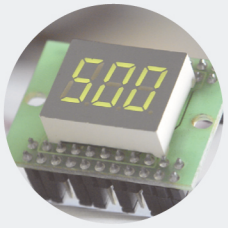
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






Previsão de Preços de Ações no Mercado Financeiro usando Modelos de Machine Learning

DIOGO MIGUEL DOS REIS TEIXEIRA
outubro de 2024

POLITÉCNICO
DO PORTO

Politécnico do Porto
Instituto Superior de Engenharia do Porto

Previsão de Preços de Ações no Mercado Financeiro usando Modelos de Machine Learning

Diogo Miguel dos Reis Teixeira

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização em Automação e Sistemas

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto

Outubro, 2024

Esta dissertação satisfaz, parcialmente, os requisitos que constam da Ficha de Unidade Curricular de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Electrotécnica e de Computadores, Área de Especialização em Automação e Sistemas.

Candidato: Diogo Miguel dos Reis Teixeira, Nº 1190522,
1190522@isep.ipp.pt

Orientação Científica: Ramiro De Sousa Barbosa, rsb@isep.ipp.pt

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
Rua Dr. António Bernardino de Almeida, 431, 4200–072 Porto

Outubro, 2024

Agradecimentos

A conclusão desta dissertação marca o fim de uma importante etapa na minha vida académica, e não poderia deixar de expressar o meu mais profundo agradecimento a todos aqueles que, de diversas formas, contribuíram para que este momento fosse possível.

Em primeiro lugar, gostaria de expressar o meu sincero reconhecimento a todos os docentes, funcionários e técnicos do Instituto Superior de Engenharia do Porto (ISEP), pelo ambiente acolhedor e pelo apoio oferecido ao longo destes 5 anos. Em especial, dirijo o meu maior agradecimento ao meu orientador, Prof. Dr. Ramiro Barbosa, cuja orientação for imprescindível ao longo de todo este processo. A partilha dos seus conhecimentos, paciência, e dedicação, bom como todas as suas valiosas sugestões, incentivos, sabedoria e rigor académico, foram fundamentais para a realização desta dissertação.

Agradeço, igualmente, à minha família, que sempre esteve do meu lado em todos os momentos desta jornada. Aos meus pais, estou muito agradecido pelo vosso apoio incondicional, compreensão e sacrifícios ao longo desta fase fornecendo-me esta oportunidade de seguir este caminho académico. Serei eternamente grato por tudo o que fizeram por mim e por me proporcionarem as ferramentas necessárias para crescer e prosperar.

Um agradecimento especial também aos meus amigos e aos meus colegas de curso, que tornaram este percurso mais leve e agradável, cuja disponibilidade e contributos foram essenciais para o meu progresso académico e a realização deste trabalho.

Resumo

Este trabalho apresenta uma análise sobre a previsão de preços de ações no mercado financeiro, com ênfase em abordagens baseadas em modelos de séries temporais e técnicas de *Deep Learning*. Foram explorados conceitos fundamentais de análise técnica, como médias exponenciais e simples, e analisados diversos índices globais, para serem utilizados como entradas para os modelos de *Machine Learning*, que incluem *Recurrent Neural Network* (RNN), *Long Short-Term Memory* (LSTM), *Gated Recurrent Unit* (GRU), *Convolutional Neural Network* (CNN) e XGBoost. Tudo isto objetivo de capturar dependências temporais nos dados, permitindo assim previsões mais precisas dos movimentos futuros do mercado. Ao abordar esta questão, foi enfatizado a importância do pré-processamento adequado dos dados, incluindo limpeza, normalização e seleção das *features* relevantes. Além disso, foi discutida a necessidade da avaliação do desempenho dos modelos através de métricas adequadas e técnicas de validação para garantir resultados confiáveis.

Palavras-Chave: *Machine Learning*, previsões, mercado de ações, redes neuronais.

Abstract

This paper presents an analysis of stock price forecasting in the financial market, with an emphasis on approaches based on time series models and Deep Learning techniques. Fundamental concepts of technical analysis were explored, such as exponential and simple averages, and various global indices were analyzed to be used as inputs for Machine Learning models, including *Recurrent Neural Network* (RNN), *Long Short-Term Memory* (LSTM), *Gated Recurrent Unit* (GRU), *Convolutional Neural Network* (CNN) and XGBoost. All of this is aimed at capturing temporal dependencies in the data, thus enabling more accurate forecasts of future market movements. In addressing this issue, the importance of proper data pre-processing was emphasized, including cleaning, normalizing and selecting the relevant features. In addition, the need to evaluate model performance using appropriate metrics and validation techniques to ensure reliable results was discussed.

Keywords: Machine Learning, predictions, stock market, neural networks.

Índice

Lista de Figuras

Lista de Tabelas

Listagens

Lista de Acrónimos

ANN	*Artificial Neural Networks*
ARIMA	*Autoregressive Integrated Moving Average*
BCE	*Binary Cross-Entropy*
BPTT	*Backpropagation Through Time*
CCE	*Categorical Cross-Entropy*
CNN	*Convolutional Neural Network*
DJI	*Dow Jones Industrial Average*
EMA	*Exponential Moving Average*
EMH	*Efficient-Market Hypothesis*
FRED	*Federal Reserve Economic Data*
GRU	*Gated Recurrent Unit*
IA	Inteligência Artificial
LSTM	*Long Short-Term Memory*
MACD	*Moving Average Convergence/Divergence*
MAE	*Mean Absolute Error*
MAPE	*Mean Absolute Percentage Error*
ML	*Machine Learning*
MLS LSTM	*Multi-Layer Sequential Long Short-Term Memory*
MSE	*Mean Squared Error*
NASDAQ	*National Association of Securities Dealers Automated Quotations*
NLP	*Natural Language Processing*
NYSE	*New York Stock Exchange*

oob	*out-of-bag*
R^2	*R-Squared*
RMSE	*Root-Mean-Square Error*
RNN	*Recurrent Neural Network*
RSI	*Relative Strength Index*
S&P 500	*Standard and Poor's 500*
SMA	*Simple Moving Average*
SSE	*Shanghai Stock Exchange*
SVM	*Support Vector Machine*
TAIEX	*Taiwan Stock Exchange*
USD	dólar dos Estados Unidos
XGBoost	*eXtreme Gradient Boosting*

Capítulo 1

Introdução

A fusão entre tecnologia e finanças tem transformado radicalmente a maneira como os mercados operam e como os investidores tomam decisões. Com o surgimento de plataformas de negociações *online*, algoritmos de negociação de alta frequência e o uso crescente de Inteligência Artificial (IA), o cenário financeiro está a passar uma revolução digital sem precedentes.

Esta convergência está a redefinir as fronteiras do possível no mercado de ações, oferecendo novas oportunidades e desafios para investidores e analistas. A capacidade de processar grandes volumes de dados em tempo real e aplicar algoritmos avançados de análise está a criar novas oportunidades na previsão e na gerência de riscos. Neste contexto, a pesquisa e o desenvolvimento de modelos de previsão baseados em IA representam uma área de interesse crescente.

1.1 Contextualização

O mercado de ações é um ambiente global onde milhões de investidores compram e vendem ações de empresas, representando uma fração do capital social de uma empresa. O objetivo dessas transações é lucrar com as flutuações de preços dos ativos. Para muitos, investir no mercado de ações é uma parte essencial da sua estratégia financeira, pois oferece uma oportunidade de aumentar o seu capital ao longo do tempo de forma passiva, muitas vezes superando as taxas de retorno oferecidas por investimentos mais tradicionais, como depósitos bancários. No entanto, o mercado de ações também é conhecido pela sua imprevisibilidade e alta volatilidade. Prever

os movimentos futuros do mercado é uma tarefa desafiadora e altamente desejada. Os investidores estão constantemente à procura de novos métodos e técnicas para antecipar as mudanças no mercado e tomar decisões mais informadas sobre as suas carteiras de investimentos.

Ao longo da história, investidores e analistas têm empregado uma diversidade de métodos e técnicas na tentativa de antecipar o comportamento do mercado de ações. Desde análises fundamentais, que avaliam o desempenho financeiro e o potencial de crescimento das empresas, até análises técnicas, que examinam padrões de preços passados para identificar tendências futuras, uma ampla gama de abordagens têm sido exploradas. No entanto, mesmo com todos esses esforços, a capacidade de prever com precisão os movimentos do mercado permanece uma questão desafiadora e em aberto.

Recentemente, com os avanços tecnológicos e a crescente disponibilidade de dados, emergiram novas oportunidades para aplicar técnicas de *Machine Learning* (ML) na previsão do mercado de ações. ML, um subcampo da IA, concentra-se no desenvolvimento de algoritmos capazes de aprender padrões e realizar previsões com base em dados. Ao analisar vastos conjuntos de dados históricos, os algoritmos de ML podem identificar correlações complexas e padrões subtis que podem escapar aos métodos de previsão tradicionais.

Neste trabalho, objetiva-se explorar o potencial dos algoritmos de ML na previsão do mercado de ações. Serão desenvolvidos modelos de previsão capazes de capturar a complexidade e a dinâmica do mercado, fornecendo valiosas informações para os investidores. Ao combinar técnicas avançadas de ML com uma compreensão aprofundada dos mercados financeiros, espera-se contribuir para o avanço do campo e proporcionar benefícios tangíveis para aqueles que operam no mercado de ações.

1.2 Definição do Problema

O problema central abordado nesta dissertação é a previsão de preços de ações utilizando técnicas de *Machine Learning*. A volatilidade destes mercados e a complexidade intrínseca das séries temporais de preços de ações tornam esta tarefa um pouco mais complexa do que a previsão de outro tipo de séries temporais. O objetivo é explorar e comparar diferentes modelos de previsão, para entender quais abordagens são mais eficazes em captar padrões e fazer previsões precisas.

1.2.1 Objetivos

Os principais objetivos deste trabalho são:

- Compreender o mercado de ações: Adquirir uma compreensão básica do funcionamento do mercado de ações e dos fatores que influenciam os preços das ações;

- Explorar técnicas de *Machine Learning*: Estudar e compreender diferentes técnicas de *Machine Learning* e as suas aplicações na previsão de séries temporais;
- Desenvolver e aplicar modelos: Implementar e treinar diversos modelos de *Machine Learning* para a previsão de preços;
- Comparar o desempenho dos modelos: Avaliar e comparar o desempenho dos diferentes modelos através de um conjunto de diferentes métricas para identificar quais técnicas oferecem melhores resultados na previsão de preços.

1.2.2 Resultados esperados

É esperado que este trabalho proporcione uma compreensão do desempenho de diferentes modelos de previsão no âmbito do mercado das ações. A principal expectativa é que os algoritmos mais complexos e aqueles desenvolvidos especificamente para lidar com séries temporais, como o LSTM, apresentem um desempenho superior na previsão dos preços de ações em comparação com algoritmos menos especializados, como o CNN.

1.3 Plano de Trabalho

A fase inicial do plano de trabalho desta tese é efetuar uma pesquisa aprofundada sobre o mercado de ações como um todo, e mais especificamente a utilização de algoritmos de *Machine Learning* para efetuar a sua previsão. Após a pesquisa inicial, o foco será o desenvolvimento dos modelos. Esta fase envolve a criação e implementação dos modelos, seguida pela sua otimização e ajustes para melhorar o seu desempenho e finalmente, obter os resultados para a posterior análise. Simultaneamente ao desenvolvimento dos modelos, será realizada a escrita do relatório da dissertação. Com uma elaboração de um plano de trabalho (Figura 1.1), consegue-se assegurar que cada etapa do projeto seja abordada de forma ordenada dentro do prazo de desenvolvimento.

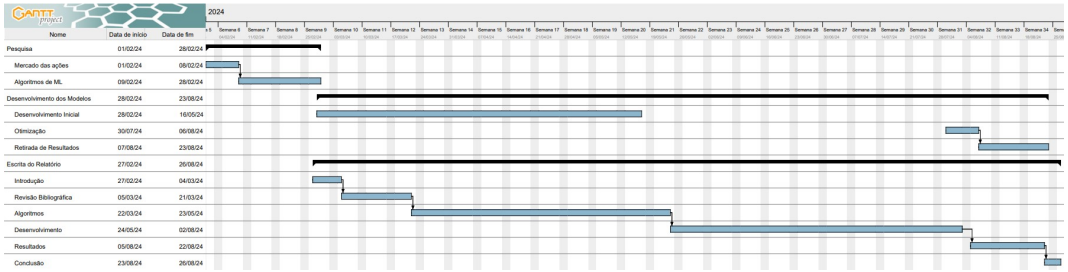


Figura 1.1: Diagrama de Gantt

1.4 Organização da Dissertação

O relatório será composto por um total de 6 capítulos, sendo estes seguidamente apresentados na ordem descrita abaixo:

- **Introdução**: Onde é apresentado o contexto e os objetivos do trabalho;
- **Revisão Bibliográfica**: Onde é examinado e apresentado a literatura existente relevante para o tema da previsão de preços de ações;
- **Algoritmos**: Detalha os diversos algoritmos de *Machine Learning* utilizados;
- **Desenvolvimento**: Descreve a implementação e a criação dos modelos usados para a previsão;
- **Resultados**: Onde são apresentados e analisados os resultados obtidos;
- **Conclusão**: Onde são resumidas as principais descobertas e refletidas as possíveis melhorias.

Capítulo 2

Revisão Bibliográfica

Neste capítulo irá ser abordada uma análise inicial do mercado de ações, explorando as suas funções, objetivos e os conceitos essenciais que fundamentam o seu funcionamento. Em seguida, irá ser discutida a previsão de movimentos do mercado, abordando algumas teóricas e métodos utilizados para tentar antecipar tendências e comportamentos dos preços das ações. Esta secção servirá também como uma introdução para o campo de *Machine Learning*, destacando as suas principais abordagens e algoritmos. Em particular, as abordagens tradicionais e os métodos mais recentes no âmbito do *Deep Learning*, discutindo como estas técnicas podem ser aplicadas para prever o comportamento dos mercados financeiros. Por fim, serão abordados alguns estudos para demonstrar o estado atual da previsão do mercado de ações através dos métodos previamente discutidos. Estes estudos incluirão uma análise detalhada de séries temporais do preço de ações e a aplicação de algoritmos de *Machine Learning* para prever os seus movimentos futuros.

2.1 *Stock Market*

O mercado de ações, conhecido como "*Stock Market*", refere-se aos mercados públicos que existem para a emissão, compra e venda de ações, seja por meio de uma bolsa de valores ou transações *over-the-counter*. As ações, também denominadas títulos de capital, representam a participação fracionada na propriedade de uma empresa. O mercado de ações oferece aos investidores a oportunidade de adquirir e alienar esses ativos financeiros. Um mercado de ações eficiente desempenha um papel crucial no

desenvolvimento económico, pois permite que as empresas obtenham financiamento do público investidor [1].

2.1.1 Finalidades da Bolsa de Valores

O mercado de ações possui dois objetivos fundamentais e interligados. Primeiramente, serve como uma fonte vital de capital para as empresas públicas, permitindo-lhes obter financiamento para expandir as suas operações. Por exemplo, quando uma empresa emite um milhão de ações e vende-as inicialmente a 10 euros por ação, isso proporciona à empresa um capital de 10 milhões de euros que podem ser investidos para impulsionar o crescimento do negócio. Ao optar por emitir ações em vez de recorrer a empréstimos para financiar a sua expansão, a empresa evita o ónus das dívidas e os encargos dos juros associados. Além disso, o mercado de ações oferece aos investidores a oportunidade de participar nos lucros das empresas listadas. Os investidores podem lucrar de duas maneiras distintas. Primeiramente, algumas ações distribuem dividendos regulares, que são pagamentos periódicos feitos aos acionistas como uma parte dos lucros da empresa. Esses dividendos são geralmente pagos em dinheiro, mas também podem ser oferecidos na forma de ações adicionais da empresa. Esses pagamentos proporcionam aos acionistas uma fonte de renda adicional [2]. Além disso, os investidores podem obter lucros através da valorização das ações ao longo do tempo. Por exemplo, se um investidor adquirir ações de uma empresa a 10 euros por ação e posteriormente o preço dessas ações aumentar para 15 euros por ação, o investidor poderá realizar um lucro de 50% ao vender as suas ações por um preço superior ao preço de compra. Essa possibilidade de valorização das ações adiciona uma dimensão adicional de oportunidade de ganho aos investidores no mercado de ações [1].

2.1.2 Valorização de Ações

A valorização das ações é um conceito fundamental no mercado financeiro, referindo-se ao aumento do valor das ações ao longo do tempo. Existem várias razões pelas quais o preço das ações de uma empresa pode aumentar, e entender esses fatores é crucial para os investidores.

Uma das principais razões para a valorização é o desempenho financeiro da empresa. Quando uma empresa regista lucros sólidos, apresenta um crescimento consistente nas receitas e nos lucros, ou anuncia iniciativas estratégicas bem-sucedidas, isso geralmente reflete-se num aumento no preço das suas ações. Os investidores têm confiança na capacidade da empresa de gerar retornos positivos no futuro e, como resultado, estão dispostos a pagar um preço mais alto pelas suas ações.

Além do desempenho financeiro, outros fatores macroeconómicos e do mercado podem também influenciar o valor das ações. Por exemplo, mudanças nas taxas de

juros, políticas governamentais, desenvolvimentos tecnológicos, condições do setor e eventos geopolíticos podem afetar o sentimento do mercado e, consequentemente, o preço das ações.

Além disso, a perceção dos investidores sobre o valor intrínseco da empresa desempenha um papel significativo na determinação do preço das ações. Se os investidores acreditam que uma empresa está subvalorizada em relação ao seu verdadeiro potencial de crescimento e geração de lucro, eles podem aumentar a demanda pelas ações da empresa, resultando na sua valorização.

2.2 Previsão do *Stock Market*

A previsão do mercado de ações é uma atividade complexa que envolve a tentativa de determinar o valor futuro de uma ação de uma determinada empresa ou de outro instrumento financeiro negociado numa bolsa de valores. Esta iniciativa é bastante desejada visto que uma boa previsão do preço futuro de uma ação pode gerar lucros significativos.

Ao explorar a previsão do mercado das ações, uma teoria que surge invariavelmente é a *Efficient-Market Hypothesis* (EMH). Esta hipótese postula que os preços dos ativos refletem completamente todas as informações disponíveis no mercado no momento em que são negociados. Em essência, isto implica que os preços das ações já incorporam todas as informações conhecidas e, portanto, quaisquer mudanças de preço que não estejam fundamentadas em novas informações reveladas são consideradas imprevisíveis conforme a EMH [3]. Essencialmente, a EMH sugere que os mercados financeiros são tão eficientes na incorporação de informações que torna extremamente desafiador, se não impossível, para os investidores preverem consistentemente os movimentos futuros dos preços das ações com base em informações disponíveis publicamente.

Esta teoria, embora seja um alicerce da teoria financeira moderna, ela é altamente controversa, produzindo diversas opiniões sobre a mesma, onde os crentes acreditam que é inútil procurar ações subvalorizadas ou tentar prever as tendências do mercado através de análises fundamentais, ou técnicas, visto que acreditam que todas as ações das empresas representam o seu valor real, e qualquer técnica de previsão do mercado, ao ser descoberta, o preço de todas as ações refletirá imediatamente esta alteração [3].

Embora existem diversos académicos que apontam para inúmeras provas que apoiam a EMH, existem também uma quantidade igual de divergências. Um ótimo exemplo de uma contradição, é o famoso investidor norte-americano Warren Buffett, conhecido pela sua abordagem de investimento fundamentalista e por superar consistentemente o mercado ao longo de décadas. Buffett e outros investidores de valor

contestam a EMH, argumentando que é possível identificar ações subvalorizadas por meio de uma análise cuidadosa dos fundamentos da empresa e do mercado [4].

Esta discordância entre a teoria e a prática levou a uma ampla gama de estratégias de investimento e métodos de previsão do mercado de ações. Alguns investidores adotam abordagens de investimento de longo prazo, focando na seleção de empresas sólidas com fundamentos robustos e perspetivas de crescimento promissoras. Outros concentram-se em estratégias de curto prazo, aproveitando padrões técnicos e flutuações de curto prazo no mercado.

2.2.1 Métodos Tradicionais de Previsão do *Stock Market*

Os métodos tradicionais de previsão dividem-se em três grandes categorias que podem, e muitas vezes, se sobrepõem. Elas são a análise fundamental, a análise técnica e os métodos tecnológicos.

Abordando primeiramente a análise fundamental, esta é um método de avaliação do valor intrínseco de uma ação financeira. Para isto, são combinadas demonstrações financeiras, influências externas, eventos e tendências do sector. Este tipo de análise emprega uma variedade de conjuntos de dados, incluindo dados históricos para incorporar o histórico de uma ação na sua análise; informações públicas conhecidas sobre a empresa, como anúncios feitos pela direção; e informações não publicas, como a forma como a direção lida com crises e outras situações [5].

Para além disso, uma análise financeira inclui a revisão mais detalhada de diversos tópicos relacionados com o preço das ações como:

- Desempenho do sector global;
- Ambiente político nacional;
- Acordos comerciais relevantes e políticas externas;
- Demonstrações financeiras da empresa;
- Comunicados de imprensa;
- Análise da concorrência.

Relativamente à análise técnica, esta é uma estratégia de negociação utilizada para avaliar investimentos e descobrir oportunidades de negociação ao analisar padrões estatísticos obtidos da atividade de negociação, como movimentos de preços e volume. Ao contrário da análise fundamental, que busca avaliar o valor de um ativo com base em resultados de negócios, como vendas e lucros, a análise técnica concentra-se no estudo do preço e do volume.

Existem centenas de padrões e sinais que foram desenvolvidos com o objetivo de apoiar a análise técnica. Estes analistas técnicos também desenvolveram vários

tipos de sistemas de negociação para os ajudar a prever e negociar com base nos movimentos de preços. Diversos indicadores focam-se na identificação da tendência atual do mercado, incluindo áreas de apoio e resistência, enquanto outros focam-se na determinação da força de uma tendência e a probabilidade da sua continuação. Os indicadores técnicos e padrões gráficos normalmente utilizados incluem linhas de tendência (Figura 2.1a), canais, médias móveis (Figura 2.1b) e indicadores de momento [6].

(a) (b)

Figura 2.1: (a) Linhas de Tendências [7] (b) Médias Móveis [8]

Para finalizar, é importante considerar os métodos tecnológicos utilizados na previsão do mercado de ações. Estes métodos têm vindo a ganhar cada vez mais relevância com o avanço da tecnologia e o acesso a grandes quantidades de dados.

Um método tecnológico comum é o uso de algoritmos de ML e IA. Estes algoritmos são capazes de analisar grandes conjuntos de dados históricos e identificar padrões complexos que podem ser utilizados para prever os movimentos futuros do mercado. Além disso, o uso de ML pode adaptar-se e melhorar continuamente à medida que recebe mais dados e *feedback* sobre a sua precisão.

Outro método tecnológico é a análise de sentimento, que envolve a análise de notícias, redes sociais e outras formas de dados não estruturados para determinar o sentimento geral em relação a um determinado ativo ou mercado. Este método pode ser útil para identificar tendências emergentes e antecipar movimentos de preços antes que ocorram [9].

Além disso, outros modelos de séries temporais, como *Autoregressive Integrated Moving Average* (ARIMA) e modelos de redes neuronais, também são amplamente utilizados na previsão do mercado de ações. Estes modelos são capazes de capturar a estrutura temporal dos dados e fazer previsões com base em padrões passados.

2.3 Machine Learning

Machine Learning é um dos diversos campos da IA e da ciência da computação que se concentra na utilização de dados e algoritmos para permitir a IA imitar a forma como os seres humanos aprendem, melhorando gradualmente a sua precisão

[10]. O conceito básico da ML envolve a utilização de métodos estatísticos de treino e otimização que permitem aos computadores analisar conjuntos de dados e identificar padrões [11].

As técnicas de ML aproveitam métodos de *Data Mining* para identificar tendências históricas para informar modelos diferentes. Um algoritmo típico de *Machine Learning* supervisionada consiste aproximadamente de três componentes:

- Um processo de decisão: uma receita de cálculos ou outros passos que recebe os dados e prevê que tipo de padrão o seu algoritmo procura encontrar;
- Uma função de erro: um método de medir o quão boa foi a previsão comparando-a com exemplos conhecidos, permitindo assim avaliar a exatidão do modelo;
- Um processo de otimização: um método pelo qual o algoritmo observa o erro e modifica o modo como o mecanismo de tomada de decisão chega à conclusão.

Dentro da ML, existem diversas abordagens, cada uma com as suas próprias técnicas e algoritmos específicos. Por exemplo, a *Supervised Learning*, é definida pelo uso de *datasets* rotulados, consistindo de pares de entradas e saídas correspondentes, para treinar algoritmos que classificam dados ou preveem resultados. À medida que os diferentes dados são introduzidos no modelo, este ajusta os seus pesos até estarem devidamente ajustados. Isto é o resultado de um processo de *cross validation* com o objetivo de garantir que o modelo evite o *overfitting* e *underfitting* [10].

Outro método é a *Unsupervised Learning* que ao contrário do *Supervised Learning* não há supervisão externa. Isso significa que não temos os pares de entrada e saída correspondentes nos dados. Em vez disso o algoritmo é deixado por conta própria para encontrar padrões e estruturas nos dados [10].

Uma abordagem intermediária é a *Semi-supervised Learning*, aqui, são utilizados conjuntos de dados rotulados, mas também uma grande quantidade de dados não rotulados. Em muitos casos rotular dados pode ser caro ou demorado. A *Semi-Supervised Learning* oferece uma solução elegante para isso, pois aproveita-se do pequeno conjunto de dados rotulados para orientar o processo de treino do algoritmo, enquanto ele explora os dados não rotulados para encontrar mais informações [10].

Além disso, a *Reinforcement Learning* envolve a interação dos sistemas com um ambiente dinâmico, recebendo recompensas ou punições com base nas suas ações, permitindo que os algoritmos aprendam através de um sistema de tentativa e erro [10].

Existe também o conceito de *Deep Learning*, que é uma área mais recente da ML que aprende automaticamente a partir de conjuntos de dados sem introduzir regras ou conhecimentos humanos e irá ser abordada posteriormente na secção 2.3.2 [11].

2.3.1 Algoritmos de Machine Learning

O campo de ML, embora um subcampo da IA, é bastante extenso, incluindo diversos algoritmos de análise de dados. Existe uma série de algoritmos usados pelas empresas de tecnologias modernas, onde cada um destes algoritmos pode ter inúmeras aplicações numa variedade de contextos nomeadamente na previsão de resultados.

Linear Regression

A Regressão Linear é uma técnica simples e poderosa frequentemente usada como ponto de partida em problemas de previsão. Esta técnica fornece uma maneira de entender a relação entre duas variáveis, onde uma é considerada a variável independente, a variável que se pretende prever, e a outra é a variável dependente, a que está a ser usada para a previsão [11].

Na sua forma mais básica, a Regressão Linear assume que a relação entre as variáveis é linear, o que significa que os dados podem ser aproximados por uma linha reta (Figura 2.2). No entanto, a Regressão Linear também pode ser estendida para modelar relações mais complexas usando técnicas como a Regressão Polinomial, onde uma linha reta é substituída por uma curva polinomial de grau superior (Figura 2.2).

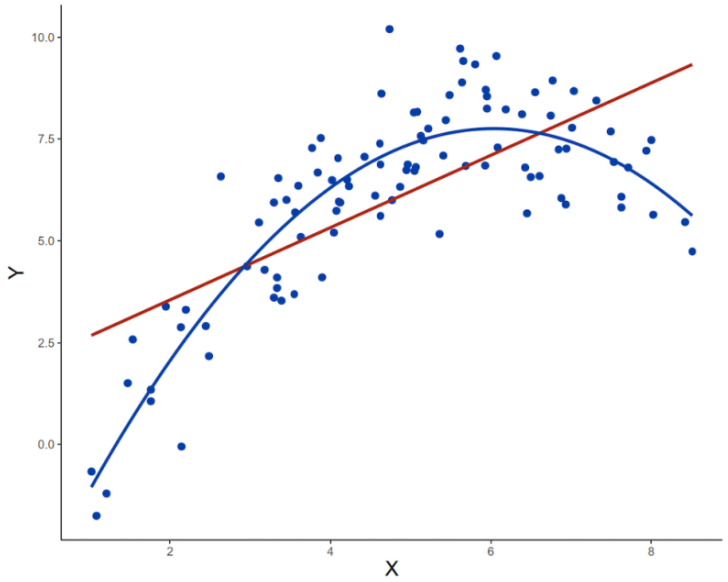


Figura 2.2: Regressão Linear (Linha Vermelha) e Regressão Polinomial (Linha Azul) [12]

Uma das principais vantagens da Regressão Linear é a sua fácil interpretação. Os coeficientes estimados no modelo fornecem informações sobre a magnitude e direção da relação entre as variáveis. Isso torna a Regressão Linear uma ferramenta útil

para análise de dados. No entanto, este método também possui as suas desvantagens, nomeadamente ao assumir uma relação linear entre as variáveis, esta torna-se sensível a valores extremos e pode não funcionar bem onde os dados são altamente não lineares.

Logistic Regression

A Regressão Logística é um algoritmo de *Supervised Learning* utilizado para problemas de classificação binária, onde o seu propósito é prever a probabilidade da ocorrência de um evento binário. Ao contrário da Regressão Linear ou Polinomial, empregadas para prever valores contínuos, a Regressão Logística foca-se na relação entre variáveis independentes e a probabilidade de um evento específico acontecer [11].

A Regressão Logística utiliza a função sigmoide (Equação (2.1)) para transformar uma combinação linear das variáveis independentes numa probabilidade que varia entre 0 e 1 (Figura 2.3). Esta probabilidade é então usada para fazer a classificação, geralmente usando um limite de decisão de 0,5, onde valores acima deste limite são atribuídos à classe positiva e valores abaixo são atribuídos à classe negativa [13].

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

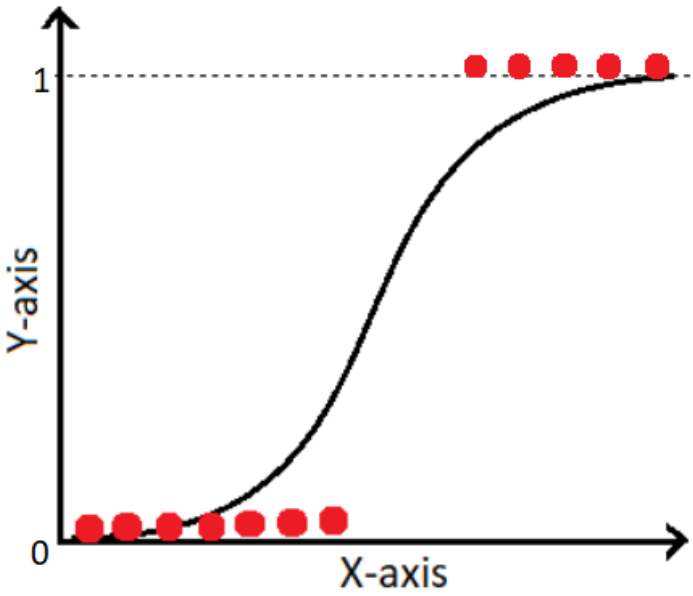


Figura 2.3: Implementação da Regressão Logística [14]

A Regressão Logística é especialmente adequada para conjuntos de dados linearmente separáveis, nos quais uma linha reta pode ser desenhada para separar as duas

classes de dados. Além disso, ela fornece informações valiosas sobre a relevância e a direção da associação entre as variáveis independentes e a variável dependente.

Dado que a regressão logística é mais simples de implementar, interpretar e treinar em comparação com outros métodos de *Machine Learning*, é frequentemente escolhida como uma ferramenta inicial em análises de dados e problemas de classificação binária como, por exemplo, a identificação de "*spam emails*" e determinação da probabilidade de ataques cardíacos [15].

Decision Trees

As *Decision Trees* são um tipo de algoritmo de *Supervised Learning* utilizado para tarefas de classificação e regressão como modelos de previsão. A sua peculiaridade é apresentar uma estrutura hierárquica em forma de árvore, que consiste num nó raiz, ramos, nós internos e nós folha (Figura 2.4).

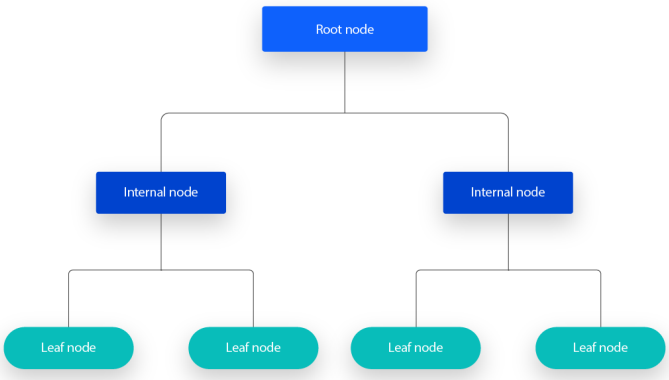


Figura 2.4: Visualização de uma *Decision Tree* [16]

A ideia por trás das *Decision Trees* é dividir repetidamente o conjunto de dados em subconjuntos menores com base em determinadas características, de modo a criar um formato de árvore. Cada nó interno da árvore representa uma condição sobre uma característica, enquanto cada folha representa uma decisão final ou uma classificação. A construção da árvore é realizada de forma recursiva, escolhendo a melhor característica para dividir o conjunto de dados em cada etapa com base em critérios como ganho de informação [16].

Estes algoritmos podem ser divididos em dois tipos principais, as *Classification Trees* e as *Regression Trees*. As *Classification Trees* são utilizadas quando o resultado previsto é uma classe discreta à qual os dados pertencem, e, por outro lado, nas *Regression Trees* as variáveis podem assumir valores contínuos em vez de etiquetas nos nós folha [17].

Random Forest

O *Random Forest*, assim como os algoritmos *Bagging*, *GradientBoosting*, entre outros, são conhecidos como métodos *ensemble*. Estes métodos são constituídos da mesma forma que os algoritmos mais básicos, como a Regressão Linear e as *Decision Trees*, mas possuem uma característica principal que os diferencia, a possibilidade de combinar diferentes modelos para se obter um único resultado. Assim, através disto os algoritmos tornam-se mais robustos e completos, levando a que um maior custo computacional corresponda normalmente a melhores resultados [18].

Random Forest representa uma abordagem popular de ML, cuja autoria é creditada a Leo Breiman e Adele Cutler [19][20]. Este algoritmo combina os resultados de diversas *Decision Trees* para produzir uma única previsão (Figura 2.5). A ampla adoção é impulsionada pela sua facilidade de uso e versatilidade, pois consegue lidar tanto com problemas de classificação como de regressão [21].

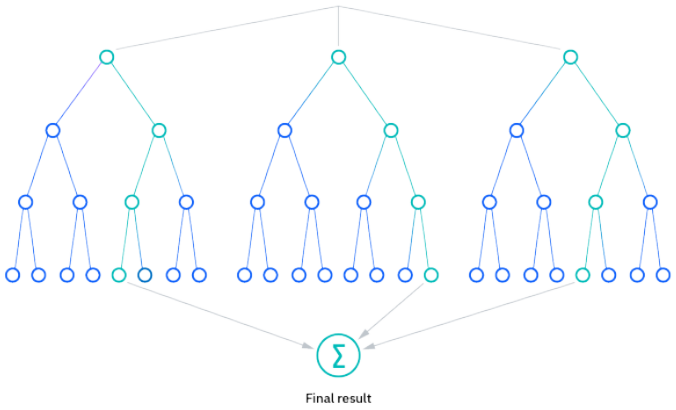


Figura 2.5: Visualização de uma *Random Forest* [21]

Estes algoritmos apresentam três hiperparâmetros principais, os quais devem ser configurados antes do treino: o tamanho do nó, o número de árvores e o número de características amostradas. O algoritmo consiste numa coleção de árvores de decisão, em que cada árvore é construida a partir de uma amostra de dados retirada do conjunto de treino com substituição, conhecida como amostra *bootstrap*. Uma porção desta amostra é reservada como dados de teste, denominada como *out-of-bag* (oob). O método de *bagging* é então empregado para introduzir mais aleatoriedade, aumentando a diversidade do conjunto de dados e reduzindo a correlação entre as árvores de decisão. A forma como a previsão é determinada varia de acordo com o tipo de problema. Para tarefas de regressão, a previsão é calculada como a média das previsões das árvores individuais; enquanto para tarefas de classificação, a previsão é determinada pelo voto maioritário, ou seja, a classe mais frequente entre as árvores de decisão. Por fim, a amostra oob é utilizada para efetuar a *cross-validation*, finalizando o processo de previsão [21].

2.3.2 Deep Learning e Redes Neuronais

O *Deep Learning* é um subcampo de ML que se destaca pelo uso de *Artificial Neural Networks* (ANN) com múltiplas camadas para aprender representações complexas dos dados e simular o complexo funcionamento do cérebro humano. Estas representações permitem que os modelos de *Deep Learning* capturem e compreendam características abstratas dos dados, tornando-os extremamente eficazes numa variedade de tarefas. Uma das melhores características do *Deep Learning* é a sua capacidade de aprender diretamente a partir dos dados brutos, eliminando a necessidade de manipular manualmente os dados. Isto significa que os modelos de *Deep Learning* podem lidar com uma ampla gama de dados, desde texto, imagens até áudio e vídeo, sem a necessidade de pré-processamento dos mesmos. Além disso, estes modelos são capazes de generalizar bem para dados não vistos, ou seja, conseguem fazer previsões mesmo em situações em que se deparam com circunstâncias não encontradas nos seus treinos. Tornando assim estes modelos bastante úteis em cenários no mundo real, onde os dados podem ser bastante imprevisíveis [22].

Como já foi previamente referido, as ANN são o bloco de construção do *Deep Learning*. O conceito de Redes Neuronais foi proposto pela primeira vez em 1944 por dois investigadores da Universidade de Chicago, Warren McCullough e Walter Pitts, este tema foi uma importante área de investigação tanto na neurociência como na informática até 1969, e apresentou outro ressurgimento nos anos 80 devido ao grande aumento na capacidade de processamento dos *chips* gráficos [23].

Uma ANN é um modelo inspirado na organização neuronal existente nas redes biológicas dos cérebros dos animais. Cada Rede Neuronal é constituída por camadas de nós, ou neurónios artificiais: uma camada de entrada (*input layer*), uma ou mais camadas ocultas (*hidden layers*) e por um uma camada de saída (*output layer*) como pode ser visualizado na Figura 2.6.

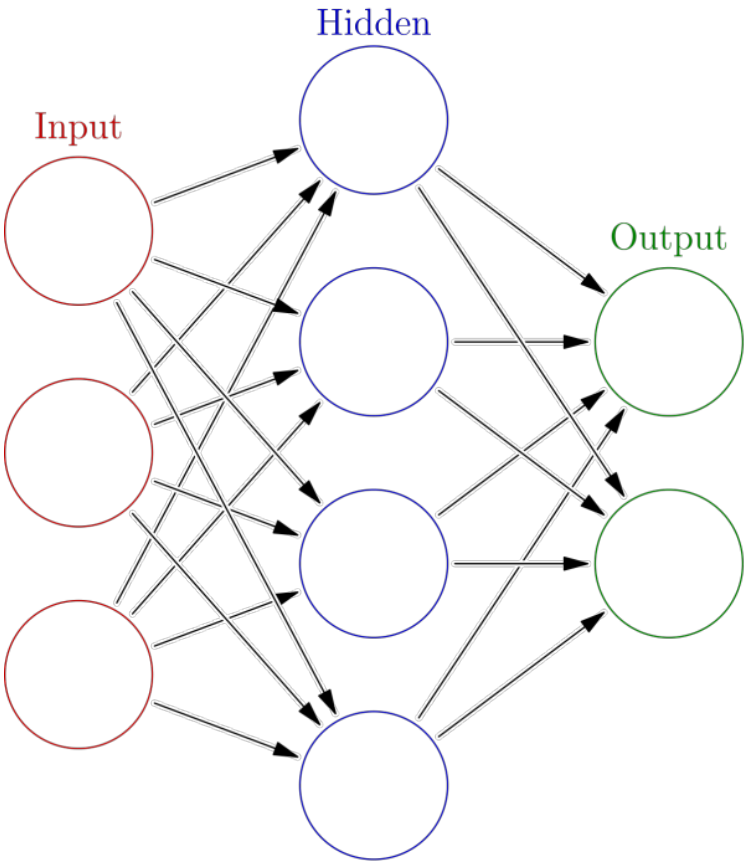


Figura 2.6: Visualização de uma Rede Neuronal [24]

Dentro de uma Rede Neuronal, cada nó é usado como uma unidade de processamento que recebe um conjunto de entradas, realiza uma computação através das mesmas e produz uma saída. Estas entradas são ponderadas por pesos, que são parâmetros ajustáveis da rede e representam a importância relativa de cada entrada para saída do neurónio, que são ajustados durante o treino da rede para que a saída da rede corresponda o mais próximo possível do resultado desejado.

A camada de entrada recebe os dados brutos ou características do problema em questão. Cada nó na camada de entrada corresponde a uma característica específica dos dados de entrada. Por exemplo, se o objetivo for construir uma Rede Neuronal para reconhecer dígitos escritos à mão, cada nó na camada de entrada poderia representar os valores de brilho de cada pixel de uma imagem, exemplificado através da Figura 2.7.

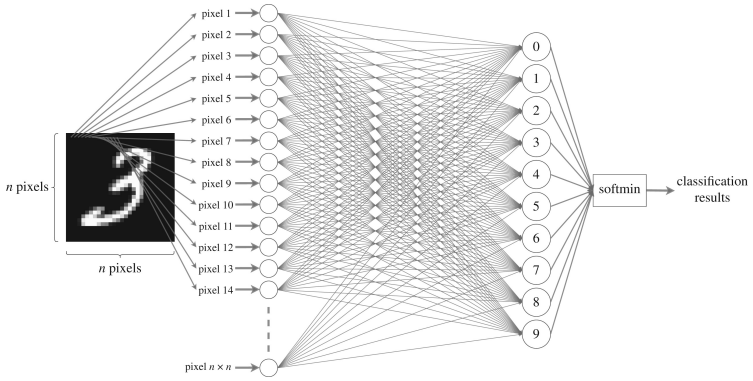


Figura 2.7: Exemplo da camada de entrada [25]

A seguir, as camadas ocultas, como o seu nome sugere, estão ocultas entre as camadas de entrada e saída. Cada nó numa camada oculta recebe entradas da camada anterior, realiza um cálculo através dessas entradas e passa a saída para a próxima camada. A presença das camadas ocultas permite que a rede aprenda representações complexas e abstratas dos dados de entrada.

Por fim, a camada de saída produz as previsões ou resultados da Rede Neuronal. A estrutura e a função da camada de saída dependem do tipo de problema que está a ser abordado. Voltando ao exemplo da previsão dos dígitos manuscritos, como se está a lidar com uma tarefa de classificação de várias classes (10 dígitos possíveis), a camada de saída terá 10 nós, um para cada classe possível.

Cada nó na camada de saída representará a probabilidade de que a entrada pertença a uma classe específica. Isso é tipicamente alcançado através de uma função de ativação `softmax` ou `softmin`, que converte as saídas dos nós numa distribuição de probabilidade sobre as classes. O nó com a maior probabilidade indicará a classe prevista para a imagem [25].

Durante o treino da rede, os pesos de cada nó são ajustados iterativamente usando um algoritmo de otimização, como o algoritmo *Gradient descent*, para minimizar a função de perda. Esta função mede a diferença entre as saídas produzidas pela rede e os resultados desejados. O processo de treino visa encontrar os pesos que minimizam esta função, o que, idealmente, permite que a rede generalize bem para novos dados, ou seja, faça previsões para dados não vistos durante o processo de treino.

Além disso, cada neurónio artificial numa rede recebe múltiplas entradas (x_i), cada uma multiplicada pelo seu peso correspondente (w_i). Estes produtos ponderados são então somados com uma *bias* (b), que é um parâmetro adicional adicionado ao neurónio. Matematicamente, isto pode ser representado pela Equação (2.2) e visualizado através da Figura 2.8.

$$z = \sum_{i=1}^{n} (w_i \cdot x_i) + b \tag{2.2}$$

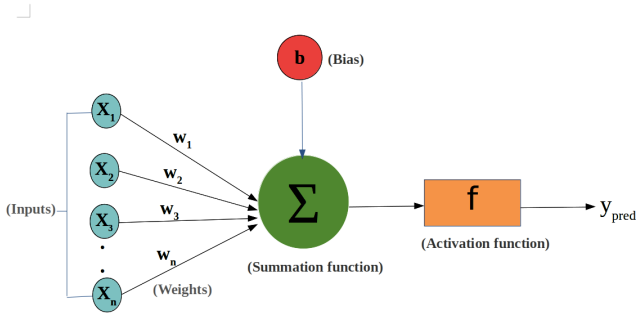


Figura 2.8: Entradas e pesos [26]

O resultado desta soma é a entrada para a função de ativação do neurónio, que são geralmente funções não-lineares, para introduzirem um limiar na saída dentro de uma gama prefixada do estado interno. Estas funções de ativação desempenham um papel crucial no processo de treino da rede, permitindo que ela aprenda e modele relações complexas nos dados. Sem as não-linearidades introduzidas pelas funções de ativação, a capacidade da rede de representar padrões complexos e realizar tarefas sofisticadas seria severamente limitada [27][28]. Existem várias funções de ativação usadas nas redes neuronais. Algumas das mais populares incluem:

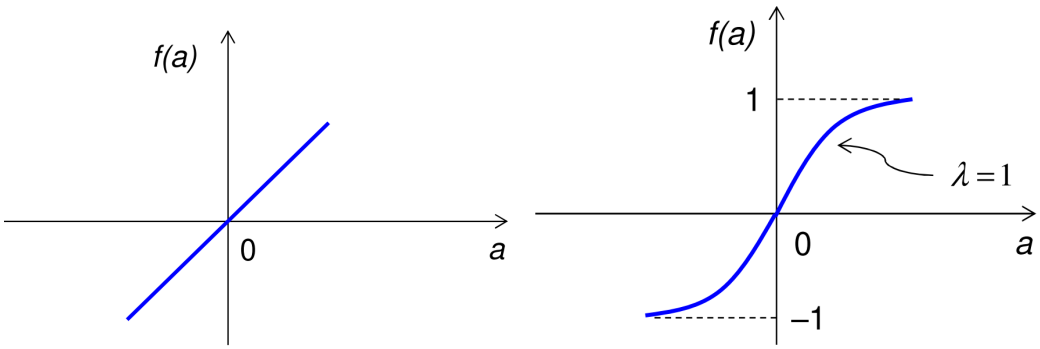
- **Função Linear** (Equação (2.3) e Figura 2.9a): É uma função identidade, onde a saída iguala a entrada. Ao contrário das demais funções, a função não introduz não-linearidade na rede e é geralmente utilizada apenas na camada de saída das redes quando a saída é uma regressão linear [29];

$$f(a) = k \cdot a \tag{2.3}$$

- **Função Sigmoide** (Sigmoid) (Equação (2.1) e Figura 2.3): A função sigmoide mapeia os valores de entrada para um intervalo entre 0 e 1, o que a torna útil para problemas de classificação binária como mencionado previamente;

- **Tangente Hiperbólica** (Tanh) (Equação (2.4) e Figura 2.9b): A função tangente hiperbólica é bastante semelhante à sigmoide, mas mapeia os valores de entrada entre -1 e 1;

$$f(a) = \frac{1 - e^{-\lambda a}}{1 + e^{-\lambda a}} \tag{2.4}$$



(a) (b)

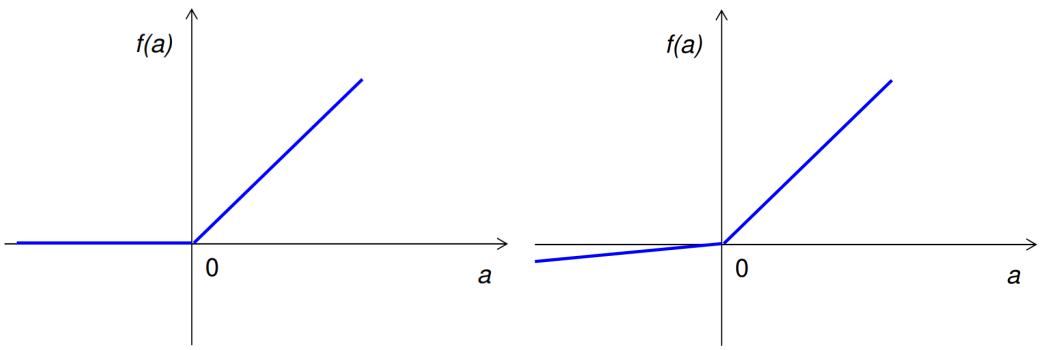
Figura 2.9: (a) Função Linear [27] (b) Função TanH [27]

- **Unidade Linear Retificada** (ReLU) (Equação (2.5) e Figura 2.10a): A função ReLU retorna o valor zero para valores negativos e é linear para valores positivos. Esta função é amplamente utilizada em redes devido à sua simplicidade e eficácia na mitigação do problema de dissipação do gradiente [30];

$$f(a) = \begin{cases} a, & \text{se } a \geq 0 \\ 0, & \text{se } a < 0 \end{cases} \tag{2.5}$$

- **Função Leaky ReLU** (Equação (2.6) e Figura 2.10b): Esta função é uma variante da função anterior que permite um pequeno valor para valores negativos, em vez de zero. Isto ajuda a mitigar a principal desvantagem da função ReLU, que é o facto de redes com esta função não podem aprender em exemplos para os quais a sua ativação é zero [31].

$$f(a) = \begin{cases} a, & \text{se } a \geq 0 \\ \alpha \cdot a, & \text{se } a < 0 \end{cases} \tag{2.6}$$



(a) (b)

Figura 2.10: (a) Função ReLU [27] (b) Função Leaky ReLU [27]

Este processo de treino de uma Rede Neuronal em que os dados de entrada são alimentados através da rede para gerar uma previsão é denominado *Forward Propagation*. Este processo ocorre em todas as camadas da rede, onde os dados fluem da camada de entrada para a camada de saída.

Após a fase de *Forward Propagation*, a rede produz uma saída que é comparada com os resultados desejados para calcular o erro ou a perda. Esta é uma etapa importante, onde é medida a diferença entre as previsões da rede e os valores reais esperados. A escolha da função de perda depende, mais uma vez, do tipo de problema que está a ser abordado.

Uma das opções de funções de perda mais populares é a *Mean Squared Error* (MSE) (Equação (2.7)) que mede o valor médio das diferenças quadráticas entre o valor real e o valor previsto. Os pontos fortes desta função de perda é, visto que calcula a diferença quadrada, não se importar se o valor previsto está a baixo ou a cima do valor alvo, mas, por outro lado, um ponto fraco é ser bastante penalizadora a valores com um grande erro [32].

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i)^2 \tag{2.7}$$

onde n é o número total de valores

x_i é o valor real

y_i é o valor previsto

Outra função bastante utilizada é a *Mean Absolute Error* (MAE) (Equação (2.8)) que determina a média das diferenças absolutas entre o valor real e previsto. Esta função é usada como alternativa à MSE pela sua desvantagem descrita. Assim sendo a MAE é usada quando os dados do treino têm um grande número de valores muito discrepantes atenuando assim este problema [32].

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - x_i| \tag{2.8}$$

onde n é o número total de valores

x_i é o valor real

y_i é o valor previsto

Por fim, para problemas de classificação costumam ser usadas as funções *Categorical Cross-Entropy* (CCE) (Equação (2.9)) e a *Binary Cross-Entropy* (BCE) (Equação (2.10)), onde a função CCE é aplicada a problemas de classificação multiclasse e a BCE é aplicada quando o problema de classificação é binário [32].

$$CCE = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} \cdot \log(p_{ij}) \tag{2.9}$$

onde N é o número total de valores

M é o número de classes

y_{ij} é a probabilidade verdadeira da classe j para o exemplo i

p_{ij} é a probabilidade prevista pela rede para a classe j no exemplo i

$$BCE = \frac{1}{N}\sum_{i=1}^{N} -(y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i)) \tag{2.10}$$

onde N é o número total de valores

y_i é a verdadeira classe binária para o exemplo i

p_i é a probabilidade prevista pela rede para o exemplo i

De seguida, após a fase de *Forward Propagation* e o cálculo da perda, segue-se a fase de *Back Propagation*, onde um algoritmo de otimização, como o *Gradient descent*, é aplicado para ajustar os pesos e os *biases* da rede de forma a minimizar a função de perda. Este ajuste é feito retroativamente, começando pela camada de saída e propagando o erro de volta para as camadas anteriores, a fim de atualizar os pesos de forma iterativa.

Após várias iterações de *Forward Propagation* e *Back Propagation*, juntamente com a atualização dos parâmetros da rede, o processo de treino é quando o modelo atinge um desempenho satisfatório de acordo com os critérios estabelecidos. Em seguida, o modelo treinado pode ser utilizado para fazer previsões sobre novos conjuntos de dados, aplicando o conhecimento adquirido durante o treino para resolver problemas do mundo real.

É importante ressaltar que o sucesso do treino de uma Rede Neuronal depende não apenas da escolha adequada da arquitetura da rede e das escolhas de funções de ativação, mas também da seleção cuidadosa da função de perda e do algoritmo de otimização. Experimentar diferentes combinações desses elementos e ajustar os

hiperparâmetros da rede é uma parte essencial do desenvolvimento de modelos de ML eficazes.

2.4 Exemplos

Nesta secção, irão ser explorados os avanços no campo da previsão do *Stock Market* através de técnicas de ML. Para entender as práticas mais atuais e eficazes neste domínio, serão analisados principalmente dois artigos, o "*News-based intelligent prediction of financial markets using text mining and machine learning: A systematic literature review*" de Matin N. Ashtiani e Bijan Raahemi [33], e o "A Survey of Forex and Stock Price Prediction Using Deep Learning" de Zexin Hu, Yiqi Zhao e Matloob Khushi [34].

Através destes artigos consegue-se obter uma noção dos diferentes algoritmos de ML que estão a ser usados na área. Estes artigos fizeram uma seleção de diversos artigos que abordam o tema de várias bibliotecas *online* como Scopus, IEEE Xplore e ScienceDirect, e dividiram-nos conforme o método de previsão utilizado. A análise destes estudos foi complementada pela apresentação de gráficos elucidativos (Figuras 2.11, 2.12 e 2.13).

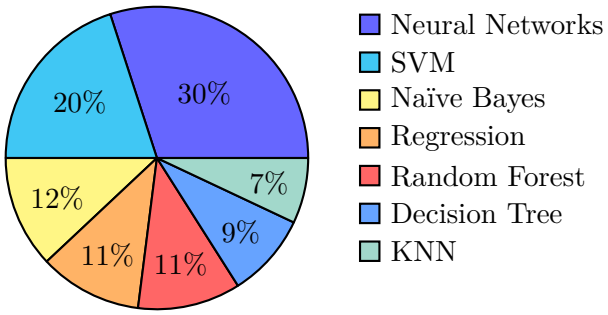


Figura 2.11: Distribuição de modelos de *Supervised Learning* [33]

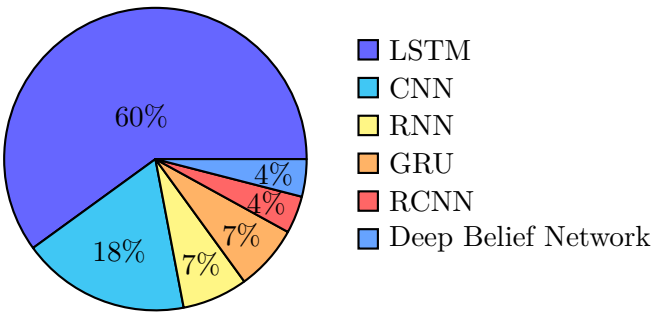


Figura 2.12: Distribuição de modelos de Redes Neuronais [33]

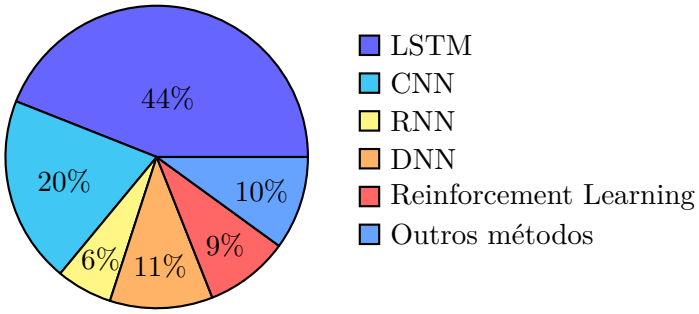


Figura 2.13: Distribuição de modelos de *Machine Learning* [34]

Assim, ao analisar estes artigos conclui-se que as Redes Neuronais são os modelos de *Machine Learning* mais usados, e dentro destes modelos, as redes *Long Short-Term Memory* (LSTM) seguidas das redes *Convolutional Neural Network* (CNN).

De seguida também foi consultado o artigo "Machine learning techniques and data for stock market forecasting: A literature review" de Mahinda Kumbure, Cristoph Lohrmann, Pasi Luukka e Jari Porras [35].

Este estudo, juntamente com os demais mencionados, realiza uma revisão da literatura na área e oferece informações mais específicas sobre os fundos e empresas utilizados na previsão do valor das suas ações. Destaca-se que o S&P 500, um índice de mercado que acompanha o desempenho das ações de 500 das maiores empresas nas bolsas de valores dos Estados Unidos, é o mais frequentemente previsto, seguido pelos índices TAIEX, NASDAQ, SSE e DJI. Relativamente às empresas, a mais prevista é a *Apple Inc.*, seguida pela *Microsoft Corporation* e antigo *Facebook, Inc.*, agora *Meta Platforms, Inc.*.

Para além de fornecer informação sobre os ativos mais previstos, este estudo também oferece uma compreensão das variáveis de entrada mais utilizadas nestes modelos, conforme ilustrado na Figura 2.14, que apresenta as variáveis base, tais como "*Close Price*", "*High Price*", "*Low Price*", "*Open Price*" e "*Volume*".

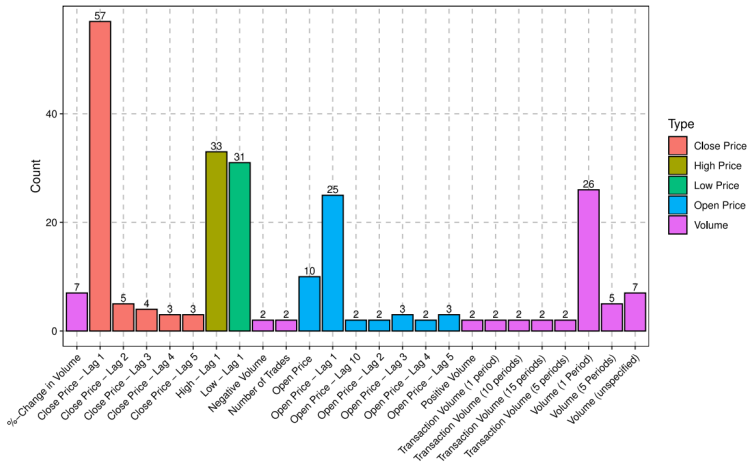


Figura 2.14: Variáveis Base [35]

Este artigo também apresenta diversos indicadores técnicos, como as mencionadas linhas de tendências e médias móveis, apresentados na Figura 2.15. Através da figura consegue-se perceber que os indicadores mais utilizados são RSI (14 períodos), SMA (10 períodos) e MACD.

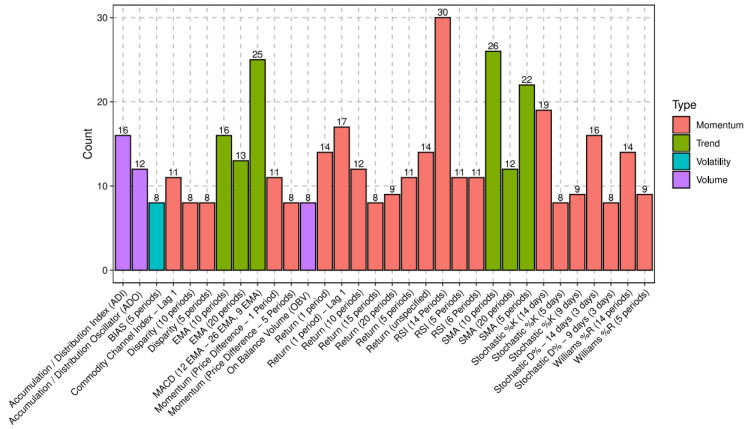


Figura 2.15: Top-30 Indicadores Técnicos [35]

Seguidamente, este artigo também nos disponibiliza um apanhado das *commodities* mais relevantes para o mercado financeiro. As *commodities* desempenham um papel crucial na economia global, sendo matérias-primas essenciais para várias indústrias. Entre as *commodities* mais negociadas e acompanhadas estão o petróleo bruto, o ouro, a prata, o trigo e o café, entre outros. O comportamento dessas *commodities* pode ser influenciado por uma série de fatores assim como o mercado das ações, sendo assim, é útil levar em consideração a alteração de preço destas matérias-primas quando temos como objetivo prever o preço de ações de empresas que as usam. Ao interpretar este artigo, verifica-se que a alteração do preço do petróleo bruto seguido da alteração do preço do ouro são as *commodities* mais utilizadas (Figura 2.16).

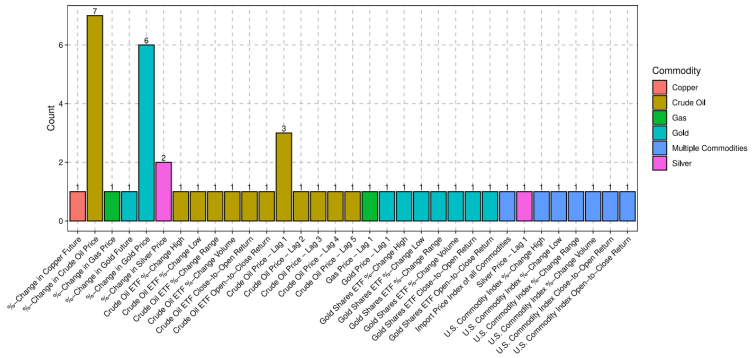


Figura 2.16: *Commodities* mais utilizadas [35]

Por fim, também temos acesso a um conjunto das mais utilizadas taxas de juros e dados relativos à massa monetária, como *Treasury Bills*, títulos de dívida de curto

prazo emitidos pelo governo dos Estados Unidos [36]; *Certificate of Deposit*, que são produtos financeiros oferecidos por bancos, como taxas de juros sobre o capital depositado; e *Term Spread*, ou seja a diferença entre taxas de juros de títulos de dívida com diferentes vencimentos.

Ao analisar a Figura 2.17, é possivel verificar que o dado mais utilizado para a previsão do mercado das ações é o *3-Month Treasury Bill*, seguido do *10-Year Treasury Note* e *5-Year Treasury Note*.

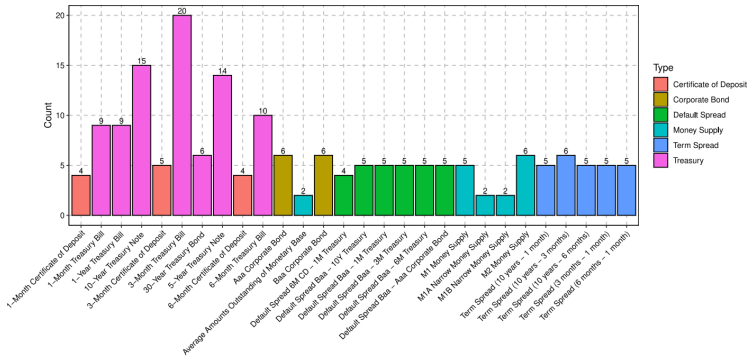


Figura 2.17: Taxas de juros e massa monetária [35]

2.4.1 Abordagem Para a Previsão do Preço das Ações Através de LSTM Sequencial de Várias Camadas

Nesta subsecção, será abordado um estudo realizado por Abdul Quadir Md, Sanjit Kapoor, Chris Junni A.V., Arun Kumar Sivaraman, Kong Fah Tee, Sabireen H., e Janakiraman N., intitulado 'Novel optimization approach for stock price forecasting using multi-layered sequential LSTM' [37].

Este estudo, utiliza um modelo de Rede Neuronal conhecido como *Multi-Layer Sequential Long Short-Term Memory* (MLS LSTM) para prever os preços de ações. Esta abordagem começa com a preparação dos dados, através da utilização de um conjunto de dados que contém informações sobre os preços das ações *Samsung Electronics Co., Ltd.* entre 2016 e 2021. Estes dados são então normalizados através da técnica `MinMaxScaler`, dimensionando os dados num intervalo entre 0 e 1.

Após a preparação dos dados, o modelo de previsão é conseguido. O modelo, como já foi referido, é baseado numa arquitetura de Rede Neuronal conhecida como MLS LSTM. Esta arquitetura é escolhida devido à sua capacidade de lembrar padrões em sequências longas de dados temporais, tornando-a adequada para a previsão de longas séries de dados como os preços das ações.

O modelo construido consiste em quatro camadas, três das quais são camadas LSTM e a quarta é uma camada densa. As camadas LSTM são responsáveis por processar a sequência temporal dos dados, enquanto a camada densa é responsável por gerar a saída final do modelo.

Após a construção do modelo, ele é compilado e ajustado aos dados de treino e teste. Durante o processo de treino, o modelo é ajustado para minimizar a função de perda, utilizando o otimizador Adam, um tipo de algoritmo de otimização *Stochastic gradient descent*, e avaliado através de métricas como R^2, RMSE e MAPE.

Os resultados do estudo demonstram que o modelo MLS LSTM supera as outras abordagens de previsão de séries temporais, como a Regressão Linear e SVM, em termos de precisão e desempenho. Isto apresenta sugestões que a abordagem MLS LSTM é altamente eficaz na previsão de preços de ações.

Por fim, este estudo servirá como base para o desenvolvimento de um modelo de previsão de preços de ações, onde forneceu informações importantes relativamente a todos os passos na realização de um projeto como este, desde a escolha de um conjunto de dados, técnicas de normalização dos mesmos, construção de um modelo e por fim avaliação através de um conjunto de métricas previamente definidas.

Capítulo 3

Algoritmos

Neste capítulo, irão ser explorados os algoritmos fundamentais utilizados na área de *Machine Learning* usados neste projeto, com foco particular nas suas aplicações e características. Para além disso também serão apresentadas um conjunto de métricas de desempenho utilizadas para medir o desempenho destes mesmos algoritmos.

3.1 Algoritmos de *Machine Learning* Usados

No campo do *Machine Learning*, é usada uma grande variedade de algoritmos para resolver diferentes tipos de problemas. Nesta secção irão ser apresentados todos os algoritmos utilizados neste projeto com o objetivo de prever os preços do mercado de ações. A maioria destes algoritmos foram escolhidos para serem aplicados neste projeto com base principalmente na pesquisa efetuada e apresentada na Secção 2.4.

3.1.1 *Recurrent Neural Network*

Uma *Recurrent Neural Network* (RNN) é um tipo de rede neuronal que processa dados sequenciais ou séries temporais. Estes algoritmos são mais utilizados em problemas que envolvem ordem ou tempo, como efetuar traduções entre linguagens, *Natural Language Processing* (NLP), reconhecimento de voz e imagem. São integrados em aplicações amplamente conhecidas, tais como assistentes de voz como a famosa Siri da *Apple Inc.* e o Google Translate. Assim como já foi apresentado na Secção 2.3.2, as RNN também utilizam dados de treino para aprender. Mas apresentam uma particularidade, a capacidade de possuírem "memória", uma vez que são

capazes de utilizar informações de entradas anteriores para influenciar a entrada e saídas atuais (Figura 3.1). Esta "memória" surge do facto das RNN não assumirem que as entradas e saídas são independentes umas das outras, possibilitando assim que as saídas das RNN dependam dos elementos anteriores da sequência. Outra característica distintiva destas redes é o facto destas partilharem parâmetros em cada camada da rede, enquanto as redes denominadas de *feedforward* têm pesos diferentes em cada nó, as RNN partilham o mesmo parâmetro de peso em cada camada da rede [38].

As RNN utilizam o método de *Backpropagation Through Time* (BPTT) para calcular os gradientes. Os princípios subjacentes ao BPTT são análogos aos da *backpropagation* tradicional, onde o modelo se autoajusta ao calcular os erros desde a sua camada de saída até a sua camada de entrada, o BPTT difere do método tradicional ao somar os erros em cada passo de tempo. Através deste processo estas redes enfrentam problemas conhecidos como "*exploding gradients*" e "*vanishing gradients*", que ocorrem devido ao tamanho do gradiente. Quando o gradiente é demasiado pequeno, continua a diminuir até se tornar insignificante, impedindo assim o treino do algoritmo. Por outro lado, quando o gradiente é muito grande, o modelo torna-se instável e os pesos podem crescer demasiado. Para solucionar estes problemas, é comum reduzir-se o número de camadas ocultas dentro da RNN, diminuindo a complexidade do modelo [38].

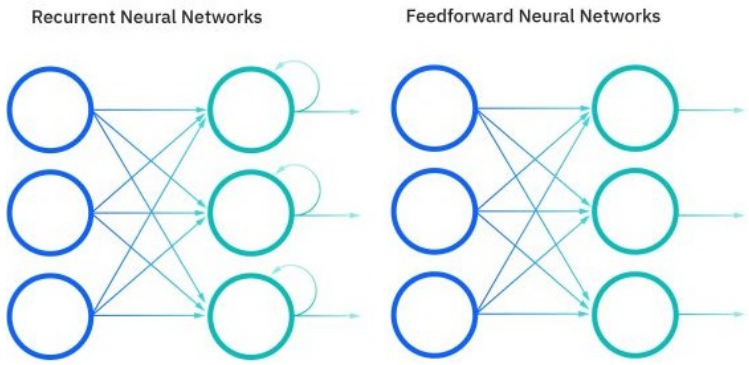


Figura 3.1: *Recurrent Neural Network* e *Feedforward Neural Network* [38]

A unidade de processamento fundamental de uma RNN é a *Recurrent Unit* (Figura 3.2). Esta unidade tem a capacidade única de manter um estado oculto, permitindo assim, que a rede capture dependências sequenciais ao se lembrar de entradas anteriores durante o processo de treino.

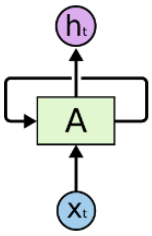


Figura 3.2: *Recurrent Unit* [39]

Na Figura 3.2 estão representados a *Recurrent Unit* como A, um valor de *input* como x_t, e um valor de *output* h_t, e um *loop* que permite que a informação possa ser passada de um passo da *Neural Network* para outro, sendo estes *loops* responsáveis por carregarem a informação de uma *Recurrent Unit* para outra. Como é possível visualizar na Figura 3.3 o valor transmitido pelo *loop* referido anteriormente, está a influenciar diretamente a ativação da próxima *Recurrent Unit*.

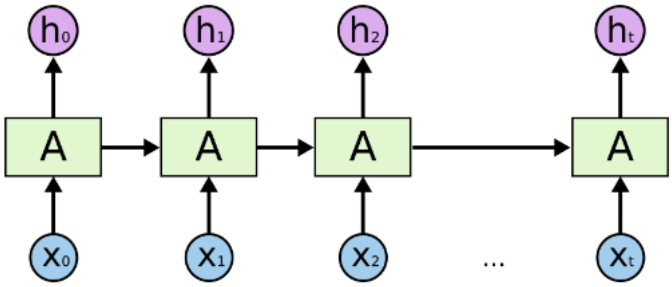
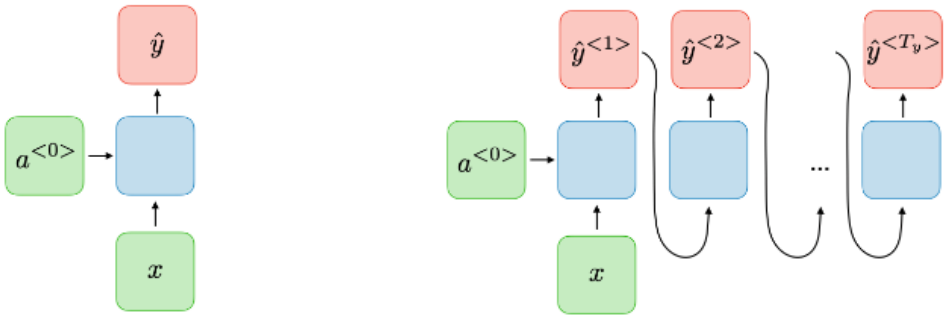


Figura 3.3: Exemplo de uma RNN [39]

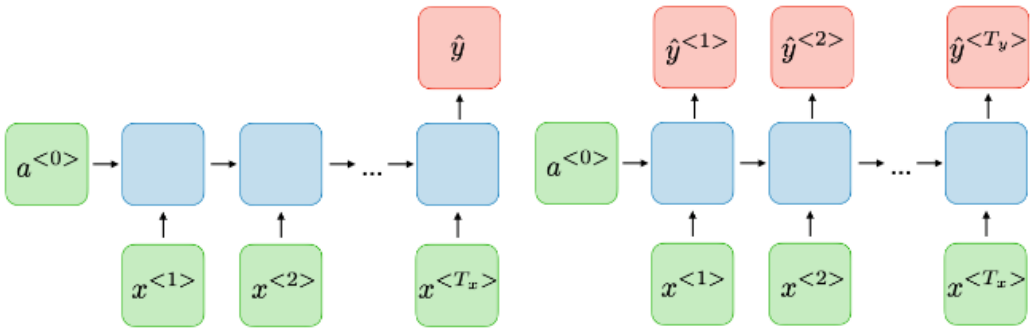
As RNN possuem diversas configurações da sua arquitetura usadas em diferentes cenários. Começando pela arquitetura One-to-one (Figura 3.4a), esta arquitetura é a arquitetura de uma rede neuronal convencional, e é utilizada para tarefas onde não há dependências nos dados. De seguida, na arquitetura One-to-many (Figura 3.4b) é usada uma única entrada para gerar uma sequência de saídas. Um exemplo comum onde é integrada esta arquitetura é na geração de legendas para imagens, onde uma única imagem é fornecida como entrada e a rede gera uma sequência de palavras para a descrever. Já na Figura 3.5a é apresentada a arquitetura Many-to-one, aqui, uma sequência de entradas é usada para produzir uma única saída, este estilo de arquitetura é mais utilizado em tarefas de classificação de sequências, onde a *Neural Network* recebe uma sequência de dados para produzir uma classificação ou um rótulo. Por fim, temos a arquitetura Many-to-many representada na Figura 3.5b. Nesta arquitetura, tanto a entrada como a saída são sequências. Existem duas variações comuns desta arquitetura: Many-to-many com correspondência temporal exata (Figura 3.5b) e Many-to-many com correspondência temporal deslocada. Na primeira variação, cada entrada corresponde diretamente com uma saída na mesma

posição temporal. Já na segunda variação, a entrada e a saída podem ter diferentes comprimentos e podem estar alinhadas de uma maneira diferente no tempo. Um exemplo onde a primeira variação é utilizada, seria a tradução de sequências, onde cada palavra de entrada corresponde a uma palavra de saída na mesma posição. Já um exemplo da segunda variação seria a tradução automática, onde muitas vezes o número de palavras de entrada não corresponde diretamente com o número de palavras na saída.



(a) (b)

Figura 3.4: (a) *One-to-one* [40] (b) *One-to-many* [40]



(a) (b)

Figura 3.5: (a) *Many-to-one* [40] (b) *Many-to-many* [40]

As RNN têm sido objeto de desenvolvimentos contínuos para lidar com os desafios de longo prazo de dependências sequenciais e para lidar melhor com os problemas de gradiente. Para isto, uma das extensões mais significativas são as *Long Short-Term Memory*, propostas para superar os problemas de "*exploding gradients*" e "*vanishing gradients*".

3.1.2 *Long Short-Term Memory*

As *Long Short-Term Memory* (LSTM) são um tipo de RNN que se especializam em incorporar dependências a longo prazo. Este tipo de tecnologia foi introduzida por Sepp Hochreiter e Jürgen Schmidhuber no ano de 1997 [41].

Esta tecnologia apresenta uma arquitetura bastante parecida à de uma RNN a um nível alto, onde a rede LSTM é composta por três partes, a *Forget Gate*, a *Input Gate* e a *Output Gate* (Figura 3.6). A primeira parte, a *Forget Gate*, é responsável por escolher informações do registo temporal anterior, podendo então decidir utiliza-la ou esquecê-la. A segunda parte, a *Input Gate*, é responsável por assimilar novas informações. E por fim a terceira parte, a *Output Gate* é responsável por passar a nova informação para a célula seguinte [42].

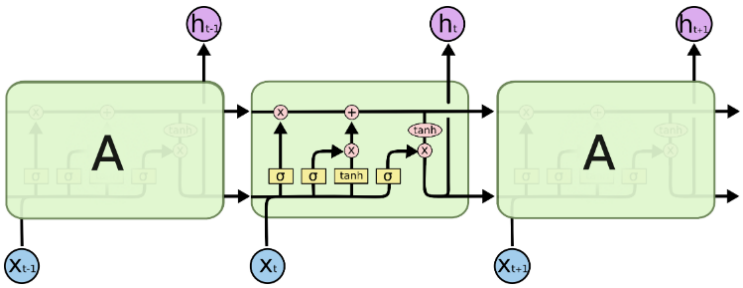


Figura 3.6: Arquitetura de uma LSTM [39]

Para além disso, o *Cell State* e o *Hidden State*, as duas linhas horizontais na Figura 3.6, são conceitos fundamentais no funcionamento das redes LSTM. O *Cell State*, também conhecido como memória de longo prazo, é responsável por armazenar e transportar informações ao longo do tempo. Ele é atualizado através de diferentes *Gates*, mencionadas previamente. Por outro lado, o *Hidden State*, ou memória a curto prazo, captura informações relevantes do momento atual. Ele é gerado através da combinação de informações a longo prazo e informações da entrada atual, permitindo assim que a rede consiga incorporar também uma componente a curto prazo, fazendo com que a rede capture padrões temporais complexos e tome decisões adequadas em tarefas de previsão.

Como já foi referido, o primeiro passo numa rede LSTM é decidir que informação devemos utilizar ou esquecer. Esta decisão é tomada por uma função `sigmoid` (Figura 3.7 e Equação (3.1)), a qual função olha para o valor do *Hidden State*, h_{t-1}, e do valor de entrada da célula, x_t, e apresenta como *output* um valor entre 0 e 1 para cada número no *Cell State*, onde um 1 representa utilização total do valor e o valor 0 representa que esse valor vai ser completamente esquecido.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3.1}$$

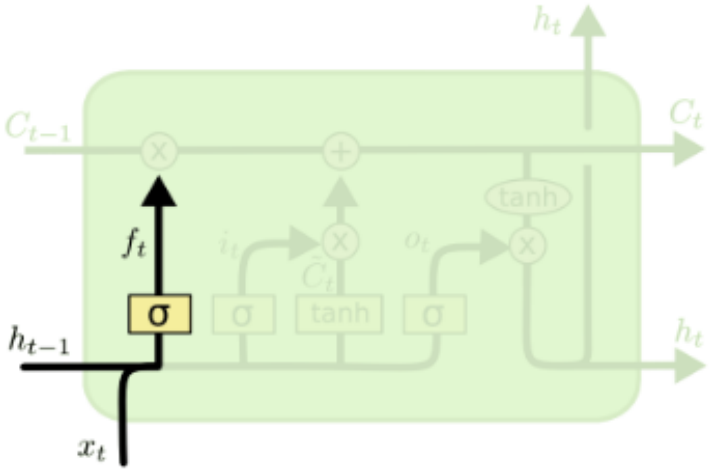


Figura 3.7: *Forget Gate* [39]

De seguida, é preciso decidir quais novas informações devem ser guardadas no *Cell State* (Figura 3.8). Para isto, são usadas duas funções distintas, apresentadas previamente, a função `sigmoid` (Equação (3.2)) e a função `TanH` (Equação (3.3)). A primeira função serve para decidir quais dos valores vão ser atualizados e a segunda é utilizada para criar um vetor de novos valores candidatos que poderão ser adicionados ao estado.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3.2}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{3.3}$$

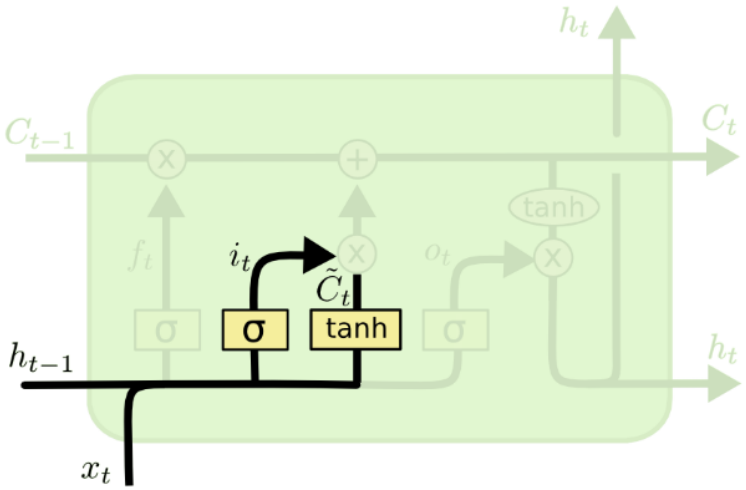


Figura 3.8: *Input Gate* [39]

O seguinte passo, será então atualizar o antigo *Cell State* através do valor novo (Figura 3.9). Para isto basta seguir a Equação (3.4), onde o valor f_t é multiplicado com o antigo *Cell State* e sequentemente somado ao produto entre i_t e $\tilde{C}_t$.

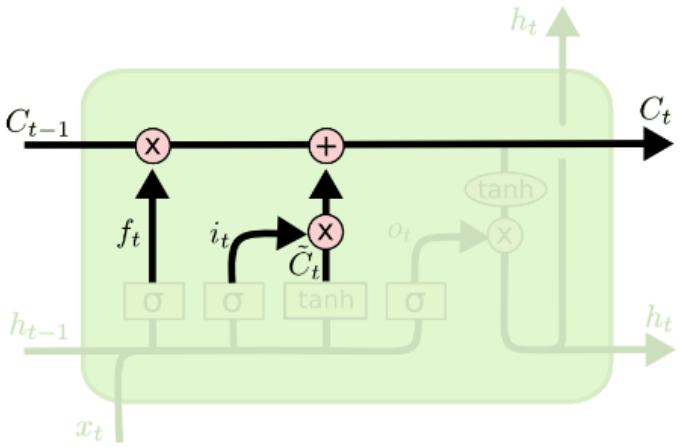


Figura 3.9: Atualização do *Cell State* [39]

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t \tag{3.4}$$

Por fim, o último passo é decidir o valor de *output* da célula (Figura 3.10). Para isto, esta saída irá ser baseada no *Cell State* atual, onde é utilizada a função sigmoide que irá decidir a percentagem do valor do *Cell State* a ser utilizado (Equação (3.5)). Após isto, é multiplicado o valor resultante da função sigmoide pelo *Cell State* obtendo assim o valor h_t, o *output* de uma célula (Equação (3.6)).

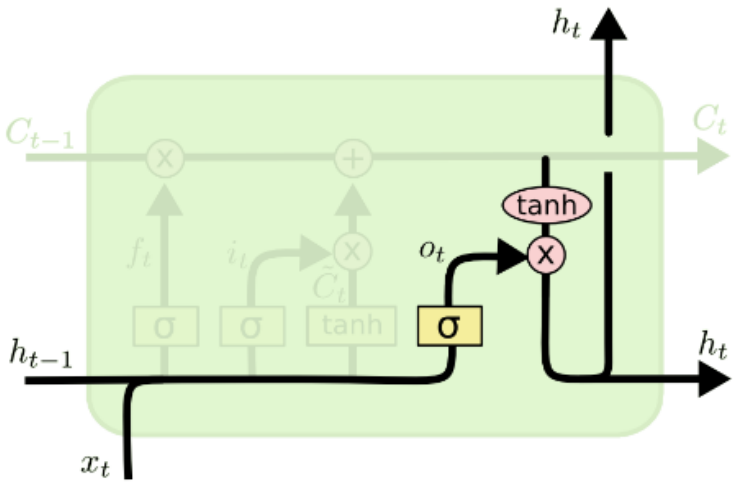


Figura 3.10: *Output* da célula [39]

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{3.5}$$

$$h_t = o_t * \tanh(C_t) \tag{3.6}$$

3.1.3 *Gated Recurrent Unit*

Gated Recurrent Unit (GRU) (Figura 3.11) é um tipo de RNN introduzida em 2014 por Kyunghyun Cho [43] como uma alternativa às LSTM. As redes GRU são utilizadas, assim como as LSTM, para processar dados sequenciais. Estas utilizam mecanismos de *gating* para atualizar seletivamente o *Hidden State* da rede em cada passo de tempo, com dois principais: a *Reset Gate* e a *Update Gate*. Estes mecanismos, assim como as diferentes *Gates* nas redes LSTM, controlam o fluxo de informação, decidindo quando o *State* anterior deve ser esquecido e quanto da nova entrada deve ser utilizada para atualizar o *Hidden State* [44].

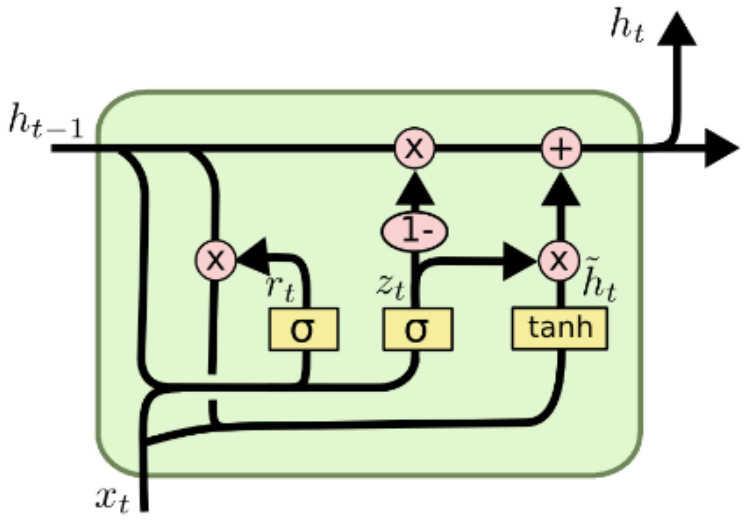


Figura 3.11: Gated Recurrent Unit [39]

Assim como as LSTM, as redes GRU foram também concebidas para resolver os problemas de gradiente apresentados na Secção 3.1.1. Mas ao contrário da LSTM, as redes GRU consistem de três *Gates* e não mantém um *State* interno da célula. As diferentes portas de uma rede GRU são:

- ***Update Gate*** (Equação (3.7)): Determina quanto do conhecimento passado precisa de ser transmitido para o futuro, similar à porta de saída de uma LSTM;

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \tag{3.7}$$

- ***Reset Gate*** (Equação (3.8)): Decide quando do conhecimento passado deve ser esquecido, similar à combinação da *Input Gate* e *Forget Gate* de uma LSTM;

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \tag{3.8}$$

- ***Current Memory Gate*** (Equação (3.9) e (3.10)): Está incorporada à *Reset Gate*. Esta *Gate* introduz não-linearidade na entrada e torna-a *Zero-mean*,

reduzindo o efeito de informação anterior sobre a informação atual que está a ser transmitida para o futuro.

$$\widetilde{h_t} = \tanh(W \cdot [r_t * h_{t-1}, x_t]) \tag{3.9}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \widetilde{h_t} \tag{3.10}$$

3.1.4 *Convolutional Neural Network*

Outro tipo de algoritmo de *Machine Learning* que pode ser usado para a previsão do mercado de ações são as *Convolutional Neural Network* (CNN). Tradicionalmente as CNN são usadas no processamento de imagens e vídeos, sistemas de recomendação, NLP, entre outros, e comparativamente às previamente apresentadas RNN são a opção menos popular na previsão do mercado de ações. Mas, mesmo assim, as CNN têm sido exploradas em algumas abordagens para a previsão do mercado de ações. Uma das razões para isso é a sua capacidade de capturar padrões espaciais em dados multidimensionais, o que pode ser útil ao analisar dados de séries temporais, como os preços das ações.

Uma CNN pode ser dividida em três partes principais, as *Convolutional Layers*, *Pooling Layers* e *Fully Connected Layers* (Figura 3.12). Começando pelas *Convolutional Layers*, estas camadas são a principal característica das CNN. A convolução é uma operação matemática que consiste em deslizar um filtro sobre os dados de entrada, multiplicando os valores do filtro pela região correspondente dos dados e somando os resultados. Nestas camadas, são aplicados vários filtros de tamanho igual, onde cada filtro é usado para reconhecer um padrão específico da imagem, como a curvatura dos dígitos se tomássemos como exemplo o objetivo de reconhecer dígitos escritos à mão.

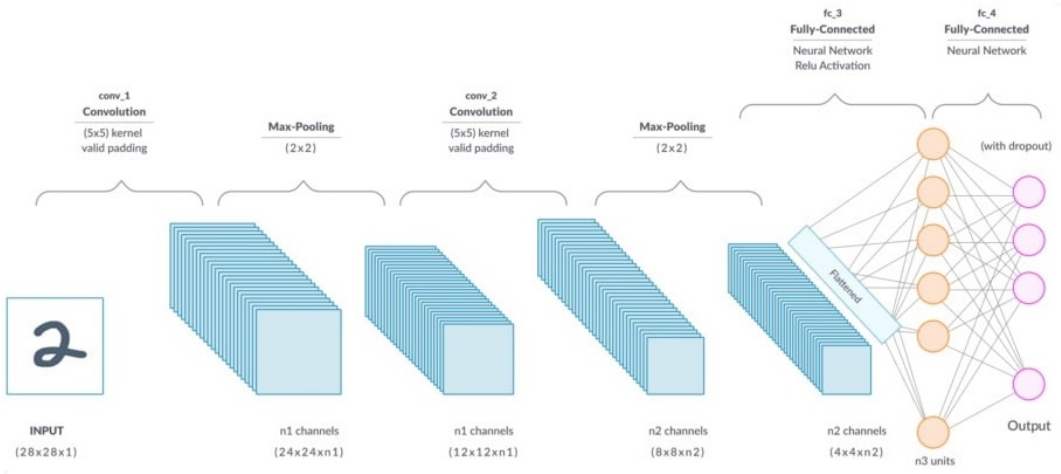


Figura 3.12: *Convolutional Neural Network* [45]

Ao aplicar a operação de convolução, cada filtro é representado por uma matriz de pesos. Durante a operação, o filtro é sobreposto à imagem de entrada e uma multiplicação elementar é realizada em cada posição (Equação (3.11)). Esta multiplicação é seguida pela soma dos resultados, resultando num único valor na saída, chamado de ativação.

$$C[i,j] = \sum_m \sum_n I[i+m, j+n] \times K[m,n] \tag{3.11}$$

onde $C[i,j]$ representa o valor da ativação na posição (i,j) da imagem de saída
$I[i+m, j+n]$ valor do pixel na posição $(i+m, j+n)$ da imagem de entrada
$K[m,n]$ é o peso do filtro na posição (m,n)

De seguida, as *pooling layers* efetuam a redução da dimensão, reduzindo assim o número de parâmetros na entrada. Semelhante à *convolutional layer*, a operação de *pooling* passa um filtro na entrada toda, onde a diferença é que este filtro não possui pesos. Em vez disso, é aplicada uma função de agregação aos valores, sendo estas `Max pooling` e `Average pooling`.

No caso do `Max pooling`, por exemplo, para cada região, apenas o valor máximo é mantido como saída. Esta operação é descrita pela Equação (3.12).

$$O[i,j] = \max_{m,n} I[i+m, j+n] \tag{3.12}$$

onde $O[i,j]$ representa o valor de saída para a região (i,j)
$I[i+m, j+n]$ valor da região na posição $(i+m, j+n)$ no mapa de entrada

Para o `Average pooling`, por sua vez, o valor de saída é a média de todos os valores da região e é descrito através da Equação (3.13).

$$O[i,j] = \frac{1}{m \times n} \sum_{m,n} I[i+m, j+n] \tag{3.13}$$

onde $m \times n$ é o tamanho da região de *pooling*

Estas operações são aplicadas independentemente em cada região do mapa de características, reduzindo a sua dimensão e preservando as informações mais relevantes. Esta redução de dimensão ajuda a diminuir a carga computacional e reduzir

o *overfitting* durante o processo de treino da rede.

Após a aplicação das *convolutional* e *pooling layers*, os mapas de características resultantes são transformados em vetores unidimensionais e alimentados às *fully connected layers*. Estas camadas são semelhantes às camadas de uma *neural network* tradicional, onde cada neurónio está conectado a todos os neurónios da camada anterior (Figura 3.13).

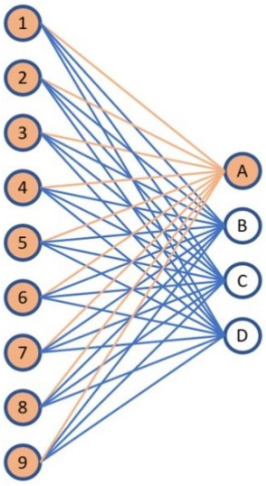


Figura 3.13: *Fully Connected Layer* [46]

A função destas camadas é aprender representações discriminativas dos dados que são úteis para a tarefa final, como classificação ou regressão. Isto é feito através da aplicação de operações de multiplicação de matriz entre os vetores de entrada e uma matriz de pesos (Equação (3.14)), seguidas por uma função de ativação não linear, como a função ReLU.

$$Z = X \cdot W + b \tag{3.14}$$

onde X é o vetor de entrada

W é a matriz de pesos

b valor de viés (bias)

Z é o vetor de saída da *Fully Connected Layer*

Depois do cálculo do vetor de saída, uma função de ativação não linear é aplicada elemento por elemento ao vetor resultante para introduzir não linearidades na rede.

Estas camadas aprendem a representar características mais abstratas e complexas dos dados, utilizando a informação extraída das *convolutional* e *pooling layers*.

No final, o vetor resultante é alimentado numa camada de saída para realizar a tarefa final, como classificação e regressão.

3.1.5 XGBoost

O XGBoost, que significa "*Extreme Gradient Boosting*", é uma técnica de *Machine Learning* baseada em *Decision Trees*, que ganhou popularidade significativa em competições de ciência de dados devido à sua eficácia e desempenho impressionante numa variedade de problemas. O XGBoost é uma implementação do algoritmo de *Gradient Boosting*, uma técnica de *Supervised Machine Learning* utilizado em problemas de regressão e classificação que tem as suas raízes no artigo "Greedy Function Approximation: A Gradient Boosting Machine" de Jerome H. Friedman [47].

Assim como *Random Forest* o *Gradient Boosting* é um método de *ensamble learning*, onde vários modelos mais fracos são combinados para formar um modelo mais robusto e preciso. A ideia principal por trás do *Gradient Boosting* é ajustar iterativamente modelos de *Decision Trees* simples para corrigir os erros cometidos pelos modelos anteriores [48]. No XGBoost, as *Decision Trees* são construidas sequencialmente, onde cada nova árvore tenta corrigir os erros residuais das árvores anteriores. Durante o treino, este algoritmo otimiza uma função de perda específica para minimizar o erro do modelo. Este processo é realizado através de um algoritmo de otimização que ajusta os pesos dos modelos individuais de acordo com a direção do gradiente da função de perda. Uma das características distintivas do XGBoost é o foco na regularização do modelo para evitar o *overfitting*. Ele utiliza termos de regularização na função de perda para penalizar modelos mais complexos, evitando assim que as árvores se tornem muito profundas e superajustadas aos dados de treino do modelo [49].

Assim como os demais algoritmos, o XGBoost utiliza dados de treino, por vezes com muitos *features* x_i, para prever um *target* y_i. Assim, um modelo de *Supervised Machine Learning* refere-se à estrutura matemática que prevê um valor, y_i, através de um *input*, x_i. Para isto, é preciso encontrar os melhores parâmetros, θ, que melhor se adequam aos *features* e às variáveis alvo, para isto, e como já foi referido anteriormente, este algoritmo usa uma função de perda juntamente com um termo de regularização para determinar uma função objetivo (Equação (3.15)).

$$obj(\theta) = L(\theta) + \Omega(\theta) \tag{3.15}$$

onde L é a função de perda

Ω é o termo de regularização

A função de perda L, pode tomar diversas formas, como apresentado na Secção 2.3.2, como por exemplo a MSE (Equação (3.16)).

$$L(\theta) = \sum_i (y_i - \widehat{y}_i)^2 \tag{3.16}$$

onde y_i é o valor real

$\widehat{y}_i$ é o valor previsto

Já o termo de regularização controla a complexidade do modelo, ajudando a evitar o *overfitting*. Através da Figura 3.14 é possível visualizar que o modelo a vermelho é o melhor visto que a complexidade transmitida pelo termo de regularização se adequa ao problema. Este algoritmo define dois parâmetros diferentes para calcular a complexidade do modelo (Equação (3.17)): o número de folhas e a pontuação das folhas.

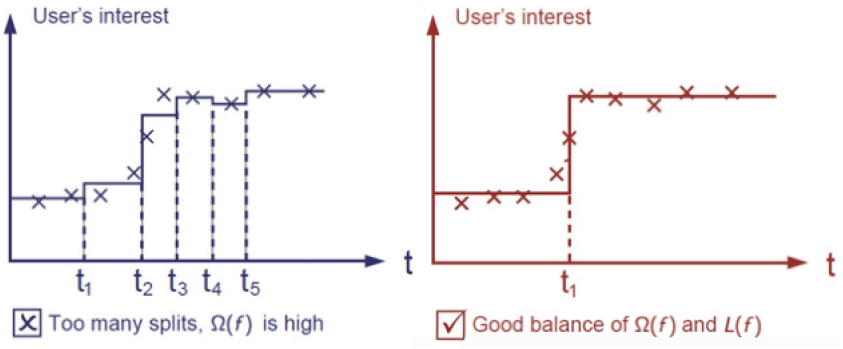


Figura 3.14: Termo de regularização [50]

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{3.17}$$

onde T é o número de folhas

λ *e* γ são hiperparâmetros

w_j é a pontuação de cada folha

O processo de treino do XGBoost pode ser entendido de maneira mais aprofundada ao considerar os componentes matemáticos e os passos iterativos envolvidos. Durante cada iteração t, uma nova árvore $f_t(x)$ é adicionada ao modelo para minimizar a função de perda (Figura 3.15 e Equação (3.18)).

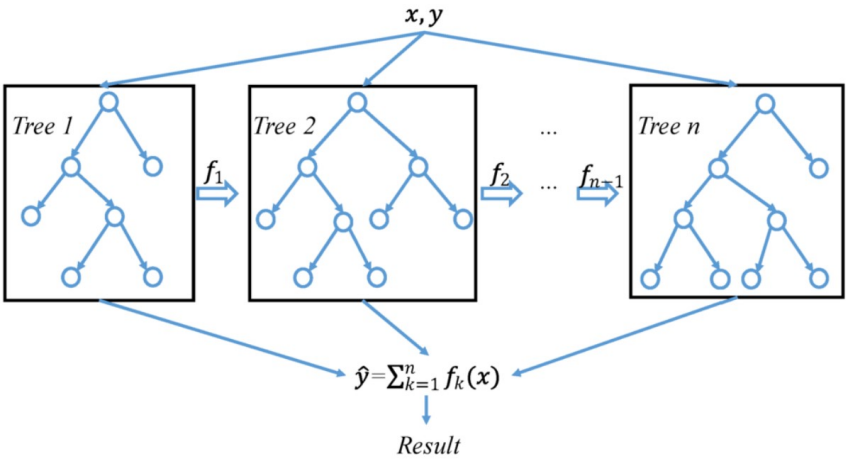


Figura 3.15: Arquitetura XGBoost [51]

$$\widehat{y_i}^{(t)} = \widehat{y_i}^{(t-1)} + f_t(x_i) \tag{3.18}$$

onde $\widehat{y_i}^{(t)}$ é a previsão do modelo

$\widehat{y_i}^{(t-1)}$ é a previsão do modelo no instante anterior

$f_t(x_i)$ é o valor previsto pela nova árvore

Assim é possível atualizar a Equação (3.15) para se igualar à Equação (3.19). De seguida, para resolver o problema de otimização, este algoritmo usa uma aproximação de segunda ordem, ou seja, expande a função de perda em torno das suas previsões atuais através de uma série de Taylor de segunda ordem (Equação (3.20)).

$$obj_i = \sum_i L(y_i, \widehat{y_i}^{(t-1)} + f_t(x_i)) + \Omega(f_t) \tag{3.19}$$

$$L(y_i, \widehat{y_i}^{(t-1)} + f_t(x_i)) \approx L(y_i, \widehat{y_i}^{(t-1)}) + g_i f_i(x_i) + \frac{1}{2} h_i f_t(x_i)^2 \tag{3.20}$$

onde g_i e h_i são o gradiente e o hessiano da função perda com respeito a $\widehat{y_i}^{(t-1)}$, respetivamente

Assim a função objetivo a ser minimizada é apresentada através da Equação (3.21). A regularização é então aplicada para garantir que o modelo não se torne excessivamente complexo controlando o crescimento e a pontuação de cada folha. O

ajuste dos hiperparâmetros λ e γ é crucial para equilibrar o viés e a variância do modelo, garantindo um desempenho robusto tanto em dados de treino como em dados de teste. Desta forma, o XGBoost combina a robustez do *Gradient Boosting* com técnicas avançadas de regularização, tornando-se uma boa ferramenta de *Machine Learning* [50].

$$obj_i \approx \sum_i [g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)^2] + \Omega(f_t) \tag{3.21}$$

3.2 Métricas de Desempenho

As métricas de desempenho desempenham um papel fundamental na avaliação de algoritmos de previsão, fornecendo uma medida objetiva de qualidade das previsões em relação aos valores reais. Para a avaliação do desempenho dos diferentes algoritmos foi decidido utilizar um conjunto de métricas especializadas para a tarefa de regressão, visto que é essencial selecionar métricas que capturem tanto a magnitude quanto a direção dos erros da previsão. Métricas comuns, como *Root-Mean-Square Error* (RMSE) e *Mean Absolute Error* (MAE), são frequentemente utilizadas devido à sua fácil interpretação e capacidade de fornecer uma medida clara da precisão das previsões.

Assim foi decidido usar um conjunto de cinco métricas diferentes para avaliar os diferentes algoritmos, sendo estas a *Mean Absolute Error*, *Mean Squared Error*, *Root-Mean-Square Error*, *Mean Absolute Percentage Error* e *R-Squared*.

Começando pelo *Mean Absolute Error* (MAE), esta é uma medida simples da média das diferenças absolutas entre as previsões e os valores reais. Esta métrica fornece uma indicação direta da magnitude média dos erros da previsão, independentemente da sua direção. Em simples termos, o MAE é calculado pela média das diferenças absolutas entre as previsões e os valores reais (Equação (2.8)), onde uma diferença absoluta menor indica uma melhor qualidade da previsão. Seguidamente, também será usado o *Mean Squared Error* (MSE), outra métrica bastante comum para avaliar a qualidade das previsões em relação aos valores reais. Esta métrica calcula a média dos quadrados das diferenças entre as previsões e os valores reais (Equação (2.7)). O MSE é particularmente sensível a grandes erros de previsão, uma vez que os erros são elevados ao quadrado antes de serem média. Isto pode ser bastante útil para destacar a presença de valores *outliers* nos dados ou para penalizar erros grandes de forma mais severa.

Para além destas duas métricas apresentadas na Secção 2.3.2, foi usado também a *Root-Mean-Square Error* (RMSE), que é uma variante do MSE. O RMSE calcula a raiz quadrada do MSE, fornecendo uma medida da magnitude média dos erros de previsão numa escala semelhante aos valores reais. O RMSE é amplamente

usado devido à usa interpretabilidade e capacidade de fornecer uma medida clara da previsão das previsões (Equação (3.22)). Como o RMSE é expresso na mesma unidade que os valores reais, é mais fácil de interpretar e comparar com os valores reais.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - x_i)^2} \tag{3.22}$$

onde n é o número total de valores

y_i é o valor previsto

x_i é o valor real

Outra métrica importante é o *Mean Absolute Percentage Error* (MAPE), que é uma medida útil para entender o erro percentual médio das previsões em relação aos valores reais. O MAPE calcula a média das diferenças percentuais absolutas entre as previsões e os valores reais (Equação (3.23)). O MAPE é especialmente útil quando se precisa de entender a precisão relativa das previsões em relação aos valores reais, independentemente da escala dos dados.

$$MAPE = 100\frac{1}{n}\sum_{i=1}^{n}|\frac{x_i - y_i}{x_i}| \tag{3.23}$$

onde n é o número total de valores

y_i é o valor previsto

x_i é o valor real

Por fim, também foi utilizado o coeficiente de determinação ou R^2, que é uma métrica estatística importante que indica a proporção da variabilidade nos dados que é explicada pelo modelo (Equação (3.24)). Valores mais próximos de 1 indicam um bom ajuste do modelo aos dados, enquanto valores mais próximos de 0 indicam um mau ajuste do modelo. O R^2 é uma métrica útil para entender o poder explicativo do modelo e será especialmente útil para uma rápida comparação entre diferentes modelos obtidos através de diferentes algoritmos.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(x_i - y_i)^2}{\sum_{i=1}^{n}(x_i - \overline{x})^2} \tag{3.24}$$

onde n é o número total de valores

y_i é o valor previsto

x_i é o valor real

$\widehat{x}$ é a média dos valores reais

Capítulo 4

Desenvolvimento

Neste capítulo irá ser discutido todo o desenvolvimento das técnicas e processos utilizados. Começando pela descrição do *dataset* utilizado, seguido da apresentação e descrição de todos os *features* integrados no *dataset*. Por fim serão detalhados os métodos utilizados para efetuar a previsão através dos algoritmos e a apresentação dos modelos desenvolvidos.

4.1 *Dataset*

O primeiro passo, para a elaboração de um modelo de previsão do mercado das ações, é adquirir um *dataset* que forneça todas as informações relativas ao preço da ação, assim como o volume das mesmas. Para isto, foram utilizados *datasets* fornecidos pelo Yahoo Finance, uma plataforma confiável e amplamente utilizada para acesso a dados financeiros históricos e em tempo real. Esta plataforma fornece uma gama abrangente de informações sobre os preços e volumes das negociações, fornecendo uma base sólida para previsões no mercado [52].

O *dataset* utilizado foi o da empresa *Apple Inc.* (Figura 4.1) que fornece um total de seis colunas: a "Date", esta coluna representa as datas em que os dados foram registados, é importante observar que, nos fins de semana e feriados não há negociações no mercado financeiro, logo não há registos de dados para esses dias; a coluna "Open", que se refere ao preço de abertura das ações no início do dia de negociações, sendo este o preço pelo qual a primeira transação do dia é realizada; as colunas "High"e "Low", como indicam os nomes, referem-se ao preço mais alto

e baixo, respetivamente, que as ações atingiram durante o dia de negociações; a coluna "Close", apresenta o preço das ações no final do dia de negociações; a coluna "Adj Close", indica o valor de "Close" da ação ajustado para levar em consideração quaisquer eventos corporativos, como dividendos, divisões de ações, etc., que possam afetar o preço das ações e proporcionar uma visão mais precisa do desempenho das ações ao longo do tempo [53], um outro ponto a ressaltar é o facto de todos estes valores de preços estarem na moeda dólar dos Estados Unidos (USD); e por fim a coluna "Volume" que representa o número total de ações que foram negociadas durante o dia de negociação, sendo uma medida da atividade de negociação e pode ajudar a indicar o interesse dos investidores nas ações.

Date	Open	High	Low	Close*	Adj Close**	Volume
Mar 20, 2024	175.73	176.38	175.09	176.02	176.02	13,775,229
Mar 19, 2024	174.34	176.61	173.03	176.08	176.08	55,130,000
Mar 18, 2024	175.57	177.71	173.52	173.72	173.72	75,604,200
Mar 15, 2024	171.17	172.62	170.29	172.62	172.62	121,664,700
Mar 14, 2024	172.91	174.31	172.05	173.00	173.00	72,913,500
Mar 13, 2024	172.77	173.19	170.76	171.13	171.13	52,488,700

Figura 4.1: *Dataset* do preço das ações da *Apple Inc.*

Para além do *dataset* com o preço das ações, também foram utilizados diversos *datasets* que incluem o preço de diversas *commodities*, taxas de juros, entre outros, mencionados na Secção 2.4, como preço do *3-Month Treasury Bill* apresentado na Figura 4.2. Para obter estes *datasets* foi utilizado o *Federal Reserve Economic Data* (FRED), uma base de dados económicos suportada pelo *Federal Reserve Bank of St. Louis*. Esta base de dados fornece uma ampla variedade de dados económicos, incluindo séries temporais sobre emprego, inflação, produção industrial, preços, entre outros indicadores económicos de todo o mundo [54].

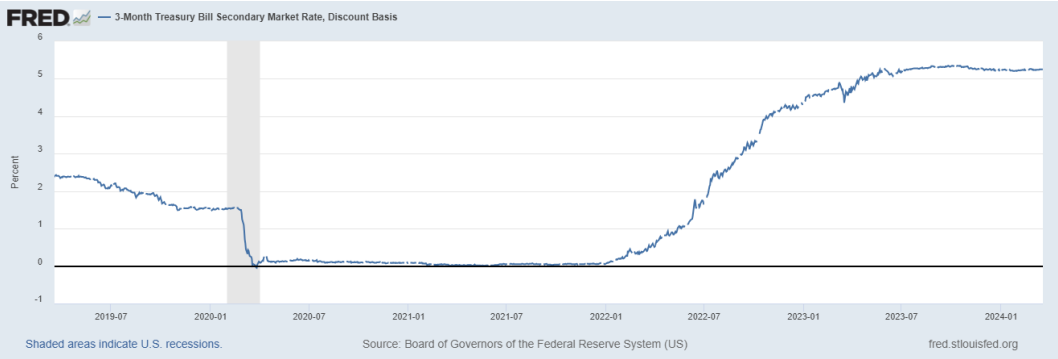


Figura 4.2: *Dataset 3-Month Treasury Bill* [55]

4.2 *Features*

Para a construção de um modelo de previsão do mercado de ações, a seleção das *features* (características) adequadas é essencial para garantir a precisão e a relevância das previsões. As *features* são variáveis independentes que fornecem informações relevantes ao modelo, ajudando-o a identificar padrões e realizar previsões precisas. As *features* têm um impacto direto na precisão e no desempenho de modelos de *Machine Learning*, onde a escolha das *features* é um processo iterativo de adição e remoção de características com o objetivo de encontrar o modelo que apresenta melhor desempenho [56].

A seleção das *features* é um dos processos mais importantes na aplicação de modelos de previsão. Para isto, foram selecionados 49 *features* sendo estes uma combinação de indicadores técnicos, dados económicos, valores de *commodities* e índices mundiais com base num estudo intitulado de "*CNNpred: CNN-based stock market prediction using a diverse set of variables*"de Ehsan Hoseinzade e Saman Haratizadeh [57] (Tabela 4.1).

Para além disso, foram utilizados dois diferentes métodos de seleção de *features* com o objetivo de utilizar no modelo apenas os *features* mais importantes. Sendo assim, foram utilizados a análise de correlação e o método SelectKBest, baseado em pontuações de correlação e testes estatísticos F de regressão, respetivamente.

Começando pela análise de correlação, cujos resultados podem ser observados na Tabela 4.2, esta indica a correlação de todos os *features* com a variável a prever, neste caso, o preço da ação no dia de amanhã (Adj Close). A análise de correlação é uma abordagem fundamental para entender a relação entre todas as *features* e a variável alvo. É de reparar que nesta análise os melhores 20 *features* apresentam valores de correlação bastante superiores aos demais, visto que são variáveis que preço, médias móveis ou índices onde é natural acompanharem a flutuação do preço da ação de perto.

Tabela 4.1: *Features* testados

#	**Feature**	**Descrição**	**Tipo**	**Fonte**
1	Open	Preço de abertura	Preço	Yahoo Finance
2	High	Preço mais alto	Preço	Yahoo Finance
3	Low	Preço mais baixo	Preço	Yahoo Finance
4	Adj Close	Preço ajustado de fechamento	Preço	Yahoo Finance
5	Volume	Volume de ações negociadas	Volume	Yahoo Finance
6	MOM2	Momento de 2 dias	Indicador Técnico	Calculado
7	MOM3	Momento de 3 dias	Indicador Técnico	Calculado
8	MOM4	Momento de 4 dias	Indicador Técnico	Calculado
9	MOM5	Momento de 5 dias	Indicador Técnico	Calculado
10	MACD	*Moving Average Convergence/Divergence*	Indicador Técnico	TA-Lib
11	RSI	*Relative Strength Index*	Indicador Técnico	TA-Lib
12	ROC5	Taxa de Mudança de 5 dias	Indicador Técnico	TA-Lib
13	ROC10	Taxa de Mudança de 10 dias	Indicador Técnico	TA-Lib
14	ROC15	Taxa de Mudança de 15 dias	Indicador Técnico	TA-Lib
15	ROC20	Taxa de Mudança de 20 dias	Indicador Técnico	TA-Lib
16	SMA5	Média Móvel Simples de 5 dias	Indicador Técnico	TA-Lib
17	SMA25	Média Móvel Simples de 25 dias	Indicador Técnico	TA-Lib
18	SMA50	Média Móvel Simples de 50 dias	Indicador Técnico	TA-Lib
19	SMA100	Média Móvel Simples de 100 dias	Indicador Técnico	TA-Lib
20	SMA200	Média Móvel Simples de 200 dias	Indicador Técnico	TA-Lib
21	EMA10	Média Móvel Exponencial de 10 dias	Indicador Técnico	TA-Lib
22	EMA12	Média Móvel Exponencial de 12 dias	Indicador Técnico	TA-Lib
23	EMA20	Média Móvel Exponencial de 20 dias	Indicador Técnico	TA-Lib
24	EMA26	Média Móvel Exponencial de 26 dias	Indicador Técnico	TA-Lib
25	EMA50	Média Móvel Exponencial de 50 dias	Indicador Técnico	TA-Lib
26	EMA100	Média Móvel Exponencial de 100 dias	Indicador Técnico	TA-Lib
27	EMA200	Média Móvel Exponencial de 200 dias	Indicador Técnico	TA-Lib
28	DTB4WK	Taxa de Títulos do Tesouro de 4 semanas	Taxa de Juro	FRED
29	DTB3	Taxa de Títulos do Tesouro de 3 meses	Taxa de Juro	FRED
30	DTB6	Taxa de Títulos do Tesouro de 6 meses	Taxa de Juro	FRED
31	DGS5	Taxa de Maturidade Constante do Tesouro de 5 anos	Taxa de Juro	FRED
32	DGS10	Taxa de Maturidade Constante do Tesouro de 10 anos	Taxa de Juro	FRED
33	DAAA	Taxa de Títulos Corporativos AAA	Taxa de Juro	FRED
34	DBAA	Taxa de Títulos Corporativos BAA	Taxa de Juro	FRED
35	TE1	Diferença entre DGS10 e DTB4WK	Spread de Taxa de Juro	Calculado
36	TE2	Diferença entre DGS10 e DTB3	Spread de Taxa de Juro	Calculado
37	TE3	Diferença entre DGS10 e DTB6	Spread de Taxa de Juro	Calculado
38	TE5	Diferença entre DTB3 e DTB4WK	Spread de Taxa de Juro	Calculado
39	TE6	Diferença entre DTB6 e DTB4WK	Spread de Taxa de Juro	Calculado
40	DE1	Diferença entre DBAA e DAAA	Spread de Crédito	Calculado
41	DE2	Diferença entre DBAA e DGS10	Indicador Técnico	Calculado
42	DE4	Diferença entre DBAA e DTB6	Indicador Técnico	Calculado
43	DE5	Diferença entre DBAA e DTB3	Indicador Técnico	Calculado
44	DE6	Diferença entre DBAA e DTB4WK	Indicador Técnico	Calculado
45	DCOILWTICO	Preço do Petróleo WTI	*Commodity*	FRED
46	IXIC	Índice NASDAQ Composite	Índice	Yahoo Finance
47	GSPC	Índice S&P 500	Índice	Yahoo Finance
48	DJI	Índice Dow Jones Industrial	Índice	Yahoo Finance
49	NYA	Índice NYSE Composite	Índice	Yahoo Finance

Para além da análise de correlação também foi usado o método `SelectKBest` para selecionar os melhores *features*. Este é um dos métodos de seleção de *features* mais utilizados, sendo baseado em filtros de *Machine Learning*. Utiliza testes estatísticos para selecionar as *features* que têm a maior relação com a variável de saída, onde o procedimento envolve definir inicialmente o teste estatístico adequado ao tipo de dados e ao problema em questão. Para dados categóricos, o teste `Chi-squared` é uma escolha comum, enquanto para dados contínuos, o teste `ANOVA F-Value` pode ser utilizado. Por outro lado, para casos como regressões o método `SelectKBest` disponibiliza a `f_regression`, sendo esta opção a utilizada para efetuar a seleção das melhores *features*. Em seguida, o teste é aplicado a cada *feature* para calcular uma pontuação de importância. As *features* com as maiores pontuações são selecionadas, e o conjunto de dados é transformado para incluir apenas estas características [58].

Tabela 4.2: Correlação das *Features*

Nº	***Feature***	**Pontuação**	**Nº**	***Feature***	**Correlação**
1	Adj Close	0.999757	26	DCOILWTICO	0.163709
2	Low	0.999527	27	TE6	0.140110
3	High	0.999503	28	MOM5	0.089892
4	Open	0.999411	29	MOM4	0.080078
5	SMA5	0.999335	30	MOM3	0.067845
6	EMA10	0.999167	31	MOM2	0.052424
7	EMA12	0.999064	32	RSI	0.007301
8	EMA20	0.998661	33	ROC5	-0.014739
9	EMA26	0.998375	34	ROC10	-0.017959
10	SMA25	0.997924	35	ROC15	-0.021291
11	EMA50	0.997375	36	ROC20	-0.023569
12	SMA50	0.996326	37	DGS5	-0.116094
13	EMA100	0.995705	38	DE1	-0.169746
14	SMA100	0.994129	39	DE2	-0.325379
15	EMA200	0.992936	40	DGS10	-0.327576
16	SMA200	0.990444	41	Volume	-0.472102
17	IXIC	0.963824	42	DBAA	-0.495441
18	GSPC	0.955474	43	DAAA	-0.502339
19	DJI	0.931631	44	DE6	-0.524144
20	NYA	0.876839	45	TE1	-0.533726
21	MACD	0.266051	46	DE5	-0.538855
22	TE5	0.228270	47	DE4	-0.543380
23	DTB3	0.185540	48	TE2	-0.556787
24	DTB6	0.184989	49	TE3	-0.563363
25	DTB4WK	0.166905			

Ao aplicar este método às 49 *features*, foram então obtidas pontuações para cada uma como evidenciado na Tabela 4.3.

Tabela 4.3: SelectKBest

Nº	***Feature***	**Pontuação**	**Nº**	***Feature***	**Pontuação**
1	Adj Close	11500000	25	TE1	2230
2	Low	5900000	26	DE6	2120
3	High	5630000	27	DAAA	1890
4	Open	4740000	28	DBAA	1820
5	SMA5	4200000	29	Volume	1600
6	EMA10	3350000	30	DGS10	672
7	EMA12	2980000	31	DE2	662
8	EMA20	2080000	32	MACD	6.72
9	EMA26	1720000	33	TE5	307
10	SMA25	1340000	34	DTB3	199
11	EMA50	1060000	35	DTB6	198
12	SMA50	756000	36	DE1	166
13	EMA100	647000	37	DTB4WK	160
14	SMA100	472000	38	DCOILWTICO	154
15	EMA200	392000	39	TE6	112
16	SMA200	288000	40	DGS5	76.4
17	IXIC	73100	41	MOM5	45.6
18	GSPC	58600	42	MOM4	36.1
19	DJI	36800	43	MOM3	25.9
20	NYA	18600	44	MOM2	15.4
21	TE3	2600	45	ROC20	3.11
22	TE2	2510	46	ROC15	2.54
23	DE4	2340	47	ROC10	1.80
24	DE5	2290	48	ROC5	1.22
			49	RSI	0.298

Ao analisar as Tabelas 4.2 e 4.3 consegue-se concluir que o desempenho das 20 melhores *features* não variam entre os dois métodos de seleção, para além disso as 20 melhores *features* obtiveram pontuações muito melhores que as demais especialmente na análise de correlação onde a 20ª melhor *feature* (NYA) obteve uma pontuação de 0.876839 enquanto a 21ª (MACD) obteve uma pontuação de 0.266051, o que é indicativo de uma grande diferença de importância da *feature* para o modelo de previsão.

Para uma melhor análise dos melhores *features*, foram combinadas as classificações de cada *feature* (Figura 4.3), onde é possível identificar mais uma vez uma clara diferença de desempenho entre as primeiras 20 *features* e as demais.

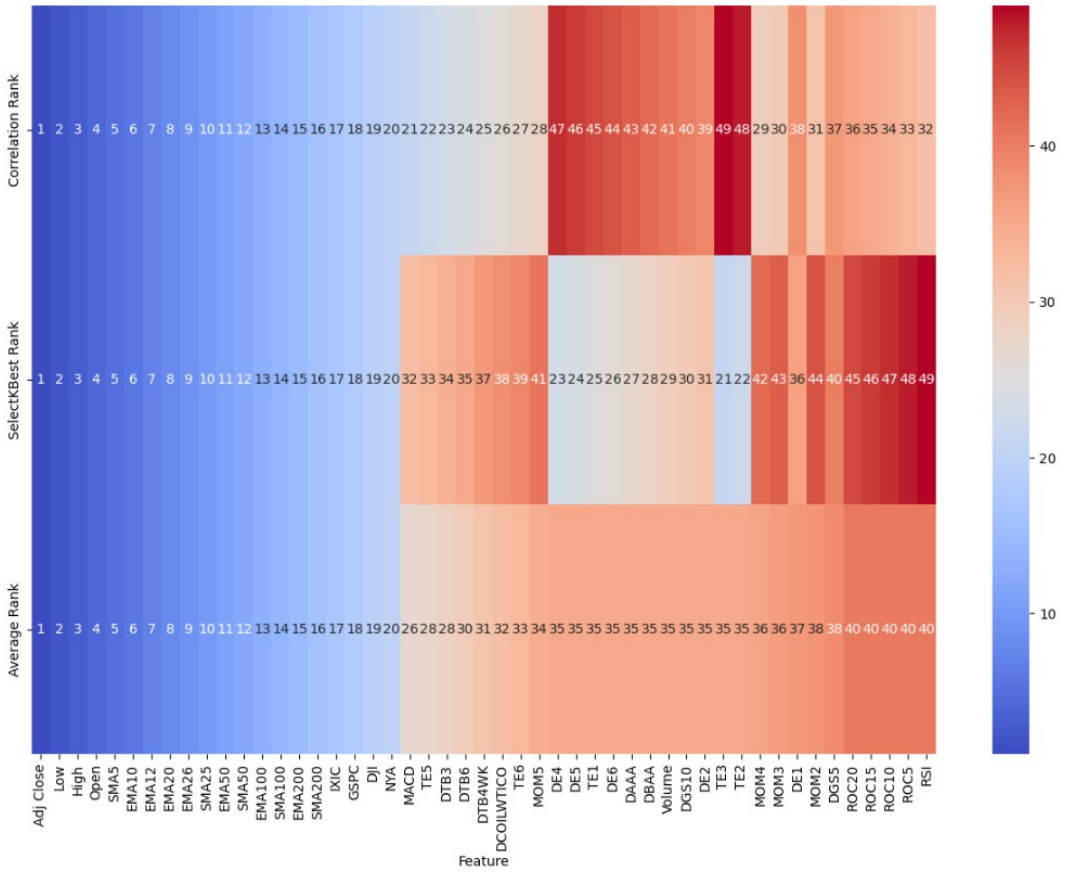


Figura 4.3: Pontuações das diferentes *features*

Após esta análise, decidiu-se utilizar apenas as 20 melhores *features* no modelo de previsão, que incluem 4 variáveis provenientes do *dataset* escolhido (Adj Close, Low, High e Open), 12 indicadores técnicos, que incluem 5 SMA e 7 EMA de diferentes tamanhos, e por fim 4 índices, o NASDAQ Composite (IXIC), S&P 500 (GSPC), Dow Jones Industrial Average (DJI) e NYSE Composite (NYA).

4.2.1 Indicadores Técnicos

Para além dos *features* obtidos através do *dataset* fornecido pelo Yahoo Finance também serão utilizados diversos indicadores técnicos para melhorar a precisão do modelo de previsão, tais como:

- *Simple Moving Average* (SMA) - É um indicador técnico que calcula a média aritmética dos preços de uma ação ao longo de um período específico, criada através da adição de preços recentes, e, em seguida dividindo esse valor pelo número de períodos de tempo na média (Equação (4.1)). A SMA é utilizada para suavizar os dados dos preços, criando uma linha de tendência que pode ajudar a identificar a direção geral do mercado (Figura 4.4) [59].

$$SMA = \frac{x_1 + x_2 + ... + x_t}{n} \tag{4.1}$$

onde x_t é o preço no período t

n é o número total de períodos

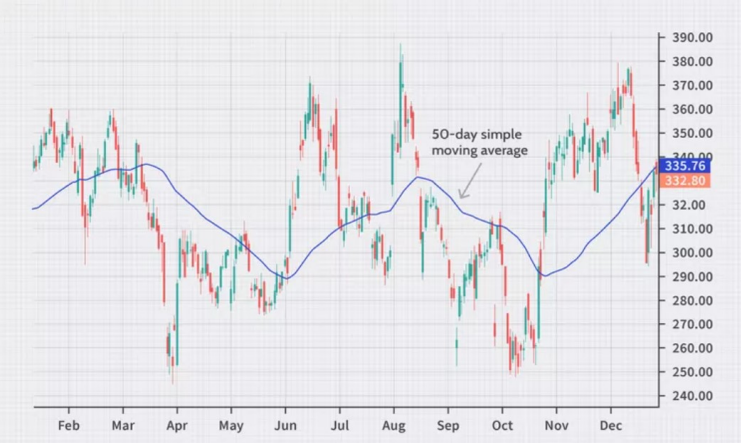


Figura 4.4: *Simple Moving Average* [59]

Para os modelos desenvolvidos foi decidido utilizar cinco SMA diferentes, em períodos de 5, 25, 50, 100 e 200 dias.

- *Exponential Moving Average* (EMA) - A EMA é outro indicador técnico que dá mais peso aos preços mais recentes tornando-a mais sensível às novas informações do que a SMA (Figura 4.5). Isto significa que a EMA responde mais rapidamente às mudanças no preço da ação. A fórmula para calcular a EMA (Equação (4.2)) envolve um coeficiente de suavização, k, que dá mais peso aos preços mais recentes. É frequentemente utilizada em períodos de 12 e 26 dias para gerar sinais de compra e venda [60].

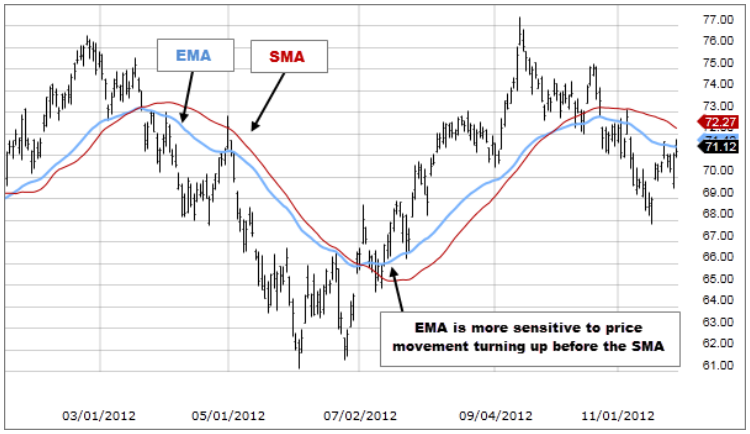


Figura 4.5: SMA e EMA [61]

$$EMA_t = \frac{x_t - EMA_{t-1}}{k} + EMA_{t-1} \tag{4.2}$$

onde EMA_t é o valor da EMA no período t

EMA_{t-1} é o valor da EMA no período anterior

x_t é o preço no período t

k é o coeficiente de suavização, que pode ser calculado como $k = \frac{2}{n+1}$, onde n é o número total de períodos

Assim, para estes modelos foram utilizadas EMA de períodos 12, 26, e também de 50 e 200, com o objetivo de fornecerem aos modelos valores melhores para curto prazo, mas também para longo prazo, assim como os utilizados na SMA.

4.2.2 Índices

- NASDAQ Composite (IXIC) - Este índice é um dos principais indicadores do mercado de ações dos Estados Unidos, representando todas as ações listadas na bolsa NASDAQ. É amplamente usado como um barômetro do desempenho do setor de tecnologia e inclui mais de 2500 empresas (Figura 4.6). O NASDAQ Composite é conhecido pela sua alta volatilidade e por ser sensível às mudanças no setor tecnológico, o que o torna um componente valioso para análises de previsões, especialmente a grandes empresas tecnológicas [62].

Technology	55.32%
Consumer Discretionary	18.80%
Healthcare	8.08%
Industrials	4.66%
Financials	3.47%
Telecommunications	3.22%
Consumer Staples	2.96%
Energy	1.05%
Real Estate	1.05%
Utilities	0.94%
Basic Materials	0.45%

Figura 4.6: *Breakdown* setorial para o índice NASDAQ Composite [62]

Este índice reflete fortemente as tendências das empresas tecnológicas devido à sua composição predominante de ações desse setor. As dez maiores empresas do NASDAQ Composite representam uma significativa porção do índice, com um impacto considerável no seu desempenho geral. A Tabela 4.4 lista as dez empresas com maior peso no índice, destacando a prevalência de gigantes da tecnologia como Microsoft, Apple e NVIDIA. Essa concentração de grandes empresas tecnológicas torna o NASDAQ Composite um excelente indicador das tendências e previsões para o mercado de ações de tecnologia. O comportamento dessas empresas pode fornecer informações para os modelos de previsão.

Tabela 4.4: Os 10 títulos com maior peso no NASDAQ Composite [63]

#	**Nome (Símbolo)**	%
1	Microsoft Corporation (MSFT)	8.44%
2	Apple Inc. (AAPL)	8.23%
3	NVIDIA Corporation (NVDA)	7.90%
4	Amazon.com Inc. (AMZN)	5.09%
5	Meta Platforms Inc. (META)	4.59%
6	Broadcom Inc. (AVGO)	4.35%
7	Alphabet Inc. Class A (GOOGL)	2.80%
8	Alphabet Inc. Class C (GOOG)	2.72%
9	Costco Wholesale Corporation (COST)	2.57%
10	Tesla Inc. (TSLA)	2.34%

- S&P 500 (GSPC) - O S&P 500 é um dos índices mais seguidos no mercado de ações dos Estados Unidos, é composto pelas 500 maiores empresas na bolsa de valores NYSE ou NASDAQ. Este índice, assim como o NASDAQ Composite, é amplamente utilizado como um indicador do desempenho geral do mercado de ações dos Estados Unidos. Este índice inclui empresas de diversos setores, oferecendo uma visão bem abrangente do mercado. A Tabela 4.5 lista as dez empresas com maior peso no S&P 500, destacando mais uma vez, uma grande influência das gigantes empresas de tecnologia, o que justifica o bom desempenho deste índice nas análises de *features*.

Tabela 4.5: Os 10 títulos com maior peso no S&P 500 [64]

#	**Nome (Símbolo)**	%
1	Microsoft Corporation (MSFT)	6.94%
2	NVIDIA Corporation (NVDA)	6.46%
3	Apple Inc. (AAPL)	6.34%
4	Amazon.com Inc. (AMZN)	3.68%
5	Meta Platforms Inc. Class A (META)	2.36%
6	Alphabet Inc. Class A (GOOGL)	2.30%
7	Alphabet Inc. Class C (GOOG)	1.94%
8	Berkshire Hathaway Class B (BRK.B)	1.67%
9	Eli Lilly & Co. (LLY)	1.49%
10	Broadcom Inc. (AVGO)	1.32%

- *Dow Jones Industrial Average* (DJI) - O *Dow Jones Industrial Average* é um dos índices mais antigos e reconhecidos no mundo, composto por 30 grandes empresas dos Estados Unidos listadas na NASDAQ e na NYSE. Estas empresas são líderes nos seus setores selecionados para refletir a economia americana, apresentando uma maior variedade de setores nas 10 empresas com maior peso (Tabela 4.6) quando comparado com o NASDAQ Composite e S&P 500 [65]. Para além disso, visto que este índice é um refletir da economia americana como um todo e não tanto sobre o sector tecnológico, é de esperar a diminuição de desempenho nas análises de *features* quando comparado diretamente ao NASDAQ Composite e S&P 500 na previsão do preço das ações da Apple Inc..

Tabela 4.6: Os 10 títulos com maior peso no DJI [66]

#	**Nome (Símbolo)**	%
1	Unitedhealth Group Inc. (UNH)	8.56%
2	Goldman Sachs Group Inc. (GS)	7.71%
3	Microsoft Corp (MSFT)	7.05%
4	Home Depot Inc. (HD)	5.56%
5	Caterpillar Inc. (CAT)	5.55%
6	Amgen Inc. (AMGN)	5.20%
7	Visa Inc. Class A (V)	4.61%
8	Mcdonald S Co (MCD)	4.45%
9	American Express Co. (AXP)	4.02%
10	Salesforce Inc. (CRM)	3.98%

- NYSE Composite (NYA) - Por fim, o NYSE Composite é um índice abrangente que inclui todas as ações listadas na bolsa de valores de Nova York (*New York Stock Exchange*), oferecendo uma visão mais ampla do mercado quando comparado com os demais índices. Por englobar todas as ações listadas na NYSE, este índice proporciona uma perspetiva geral do mercado, representando empresas de uma ampla gama de setores e tamanhos. Embora possa não ter o mesmo foco no setor tecnológico como o NASDAQ Composite e o S&P 500, o NYSE Composite ainda é útil para a previsão de empresas americanas, sendo isto refletido na pontuação na análise de *features*.

4.3 *Framework*

Nesta secção, será detalhado o *framework* utilizado para a implementação dos modelos a usar na previsão do preço das ações. Esta escolha desempenha um papel importante no desenvolvimento eficaz e na análise dos modelos de *Machine Learning*, permitindo uma implementação eficiente, experimentação rápida e avaliação precisa dos resultados.

Para este trabalho foram utilizadas um conjunto vasto de ferramentas que auxiliaram todo este processo, desde a recolha de dados até a implementação e avaliação dos modelos. O principal ambiente de desenvolvimento utilizado foi o Jupyter Notebook, uma aplicação *web*, *open source* que permite a criação e partilha de documentos que incluem código em tempo real, equações e outros recursos multimédia [67]. O uso desta ferramenta facilitou a interação com o código, a visualização dos resultados intermédios e a documentação das etapas de todo o processo.

Além de mencionar o ambiente de desenvolvimento, também será necessário abordar a linguagem de programação associada, o Python. Python é uma linguagem de programação *high level*, interpretada e orientada a objetos, amplamente utilizada no desenvolvimento de *software* especialmente aplicada em projetos de *data science* e *Machine Learning*. A sua sintaxe simples, juntamente com uma grade variedade de bibliotecas, *frameworks* e material de estudo disponíveis, tornam Python uma escolha popular para projetos de análise de dados e modelação estatística.

Após definir o contexto do ambiente de desenvolvimento e a importância das ferramentas utilizadas, é essencial compreender os componentes específicos que foram incorporados na implementação dos modelos de previsão de preços das ações. Esta implementação foi viabilizada por meio de várias bibliotecas e pacotes em Python, onde cada um é utilizado no processo de análise e modelação. Abaixo estão algumas dessas bibliotecas e as suas funções:

- `yfinance`: Esta biblioteca foi utilizada para recolher dados financeiros históricos do preço da ação a prever e os diversos índices utilizados, diretamente do Yahoo Finance, essencial para obter dados precisos e atualizados [68];

- `pandas`: Biblioteca fundamental para a manipulação e análise de dados, onde oferece estruturas de dados rápidas e flexíveis, como *dataframes* que facilitam o processamento e a limpeza dos dados antes da previsão [69];
- `pandas_ta`: Pandas Technical Analysis é uma biblioteca focada em análise técnica que forneceu uma variedade de indicadores técnicos que podem ser aplicados aos dados financeiros diretamente, como os previamente mencionados SMA e EMA [70];
- `numpy`: Biblioteca utilizada para realizar operações numéricas em grande escala, útil para toda a manipulação de *arrays* e matrizes de grandes dimensões [71];
- `sklearn.preprocessing`: Módulo da biblioteca `scikit-learn` que fornece o método `MinMaxScaler` utilizado para realizar o pré-processamento dos dados, um passo a realizar antes da utilização dos dados como entradas para os modelos [72];
- `keras`: Biblioteca de alto nível para a contrução e treino de modelos de *Neural Networks*, que fornece uma interface intuitiva para criar, configurar e avaliar modelos de *Deep Learning*, facilitando o desenvolvimento de redes complexas como LSTM, GRU e CNN [73];
- `keras_tuner`: Biblioteca utilizada para a otimização de hiperparâmetros dos modelos de *Deep Learning* através da sua função `BayesianOptimization` [74];
- `matplotlib`: Biblioteca utilizada exclusivamente para a visualização de dados, nomeadamente os dados previstos e os reais, auxiliando assim o processo de ilustração de tendências e avaliação do desempenho dos modelos [75].

4.4 Pré-processamento de dados

O pré-processamento de dados é uma etapa essencial na construção deste tipo de modelos de previsão, garantindo que os dados estejam adequadamente preparados para os utilizar nos algoritmos de *Machine Learning* resultando em modelos mais precisos e confiáveis.

Aquisição e Limpeza de Dados

Os dados históricos da Apple Inc. e dos principais índices financeiros foram adquiridos usando a função `download` da biblioteca `yfinance` como evidenciado pela Listagem 4.1.

```
data = yf.download(tickers= 'AAPL', period='max')
IXIC = yf.download(tickers='^IXIC', period='max')
GSPC = yf.download(tickers='^GSPC', period='max')
DJI = yf.download(tickers='^DJI', period='max')
NYA = yf.download(tickers='^NYA', period='max')
```

Listagem 4.1: Aquisição de dados.

Estes dados foram coletados desde a abertura da empresa e desde a criação dos diversos índices, fornecendo um total de 6 colunas de dados como previamente referido pela Secção 4.1 e referenciado pela Figura 4.7.

Date	Open	High	Low	Close	Adj Close	Volume
1980-12-12	0.128348	0.128906	0.128348	0.128348	0.099058	469033600
1980-12-15	0.122210	0.122210	0.121652	0.121652	0.093890	175884800
1980-12-16	0.113281	0.113281	0.112723	0.112723	0.086998	105728000
1980-12-17	0.115513	0.116071	0.115513	0.115513	0.089152	86441600
1980-12-18	0.118862	0.119420	0.118862	0.118862	0.091737	73449600
...	...	...	...	...	...	...
2024-05-30	190.759995	192.179993	190.630005	191.289993	191.289993	49947900
2024-05-31	191.440002	192.570007	189.910004	192.250000	192.250000	75158300
2024-06-03	192.899994	194.990005	192.520004	194.029999	194.029999	50080500
2024-06-04	194.639999	195.320007	193.029999	194.350006	194.350006	47471400
2024-06-05	195.399994	196.899994	194.869995	195.869995	195.869995	54081300

Figura 4.7: *Dataset* do preço das ações da *Apple Inc.*

Utilização de Indicadores Técnicos

Além dos dados brutos dos preços das ações, também foram calculados os diversos indicadores técnicos, isto através das funções `sma` e `ema` para calcular a SMA de tamanhos 5, 25, 50, 100 e 200 e a EMA de tamanhos 10, 12, 20, 26, 50, 100 e 200 respectivamente (Listagem 4.2).

```
data['SMA5'] = ta.sma(data.Close, length=5)
data['SMA25'] = ta.sma(data.Close, length=25)
data['SMA50'] = ta.sma(data.Close, length=50)
data['SMA100'] = ta.sma(data.Close, length=100)
data['SMA200'] = ta.sma(data.Close, length=200)

data['EMA10'] = ta.ema(data.Close, length=10)
data['EMA12'] = ta.ema(data.Close, length=12)
data['EMA20'] = ta.ema(data.Close, length=20)
data['EMA26'] = ta.ema(data.Close, length=26)
data['EMA50'] = ta.ema(data.Close, length=50)
data['EMA100'] = ta.ema(data.Close, length=100)
data['EMA200'] = ta.ema(data.Close, length=200)
```

Listagem 4.2: Utilização de indicadores técnicos

Integração dos Índices Financeiros

Nesta etapa, os dados dos índices financeiros, como o NASDAQ Composite (`IXIC`), são manipulados e integrados aos dados das ações da Apple Inc. (Listagem 4.3). Primeiro os índices são redefinidos, para garantir que a coluna `Date` se torne uma coluna de dados. Em seguida, as colunas irrelevantes para este processo, como `Open`, `High`, `Low`, `Close` e `Volume`, são removidas do *dataset*.

Após esta pequena preparação, os dados do índice são unidos com os dados das ações com base na coluna `Date`, garantindo que todas as datas presentes no conjunto de dados das ações sejam preservadas e as datas dos dois *datasets* sejam alinhadas.

```
IXIC.reset_index(inplace=True)
IXIC = IXIC.drop(columns=['Open','High','Low','Close','
    Volume'])
IXIC.rename(columns= {'Adj Close' : 'IXIC'}, inplace = True
    )
data = pd.merge(data, IXIC, on='Date', how='left')
```

Listagem 4.3: Adição de Índices

Pré-processamento dos Dados

Após a integração dos diversos índices e indicadores técnicos num só *dataset*, as colunas irrelevantes restantes, como `Date` e `Volume`, são então removidas. Além disso, uma nova coluna nomeada de `Target` é criada, que representa o preço da ação no dia seguinte, sendo esta a variável alvo para ser prevista (Listagem 4.4). Com isto, consegue-se obter um *dataset* que apresenta um total de 21 colunas, sendo as

primeiras 20 colunas os 20 *features* selecionados, e a 21º coluna representa a variável alvo.

```
features = data.drop(columns=['Date','Volume'])
features['Target'] = features['Adj Close'].shift(-1)
features.dropna(inplace=True)
```

Listagem 4.4: Criação da variável `Target`

De seguida foi realizada a normalização dos valores. O pré-processamento dos dados inclui a normalização das *features* para assegurar que todas tenham a mesma escala, facilitando o treino do modelo. A normalização foi feita através da função `MinMaxScaler` da biblioteca `sklearn.preprocessing`, que escala os dados para um intervalo de 0 a 1 (Listagem 4.5). Este passo é bastante importante para melhorar o desempenho e a eficiência dos modelos de *Machine Learning*, ajudando a garantir que todas as *features* contribuam igualmente no processo de modelação, prevenindo que algumas variáveis com valores mais altos dominem outras com valores menores.

```
scaler = MinMaxScaler(feature_range=(0,1))
features = scaler.fit_transform(features)
```

Listagem 4.5: Normalização dos dados

Preparação dos Dados de Entrada e Saída

Nesta etapa, os dados são preparados para entrada nos modelos de *Machine Learning*. Os dados são organizados em sequências temporais de tamanho fixo, onde cada sequência contém todos os *features*. Para isso, é preciso inicialmente utilizar um conceito chamado de *input window* que envolve incorporar conjuntos de observações sequenciais; o número das mesmas estará guardado na variável `window` apresentada na Listagem 4.6, numa estrutura adequada para o treino do modelo.

Para estabelecer um valor para a variável `window`, foram feitos alguns testes com dois simples modelos, um que possui duas camadas GRU, e outro que possui duas camadas LSTM que concluiram que o melhor valor seria 100. Estes testes consistem em treinar cada modelo e prever o valor dos preços 10 vezes, armazenando o valor de cada métrica destas previsões e seguidamente calcular o valor médio para as mesmas, como mostrado nas Tabelas 4.7 e 4.8. Para além disso também foram tabelados os melhores e piores valores de cada variável e calculado o valor do desvio padrão, conforme consta na Equação (4.3) e tabelados no Anexo A, na Secção A.1.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N}} \tag{4.3}$$

onde σ é o desvio padrão,

x_i é um elemento do conjunto,

$\bar{x}$ é a média aritmética do conjunto, e

N é a quantidade de elementos no conjunto.

Tabela 4.7: *Input Window* Modelo GRU - Valor Médio

Input Window	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
25	6.26876	77.20209	8.65152	4.65%	0.97240
50	7.31752	101.58542	9.99415	5.32%	0.96358
75	7.01544	95.13397	9.61283	5.11%	0.96580
100	6.05663	70.33266	8.32452	4.52%	0.97458
125	6.15711	73.20857	8.46008	4.58%	0.97361
150	7.01215	93.64945	9.54157	5.13%	0.96606

Tabela 4.8: *Input Window* Modelo LSTM - Valor Médio

Input Window	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
25	5.32076	59.54124	7.58995	4.21%	0.97871
50	4.80348	47.84422	6.80072	3.88%	0.98285
75	5.10483	55.17844	7.25437	4.07%	0.98017
100	4.47343	40.44012	6.30300	3.67%	0.98542
125	5.04923	57.31991	7.17754	4.03%	0.97928
150	5.15170	55.66220	7.23008	4.10%	0.97982

A Listagem 4.6 mostra como este processo é realizado. A variável `window` representa o tamanho da janela, ou seja, quantos pontos de dados anteriores são incluídos em cada sequência temporal. O código itera sobre todos os *features* do *dataset*, criando janelas de entrada com o tamanho especificado. Estas janelas são então armazenadas na matriz `X`, enquanto as correspondentes saídas esperadas são armazenadas na matriz `y`.

```
for i in range(20):
    X.append([])
    for j in range(window,features.shape[0]):
        X[i].append(features[j-window:j,i])

X = np.moveaxis(X,[0],[2])

X = np.array(X)

yi = np.array(features[window:,-1])

y = np.reshape(yi,(len(yi),1))
```

Listagem 4.6: Preparação dos dados

O último passo neste processo é realizar a divisão dos dados em conjuntos de treino e de teste, com o objetivo de avaliar os modelos com dados que não foram vistos durante o processo de treino. Na Listagem 4.7, após a criação das matrizes `X` e `y`, é determinado um limite de divisão entre os conjuntos de treino e teste. Geralmente, uma proporção comum de ser utilizada são 80% dos dados reservados para o treino e 20% para o teste. De seguida são então criadas 4 matrizes, a `X_train`, `X_test`, `y_train` e `y_test` que correspondem às matrizes de entrada de treino e de teste, e às matrizes de saída de treino e de teste, respetivamente. É de frisar que também foi testada a incorporação de um *subset* de validação incorporado entre os dados de treino e de teste com um tamanho de 15%. O resultado da incorporação deste *subset* resultou no pior desempenho dos modelos em geral, especialmente os modelos que envolviam os algoritmos CNN e RNN, apresentando diminuições da métrica R^2 de 21 e 33 pontos percentuais, respetivamente. Sendo assim, foi escolhido incorporar apenas os *subsets* de treino e teste, visto que a principal razão pela incorporação de um *subset* de validação é diminuir o *overfitting* aos dados de treino para melhorar o desempenho nos dados de teste, o que não foi evidenciado.

```
split_limit = int(len(X)*0.8)

X_train = X[:split_limit]
X_test = X[split_limit:]

y_train = y[:split_limit]
y_test = y[split_limit:]
```

Listagem 4.7: Divisão em *subsets* de treino e teste

Após a execução destes passos, os dados estão prontos a serem utilizados no

processo de treino e de avaliação dos modelos, onde o conjunto de treino será utilizado para ajustar os parâmetros dos modelos, enquanto o conjunto de teste será usado para avaliar o desempenho do mesmo em dados ainda não vistos, através das diferentes métricas de desempenho referidas na Secção 3.2.

4.5 Modelos de Previsão

Nesta secção, serão detalhados os modelos de previsão utilizados para prever os preços das ações da Apple Inc.. Foram implementados um total de 44 modelos diferentes, desenvolvidos a partir dos algoritmos descritos na Secção 3.1 e suas combinações. Cada modelo foi treinado e avaliado com base nas métricas apresentadas anteriormente, utilizando um conjunto de 10 testes. O valor médio dos testes será apresentado nesta secção, enquanto o desvio padrão, bem como os melhores e piores resultados desses testes, serão detalhados no Anexo A, na Secção A.2.

4.5.1 Modelo LSTM

O primeiro modelo, consiste apenas em camadas de LSTM intercaladas por camadas de *dropout* seguidas de uma camada densa de *output*. Para isto, idealizou-se diversas configurações do modelo, como duas, três, quatro e cinco camadas de LSTM, sendo os hiperparâmetros 256 e 0.1 para as células de memória e a taxa de *dropout* respetivamente, que serão os valores iniciais a ser utilizados ao longo dos testes dos modelos. Ao fim disto, e depois de diversos testes de desempenho (Tabela 4.9), chegou-se à conclusão que a melhor versão do modelo era a que possuía duas camadas LSTM, visto que possui os menores valores das primeiras quatro métricas e o maior valor de R^2 entre os modelos.

Tabela 4.9: Modelos LSTM - Valor Médio

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
2 LSTM + 2 dropout	4.47343	40.44012	6.30300	3.67%	0.98542
3 LSTM + 3 dropout	6.39245	93.64143	8.81530	5.02%	0.96624
4 LSTM + 4 dropout	13.27586	326.50295	17.63884	9.49%	0.88230
5 LSTM + 5 dropout	12.78660	296.52519	16.96558	9.26%	0.89311

Outra ferramenta utilizada com o objetivo de otimizar este modelo foi o `BayesianOptimization` do `keras_tuner`. A função `BayesianOptimization` permite realizar uma procura sobre os hiperparâmetros do modelo, o que facilita a identificação das melhores combinações de hiperparâmetros para o modelo. Este método é útil para explorar uma vasta gama de possíveis configurações de maneira eficiente. Esta função foi então utilizada para procurar o número ideal de células de memória das camadas LSTM e também a melhor taxa a ser utilizadas nas camadas de *dropout*, sendo esta procura feita entre 64 e 256 unidades nas camadas LSTM e entre 0.1 e 0.5 para a taxa de *dropout*. Com isto, através deste método chegou-se à conclusão que o número ideal de células de memória são 256 e 256 para a primeira e segunda camada LSTM respetivamente, e 0.1 de taxa de *dropout* para todas as camadas de *dropout*.

O modelo foi compilado através do otimizador `Adam` com uma *learning rate* de 0.001 e a função de perda MSE. A camada de *output* utiliza uma função de ativação linear para prever os valores contínuos dos preços das ações. Além disso, foi também utilizados os *callbacks*, `EarlyStopping` e `ReduceLROnPlateau`, este ultimo utilizado para ajustar a *learning rate* durante o processo de treino do modelo, diminuindo-a quando o desempenho do modelo parar de melhorar, ajudando assim a melhorar a convergência do modelo. A arquitetura do modelo final e otimizado está evidenciada na Figura 4.8.

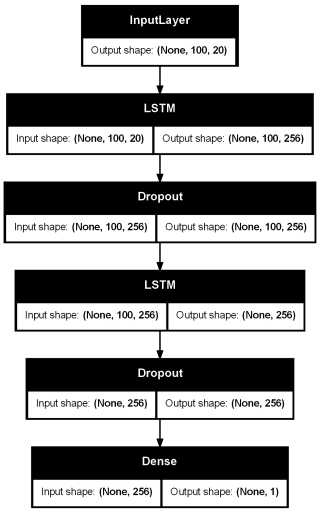


Figura 4.8: Modelo LSTM

4.5.2 Modelo GRU

Para o segundo modelo, GRU, adotou-se uma arquitetura semelhante ao modelo LSTM, com a diferença de que as camadas LSTM foram substituídas por camadas GRU. O melhor modelo evidenciado na Tabela 4.10 possui 2 camadas GRU. Assim como no caso anterior, foram realizadas buscas de hiperparâmetros através do

`BayesianOptimization`. A arquitetura final do modelo GRU incluiu duas camadas de GRU intercaladas por camadas de *dropout*, onde foram encontrados os valores ótimos de 192 e 256 unidades para a primeira e segunda camada GRU respetivamente, e 0.1 como taxas de *dropout* para ambas as camadas de *dropout*, como evidenciado através da Figura 4.9. Mais uma vez, foi também utilizado o otimizador `Adam` com uma *learning rate* inicial de 0.001 e MSE como função de perda, os *callbacks* `ReduceLROnPlateau` e `EarlyStopping`, e a camada de *output* utiliza a função de ativação linear.

Tabela 4.10: Modelos GRU - Valor Médio

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
2 GRU + 2 dropout	6.05663	70.33266	8.32452	4.52%	0.97458
3 GRU + 3 dropout	6.68641	86.15809	9.19105	4.98%	0.96894
4 GRU + 4 dropout	7.18087	101.49023	9.90276	5.33%	0.96341

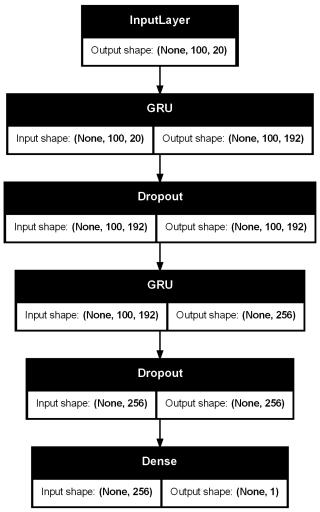


Figura 4.9: Modelo GRU

4.5.3 Modelo LSTM + GRU

Para o terceiro modelo, foi implementada uma combinação das arquiteturas LSTM e GRU e selecionado o melhor modelo conforme o desempenho apresentado na Tabela 4.11, aproveitando os pontos fortes de ambas para capturar melhor a complexidade dos dados.

Tabela 4.11: Modelos LSTM + GRU - Valor Médio

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
2 LSTM + 1 GRU	4.66154	44.05334	6.50991	3.84%	0.98412
2 LSTM + 2 GRU	5.67349	67.02369	7.84312	4.56%	0.97584
2 GRU + 1 LSTM	6.61557	90.00225	9.30896	4.96%	0.96756

A arquitetura final e otimizada do modelo, ilustrada pela Figura 4.10, é composta por:

- Uma camada LSTM com 160 unidades;
- Uma camada de *dropout* com uma taxa de 0.1;
- Uma camada GRU com 192 unidades;
- Uma camada de *dropout* com uma taxa de 0.1;
- Uma camada LSTM com 256 unidades;
- Uma camada de *dropout* com uma taxa de 0.4;
- Uma camada densa de *output* com uma unidade e função de ativação linear.

O modelo foi compilado utilizando o otimizador `Adam` com uma *learning rate* inicial de 0.001 e a função de perda MSE. Para melhorar a convergência do modelo, utilizou-se mais uma vez o *callback* `ReduceLROnPlateau` e também o `EarlyStopping`, reduzindo a *learning rate* num fator de 0.5 se a perda parar de melhorar por 5 passagens consecutivas.

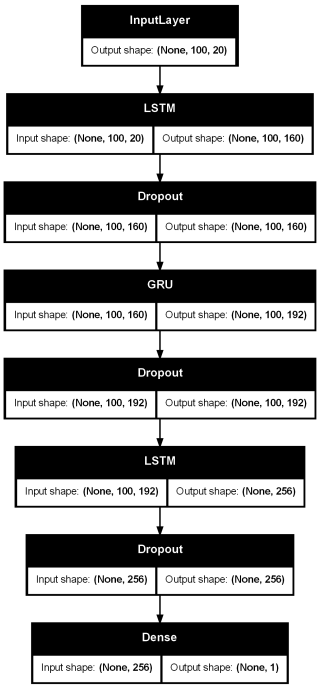


Figura 4.10: Modelo LSTM + GRU

4.5.4 Modelo CNN

Para avaliar o desempenho das CNN neste tipo de modelos e contexto, elas foram inicialmente utilizadas sozinhas, depois combinadas com LSTM e, por fim, com GRU, em diferentes combinações. Num primeiro estágio, as CNN foram testadas de forma isolada para observar como se comportam na tarefa da previsão dos preços das ações. Embora as CNN tenham demonstrado uma boa capacidade de identificar padrões nos dados, faltou-lhes a capacidade de acompanhar o preço das ações com uma previsão com um erro minimamente razoável, como evidenciado pela Figura 4.11.

Figura 4.11: Gráfico do modelo CNN

Tabela 4.12: Modelos CNN - Valor Médio

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
2 CNN	20.76584	724.55197	26.22878	14.65%	0.73881
3 CNN	20.06864	638.61763	25.10520	14.28%	0.76979
4 CNN	19.67139	625.07460	24.67110	13.97%	0.77467
2 CNN + 1 GRU	15.11428	363.29719	18.89790	10.87%	0.86904
2 CNN + 2 GRU	15.53270	388.34495	19.55753	11.08%	0.86001
2 CNN + 1 LSTM	16.25192	427.81101	20.43434	11.58%	0.84578
2 CNN + 2 LSTM	12.39301	268.00793	15.79895	8.95%	0.90339
3 CNN + 1 GRU	12.40942	255.89493	15.63696	8.97%	0.90775
3 CNN + 2 GRU	15.67831	396.61022	19.67131	11.21%	0.85703
3 CNN + 1 LSTM	16.20975	418.35857	20.35480	11.55%	0.84919
3 CNN + 2 LSTM	14.10113	332.74085	17.91368	10.06%	0.88005
3 CNN + 3 LSTM	18.60184	589.70890	24.19487	12.84%	0.78742
2 LSTM + 2 CNN	21.08852	709.25973	26.36363	15.25%	0.74432
2 LSTM + 3 CNN	16.82985	444.44235	20.96579	12.32%	0.83978
1 LSTM + 2 CNN	19.62809	613.01076	24.52411	14.26%	0.77902
1 LSTM + 3 CNN	17.40287	471.46663	21.60685	12.79%	0.83004
2 GRU + 2 CNN	21.17739	714.77203	26.39351	15.30%	0.74233
2 GRU + 3 CNN	16.98616	453.64580	21.13578	12.38%	0.83647
1 GRU + 2 CNN	19.42900	597.05003	24.13958	14.13%	0.78477
1 GRU + 3 CNN	16.49147	425.36262	20.47938	12.04%	0.84666

De seguida, foram feitas combinações de CNN com LSTM e GRU (Tabela 4.12), chegando à conclusão que os melhores modelos são os que combinam 3 camadas

CNN e 1 camada GRU e 2 camadas CNN com 2 camadas LSTM, representados nas Figuras 4.12 e 4.13, respetivamente.

Assim como os demais modelos, foi utilizado o algoritmo `BayesianOptimization` para encontrar os melhores hiperparâmetreos dos modelos, sendo estes 256, 128, 224 e 256 para as camadas CNN e GRU respetivamente no modelo composto por 3 CNN + 1 GRU, e 256 e 128 para as duas camadas CNN e 256 e 224 para as duas camadas LSTM do modelo composto por duas camadas de CNN e LSTM. Para além disso foi também utilizado os *callbacks* `ReduceLROnPlateau` e `EarlyStopping` e o otimizador `Adam`.

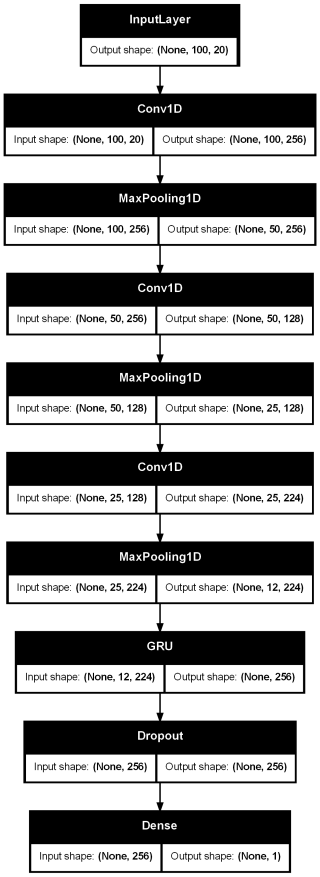


Figura 4.12: Modelo CNN + GRU

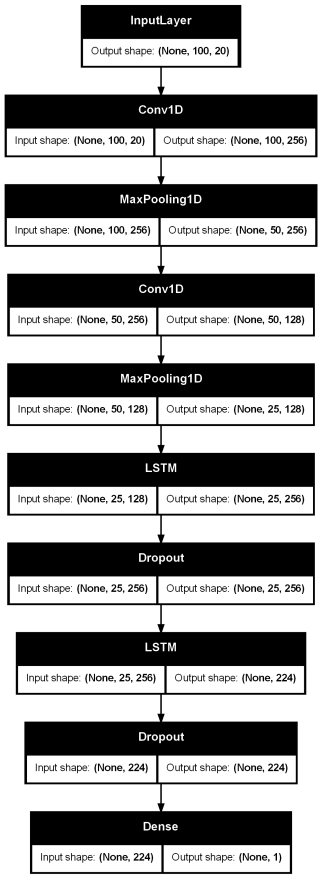


Figura 4.13: Modelo CNN + LSTM

4.5.5 Modelo RNN

Para o modelo RNN, assim como os modelos anteriores foram testadas diferentes combinações entre camadas RNN, sozinhas, e com camadas GRU e LSTM, como evidenciados na Tabela 4.13. Diferentemente dos modelos CNN a maioria das combinações destas camadas produziam resultados pouco desejados, mas, mesmo assim, os modelos compostos por duas camadas de GRU seguido de duas camadas de RNN (Figura 4.14) e o composto por uma camada LSTM seguido de duas camadas RNN (Figura 4.15) obtiveram os melhores resultados. Para além destes dois modelos foi também escolhido o modelo composto por apenas RNN (Figura 4.16) visto que este também apresentou um desempenho bastante bom sem a necessidade da implementação de outro tipo de camadas.

Tabela 4.13: Modelos RNN - Valor Médio

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
2 RNN	13.42476	312.82043	17.50414	9.40%	0.88723
3 RNN	11.00183	203.15732	14.07731	8.00%	0.92676
4 RNN	54.26821	5309.99157	67.45029	38.61%	-0.91418
2 RNN + 1 GRU	16.08922	487.06017	21.59222	11.02%	0.82442
2 RNN + 1 LSTM	35.61142	2324.05876	46.58004	24.15%	0.16221
3 RNN + 1 GRU	52.63348	4653.71260	67.37217	36.14%	-0.67760
3 RNN + 2 GRU	58.07854	5481.42100	73.94511	39.95%	-0.97597
3 RNN + 1 LSTM	51.58560	4527.09365	65.51352	36.24%	-0.63195
3 RNN + 2 LSTM	49.62855	4271.30501	63.29981	34.38%	-0.53974
1 GRU + 2 RNN	34.71628	3311.86846	43.18345	24.90%	-0.19388
2 GRU + 2 RNN	6.86930	85.52800	9.10183	5.18%	0.96917
1 LSTM + 2 RNN	5.20305	59.16789	7.28287	4.05%	0.97867
2 LSTM + 2 RNN	15.04343	360.47080	18.85734	10.79%	0.87006

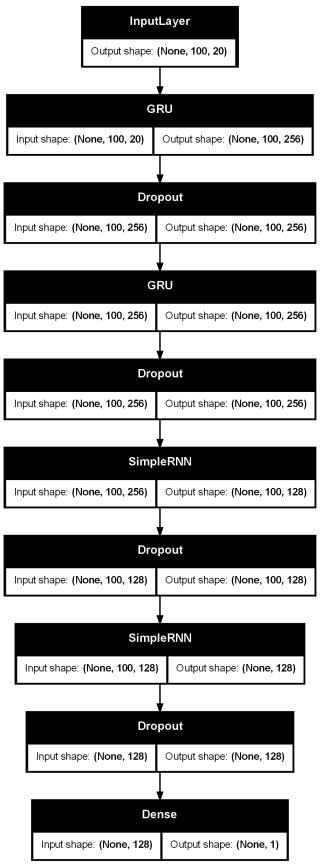


Figura 4.14: Modelo RNN + GRU

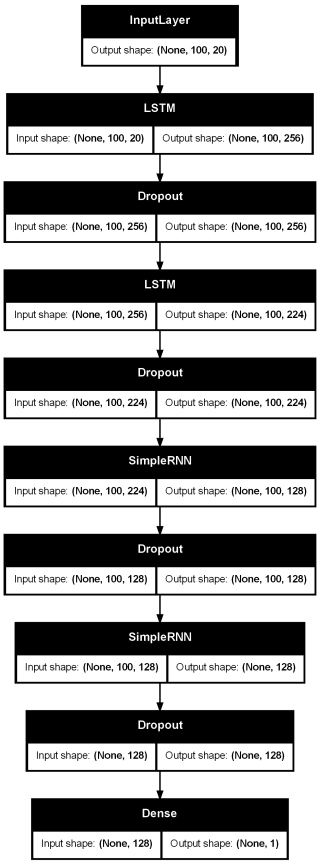


Figura 4.15: Modelo RNN + LSTM

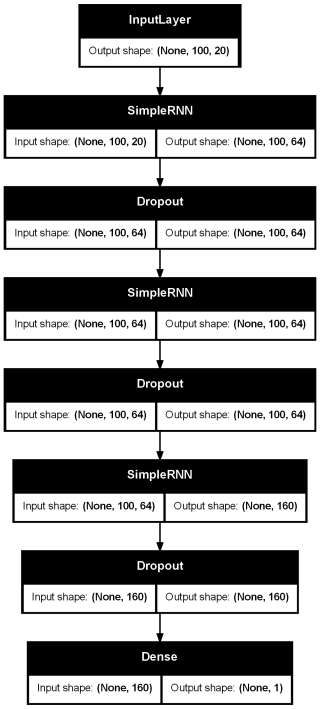


Figura 4.16: Modelo RNN

Ao efetuar a pesquisa dos melhores parâmetros para utilizar nestes modelos através do algoritmo `BayesianOptimization`, obtiveram-se os seguintes valores para o modelo composto por camadas RNN e GRU:

- Uma camada GRU com 256 unidades;
- Uma camada de *dropout* com uma taxa de 0.1;
- Uma camada GRU com 224 unidades;
- Uma camada de *dropout* com uma taxa de 0.1;
- Uma camada RNN com 128 unidades;
- Uma camada de *dropout* com uma taxa de 0.1;
- Uma camada RNN com 128 unidades;
- Uma camada de *dropout* com uma taxa de 0.1;

A otimização do modelo composto por camadas RNN e LSTM apresentou os seguintes hiperparâmetros:

- Uma camada LSTM com 256 unidades;
- Uma camada de *dropout* com uma taxa de 0.1;
- Uma camada RNN com 128 unidades;
- Uma camada de *dropout* com uma taxa de 0.1;
- Uma camada RNN com 128 unidades;
- Uma camada de *dropout* com uma taxa de 0.1;

E por último a otimização do modelo composto por camadas RNN apresentou os seguintes hiperparâmetros:

- Uma camada RNN com 64 unidades;
- Uma camada de *dropout* com uma taxa de 0.1;
- Uma camada RNN com 64 unidades;
- Uma camada de *dropout* com uma taxa de 0.1;
- Uma camada RNN com 160 unidades;
- Uma camada de *dropout* com uma taxa de 0.1.

4.5.6 Modelo XGBoost

Para o modelo XGBoost, os dados de entrada foram preparados de maneira diferente comparado aos demais modelos. Primeiramente, a preparação dos dados sofreu alterações, para poder acomodar as especificidades do modelo em questão, em relação ao descrito na Subsecção 4.4.

Diferentemente do processo anterior, onde os dados foram organizados em sequências temporais fixas através de uma *input window*, para o XGBoost, os dados foram divididos em conjuntos de treino e teste utilizando uma simples divisão 80% e 20% respetivamente (Listagem 4.8).

```
split_limit = int(len(features) * 0.8)
train, test = features[:split_limit], features[split_limit
    :]
```

Listagem 4.8: *Train* e *Test Split*

Após esta divisão, as variáveis independentes foram separadas da variável dependente, assim como o processo anterior (Listagem 4.9).

```
X = train[:,:-1]
y = train[:,-1]
```

Listagem 4.9: Entrada e Saída do modelo

De seguida, o modelo foi configurado através da função `XGBRegressor` e dos parâmetros evidenciados na Tabela 4.14 que foram obtidos através de uma otimização dos valores através da função `GridSearchCV` (Listagem 4.10). Estes intervalos de valores foram obtidos através do documento disponível na plataforma Kaggle denominado de "*A Guide on XGBoost hyperparameters tuning*" [76].

```
params = {'max_depth': [3,10],
          'learning_rate': [0.01, 0.2],
          'n_estimators': [500, 2000],
          'colsample_bytree': [0.5, 1],
          'gamma': [0,10],
          ''}
xgbr = XGBRegressor(seed = 20)
modl = GridSearchCV(estimator=xgbr, param_grid=params,
    scoring='neg_mean_squared_error', verbose=1)
```

Listagem 4.10: Otimização dos parâmetros

Tabela 4.14: Parâmetros do modelo XGBoost

Parâmetro	**Valor**
`n_estimators`	2000
`colsample_bytree`	1
`learning_rate`	0.2
`max_depth`	3
`gamma`	0

Após a preparação dos dados e a otimização dos hiperparâmetros, o próximo passo é criar funções para realizar todas as previsões através do modelo. Sendo assim, foram criadas as funções `modelo_predict` e `modelo_predict_2` apresentadas nas Listagens 4.11 e 4.12, respetivamente.

```
1 def modelo_predict(train, val):
2     train = np.array(train)
3     X = train[:, :-1],
4     y = train[:, -1]
5     model = XGBRegressor(objective='reg:squarederror',
          n_estimators=2000, colsample_bytree=1, learning_rate
          =0.2, max_depth=3, gamma=0)
6     model.fit(X, y)
7     val = np.array(val).reshape(1, -1)
8     predict = model.predict(val)
9     return predict[0]
```

Listagem 4.11: Função `modelo_predict`

```
1 def modelo_predict_2(train, test):
2     predictions = []
3     history = [x for x in train]
4
5     for i in range(len(test)):
6         X_test, y_test = test[i, :-1], test[i, -1]
7         pred = modelo_predict(history, X_test)
8         predictions.append(pred)
9
10        history.append(test[i])
11
12    return test[:, -1], predictions
```

Listagem 4.12: Função `modelo_predict_2`

De seguida é chamada a função `modelo_predict_2`, e as previsões são introduzidas num *dataframe* juntamente com os dados iniciais da variável `test`.

Capítulo 5

Resultados

Neste capítulo são apresentados os resultados referentes ao desempenho de todos os modelos criados na previsão de séries temporais. A análise inclui a avaliação das métricas de desempenho escolhidas, a comparação entre os diferentes modelos testados e a interpretação das previsões dos modelos através de gráficos.

5.1 *Time Series Cross Validation*

Foi utilizada a técnica *Time Series Cross Validation* para avaliar e comparar o desempenho dos modelos com diferentes divisões dos *subsets* de treino e teste e também diferentes curvas de preço. Esta técnica é ideal para séries temporais, pois mantém a sequência temporal dos dados, ao contrário da *Cross Validation* tradicional, onde a ordem dos dados não precisa ser preservada. No *Time Series Cross Validation*, o *test set* consiste sempre em dados posteriores ao *training set*, garantindo assim que o modelo seja avaliado com base na sua capacidade de prever dados futuros a partir de informações passadas [77].

Para esta análise, foram utilizadas 10 *folds* (Figura 5.1). Significando que o modelo foi treinado 10 vezes, cada vez com uma janela de tempo menor, validando sempre com os dados futuros não vistos até aquele momento. Assim como todas as análises de desempenho até agora, foram utilizadas as métricas MAE, MSE, RMSE, MAPE e R^2 para cada *fold*.

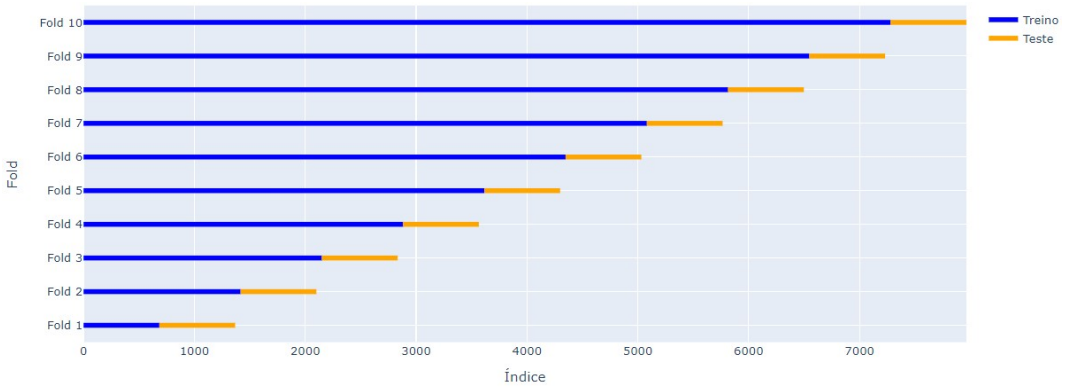


Figura 5.1: Divisão de *subsets*

Um passo extra a ressaltar é o facto de cada *subset* estar individualmente redimensionado entre 0 e 1, através da função `MinMaxScaler`, com o objetivo de fornecer uma entrada constante ao modelo, conseguindo assim variar apenas a dimensão dos *subsets* e não a dimensão dos valores em si. De seguida, após a previsão do modelo ser efetuada é necessário voltar a redimensionar a previsão para os valores reais. Para isso, os valores que previamente estavam dimensionados entre 0 e 1 foram redimensionados para se conterem entre 0 e 100.

Com este método, consegue-se resolver dois problemas que possam surgir. Inicialmente a ilegibilidade dos valores, visto que as métricas calculadas apresentavam valores bastante pequenos, e também manter uma escala constante entre os diversos *subsets*, visto que se redimensionassem os *subsets* para os seus valores reais (antes da passagem pela função `MinMaxScaler`), estes apresentariam diferentes escalas, inutilizando a interpretação das métricas absolutas.

Sendo assim, foi aplicado esta técnica a todos os modelos e tabelados os seus resultados nas Tabelas 5.1-5.9.

Tabela 5.1: Resultados do Modelo LSTM

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
Fold 1	11.91214	173.74637	13.18129	28.19%	0.72660
Fold 2	5.48417	64.95642	8.05955	11.48%	0.93242
Fold 3	7.12520	90.16903	9.49574	18.75%	0.89422
Fold 4	7.82829	118.52146	10.88676	17.69%	0.85099
Fold 5	5.50201	49.62010	7.04415	16.50%	0.93648
Fold 6	4.23446	29.60616	5.44115	12.99%	0.94402
Fold 7	4.62862	34.22093	5.84987	16.33%	0.94550
Fold 8	2.90414	13.93436	3.73288	13.31%	0.98302
Fold 9	2.37138	12.61277	3.55145	9.07%	0.98228
Fold 10	7.26365	82.00893	9.05588	18.41%	0.86375

Tabela 5.2: Resultados do Modelo GRU

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
Fold 1	9.38743	112.10150	10.58780	24.17%	0.81665
Fold 2	3.80284	30.20025	5.49548	9.05%	0.96462
Fold 3	6.53096	80.15704	8.95305	17.95%	0.90176
Fold 4	4.52784	40.58004	6.37025	11.26%	0.94898
Fold 5	3.46230	21.34289	4.61984	11.55%	0.97268
Fold 6	2.92680	15.30314	3.91192	9.35%	0.97106
Fold 7	3.27320	18.19441	4.26549	12.26%	0.97102
Fold 8	2.85786	12.80357	3.57821	11.79%	0.98474
Fold 9	1.74535	7.79488	2.79193	7.20%	0.98905
Fold 10	4.22626	28.97903	5.38322	14.54%	0.95185

Tabela 5.3: Resultados do Modelo LSTM + GRU

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
Fold 1	15.61992	304.28257	17.44370	32.22%	0.42145
Fold 2	6.58491	90.83563	9.53077	12.86%	0.91279
Fold 3	9.75717	146.71980	12.11279	23.25%	0.83451
Fold 4	8.35856	135.42022	11.63702	18.56%	0.82974
Fold 5	6.63776	71.98634	8.48448	17.41%	0.90786
Fold 6	4.11538	30.06600	5.48325	12.28%	0.94315
Fold 7	5.46603	48.51339	6.96516	18.34%	0.92274
Fold 8	3.64409	20.73590	4.55367	14.48%	0.97792
Fold 9	3.40643	26.82287	5.17908	10.67%	0.96232
Fold 10	13.67531	288.78473	16.99367	27.59%	0.52021

Tabela 5.4: Resultados do Modelo CNN + GRU

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
Fold 1	15.84411	394.05949	19.85093	115.52%	0.44548
Fold 2	5.13977	39.20318	6.26124	16.41%	0.95428
Fold 3	6.78728	80.04204	8.94662	20.03%	0.89396
Fold 4	7.10962	86.71263	9.31196	17.15%	0.89098
Fold 5	8.52569	106.01388	10.29630	20.45%	0.89054
Fold 6	1.78212	6.78008	2.60386	6.62%	0.98718
Fold 7	3.08410	15.72239	3.96515	12.45%	0.97496
Fold 8	3.51261	18.40017	4.28954	12.85%	0.97906
Fold 9	6.00269	79.04072	8.89048	14.95%	0.88897
Fold 10	7.17542	74.19858	8.61386	18.17%	0.87673

Tabela 5.5: Resultados do Modelo CNN + LSTM

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
Fold 1	9.13247	112.04292	10.58503	31.55%	0.84233
Fold 2	5.02402	42.28999	6.50308	14.17%	0.95217
Fold 3	9.32432	141.95356	11.91443	23.72%	0.82479
Fold 4	7.49673	101.83279	10.09122	17.12%	0.87197
Fold 5	13.93033	298.15672	17.26722	32.21%	0.61835
Fold 6	2.48617	12.01712	3.46657	8.58%	0.97728
Fold 7	5.35937	49.82275	7.05852	17.21%	0.92065
Fold 8	3.30675	17.96454	4.23846	13.03%	0.97811
Fold 9	4.73759	51.08354	7.14728	13.07%	0.92824
Fold 10	13.72379	268.47856	16.38532	28.15%	0.55395

Tabela 5.6: Resultados do Modelo GRU + RNN

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
Fold 1	20.43294	520.61169	22.81692	36.14%	-0.25661
Fold 2	5.03719	54.84815	7.40595	10.53%	0.94288
Fold 3	12.52122	236.24971	15.37042	26.12%	0.74038
Fold 4	4.65742	41.43034	6.43664	12.35%	0.94791
Fold 5	5.17978	43.72179	6.61225	15.30%	0.94403
Fold 6	2.89011	15.50343	3.93744	9.56%	0.97068
Fold 7	3.10744	16.44907	4.05575	13.83%	0.97380
Fold 8	3.02516	15.28663	3.90981	12.22%	0.98314
Fold 9	1.95716	9.06983	3.01162	8.35%	0.98726
Fold 10	4.46924	32.06349	5.66246	16.79%	0.94673

Tabela 5.7: Resultados do Modelo LSTM + RNN

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
Fold 1	21.60152	584.82707	24.18320	37.04%	-0.43740
Fold 2	12.30615	311.19502	17.64072	18.32%	0.79094
Fold 3	9.75456	147.50570	12.14519	40.51%	0.79226
Fold 4	8.82793	153.85131	12.40368	19.76%	0.80657
Fold 5	9.42369	143.48814	11.97865	20.42%	0.88240
Fold 6	5.96017	65.14044	8.07096	15.85%	0.87682
Fold 7	6.84051	71.70855	8.46809	18.72%	0.91875
Fold 8	4.25580	27.70397	5.26346	14.54%	0.97135
Fold 9	3.99332	35.35340	5.94587	11.98%	0.95034
Fold 10	8.53610	106.69535	10.32934	20.17%	0.82274

Tabela 5.8: Resultados do Modelo RNN

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
Fold 1	18.79646	450.99004	21.23653	36.52%	-0.13532
Fold 2	7.83663	146.67303	12.11086	12.44%	0.88968
Fold 3	10.08343	169.39137	13.01504	23.21%	0.80053
Fold 4	7.55458	117.31707	10.83130	17.20%	0.85250
Fold 5	5.35813	50.49704	7.10613	15.70%	0.93536
Fold 6	6.21144	78.75409	8.87435	15.83%	0.85108
Fold 7	6.44382	71.14777	8.43491	16.84%	0.92295
Fold 8	6.78867	77.44861	8.80049	15.54%	0.93456
Fold 9	4.94555	57.88303	7.60809	13.15%	0.91869
Fold 10	7.40324	80.10582	8.95019	18.35%	0.86691

Tabela 5.9: Resultados do Modelo XGBoost

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
Fold 1	3.69458	23.64308	4.86242	9.75%	0.96143
Fold 2	6.79941	112.42946	10.60328	32.67%	0.84349
Fold 3	5.93764	56.58240	7.52213	23.35%	0.91745
Fold 4	5.70026	69.96937	8.36477	13.82%	0.94610
Fold 5	5.62707	53.09575	7.28668	16.83%	0.94447
Fold 6	2.55833	11.56099	3.40014	8.48%	0.98544
Fold 7	2.27182	9.39108	3.06449	7.90%	0.98974
Fold 8	2.16358	8.24167	2.87083	9.37%	0.99018
Fold 9	1.96006	8.20051	2.86365	6.53%	0.99105
Fold 10	3.69458	23.64308	4.86242	9.75%	0.96143

A análise dos resultados revela que o modelo constituído por apenas duas camadas GRU e o modelo XGBoost apresentaram o melhor desempenho geral em comparação aos outros modelos testados, conforme evidenciado pelas baixas médias de MAE, MSE, RMSE, MAPE e R^2 em todas as *folds*. As Tabelas 5.10 e 5.11 e as subsequentes Figuras 5.2-5.6 fornecem um resumo dos resultados médios e dos desvios padrão das métricas calculadas para cada modelo testado.

Tabela 5.10: Média das métricas calculadas

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
LSTM	5.92541	66.93970	7.62987	16.27%	0.90593
GRU	4.27408	36.74568	5.59572	12.91%	0.94724
LSTM + GRU	7.72656	116.41675	9.83836	18.77%	0.82327
CNN + GRU	6.49634	90.01732	8.30299	25.46%	0.87821
CNN + LSTM	7.45215	109.56400	9.46571	19.88%	0.84678
GRU + RNN	6.32777	98.52341	7.92193	16.12%	0.81802
LSTM + RNN	9.14998	164.74689	11.64290	21.73%	0.73748
RNN	8.14220	130.02079	10.69679	18.48%	0.78369
XGBoost	4.04073	37.67574	5.57008	13.85%	0.95308

Tabela 5.11: Desvio padrão das métricas calculadas

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
LSTM	2.63375	48.33650	2.95376	4.99%	0.07316
GRU	2.08338	31.83310	2.33101	4.70%	0.04938
LSTM + GRU	3.97725	99.07641	4.42984	6.65%	0.18335
CNN + GRU	3.70506	106.46030	4.59103	30.27%	0.15004
CNN + LSTM	3.83563	95.58068	4.46817	8.00%	0.14019
GRU + RNN	5.46571	154.36099	5.95051	8.19%	0.36459
LSTM + RNN	4.81639	160.07914	5.40272	8.93%	0.39642
RNN	3.80369	113.08736	3.94962	6.64%	0.30914
XGBoost	1.72911	32.91994	2.57875	7.90%	0.04330

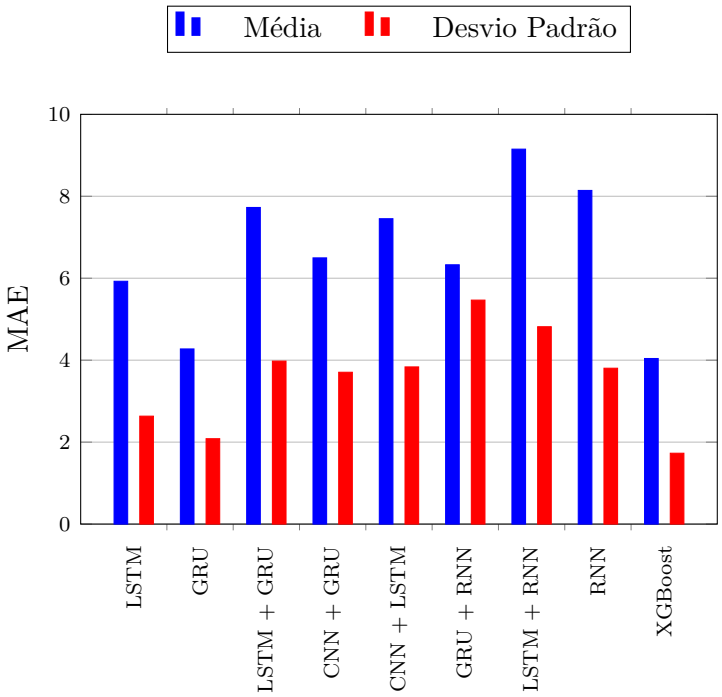


Figura 5.2: Média e Desvio padrão da métrica MAE

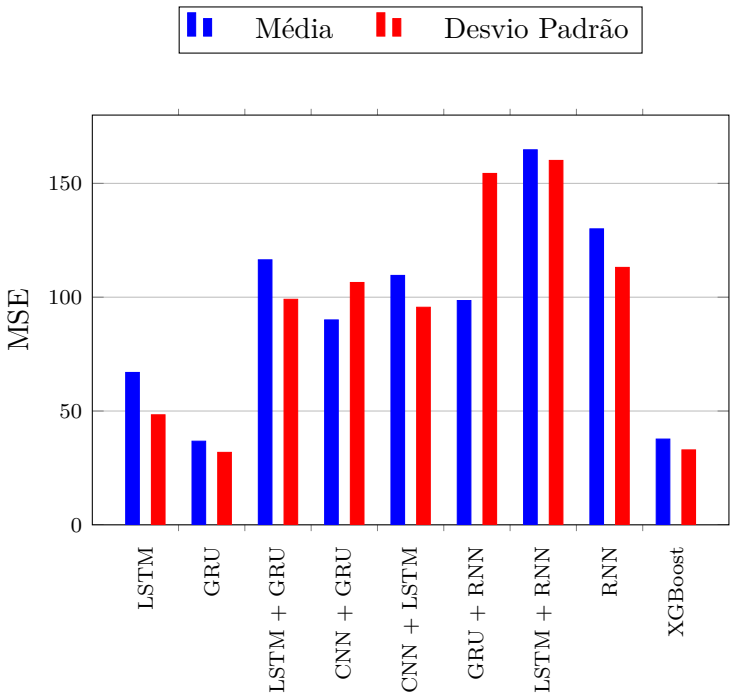


Figura 5.3: Média e Desvio padrão da métrica MSE

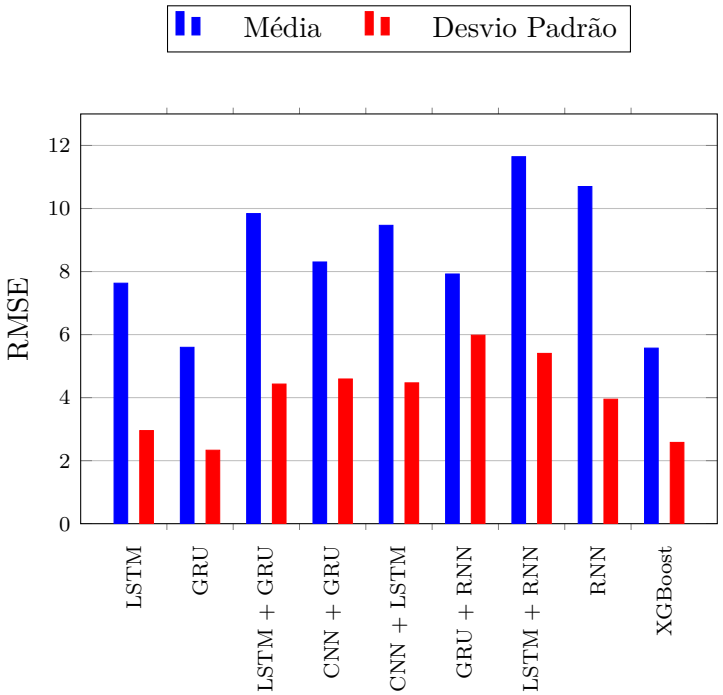


Figura 5.4: Média e Desvio padrão da métrica RMSE

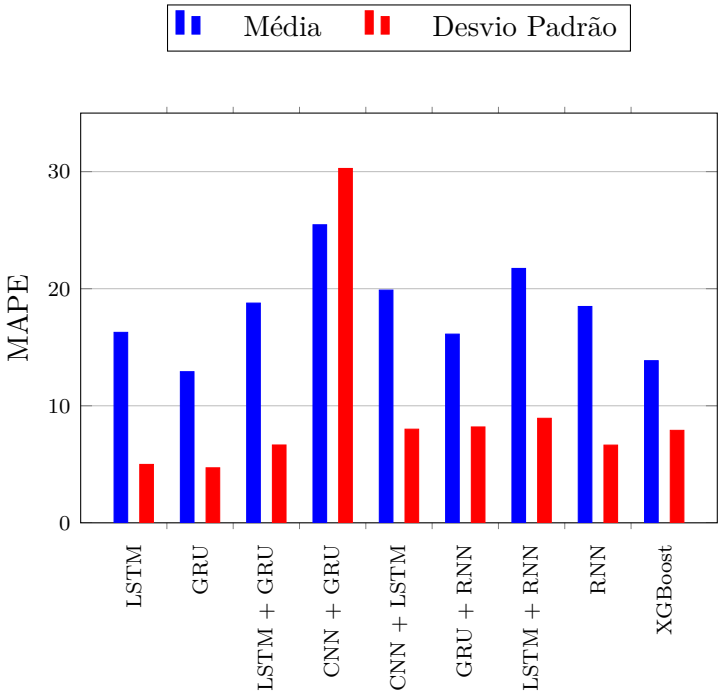


Figura 5.5: Média e Desvio padrão da métrica MAPE

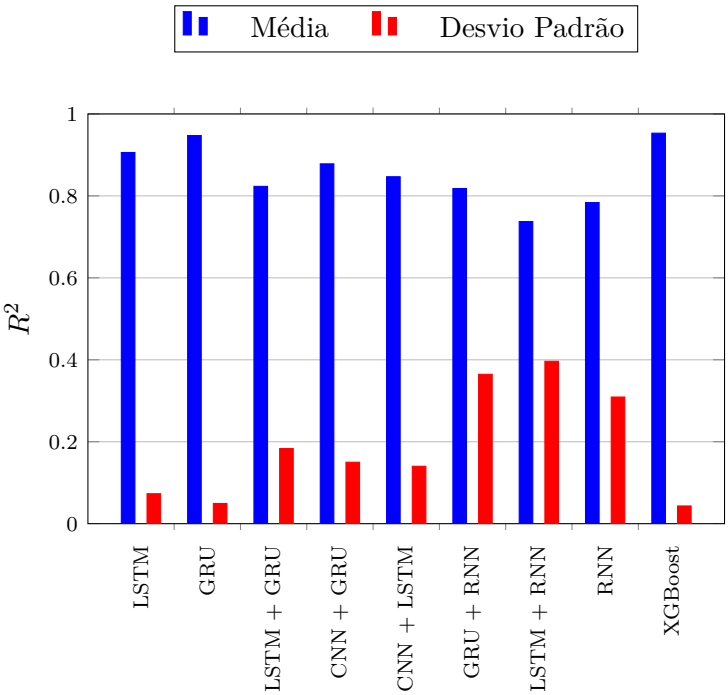


Figura 5.6: Média e Desvio padrão da métrica R^2

5.2 Análise dos Resultados

Entre os modelos testados, GRU obteve a melhor MSE e MAPE, enquanto o modelo XGBoost obteve a melhor MAE, RMSE e R^2, sendo, portanto, os dois modelos que mais se destacaram, com valores bastante similares em todas as métricas.

De seguida o modelo com melhor desempenho é o modelo LSTM. Este modelo, embora apresente resultados bons, quando comparado com os dois mencionados previamente é simplesmente pior em todas as métricas. Ao analisar de perto este modelo, verifica-se que a grande razão pelos seus piores valores é o desempenho na *fold* 1, assim como o modelo composto por camadas de RNN e GRU e o modelo composto por CNN e GRU que consistentemente ficaram classificados na 5ª e 4ª posição, respetivamente.

Relativamente aos modelos LSTM + GRU, CNN + LSTM e RNN, estes apresentaram resultados consideravelmente piores relativamente a todos os modelos mencionados previamente. Embora tenham apresentado, em média, valores piores, estes modelos ainda são capazes de fazer ótimas previsões evidenciado pelo desempenho do modelo CNN + LSTM na *fold* 6 onde ficou apenas atrás do modelo CNN + GRU.

Por fim, o modelo que apresentou o pior desempenho foi o modelo composto por camadas RNN e LSTM, o que se tornou uma surpresa visto que seria esperado que a adição de uma camada LSTM melhorasse o desempenho do modelo quando comparado a um modelo composto apenas por camadas RNN.

Um ponto a ressaltar é a inclusão de camadas GRU que mostrou ter um impacto bastante positivo no desempenho dos modelos, para o GRU isolado, o modelo LSTM + GRU, o modelo CNN + GRU e o modelo GRU + RNN.

Para concluir este estudo, são apresentados os gráficos de previsão dos modelos nas suas melhores *folds* (Figuras 5.7-5.15). A *fold* 9 foi a melhor para todos os modelos, com exceção dos compostos por camadas CNN, ou seja, os modelos CNN + LSTM e CNN + GRU, nos quais o melhor resultado foi obtido na *fold* 6.

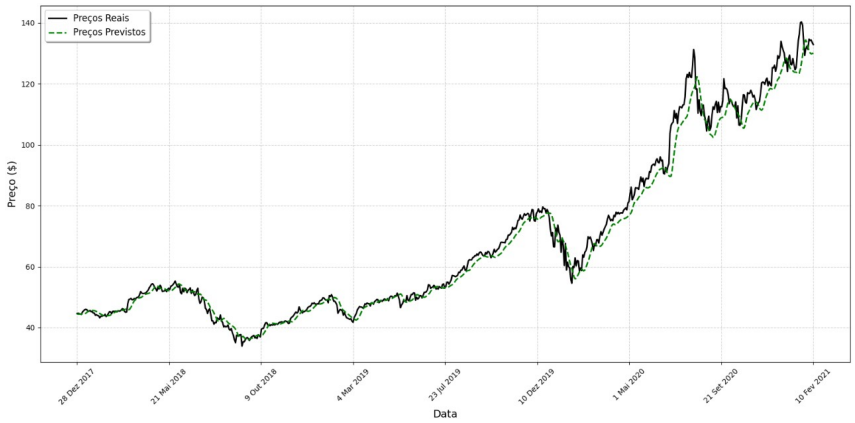


Figura 5.7: Gráfico de previsão da 9ª *fold* do modelo LSTM

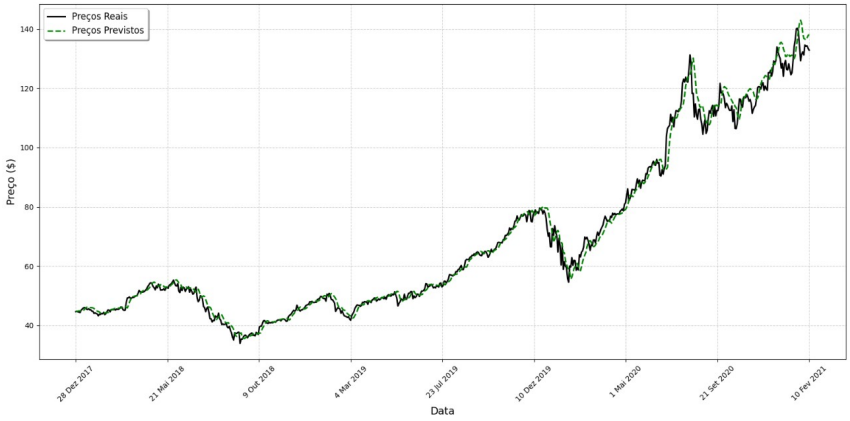


Figura 5.8: Gráfico de previsão da 9ª *fold* do modelo GRU

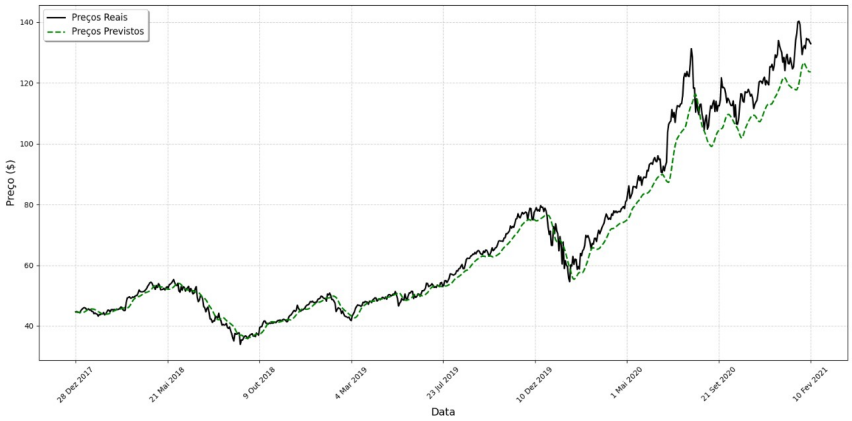


Figura 5.9: Gráfico de previsão da 9ª *fold* do modelo LSTM + GRU

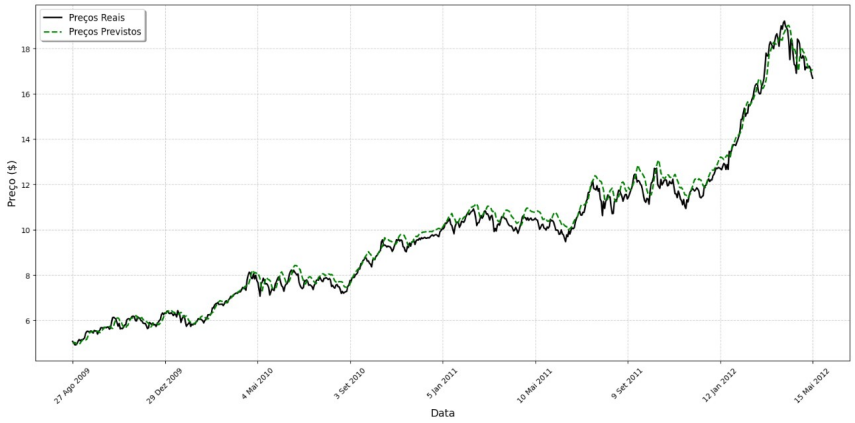


Figura 5.10: Gráfico de previsão da 6ª *fold* do modelo CNN + GRU

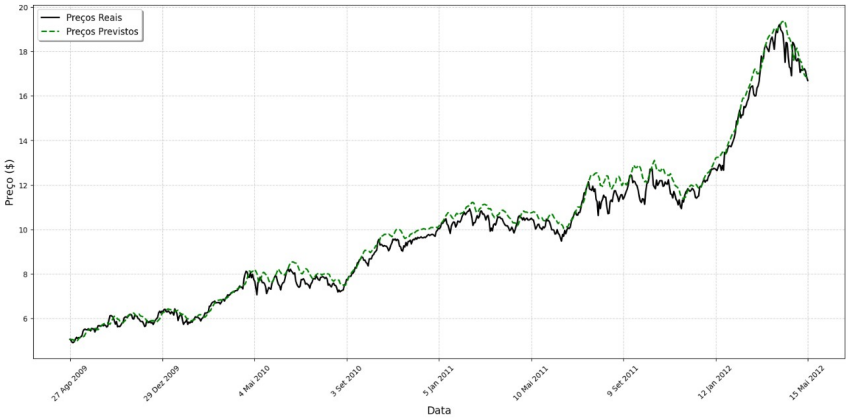


Figura 5.11: Gráfico de previsão da 6ª *fold* do modelo CNN + LSTM

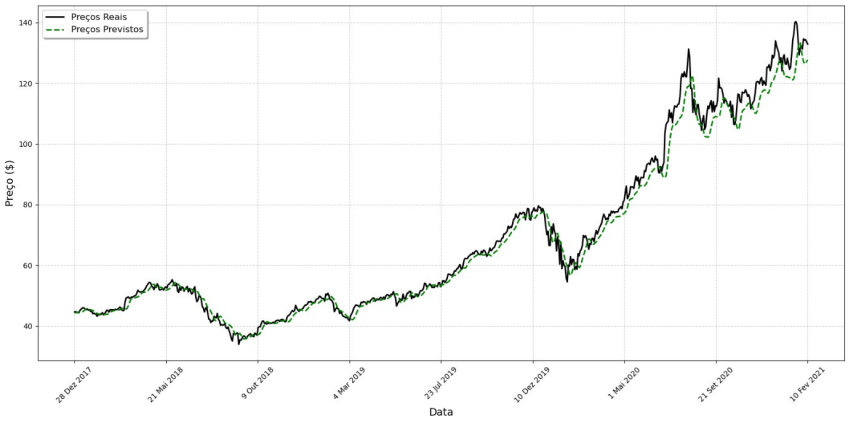


Figura 5.12: Gráfico de previsão da 9ª *fold* do modelo GRU + RNN

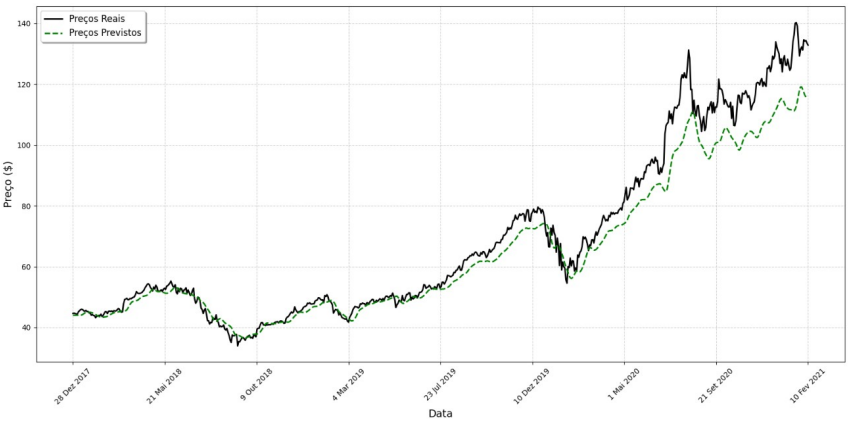


Figura 5.13: Gráfico de previsão da 9ª *fold* do modelo LSTM + RNN

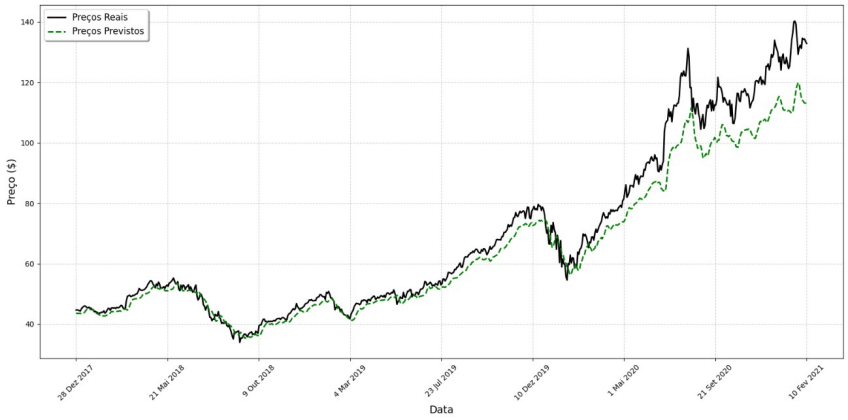


Figura 5.14: Gráfico de previsão da 9ª *fold* do modelo RNN

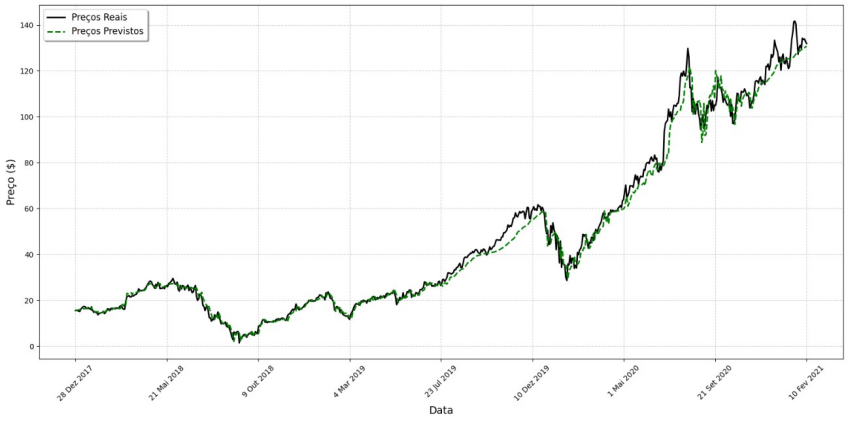


Figura 5.15: Gráfico de previsão da 9ª *fold* do modelo XGBoost

Ao analisar os gráficos, percebe-se que os modelos GRU, LSTM e XGBoost demonstram o melhor desempenho para a *fold* 9, destacando-se pela sua capacidade de acompanhar de perto e precisa as diversas oscilações de preço. O modelo GRU + RNN também apresenta um bom desempenho, ficando próximo dos melhores modelos. Na *fold* 6, todos os modelos tiveram um bom desempenho, atribuido à menor complexidade desta *fold*, que apresenta menos oscilações de grande amplitude em comparação com as demais. Isto indica que estes modelos conseguem lidar muito bem com cenários de menor variabilidade.

Por outro lado, observamos que os modelos RNN e LSTM + RNN não conseguem acompanhar o preço real com a mesma precisão que os demais, especialmente nas previsões finais, onde o erro percentual tende a aumentar em relação às previsões iniciais, sugerindo que estes modelos podem ter dificuldades para capturar padrões que exibam altas oscilações e a acompanhar aumentos repentinos de amplitude.

Capítulo 6

Conclusões

Este trabalho explorou a aplicação de diferentes técnicas de previsão de preços de ações, através da aplicação e comparação de modelos como LSTM, GRU, CNN, RNN e XGBoost, onde foi obtido uma visão das capacidades e limitações destas abordagens no contexto de séries temporais.

A análise mostrou que, embora cada modelo tenha as suas próprias características, a escolha da abordagem mais eficiente depende fortemente das especificidades dos dados e do objetivo da previsão. A complexidade dos modelos avançados, como XGBoost e GRU refletiu-se pelo seu desempenho geral, sugerindo que podem ser particularmente eficazes a capturar padrões e fazer previsões precisas em séries temporais mais complexas como o preço de ações. Em contraste, modelos como RNN e CNN mostraram desempenhos variáveis, mostrando que a configuração do modelo precisa de ser ajustada, visto que quando combinado com camadas GRU mostravam um desempenho bastante superior do que quando testadas sozinhas ou em combinação com camadas LSTM.

6.1 Trabalho Futuro

Para trabalho futuro e possíveis melhorias deste trabalho, há várias direções que podem ser exploradas. Uma abordagem inicial seria a inclusão de novos algoritmos de *Machine Learning*, assim como efetuar testes com ainda mais configurações de diferentes algoritmos. Outra possível melhoria seria diversificar o *dataset* utilizado, incluindo um conjunto mais diversificado de ações abrangendo diferentes mercados

financeiros e percentuais de crescimento. Além disso, a implementação de novos *features* mais avançados como indicadores económicos ou até a inclusão de diferentes *stocks* no mesmo *dataset* para refletir uma noção mais robusta do mercado como um todo.

Outra possível linha de pensamento seria a aplicação de técnicas de classificação para prever a direção dos preços, ou seja se iriam subir ou descer. Isto permitiria uma análise detalhada e uma abordagem mais relevante para a tomada de decisões financeiras. A utilização de modelos de *ensemble*, mencionados previamente na Secção 2.3.1, poderia ser uma opção viável para atingir este objetivo.

Em suma, a exploração contínua destas áreas poderá trazer avanços significativos na previsão de preços de ações e na aplicação de técnicas de *Machine Learning* em séries temporias, mais especificamente no domínio financeiro.

Referências

[1] CFI Team, "Stock market." Corporate Finance Institute, Available at https://corporatefinanceinstitute.com/resources/career-map/sell-side/capital-markets/stock-market/, 2023. (Last accessed in 02/03/2024). [Citado na página 6]

[2] A. Hayes, "Dividends: Definition in stocks and how payment work." Investopedia, Available at https://www.investopedia.com/terms/d/dividend.asp, 2023. (Last accessed in 02/03/2024). [Citado na página 6]

[3] L. Downey, "Efficient market hypothesis (emh): Definition and critique." Investopedia, Available at https://www.investopedia.com/terms/e/efficientmarkethypothesis.asp, 2024. (Last accessed in 04/03/2024). [Citado na página 7]

[4] Insider Monkey, "Here's what warren buffet thinks about the efficient market hypothesis." Business Insider, Available at https://www.businessinsider.com/warren-buffett-on-efficient-market-hypothesis-2010-12, 2010. (Last accessed in 06/03/2024). [Citado na página 8]

[5] Winvesta, "Fundamental analysis: A complete guide." Winvesta, Available at https://www.winvesta.in/blog/fundamental-analysis-a-complete-guide. (Last accessed in 06/03/2024). [Citado na página 8]

[6] A. Hayes, "Technical analysis: What it is and how to use it in investing." Investopedia, Available at https://www.investopedia.com/terms/t/technicalanalysis.asp, 2024. (Last accessed in 06/03/2024). [Citado na página 9]

[7] C. Mitchell, "How to use trendlines in your trading." the balance, Available at https://www.thebalancemoney.com/how-to-effectively-use-trendlines-in-your-trading-1030884, 2021. (Last accessed in 06/03/2024). [Citado nas páginas vii e 9]

[8] D. Horstmeyer, A. El Boury and D. Hardin, "Technical analysis revisited: Moving averages = above average returns?." Enterprising Investor, Available at https://blogs.cfainstitute.org/investor/2022/02/08/technical-

analysis-revisited-moving-averages-above-average-returns/, 2022. (Last accessed in 06/03/2024). [Citado nas páginas vii e 9]

[9] C. Mitchell, "Market sentiment indicator: How it's used in analysis and types." Investopedia, Available at https://www.investopedia.com/terms/s/sentimentindicator.asp, 2024. (Last accessed in 07/03/2024). [Citado na página 9]

[10] "What is machine learning (ml)?." IBM, Available at https://www.ibm.com/topics/machine-learning. (Last accessed in 08/03/2024). [Citado na página 10]

[11] "What is machine learning (ml)?." Berkeley School of Information, Available at https://ischoolonline.berkeley.edu/blog/what-is-machine-learning/, 2020. (Last accessed in 08/03/2024). [Citado nas páginas 10, 11 e 12]

[12] R. Spanton, "Polynomial regression: An introduction." Built In, Available at https://builtin.com/machine-learning/polynomial-regression, 2023. (Last accessed in 09/03/2024). [Citado nas páginas vii e 11]

[13] "What is logistic regression?." IBM, Available at https://www.ibm.com/topics/logistic-regression. (Last accessed in 09/03/2024). [Citado na página 12]

[14] K. Kai, "The math behind logistic regression." Medium, Available at https://medium.com/analytics-vidhya/the-math-behind-logistic-regression-c2f04ca27bca, 2020. (Last accessed in 11/03/2024). [Citado nas páginas vii e 12]

[15] V. Kanade, "What is logistic regression? equation, assumptions, types, and best practices." Spiceworks, Available at https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/, 2022. (Last accessed in 11/03/2024). [Citado na página 13]

[16] "What is a decision tree?." IBM, Available at https://www.ibm.com/topics/decision-trees. (Last accessed in 11/03/2024). [Citado nas páginas vii e 13]

[17] "Regression trees." IBM, Available at https://www.ibm.com/docs/en/db2-warehouse?topic=procedures-regression-trees. (Last accessed in 11/03/2024). [Citado na página 13]

[18] "O que é e como funciona o algoritmo randomforest." Didática Tech, Available at https://didatica.tech/o-que-e-e-como-funciona-o-algoritmo-randomforest/. (Last accessed in 11/03/2024). [Citado na página 14]

[19] L. Breiman, "Rndom forests," *Machine Learning*, vol. 45, pp. 5 – 32, 2001. [Citado na página 14]

[20] A. Cutler, D. R. Cutler, and J. R. Stevens, *Random Forests*, pp. 157–175. New York, NY: Springer New York, 2012. (Last accessed in 24/05/2024). [Citado na página 14]

[21] "What is random forest?." IBM, Available at `https://www.ibm.com/topics/random-forest`. (Last accessed in 11/03/2024). [Citado nas páginas vii e 14]

[22] "What is deep learning?." IBM, Available at `https://www.ibm.com/topics/deep-learning`. (Last accessed in 12/03/2024). [Citado na página 15]

[23] L. Hardesty, "Explained: Neural networks." MIT News Office, Available at `https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414`, 2017. (Last accessed in 12/03/2024). [Citado na página 15]

[24] Deeptimann Info System Software Pvt Ltd, "The main areas where neural network is widely used." LinkedIn, Available at `https://www.linkedin.com/pulse/main-areas-where-neural-network-widely/`, 2023. (Last accessed in 12/03/2024). [Citado nas páginas vii e 16]

[25] O. C. Akgun and J. Mei, "An energy efficient time-mode digit classification neural network implementation," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 378, no. 2164, p. 20190163, 2020. (Last accessed in 12/03/2024). [Citado nas páginas vii e 17]

[26] K. S. Ganesh, "What's the role of weights and bias in a neural network?." Medium, Available at `https://towardsdatascience.com/whats-the-role-of-weights-and-bias-in-a-neural-network-4cf7e9888a0f`, 2020. (Last accessed in 13/03/2024). [Citado nas páginas vii e 18]

[27] R. Barbosa, "Fundamentos das redes neuronais." Moodle Isep, Available at `https://moodle.isep.ipp.pt/pluginfile.php/307341/mod_resource/content/6/SISCA2023_24_Fundamentos_Redes_Neuronais.pdf`, 2023. (Last accessed in 13/03/2024). [Citado nas páginas vii, 18 e 19]

[28] Gautham S., "An introduction to mathematics behind neural networks." Medium, Available at `https://medium.com/analytics-vidhya/an-introduction-to-mathematics-behind-neural-networks-135df0b85fa1`, 2020. (Last accessed in 13/03/2024). [Citado na página 18]

[29] P. Baheti, "Neural networks [12 types & use cases]." V7, Available at `https://www.v7labs.com/blog/neural-networks-activation-functions`, 2021. (Last accessed in 13/03/2024). [Citado na página 18]

[30] J. Brownlee, "A gentle introduction to the rectified linear unit (relu)." Machine Learning Mastery, Available at `https://machinelearningmastery.com/`

rectified-linear-activation-function-for-deep-learning-neural-networks/, 2020. (Last accessed in 13/03/2024). [Citado na página 19]

[31] T. Esteves, "A desvantagem da função relu." Medium, Available at https://estevestoni.medium.com/a-desvantagem-de-utilizar-relu-4478589ef834, 2022. (Last accessed in 13/03/2024). [Citado na página 19]

[32] V. Yathish, "Loss functions and their use in neural networks." Medium, Available at https://towardsdatascience.com/loss-functions-and-their-use-in-neural-networks-a470e703f1e9, 2022. (Last accessed in 13/03/2024). [Citado nas páginas 20 e 21]

[33] M. N. Ashtiani and B. Raahemi, "News-based intelligent prediction of financial markets using text mining and machine learning: A systematic literature review," *Expert Systems with Applications*, vol. 217, p. 119509, 2023. (Last accessed in 14/03/2024). [Citado nas páginas vii e 22]

[34] Z. Hu, Y. Zhao, and M. Khushi, "A survey of forex and stock price prediction using deep learning," *Applied System Innovation*, vol. 4, no. 1, 2021. (Last accessed in 14/03/2024). [Citado nas páginas vii, 22 e 23]

[35] M. M. Kumbure, C. Lohrmann, P. Luukka, and J. Porras, "Machine learning techniques and data for stock market forecasting: A literature review," *Expert Systems with Applications*, vol. 197, p. 116659, 2022. (Last accessed in 14/03/2024). [Citado nas páginas vii, 23, 24 e 25]

[36] A. Hayes, "Treasury bills (t-bills): What they are and what you need to know to invest." Investopedia, Available at https://www.investopedia.com/terms/t/treasurybill.asp, 2024. (Last accessed in 20/03/2024). [Citado na página 25]

[37] A. Q. Md, S. Kapoor, C. J. A.V., A. K. Sivaraman, K. F. Tee, S. H., and J. N., "Novel optimization approach for stock price forecasting using multi-layered sequential lstm," *Applied Soft Computing*, vol. 134, p. 109830, 2023. [Citado na página 25]

[38] "What are recurrent neural networks?." IBM, Available at https://www.ibm.com/topics/recurrent-neural-networks. (Last accessed in 22/03/2024). [Citado nas páginas vii e 28]

[39] C. Olah, "Understanding lstm networks." colah's blog, Available at https://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015. (Last accessed in 07/04/2024). [Citado nas páginas vii, 29, 31, 32, 33 e 34]

[40] A. Amidi and S. Amidi, "Recurrent neural networks cheatsheet." Stanford, Available at https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-

recurrent-neural-networks. (Last accessed in 07/04/2024). [Citado nas páginas vii e 30]

[41] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, pp. 1735–1780, 11 1997. [Citado na página 30]

[42] S. Saxena, "What is lstm? introduction to long short-term memory." Analytics Vidhya, Available at https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/, 2024. (Last accessed in 08/04/2024). [Citado na página 31]

[43] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2014. (Last accessed in 19/04/2024). [Citado na página 34]

[44] "Gated recurrent unit networks." GeeksForGeeks, Available at https://www.geeksforgeeks.org/gated-recurrent-unit-networks/, 2023. (Last accessed in 19/04/2024). [Citado na página 34]

[45] "Fully connected layers in convolutional neural networks." IndianTechWarrior, Available at https://indiantechwarrior.com/fully-connected-layers-in-convolutional-neural-networks/. (Last accessed in 24/05/2024). [Citado nas páginas vii e 35]

[46] D. Unzueta, "Fully connected layer vs. convolutional layer: Explained." Built In, Available at https://builtin.com/machine-learning/fully-connected-layer, 2022. (Last accessed in 29/04/2024). [Citado nas páginas viii e 37]

[47] J. H. Friedman, "Greedy function approximation: A gradient boosting machine.," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189 – 1232, 2001. [Citado na página 38]

[48] "Xgboost." H2O.ai, Available at https://h2o.ai/wiki/xgboost/. (Last accessed in 30/04/2024). [Citado na página 38]

[49] "Xgboost." Nvidia, Available at https://www.nvidia.com/en-us/glossary/xgboost/. (Last accessed in 30/04/2024). [Citado na página 38]

[50] "Introduction to boosted trees." XGBoost, Available at https://xgboost.readthedocs.io/en/stable/tutorials/model.html. (Last accessed in 17/05/2024). [Citado nas páginas viii, 39 e 41]

[51] Y. Wang, Z. Pan, J. Zheng, L. Qian, and L. Mingtao, "A hybrid ensemble method for pulsar candidate classification," *Astrophysics and Space Science*, vol. 364, 08 2019. (Last accessed in 24/05/2024). [Citado nas páginas viii e 40]

[52] "Yahoo finance - stock market live, quotes, business & finance news." Yahoo! Finance, Available at `https://finance.yahoo.com/`, 2024. (Last accessed in 20/03/2024). [Citado na página 45]

[53] "What is the adjusted close?." Yahoo!, Available at `https://help.yahoo.com/kb/SLN28256.html`. (Last accessed in 20/03/2024). [Citado na página 46]

[54] Federal Reserve Bank of St. Louis, "Federal reserve economic data (fred)." `https://fred.stlouisfed.org/`, 2024. (Last accessed in 24/05/2024). [Citado na página 46]

[55] Board of Governors of the Federal Reserve System (US), "3-month treasury bill secondary market rate, discount basis." `https://fred.stlouisfed.org/series/TB3MS`, May 2024. Retrieved from FRED, Federal Reserve Bank of St. Louis. [Citado nas páginas viii e 46]

[56] "Feature." HOPSWORKS, Available at `https://www.hopsworks.ai/dictionary/feature`. (Last accessed in 22/05/2024). [Citado na página 47]

[57] E. Hoseinzade and S. Haratizadeh, "Cnnpred: Cnn-based stock market prediction using several data sources." Expert Systems with Applications, Available at `https://www.researchgate.net/publication/331911968_CNNpred_CNN-based_stock_market_prediction_using_a_diverse_set_of_variables`, 10 2018. (Last accessed in 03/06/2024). [Citado na página 47]

[58] K. D, "Optimizing performance: Selectkbest for efficient feature selection in machine learning." Medium, Available at `https://medium.com/@Kavya2099/optimizing-performance-selectkbest-for-efficient-feature-selection-in-machine-learning-3b635905ed48`, 2023. (Last accessed on 04/06/2024). [Citado na página 48]

[59] A. Hayes, "Simple moving average (sma): What it is and the formula." Investopedia, Available at `https://www.investopedia.com/terms/s/sma.asp`, 2023. (Last accessed in 22/05/2024). [Citado nas páginas viii, 51 e 52]

[60] J. Maverick, "Most commonly-used periods in creating moving average (ma) lines." Investopedia, Available at `https://www.investopedia.com/ask/answers/122414/what-are-most-common-periods-used-creating-moving-average-ma-lines.asp`, 2023. (Last accessed in 25/05/2024). [Citado na página 52]

[61] "Exponential moving average (ema)." Fidelity, Available at `https://www.fidelity.com/learning-center/trading-investing/technical-analysis/technical-indicator-guide/ema`. (Last accessed in 25/05/2024). [Citado nas páginas viii e 53]

[62] J. Chen, "What does the nasdaq composite index measure?." Investopedia, Available at `https://www.investopedia.com/terms/n/nasdaqcompositeindex.asp`, 2023. (Last accessed in 05/06/2024). [Citado nas páginas viii, 53 e 54]

[63] "Nasdaq 100 companies." Slickcharts, Available at `https://www.slickcharts.com/nasdaq100`, 2024. (Last accessed in 05/06/2024). [Citado nas páginas ix e 55]

[64] "S&p 500 100 companies." Slickcharts, Available at `https://www.slickcharts.com/sp500`, 2024. (Last accessed in 05/06/2024). [Citado nas páginas ix e 55]

[65] T. Vipond, "Dow jones industrial average (djia)." Corporate Finance Institute, Available at `https://corporatefinanceinstitute.com/resources/equities/dow-jones-industrial-average-djia/`. (Last accessed in 05/06/2024). [Citado na página 56]

[66] "Dow jones 100 companies." Slickcharts, Available at `https://www.slickcharts.com/dowjones`, 2024. (Last accessed in 05/06/2024). [Citado nas páginas ix e 56]

[67] "Jupyter notebook." databriks, Available at `https://www.databricks.com/glossary/jupyter-notebook`. (Last accessed in 10/06/2024). [Citado na página 57]

[68] "yfinance 0.2.40." PyPI, Available at `https://pypi.org/project/yfinance/`. (Last accessed in 11/06/2024). [Citado na página 57]

[69] "pandas." pandas, Available at `https://pandas.pydata.org/`. (Last accessed in 11/06/2024). [Citado na página 58]

[70] "pandas-ta 0.3.14b." PyPI, Available at `https://pypi.org/project/pandas-ta/`. (Last accessed in 11/06/2024). [Citado na página 58]

[71] "Numpy." NumPy, Available at `https://numpy.org/`. (Last accessed in 11/06/2024). [Citado na página 58]

[72] "sklearn.preprocessing." scikit-learn, Available at `https://scikit-learn.org/stable/api/sklearn.preprocessing.html`. (Last accessed in 11/06/2024). [Citado na página 58]

[73] "Keras." Keras, Available at `https://scikit-learn.org/stable/api/sklearn.preprocessing.html`. (Last accessed in 11/06/2024). [Citado na página 58]

[74] "Keras tuner." KerasTuner, Available at `https://keras.io/keras_tuner/`. (Last accessed in 11/06/2024). [Citado na página 58]

[75] "Matplotlib: Visualization with python." Matplotlib, Available at `https://matplotlib.org/`. (Last accessed in 11/06/2024). [Citado na página 58]

[76] P. Banerjee, "A guide on xgboost hyperparameters tuning." kaggle, Available at `https://www.kaggle.com/code/prashant111/a-guide-on-xgboost-hyperparameters-tuning/`, 2020. (Last accessed in 02/08/2024). [Citado na página 75]

[77] "Time series cross-validation." GeeksForGeeks, Available at `https://www.geeksforgeeks.org/time-series-cross-validation/`, 2024. (Last accessed in 06/08/2024). [Citado na página 77]

Anexo A

Tabelas das Métricas

A.1 Tabelas - *Input Window*

Tabela A.1: *Input Window* Modelo GRU - Melhor Valor

Input Window	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
25	5.02846	49.65435	7.04658	3.89%	0.98225
50	6.31327	76.09781	8.72341	4.66%	0.97272
75	5.08089	50.13035	7.08028	3.92%	0.98198
100	4.94083	47.41820	6.88609	3.80%	0.98286
125	4.88618	46.96854	6.85336	3.75%	0.98307
150	5.14778	51.18505	7.15437	3.95%	0.98145

Tabela A.2: *Input Window* Modelo GRU - Pior Valor

Input Window	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
25	9.10637	152.69608	12.35703	6.45%	0.94540
50	9.90992	179.45965	13.39625	6.96%	0.93566
75	9.45073	163.51755	12.78740	6.63%	0.94122
100	7.33494	101.27654	10.06362	5.29%	0.96339
125	7.83548	112.76767	10.61921	5.68%	0.95935
150	8.76174	139.21276	11.79885	6.23%	0.94954

Tabela A.3: *Input Window* Modelo GRU - Desvio Padrão

Input Window	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
25	1.18177	29.46482	1.53403	0.75%	0.01054
50	0.99763	28.83715	1.30479	0.63%	0.01034
75	1.26314	32.95962	1.65151	0.79%	0.01185
100	0.78911	17.21170	1.01737	0.51%	0.00622
125	1.00350	22.48994	1.27894	0.64%	0.00811
150	1.27101	30.61886	1.61489	0.82%	0.01110

Tabela A.4: *Input Window* Modelo LSTM - Melhor Valor

Input Window	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
25	3.94977	29.69416	5.44923	3.37%	0.98938
50	3.56138	23.04814	4.80085	3.09%	0.99174
75	3.97907	30.00899	5.47805	3.39%	0.98921
100	3.77981	26.26533	5.12497	3.26%	0.99053
125	3.81703	27.54905	5.24872	3.29%	0.99004
150	3.98095	29.04822	5.38964	3.37%	0.98947

Tabela A.5: *Input Window* Modelo LSTM - Pior Valor

Input Window	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
25	7.08467	102.80521	10.13929	5.22%	0.96324
50	6.01871	78.30179	8.84883	4.63%	0.97193
75	7.18433	110.39494	10.50690	5.30%	0.96032
100	5.37531	62.10715	7.88081	4.24%	0.97761
125	8.17676	144.98672	12.04104	5.94%	0.94759
150	7.49130	115.87539	10.76454	5.48%	0.95800

Tabela A.6: *Input Window* Modelo LSTM - Desvio Padrão

Input Window	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
25	0.90390	21.69876	1.39062	0.53%	0.00776
50	0.77884	16.80474	1.26272	0.49%	0.00602
75	1.02811	25.34774	1.59766	0.62%	0.00911
100	0.50440	10.94269	0.84397	0.30%	0.00394
125	1.54388	42.20807	2.40891	0.93%	0.01526
150	1.18649	29.60770	1.84068	0.72%	0.01073

A.2 Tabelas - Modelos

Tabela A.7: Modelos LSTM - Melhor Valor

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
2 LSTM + 2 dropout	3.77981	26.26533	5.12497	3.26%	0.99053
3 LSTM + 3 dropout	4.52288	39.47497	6.28291	3.80%	0.98577
4 LSTM + 4 dropout	9.84775	176.02470	13.26743	7.33%	0.93655
5 LSTM + 5 dropout	9.67074	170.59401	13.06116	7.28%	0.93850

Tabela A.8: Modelos LSTM - Pior Valor

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
2 LSTM + 2 dropout	5.37531	62.10715	7.88081	4.24%	0.97761
3 LSTM + 3 dropout	15.17114	407.53920	20.18760	10.55%	0.85309
4 LSTM + 4 dropout	20.29869	698.06299	26.42088	14.11%	0.74836
5 LSTM + 5 dropout	16.26053	454.22995	21.31267	11.50%	0.83626

Tabela A.9: Modelos LSTM - Desvio Padrão

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
2 LSTM + 2 dropout	0.50440	10.94269	0.84397	0.30%	0.00394
3 LSTM + 3 dropout	3.06332	106.59843	3.99147	1.93%	0.03843
4 LSTM + 4 dropout	3.17343	153.92141	3.92100	2.08%	0.05549
5 LSTM + 5 dropout	2.38199	98.18865	2.94859	1.54%	0.03540

Tabela A.10: Modelos GRU - Melhor Valor

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
2 GRU + 2 dropout	4.94083	47.41820	6.88609	3.80%	0.98286
3 GRU + 3 dropout	4.94785	47.99619	6.92793	3.90%	0.98270
4 GRU + 4 dropout	5.11614	51.41216	7.17023	4.05%	0.98147

Tabela A.11: Modelos GRU - Pior Valor

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
2 GRU + 2 dropout	7.33494	101.27654	10.06362	5.29%	0.96339
3 GRU + 3 dropout	8.69291	138.67896	11.77620	6.25%	0.95001
4 GRU + 4 dropout	9.52870	167.46254	12.94073	6.83%	0.93963

Tabela A.12: Modelos GRU - Desvio Padrão

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
2 GRU + 2 dropout	0.78911	17.21170	1.01737	0.51%	0.00622
3 GRU + 3 dropout	1.01174	24.20344	1.29716	0.65%	0.00872
4 GRU + 4 dropout	1.43045	38.10694	1.85082	0.90%	0.01374

Tabela A.13: Modelos LSTM + GRU - Melhor Valor

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
2 LSTM + 1 GRU	4.95480	49.20979	7.01497	3.91%	0.98226
2 LSTM + 2 GRU	4.24223	32.67327	5.71605	3.65%	0.98822
2 GRU + 1 LSTM	3.74880	25.69049	5.06858	3.23%	0.99074

Tabela A.14: Modelos LSTM + GRU - Pior Valor

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
2 LSTM + 1 GRU	8.96369	154.38134	12.42503	6.50%	0.94435
2 LSTM + 2 GRU	9.69856	171.25867	13.08658	7.07%	0.93826
2 GRU + 1 LSTM	6.31072	82.67172	9.09240	4.80%	0.97020

Tabela A.15: Modelos LSTM + GRU - Desvio Padrão

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
2 LSTM + 1 GRU	1.37283	35.86149	1.82905	0.88%	0.01293
2 LSTM + 2 GRU	1.68813	42.91929	2.34715	1.05%	0.01547
2 GRU + 1 LSTM	0.79962	18.23453	1.29398	0.48%	0.00657

Tabela A.16: Modelos CNN - Melhor Valor

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
2 CNN	10.63822	191.14259	13.82543	7.84%	0.93110
3 CNN	14.90957	345.86901	18.59755	10.83%	0.87532
4 CNN	15.30557	365.81221	19.12622	11.07%	0.86813
2 CNN + 1 GRU	11.54857	211.72904	14.55091	8.40%	0.92367
2 CNN + 2 GRU	10.81101	193.25316	13.90155	7.75%	0.93034
2 CNN + 1 LSTM	9.53292	148.80694	12.19865	6.95%	0.94636
2 CNN + 2 LSTM	8.35199	118.31870	10.87744	6.13%	0.95735
3 CNN + 1 GRU	8.47159	117.94863	10.86042	6.27%	0.95748
3 CNN + 2 GRU	11.89722	227.91313	15.09679	8.55%	0.91784
3 CNN + 1 LSTM	13.92447	308.40158	17.56137	9.95%	0.88883
3 CNN + 2 LSTM	7.97002	107.50603	10.36851	5.96%	0.96125
3 CNN + 3 LSTM	15.78657	412.47856	20.30957	11.06%	0.85131
2 LSTM + 2 CNN	17.41048	471.94348	21.72426	12.74%	0.82987
2 LSTM + 3 CNN	14.79069	333.43473	18.26020	10.87%	0.87980
1 LSTM + 2 CNN	12.76768	257.40614	16.04388	9.47%	0.90721
1 LSTM + 3 CNN	14.21209	312.29963	17.67200	10.56%	0.88742
2 GRU + 2 CNN	17.15017	457.07881	21.37940	12.48%	0.83523
2 GRU + 3 CNN	14.44510	316.29845	17.78478	10.80%	0.88598
1 GRU + 2 CNN	14.14756	299.49161	17.30583	10.80%	0.89204
1 GRU + 3 CNN	12.59941	243.74608	15.61237	9.40%	0.91213

Tabela A.17: Modelos CNN - Pior Valor

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
2 CNN	27.27576	1198.78212	34.62343	18.97%	0.56786
3 CNN	22.38688	796.31091	28.21898	15.71%	0.71294
4 CNN	25.76654	1045.88786	32.34019	18.05%	0.62297
2 CNN + 1 GRU	17.28529	463.48923	21.52880	12.41%	0.83292
2 CNN + 2 GRU	17.10678	456.86963	21.37451	12.22%	0.83531
2 CNN + 1 LSTM	18.73431	545.75812	23.36147	13.38%	0.80326
2 CNN + 2 LSTM	19.20578	578.79518	24.05816	13.70%	0.79135
3 CNN + 1 GRU	17.56592	488.53538	22.10284	12.45%	0.82389
3 CNN + 2 GRU	20.35504	646.84434	25.43313	14.45%	0.76682
3 CNN + 1 LSTM	19.16631	586.16680	24.21088	13.45%	0.78870
3 CNN + 2 LSTM	18.38609	548.50028	23.42008	12.84%	0.80227
3 CNN + 3 LSTM	20.45636	698.19176	26.42332	14.11%	0.74831
2 LSTM + 2 CNN	27.73474	1216.52898	34.87878	19.93%	0.56146
2 LSTM + 3 CNN	20.43291	653.62900	25.56617	14.67%	0.76438
1 LSTM + 2 CNN	22.71823	805.32007	28.37816	16.49%	0.90969
1 LSTM + 3 CNN	20.27802	631.78056	25.13525	14.78%	0.88742
2 GRU + 2 CNN	27.74469	1207.57043	34.75011	19.66%	0.56469
2 GRU + 3 CNN	20.61018	655.76579	25.60792	14.99%	0.76361
1 GRU + 2 CNN	26.34271	1084.05778	32.92503	18.70%	0.60921
1 GRU + 3 CNN	19.63353	602.57503	24.54740	14.02%	0.78278

Tabela A.18: Modelos CNN - Desvio Padrão

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
2 CNN	4.82493	290.46558	6.05007	3.22%	0.10471
3 CNN	2.27777	137.76237	2.88906	1.53%	0.04966
4 CNN	3.21978	204.59287	4.05110	2.16%	0.07375
2 CNN + 1 GRU	1.98897	91.26006	2.48328	1.35%	0.03290
2 CNN + 2 GRU	1.99471	85.65900	2.41825	1.40%	0.03088
2 CNN + 1 LSTM	2.60779	115.79746	3.20137	1.80%	0.04174
2 CNN + 2 LSTM	3.50912	148.32061	4.28966	2.41%	0.05347
3 CNN + 1 GRU	2.72192	110.44057	3.37351	1.85%	0.03981
3 CNN + 2 GRU	2.54935	125.64395	3.10642	1.79%	0.04529
3 CNN + 1 LSTM	1.60133	83.81926	2.01013	1.09%	0.03022
3 CNN + 2 LSTM	2.76619	116.11741	3.44105	1.85%	0.04186
3 CNN + 3 LSTM	1.59064	98.39508	2.07777	1.04%	0.03547
2 LSTM + 2 CNN	2.91709	210.72225	3.77080	1.98%	0.07596
2 LSTM + 3 CNN	1.76796	95.59913	2.20864	1.21%	0.03446
1 LSTM + 2 CNN	2.73945	152.98834	3.40278	1.91%	0.05515
1 LSTM + 3 CNN	1.75288	91.68484	2.14729	1.22%	0.03305
2 GRU + 2 CNN	3.35128	237.32096	4.26081	2.28%	0.08555
2 GRU + 3 CNN	2.03566	114.00822	2.63150	1.36%	0.04110
1 GRU + 2 CNN	2.97473	193.03760	3.78559	1.96%	0.06959
1 GRU + 3 CNN	1.92540	97.27286	2.44080	1.28%	0.03507

Tabela A.19: Modelos RNN - Melhor Valor

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
2 RNN	10.69827	193.42192	13.90762	7.62%	0.93027
3 RNN	8.07330	106.22527	10.30656	6.08%	0.96171
4 RNN	27.98883	1273.36479	35.68424	19.60%	0.54097
2 RNN + 1 GRU	10.85959	207.46581	14.40367	7.68%	0.92521
2 RNN + 1 LSTM	18.99778	664.94172	25.78646	12.84%	0.76030
3 RNN + 1 GRU	37.27944	2386.47301	48.85154	25.42%	0.13971
3 RNN + 2 GRU	50.61417	4276.27961	65.39327	34.62%	-0.54154
3 RNN + 1 LSTM	25.36567	1304.93685	36.12391	20.48%	0.52959
3 RNN + 2 LSTM	17.17409	610.23537	24.70294	11.88%	0.78002
1 GRU + 2 RNN	3.94773	28.65353	5.35290	3.22%	0.98967
2 GRU + 2 RNN	4.79734	42.76413	6.53943	3.80%	0.98458
1 LSTM + 2 RNN	3.18217	18.79280	4.33507	2.80%	0.99323
2 LSTM + 2 RNN	13.00344	260.49538	16.13987	9.48%	0.90610

Tabela A.20: Modelos RNN - Pior Valor

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
2 RNN	18.45188	571.05893	23.89684	12.79%	0.79414
3 RNN	14.56069	349.62778	18.69834	10.33%	0.87396
4 RNN	80.24647	9853.89143	99.26677	57.02%	-2.55219
2 RNN + 1 GRU	22.17816	874.21257	29.56709	15.00%	0.68486
2 RNN + 1 LSTM	49.34018	4080.12043	63.87582	33.67%	-0.47082
3 RNN + 1 GRU	64.35859	6578.96312	81.11081	44.68%	-1.37162
3 RNN + 2 GRU	62.70321	6306.82184	79.41550	43.22%	-1.27352
3 RNN + 1 LSTM	80.26455	9588.08764	97.91878	58.18%	-2.45637
3 RNN + 2 LSTM	63.34151	6272.90795	79.20169	44.38%	-1.26129
1 GRU + 2 RNN	75.82449	8514.55580	92.27435	55.37%	-2.06937
2 GRU + 2 RNN	9.26760	149.81225	12.23978	6.71%	0.94599
1 LSTM + 2 RNN	8.37375	145.96470	12.08159	5.93%	0.94738
2 LSTM + 2 RNN	18.63531	545.58216	23.35770	13.25%	0.80333

Tabela A.21: Modelos RNN - Desvio Padrão

Modelo	**MAE**	**MSE**	**RMSE**	**MAPE**	**R2**
2 RNN	1.96071	97.18043	2.53484	1.31%	0.03503
3 RNN	1.75822	65.30800	2.23308	1.20%	0.02354
4 RNN	23.14251	3713.73493	27.57625	17.09%	1.33875
2 RNN + 1 GRU	3.43039	204.76479	4.56465	2.22%	0.07381
2 RNN + 1 LSTM	10.01399	1124.29861	12.42411	6.88%	0.40529
3 RNN + 1 GRU	9.02763	1381.66906	10.70994	6.46%	0.49807
3 RNN + 2 GRU	3.19223	532.18972	3.67981	2.28%	0.19185
3 RNN + 1 LSTM	13.41004	2100.21015	15.33205	9.36%	0.75710
3 RNN + 2 LSTM	13.60089	1755.90268	16.26158	9.46%	0.63298
1 GRU + 2 RNN	31.11441	3842.88380	38.04022	22.18%	1.38530
2 GRU + 2 RNN	1.29137	30.58936	1.63850	0.84%	0.01103
1 LSTM + 2 RNN	1.73070	39.73439	2.47541	1.06%	0.01432
2 LSTM + 2 RNN	1.75872	86.43862	2.20716	1.20%	0.03116