser explicitamente programado (GÉRON, 2019). Em Chollet (2018), difere-se a programação clássica do aprendizado de máquina de forma bastante sintética: a primeira recebe regras e dados e retorna respostas; a

segunda recebe os dados e as respostas e retorna as regras, conforme a Figura 7.

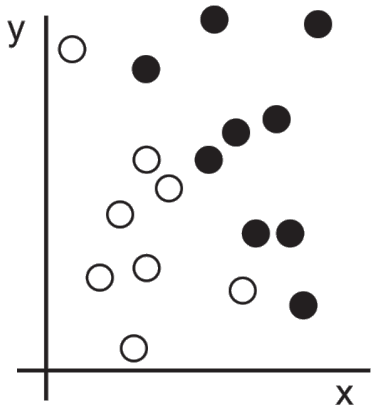
Figura 7 – Abordagem do Aprendizado de Máquina, segundo Chollet (2018). Na programação clássica, a partir de dados e regras, tem-se a intenção de obter uma resposta; no aprendizado de máquina são fornecidos dados e respostas, buscando-se determinar automaticamente as regras.



Fonte: Adaptado de Chollet (2018).

O Aprendizado de Máquina procura, essencialmente, uma forma de representação significativa dos dados a partir da qual seja possível fazer previsões. Essa representação pode ser realizada de diversas maneiras, como, por exemplo, tipagem de arquivos `txt` (texto simples) e `pdf` (*portable document format*). A representação em `txt` torna possível a edição do texto, enquanto o `pdf` se qualifica melhor para visualização. Encontrar a melhor representação possível para um determinado conjunto de dados é fundamental para os algoritmos de aprendizado, com o objetivo de classificar ou prever valores de outros dados definidos por esse conjunto (CHOLLET, 2018). Tomemos como exemplo os dados representados na Figura 8.

Figura 8 – Representação gráfica de uma amostra de dados. Os dados apresentam duas categorias de cor, "preto" e "branco", sem uma separação natural ou óbvia entre os dois subconjuntos.

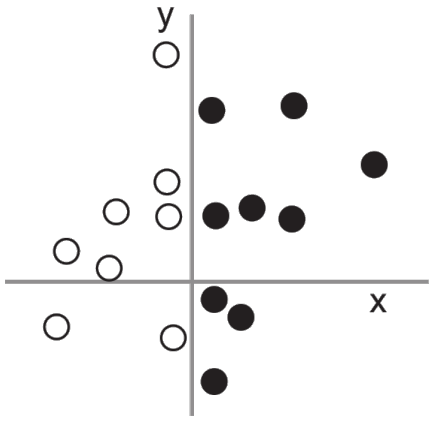


Fonte: Chollet (2018).

Supondo-se que a intenção do algoritmo seja prever se determinado ponto com coordenadas (x, y) é preto ou branco, esse irá procurar uma representação apropriada para realizar a previsão. Nesse caso, o valor de entrada são as coordenadas do ponto, o valor de saída é "preto" ou "branco" e a avaliação da qualidade do algoritmo pode ser realizada calculando-se a porcentagem de acertos da previsão. Uma adequação que o algoritmo poderia propor para efetuar essa previsão está

representada na Figura 9. Nota-se que, através dessa representação, o algoritmo pode inferir que valores de entrada em que $x > 0$ na modelagem realizada determinam que o ponto será preto; caso contrário, será branco. O aprendizado é descrito pelo processo automático de busca por uma representação mais adequada dos dados.

Figura 9 – Representação mais apropriada dos dados para previsão, pois há uma divisão matematicamente previsível entre as duas categorias de cor. Essa nova representação é obtida de forma automática pelos algoritmos de aprendizado.



Fonte: Chollet (2018).

Em linhas gerais, a utilização de um algoritmo de Aprendizado de Máquina acontece em três passos:

1. Tratamento dos dados de treinamento e teste para inserção no algoritmo;
2. Determinação da característica alvo, ou dos resultados esperados;
3. Definição, obtenção e avaliação de métricas de qualidade do treinamento.

1.5.1 Tratamento dos dados

Os algoritmos de *Aprendizado de Máquina* procuram encontrar padrões, anomalias e correlações em um conjunto de dados. Esse processo é chamado de *mineração de dados*, e permite a identificação de ruídos, dados repetidos e nulos dentro do conjunto de dados (GÉRON, 2019). Segundo Chapman (2005), é fundamental que, antes de analisados, os dados sejam tratados para torná-los "próprios para uso", a fim de reduzir erros e melhorar sua visualização.

Durante a preparação dos dados, deve-se atentar para a existência de *outliers*, ou ainda na tradução livre, "valores atípicos". Os *outliers* correspondem a valores com grande afastamento dos demais dados, representando inconsistência. Os *outliers* podem afetar os algoritmos influenciando na modelagem, causando anomalias e enviesando negativamente o resultado da análise. Dessa forma, torna-se importante sua detecção a fim de que o algoritmo não seja afetado (SHARMA, 2021).

Além de detectar *outliers*, a etapa de tratamento de dados engloba a identificação de valores nulos ou repetidos e, posteriormente, a avaliação da sua manutenção no conjunto de dados, preenchendo-os com algum critério, ou os excluindo.

O conjunto de dados inseridos em um algoritmo de Aprendizado de Máquina é dividido em dois subconjuntos, um denominado *conjunto de treinamento* e outro denominado *conjunto de teste*. A parcela de treinamento é a responsável por treinar o algoritmo a criar um padrão de previsão dos dados de teste, que servem para certificar a qualidade dessas previsões. Escolhe-se a porcentagem em que os dados serão divididos para treinar ou testar. Essa divisão pode ser feita de maneira sortida, periódica, cronológica ou sequencial. A inserção dos dados inclui o resultado final esperado, ou característica-alvo, a fim de que o algoritmo preveja qual padrão de características chega até esse resultado.

1.5.2 Métricas de avaliação

Para avaliar a qualidade das previsões realizadas, deve-se utilizar de métricas de avaliação. Essas métricas servem para calcular a distância entre as previsões ideais e as que estão sendo realizadas, a fim de ajustar o algoritmo, processo chamado de aprendizagem.

Deve-se atentar para a possibilidade de *overfitting* (em tradução livre do Inglês, "sobreajuste"), significando que o modelo é precisamente afinado para os dados de treinamento, tornando-os irrelevantes para testagem dos algoritmo. Nesse caso, a previsão beira os 100% de acurácia, mas modela apenas o conjunto de treinamento. Para evitar que isso ocorra, os dados de teste devem ser dados ainda não vistos pelo algoritmo, para que a avaliação das previsões ocorra imparcialmente (HURWITZ; KIRSCH, 2018).

A fim de utilizar as métricas de avaliação para qualificar as previsões feitas, é imprescindível definir os vetores X_i, y_i e $\hat{y}_i$, todos para $i = 1, 2, \dots, D$. X_i e y_i representam os dados de treinamento e teste, sendo o primeiro as características que o algoritmo utiliza para previsão e o segundo o valor real atribuído a cada uma dessas características. $\hat{y}_i$ são os valores previstos pelo algoritmo. As métricas de avaliação designam a qualidade do algoritmo de $\hat{y}_i$ em relação ao valor de y_i, ou seja, o quão precisas as previsões foram em relação aos valores reais.

Erro Quadrático Médio (EQM) O Erro Quadrático Médio, Equação (1.1), é, normalmente, usado para medir a diferença entre os valores previstos e reais, de forma similar à Equação (1.2). A diferença é que o EQM penaliza as diferenças maiores dando um peso maior (LI et al., 2018).

$$EQM = \sqrt{\frac{\sum_{i=1}^{D}(y_i - \hat{y}_i)^2}{D}} \tag{1.1}$$

Erro Médio Absoluto (EMA) O Erro Médio Absoluto calcula a média da diferença, em módulo, entre os valores reais e previstos. Quanto menor o resultado, mais precisas as previsões. Os erros têm o mesmo peso, que determina que a métrica é linear. Essa métrica é definida pela

Equação (1.2) (LI et al., 2018).

$$EMA = \frac{1}{D}\sum_{i=1}^{D}|y_i - \hat{y}_i| \tag{1.2}$$

R-quadrado (R^2) Também conhecido como coeficiente de determinação, o R-quadrado é uma métrica de proporção da variância definida pela Equação (1.3), onde $\overline{y}$ representa o valor médio do conjunto de dados (LI et al., 2018).

$$R^2 = 1 - \frac{\sum_{i=1}^{D}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{D}(y_i - \overline{y})^2} \tag{1.3}$$

R-quadrado varia de 0 a 1. A métrica próxima de 1 indica uma boa correlação entre os dados reais e os previstos.

1.5.3 Tipos de Aprendizado de Máquina

Os algoritmos de Aprendizado de Máquina procuram resolver dois tipos de situações: classificação, na qual a previsão feita é uma variável qualitativa e regressão, que trata-se da previsão de valores numéricos (GÉRON, 2019). A partir disso, e dependendo da natureza do problema e dos dados, pode-se definir algumas categorias de Aprendizado de Máquina, como a seguir.

Aprendizado de Máquina Supervisionado Os modelos de *Aprendizado Supervisionado* têm como ponto de partida conjuntos de dados de entrada estabelecidos, junto de suas respectivas saídas ou rótulos. O Aprendizado Supervisionado visa encontrar padrões nesses dados que possam ser aplicados num processo analítico, ou seja, os dados apresentam características que definem seu significado. Com o intuito de poder generalizar futuras entradas, o algoritmo parte de exemplos (modelos) para extrair os conhecimentos necessários. Os dados de treinamento são rotulados. Isso significa que tem-se as instâncias divididas em características e o resultado esperado, em que o algoritmo encontra o padrão a partir desses para prever com dados ainda não vistos (GÉRON, 2019).

Um exemplo de *problema de classificação* é a determinação se um determinado gene está relacionado à formação de tumores benignos ou malignos. No caso de *problemas de regressão*, pode-se determinar o nível de expressão de um gene em certas condições. Normalmente, esses algoritmos funcionam de forma satisfatória em dados que têm detalhes e compreensão suficientes para representar uma situação real (ELHAIK; GRAUR, 2021).

Dentre os algoritmos de aprendizado de máquina supervisionado, estão: *K Nearest Neighbors* (*KNN*), regressão linear, regressão logística, máquina de vetores de suporte (SVM), árvores de decisão, florestas aleatórias, redes neurais (que também podem ser não supervisionadas), entre outros (GÉRON, 2019). Caso não haja um conjunto de treinamento, não é possível fazer a utilização desse método, partindo-se assim para os algoritmos de Aprendizado de Máquina Não-Supervisionado.

Aprendizado de Máquina Não-Supervisionado Os algoritmos de *Aprendizado Não-Supervisionado* são adequados para problemas com um grande conjunto de dados não rotulados, ou seja, que não possuem uma resposta atrelada aos dados de entrada. Algoritmos nesta categoria atribuem de forma autônoma significado aos dados e preveem através dos padrões e aglomerações encontrados, na prática, adicionando um rótulo aos dados para que esses possam ser tratados como um problema de Aprendizado Supervisionado.

Necessitando de muito menos informações do que no Aprendizado Supervisionado, a identificação de padrões nos dados é feita sem o treinamento, requerendo apenas os dados não rotulados e a quantidade de diferentes rótulos a serem designados. Assim, a rotulação manual dos dados é descartada, mostrando-se sua vantagem tanto em relação ao tempo utilizado e o custo de criação (JONES et al., 2021; ELHAIK; GRAUR, 2021). Segundo Géron (2019), "é aprender sem um professor". Um exemplo desse tipo de Aprendizado de Máquina é a detecção de anomalias, onde o algoritmo é treinado para encontrar padrões e pontos que se diferenciam da maioria dentro do conjunto de dados.

Aprendizado de Máquina Semi-Supervisionado O *Aprendizado Semi-Supervisionado* é a junção dos conceitos dos algoritmos Supervisionados e Não-Supervisionados, tendo como dados de treinamento amostras rotuladas e não rotuladas. A ideia por trás desse tipo de aprendizado é utilizar os exemplos rotulados para obter informações dos dados e usá-los para modelar o processo a partir de exemplos não rotulados. Em contrapartida ao modelo supervisionado, no não-supervisionado não existe a presença de um conjunto de dados pré-estabelecidos para que a máquina aprenda com base nele (LORENA; LEON FERREIRA DE CARVALHO, 2007). Um exemplo de Aprendizado Semi-Supervisionado é treinar o algoritmo para realizar análise de sentimentos[6] em revisões de filmes ou livros. Nesse caso, uma parcela das revisões estão classificadas como positivas ou negativas, e a outra não. Assim, o objetivo do algoritmo é utilizar dos dados Supervisionados para classificar, através do treinamento do algoritmo. os dados Não-Supervisionados.

Aprendizado de Máquina por Reforço No *Aprendizado por Reforço*, o modelo de aprendizado é treinado para tomar decisões através de tentativa e erro, enfrentando uma situação a fim de solucionar um determinado problema. O objetivo desse tipo de Aprendizado de Máquina é maximizar a recompensa total, levando em conta recompensas e penalidades conforme a execução do algoritmo. O ajuste do sistema de recompensas e penalidades é feito pelo programador, a fim de tornar o algoritmo capaz de solucionar o problema. Caso não haja uma maneira "adequada" de realizar uma tarefa, o aprendizado por reforço mesmo assim mostra-se útil, procurando maneiras inesperadas de solucioná-la e chegar ao objetivo. Denomina-se o programador como "agente", que interage com o ambiente a fim de procurar uma forma de maximizar a recompensa acumulada durante o tempo (MILITANI et al., 2021).

[6] Processamento de linguagem natural e linguística computacional para extrair estados de emoções e informações subjetiva de textos.

Um exemplo de aprendizado por reforço é a aprendizagem de robôs para andar em jogos, utilizando como dado outros jogos variados como subsídio para obter a maior recompensa para se movimentar. O agente (ou seja, o próprio sistema de aprendizado por reforço) observa, toma uma ação, realiza, recebe uma recompensa ou penalidade sobre essa, atualiza suas observações, toma outra ação e repete o processo até encontrar a melhor forma de desenvolver determinada tarefa.

1.6 Lista de siglas

Com o objetivo de facilitar a leitura, apresenta-se a Tabela 1 contendo as siglas utilizadas ao longo do texto, bem como seus respectivos significados.

Tabela 1 – Tabela de siglas e seus significados, bem como da seção correspondente no texto.

Sigla	Significado	Seção
ADA	Criptomoeda da *Cardano*	1.4.5
BNB	Criptomoeda da *Binance Coin*	1.4.4
BTC	Criptomoeda *Bitcoin*	1.4.1
DFF	Rede neural tipo *Deep Feed Forward*	2.2.3
ETH	Criptomoeda da *Ethereum*	1.4.3
LSTM	Rede neural tipo *Long Short-Term Memory*	2.2.5
RNN	*Recurrent Neural Network*	2.2.4
SVM	Algoritmo *Support Vector Machines*	2.1
SVR	Algoritmo *Support Vector Regression*	2.1.8
XRP	Criptomoeda da *Ripple*	1.4.2

2 Algoritmos de aprendizado

2.1 *Support Vector Machines*

As Máquinas de Vetores de Suporte (do Inglês, *Support Vector Machines*, doravante simplesmente SVM) pertencem à classe de algoritmos de Aprendizado de Máquina Supervisionado (Seção 1.5.3), significando que os dados de entrada são rotulados e o algoritmo é capaz de classificar um conjunto de dados desconhecido (ABE, 2005). Através da Teoria de Aprendizado Estatístico, segundo Lorena e Leon Ferreira de Carvalho (2003), o SVM procura estabelecer condições matemáticas a fim de adotar, a partir do conjunto de dados (treinamento e teste), um classificador de melhor desempenho (VAPNIK, 1998).

A Teoria de Aprendizado Estatístico, fundamentada por Vapnik (2013), embasa essa técnica de aprendizado de máquina, estabelecendo princípios a serem seguidos pelo algoritmo a fim de obter classificadores que generalizem o conjunto de dados com maior acurácia, isto é, maior relação entre os valores reais e os previstos (VAPNIK, 1998). Dessa forma, previne-se o *overfitting*, conforme definido na Subseção 1.5.2.

Para melhor compreensão do funcionamento do SVM, alguns conceitos iniciais são necessários.

2.1.1 SVM para classificação

Seja $L \subset \mathbb{R}^D$ o conjunto de características, em que cada dado de entrada $X_i \in L, i = 1, \dots, \ell$ tem dimensão D. O conjunto L é dito *linearmente separável* se esses dados de entrada X_i podem ser divididos em duas classes, usualmente representadas por $y_i \in \{+1, -1\}$. Dessa forma, define-se a forma dos dados de treinamento de acordo com a Equação (2.1) (FLETCHER, 2009)

$$(X_i, y_i), \quad X_i \in \mathbb{R}^D, y_i \in \{+1, -1\}, \quad i = 1, \dots, \ell. \tag{2.1}$$

O *problema de classificação binária* consiste em, após realizado o treinamento, e em face de novos dados, determinar a qual classe $\{+1, -1\}$ estes pertencem.

Um conjunto de dados ser linearmente separável significa que é possível construir um subespaço afim de dimensão $D - 1$ do $\mathbb{R}^D$ denominado *hiperplano*, capaz de separá-lo nas duas classes distintas $y_i \in \{+1, -1\}$. O objetivo do SVM é orientar esse hiperplano de forma que ele esteja o mais distante possível dos dados mais próximos entre si de ambas as classes (FLETCHER, 2009).

Matematicamente, podemos representar um hiperplano geral de acordo com a Equação (2.2)

$$w^T x + b = 0, \tag{2.2}$$

onde $w \in \mathbb{R}^D$ é um parâmetro específico para cada hiperplano, $b \in \mathbb{R}$ é o termo independente, que torna o hiperplano um subespaço afim se $b \neq 0$ e $x \in \mathbb{R}^D$ é a variável independente (DEISENROTH; FAISAL; ONG, 2020).

Obter um hiperplano para a classificação binária dos dados, ou seja, a cada X_i há apenas um valor associado no conjunto $\{+1, -1\}$, corresponde a separá-los de forma que todos os dados que estejam de um lado do hiperplano pertençam à classe $+1$, enquanto os dados do outro lado, pertençam à outra classe (DEISENROTH; FAISAL; ONG, 2020). Em suma, deseja-se que w e b sejam tais que

$$\begin{cases} w^T X_i + b \geq 0 & \text{quando } y_i = +1, \\ w^T X_i + b < 0 & \text{quando } y_i = -1, \end{cases} \tag{2.3}$$

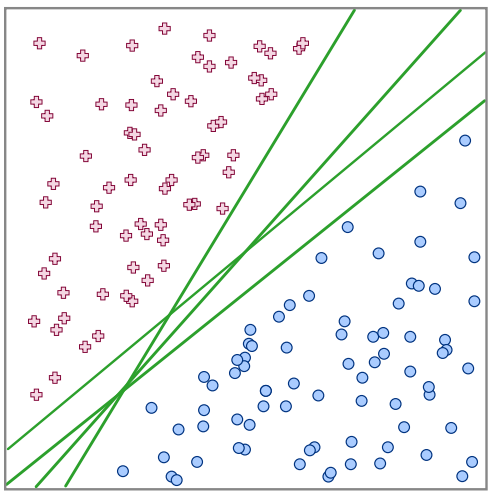
ou, de forma equivalente,

$$y_i(w^T X_i + b) \geq 0. \tag{2.4}$$

Em um primeiro momento, pode-se ter a impressão de que essa separação possa ser feita simplesmente por uma reta, tornando o nome "hiperplano" inadequado. Entretanto, os algoritmos de SVM podem trabalhar com qualquer número D de dimensões, daí a generalização na forma de um hiperplano (HASTIE; TIBSHIRANI; FRIEDMAN, 2001). Caso esteja-se trabalhando com uma dimensão, o hiperplano é um ponto; com duas, uma linha; com três, um plano e, assim por diante.

Segundo Apostolidis-Afentoulis e Liou (2015), a distância entre o hiperplano e o ponto mais próximo dos dados é denominado *margem*, e representado por ρ. Dado um conjunto de dados, é possível gerar infinitos planos separadores, conforme ilustra a Figura 10.

Figura 10 – Exemplos de possíveis hiperplanos separadores para duas classes de dados, uma indicada por ✚ e a outra por ●.

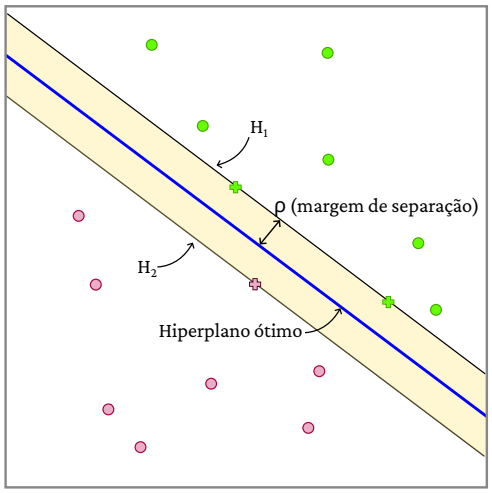


Fonte: Adaptado de Deisenroth, Faisal e Ong (2020).

Assim, procura-se o hiperplano ideal, caracterizado por uma margem máxima entre as duas categorias, conforme enuncia Vapnik (1998). O hiperplano ideal é encontrado através da Minimização do Risco Estrutural e, segundo James et al. (2013), será o hiperplano com maior distância mínima, ou seja, maior margem, entre os dados. O hiperplano ideal procura generalizar da melhor forma os dados utilizados no treinamento (HASTIE; TIBSHIRANI; FRIEDMAN, 2001). A distância entre os dois extremos da margem é chamada *largura da via*, determinada pelo dobro da margem máxima.

Na Figura 11, a área sombreada delimita a margem ótima de separação das duas classes, observando-se três pontos (denominados *vetores de suporte*), indicados nos limites dessa. É importante destacar que a margem ótima depende exclusivamente dos vetores de suporte. A posição dos outros elementos não afeta a margem, desde que estes não a cruzem. A linha azul determina o hiperplano ótimo separador das classes.

Figura 11 – Hiperplano separador ótimo (linha em azul) e margem (faixa em amarelo claro). As duas classes são representadas por ○ e ●, enquanto os vetores de suporte são representados por ✚ e ✚.



Fonte: Adaptado de Hastie, Tibshirani e Friedman (2001).

2.1.2 Hiperplano de margem rígida

É importante ressaltar que neste momento o interesse é apenas no cálculo da distância entre um ponto qualquer X_a e o hiperplano. A Figura 12 será usada como base para determinação desta distância.

Na Figura 12, observa-se o ponto X_a escolhido, sem perda de generalidade, como sendo da classe $y_i = +1$, ou seja, $w^T X_a + b > 0$. X'_a é a projeção ortogonal de X_a no hiperplano, ρ é a distância entre X_a e X'_a, e portanto entre X_a e o hiperplano e $w/\|w\|$ é o vetor unitário de w.

Usando adição vetorial, é possível escrever

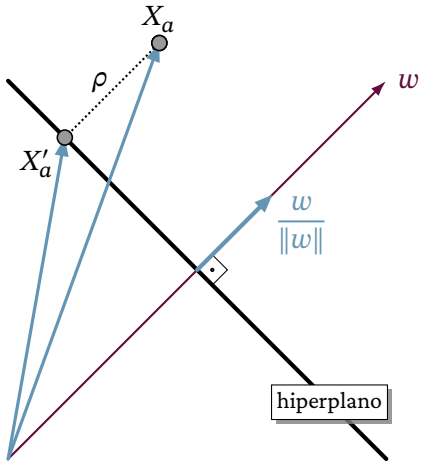
$$X_a = X'_a + \rho \frac{w}{\|w\|}, \tag{2.5}$$

ou seja, ρ é um fator de escala em cima do vetor unitário $w/\|w\|$. É importante notar que se X_a for escolhido como o ponto mais próximo do hiperplano, ρ será exatamente a margem (DEISENROTH; FAISAL; ONG, 2020).

Dessa forma, deseja-se que todos os dados da classe $y_i = +1$ estejam a uma distância ρ ou maior do hiperplano, e que todos os da classe $y_i = -1$ estejam a uma distância ρ ou maior na direção oposta, requerimento que pode ser adicionado à Equação (2.4), resultando em

$$y_i(w^T X_i + b) \geq \rho. \tag{2.6}$$

Figura 12 – Hiperplano separando duas classes de dados. Sem perda de generalidade, assume-se que X_a é classificado como $+1$. X'_a é a projeção ortogonal de X_a no hiperplano ótimo, distando ρ de X_a.



Fonte: Adaptado de Deisenroth, Faisal e Ong (2020).

Assumir que w é tal que $\|w\| = 1$ reforça a interpretação de que só sua direção importa, fazendo com que ρ tenha uma interpretação de distância de fato, já que não é aplicada nenhuma escala nos eixos. Isso também permite escrever a determinação do hiperplano ótimo como um problema de otimização do tipo

$$\begin{aligned} \max_{w,b,\rho} \quad & \rho \\ \text{s.a.} \quad & y_i\left(w^T X_i + b\right) \geq \rho, \quad \rho > 0, \quad i = 1, \ldots, \ell \\ & \|w\| = 1. \end{aligned} \tag{2.7}$$

O problema de otimização da Equação (2.7) foi determinado assumindo-se que $\|w\| = 1$; uma segunda forma de abordar o problema da margem é a de aplicar uma escala nos eixos. A Figura 13 ilustra essa situação, onde o ponto X_a, que pertence à margem, satisfaz exatamente a $w^T X_a + b = 1$.

Como X'_a é a projeção ortogonal de X_a no hiperplano, deve obedecer $w^T X'_a + b = 0$. Usando (2.5), tem-se

$$w^T X'_a + b = 0 \;\Rightarrow\; w^T\left(X_a - \rho\frac{w}{\|w\|}\right) + b = 0 \;\Rightarrow\; \underbrace{w^T X_a + b}_{=1} - \rho\frac{w^T w}{\|w\|} = 0.$$

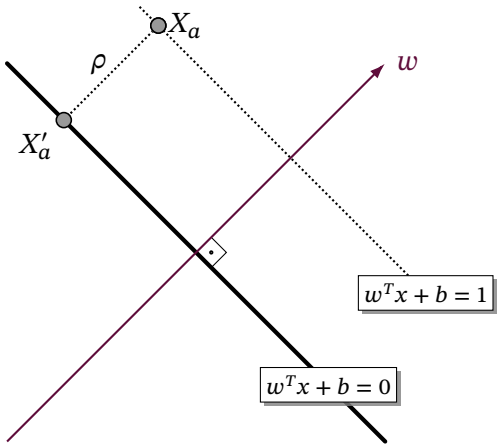
Como $w^T w = \|w\|^2$, obtém-se $1 - \rho\,\|w\| = 0$, e portanto

$$\rho = \frac{1}{\|w\|}. \tag{2.8}$$

Em Deisenroth, Faisal e Ong (2020) os autores demonstram que a Equação (2.8) é equivalente à hipótese de $\|w\| = 1$ feita para o problema (2.7), que pode então ser reformulado como

$$\begin{aligned} \max_{w,b} \quad & \frac{1}{\|w\|} \\ \text{s.a.} \quad & y_i\left(w^T X_i + b\right) \geq 1, \quad i = 1, \ldots, \ell. \end{aligned} \tag{2.9}$$

Figura 13 – Hiperplano separando duas classes de dados, agora normalizados de forma que, se X_a está na margem, então $w^T X_a + b = 1$.



Fonte: Adaptado de Deisenroth, Faisal e Ong (2020).

Por um questão de conveniência, troca-se a função objetivo de maximização de $1/\|w\|$ pela de minimização de $\|w\|^2/2$, tornando o problema de determinação do hiperplano separador ótimo em um problema restrito de otimização quadrática da forma

$$\begin{aligned} \min_{w,b} \quad & \frac{\|w\|^2}{2} \\ \text{s.a.} \quad & y_i\left(w^T X_i + b\right) \geq 1, \quad i = 1, \ldots, \ell. \end{aligned} \tag{2.10}$$

Esse problema recebe o nome de *SVM de margem rígida*, pois em sua formulação não há a previsão de violações das margens: ou os dados são completamente linearmente separáveis, ou o problema não possui solução viável (DEISENROTH; FAISAL; ONG, 2020).

2.1.3 Hiperplano de margem suave

Segundo Lorena e Leon Ferreira de Carvalho (2007), os casos em que os dados são linearmente separáveis são raros na prática. No geral, isso deve-se ao fato de presença de *outliers*, isto é, de valores atípicos que fogem do padrão do conjunto de dados. Levar em conta as *margens suaves* nos hiperplanos significa introduzir variáveis de folga ξ_i, para $i = 1, \ldots, \ell$, à Equação (2.10). Assim sendo, essa folga suaviza as restrições impostas pelo hiperplano de *margem rígida*, obtendo-se a Equação (2.11)

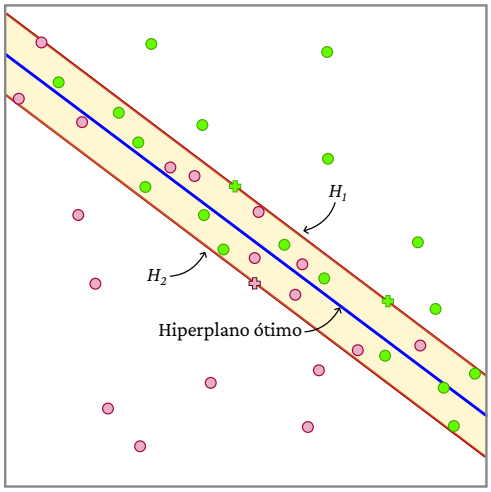
$$y_i\left(w^T X_i + b\right) \geq 1 - \xi_i, \quad i = 1, \ldots, \ell, \quad \xi_i \geq 0. \tag{2.11}$$

Essa suavização permite que alguns dados classificados fiquem entre os hiperplanos H_1 e H_2 (ver Figura 14). A variável de folga ξ_i representa um erro no conjunto de treinamento caso seja maior que 1, conforme Lorena e Leon Ferreira de Carvalho (2007). A fim de minimizar os erros sobre os dados de treinamento, formula-se a função objetivo (2.12)

$$\min_{w,b,\xi} \frac{\|w\|^2}{2} + C \sum_{i=1}^{\ell} \xi_i, \tag{2.12}$$

onde C representa um peso imposto para minimização do erro do conjunto de treinamento, enquanto $\sum_{i=1}^{\ell} \xi_i$ denota a minimização dos erros das margens, pois $0 \leq \xi_i < 1$ indica um dado entre as margens. O termo $C \sum_{i=1}^{\ell} \xi_i$ é chamado de *regularização*, cuja importância dependerá diretamente do valor de C. Não há uma maneira precisa de escolher esse parâmetro, já que este varia de acordo com os dados de entrada. Na prática, é feita uma busca pelo melhor valor de C dentro de um espaço pré-determinado de valores possíveis (KOWALCZYK, 2017).

Figura 14 – No SVM de margem suave há a introdução de variáveis de folga que permitem que alguns pontos violem as restrições e fiquem dentro da margem de separação criada entre os hiperplanos H_1 e H_2.



Fonte: Adaptado de Mello e Ponti (2018).

Unindo a função objetivo (2.12) com as restrições em (2.11) obtemos o seguinte problema de minimização para o SVM de margem suave:

$$\begin{aligned} \min_{w,b} \quad & \frac{\|w\|^2}{2} + C \sum_{i=1}^{\ell} \xi_i \\ \text{s.a.} \quad & y_i\left(w^T X_i + b\right) \geq 1 - \xi_i, \quad i = 1, \ldots, \ell \\ & \xi_i \geq 0. \end{aligned} \tag{2.13}$$

2.1.4 Multiplicadores de Lagrange

Problemas de otimização da forma

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.a.} \quad & g_i(x) = 0, \quad i = 1, 2, \ldots, \ell \end{aligned} \tag{2.14}$$

onde $x \in \mathbb{R}^n$ podem ser resolvidos usando o *método dos multiplicadores de Lagrange* (SWOKOWSKI, 1994; KOWALCZYK, 2017). De forma bem sucinta, o Teorema de Lagrange (SWOKOWSKI, 1994) diz que se as funções f e g_i possuem derivadas parciais primeiras contínuas, com $\nabla g_i \neq 0$, f terá um extremo sujeito ao *vínculo* g_i se seu gradiente tiver mesma direção que o gradiente de g_i

$$\nabla f = \sum_{i=0}^{k} \alpha_i \nabla g_i,$$

bastando portanto resolver $\nabla f - \sum_{i=0}^{k} \alpha_i \nabla g_i = 0$. Neste contexto as constantes α_i recebem o nome de *multiplicadores de Lagrange*.

Seguindo Kowalczyk (2017), o métodos dos multiplicadores de Lagrange pode ser resumido nos seguintes passos:

1. Construa a função lagrangeana $\mathcal{L}(x; \alpha_1, \dots, \alpha_k) = f(x) - \sum_{i=0}^{\ell} \alpha_i g_i(x)$, onde x é o vetor de variáveis de entrada, $\alpha_i, i = 1, \dots, \ell$ são os multiplicadores de Lagrange, f é a função objetivo e $g_i, i = 1, \dots, \ell$ são as restrições;

2. Obtenha o gradiente da lagrangeana em relação às variáveis de entrada x, $\nabla_x \mathcal{L}(x; \alpha_1, \dots, \alpha_k)$;

3. Resolva $\nabla_x \mathcal{L}(x; \alpha_1, \dots, \alpha_k) = 0$.

2.1.5 Lagrange para hiperplanos de margem rígida

O método dos multiplicadores de Lagrange permite construir a lagrangeana para o problema da obtenção do hiperplano ótima de margem rígida (2.10), na forma

$$\mathcal{L}(w, b; \alpha_1, \dots, \alpha_\ell) = \frac{\|w\|^2}{2} - \sum_{i=0}^{\ell} \alpha_i \big[y_i (w^T X_i + b) - 1 \big], \tag{2.15}$$

onde w e b definem o hiperplano e $\alpha_i, i = 1, \dots, \ell$ são os multiplicadores de Lagrange. Dessa forma, o problema de determinação do hiperplano de margem rígida continua sendo um problema de minimização em w e b como na Equação (2.10), mas agora requerendo que as derivadas parciais de $\mathcal{L}(w, b; \alpha_1, \dots, \alpha_\ell)$ em relação a cada α_i se anulem, sujeito às restrições de que $\alpha_i \geq 0$ (BURGES, 1998). Ainda de acordo com Burges (1998), este problema pode ser escrito como uma otimização convexa chamada *problema primal*, na forma

$$\begin{aligned} \min_{w,b} \max_{\alpha_i} \quad & \mathcal{L}(w, b; \alpha_1, \dots, \alpha_\ell) \\ \text{s.a.} \quad & \alpha_i \geq 0, \quad i = 1, \dots, \ell. \end{aligned} \tag{2.16}$$

O princípio da dualidade permite obter uma forma mais conveniente de resolver esse problema. Nele, um problema de otimização primal pode ser escrito de forma alternativa — e em alguns casos, de resolução mais simples — chamada *forma dual* (BOYD, 2004; BAZARAA; JARVIS; SHERALI, 2010). Como o problema (2.16) é de minimização, necessariamente seu dual será de maximização, cuja solução fornece um limite superior para a solução do primal. Mais: a convexidade do problema primal permite afirmar que a *dualidade forte* vale através do Teorema de Slater, e portanto a solução do problema dual é igual à do primal (BOYD, 2004; KOWALCZYK, 2017).

Para a construção do problema dual associado a (2.16) deve-se obter as derivadas parciais em relação a w e b da Equação (2.15)

$$\nabla_w \mathcal{L} = w - \sum_{i=0}^{\ell} \alpha_i y_i X_i = 0, \tag{2.17}$$

$$\frac{\partial \mathcal{L}}{\partial b} = -\sum_{i=0}^{\ell} \alpha_i y_i = 0, \tag{2.18}$$

já igualadas a zero de acordo com o preconizado pelo método de Lagrange. A Equação (2.17) pode ser resolvida diretamente, levando a

$$w = \sum_{i=0}^{\ell} \alpha_i y_i X_i. \tag{2.19}$$

Substituindo a solução w dada por (2.19) em (2.18) e denotando $W(b; \alpha_1, \dots, \alpha_\ell) = \mathcal{L}(w, b; \alpha_1, \dots, \alpha_\ell)$, obtemos (KOWALCZYK, 2017)

$$\begin{aligned}
W(b; \alpha_1, \dots, \alpha_\ell) &= \frac{1}{2}\left(\sum_{i=1}^{\ell} \alpha_i y_i X_i^T\right)\left(\sum_{j=1}^{\ell} \alpha_j y_j X_j\right) - \sum_{i=1}^{\ell} \alpha_i \left\{ y_i \left[\left(\sum_{j=1}^{\ell} \alpha_j y_j X_j^T \right) X_i + b \right] - 1 \right\} \\
&= \frac{1}{2} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} \alpha_i \alpha_j y_i y_j X_i^T X_j - \sum_{i=1}^{\ell} \alpha_i y_i \left[\left(\sum_{j=1}^{\ell} \alpha_j y_j X_j^T \right) X_i + b \right] + \sum_{i=1}^{\ell} \alpha_i \\
&= \frac{1}{2} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} \alpha_i \alpha_j y_i y_j X_i^T X_j - \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} \alpha_i \alpha_j y_i y_j X_i^T X_j - b \sum_{i=1}^{\ell} \alpha_i y_i + \sum_{i=1}^{\ell} \alpha_i \\
&= \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} \alpha_i \alpha_j y_i y_j X_i^T X_j - b \underbrace{\sum_{i=1}^{\ell} \alpha_i y_i}_{=0,\ (2.18)}.
\end{aligned}$$

Definindo a matriz $H \in \mathbb{R}^{\ell \times \ell}$ por $h_{ij} = y_i y_j X_i^T X_j$ e o vetor $\alpha = [\alpha_1, \dots \alpha_\ell] \in \mathbb{R}^\ell$, a Equação (2.18) pode ser escrita como

$$W(\alpha_1, \dots, \alpha_\ell) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \alpha^T H \alpha. \tag{2.20}$$

que é a função objetivo do *problema dual de Wolfe* (KOWALCZYK, 2017)

$$\begin{aligned}
\max_{\alpha_i} \quad & \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \alpha^T H \alpha \\
\text{s.a.} \quad & \alpha_i \geq 0, \quad i = 1, \dots, \ell \\
& \sum_{i=1}^{\ell} \alpha_i y_i = 0.
\end{aligned} \tag{2.21}$$

É importante ressaltar que na formulação do problema dual de Wolfe, existe exatamente um multiplicador de Lagrange para cada ponto do conjunto de treinamento. Neste caso, as seguintes possibilidades são válidas (BURGES, 1998):

- Os pontos de treinamento em que $\alpha_i > 0$ estarão necessariamente ou em H_1 ou em H_2 (*cf.* Figura 11) e recebem o nome de *vetores de suporte*.

- Os outros pontos correspondem a $\alpha_i = 0$ e estão ou em H_1 ou em H_2, caso em que $y_i(w^T X_i + b) = 1$, ou estão dos lados respectivos de H_1 ou H_2, com $y_i(w^T X_i + b) > 1$.

Ainda de acordo com Burges (1998), os vetores de suporte são tais que se, se todos os outros pontos fossem removidos do conjunto de treinamento, ou movidos sem cruzar as fronteiras de H_1 ou H_2, o hiperplano obtido seria exatamente o mesmo.

2.1.6 Lagrange para hiperplanos de margem suave

O mesmo desenvolvimento (VAPNIK, 1998) pode ser feito para obter-se o problema dual de Wolfe para o SVM com hiperplanos de margem suave (2.13), chegando a

$$\begin{aligned} \max_{\alpha_i} \quad & \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2}\alpha^T H \alpha \\ \text{s.a.} \quad & 0 \leq \alpha_i \leq C, \quad i = 1, \dots, \ell \\ & \sum_{i=1}^{\ell} \alpha_i y_i = 0. \end{aligned} \tag{2.22}$$

2.1.7 Obtendo o hiperplano ótimo

Após o cálculo dos multiplicadores de Lagrange, seja no problema de margem rígida (2.21) ou de margem suave (2.22), resta obter o hiperplano, ou seja, efetuar o cálculo de w e b. O cálculo de w é bastante simples, bastando usar a Equação (2.19). O cálculo de b merece algumas considerações adicionais.

Uma opção vem da restrição $y_i(w^T X_i + b) \geq 1$, proveniente do problema (2.10). Os pontos mais próximos do hiperplano ótimo vão apresentar uma margem de 1, ou seja, $y_i(w^T X_i + b) = 1$ para os vetores de suporte. Lembrando que $y_i^2 = 1$, multiplicando por y_i de ambos os lados obtemos

$$y_i^2(w^T X_i + b) = y_i \quad \Rightarrow \quad w^T X_i + b = y_i,$$

que permite escrever

$$b = y_i - w^T X_i. \tag{2.23}$$

Note que a escolha de X_i é arbitrária: qualquer vetor de suporte é válido. Em Bishop (2006) preconiza-se o uso da média no conjunto de vetores de suporte, para maior estabilidade numérica:

$$b = \frac{1}{\overline{\ell}} \sum_{i \in S} (y_i - w^T X_i) \tag{2.24}$$

onde S é o conjunto de vetores de suporte e $\overline{\ell} < \ell$ é o número de vetores de suporte. Já Vapnik (1998) e Cristianini e Shawe-Taylor (2000) propõem uma abordagem baseada na média dos vetores de suporte mais próximos dos lados positivo e negativo:

$$b = -\frac{\max_{y_i=-1} w^T X_i + \min_{y_i=+1} w^T X_i}{2}. \tag{2.25}$$

2.1.8 SVM para Regressão

A principal mudança no problema de regressão é que agora $y_i \in \mathbb{R}$, em lugar de apenas duas classes discretas, $\{+1, -1\}$ como no problema de classificação. Assim, podemos tomar $L \subset \mathbb{R}^D$ como o conjunto de características, em que cada dado de entrada $X_i \in L, i = 1, \dots, \ell$ tem dimensão D, como anteriormente, mas agora com $y_i \in \mathbb{R}$. Assim, define-se a forma dos dados de treinamento de acordo com a Equação (2.26) (VAPNIK, 1998; FLETCHER, 2009)

$$(X_i, y_i), \quad X_i \in \mathbb{R}^D, y_i \in \mathbb{R}, \quad i = 1, \dots, \ell. \tag{2.26}$$

Por simplicidade será abordado o *problema de regressão linear* que consiste em, após realizado o treinamento, e em face de novos dados $X \in \mathbb{R}^D$, determinar w^T e b em $f(X) = w^T X + b$, minimizando uma métrica conveniente[1]. É importante notar que a função linear $f(X)$ continua definindo um hiperplano, que será denominado H.

A métrica mais comumente utilizada é a ϵ-insensível, definida por

$$|f(X) - y|_\epsilon = \begin{cases} 0, & \text{se } |f(X) - y| \leq \epsilon \\ |f(X) - y| - \epsilon, & \text{caso contrário} \end{cases}, \tag{2.27}$$

onde $X \in \mathbb{R}^D$ e $y \in \mathbb{R}$ (VAPNIK, 1998). Essa métrica não penaliza dados próximos do hiperplano H, ao contrário do que ocorre com dados que se afastam destes: de fato, quanto mais distante um dado está de H, maior sua penalização (FLETCHER, 2009).

A formulação primal do problema de regressão é totalmente similar a do problema de classificação binária de margem suave (2.13), porém considerando a introdução de duas variáveis de folga $\xi_i^+ \geq 0$ e $\xi_i^- \geq 0$, uma para pontos de um lado do hiperplano, outra para pontos do lado oposto. Dessa forma obtém-se

$$\min_{w,b} \quad \frac{\|w\|^2}{2} + C \sum_{i=1}^{\ell} (\xi_i^+ + \xi_i^-) \tag{2.28}$$

$$\text{s.a.} \quad t_i - y_i \leq \epsilon + \xi_i^+, \quad i = 1, \ldots, \ell \tag{2.29}$$

$$y_i - t_i \leq \epsilon + \xi_i^- \tag{2.30}$$

$$\xi_i^+, \xi_i^- \geq 0, \tag{2.31}$$

onde $t_i = w^T X_i + b$.

Introduzindo-se os multiplicadores de Lagrange α_i^+ e α_i^- para as restrições (2.29) e (2.30), μ_i^+ e μ_i^- para as restrições (2.31), todos não negativos, pode-se construir a lagrangeana da forma

$$\mathcal{L}(t_i; \alpha^+, \alpha^-, \mu^+, \mu^-) = \frac{\|w\|^2}{2} + C \sum_{i=1}^{\ell} (\xi_i^+ + \xi_i^-) - \sum_{i=1}^{\ell} (\mu_i^+ \xi_i^+ + \mu_i^- \xi_i^-) - \\ - \sum_{i=1}^{\ell} \alpha_i^+ (\epsilon + \xi_i^+ + y_i - t_i) - \sum_{i=1}^{\ell} \alpha_i^- (\epsilon + \xi_i^- - y_i + t_i), \tag{2.32}$$

lembrando que as incógnitas w e b compõem a variável t_i.

Para a construção do dual, deve-se obter as derivadas parciais em relação a w, b, ξ_i^+ e ξ_i^- e

[1] No contexto do SVM, várias métricas (também chamadas de *funções de perda*) podem ser utilizadas, inclusive recuperando o método dos quadrados mínimos original. Neste caso específico, a métrica utilizada chama-se *ϵ-insensível.* A discussão teórica sobre essas métricas foge do escopo deste trabalho, mas um tratamento completo do assunto pode ser encontrado em Vapnik (1998).

zerá-las, resultando em

$$\nabla_w \mathcal{L} = 0 \quad \Rightarrow \quad w = \sum_{i=1}^{\ell} (\alpha_i^+ - \alpha_i^-) X_i, \tag{2.33}$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \quad \Rightarrow \quad \sum_{i=1}^{\ell} (\alpha_i^+ - \alpha_i^-) = 0, \tag{2.34}$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i^+} = 0 \quad \Rightarrow \quad C = \alpha_i^+ + \mu_i^+, \tag{2.35}$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i^-} = 0 \quad \Rightarrow \quad C = \alpha_i^- + \mu_i^-. \tag{2.36}$$

Observando que (2.35) e (2.36), junto da não negatividade de μ_i^+ e μ_i^-, implicam que $0 \leq \alpha_i^+ \leq C$ e $0 \leq \alpha_i^- \leq C$ e introduzindo as Equações (2.33) e (2.34) em (2.32), pode-se escrever o problema dual para regressão

$$\begin{aligned} \max_{\alpha_i^+, \alpha_i^-} \quad & \sum_{i=1}^{\ell} (\alpha_i^+ - \alpha_i^-) t_i - \epsilon \sum_{i=1}^{\ell} (\alpha_i^+ - \alpha_i^-) - \frac{1}{2} (\alpha^+ - \alpha^-)^T H (\alpha^+ - \alpha^-) \\ \text{s.a.} \quad & 0 \leq \alpha_i^+, \alpha_i^- \leq C, \quad i = 1, \ldots, \ell \\ & \sum_{i=1}^{\ell} (\alpha_i^+ - \alpha_i^-) = 0, \end{aligned} \tag{2.37}$$

onde $H \in \mathbb{R}^{\ell \times \ell}$ é tal que $h_{ij} = X_i^T X_j$.

A obtenção de w dá-se pela Equação (2.33), enquanto b pode ser obtido por uma média similar a da Equação (2.24):

$$b = \frac{1}{\ell} \sum_{i \in S} \Big[t_s - \epsilon \sum_{m \in S} (\alpha_m^+ - \alpha_m^-) X_m^T X_i \Big]. \tag{2.38}$$

2.1.9 SVM não linear e *kernels*

Até o momento todo o desenvolvimento do SVM foi baseado na premissa de que o conjunto de treinamento é linearmente separável (SVM de margem rígida) ou próximo disso, permitindo alguma flexibilidade (SVM de margem suave), ou ainda, no caso da regressão, de que o ajuste será feito com uma função linear.

E no caso mais geral, em que os dados não são linearmente separáveis, ou com ajustes que não são bem representados por funções lineares? A resposta está no problema dual de Wolfe, Equação (2.21): a matriz H é construída a partir dos produtos escalares do dados de treinamento, já que $h_{ij} = y_i y_j X_i^T X_j$, o que confere a natureza linear ao SVM clássico.

Essa ideia de produto escalar pode ser generalizada através de uma classe de funções denominadas *kernels* (DEISENROTH; FAISAL; ONG, 2020), definidas por $k : L \times L \to \mathbb{R}$ de tal forma que

$$k(X_i, X_j) = \varphi(X_i)^T \varphi(X_j), \tag{2.39}$$

para $\varphi : L \to \mathcal{H}$, com $\mathcal{H}$ um espaço de Hilbert conveniente e dependente da escolha de φ. De fato, a escolha da função φ é que permitirá a inclusão de não linearidades no método[2].

Algumas escolhas tradicionais para a função *kernel* k são:

Polinomial A função *kernel* polinomial é dada pela a Equação (2.40)

$$k(x, y) = (\gamma x^T y + \beta)^d, \tag{2.40}$$

onde x e y são os vetores de entrada, d é o grau do polinômio β e γ são constantes.

RBF – Radial Basis Function Também chamada de *kernel* gaussiano, a Equação (2.41) ilustra o *kernel* de base radial

$$k(x, y) = \exp(-\gamma |x - y|^2) \tag{2.41}$$

onde x e y são os vetores de entrada, e γ é uma constante.

Sigmoide dado pela Equação (2.42), essa função também é conhecida como *kernel* de tangente hiperbólica

$$k(x, y) = \tanh(\gamma x^T y + \beta), \tag{2.42}$$

onde x e y são os vetores de entrada e β e γ são constantes.

2.1.10 Exemplos de SVM

SVM margem rígida

Para ilustrar o processo de classificação com SVM, um conjunto linearmente separável de dados será usado para treinamento da implementação do método presente na biblioteca SCIKIT-LEARN (PEDREGOSA et al., 2011). O conjunto de dados é composto de pares ordenados no plano (x, y), bem como a classe a que cada par pertence, dentro do conjunto $\{+1, -1\}$, como pode ser apreciado na Tabela 2.

Tabela 2 – Coordenadas (x, y) no plano de um conjunto de dados linearmente separável, com suas respectivas classes.

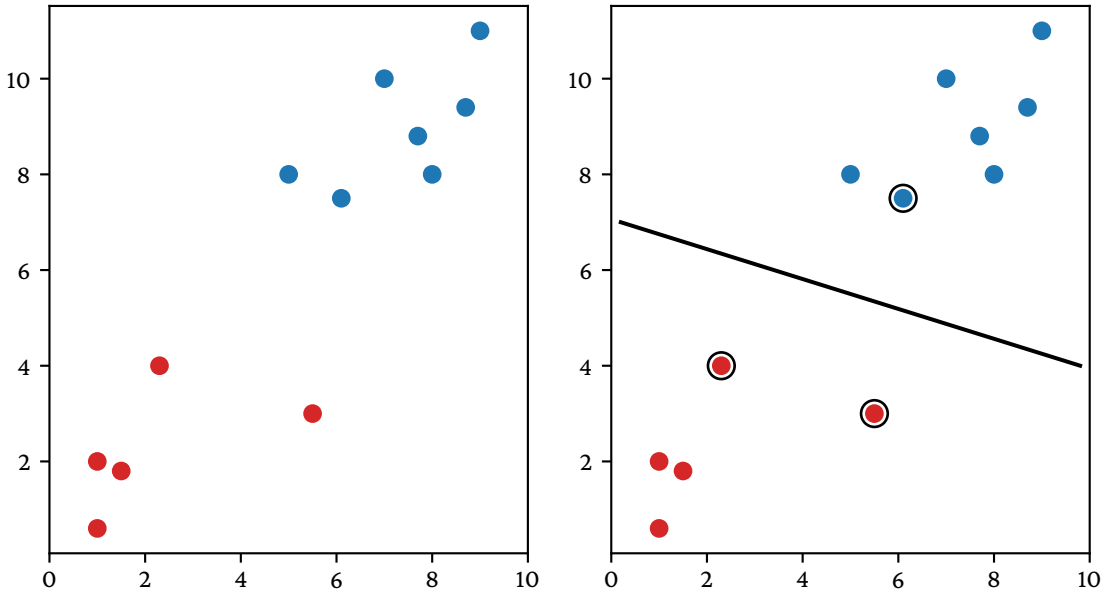
x	1,0	5,0	1,5	8,0	1,0	9,0	7,0	8,7	2,3	5,5	7,7	6,1
y	2,0	8,0	1,8	8,0	0,6	11,0	10,0	9,4	4,0	3,0	8,8	7,5
Classe	−1	1	−1	1	−1	1	1	1	−1	−1	1	1

Fonte: Adaptado de McGregor (2020).

Como pode ser observado na Figura 15, ambos os conjuntos de pontos foram perfeitamente separados pelo hiperplano (de fato, uma reta) de equação $y = -0{,}312x + 7{,}062$.

[2] A construção de funções de *kernel* k válidas, ou seja, que sejam de fato produtos internos, deve seguir as hipóteses do *Teorema de Mercer*. Esse teorema está fora do escopo deste trabalho, mas uma discussão mais aprofundada sobre ele pode ser encontrada em Vapnik (1998).

Figura 15 – Exemplo de aplicação do SVM para classificação no $\mathbb{R}^2$. O plano separador tem equação $y = -0{,}312x + 7{,}062$. Cada classe é representada por uma cor (vermelha ou azul), enquanto os vetores de suporte estão destacados com uma circunferência na cor preta.



Fonte: Adaptado de Varoquaux, Müller e Grobler (2023).

SVM com *kernel* RBF

Em contraste ao exemplo anterior, são fornecidos três conjuntos de dados claramente não separáveis de forma linear. Para que o SVM consiga classificar e separar os dados, foi utilizado o *kernel* RBF. Como pode ser visto na Figura 16, em cada um dos conjuntos, SVM associado ao *kernel* RBF foi capaz de gerar classificações capazes de separar os pontos vermelhos e azuis.

A construção dos conjuntos foi feita com o auxílio das funções `make_moons`[3], `mark_circles`[4] e `make_classification`[5], todas parte do módulo gerador de conjuntos de dados do SCIKIT-LEARN (`sklearn.datasets`).

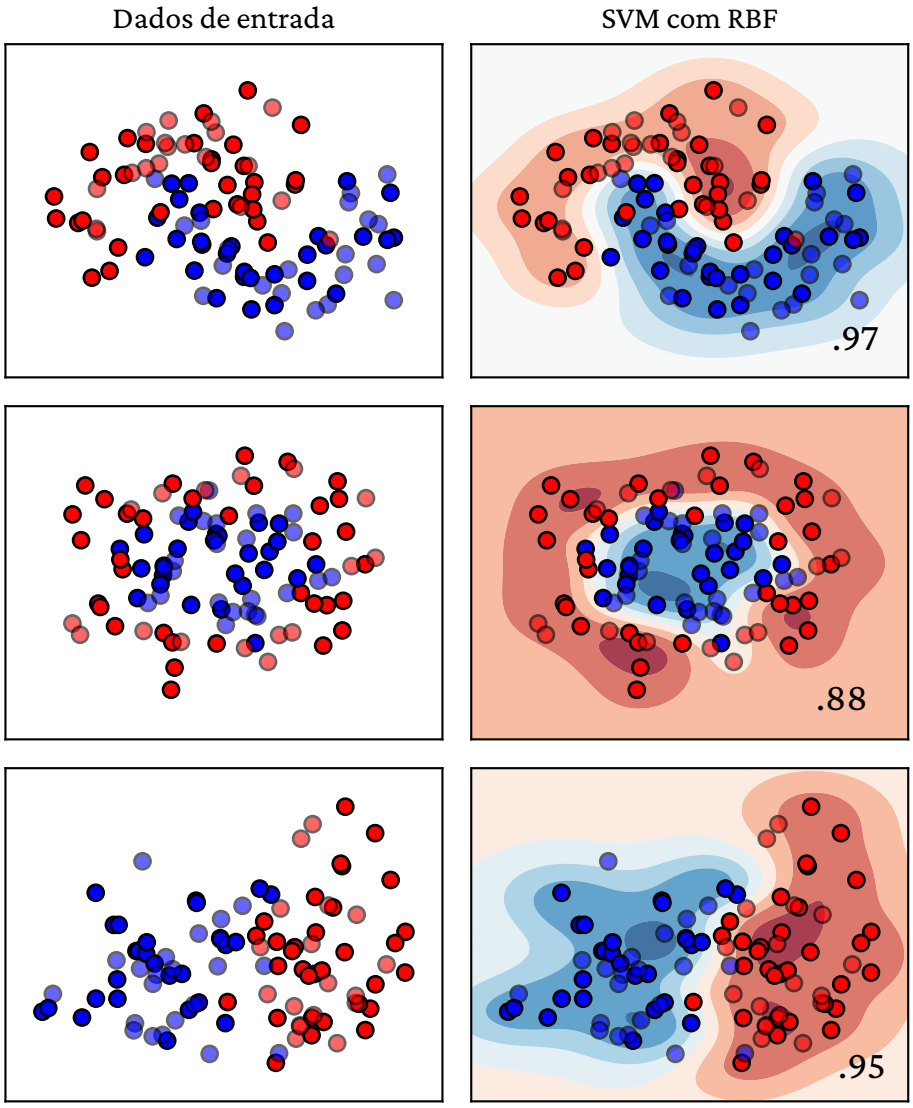
SVM para regressão

Neste exemplo será utilizada a classe `SVR` do pacote SCIKIT-LEARN (PEDREGOSA et al., 2011). Ela implementa o método SVM para regressão, dispobilizando diversos *kernels*. Para este exemplo serão utilizados o RBF (*radial basis function*) e o polinomial de grau 3. A função modelo para criação dos dados de treinamento é $f(x) = \text{sen}\, x$, adicionada de um ruído aleatório uniforme no intervalo $[-1{,}5; 1{,}5]$ em alguns pontos de seu domínio.

Na Figura 17 podem ser apreciados os resultados da aplicação de ambos os *kernels*, que apresentam bons resultados mesmo levando em consideração os pontos com o ruído aleatório adicionado.

[3] `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_moons.html`
[4] `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_circles.html`
[5] `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_classification.html`

Figura 16 – Exemplo de aplicação do SVM para classificação no $\mathbb{R}^2$ usando um *kernel* do tipo RBF. Os pontos das duas classes — vermelhos e azuis — claramente não são linearmente separáveis, mas o uso do *kernel* RBF garante a qualidade de classificação, separando os dois conjuntos.



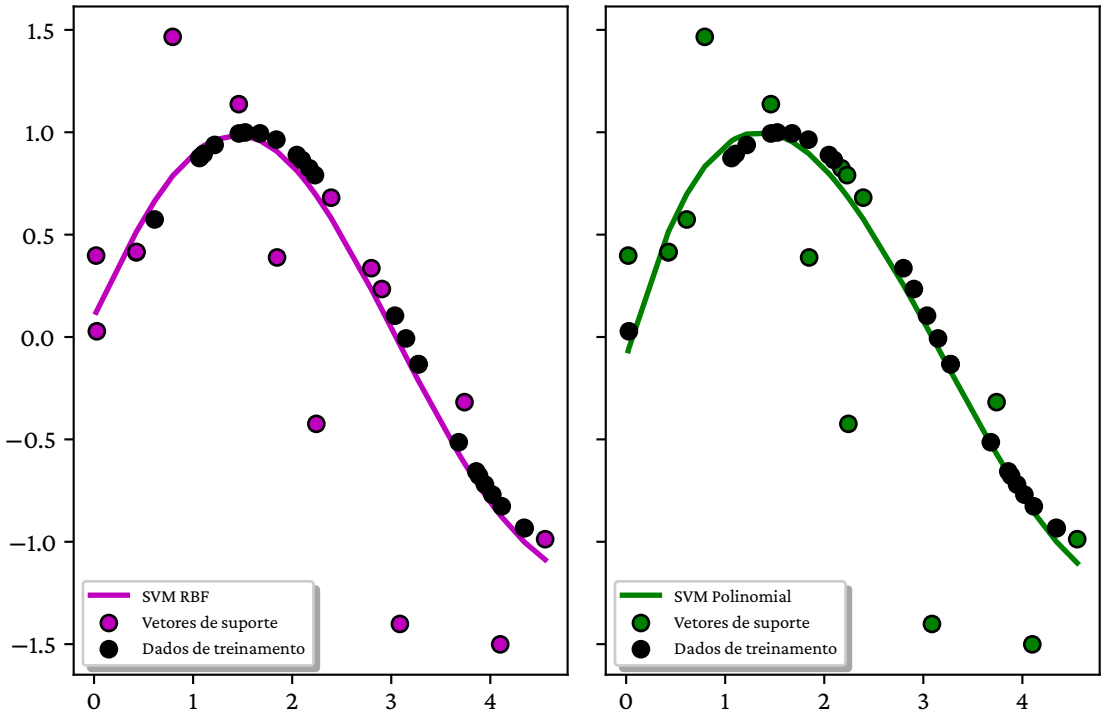
Fonte: Adaptado de Varoquaux, Müller e Grobler (2023).

2.2 Redes Neurais Artificiais

Nessa seção a referência base é a obra de Aggarwal (2018). Outras referências que se fizerem necessárias serão citadas da forma usual.

As redes neurais têm se tornado uma área de grande foco nos últimos tempos em Ciência da Computação, em particular no Aprendizado de Máquina, revolucionando a forma como os computadores emulam o aprendizado humano. Elas vêm sendo estudadas há muitas décadas: isso torna-se evidente, uma vez que, em 1996, Warner e Misra (1996) propuseram um olhar aprofundado da história das redes neurais e de seu desenvolvimento, atribuindo seu início a uma pesquisa da década de 1940.

Figura 17 – Exemplo de aplicação do SVM para regressão usando *kernels* do tipo RBF e polinomial de grau 3. Os dados de treinamento (em preto) foram gerados a partir da função $f(x) = \text{sen}\, x$, acrescentados de um ruído aleatório. Em roxo e verde estão representados os vetores de suporte respectivamente para os *kernels* RBF e polinomial.



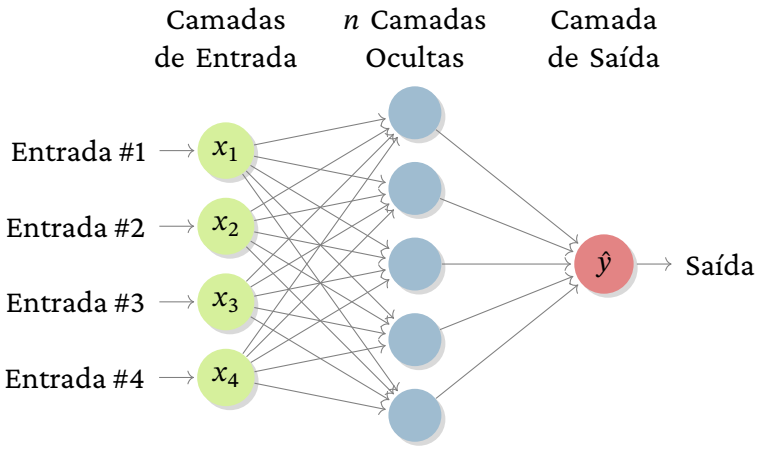
Fonte: Adaptado de Developers (2023).

A ideia primordial de uma rede neural vem do conceito biológico de que os neurônios do cérebro humano são conectados e transmitem informações na forma de impulsos elétricos, o que possibilitou a criação de redes neurais artificiais, que são elementos computacionais interconectados que simulam o comportamento de dados, representando os neurônios. A Figura 18 representa um exemplo de arquitetura básica de uma rede neural artificial.

Na Figura 18 as conexões são representadas pelas setas que conectam diferentes neurônios, representados pelos círculos, de forma ponderada por pesos. Os círculos azuis representam os neurônios das camadas intermediárias, onde os dados são efetivamente processados. A camada de entrada é representada em verde (aqui com quatro neurônios), e serve para que os dados sejam alimentados na rede e para posterior processamento nas camadas intermediárias, até que, enfim, os resultados chegam aos neurônios da camada de saída (aqui com apenas um neurônio), representado pelo círculo vermelho (ACADEMY, 2022). Os neurônios, também chamados de nós, conectados entre si formam um conjunto de funções matemáticas ponderadas pelos pesos que processam e transmitem as informações.

A seleção da arquitetura de uma rede neural, ou seja, o número de camadas e de neurônios e a forma de conexão entre estes, depende dos dados a serem processados e o objetivo. Segundo Chollet (2018), a escolha de uma arquitetura adequada para uma rede neural é uma tarefa que se aproxima mais da arte do que da Ciência.

Figura 18 – Componentes básicos de uma rede neural. Em verde tem-se a camada de entrada (no caso, com quatro neurônios), em azul as camadas ocultas e finalmente a de saída em vermelho (com um neurônio). Cada círculo representa um neurônio e cada seta uma conexão de dados direcional.



Fonte: Adaptado de Shiruru (2015).

Com o tempo, a construção de redes neurais artificiais viáveis foi alcançada pela disponibilidade de armazenamento de grandes conjuntos de dados e pela criação de processadores poderosos, possibilitando um aumento na complexidade das arquiteturas e, consequentemente, a realização de tarefas mais complexas, impulsionando o desenvolvimento de técnicas modernas de Aprendizado de Máquina (PUJOL-PERICH et al., 2021).

Mishra e Gupta (2017) e Aggarwal (2018) pontuam exemplos de aplicações que se utilizam de algoritmos que envolvem redes neurais, como reconhecimento facial, classificação de imagens e reconhecimento de voz, dentre outros. Isso deve-se à capacidade da rede neural de aprender por experiência e capturar padrões em grandes volumes de dados. Com a combinação de algoritmos e arquiteturas complexas, o poder de solucionar problemas das redes neurais vem crescendo, mostrando que estas ainda perdurarão como uma ferramenta importante na área de Aprendizado de Máquina e aplicações que envolvam Inteligência Artificial (MISHRA; GUPTA, 2017).

2.2.1 Cérebro Humano versus Redes Neurais

O debate acerca do potencial da inteligência artificial de dominar e substituir a inteligência humana é algo discutido há muito tempo. Todavia, o objetivo primordial é utilizar da inteligência artificial para facilitar a vida humana e melhorar suas capacidades. Ela pode prover uma gama maior de habilidades que ultrapassam a capacidade humana, tanto em complexidade quanto em tempo.

Segundo Basu, Han e Garimella (2019), a inteligência artificial pode tornar-se muito importante no ramo de tomadas de decisão, provendo *insights* e recomendações sobre conjuntos de dados para uma tarefa específica que ultrapassam a capacidade humana, seja para obter soluções ou detectar padrões. A inteligência artificial também pode ser amplamente utilizada para melhorar a

segurança, no sentido de realizar previsões mais corretas, com camadas adicionais de proteção e reduzir a quantidade de erros humanos. Explorando-se esse potencial, espera-se aumentar a qualidade de vida dos humanos com os computadores.

A inteligência artificial baseada em redes neurais permite a comunicação direta entre computadores e humanos para que ocorra uma interação mais natural e maior entendimento dos processos a serem realizados. Um exemplo de comunicação direta entre a máquina e o humano é o *ChatGPT*, inteligência artificial capaz de realizar inúmeras tarefas atendendo a requisições específicas, como montar listas de compras ou criar cronogramas de alimentação e blocos de códigos, para citar alguns exemplos (OPENAI, 2021).

O crescimento do uso de inteligência artificial vem causando vários debates acerca de questões éticas. Em 2023, por exemplo, *Elon Musk* iniciou testes com seu *chip* cerebral, que procura tratar danos cerebrais em humanos (METRÓPOLES, 2023). De acordo com um estudo realizado por Zhang et al. (2021), o maior problema das pessoas "aumentadas" com inteligência artificial está relacionado, principalmente, a situações de segurança de dados, privacidade, algoritmos maliciosamente enviesados e responsabilidade. Os dados podem estar dispostos na *internet* de forma que algoritmos de inteligência artificial consigam coletá-los, processá-los e distribuí-los de maneira que o usuário sequer faça ideia. Surge assim uma preocupação acerca dos limites dos computadores.

Humanos e computadores, de forma conjunta, conseguiram dar grandes passos em relação ao avanço tecnológico. Sem dúvidas, a inteligência artificial desempenhou um papel substancial para que isso acontecesse. Os computadores se mostraram capazes de processar e analisar uma grande quantidade de dados de forma mais rápida que os humanos, que têm capacidades mais limitadas nesse sentido, apesar de serem esses os responsáveis por programar os algoritmos. Espera-se, que no futuro, essa integração entre inteligência artificial e humanos se estenda, unindo forças para criar algo memorável e que impacte positivamente na sociedade (VALLE-CRUZ; FERNANDEZ--CORTEZ; GIL-GARCIA, 2022).

2.2.2 Redes Neurais de Camada única — *Perceptron*

Os *perceptrons* são uma arquitetura de rede neural que possui apenas uma camada e seu uso tornou-se popular recentemente devido a sua habilidade de facilmente classificar dados e identificar padrões, além de serem utilizados para construção de redes neurais mais complexas. Park e Lek (2016) pontuam que esse tipo de rede neural possui, assim como as outras, suas vantagens e desvantagens. A principal vantagem é a facilidade de treinar e prover generalizações rápidas. Para casos não lineares, é viável o uso do *perceptron* para classificação com diversas características. Em contraponto, redes neurais de camada única são sensíveis à ocorrência de sobreajuste (*overfitting*), resultando em previsões ineficazes fora do conjunto de treinamento.

Os *perceptrons* são altamente versáteis, sendo usados em várias aplicações que envolvem processamento e reconhecimento de imagens e operações lógicas. Eles podem ser utilizados para criar e reconhecer padrões em sistemas de tempo discretos lineares ou não lineares. Um sistema

de tempo discreto é uma sequência de números que acontecem em situações discretas de tempo, como modelos de renda nacional, rastreamento de radar e estudos populacionais, entre outros. Ademais, eles são usados para controlar movimentos de robôs, através do reconhecimento do ambiente (POPESCU et al., 2009).

Segundo Sussner (1998), os *perceptrons* são limitados em capacidade para resolver problemas complexos devido ao fato de apresentarem apenas uma relação linear entre as entradas e saídas. Isso significa que eles não são aptos para resolver situações não lineares. Ademais, os *perceptrons*, por serem compostos por apenas uma camada, não determinam os parâmetros durante o treinamento, ou seja, esses devem ser pré-determinados. Assim, eles são limitados a um único e estático conjunto de pesos W.

Seja $L \subset \mathbb{R}^D$ o conjunto de características em que o dado de entrada $X \in L, X = [x_1, \dots, x_D]$ representa um vetor de características com dimensão D e $y \in \{-1, +1\}$ seja a variável de classe binária. O vetor $W \in \mathbb{R}^D, W = [w_1, \dots, w_D]$ representa o vetor de pesos, que ponderam o quão relevantes as informações são para o modelo. No *perceptron*, o nó de saída computa a função linear $W^T X = \sum_{i=1}^{D} w_i x_i$ e a função sinal classifica o valor de $W^T X$, conforme a Equação (2.43)

$$\hat{y} = \operatorname{sign} \sum_{i=1}^{D} w_i x_i. \tag{2.43}$$

A função sign retorna um valor -1 ou $+1$, denotando uma classificação binária. O erro E da predição pode ser definido por $E(x) = y - \hat{y}$, que, considerando que os valores de y e $\hat{y}$ são -1 ou $+1$, restringe $E(x) \in \{-2, 0, +2\}$. O algoritmo adapta e altera os pesos calculados quando o erro da previsão é diferente de 0, significando que esse previu errado. Isso possibilita a criação de modelos mais poderosos e que consigam realizar previsões mais corretas (AGGARWAL, 2018).

Em vários casos, pode haver um enviesamento[6] do conjunto de dados, representando desbalanceamento. Dessa forma, adiciona-se a variável de viés b nos cálculos realizados pelo algoritmo, resultando na Equação (2.44)

$$\hat{y} = \operatorname{sign} \sum_{i=1}^{D} w_i x_i + b. \tag{2.44}$$

O viés é incorporado ao algoritmo através de uma ponderação da conexão utilizando um neurônio de viés. Esse neurônio sempre transmite um valor ao neurônio responsável pelo resultado.

Assim como em outros algoritmos de aprendizado de máquina, o objetivo do *perceptron* é a minimização do erro das previsões. Definindo $\mathcal{D} = \{(X, y) \mid X \in L, y \in \{-1, +1\}\}$ e considerando o cálculo do erro como de quadrados mínimos, o função objetivo do algoritmo pode ser definida pela Equação (2.45)

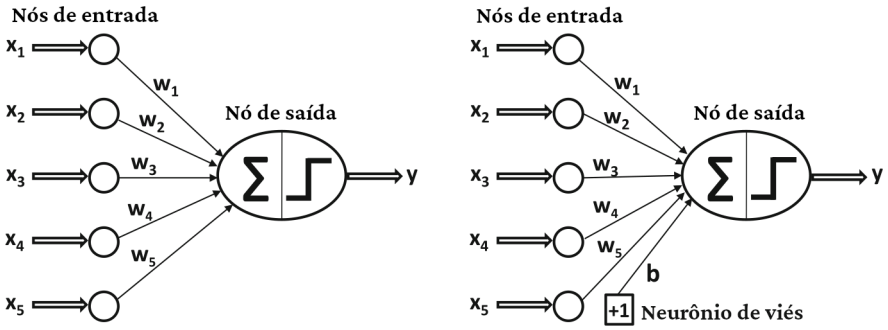
$$\min_{W \in \mathbb{R}^D} L = \sum_{(X,y) \in \mathcal{D}} \left(y - \operatorname{sign} W^T X\right)^2. \tag{2.45}$$

A função L é chamada de *função de perda* (do Inglês, *loss function*). O papel da função de perda numa rede neural, melhor detalhado no Apêndice B, é o de calcular uma relação de distância entre os dados reais y e as aproximações $\hat{y}$. Posteriormente ela será minimizada por um algoritmo

[6] Desvio na média das amostras, resultando em imprecisões nas previsões.

conveniente, que em geral se utiliza de seu gradiente. A arquitetura de um *perceptron* pode ser vista na Figura 19.

Figura 19 – Esquema de uma rede neural do tipo *perceptron*. Em ambas as figuras pode-se observar a camada de entrada com cinco nós, representando cinco características $x_1, \dots, x_5$. Esses neurônios são diretamente conectados à camada de saída de forma ponderada, com os pesos $w_1, \dots, w_5$. O neurônio de saída faz a combinação linear de entradas e pesos, e aplicando por fim a função sign. À direita tem-se o mesmo *perceptron* com o neurônio de viés.



Fonte: Adaptado de Aggarwal (2018).

Numa rede neural, pontua-se as funções de ativação como essenciais e presentes em todos os seus neurônios, apresentadas no Apêndice C.

2.2.3 Redes Neurais tipo *Deep Feed Forward* – DFF

Segundo Ojha, Abraham e Snášel (2017), as redes neurais *Deep Feed Forward*, de tradução livre "avanço profundo", têm sido particularmente bem sucedidas em processamento de grandes conjuntos de dados. Esse tipo de rede neural é formado por várias camadas conectadas entre si, ou seja, sua estrutura é a junção de vários *perceptrons*. Essas camadas contêm neurônios que utilizam funções de ativação não lineares para processar os dados, além de serem capazes de modelar padrões complexos e são altamente usadas em diversas aplicações de inteligência artificial. Uma das vantagens do uso desse tipo de rede neural é a obtenção de mais acurácia e eficiência em diversas tarefas, no sentido de obter melhores resultados, como reconhecimento de imagens e processamento de linguagem. Em contraponto, as desvantagens são a dificuldade de treinar o algoritmo corretamente e, potencialmente, a ocorrência de resultados enviesados, podendo ocorrer o fenômeno de sobreajuste do modelo. A Figura 18 representa a estrutura básica de uma rede neural do tipo *Deep Feed Forward*. Esse tipo de rede neural é totalmente conectada, ou seja, cada neurônio de uma camada é conectado com todos os neurônios da próxima camada.

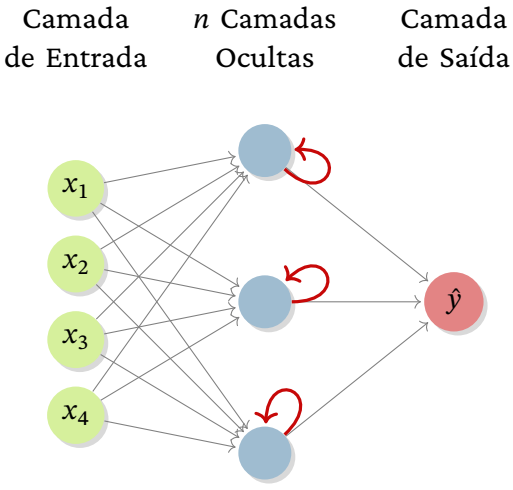
2.2.4 *Recurrent Neural Network* – RNN

As Redes Neurais Recorrentes, do Inglês *Recurrent Neural Networks*, são um tipo específico de rede neural que possuem conexões recorrentes, ou seja, adequadas para dados em sequência, como séries temporais. Diferentemente das DFF, essas redes possuem conexões cíclicas, em que as

informações persistem e são reutilizadas, permitindo que o algoritmo mantenha memória de entradas anteriores. Esse tipo de rede neural é comumente utilizado em tarefas como processamento de linguagem natural (PLN), séries temporais, reconhecimento de voz, previsão de preços em mercado financeiro ou de criptoativos. Nota-se que os exemplos têm uma dependência temporal dos dados, substancial nesse modelo de redes neurais.

Conforme Silva, Spatti e Flauzino (2016), a capacidade de utilizar a informação da saída na próxima etapa de entrada é fundamental nas RNN, que significa que a rede possui informações não somente da entrada atual, mas também do que já foi aprendido em etapas anteriores. Nas camadas intermediárias desse tipo de rede neural, os dados sequenciais são mantidos para futuras previsões, executando tarefas não executadas pelo tipo DFF, ou seja, cada novo dado de entrada influencia a si próprio e aos outros dados que entrarão no modelo. Na Figura 20, exemplifica-se um modelo de rede neural recorrente na qual é possível visualizar que os dados são cíclicos, ou seja, retornam das saídas às camadas intermediárias para reconsideração no modelo.

Figura 20 – Esquema de uma rede beural do tipo *Recurrent Neural Network*. Os neurônios das camadas ocultas, representados em roxo, possuem uma conexão extra (todas destacadas em vermelho), realimentando a saída de volta ao próprio neurônio, daí o nome "recorrente".



Fonte: Adaptado de Academy (2022).

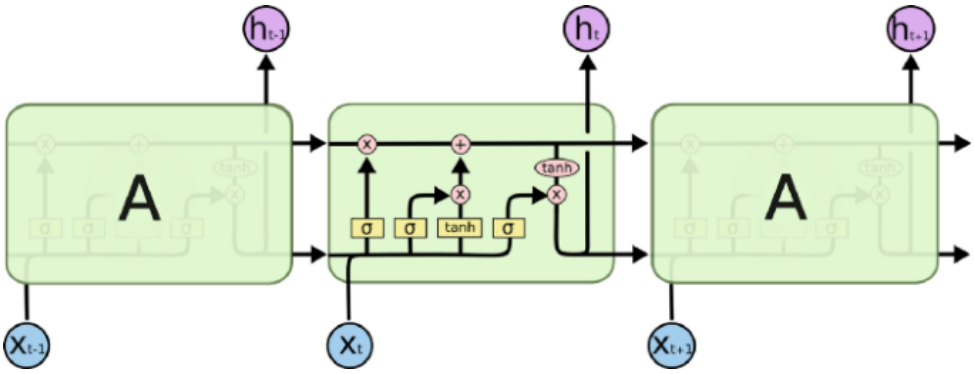
Usando um exemplo do ramo de processamento de linguagem, retirado de Aggarwal (2018), demonstra-se o processo do algoritmo em processar palavras do neurônio de entrada e desenvolver uma frase, que é o objetivo desse. Nota-se na Figura 22 da próxima seção, que os dados retornam a uma mesma camada intermediária, demonstrando a parte recorrente do modelo.

Nesse tipo de rede neural, em contrapartida, os pesos de cada análise, são progressivamente perdidos ao longo do tempo, atribuindo maior relevância a entradas recentes. O tipo de rede neural *Long Short-Term Memory* procura resolver esse problema, minimizando a dependência ao tempo a longo prazo.

2.2.5 *Long Short-Term Memory* – LSTM

Long Short-Term Memory (LSTM) é uma arquitetura sofisticada e derivada da rede neural recorrente, uma vez que permite que as informações persistam, isto é, sejam levadas em consideração para futuras previsões e remodelagem do algoritmo. Esse tipo de algoritmo é utilizado para classificação e previsão de séries temporais com intervalo de tempo indeterminado (HOCHREITER; SCHMIDHUBER, 1997). A estrutura básica de um elemento componente de uma rede neural do tipo LSTM pode ser verificada na Figura 21. Nela, o neurônio A tem como entrada a variável X_t e saída h_t, ambas dependentes do tempo, e é realimentado. Internamente a esse neurônio, tudo se passa como se a sequência temporal das variáveis X_t e h_t fossem alimentadas uma a uma em um neurônio do tipo A (conexões verticais), ao passo que cada neurônio se comunica com seu vizinho na escala temporal (conexões horizontais) (SILVA; SPATTI; FLAUZINO, 2016).

Figura 21 – Esquema de uma rede neural do tipo *Long Short-Term Memory*. O neurônio A tem como entrada X_t e saída h_t, ambas dependentes do tempo, e é realimentado. A sequência temporal das variáveis X_t e h_t pode ser desmembrada uma a uma em neurônios do tipo A (conexões verticais), ao passo que cada um deles se comunica com seu vizinho na escala temporal (conexões horizontais).



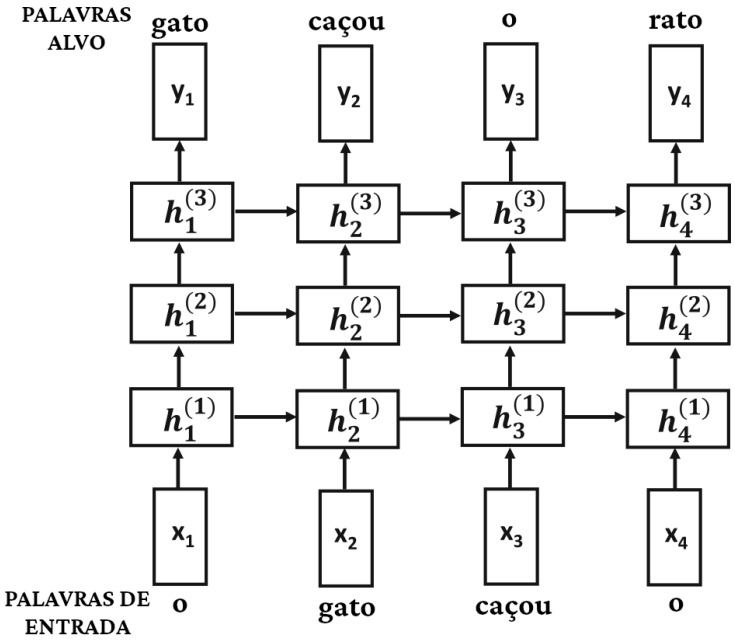
Fonte: Academy (2022).

Nessas arquiteturas complexas, a estrutura de células de memória permite o fluxo controlado de informações devido a presença dos portões (do Inglês, *gates*), que são estruturas da rede que modulam o fluxo de informações com base nas novas entradas e nos estados anteriores da célula. Para Xiaoyun et al. (2016), esse tipo de algoritmo de rede neural vem mostrando resultados promissores em tarefas como modelagem de linguagem[7], reconhecimento de discurso e processamento de linguagem natural. Isso deve-se ao fato de o algoritmo ser eficaz em modelar variáveis dependentes a longo prazo. Ademais, esse algoritmo é capaz de receber uma grande quantidade de dados, parâmetros e camadas no modelo, sendo assim mais robusto para tarefas complexas e, em contrapartida, mais custoso computacionalmente de ser treinado (SAINATH et al., 2015).

A notação $h_t^{(k)}$ representa o estado da k-ésima camada, de dimensão p, no instante único t e x_t o vetor de entrada, com dimensão D. Dessa forma, pode-se assumir que $h_0^{(k)}$ representa x_t, uma vez

[7] Define uma série de características para ajudar na modelagem e documentação de sistemas desenvolvidos usando orientação a objetos.

Figura 22 – Esquema de uma rede neural LSTM ilustrando a dependência sequencial das entradas (conexões horizontais).



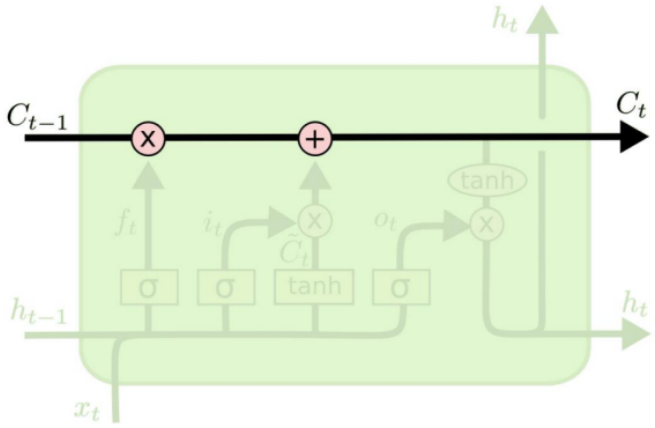
Fonte: Aggarwal (2018).

que é o primeiro instante em que os dados estão no algoritmo. Na Figura 22, é possível ver como os estados das camadas ocultas $h_t^{(k)}$ são propagados na rede neural, reafirmando que a rede do tipo LSTM herda as características de uma rede neural recorrente, conforme Seção 2.2.4 (AGGARWAL, 2018). Com o objetivo de alterar as condições de recorrência de uma RNN, adiciona-se um vetor $c_t^{(k)}$, também de dimensão p, que representa o *estado da célula*. O estado da célula é implementado em termos computacionais como uma memória interna. Este estado é simbolizado por meio de um vetor contendo diversas variáveis, capaz de representar integralmente a situação atual do sistema. Mais ainda, conhecendo o estado presente do sistema, a sua dinâmica e as entradas que influenciam sobre ele daqui em diante, o comportamento futuro do sistema não depende de suas ações passadas. Isso porque todas as informações necessárias estão contidas nas variáveis de estado. Assim, por definição, essas variáveis de estado agregam tudo o que já aconteceu no passado do sistema. Além disso, como a LSTM se remodela conforme o processamento dos dados, a rede neural é capaz de mudar as dependências através de padrões reconhecidos por essa, alterando o estado da célula conforme o tempo. Assim, os componentes principais da célula de memória desse tipo de rede neural são o estado da célula C_t, que permite a persistência da informação, e o estado oculto h_t, que contém a saída da célula, ambos no instante de tempo t. A Figura 23 demonstra o vetor de estados presente na arquitetura de uma rede LSTM.

Ademais, exemplifica-se na Figura 24, o estado da célula "preço da casa" armazena as variáveis temporais que definem a dinâmica do preço de um imóvel.

Após passar pelos vetores de estados no bloco LSTM, as informações passam por operações algébricas, removendo ou adicionando informação nesse vetor. Essas operações são definidas através de três portões, definidos durante o processo de treinamento. Nas camadas intermediárias

Figura 23 – Vetor de estados na arquitetura LSTM.



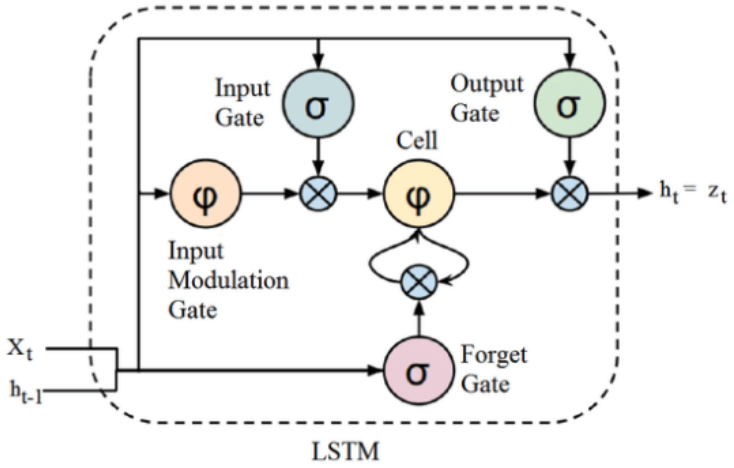
Fonte: Academy (2022).

Figura 24 – Exemplo de variáveis temporais que definem a célula do preço de um imóvel.

Fonte: Fonte autoral.

da LSTM, há uma estrutura interna em cadeia, contendo portões e células, que são os blocos de memória da rede.

Figura 25 – Estrutura em cadeia das redes neurais do tipo LSTM.



Fonte: Academy (2022).

Os portões (do Inglês, *gates*) têm funções diversas (Figura 25), a saber:

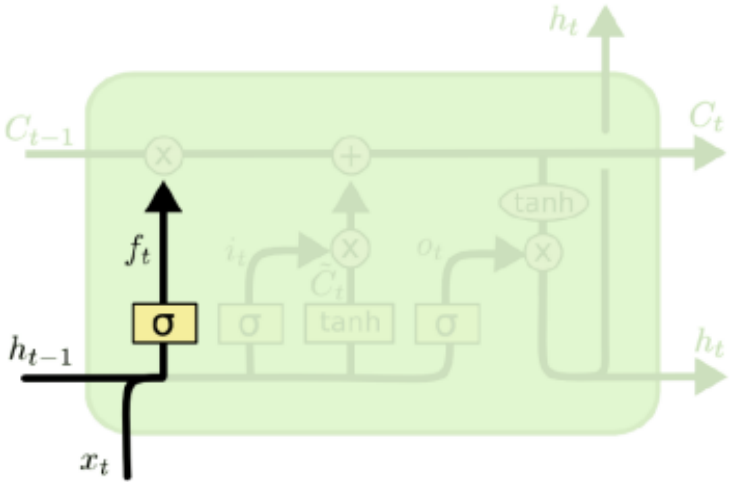
Forget Gate de tradução livre, o "portão de esquecimento" é responsável por filtrar e descartar

informações que não são mais úteis nas células. Os dados de entrada são processados nesse portão e classificados binariamente através de uma função de ativação, sendo 1 para dados a serem mantidos e 0 para os descartáveis. A função recebe duas entradas, x_t e h_{t-1}, conforme a Figura 26, processa-as e determina sua manutenção no algoritmo.

Esse é o primeiro portão em que as informações passam e são multiplicadas por um valor entre 0 e 1 através de uma função sigmoide, ponderando a relevância desses dados pro algoritmo, conforme a Equação (2.46) e Figura 26 (OLAH, 2015).

$$F_t = \sigma\big(W_f(h_{t-1} \cdot x_t) + b_f\big). \tag{2.46}$$

Figura 26 – Portão de esquecimento nas redes do tipo LSTM.



Fonte: Academy (2022).

Input Gate de tradução livre, o "portão de entrada" é onde ocorre a adição de informações que passaram pelo portão anterior, ou seja, que são julgadas úteis para o algoritmo. Primeiramente, o estado atual x_t e o estado da camada anterior h_{t-1} passam por uma função sigmoide, retornando um valor entre 0 e 1, resultando no novo estado atual c_t. Posteriormente, a mesma informação do estado oculto anterior e atual passam pela função tangent hiperbólica (tanh), criando-se um vetor C_t com valores entre -1 e 1. A saída desse portão será gerada através da função de ativação com esses valores.

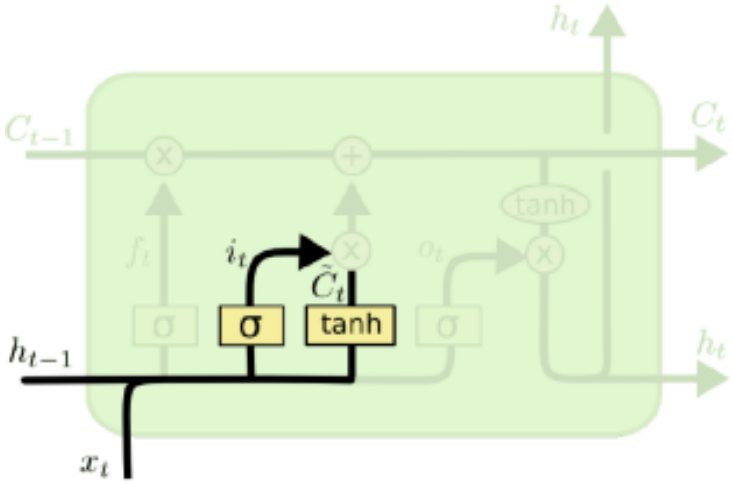
Na próxima etapa, o *input gate* irá verificar quais dados serão armazenados no estado da célula. Há duas camadas nessa etapa, sendo a primeira equivalente a (2.46), conforme (2.47), em que o algoritmo decide quais dados serão atualizados pela função de ativação C_t, correspondente à segunda camada, que podem variar conforme definidas na Subseção C (OLAH, 2015). Além disso, nessa etapa considera-se as informações do estado do portão anterior, ou seja, em $t-1$ para atualizar e passar essas informações processadas, somando o produto desse pelo resultado do quanto manter (*forget gate*) e dos dados atuais pelo impacto deles

(*input gate*), resultando na Equação (2.48) e Figura 27 (OLAH, 2015).

$$I_t = \sigma\big(W_i(h_{t-1} \cdot x_t) + b_i\big), \tag{2.47}$$

$$C_t = C_{t-1} f_t + \vec{C}_t i_t. \tag{2.48}$$

Figura 27 – Portão de *input* no LSTM.

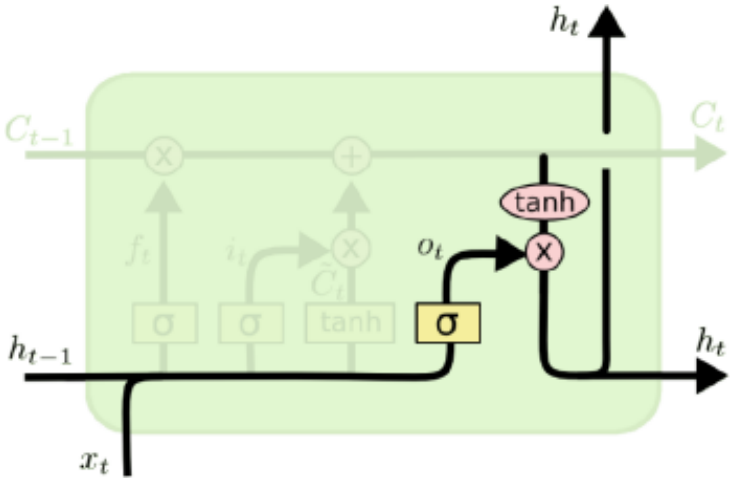


Fonte: Academy (2022).

Output Gate de tradução livre, o "portão de saída" tem como objetivo extrair informações relevantes do estado da célula para retornar uma saída. Ele será responsável por definir o valor do próximo estado oculto. Novamente, x_t e h_{t-1} passam por uma função sigmoide e o estado atual pela tanh. Esses valores são multiplicados e, baseado nestes, a rede decide quais informações essa camada oculta irá manter. Esse portão pode ser denotado pela Equação (2.49) e Figura 28 (OLAH, 2015).

$$\begin{aligned} O_t &= \sigma\big(W_i(h_{t-1} \cdot x_t) + b_o\big), \\ h_t &= O_t * \tanh C_t. \end{aligned} \tag{2.49}$$

Figura 28 – Portão de *output* no LSTM.



Fonte: Academy (2022).

De toda forma, segundo Hochreiter e Schmidhuber (1997), o ajuste e escolha dos parâmetros em redes neurais recorrentes, como no caso de LSTM, é bem desafiador.

Com os algoritmos apresentados, recapitula-se que, neste trabalho, serão utilizados os algoritmos: SVM para Regressão e LSTM.

3 Metodologia

3.1 Pacotes utilizados

Os pacotes em linguagem Python que foram utilizados durante a execução deste trabalho podem ser conferidos na Tabela 3. As funções de cada um serão explicitadas ao longo desta metodologia.

Tabela 3 – Pacotes Python utilizados ao longo da execução deste trabalho.

PACOTE	FUNCIONALIDADE
`math`	cálculo das métricas
`datetime`	padronização das datas para análises e previsões
`warnings`	desconsidera avisos desnecessários durante a execução
`yfinance`	obtenção dos valores históricos das criptomoedas
`numpy`	biblioteca numérica matricial
`pandas`	coleta e análise de dados tabulares
`keras`	construção das rede neurais
`tensorflow`	treinamento e previsão com rede neurais
`matplotlib`	gráficos
`MinMaxScaler` e `RobustScaler`	normalização dos dados

3.2 Coleta dos dados

A coleta dos dados das criptomoedas se dá com o pacote `yfinance`, que permite baixá-los diretamente do *Yahoo Finance*[1], sendo necessária apenas a definição de uma data de início da coleta. Os dados serão analisados com a coleta de preços de 2 anos antes de Junho de 2023 e em seu periodo máximo (série histórica completa), com intervalo de 1 dia entre medidas.

Por meio da função `DataReader` do `pandas_datareader` é possível coletar os dados e armazená-los em forma tabular no padrão definido pelo `pandas` (conhecido como *dataframe*, do Inglês). Na Figura 29, é possível ver as características armazenadas do histórico da criptomoeda XRP desde a sua criação, em 2015. Os dados coletados, em dólares, são os seguintes (o correspondente em Inglês, na Figura 29, está entre parênteses): valor de abertura (*Open*), valor de alta (*High*), valor de baixa (*Low*), valor final do dia (*Close*), volume (*Volume*). Apesar das diferentes características coletadas para as criptomoedas, será utilizado apenas o preço de fechamento para o treinamento e previsão do algoritmo.

[1] https://finance.yahoo.com/

Figura 29 – Exemplo de organização dos preços no histórico completo da criptomoeda XRP, em forma tabular.

Date	Open	High	Low	Close	Volume
2017-11-09 00:00:00+00:00	0.217911	0.221791	0.214866	0.217488	147916992
2017-11-10 00:00:00+00:00	0.218256	0.219068	0.205260	0.206483	141032992
2017-11-11 00:00:00+00:00	0.205948	0.214456	0.205459	0.210430	134503008
2017-11-12 00:00:00+00:00	0.210214	0.210214	0.195389	0.197339	251175008
2017-11-13 00:00:00+00:00	0.197472	0.204081	0.197456	0.203442	132567000
...	...	...	...	...	...
2023-03-12 00:00:00+00:00	0.366827	0.373318	0.352473	0.373318	1102164948
2023-03-13 00:00:00+00:00	0.373354	0.381001	0.358142	0.373786	1694593960
2023-03-14 00:00:00+00:00	0.373790	0.387479	0.367318	0.374279	1527899940
2023-03-15 00:00:00+00:00	0.374326	0.377221	0.358760	0.360558	1062482118
2023-03-16 00:00:00+00:00	0.360623	0.366620	0.358547	0.363276	896554880

[1954 rows x 5 columns]

Fonte: Feito pelo autor.

3.3 Pré-processamento

Uma vez que os dados coletados apresentam valores reais de mercado, não será necessária a correção de valores nulos ou fora de padrão (o que é sempre recomendado e foi feito neste trabalho, por precaução). Os seguintes procedimentos são executados nos dados originais:

Divisão dos conjuntos Os dados são divididos em conjuntos de treinamento e de teste, numa proporção de 0,75 e 0,25, respectivamente.

Conversão das datas As datas devem ser convertidas em valores inteiros sequenciais através do pacote `datetime`, também usado na construção de gráficos. Isso é feito para que seja possível utilizar as datas como parâmetros nos algoritmos já que estes não as aceitam no formato usual.

Normalização As funções `MinMaxScaler` e `RobustScaler` do pacote Scikit-Learn são usadas para normalizar os dados para que fiquem no intervalo $[0, 1]$. As equações que definem essas normalizações são, respectivamente, (3.1) e (3.2):

$$X_{\text{norm}} = \frac{X_i - \min(X)}{\max(X) - \min(X)}, \tag{3.1}$$

$$X_{\text{norm}} = \frac{X_i - \text{mediana}(X)}{Q_3(X) - Q_1(X)}. \tag{3.2}$$

3.4 Parâmetros

Os parâmetros usados nas simulações constam da Tabela 4.

Tabela 4 – Parâmetros utilizados nos algoritmos SVM e LSTM.

PARÂMETRO	SVM	LSTM
Escaladores	`RobustScaler` e `MinMaxScaler`	`RobustScaler` e `MinMaxScaler`
Fração para treinamento	0,75	0,75
Função de perda	Erro quadrático médio Erro máximo Erro quadrático absoluto R^2	Erro quadrático médio Erro máximo Erro quadrático absoluto R^2
Janela (em dias)	5, 15, 30, 60 e 90	5, 15, 30, 60 e 90
Número de camadas	Não se aplica	24
dropout	Não se aplica	0,2
C	0,1; 1 e 100	Não se aplica
Kernel	RBF	Não se aplica

Fonte: Feito pelo autor.

3.5 *Dropout*

Dropout, de tradução direta "jogar fora", é uma técnica utilizada em redes neurais para regularização dos dados, na tentativa de minimizar a ocorrência do fenômeno de sobreajuste. Do Inglês *overfitting*, sobreajuste é quando o algoritmo funciona tão bem que se torna capaz de prever apenas os dados de treinamento, errando grosseiramente em dados inéditos. A técnica prediz o quanto cada derivada imputada deve influenciar para que a função de perda seja reduzida. Dessa forma, as unidades podem ser alteradas de forma a consertar os erros causados por outras. Basicamente, alguns neurônios nas camadas podem ser desconsiderados para que outros ativos aprendam melhor, reduzindo o erro total.

3.6 Criação da rede neural

No caso do algoritmo LSTM, são adicionadas duas camadas intermediárias com 24 neurônios cada, com a função de ativação ReLU, tornando o algoritmo caro computacionalmente, mas potencialmente mais adequado para previsões temporais.

Na implementação da rede neural, as camadas são criadas com a biblioteca `keras`.

3.7 Ciclo de treinamento e avaliação

Finalmente chega-se às fases de treinamento, previsão e geração de resultados:

Treinamento do modelo Após a criação da rede neural, no caso de LSTM, ou da divisão em treinamento e teste, no caso do SVM para regressão (SVR), o algoritmo é treinado com a função `fit`. No treinamento do SVR, é possível utilizar de forma automática uma combinação de parâmetros a serem testados a fim de selecionar os que produzem melhores resultados.

Convergência Nas redes neurais, é possível calcular e construir o gráfico da função de perda ao longo das iterações. Para esse caso, constrói-se o gráfico de `epochs` × `loss`, que dá uma ideia de quão o algoritmo converge ao longo do treinamento da rede neural.

Realização da previsão Após o treinamento os algoritmos, através da função `predict`, prevê-se os valores no conjunto de testes. Os resultados previstos são colocados em um gráfico junto dos valores reais para fins de comparação e avaliação qualitativa.

Métricas de avaliação Após a previsão, as métricas de avaliação (Tabela 4) quantificam a qualidade do algoritmo com os diferentes parâmetros utilizados. Assim, é possível analisar as previsões de um ponto de vista quantitativo.

Geração dos gráficos Por fim, em relação a todas as combinações possíveis de parâmetros, escaladores e normalizadores, constrói-se um gráfico com para comparação entre os valores previstos pelo algoritmo e os valores reais, gerando um arquivo cujo nome codifica os parâmetros utilizados, a qual criptomoeda pertence, métricas, etc.

4 Resultados e discussão

Diante dos conteúdos abordados nesse trabalho, aplicaram-se os algoritmos de *SVR* e *LSTM* para previsão das criptomoedas *Cardano*, *Binance Coin*, *Ethereum*, *Bitcoin* e *Ripple*, com o objetivo de encontrar a melhor combinação de algoritmo e parâmetros para previsão de preços, conforme a metodologia apresentada no Capítulo 3, com intervalo de 1 dia na coleta dos dados. Os resultados para cada período (2 anos e máximo), cada tipo de escalador (`RobustScaler` e `MinMaxScaler`) e janelas (5, 15, 30, 60 e 90 dias) podem ser conferidos na Seção 4.2.1 para o algoritmo SVR e na Seção 4.2.2 para o algoritmo LSTM. Dessa forma, foram analisadas, ao todo, 200 combinações, desconsiderando os testes individuais das camadas das redes neurais, para geração e comparação dos resultados, com um tempo de execução de código de cerca de 10 minutos para o algoritmo *SVR* e 5 horas para *LSTM*.

4.1 Discussões

Diante dos resultados, é possível visualizar e apontar algumas observações quanto à escolha dos parâmetros e do algoritmo num contexto geral.

4.1.1 SVR

Escaladores Em relação aos escaladores utilizados no algoritmo de SVR, nota-se que, para todas as criptomoedas, os resultados do `RobustScaler` foram superiores aos do `MinMaxScaler`, na maioria dos indicadores. Essa pontuação sugere que o primeiro escalador é mais eficaz em considerar a volatilidade e discrepância inerentes aos dados das criptomoedas.

Janela Em relação à janela de tempo utilizada nas previsões (em dias), nota-se que não houve uma relação clara com o desempenho do modelo. Em contraponto, nota-se que, para o período máximo retroativo de coleta dos dados, as janelas menores produziram erros menores, ou seja, foram parâmetros nos quais o algoritmo obteve resultados melhores. Isso pode-se dever ao fato de que, com uma coleta menor, o algoritmo está menos sujeito à volatilidade das criptomoedas em janelas menores.

Período de coleta dos dados Quando analisado o período máximo da coleta dos dados da criptomoedas, ou seja, toda sua variação de preço desde sua criação e não apenas no período de dois anos anteriores à data da simulação, nota-se valores ligeiramente mais consistentes, sobretudo com o escalador `RobustScaler`.

Desempenho entre criptomoedas O desempenho entre as criptomoedas foi variado. Considerando o escalador `RobustScaler`, notou-se que o modelo foi capaz de demonstrar grande

variação conforme os parâmetros utilizados, exceto no caso da XRP, que manteve uma consistência nos valores.

Melhores desempenhos Para todos os casos, notou-se que os parâmetros para os quais o algoritmo desempenhou melhor foram, uma janela de 60 dias, com o escalador `RobustScaler`, no seu período máximo.

4.1.2 LSTM

Escaladores Em contraponto ao analisado nos casos de SVR, nota-se, através da Tabela 6, que houve vários casos em que as previsões de menores erros ocorreram em ambos os escaladores, com `RobustScaler` apresentando uma frequência ligeiramente maior.

Janela No caso das janelas de previsões utilizadas, nota-se que, quanto maior a janela, as previsões apresentam erros menores. Pode-se observar isso ao comparar resultados para diferentes tamanhos de janela e observar que, em várias instâncias, esse erro diminui à medida que o tamanho da janela aumenta.

Período de coleta dos dados Em relação ao período de coleta de dados, observa-se que utilizar o período máximo da coleta dos dados resulta em um desempenho melhor do algoritmo. Isso mostra que um conjunto de dados mais amplo é benéfico ao algoritmo para realização das previsões.

Desempenho entre criptomoedas Comparando as criptomoedas, nota-se que as criptomoedas de maior duração, BTC e ETH, apresentam erros menores, devido ao fato de o conjunto de dados coletados ser maior, assim disponibilizando mais informações ao algoritmo para realizar as previsões. De toda forma, todas as criptomoedas apresentaram erros pequenos nas previsões realizadas.

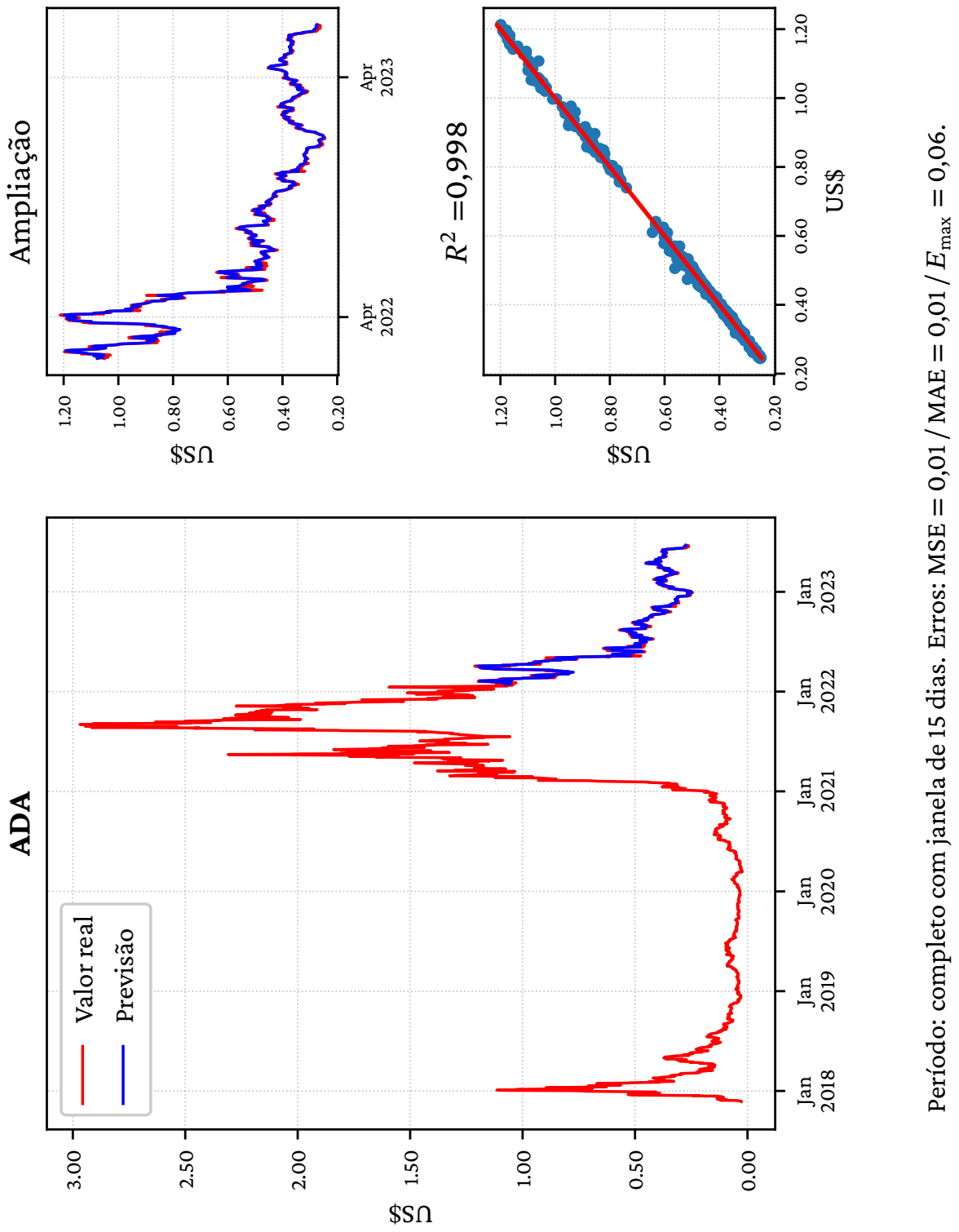
4.2 Gráficos e Tabelas com Resultados

A seguir, serão apresentados os gráficos de melhor desempenho para cada criptomoeda. No primeiro gráfico, é possível visualizar o preço real histórico da moeda e a previsão feita. No segundo, é feita uma ampliação na parte prevista e, no terceiro, uma correlação entre o valor previsto (eixo y) e o valor real (eixo x), ou seja, quanto mais perto os pontos estão da função afim $y = x$, mais próximo R^2 está de 1, ou seja, melhor é a previsão. Na legenda, é possível verificar os parâmetros utilizados e os erros obtidos.

Após todos os gráficos, é possível apreciar nas Tabelas 5 e 6 todas as métricas de avaliação das previsões, separadas por moeda, janela de dias, escalador e período de coleta.

4.2.1 Algoritmo SVR

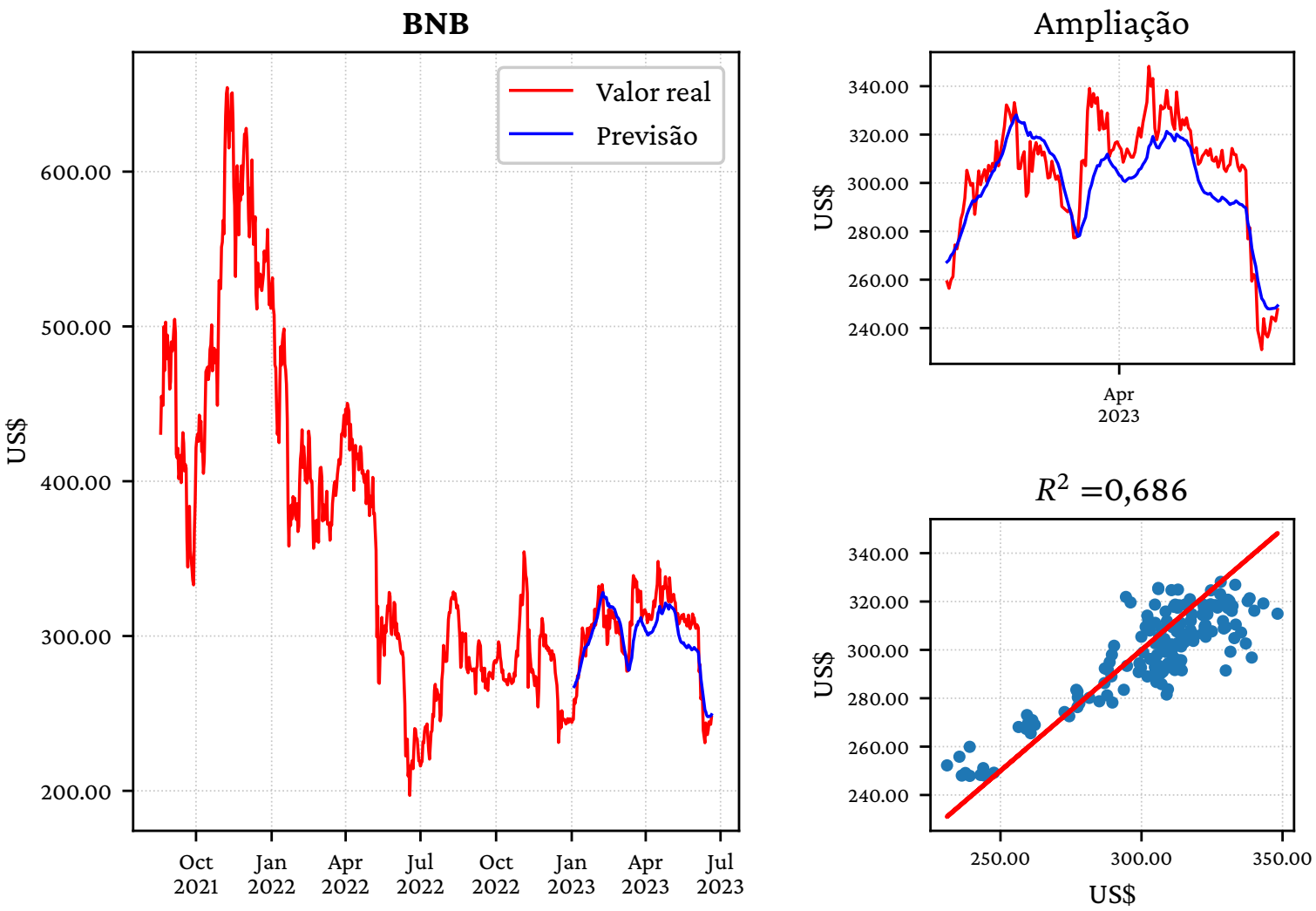
Figura 30 – Previsão SVR com menores erros de ADA.



Período: completo com janela de 15 dias. Erros: MSE = 0,01 / MAE = 0,01 / E_{max} = 0,06.

Fonte: Autoral.

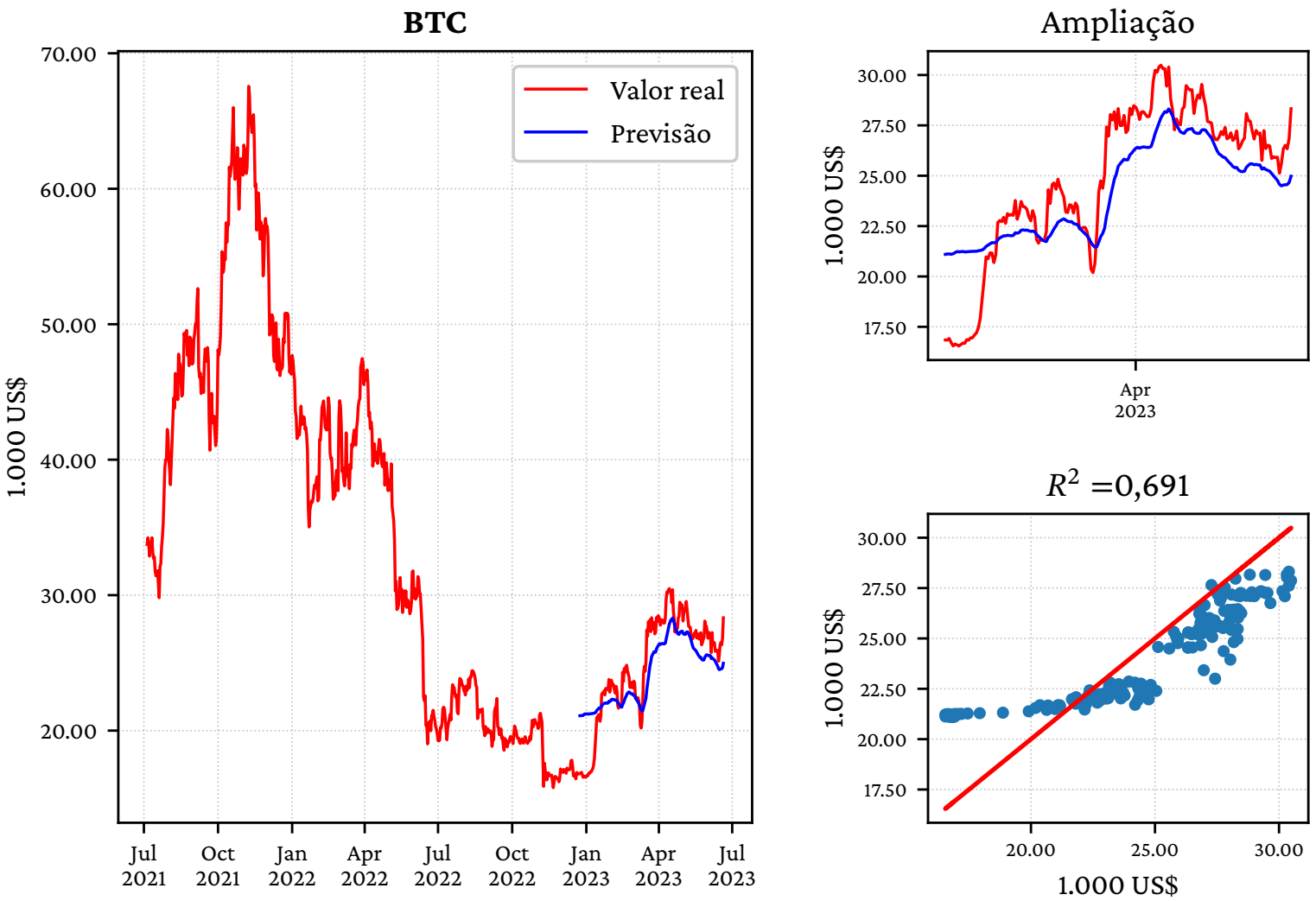
Figura 31 – Previsão SVR com menores erros de BNB.



Período: 2 anos com janela de 60 dias. Erros: MSE = 13,78 / MAE = 11,31 / $E_{\max}$ = 42,25.

Fonte: Autoral.

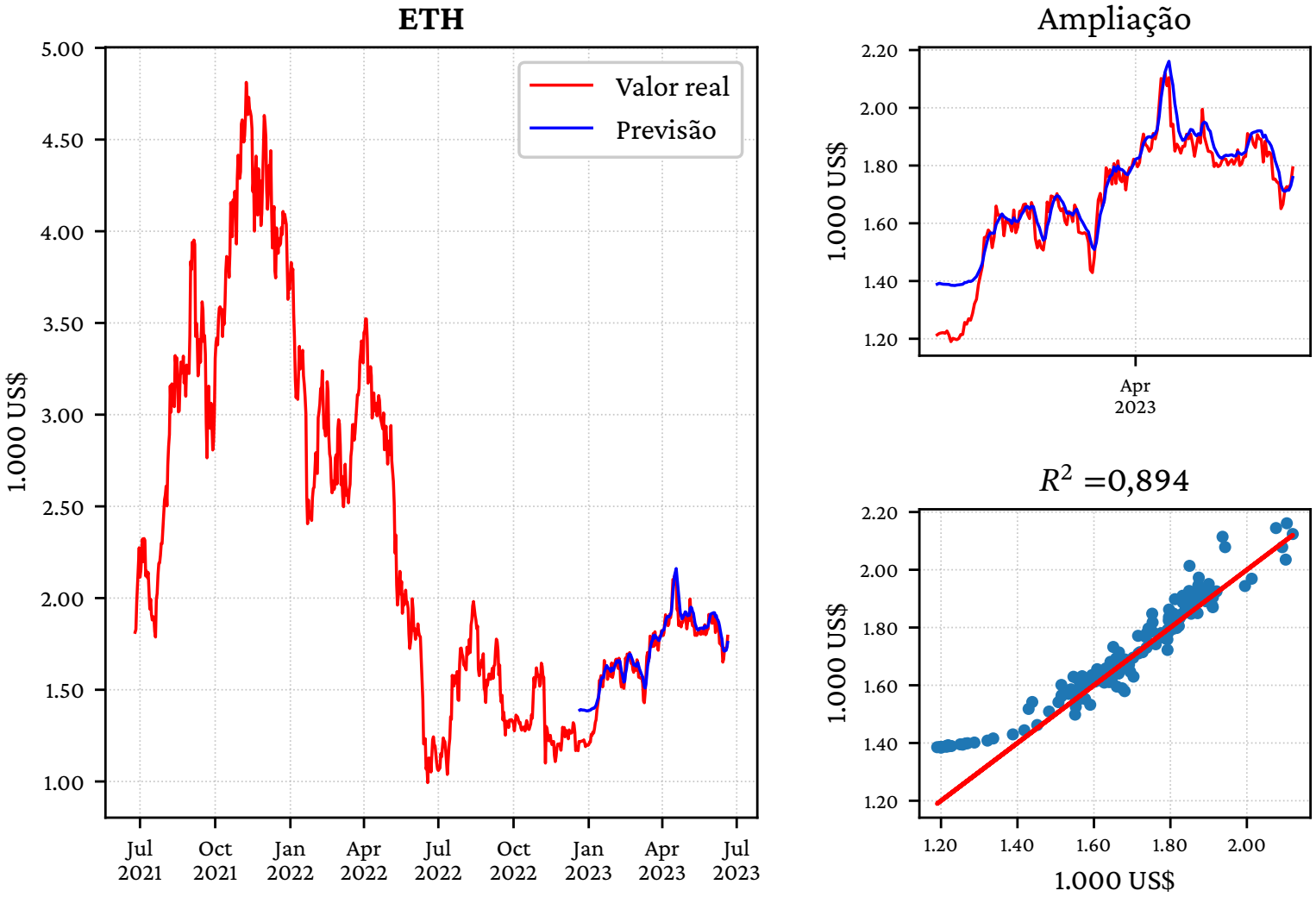
Figura 32 – Previsão SVR com menores erros de BTC.



Período: 2 anos com janela de 15 dias. Erros: MSE = 2095,21 / MAE = 1710,83 / E_{max} = 4671,71.

Fonte: Autoral.

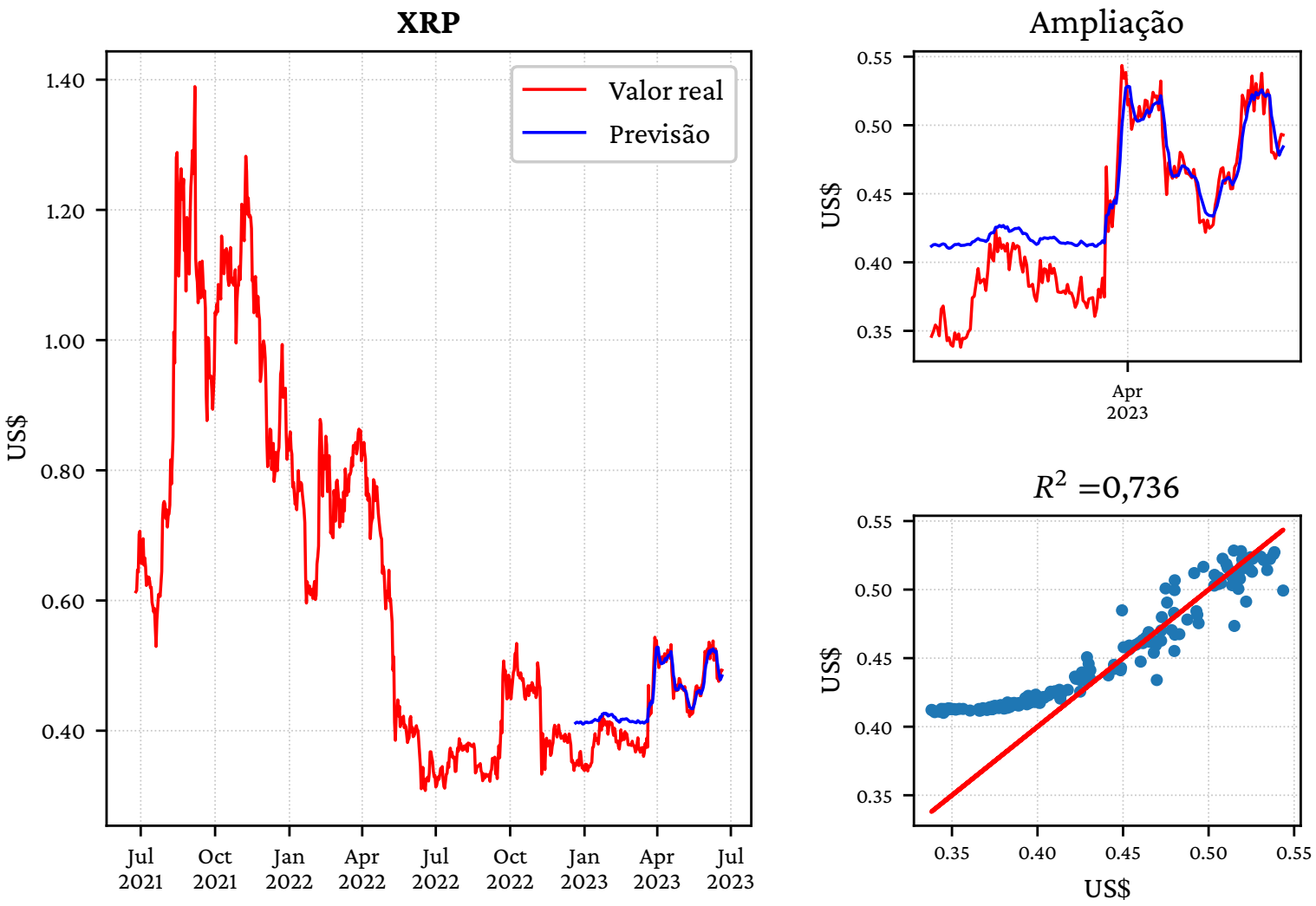
Figura 33 – Previsão SVR com menores erros de ETH.



Período: 2 anos com janela de 5 dias. Erros: MSE = 69,74 / MAE = 49,41 / $E_{\max}$ = 195,44.

Fonte: Autoral.

Figura 34 – Previsão SVR com menores erros de XRP.



Período: 2 anos com janela de 5 dias. Erros: MSE = 0,03 / MAE = 0,02 / E_{max} = 0,07.

Fonte: Autoral.

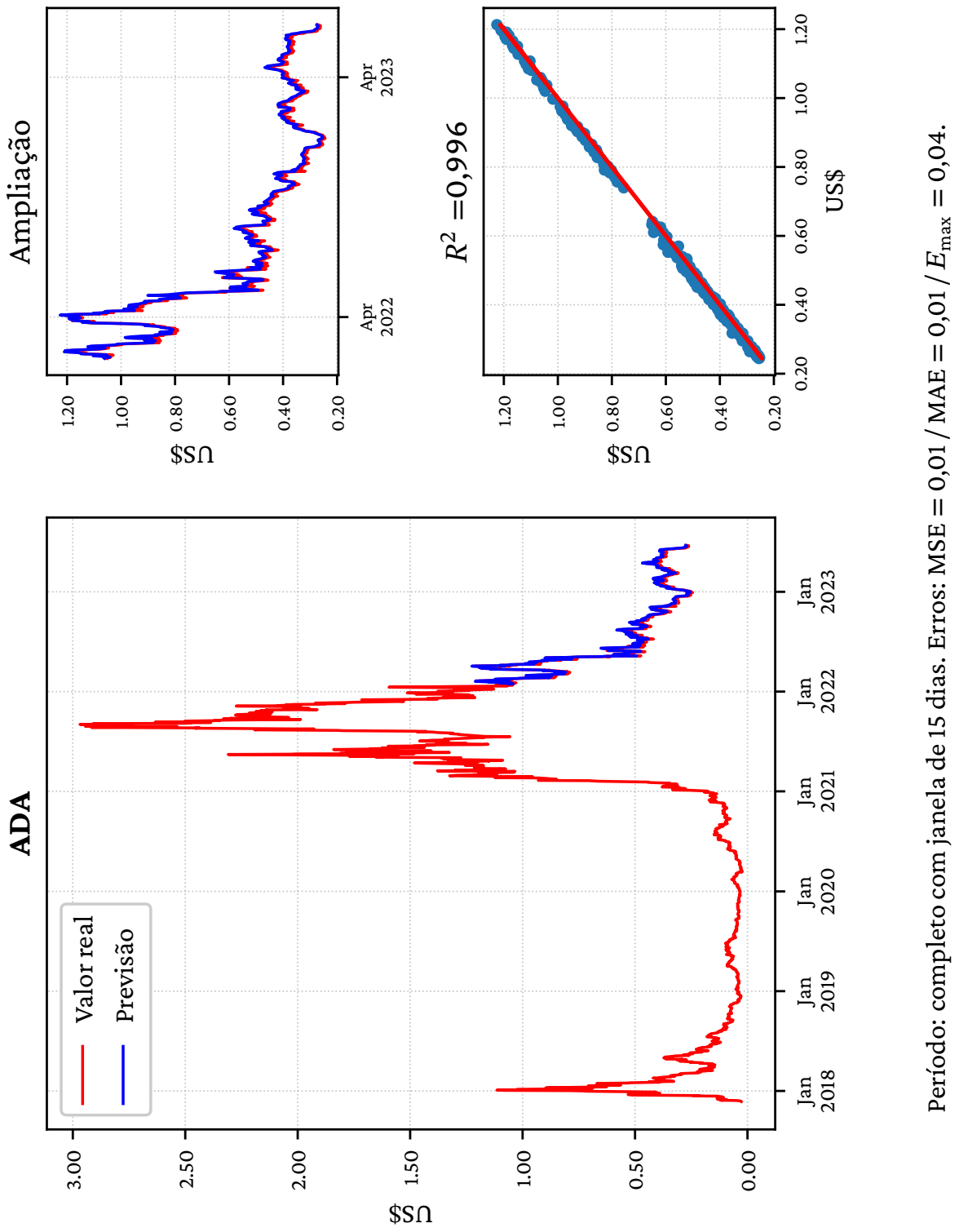
Tabela 5 – Resultados do algoritmo SVR aplicado às moedas ADA, BTC, BNB, ETH e XRP, variando o período de obtenção dos dados (2a – dois anos anteriores ou máx – completo), o tamanho da janela (5, 15, 30, 60 ou 90 dias) e o escalador (rb – `RobustScaler` ou mm – `MinMaxScaler`).

			ADA				BNB				BTC				ETH				XRP			
	Per.	Esc.	MSE	MAE	E_{max}	R^2	MSE	MAE	E_{max}	R^2	MSE	MAE	E_{max}	R^2	MSE	MAE	E_{max}	R^2	MSE	MAE	E_{max}	R^2
5 dias	2a	mm	0,18	0,19	0,28	---	7,86	9,20	25,07	0,896	956,91	1557,69	4414,38	0,838	49,41	69,74	195,44	0,894	0,02	0,03	0,07	0,736
		rb	0,02	0,04	0,10	0,415	5,07	5,33	10,33	**0.965**	1245,41	1308,11	2657,60	0,886	52,98	60,79	139,79	0,919	0,01	0,02	0,04	0,935
	máx	mm	0,10	0,11	0,25	0,817	21,04	25,03	70,08	0,781	3424,92	4514,62	22759,84	0,883	209,62	227,65	602,54	0,858	0,12	0,13	0,25	0,276
		rb	0,01	0,01	0,04	**0.998**	8,06	10,06	33,17	**0.965**	1045,40	2498,99	19314,60	**0.964**	39,23	50,86	245,28	**0.993**	0,03	0,03	0,03	**0.963**
15 dias	2a	mm	0,17	0,18	0,29	---	18,02	19,94	44,80	0,485	1710,83	2095,21	4671,71	0,691	115,65	127,94	272,40	0,618	0,03	0,03	0,08	0,654
		rb	0,03	0,04	0,10	0,326	5,59	6,01	11,74	0,953	1220,30	1313,52	3131,14	0,879	65,52	74,22	175,52	0,872	0,01	0,01	0,05	0,937
	máx	mm	0,11	0,11	0,32	0,782	30,40	33,01	75,90	0,618	3754,40	4512,32	16359,13	0,882	175,93	208,33	678,33	0,881	0,09	0,09	0,24	0,601
		rb	0,01	0,01	0,06	**0.998**	6,06	8,73	37,84	**0.973**	1619,24	2369,36	12901,85	**0.967**	40,93	53,88	223,11	**0.992**	0,03	0,03	0,04	**0.968**
30 dias	2a	mm	0,16	0,17	0,29	---	12,67	14,86	43,97	0,697	1775,84	2213,93	4976,31	0,633	93,95	110,67	265,69	0,692	0,03	0,03	0,08	0,668
		rb	0,02	0,04	0,10	0,409	5,35	6,12	14,76	**0.949**	1388,28	1500,20	3445,77	0,831	73,19	81,82	179,56	0,832	0,01	0,01	0,05	0,936
	máx	mm	0,09	0,11	0,33	0,806	38,59	41,34	90,36	0,398	5057,54	5847,08	16654,96	0,799	188,46	229,31	760,78	0,855	0,08	0,09	0,27	0,630
		rb	0,01	0,02	0,10	**0.995**	17,16	19,71	51,32	0,863	4366,68	4996,65	13592,24	**0.853**	45,37	60,69	224,39	**0.990**	0,02	0,02	0,04	**0.976**
60 dias	2a	mm	0,16	0,16	0,26	---	11,31	13,78	42,25	0,686	2534,89	3005,09	5924,02	0,144	172,64	204,09	455,29	---	0,04	0,05	0,10	0,313
		rb	0,02	0,03	0,07	0,432	8,55	9,75	23,25	**0.843**	1431,83	1772,09	4277,54	0,702	140,66	161,25	296,59	0,113	0,02	0,03	0,08	0,680
	máx	mm	0,11	0,13	0,37	0,677	58,51	64,82	114,01	---	6041,23	6950,57	17482,12	**0.712**	344,63	390,73	1000,88	0,564	0,14	0,15	0,38	---
		rb	0,01	0,02	0,08	**0.993**	23,27	27,79	70,31	0,720	6127,79	6942,96	16704,96	**0.712**	73,32	103,11	398,28	**0.970**	0,02	0,02	0,04	**0.982**
90 dias	2a	mm	0,10	0,11	0,18	---	16,40	19,78	48,20	0,301	3633,70	4301,27	7835,86	---	228,34	278,64	578,32	---	0,05	0,06	0,14	---
		rb	0,03	0,04	0,08	---	9,26	10,62	24,96	**0.799**	3384,02	3604,75	6087,55	---	164,32	182,93	348,91	---	0,03	0,04	0,08	0,577
	máx	mm	0,12	0,14	0,36	0,592	65,93	70,49	126,02	---	7483,13	8629,00	19559,70	0,547	465,80	509,43	1103,06	0,224	0,06	0,09	0,26	0,643
		rb	0,04	0,05	0,15	**0.947**	58,30	63,54	103,27	---	7152,20	8136,33	17170,87	**0.598**	181,75	234,23	662,53	**0.836**	0,02	0,03	0,08	**0.966**

Fonte: Autoral.

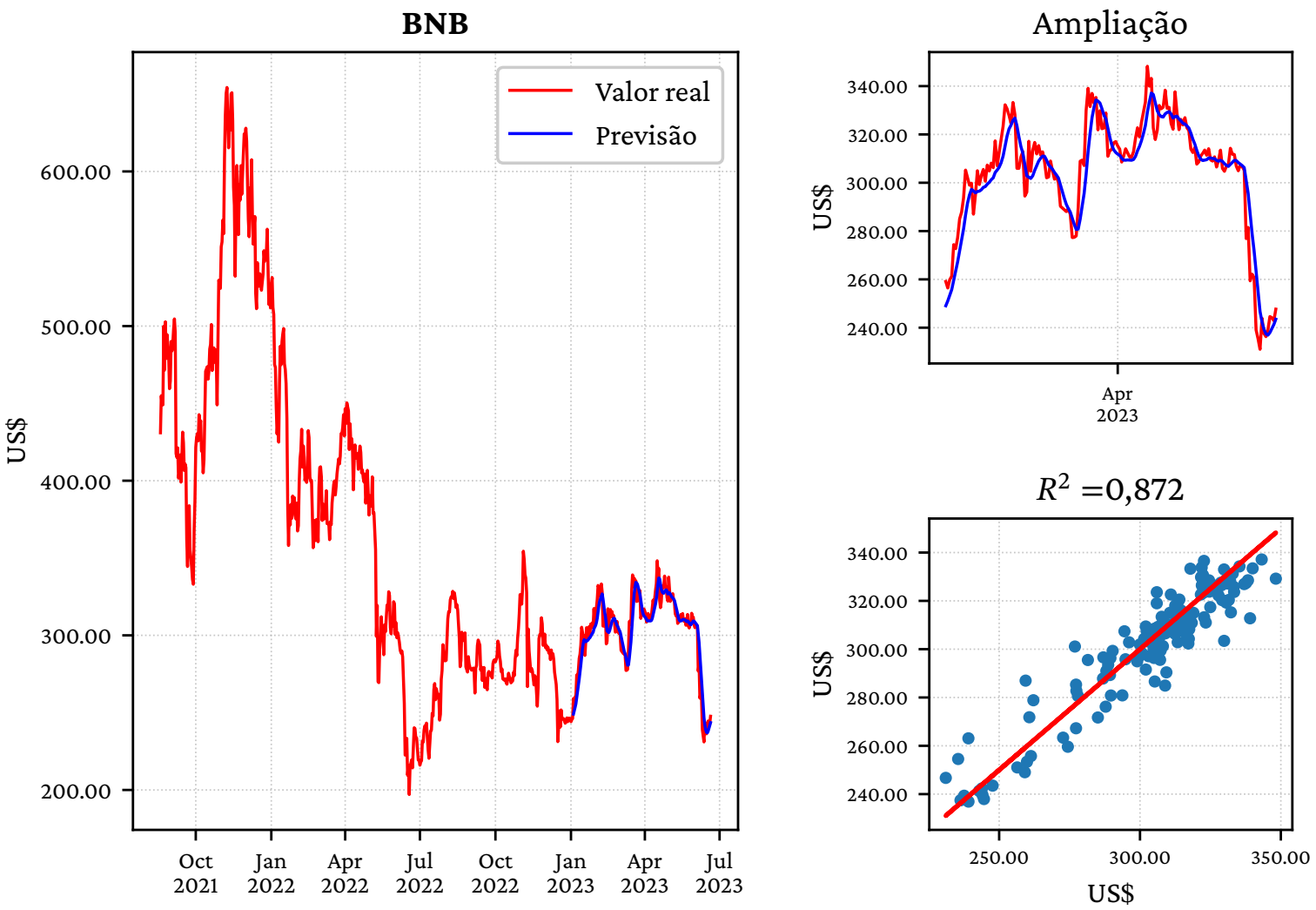
4.2.2 Algoritmo LSTM

Figura 35 – Previsão LSTM com menores erros de ADA.



Fonte: Autoral.

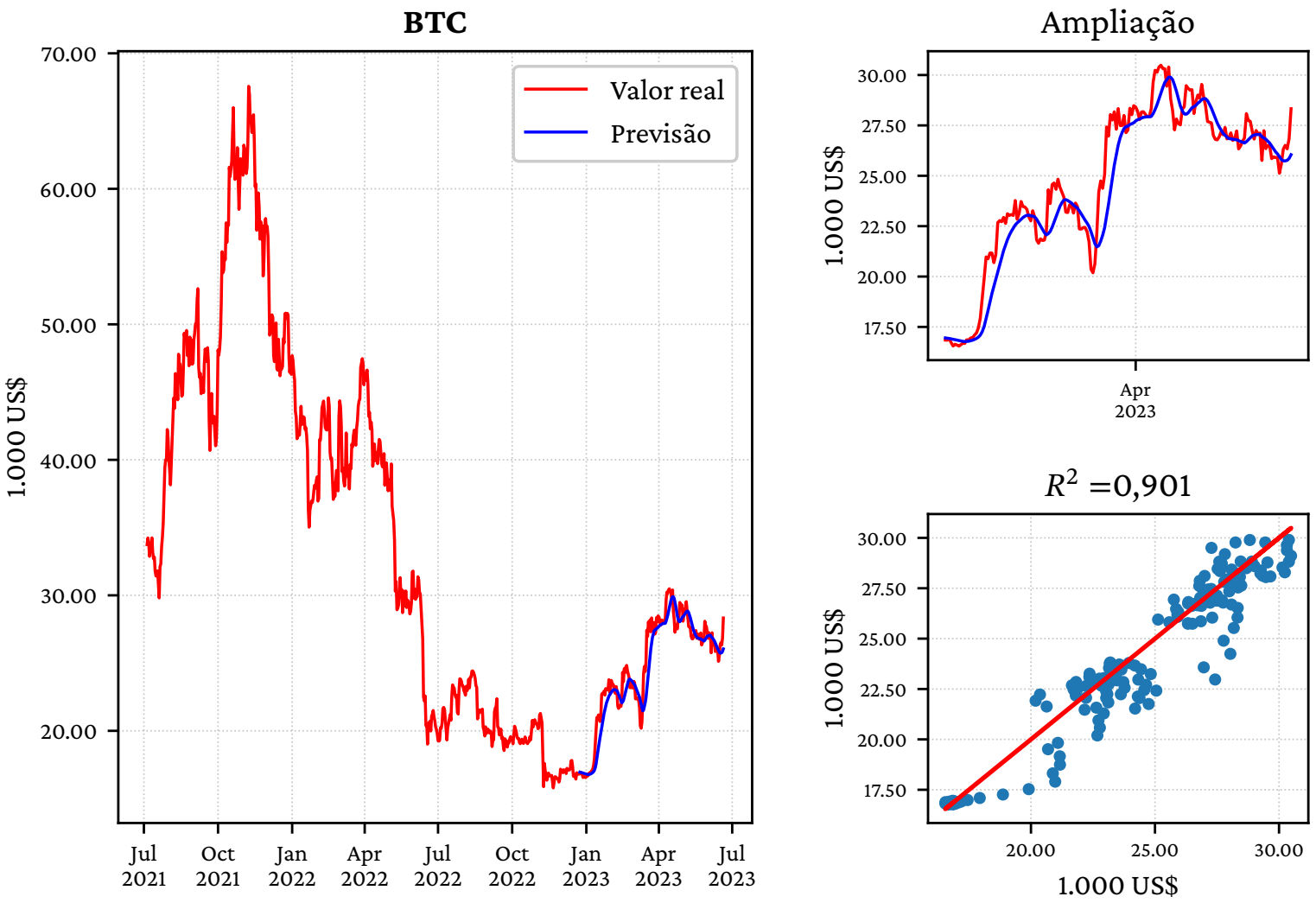
Figura 36 – Previsão LSTM com menores erros de BNB.



Período: 2 anos com janela de 60 dias. Erros: MSE = 8,80 / MAE = 6,52 / $E_{\max}$ = 27,58.

Fonte: Autoral.

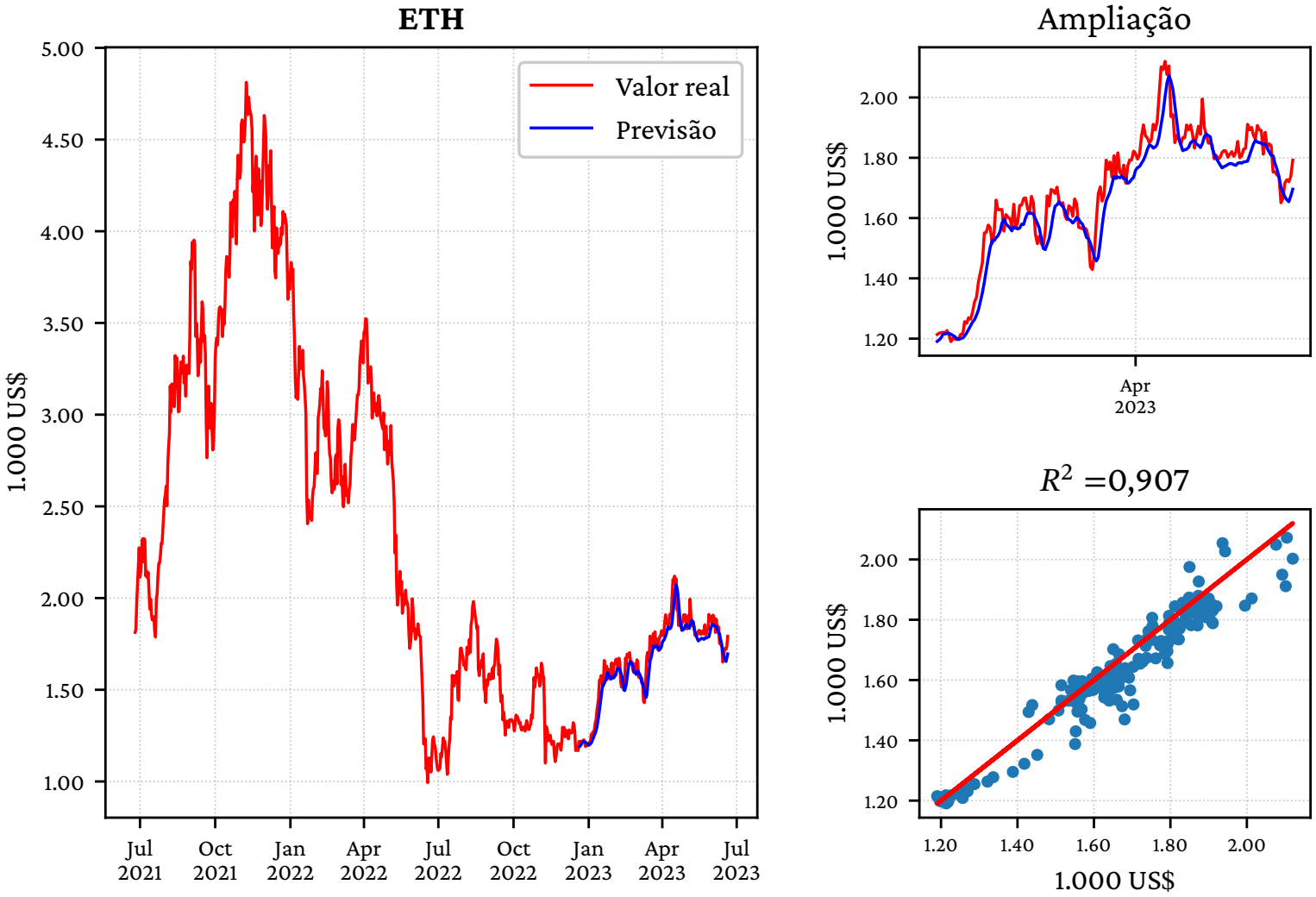
Figura 37 – Previsão LSTM com menores erros de BTC.



Período: 2 anos com janela de 15 dias. Erros: MSE = 1184,22 / MAE = 844,46 / E_{max} = 4451,42.

Fonte: Autoral.

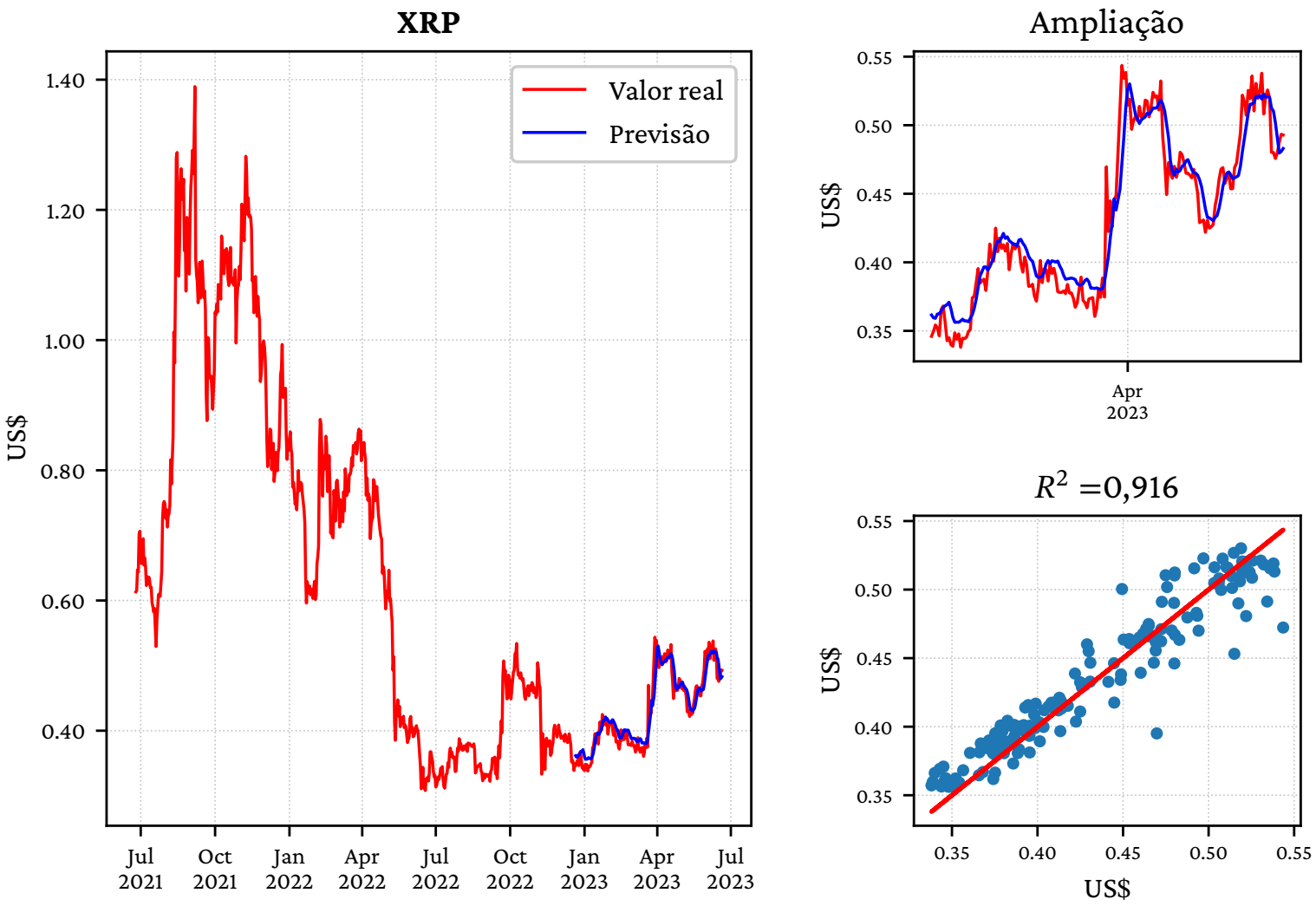
Figura 38 – Previsão LSTM com menores erros de ETH.



Período: 2 anos com janela de 5 dias. Erros: MSE = 65,15 / MAE = 50,37 / $E_{\max}$ = 210,94.

Fonte: Autoral.

Figura 39 – Previsão LSTM com menores erros de XRP.



Período: 2 anos com janela de 5 dias. Erros: MSE = 0,02 / MAE = 0,01 / E_{max} = 0,07.

Fonte: Autoral.

Tabela 6 – Resultados do algoritmo LSTM aplicado às moedas ADA, BTC, BNB, ETH e XRP, variando o período de obtenção dos dados (2a – dois anos anteriores ou máx – completo), o tamanho da janela (5, 15, 30, 60 ou 90 dias) e o escalador (rb – `RobustScaler` ou mm – `MinMaxScaler`).

			ADA				BNB				BTC				ETH				XRP			
	Per.	Esc.	MSE	MAE	E_{max}	R^2	MSE	MAE	E_{max}	R^2	MSE	MAE	E_{max}	R^2	MSE	MAE	E_{max}	R^2	MSE	MAE	E_{max}	R^2
5 dias	2a	mm	0,01	0,02	0,05	0,880	5,84	8,20	28,75	0,917	592,97	858,53	3389,88	0,951	50,37	65,15	210,94	0,907	0,01	0,02	0,07	0,916
		rb	0,01	0,02	0,05	0,895	4,79	6,86	22,98	0,942	650,26	887,60	3199,96	0,947	38,56	51,92	185,98	0,941	0,01	0,01	0,08	0,938
	máx	mm	0,02	0,03	0,16	0,984	8,40	11,51	60,64	0,954	1048,59	1447,76	6156,50	0,988	66,32	91,95	412,92	0,977	0,01	0,02	0,11	0,984
		rb	0,02	0,02	0,08	**0.993**	7,05	9,35	49,56	**0.970**	636,22	964,47	3998,06	**0.995**	29,42	39,82	198,93	**0.996**	0,01	0,01	0,02	**0.997**
15 dias	2a	mm	0,02	0,02	0,08	0,774	6,79	9,55	31,95	0,882	844,46	1184,22	4451,42	0,901	47,26	62,09	215,07	0,910	0,02	0,02	0,09	0,863
		rb	0,02	0,02	0,06	0,825	2,56	3,61	13,52	**0.983**	694,42	992,65	3776,05	0,931	40,86	53,63	154,36	0,933	0,01	0,02	0,08	0,932
	máx	mm	0,02	0,02	0,13	0,990	22,32	24,91	84,88	0,783	1337,32	1701,27	6482,86	0,983	57,95	79,86	371,27	0,982	0,01	0,01	0,04	0,993
		rb	0,01	0,01	0,04	**0.996**	8,70	10,25	41,70	0,963	1356,03	1626,51	4665,93	**0.985**	25,66	34,67	169,86	**0.997**	0,00	0,00	0,01	**0.999**
30 dias	2a	mm	0,03	0,03	0,07	0,424	7,42	10,48	33,81	0,850	789,25	1084,25	3931,85	0,912	72,85	92,52	244,74	0,785	0,01	0,02	0,08	0,918
		rb	0,02	0,02	0,06	0,704	2,54	3,49	12,18	**0.983**	467,30	644,35	2619,10	0,969	52,83	59,55	177,21	0,911	0,01	0,01	0,03	0,986
	máx	mm	0,02	0,04	0,20	0,978	8,17	11,15	62,19	0,956	1385,55	1851,34	7331,46	0,980	54,32	74,37	326,85	0,985	0,01	0,02	0,09	0,989
		rb	0,01	0,01	0,04	**0.998**	5,92	8,13	47,22	0,977	481,47	646,24	2734,87	**0.998**	65,13	67,92	121,39	**0.987**	0,00	0,00	0,01	**0.999**
60 dias	2a	mm	0,03	0,03	0,08	0,312	6,52	8,80	27,58	0,872	923,50	1233,62	4364,95	0,856	52,42	66,89	195,12	0,847	0,02	0,03	0,10	0,795
		rb	0,02	0,02	0,06	0,686	5,74	7,66	24,19	0,903	781,67	1036,49	3569,11	0,898	47,74	59,21	179,51	0,880	0,01	0,02	0,07	0,917
	máx	mm	0,01	0,02	0,09	0,995	9,21	11,26	53,49	0,954	1793,85	2292,93	7870,09	0,969	54,26	75,04	362,24	0,984	0,02	0,02	0,11	0,981
		rb	0,01	0,01	0,03	**0.999**	4,96	6,88	37,46	**0.983**	1513,83	1942,29	6981,55	**0.977**	15,86	21,38	81,58	**0.999**	0,01	0,01	0,07	**0.991**
90 dias	2a	mm	0,02	0,03	0,08	0,488	8,31	11,55	38,54	0,762	1143,29	1467,58	4782,39	0,727	75,05	91,82	263,13	0,612	0,01	0,02	0,09	0,864
		rb	0,02	0,02	0,05	0,729	5,62	7,96	26,30	0,887	971,00	1254,67	4206,30	0,800	59,22	73,13	200,52	0,754	0,01	0,02	0,08	0,915
	máx	mm	0,02	0,03	0,15	0,985	5,52	7,79	45,75	0,977	1008,62	1419,32	6017,23	0,988	53,73	73,28	343,09	0,984	0,00	0,01	0,03	**0.999**
		rb	0,01	0,01	0,06	**0.996**	3,77	5,10	22,97	**0.990**	835,51	1067,81	3911,03	**0.993**	32,25	41,02	168,48	**0.995**	0,01	0,01	0,01	0,998

Fonte: Autoral.

4.3 Comparativo entre os métodos

Na sequência, é possível visualizar os erros máximos de cada criptomoeda em dólares, do período de coleta, da janela de tempo, em função do algoritmo e escalador utilizado. Nas legendas tem-se o nome do método (SVR ou LSTM) e a abreviação do nome do escalados (RB para `RobustScaler` e MM para `MinMaxScaler`). Dos 100 resultados obtidos de cada combinação com ambos os algoritmos, *SVR* teve 25% com o R-quadrado maior do que 0.9, enquanto *LSTM* obteve 72%

4.3.1 ADA

Analisando-se ambos os gráficos de ADA (Figuras 40 e 41), é possível notar que o algoritmo SVR com o escalador `MinMaxScaler` apresentou os piores resultados, enquanto, em contrapartida, a combinação LSTM com `RobustScaler` (LSTM + RB) gerou o melhor resultado em todos os casos, com exceção da janela de 5 dias na Figura 41, em que o melhor resultado foi desse mesmo escalador, com o algoritmo SVR. No período máximo, observa-se que as previsões obtiveram erros menores nas melhores previsões.

4.3.2 BNB

Em relação à BNB, é possível observar na Figura 42 que os valores de erro máximo para o período de coleta dos dados de 2 anos são próximos para o escalador `RobustScaler` e representam também os melhores resultados, para ambos os algoritmos. Novamente, nota-se que o escalador `MinMaxScaler` não gerou bons resultados. Para o período máximo, na Figura 43, observa-se que, nas janelas de tempo de 5 e 15 dias, os melhores resultados foram do escalador `RobustScaler` com o algoritmo SVR, enquanto nas outras janelas foi o algoritmo LSTM, com o mesmo escalador. Ainda é possível notar que, com o aumento da janela de tempo, o SVR tem um desempenho consideravelmente pior com qualquer um dos escaladores, notando-se uma progressão no erro máximo. Em contrapartida, o comportamento de LSTM é o oposto, considerando o período máximo de coleta dos dados.

4.3.3 BTC

Bitcoin é a foi pioneira no mundo das criptomoedas. O período máximo de coleta apresenta, aproximadamente, 15 anos de dados. Dessa forma, é possível observar na Figura 45 que o algoritmo SVR apresentou, de longe, os piores resultados para as previsões, independente do escalador utilizado. O menor erro para esse algoritmo foi superior a U$ 10 000,00. Nesse aspecto, nota-se que o LSTM se adaptou melhor às variações de preço ao longo do tempo, observando-se que, no período máximo, todos os erros foram inferiores a U$ 5000,00 para o escalador `RobustScaler`, com exceção da janela de 60 dias. No período de 2 anos, conforme a Figura 44, observa-se, pela escala do eixo y, que os resultados apresentaram, em geral, erros menores que o da coleta máxima. Assim

como no BNB, nota-se que, com o aumento da janela de tempo, o algoritmo SVR apresenta erros maiores, enquanto LSTM não varia muito. Os menores erros para BTC no período de coleta de 2 anos foram na janela de 30 dias, utilizando o algoritmo LSTM e o escalador `RobustScaler` e na janela de 5 dias, com o mesmo escalador, mas utilizando o algoritmo SVR.

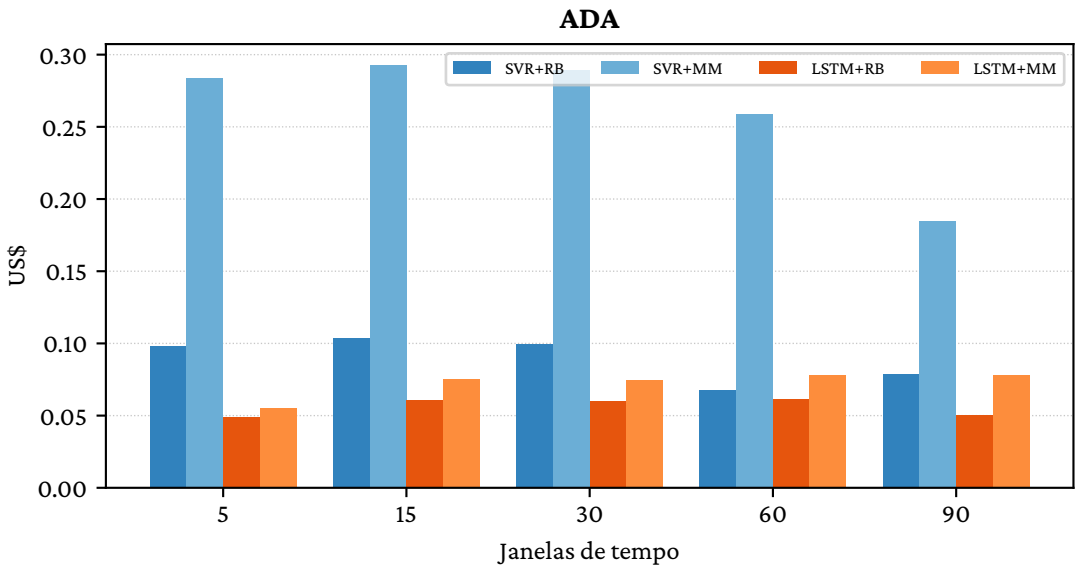
4.3.4 ETH

Comparando as Figura 46 e 47, observa-se que os erros foram, no geral, maiores quando considera-se o período máximo de coleta dos dados. Isso pode dever-se ao fato de que, num conjunto menor de dados, a volatilidade do mercado de criptomoedas influencia menos e o algoritmo prevê melhor. Novamente, o aumento da janela de tempo fez com que o algoritmo SVR apresentasse pior desempenho e LSTM com o escalador `RobustScaler` obteve a maioria dos melhores resultados. Observa-se que os menores erros foram, respectivamente, para o período de 2 anos e máximo, SVR e `RobustScaler` e LSTM e `RobustScaler`. Novamente, nota-se o melhor desempenho do `RobustScaler`.

4.3.5 XRP

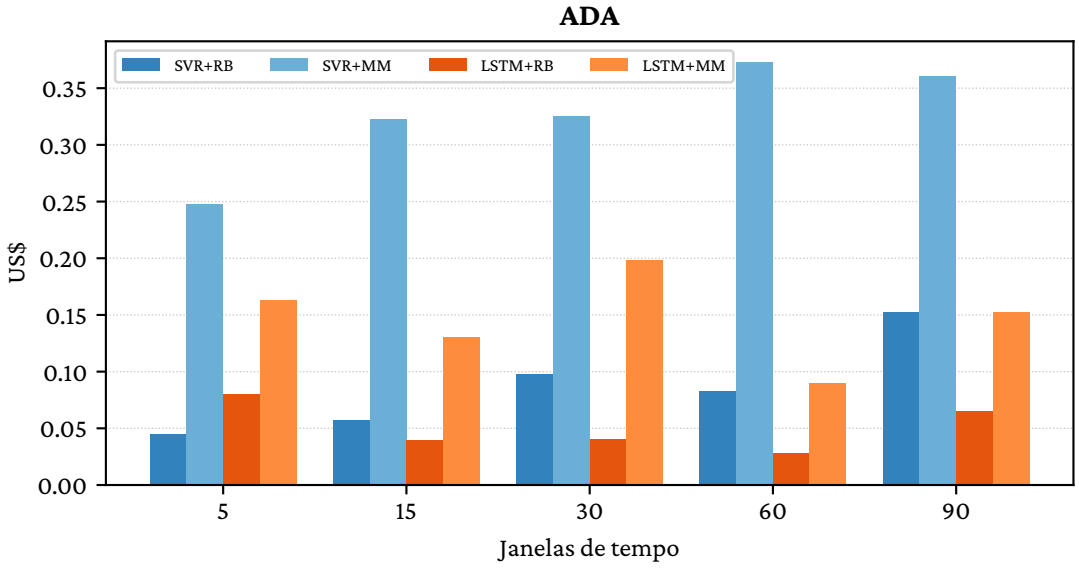
Por fim, para a XRP, observa-se que quase todos os erros para o período de 2 anos, na Figura 48, foram menores que U$ 0,10. Isso deve-se ao fato do próprio valor da criptomoeda ser o menor dentre as analisadas. De toda forma, observa-se que, nesse período, LSTM obteve melhor resultado em 3 das 5 janelas (30, 60 e 90 dias), com o escalador `RobustScaler`, enquanto SVR obteve nas outras duas (5 e 15), com esse mesmo escalador. Isso retifica que o escalador `MinMaxScaler` não apresentou os melhores resultados nas previsões. Para o período máximo (Figura 49), pode-se observar que o escalador `RobustScaler` obteve os melhores resultados em todos os casos, sendo 4 deles (com as janelas de tempo de 5, 15, 30 e 90 dias) com o algoritmo LSTM.

Figura 40 – Comparativo dos valores de erro máximo de ADA nos algoritmos com o período de coleta dos dados de 2 anos.



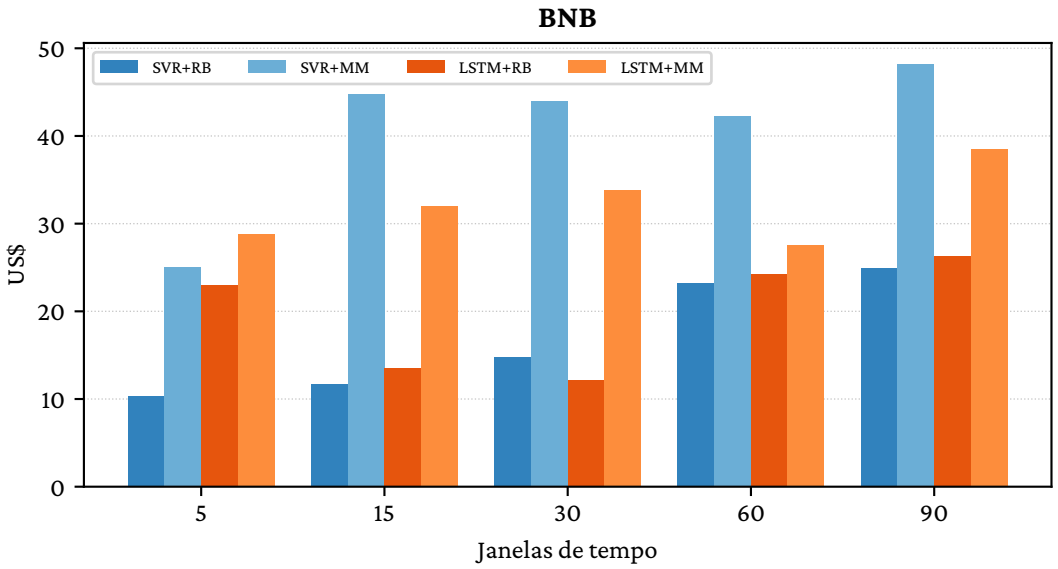
Fonte: Autoral.

Figura 41 – Comparativo dos valores de erro máximo de ADA nos algoritmos com o período de coleta dos dados máximo.



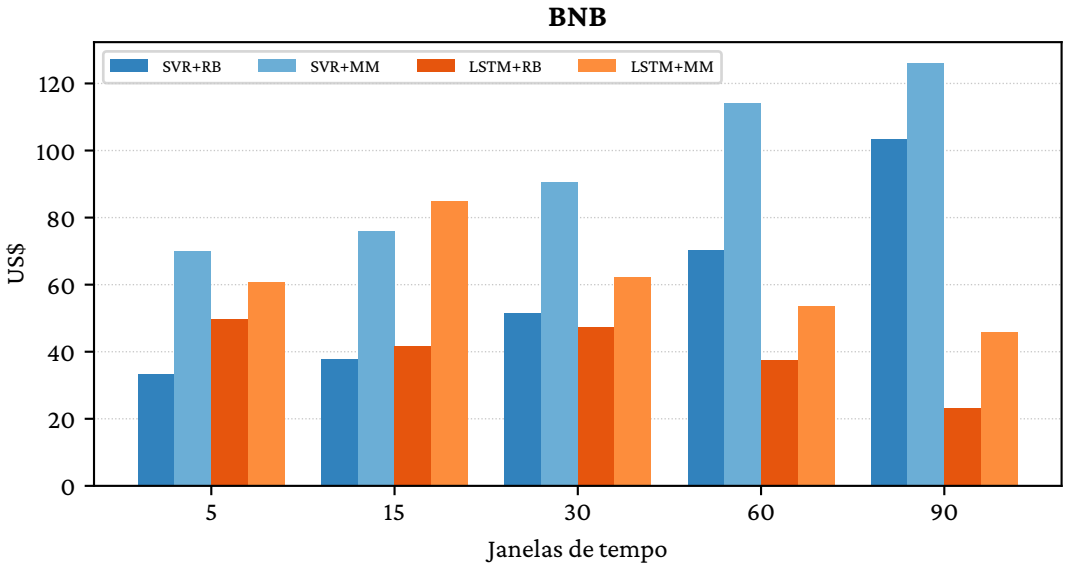
Fonte: Autoral.

Figura 42 – Comparativo dos valores de erro máximo de BNB nos algoritmos com o período de coleta dos dados de 2 anos.



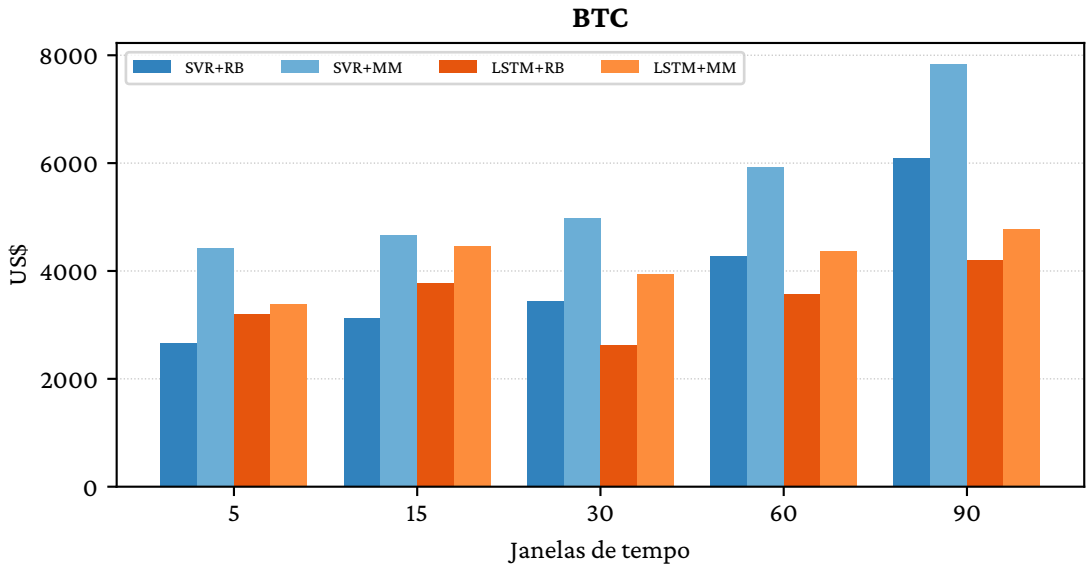
Fonte: Autoral.

Figura 43 – Comparativo dos valores de erro máximo de BNB nos algoritmos com o período de coleta dos dados máximo.



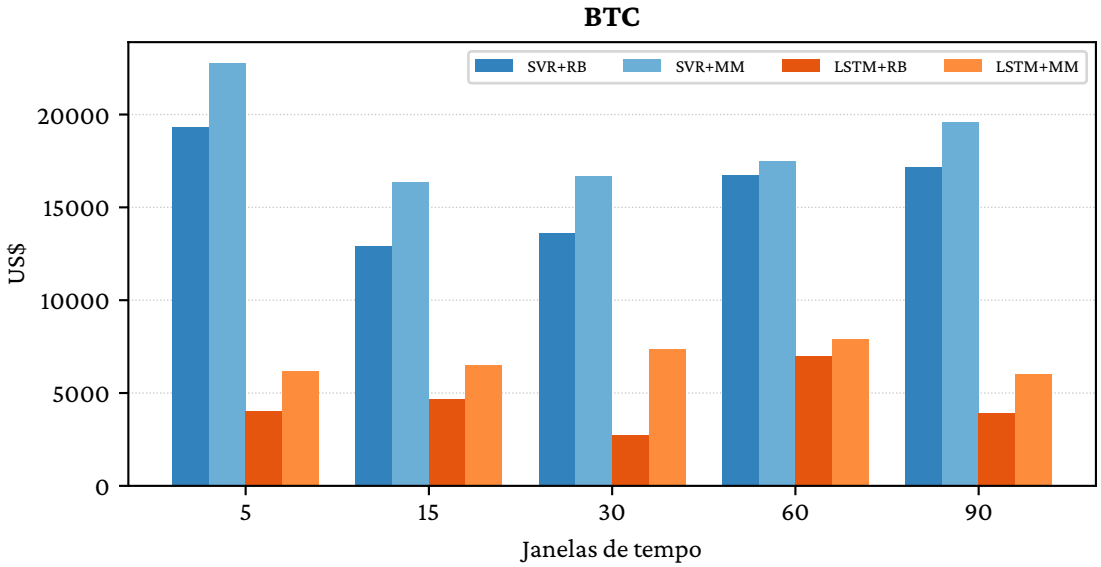
Fonte: Autoral.

Figura 44 – Comparativo dos valores de erro máximo de BTC nos algoritmos com o período de coleta dos dados de 2 anos.



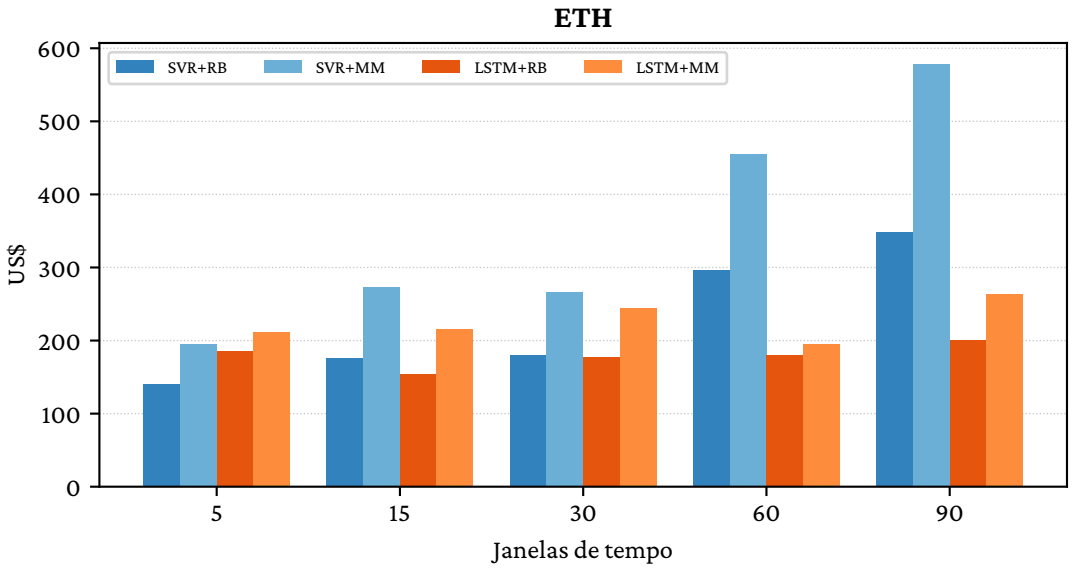
Fonte: Autoral.

Figura 45 – Comparativo dos valores de erro máximo de BTC nos algoritmos com o período de coleta dos dados máximo.



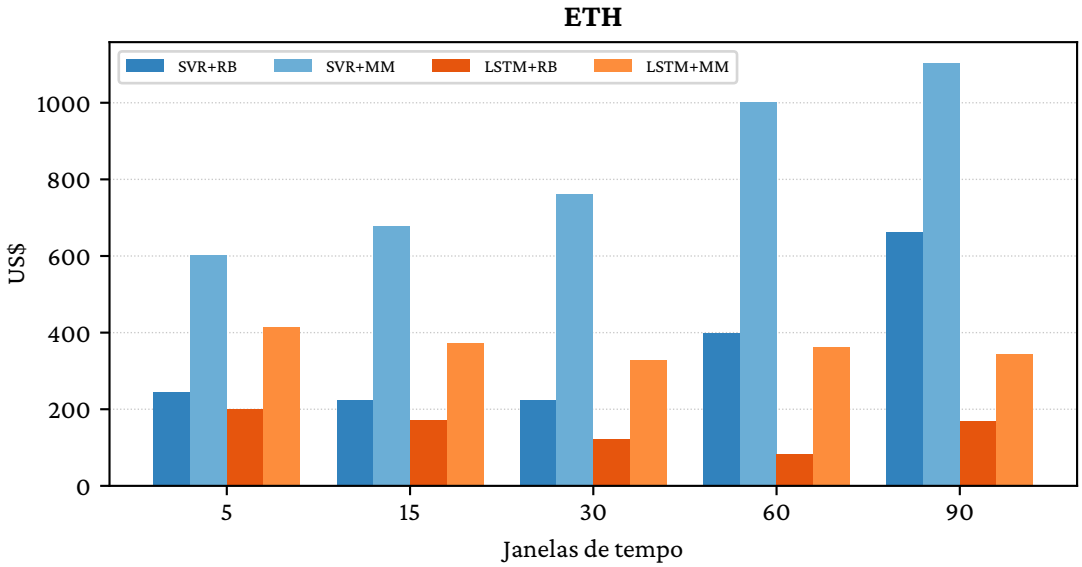
Fonte: Autoral.

Figura 46 – Comparativo dos valores de erro máximo de ETH nos algoritmos com o período de coleta dos dados de 2 anos.



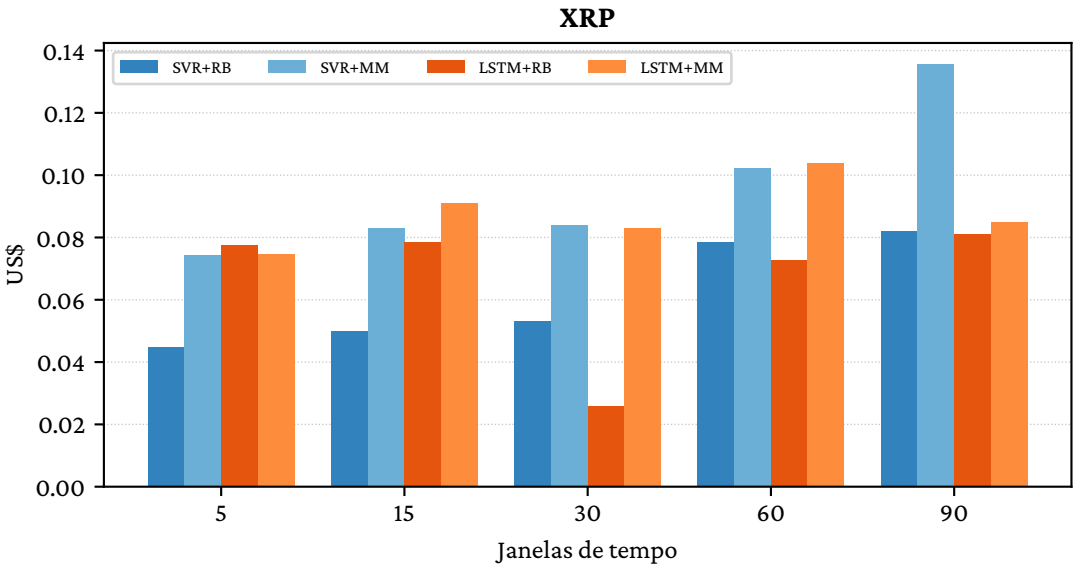
Fonte: Autoral.

Figura 47 – Comparativo dos valores de erro máximo de ETH nos algoritmos com o período de coleta dos dados máximo.



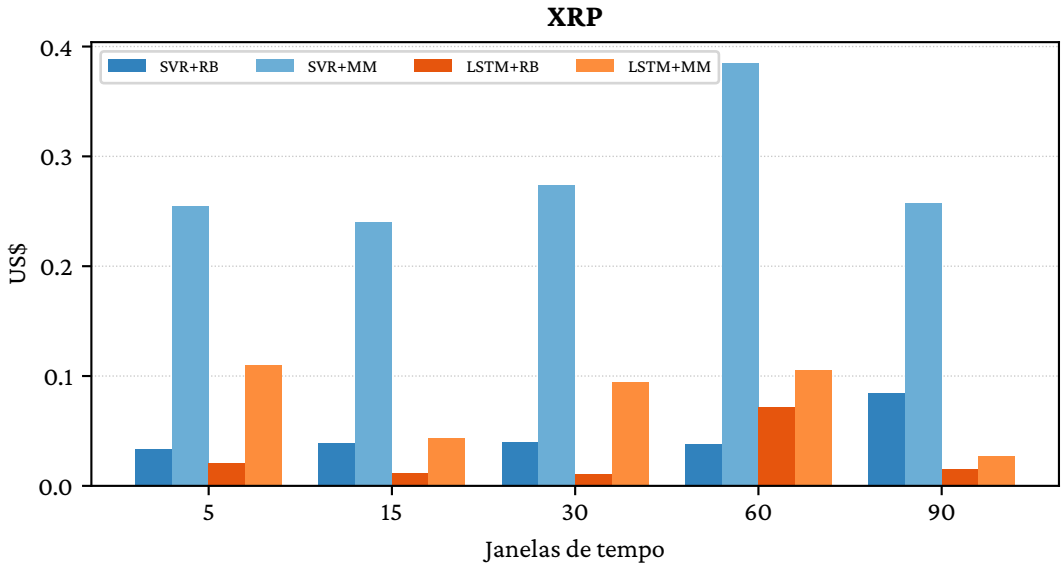
Fonte: Autoral.

Figura 48 – Comparativo dos valores de erro máximo de XRP nos algoritmos com o período de coleta dos dados de 2 anos.



Fonte: Autoral.

Figura 49 – Comparativo dos valores de erro máximo de XRP nos algoritmos com o período de coleta dos dados máximo.



Fonte: Autoral.

4.4 Conclusão

Prever preços de criptomoedas é uma tarefa bastante desafiadora, uma vez que a volatilidade é uma característica inerente a sua variação de preços. No mundo das finanças, no geral, desempenhos bons de algoritmos baseados em valores passados não garantem a qualidade de suas previsões em outro intervalo de tempo, embora possam fornecer subsídios para tomadores de decisão. O `RobustScaler` mostrou-se uma alternativa melhor de escalador para previsão dos preços, devido sua característica de possibilitar a redução de impacto de *outliers*, aspecto comum no mercado financeiro. Em relação ao algoritmo, observa-se que *LSTM* gerou melhores resultados, devido a sua característica de ser eficaz para séries temporais. Ademais, no geral, o período de coleta de 2 anos gerou erros menores.

5 Considerações finais

As previsões de preços de criptomoedas com a utilização de algoritmos de Aprendizado de Máquina como SVR e LSTM possibilitam uma análise dos movimentos do mercado de criptoativos e suas possíveis tendências. É importante ressaltar que a previsão de preços de criptomoedas é complexa, uma vez que esse mercado é altamente volátil, tendo variações drásticas de preços em questão de dias por motivos variados, como situação atual da economia norte-americana, alguma publicação em rede social, lançamentos de produtos, entre outros. Dessa forma, considera-se esses algoritmos apenas como uma ferramenta de auxílio, sem garantia de precisão nos resultados.

No projeto inicial, dois dos objetivos não foram contemplados, devido o tempo de realização e complexidade do projeto. A intenção inicial era trabalhar também com as previsões de tendências das criptomoedas, ou seja, se os preços iriam subir ou descer e com o algoritmo Random Forests, através de árvores de decisão. De toda forma, é válido destacar que o projeto pode tomar continuidade nesse escopo, estudando as variações de tendência das criptomoedas explorando outros algoritmos. Nesse sentido, também não foram utilizadas os indicadores, uma vez que, para regressão, não se fazem necessários.

No geral, nos resultados obtidos, notou-se que o LSTM obteve resultados melhores. Isso pode dever-se ao fato de esse algoritmo ser apropriado para séries temporais, que é o caso dos preços de criptomoedas, em que sua variação ocorre através de um determinado espaço de tempo. Outro ponto importante na comparação dos algoritmos é a escolha dos parâmetros e o teste com diferentes possibilidades. Por exemplo, considerar os preços dos últimos 2 anos, na maioria das vezes, fez com que o modelo previsse mais corretamente os valores, assim como a escolha dos escaladores, janela de tempo, entre outros.

Para que fosse possível avaliar os melhores resultados, foram utilizadas métricas de avaliação, ferramentas responsáveis por comparar, através de diferentes técnicas e cálculos matemáticos, os valores obtidos e os reais. Como uma extensão e alavancagem deste trabalho, sugere-se a exploração de outras janelas de tempo, escaladores e período de coleta de dados. Ademais, é importante ressaltar a escolha de um algoritmo próprio para séries temporais, assim como *LSTM*, para que a sequencialidade das informações seja respeitada.

Referências

ABADI, Joseph; BRUNNERMEIER, Markus. **Blockchain economics**. 2018. 47 p. Disponível em: <https://www.nber.org/system/files/working_papers/w25407/w25407.pdf>. Acesso em: 25 nov. 2022. Citado na p. 26.

ABE, Shigeo. **Support Vector Machines for Pattern Classification**. 1. ed.: Springer, 2005. 343 p. ISBN 9781849960984. Citado na p. 39.

ACADEMY, Data Science (Ed.). **Deep Learning Book**. 2022. Disponível em: <https://www.deeplearningbook.com.br/>. Acesso em: 2 fev. 2023. Citado nas pp. 53, 58, 59, 61–64.

AGGARWAL, Charu C. **Neural Networks and Deep Learning**: A Textbook. Springer, 2018. ISBN 9783319944623. DOI: 10.1007/978-3-319-94463-0. Citado nas pp. 52, 54, 56–58, 60, 107, 110.

ANGELO, Monika di; SALZER, Gernot. Tokens, Types, and Standards: Identification and Utilization in Ethereum. In: 2020 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS). 2020. P. 1–10. DOI: 10.1109/DAPPS49028.2020.00001. Citado nas pp. 25, 26.

APOSTOLIDIS-AFENTOULIS, Vasileios; LIOU, Konstantina-Ina. **SVM Classification with Linear and RBF kernels**. 2015. DOI: 10.13140/RG.2.1.3351.4083. Citado na p. 40.

BANK, World. **The Decline in Access to Correspondent Banking Services in Emerging Markets: Trends, Impacts, and Solutions**. Washington, DC, 2018. Disponível em: <url%7Bhttps://documents1.worldbank.org/curated/en/552411525105603327/pdf/The-decline-in-access-to-correspondent-banking-services-in-emerging-markets-trends-impacts-and-solutions-lessons-learned-from-eight-country-case-studies.pdf%7D>. Acesso em: 5 mar. 2022. Citado na p. 28.

BASU, Saunak; HAN, Wencui; GARIMELLA, Aravinda. Impact of Artificial Intelligence on Human Decision Making on ICO Platforms. In: ICIS. 2019. Citado na p. 54.

BAZARAA, Mokhtar S.; JARVIS, John J.; SHERALI, Hanif D. **Linear Programming and Network Flows**. 4ª edição: John Wiley & Sons, 2010. ISBN 9780470462720. Citado na p. 45.

BEZERRA, Waldemar; OLIVEIRA, Alexandre; SANTOS, Daiane. A tecnologia blockchain e economia do token desmaterialização dos investimentos. In: BETIM, Leozenir Mendes. **Oportunidades e desafios da administração contemporânea**. Dez. 2020. ISBN 978-65-88580-14-1. DOI: 10.47573/aya.88580.2.10.4. Citado na p. 26.

BISHOP, Christopher M. **Pattern recognition and machine learning**. springer, 2006. Citado nas pp. 47, 106.

BLUM, Avrim. **Machine Learning Theory**. Department of Computer Science, 2000. Disponível em: <https://www.cs.cmu.edu/~avrim/Talks/mlt.pdf>. Acesso em: 9 nov. 2021. Citado na p. 31.

BOYD, Stephen P. **Convex Optimization**. Cambridge University Press, 2004. ISBN 9780521833783. Citado na p. 45.

BUNJAKU, Flamur; GJORGIEVA-TRAJKOVSKA, Olivera; MITEVA-KACARSKI, Emilija. Cryptocurrencies – advantages and disadvantages. **Journal of Economics**, v. 2, n. 1, p. 31–39, 2017. Citado na p. 24.

BURGES, Christopher J.C. A Tutorial on Support Vector Machines for Pattern Recognition. **Data Mining and Knowledge Discovery**, n. 2, p. 121–167, 1998. DOI: https://doi.org/10.1023/A:1009715923555. Citado nas pp. 45, 46.

CHAPMAN, Arthur D. **Principles and methods of data cleaning**. GBIF, 2005. Citado na p. 33.

CHOLLET, François. **Deep Learning with Python**. Manning Publications Co., 2018. ISBN 9781617294433. Citado nas pp. 31–33, 53.

CRISTIANINI, Nello; SHAWE-TAYLOR, John. **An Introduction to Support Vector Machines and Other Kernel-based Learning Methods**. Cambridge University Press, 2000. DOI: 10.1017/CBO9780511801389. Citado na p. 47.

CROSBY, Michael et al. **Blockchain Technology**: Beyond Bitcoin. 16 out. 2015. 35 p. Disponível em: <https://scet.berkeley.edu/wp-content/uploads/BlockchainPaper.pdf>. Acesso em: 9 nov. 2021. Citado na p. 25.

_____. Blockchain technology: Beyond Bitcoin. **Applied Innovation**, v. 2, n. 6-10, p. 71, 2016. Citado na p. 24.

CROWELL, Brendan W; BOCK, Yehuda; LIU, Zhen. Single-station automated detection of transient deformation in GPS time series with the relative strength index: A case study of Cascadian slow slip. eng. **Journal of geophysical research. Solid earth**, v. 121, n. 12, p. 9077–9094, 2016. ISSN 2169-9313. Citado na p. 111.

DAVID SCHWARTS, Noah Youngs; BRITTO, Arthur. The Ripple Protocol Consensus Algorithm. **Ripple Labs Inc**, 2014. Citado na p. 28.

DEISENROTH, Marc Peter; FAISAL, A. Aldo; ONG, Cheng Soon. **Mathematics for Machine Learning**. Cambridge: Cambridge University Press, 2020. DOI: 10.1017/9781108679930. Citado nas pp. 39–43, 49.

DEVELOPERS, Scikit. **Support Vector Regression (SVR) using linear and non-linear kernels**. 2023. Disponível em: <https://scikit-learn.org/stable/auto_examples/svm/plot_svm_regression.html#sphx-glr-auto-examples-svm-plot-svm-regression-py>. Acesso em: 30 mar. 2023. Citado na p. 53.

DUCH, Włodzisław; JANKOWSKI, Norbert. Survey of neural transfer functions. **Neural computing surveys**, v. 2, n. 1, p. 163–212, 1999. Citado nas pp. 108, 109.

EHLERS, R. Análise de séries temporais: Relatório Técnico. **São Paulo: USP**, 2009. Citado na p. 101.

ELHAIK, Eran; GRAUR, Dan. On the Unfounded Enthusiasm for Soft Selective Sweeps III: The Supervised Machine Learning Algorithm That Isn't. **Genes**, v. 12, n. 4, 2021. ISSN 2073-4425. DOI: 10.3390/genes12040527. Disponível em: <https://www.mdpi.com/2073-4425/12/4/527>. Citado nas pp. 35, 36.

ELLIS, Craig A; PARBERY, Simon A. Is smarter better? A comparison of adaptive, and simple moving average trading strategies. eng. **Research in international business and finance**, Elsevier B.V, v. 19, n. 3, p. 399–411, 2005. ISSN 0275-5319. Citado na p. 111.

EXCHANGE, Binance. **Binance Whitepaper**. 2017. Citado na p. 29.

FERREIRA, Frederico Lage. **Blockchain e Ethereum: Aplicações e Vulnerabilidades**. 2017. Monografia – Universidade de São Paulo. Citado na p. 29.

FERREIRA, Rosana. **A Blockchain CARDANO**. Set. 2018. Citado nas pp. 29, 30.

FINZER, Devin; HOLLANDER, Nadav. **Open Sea**. 2017. Disponível em: <opensea.io>. Acesso em: 25 nov. 2022. Citado na p. 26.

FLETCHER, Tristan. **Support Vector Machines Explained**. 2009. 19 p. Disponível em: <https://www.cs.ucl.ac.uk/staff/T.Fletcher/>. Acesso em: 25 nov. 2022. Citado nas pp. 39, 47, 48.

GÉRON, Aurélien. **Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow**: Concepts, Tools, and Techniques to Build Intelligent Systems. 2nd: O'Reilly Media, Inc., 2019. ISBN 9781492032649. Citado nas pp. 31, 33, 35, 36.

GUPTA, Manav. **Blockchain for Dummies**. 3. ed.: International Business Machines Corporation, 2020. 43 p. ISBN 9781119621973. Citado nas pp. 24, 27, 29.

HÄRDLE, Wolfgang Karl; HARVEY, Campbell R; REULE, Raphael C G. Understanding Cryptocurrencies. **Journal of financial econometrics**, v. 18, n. 2, p. 181–208, 2020. ISSN 1479-8409. Citado nas pp. 24–26.

HASTIE, Trevor; TIBSHIRANI, Robert; FRIEDMAN, Jerome. **The Elements of Statistical Learning**. New York, NY, USA: Springer New York Inc., 2001. (Springer Series in Statistics). Citado nas pp. 40, 41.

HAYKIN, Simon. **Redes Neurais: Princípios e Prática**. Bookman, 2001. ISBN 8573077182. Citado na p. 110.

HOCHREITER, Sepp; SCHMIDHUBER, Jürgen. Long short-term memory. **Neural computation**, MIT Press, v. 9, n. 8, p. 1735–1780, 1997. Citado nas pp. 59, 64.

HOROWITZ, Ellis; SARTAJ, Sahni. **Fundamentals of data structures**. Pitman, 1984. ISBN 9780273020721. Citado na p. 24.

HURWITZ, Judith; KIRSCH, Daniel. **Machine Learning for Dummies**. International Business Machines Corporation, 2018. ISBN 9781119454946. Citado nas pp. 31, 34.

JAMES, Gareth et al. **An introduction to statistical learning: With applications in R**. 1. ed. New York, NY: Springer, 2013. Citado na p. 40.

JONES, Petra J et al. Feature selection for unsupervised machine learning of accelerometer data physical activity clusters – A systematic review. eng. **Gait & posture**, Elsevier B.V, England, v. 90, p. 120–128, 2021. ISSN 0966-6362. Citado na p. 36.

KIM, Daeho; KIM, Jinah; KIM, Jaeil. Elastic exponential linear units for convolutional neural networks. **Neurocomputing**, Elsevier, v. 406, p. 253–266, 2020. Citado nas pp. 108, 109.

KOCER, Baris. Bollinger bands approach on boosting ABC algorithm and its variants. eng. **Applied soft computing**, Elsevier B.V, v. 49, p. 292–312, 2016. ISSN 1568-4946. Citado na p. 111.

KOWALCZYK, Alexandre. **Support Vector Machines Succinctly**. Syncfusion, 2017. 116 p. Citado nas pp. 44–46.

LI, Yi et al. Random forest regression for online capacity estimation of lithium-ion batteries. eng. **Applied energy**, Elsevier Ltd, v. 232, p. 197–210, 2018. ISSN 0306-2619. Citado nas pp. 34, 35.

LOPES, Dercilio Junior Verly; SANTOS BOBADILHA, Gabrielly dos; BEDETTE, Amanda Peres Vieira. Analysis of Lumber Prices Time Series Using Long Short-Term Memory Artificial Neural Networks. **Forests**, MDPI AG, v. 12, n. 428, p. 428, 2021. ISSN 1999-4907. Citado na p. 101.

LORENA, Ana Carolina; LEON FERREIRA DE CARVALHO, André Carlos Ponce de. **Introdução aos Classificadores de Margens Largas (Large Margin Classifiers)**. São Carlos, SP, 2003. 52 p. Disponível em: <https://repositorio.usp.br/directbitstream/b7425783-382d-44a6-a6c4-220b0a2bc574/BIBLIOTECA_113_RT_195.pdf>. Acesso em: 2 dez. 2022. Citado na p. 39.

_____. Uma Introdução às Support Vector Machines. **Revista de Informática Teórica e Aplicada**, v. 14, n. 2, p. 43–67, dez. 2007. DOI: 10.22456/2175-2745.5690. Disponível em: <https://seer.ufrgs.br/index.php/rita/article/view/rita_v14_n2_p43-67>. Acesso em: 10 jan. 2023. Citado nas pp. 31, 36, 43.

MA, Chao; WU, Lei; YING, Lexing. The multiscale structure of neural network loss functions: The effect on optimization and origin. **arXiv preprint arXiv:2204.11326**, 2022. Citado na p. 105.

MCGREGOR, Milecia. **SVM Machine Learning Tutorial – What is the Support Vector Machine Algorithm, Explained with Code Examples**. 2020. Disponível em: <https://www.freecodecamp.org/news/svm-machine-learning-tutorial-what-is-the-support-vector-machine-algorithm-explained-with-code-examples/>. Acesso em: 27 mar. 2023. Citado na p. 50.

MELLO, Rodrigo Fernandes de; PONTI, Moacir Antonelli. **Machine Learning - A Practical Approach on the Statistical Learning Theory**. Springer, 2018. ISBN 9783319949888. DOI: 10.1007/978-3-319-94989-5. Disponível em: <https://doi.org/10.1007/978-3-319-94989-5>. Citado na p. 44.

MELLO CANALLI, Ygor de. Funções de Ativação Hiperbólica em Redes Neurais. **Instituto Alberto Luiz de Coimbra de Pós-Graduação e Pesquisa de Engenharia**, COPPE - UFRJ, 2017. Citado na p. 107.

METRÓPOLES. **Neuralink de Elon Musk inicia testes com chip cerebral em humanos**. 2023. Disponível em: <https://www.metropoles.com/mundo/ciencia-e-tecnologia-int/neuralink-de-elon-musk-inicia-testes-com-chip-cerebral-em-humanos>. Acesso em: 8 fev. 2023. Citado na p. 55.

MILITANI, Davi Ribeiro et al. Enhanced Routing Algorithm Based on Reinforcement Machine Learning-A Case of VoIP Service. eng. **Sensors (Basel, Switzerland)**, MDPI AG, Switzerland, v. 21, n. 2, p. 504, 2021. ISSN 1424-8220. Citado na p. 36.

MISHRA, Chandrahas; GUPTA, DL. Deep machine learning and neural networks: An overview. **IAES International Journal of Artificial Intelligence**, IAES Institute of Advanced Engineering e Science, v. 6, n. 2, p. 66, 2017. Citado na p. 54.

MOREIRA, Kleber Brito. **Blockchain: tecnologia, arquitetura e aplicações**. 2019. Trabalho de Conclusão de Curso – Universidade de Brasília – UnB, Brasília, DF. Disponível em: <https://bdm.unb.br/handle/10483/26357>. Acesso em: 2 dez. 2022. Citado na p. 24.

MORETTIN, Pedro A; TOLOI, Clélia MC. **Análise de séries temporais: modelos lineares univariados**. Editora Blucher, 2018. Citado na p. 102.

NASCIMENTO CAMELO, Henrique do et al. Métodos de Previsão de Séries Temporais e Modelagem Híbrida ambos Aplicados em Médias Mensais de Velocidade do Vento para Regiões do Nordeste do Brasil. **Revista Brasileira de Meteorologia**, FapUNIFESP (SciELO), v. 32, n. 4, p. 565–574, dez. 2017. DOI: 10.1590/0102-7786324005. Disponível em: <https://doi.org/10.1590/0102-7786324005>. Citado na p. 102.

OJHA, Varun Kumar; ABRAHAM, Ajith; SNÁŠEL, Václav. Metaheuristic design of feedforward neural networks: A review of two decades of research. **Engineering Applications of Artificial Intelligence**, v. 60, p. 97–116, 2017. ISSN 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2017.01.013. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0952197617300234>. Citado na p. 57.

OLAH, Chris. Understanding LSTMs, 2015. Citado nas pp. 62, 63.

OPENAI. **OpenAI Chatbot**. 2021. Disponível em: <https://openai.com/api-docs/#chatbot-api>. Acesso em: 8 fev. 2023. Citado na p. 55.

PARK, Y-S; LEK, S. Artificial neural networks: multilayer perceptron for ecological modeling. In: DEVELOPMENTS in environmental modelling. Elsevier, 2016. v. 28. P. 123–140. Citado na p. 55.

PARMEZAN, Antonio Rafael Sabino. **Predição de séries temporais por similaridade**. 2016. Mestrado – Instituto de Ciências Matemática e de Computação, Universidade de São Paulo, São Carlos. DOI: 10.11606/d.55.2016.tde-21112016-150659. Disponível em: <https://doi.org/10.11606/d.55.2016.tde-21112016-150659>. Acesso em: 18 jan. 2023. Citado nas pp. 101–103.

PEDREGOSA, Fabian et al. Scikit-learn: Machine learning in Python. **Journal of machine learning research**, v. 12, Oct, p. 2825–2830, 2011. Citado nas pp. 50, 51.

POPESCU, Marius-Constantin et al. Multilayer perceptron and neural networks. **WSEAS Transactions on Circuits and Systems**, v. 8, jul. 2009. Citado na p. 56.

PUJOL-PERICH, David et al. Ignnition: Bridging the gap between graph neural networks and networking systems. **IEEE network**, IEEE, v. 35, n. 6, p. 171–177, 2021. Citado na p. 54.

SAINATH, Tara N. et al. Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks. In: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2015. P. 4580–4584. DOI: 10.1109/ICASSP.2015.7178838. Citado na p. 59.

SAMUEL, A. L. Some Studies in Machine Learning Using the Game of Checkers. **IBM Journal of Research and Development**, v. 3, n. 3, p. 210–229, 1959. DOI: 10.1147/rd.33.0210. Citado na p. 31.

SARMAH, Simanta Shekhar. Understanding Blockchain Technology. **Computer Science and Engineering**, v. 8, p. 23–29, 2 2018. DOI: 10.5923/j.computer.20180802.02. Citado na p. 24.

SCHUEFFEL, Patrick; GROENEWEG, Nikolaj; RICO, Baldegger. **The Crypto Encyclopedia: Coins, Tokens and Digital Assets from A to Z**. Ago. 2019. ISBN 978-2-940384-47-1. Citado na p. 26.

SHARMA, Vandana. Outlier. eng. **International journal of epidemiology**, 2021. ISSN 0300-5771. Citado na p. 33.

SHIRURU, Kuldeep. Neural Network Approach for Processing Substation Alarms. **International Journals of Power Electronics Controllers and Converters**, v. 1, p. 21–28, out. 2015. Citado na p. 54.

SILVA, Ivan Nunes da; SPATTI, Danilo Hernane; FLAUZINO, Rogério Andrade. **Redes Neurais Artificiais para Engenharia e Ciências Aplicadas**. Artliber, 2016. Citado nas pp. 58, 59.

SOCHER, Richard; MUNDRA, Richard Socher. CS 224D: Deep Learning for NLP1, 2016. Citado na p. 109.

SUSSNER, Peter. Morphological perceptron learning. In: IEEE. PROCEEDINGS of the 1998 IEEE International Symposium on Intelligent Control (ISIC) held jointly with IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA) Intell. 1998. P. 477–482. Citado na p. 56.

SWOKOWSKI, Earl William. **Cálculo com Geometria Analítica**. 2ª edição. São Paulo, SP: Markon Books do Brasil, 1994. v. 2. Citado na p. 44.

SZANDALA, Tomasz. Enhancing Deep Neural Network Saliency Visualizations with Gradual Extrapolation. **CoRR**, abs/2104.04945, 2021. arXiv: 2104.04945. Disponível em: <https://arxiv.org/abs/2104.04945>. Citado na p. 107.

TRUNG THANH NGUYEN, Aleksander Leonard Larsen. **Axie Infinity Whitepaper**. 2018. Citado na p. 26.

VALLE-CRUZ, David; FERNANDEZ-CORTEZ, Vanessa; GIL-GARCIA, J. Ramon. From E-budgeting to smart budgeting: Exploring the potential of artificial intelligence in government decision-making for resource allocation. **Government Information Quarterly**, v. 39, n. 2, p. 101644, 2022. ISSN 0740-624X. DOI: https://doi.org/10.1016/j.giq.2021.101644. Citado na p. 55.

VAPNIK, Vladimir Naumovich. **Statistical learning theory**. Nashville, TN: John Wiley & Sons, 1998. Citado nas pp. 39, 40, 47, 48, 50.

_____. **The nature of statistical learning theory**. 2. ed. New York, NY: Springer, jun. 2013. (Information Science and Statistics). Citado na p. 39.

VAROQUAUX, Gaël; MÜLLER, Andreas; GROBLER, Jaques. **Classifier comparison**. 2023. Disponível em: <https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html#sphx-glr-auto-examples-classification-plot-classifier-comparison-py>. Acesso em: 27 mar. 2023. Citado nas pp. 51, 52.

VIEIRA, Raquel. **SEC vs Ripple: Data importante se aproxima, entenda**. 2021. Disponível em: <https://esportes.yahoo.com/noticias/sec-vs-ripple-data-importante-182643192.html>. Acesso em: 25 nov. 2022. Citado na p. 28.

WARNER, Brad; MISRA, Manavendra. Understanding neural networks as statistical tools. Taylor & Francis, v. 50, n. 4, p. 284–293, 1996. Citado na p. 52.

XIAOYUN, Qu et al. Short-term prediction of wind power based on deep long short-term memory. In: IEEE. 2016 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC). 2016. P. 1148–1152. Citado na p. 59.

ZHANG, Yi et al. Ethics and privacy of artificial intelligence: Understandings from bibliometrics. **Knowledge-Based Systems**, Elsevier, v. 222, p. 106994, 2021. Citado na p. 55.

ZHAO, Hang et al. Loss functions for image restoration with neural networks. **IEEE Transactions on computational imaging**, IEEE, v. 3, n. 1, p. 47–57, 2016. Citado na p. 105.

APÊNDICE A – Séries temporais

A.1 Introdução

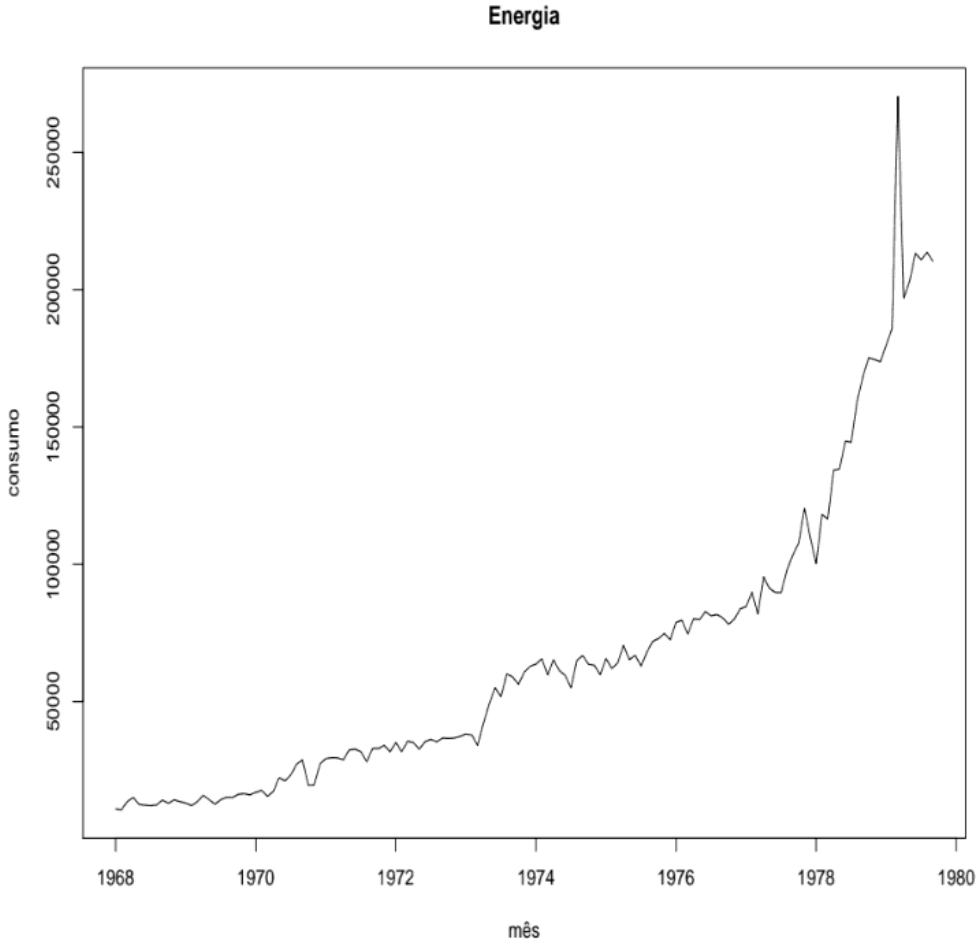
Um conjunto de dados estar descrito na forma de uma série temporal significa que este está atrelado ao tempo, como por exemplo o número de novos infectados por COVID-19 por dia (LOPES; SANTOS BOBADILHA; BEDETTE, 2021). Matematicamente, pode-se dizer que um conjunto de dados em série temporal é definido por $f(t) = (y_1, y_2, \dots, y_n)$, em que y representa o valor observado do dado num instante t, com um total de n observações. Há três fatores que são intrínsecos a esse tipo de conjunto de dados: a tendência dessa, representada por T, a sazonalidade, S, e o ruído, R.

Tendência Segundo EHLERS (2009), é um movimento regular, crescente ou decrescente, que se mantém por um período de tempo ao longo da série. Parmezan (2016) pontua que os tipos de padrões que definem essa regularização do movimento dentro de uma série temporal são: linear, em que o conjunto de dados têm um crescimento (ou decrescimento) constante em proporção linear; exponencial, no qual esse cresce ou decresce de forma correspondente a uma função exponencial (ou seja, da forma $f(x) = a^x$); e amortecido, que representa a queda de dados futuros, antes em crescimento.

É importante pontuar o tipo de tendência para encaixar os dados no modelo de previsão de forma mais precisa, com o intuito de conseguir identificar outras características da série. Uma das formas de se determinar a tendência de uma série temporal é através da correlação entre uma variável dependente e outra independente. Na Figura 50, observa-se que a relação entre os eixos x e y é crescente (PARMEZAN, 2016).

Sazonalidade Sazonalidade é definida como uma medida de tendência regular atrelada aos dados de uma série temporal. Trata-se de uma janela de tempo em que pode-se encontrar um padrão no conjunto de dados, como, por exemplo, variação de temperaturas em determinada estação, quantidade de vendas de material escolar durante o ano, entre outros. Em contraponto, torna-se necessário a remoção da sazonalidade do conjunto de dados da série temporal, permitindo que a previsão seja feita através da tendência e das variações aleatórias (reais) ao longo do tempo, que influenciam na tomada de decisão com as variações do mesmo. Todavia, remover a sazonalidade da série temporal pode significar perder parcela dos dados (PARMEZAN, 2016).

Ruído Ruídos, em questões de séries temporais, normalmente são causados por situações inesperadas, como eventos naturais, influência social, entre outros. Levar esses fatores em conta pode comprometer o resultado das previsões, tornando-se necessária a identificação dos ruídos e exclusão do conjunto de dados (PARMEZAN, 2016).



Fonte: Morettin e Toloi (2018)

Figura 50 – Relação crescente entre variáveis de uma série temporal. Representação de consumo de energia elétrica no Espírito Santo.

A.2 Modelagem de Séries Temporais

De Nascimento Camelo et al. (2017), define-se modelagem como a construção de modelos com a capacidade de descrever dados futuros.

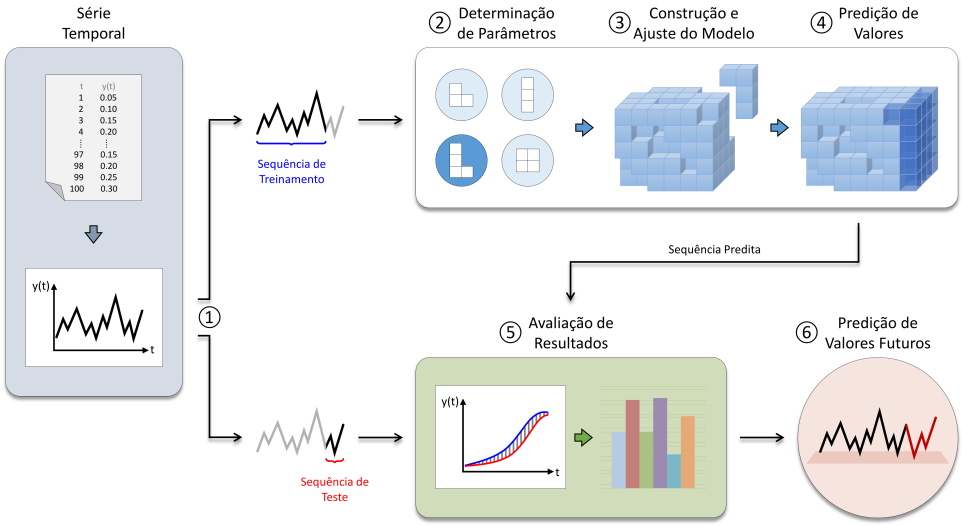
Conforme a Figura 51, seis etapas formam o processo de previsão com séries temporais (PARMEZAN, 2016), definidas como se segue.

1. Assim como nas outras técnicas de Aprendizado de Máquina, o conjunto de dados é dividido em treinamento e teste, sendo o segundo utilizado para avaliação do algoritmo sob a modelagem no primeiro.

2. Nessa etapa, ocorre a definição dos parâmetros com base no conjunto de dados. Alguns parâmetros são pré-definidos, como a janela de tempo para divisão dos dados e outros o próprio algoritmo procura os ideais a fim de minimizar a taxa de erro da previsão.

3. Após a definição dos parâmetros, ocorre a construção e o ajuste do modelo com os dados de treinamento.

Figura 51 – Etapas para previsão de séries temporais.



Fonte: Parmezan (2016).

4. Com o modelo ajustado, ocorrem as previsões do algoritmo.

5. Métricas de avaliação são utilizadas para analisar a qualidade das previsões realizadas pelo algoritmo, avaliando a taxa de erro e acerto, como as definidas em 1.5.2.

6. Continuamente, são realizadas previsões em períodos futuros da série. O algoritmo monitora as métricas para reajustar o modelo e avaliar se esse ainda está prevendo de maneira eficaz.

APÊNDICE B – Função de Perda

A função de perda é um elemento crucial num modelo de rede neural, já que fornece uma medida que avalia se esse está ajustado aos dados de treinamento. Escolher a função de perda adequada é fundamental, já que essa também influencia as previsões das saídas do modelo em função dos dados de entrada. Ela é utilizada para medir o desempenho do algoritmo, comparando as previsões feitas $\hat{y}$ com os valores verdadeiros y. Assim, o objetivo torna-se minimizar a diferença entre $\hat{y}$ e y, por meio de uma função de perda, também conhecida como função objetivo ou função de custo.

Por exemplo, uma regressão de quadrados mínimos com saída numérica requer uma função de perda do tipo $\sum_i (y_i - \hat{y}_i)^2$. Ainda para previsões de valores numéricos, existe uma função de perda chamada *hinge loss*, definida pela Equação (B.1)

$$L(y, \hat{y}) = \max(0, 1 - y\hat{y}). \tag{B.1}$$

Na Equação (B.1), $\hat{y}$ representa a previsão e y o rótulo real. O resultado da função é 0 se $y\hat{y} \geq 1$, indicando que a previsão está correta dentro de certa margem. Caso contrário, ou seja, se $y\hat{y} < 1$, a função de perda tem um valor maior, representando um erro na previsão feita pelo algoritmo.

Para Ma, Wu e Ying (2022), a função de perda é usada para medir a qualidade de previsão de um algoritmo, utilizada para otimizar os pesos de uma modelo durante a fase de treinamento. Ela está ligada diretamente ao impacto da performance de um algoritmo. Existem diversas funções de perda, e cada uma tem seu efeito no processo de aprendizagem do algoritmo. Essas podem ser divididas em duas categorias principais: absolutas ou erro quadrático e perda logarítmica. As funções absolutas, ou de erro quadrático, tendem a ser utilizadas em tarefas que requerem que a rede neural faça uma regressão. As funções de perdas logarítmicas medem a probabilidade de prever os dados corretamente, e são normalmente utilizadas em classificações. Ademais, também divide-se o uso das funções de perda em dados dependentes e de regularização. O tipo dependente é responsável pela performance da rede neural nos dados atuais, enquanto a de regularização estimula a rede a generalizar as características. Usualmente, a função de perda era escolhida anteriormente à aplicação do algoritmo. Em virtude disso, Zhao et al. (2016) propõe uma análise em grade de diversas funções de perda, a fim de escolher qual encaixa melhor no algoritmo.

Pelo fato da escolha da função de perda ser muito dependente do tipo do problema a ser trabalhado pelo algoritmo, é importante entender suas aplicações.

Entropia cruzada binária Em classificações do tipo binária, o objetivo é prever a probabilidade de um dado observado ser de uma classe A ou de uma classe B. Uma das funções de perda utilizada nesse caso, e também uma das mais comuns, é a *entropia cruzada binária*, também chamada, do Inglês, de *log loss*, que mede a diferença entre a distribuição de probabilidade prevista pelo modelo e a verdadeira. Seja $y \in \{0, 1\}$ a classe verdadeira e $\hat{y} \in \{0, 1\}$ a prevista;

a entropia cruzada binária é definida por $L(y, \hat{y}) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$. Conforme Bishop (2006), essa função de custo representa a derivada da *função de verossimilhança de Bernoulli*, representação adequada para variáveis binárias.

Entropia cruzada categórica Em previsões multiclasse, ou seja, onde há mais de uma classe para os dados observados serem classificados, a função mais comumente utilizada é a *entropia cruzada categórica*, que é uma generalização do caso binário para diversas classes. Ela é dada por $L(y, \hat{y}) = -\sum_{i=1}^{K} y_i \log(\hat{y}_i)$, onde K representa o número de classes.

Perda de Articulação Do Inglês *hinge loss*, essa função é comumente utilizada para problemas de classificação, principalmente em SVMs. É uma função baseada em margens que penalizam classificações erradas em relação à distância destas.

Erro Quadrático Médio (EQM) Do Inglês *Mean Squared Error* (MSE), essa função de perda é comumente usada para tarefas de regressão, onde o objetivo é prever um valor contínuo.

APÊNDICE C – Função de Ativação

O objetivo de uma função de ativação é adicionar um componente independente não linear à relação dependente que as variáveis têm entre si, para que a rede consiga aprender além dessas e resolver problemas de maior complexidade. Existem diversos tipos de funções de ativação numa rede neural, sendo as mais usadas representadas na Figura 53 (MELLO CANALLI, 2017). Elas determinam a saída de um neurônio através do somatório dos dados ponderados recebidos. Ademais, as funções de ativação ajudam o modelo a não sobreajustar as previsões, devido ao fato de haver interações mais complexas entre os neurônios.

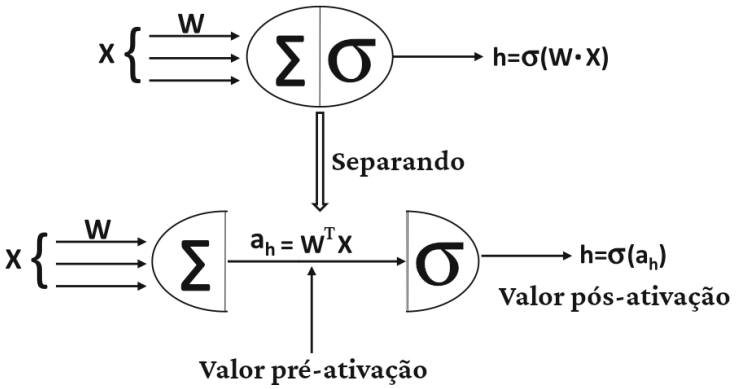
Em contraponto, segundo Szandala (2021), há alguns desafios ao implementar funções de ativação numa rede neural. O maior desafio é encontrar a melhor função de ativação que modele os dados de maneira ideal. Também é importante notar a dificuldade em balancear a complexidade de função de ativação com a complexidade do modelo.

No caso do *perceptron* padrão, leva-se em conta que uma previsão binária será realizada. Em versões diferentes do *perceptron*, é necessária a escolha de uma função de ativação não linear. O resultado da aplicação da função de ativação σ pode ser definido pela Equação (C.1)

$$\hat{y} = \sigma(W^T X). \tag{C.1}$$

Antes de passar pela função de ativação os dados são chamados de *valores pré-ativação* e, posteriormente, *valores pós-ativação*. O resultado final será sempre o valor pós-ativação, uma vez que todos os dados passam pelos neurônios e são atualizados por σ. A Figura 52 demonstra esses valores antes e depois de serem processados pela função de ativação.

Figura 52 – Valores pré e pós função de ativação num neurônio.



Fonte: Adaptado de Aggarwal (2018).

A seguir serão apresentadas brevemente as funções de ativação clássicas, com a indicação de onde elas podem ser aplicadas.

Identidade A função identidade é uma função matemática em que a imagem é o mesmo que o valor de entrada, ou seja, mapeia cada valor do domínio nele mesmo. A notação usual de

uma função identidade é $\sigma(x) = x$. No contexto de redes neurais, a função identidade pode ser implementada como uma rede de apenas um neurônio de entrada e saída, em que os pesos e o termo de viés são construídos de forma a serem os valores de entrada e de saída.

Ela pode ser usada em casos de regressão em que o valores de saída devem ser contínuos num intervalo igual aos valores de entrada. Usa-se também para tarefas de otimização, em que essa pode ser usadas como função de perda em certas aplicações, como detectar *outliers*, penalizando e minimizando a diferença entre o valor previsto e o valor real.

Sign A função sign retorna o sinal do valor de entrada, sendo -1 para valores negativos, $+1$ para valores positivos e 0 para valores nulos. De forma similar à função identidade, a função sign pode ser implementada em uma rede com um neurônio de entrada e um de saída. Essa função de ativação é usada quando a saída de um neurônio individual deve ser um valor discreto que indica a polaridade do valor de entrada. Assim como a função identidade, a função sign também pode ser usada como função de perda a fim de minimizar os erros das previsões.

Logística Conforme Duch e Jankowski (1999), a função logística — também chamada de função sigmoide — possui como imagem o intervalo $[0, 1]$ e pode ser definida pela Equação (C.2)

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{C.2}$$

Por ter sua imagem no intervalo $[0, 1]$, essa função de ativação é bastante usada em classificações binárias. Todavia é comum que com o uso da função sigmoide, ocorra o *problema de gradiente desvanecente*[1].

Unidade Linear Retificada (ReLU) Do Inglês *Rectified Linear Unit*, essa função de ativação tem como imagem o intervalo $[0, +\infty]$. É considerada uma função computacionalmente simples, todavia, retorna zero para valores negativos de um neurônio. A Equação (C.3) define a função ReLU, que atenua o problema de gradiente desvanecente

$$\sigma(x) = \max(0, x). \tag{C.3}$$

Unidade Linear Exponencial (ELU) Do Inglês *Exponential Linear Unit*, a função unidade linear exponencial retorna valores exponenciais para x negativo ou o próprio x, para x positivo. Conforme Kim, Kim e Kim (2020), essa função pode ser computacionalmente cara, mas se mostra eficiente em diversos problemas de redes neurais. A Equação (C.4) define esse tipo de função de ativação

$$\sigma(x) = \begin{cases} x & \text{se } x \geq 0 \\ \alpha e^x - \alpha & \text{se } x < 0 \end{cases}, \tag{C.4}$$

[1] Esse problema se manifesta quando os gradientes de uma rede neural ficam muito pequenos, o que pode desacelerar a aprendizagem do algoritmo. Ele acontece quando a derivada da função de ativação é próxima de 0. Os gradientes são usados para atualizar os pesos dos neurônios durante o treinamento.

onde α é um hiperparâmetro que controla a saída de entradas negativas. É utilizada, comumente, em tarefas que incluem reconhecimento de imagens e textos.

Unidade Linear Exponencial Escalonada (SeLU) Do Inglês *Scaled Exponential Linear Units*, essa função de ativação é uma versão alterada da ELU, considerada eficiente em redes neurais profundas, uma vez que é normalizada por duas constantes, λ e α, conforme a Equação (C.5) (KIM; KIM; KIM, 2020)

$$\sigma(x) = \lambda \begin{cases} x & \text{se } x > 0 \\ \alpha e^x - \alpha & \text{se } x \geq 0 \end{cases}, \tag{C.5}$$

onde α e λ são hiperparâmetros que controlam a saída de entradas negativas e positivas, respectivamente. Ela também tem a característica de ser auto-normalizadora, ou seja, consegue manter média e variação constantes das ativações através da rede.

Na Equação (C.5) as constantes assumem valores $\lambda = 1{,}050\,700\,98$ e $\alpha = 1{,}673\,263\,24$, definidas de forma empírica através de uma série de experimentos. Esses experimentos demonstraram que estes valores ajudam a evitar o problema de gradiente desvanecente, resultando num melhor desempenho para algoritmos de redes neurais.

Tangente Hiperbólica Também considerada uma função sigmoide, a função de ativação tangente hiperbólica possui imagem no intervalo $[-1, +1]$ e pode ser definida pela Equação (C.6) (DUCH; JANKOWSKI, 1999)

$$\sigma(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{C.6}$$

Essa função de ativação é bastante usada para classificações com saídas que podem ser positivas ou negativas. Também sofre do problema de gradiente desvanecente.

Tangente Rígida De forma similar à ativação tanh, a função da tangente rígida também é conhecida como *Leaky ReLU*, de tradução livre ReLU com "vazamento". A definição matemática dessa função é dada pela Equação (C.7)

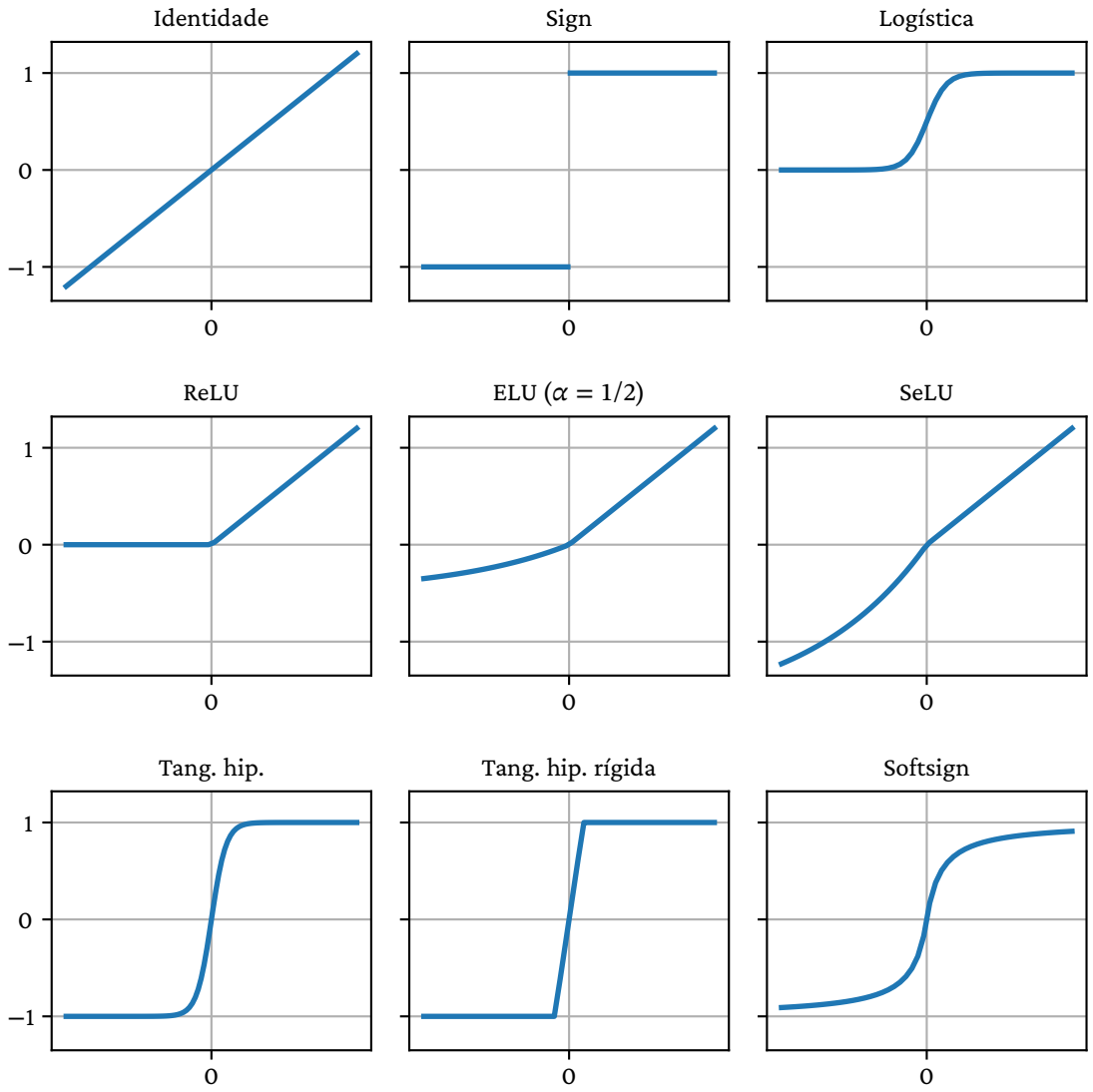
$$\sigma(x) = \begin{cases} -1, & \text{se } x < -1 \\ x, & \text{se } -1 \leq x \leq 1 \\ 1, & \text{se } x > 1 \end{cases}. \tag{C.7}$$

Softsign Essa função, diferentemente da função tangente hiperbólica, converge para um intervalo $[-1, +1]$. Em contraponto, Socher e Mundra (2016) explicita que essa função não gerou tanto impacto em redes neurais. Ela pode ser definida pela Equação (C.8)

$$\sigma(x) = \frac{x}{1 + |x|}. \tag{C.8}$$

Na Figura 53, é possível ver os gráficos das funções de ativação mais comuns.

Figura 53 – Gráficos de diferentes tipos de função de ativação.



Fonte: Adaptado de Aggarwal (2018).

Entre as camadas de entrada e saída, há camadas intermediárias, em que os dados são submetidos às funções de ativação e ponderação dos dados para treinamento no algoritmo. A quantidade de camadas intermediárias é definida antes da execução do algoritmo. O custo computacional é mais elevado para uma quantidade maior de camadas e, além disso, pode ocasionar sobreajuste dos dados (HAYKIN, 2001).

APÊNDICE D – Indicadores

D.1 Média Móvel Simples

Média móvel é um indicador de atraso que denota a média de um conjunto de preços num período de tempo determinado. A fórmula que define a *Média Móvel Simples* (MMS) de um conjunto de dados temporal é dada por:

$$\text{MMS}_n = \frac{1}{n}\sum_{t=k-n+1}^{k} P_t, \tag{D.1}$$

onde n é a quantidade de períodos na média, k é a posição relativa de um período dentre todos e P_t é o preço no período t. Na MMS, todos os preços são igualmente ponderados, diferentemente nas médias móveis linearmente e exponencialmente ponderadas (ELLIS; PARBERY, 2005).

D.2 Bandas de Bollinger

As *Bandas de Bollinger* (BB) são uma ferramenta de análise técnica para previsão de preços de ações e criptomoedas, por exemplo. Todavia, o ideal é que use-se esse indicativo com outros complementares. O cálculo é feito através de uma quantidade pré-definida de preços, o valor médio desses e a variação padrão no intervalo. As Bandas de Bollinger consistem na definição de um intervalo superior e inferior, nos quais as previsões devem estar contidas. Esse indicador desdobra-se em três "bandas", uma superior (BS), uma inferior (BI) e uma central (BC – calculada pela média das anteriores). Assim, um modelo simples de Bandas de Bollinger pode ser calculado por:

$$BS = M + a\sigma, \tag{D.2}$$

$$BI = M - a\sigma, \tag{D.3}$$

$$BC = \frac{BS}{BI}, \tag{D.4}$$

onde M é a média dos preços no tempo determinado, a é uma constante definida pelo algoritmo e σ é o desvio padrão dos preços anteriores (KOCER, 2016).

D.3 Índice de Força Relativa

O *Índice de Força Relativa* (IFR – do Inglês, *Relative Strength Index*) foi pioneiramente apresentado por Wilder no intuito de auxiliar na explicação das mudanças no mercado financeiro que apresentam alto grau de volatilidade. Nesse aspecto, é cabível ao mercado de criptomoedas, uma vez que esse tem a volatilidade como característica. Esse modelo é definido por diversas funções, as quais compõem a fórmula do IFR (CROWELL; BOCK; LIU, 2016).

Primeiramente, são definidas duas funções, uma superior (S) e uma inferior (I), através de uma série temporal d_i deslocada. Assim, há dois casos:

1. Período de alta ($d_i > d_{i-1}$):

$$S(t_i) = d_i - d_{i-1}, \tag{D.5}$$

$$I(t_i) = 0. \tag{D.6}$$

2. Período de baixa ($d_i < d_{i-1}$):

$$S(t_i) = 0, \tag{D.7}$$

$$I(t_i) = d_{i-1} - d_i. \tag{D.8}$$

Com essas funções definidas, é possível calcular a razão das duas médias móveis supracitadas, denominada Força Relativa (FR):

$$FR(t_i) = \frac{\sum_{k=i-n}^{i} S(t_k)}{\sum_{k=i-n}^{i} I(t_k)}. \tag{D.9}$$

Por fim, a FR é convertida em IFR em:

$$IFR(t_i) = 100 - \frac{100}{1 + FR(t_i)}. \tag{D.10}$$