DISSERTAÇÃO DE MESTRADO

Framework de apoio à tomada de decisão no mercado de ações baseado em aprendizado por reforço profundo

Iure Vieira Brandão

Brasília, Fevereiro de 2021

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

DISSERTAÇÃO DE MESTRADO

Framework de apoio à tomada de decisão no mercado de ações baseado em aprendizado por reforço profundo

Iure Vieira Brandão

Dissertação de Mestrado submetida ao Departamento de Engenharia Mecânica como requisito parcial para obtenção do grau de Mestre em Sistemas Mecatrônicos

Banca Examinadora

João Paulo C. L. da Costa, Prof. Dr.-Ing., ENE/UnB, Hochschule Hamm-Lippstadt
Orientador

Rafael T. de Sousa Jr., Prof. Dr., ENE/UnB
Examinador interno

José Alfredo Ruiz Vargas, Prof. Dr., ENE/UnB
Examinador interno

FICHA CATALOGRÁFICA

BRANDÃO, IURE VIEIRA
Framework de apoio à tomada de decisão no mercado de ações baseado em aprendizado por reforço profundo [Distrito Federal] 2021.
xvi, 87 p., 210 x 297 mm (ENM/FT/UnB, Mestre, Engenharia Mecânica, 2021).
Dissertação de Mestrado - Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Mecânica

1. Mercado de ações	2. Aprendizado por reforço
3. Aprendizado profundo	4. Inteligência Artificial
I. ENM/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

BRANDÃO, I. V. (2021). *Framework de apoio à tomada de decisão no mercado de ações baseado em aprendizado por reforço profundo*. Dissertação de Mestrado, Departamento de Engenharia Mecânica, Universidade de Brasília, Brasília, DF, 87 p.

CESSÃO DE DIREITOS

Iure Vieira Brandão
Depto. de Engenharia Mecânica (ENM) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

Agradecimentos

Primeiramente, agradeço a Deus por sempre me abençoar, me guiar e me mostrar os passos que precisam ser seguidos para completar cada etapa da minha vida. Em segundo lugar, agradeço imensamente a toda minha família, em especial, aos meus pais e meu irmão por serem a base da minha vida pois, sem eles, eu não chegaria aonde estou. Agradeço também ao Prof. Dr.-Ing. João Paulo C. L. da Costa pela oportunidade da realização deste trabalho e, principalmente, pelo apoio e confiança prestados em mim, sempre me motivando a dar o meu melhor; ao Bruno Justino por sempre me dar suporte e ajuda quando precisei. Agradeço especialmente a Camila França por todo o amor, carinho, confiança, paciência e apoio durante todo esse tempo, no qual sempre me ensinou a perserverar e persistir até o fim. Aos meus grandes amigos Fernanda, Gabriel Junqueira, Gabriel Pinheiro, Ismael, Iago, Luiza e Yuri, que independentemente do momento, sempre estenderam as suas mãos para me ajudar e apoiar. Agradeço o apoio técnico e computacional do Laboratório de Tecnologias para Tomada de Decisão - LATITUDE, da Universidade de Brasília, que conta com apoio do CNPq - Conselho Nacional de Pesquisa (Outorgas 312180/2019-5 PQ-2, BRICS2017-591 LargEWiN e 465741/2014-2 INCT em Cibersegurança), da CAPES - Coordenação de Aperfeiçoamento do Pessoal de Nível Superior (Outorgas PROAP PPGEE/UnB, 23038.007604/2014-69 FORTE, e 88887.144009/2017-00 PROBRAL), bem como de projetos com recursos do Ministério da Economia (Outorgas 005/2016 DIPLA e 083/2016 ENAP) e da Secretaria de Segurança Institucional da Presidência da República do Brasil (Outorga ABIN 002/2017), projetos estes dos quais fui bolsista.

Iure Vieira Brandão

RESUMO

No mercado de ações, investidores adotam diferentes estratégias para identificar uma sequência de decisões de investimento a fim de maximizar seus lucros. Para apoiar a decisão dos investidores, uma framework de aprendizado de máquina (*machine learning*) foi proposta. Em particular, as abordagens de aprendizado profundo (*deep learning*) são muito atraentes, uma vez que o mercado de ações apresenta um comportamento altamente não linear e as técnicas de aprendizado profundo podem rastrear variações de curto e longo prazo. Em contraste com as técnicas de aprendizado supervisionadas, o aprendizado por reforço profundo reúne benefícios de aprendizado profundo e adiciona adaptação e melhoria em tempo real do modelo de aprendizado de máquina. Neste trabalho, propomos uma framework de suporte à decisão para o mercado de ações baseado no aprendizado por reforço profundo. Ao aprender as regras de negociação, a framework reconhece padrões, maximiza o lucro obtido e fornece recomendações aos investidores. A framework proposta supera o estado da arte com 86% da métrica F1-Score para operações de compra e 88% da pontuação F1-Score para operações de venda em termos de avaliação da estratégia de posicionamento.

Palavras-chave: Aprendizado por reforço profundo, Q-Learning, Mercado de ações, Inteligência artificial, Processamento de dados

ABSTRACT

In stock markets, investors adopt different strategies to identify a sequence of profitable investment decisions to maximize their profits. To support the decision of investors, machine learning (ML) softwares are being applied. In particular, deep learning (DL) approaches are attractive since the stock market parameter presents a highly non-linear behavior, and since DL techniques can track short time and long time variations. In contrast to supervised ML techniques, deep reinforcement learning (DRL) gathers DL's benefits and adds the real-time adaptation and improvement of the machine learning model. In this paper, we propose a decision support framework for the stock market based on DRL. By learning the trading rules, our framework recognizes patterns, maximizes the profit obtained and provides recommendations to the investors. The proposed DRL framework outperforms the state-of-the-art framework with 86 % of F1 score for buy operations and 88 % of F1 score for sale operations in terms of evaluating the positioning strategy.

Keywords: Deep Reinforcement Learning, Q-Learning, Stock Market, Artificial Intelligence, Features pre-processing

Sumário

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AM	Aprendizado de Máquina, do inglês *Machine Learning*
AD	Chaikin A/D Line (AD).
APO	Oscilador de preço absoluto, do inglês *Absolute Price Oscillator*.
ANN	Redes Neurais Artificiais, do inglês *Artificial Neural Networks*.
ADOSC	Oscilador de Chaikin, do inglês *Chaikin A/D Oscillator*.
BB	Bandas de Bollinger, do inglês *Bollinger Bands*.
BI	Inteligência de Negócio, do inglês *Business Inteligence*.
CDB	Certificado de Depósito Bancário
CDI	Certificado de Depósito Interbancário
DL	Aprendizado profundo, do inglês *Deep Learning*
DRL	Aprendizado por reforço profundo, do inglês *Deep Reinforcement Learning*
DNN	Redes Neurais Profundas, do inglês *Deep Neural Networks*.
EMA	Média móvel exponencial, do inglês *Exponential Moving Average*.
ETL	Extração, Transformação e Carga, do inglês *Extract, Transform and Load*.
EUA	Estados Unidos da América.
FN	Falso negativo, do inglês *False negative*.
FP	Falso positivo, do inglês *False positive*.
GA	Algoritmos Genéticos, do inglês *Genetic Algorithms*.
IA	Inteligência Artificial, do inglês *Artificial Intelligence*
IPCA	Índice de Preços ao Consumidor Amplo
LDA	Linha de Distribuição Acumulada.
MDP	Processo de Decisão de Markov, do inglês *Markov Decision Process*.
MACD	Média Móvel Convergente e Divergente, do inglês *Moving Average Convergence Divergence*.
MACD_{EMA}	Média Móvel Exponencial dos valores de MACD
OAA	Um contra todos, do inglês *One-Against-All*.
OBV	Saldo de Volume, do inglês *On Balance Volume*.
PCA	Análise de Componentes Principais , do inglês *Principal Components Analysis*.
RL	Aprendizado por reforço, do inglês *Reinforcement Learning*
RN	Redes Neurais.
RSI	Índice de força relativa, do inglês *Relative Strength Index*.
SMA	Média móvel aritmética, do inglês *Simple Moving Average*.
SVM	Máquina de Vetor de Suporte, do inglês *Support Vector Machine*.
TD	Diferença Temporal, do inglês *Temporal Difference*.
TN	Verdadeiro negativo, do inglês *True negative*.
TP	Verdadeiro positivo, do inglês *True positive*.

1 INTRODUÇÃO

O mercado de ações, também conhecido como bolsa de valores, é um mercado, onde indivíduos e empresas negociam através da compra e venda de títulos de ações com o objetivo de obter lucro sobre o investimento. Neste mercado, as entidades abrem seu capital e se reúnem para comercializarem os ativos disponíveis a fim de compartilhar os riscos da empresa e, consequentemente, obter investimentos e grandes recompensas financeiras (3). Isso torna este mercado altamente importante para o desenvolvimento de qualquer país, pois estimula o investimento do capital da população nas empresas e, por consequência, na economia (4).

Entretanto, 90 % dos investidores do mercado de ações perdem seu capital investido, mesmo aqueles apoiados por sistemas profissionais (5). As pessoas que não conhecem este mercado geralmente associam a bolsa de valores a mecanismos de perda de dinheiro e não acreditam que é possível construir um patrimônio sólido com investimentos de alto retorno (6). O mercado de ações é um sistema bastante complexo, porém com um vasto e detalhado conhecimento sobre ele, é possível que sejam tomadas decisões que resultem em lucros substancialmente maiores para o investidor quando se comparado com todos os outros tipos de investimento no mercado financeiro (7).

Atualmente, diversos modelos de aprendizado de máquina (AM), que fazem parte da inteligência artificial (IA), estão sendo aperfeiçoados e desenvolvidos com o objetivo de apoiar investidores iniciantes e profissionais a tomar as decisões corretas no mercado de ações (8). Os modelos de AM baseados em aprendizado por reforço profundo, do inglês *Deep Reinforcement Learning* (DRL), demonstram um retorno sobre o investimento significativamente maior quando se comparado as técnicas de análises tradicionais (1). Dessa forma, estes modelos trazem resultados de maior relevância para os investidores e, por consequência, demonstram a sua importância neste tipo mercado financeiro tão complexo.

1.1 MOTIVAÇÃO

O mercado de ações é crucial para a expansão de toda a economia global, uma vez que ele permite que indivíduos obtenham lucros e empresas abram seu capital a fim de receber grandes recompensas, além de espalhar seus riscos com os investidores (3). Logo, a bolsa de valores é vital para o desenvolvimento de uma economia de um país (9) pois, fomenta o reinvestimento do capital da população na economia e, por consequência, proporciona a geração de emprego, investimentos de recursos no país e estimula o crescimento das empresas.

Nesta circunstância, os investidores estão continuamente buscando as melhores estratégias para ajudá-los a tomar decisões de investimento com o objetivo de maximizar seus lucros (1).

A bolsa de valores é o investimento com o maior potencial de retorno do mercado, porém o investimento nela é sempre acompanhada de um alto risco e bastante complexo (5). Portanto, o mercado de ações é altamente complexo e exige um nível de conhecimento muito alto para se investir com o objetivo de obter lucro sobre o investimento.

Recentemente, modelos de AM têm sido aplicados para apoiar investidores e fornecer retorno sobre o investimento mais significativo do que as técnicas de análise tradicionais (10). Os resultados mostram que aplicação de IA no mercado de ações pode melhorar consideravelmente o suporte à decisão do investidor e maximizar os lucros em mais de 28 % (1). Logo, este contexto torna bastante atrativo a aplicação de técnicas de DRL a fim de apoiar tomadas de decisões no mercado de ações.

1.2 DEFINIÇÃO DO PROBLEMA

A bolsa de valores é o ambiente em que as compras e vendas de ações acontecem de fato. Neste tipo de mercado, a negociação é feita por meio de sistemas sofisticados e processos padronizados para assegurar transparência e liquidez ao mercado. Nesta circunstância, os investidores estão continuamente buscando as melhores estratégias para ajudá-los a tomar decisões de investimento com o objetivo de maximizar seus lucros (11).

Entretanto, devido à alta complexidade dos sistemas financeiros, boa parte dos acionistas, mesmo aqueles apoiados por sistemas profissionais, perdem seu capital na bolsa de valores (1). Dessa forma, a negociação no mercado de ações exige um amplo conhecimento do mercado a fim de se obter tomadas de decisões lucrativas (12).

Portanto, o propósito é desenvolver uma *framework* baseada em DRL que aprende regras de negociação no mercado de ações com o objetivo de reconhecer padrões e maximizar o lucro. Ademais, para melhorar a usabilidade dos investidores, uma camada de visualização foi criada, sugerindo qual ação deve ser tomada em determinado intervalo de tempo.

1.3 OBJETIVOS

A partir da motivação apresentada na seção 1.1, o objetivo principal desse trabalho foi sumarizado na seção 1.3.1. Para alcançar este objetivo, foi delimitado um conjunto de objetivos específicos listado na seção 1.3.2.

1.3.1 Objetivo principal

Desenvolver uma *framework*, com as técnicas de DRL no âmbito do mercado de ações a fim de tornar a ferramenta capaz de aprender e reconhecer padrões para apoiar a tomada de decisões

dos investidores e, por fim, obter retornos sobre os investimentos.

1.3.2 Objetivos específicos

- Analisar e desenvolver códigos para extração dos dados relacionados as ações nas bolsas de valores.
- Pré-processar os dados a fim de transformá-los em um formato apropriado para execução dos algoritmos de AM.
- Cálculo dos indicadores técnicos e estratégias de posicionamento a partir de fórmulas matemáticas e algoritmos.
- Seleção de *features* e redução de dimensionalidade para reduzir a complexidade do algoritmo e aumentar sua performance.
- Desenvolver um modelo baseado em DRL com o intuito de torná-lo capaz de aprender e reconhecer padrões de curto e longo prazo na bolsa de valores.
- Estudar e comparar os resultados obtidos sob a perspectiva de avaliação da performance dos algoritmos.
- Criar uma ferramenta de visualização com o objetivo de fornecer apoio aos investidores de forma interativa.

1.4 ARTIGOS PUBLICADOS

Em simultaneidade com o desenvolvimento deste trabalho, foram escritos artigos no âmbito da ciência da computação, especificamente da ciência de dados, no qual foi possível captar conhecimentos de multiplas áreas para realização dessa dissertação. Os seguintes artigos foram publicados em conjunto à busca do título de mestre:

1. **I. V. Brandão**, J. P. C. L. da Costa, G. A. Santos, B. J. G. Praciano, F. C. M. D. Júnior and R. T. de S. Júnior, "Classification and predictive analysis of educational data to improve the quality of distance learning courses" 2019 Workshop on Communication Networks and Power Systems (WCNPS), Brasilia, Brazil, 2019, pp. 1-6, doi: 10.1109/WCNPS.2019.8896312.
2. **I. V. Brandão**, J. P. C. L. da Costa, B. J. G. Praciano, R. T. de Sousa and F. L. L. de Mendonça, "Decision support framework for the stock market using deep reinforcement learning" 2020 Workshop on Communication Networks and Power Systems (WCNPS), Brasilia, Brazil, 2020, pp. 1-6, doi: 10.1109/WCNPS50723.2020.9263712.

1.5 DESCRIÇÃO DOS CAPÍTULOS

Este trabalho é apresentado nos capítulos restantes da seguinte forma:

O capítulo 2 - é designado a explicar os conceito teóricos de bolsa de valores, de aprendizado por reforço, do inglês *Reinforcement Learning* (RL), e aprendizado profundo, do inglês *Deep Learning* (DL). Além disso, é explicado todo o processo de AM, com a descrição de seus algoritmos e as métricas de avaliação deles.

No capítulo 3 - é apresentado a framework proposta com todos os passos bem definidos e explicados, de forma a analisar e entender o mercado de ações, com o objetivo de encontrar o modelo ideal que possa servir de apoio às tomadas de decisões dos investidores visando o lucro.

O capítulo 4 - contém a apresentação dos resultados a partir da aplicação da framework proposta nos dados das ações dos Estados Unidos da América (EUA) e Tailândia. Ademais, é mostrada a camada de visualização para tornar a ferramenta intuitiva para os usuários finais.

Por fim, no capítulo 5 é apresentado as conclusões para os estudos da framework proposta, as implicações dos resultados gerados e os trabalhos a serem realizados no futuro.

2 REFERENCIAL TEÓRICO E ESTADO DA ARTE

Neste capítulo é apresentado o embasamento teórico utilizado para que fosse possível o desenvolvimento e a realização deste trabalho, além das pesquisas, livros e artigos, disponíveis na literatura, relacionados a este tema. Nele, são apresentados os conceitos relacionados a bolsa de valores e ao DRL. Ademais, serão apresentados os algoritmos de AM utilizados no DRL, e por conseguinte, as métricas de avaliação e performance desses algoritmos.

2.1 REFERENCIAL TEÓRICO

2.1.1 Mercado financeiro

O mercado financeiro é um ambiente de compra e venda de valores mobiliários, como ações, opções e títulos, além de câmbio e mercadorias, como ouro e *commodities*. Este mercado é definido pelo contexto em que ocorrem estas operações entre superavitários, aqueles que possuem recursos para investir, e deficitários, aqueles que necessitam dos recursos. O objetivo deste ambiente é conectar quem precisa do capital com quem tem recursos para investir, sejam pessoas físicas ou jurídicas (13).

Os intermediários surgem como os responsáveis por promover o encontro entre estes agentes e constituem a essência dos mercados financeiros pois, sem eles não haveria encontro entre os tomadores e os investidores. Dessa forma, surgem as instituições financeiras, no qual exercem o papel de intermediação entre os negociadores e, também, as entidades reguladoras, onde são criadas normas e regras gerando segurança para este mercado como um todo (14).

O mercado financeiro basicamente oferece dois tipos de investimento: a renda fixa e a renda variável. Cada tipo de investimento atende as necessidades do perfil do investidor, isto é, se o investidor prioriza a segurança, mesmo que isso implique uma remuneração mais baixa, ele deverá escolher a renda fixa. Caso ele busque um retorno elevado sobre o investimento, inclusive se isso trouxer alta probabilidade de perdas, o tipo indicado é a renda variável. Na seção 2.1.1.1, é exposto as características do tipo de investimento em renda fixa e na seção 2.1.1.2 são explicados as peculiaridades da renda variável. Na seção 2.1.1.3, é elucidado de forma detalhada o mercado de ações.

2.1.1.1 Renda fixa

Renda fixa é o tipo de investimento que oferece uma base de projeção ou o cálculo do retorno exato antes da aplicação. Esse tipos de títulos se caracterizam por ter um rendimento prefixado, com um juro anual definido, ou pós-fixado, atrelado a um indicador como o Certificado de

Depósito Interbancário (CDI). Ademais, esse tipo de título também contém a modalidade híbrida, no qual existe um juro fixo mais alguma variável, como o Índice de Preços ao Consumidor Amplo (IPCA).

Existem diversos tipos de aplicações na renda fixa, cujo são caracterizadas por risco, emissor, rentabilidade entre outros aspectos. Essas aplicações podem ser emitidas por instituições financeiras privadas e públicas, como os bancos, empresas ou pelo governo. Alguns dos exemplos dos títulos de renda fixa são (15):

- Tesouro Direto
- Poupança
- Certificado de Depósito Bancário (CDB)
- Letra de Câmbio

Essa modalidade de investimento tem como principal característica a sua previsibilidade, isto é, o investidor sabe o quanto terá de retorno ao final do investimento. Dessa forma, a renda fixa é considerada como um investimento conservador (14). Além disso, pelo fato dos títulos de renda fixa não estarem associados às oscilações do mercado, este tipo de investimento é considerado como seguro e menos arriscado. O que resulta em sua primordial desvantagem em relação a renda variável, que são as baixas rentabilidades dos investimentos (13). A sua principal vantagem é a segurança, no qual a possibilidade de perda de capital é bem menor ou quase nula, quando se comparado com a renda variável (15).

2.1.1.2 Renda variável

A renda variável é o tipo de investimento que contempla ativos financeiros que possuem retornos não previsíveis. Logo, os investidores dessa categoria não conseguem afirmar com veracidade o quanto o terá de rentabilidade ao longo do tempo (13). Ao contrário da renda fixa, onde os investimentos possuem taxa de rentabilidade definida no momento da compra do título. Portanto, a renda variável é considerada todo tipo de investimento que não garante nem um ganho fixo nem a devolução do total que foi aplicado.

Por não ter uma previsibilidade do retorno sobre os investimentos e por apresentar um alto nível de complexidade (7), a sua principal desvantagem é por ser considerada como um investimento arriscado ou inseguro. Segundo (5), 90 % dos investidores perdem seu capital na renda variável, mesmo aqueles apoiados por sistemas profissionais. Entretanto, a sua principal vantagem é justamente por apresentar altos retornos sobre os investimentos, quando se comparado com a renda fixa e quando se tem um amplo e vasto conhecimento sobre o assunto (12). Dessa forma, os investidores estão constantemente buscando as melhores estratégias, na renda variável, para ajudá-los a tomar decisões de investimento com o objetivo final de maximizar seus lucros (1).

Abaixo são explicitados os tipos de renda variável mais comuns (13):

- **Mercado de ações (seção 2.1.1.3)**: É um mercado que oferece papéis (títulos) que representam a menor quantia de uma empresa que decidiu abrir seu capital, isto é, oferecer sociedade aos investidores. Assim, quem adquire uma ação se torna sócio de uma empresa. A ação é negociada na Bolsa de seu país de origem, sendo que seu valor pode subir ou cair de acordo com o interesse dos investidores.
- **Fundos Imobiliários**: Fundos de investimento destinados à aplicação em empreendimentos imobiliários.
- **Fundos de ações**: Fundos de investimento que investem em ações e outros ativos que têm renda variável.
- **Fundos multimercados**: Fundos de investimento que investem em ações e outros ativos que têm renda variável.
- **Câmbio**: Investimentos em moedas, como dólar, euro e libra, são aplicações de renda variável, onde o investidor aposta na variação de uma determinada moeda.
- **Derivativos**: São contratos negociados em Bolsa. O valor desse contrato depende de um outro ativo, que pode ser físico, como uma ação de empresa, o ouro, café, ou financeiro, como algum índice da Bolsa ou uma taxa de juros.

Os fatores que determinam a valorização ou desvalorização de um investimento em renda variável dependem do tipo de aplicação considerado. O preço de uma ação, por exemplo, vai depender do desempenho da empresa e seus indicadores pois, pode haver mais ou menos investidores interessadas na empresa (16). Da mesma forma, se a economia de um país está boa, os ativos dessa economia, como sua moeda e suas empresas, tendem a se valorizar. Por outro lado, uma crise econômica pode derrubar os valores desses ativos.

Além disso, juros baixos reduzem custos das empresas e estimulam o consumo. Por isso, podem favorecer investimentos em renda variável, como ações em bolsa de valores. Ademais, um ativo de renda variável pode se valorizar ou desvalorizar com base na expectativa dos agentes de mercado, que costumam se antecipar a determinado fato antes mesmo que ele venha a acontecer (16).

2.1.1.3 Mercado de ações

O mercado de ações é o mercado organizado onde se negociam ações de sociedades de capital aberto e outros valores mobiliários. Dessa forma, a bolsa de valores é um ambiente de negociação no qual investidores podem comprar ou vender seus títulos emitidos por empresas, sejam elas com capitais públicos, mistos ou privados. Esse processo é intermediado com auxílio de correspondentes de negociações através de instituições financeiras, como as corretoras.

No mercado de ações, existe duas formas de operar (negociar ações): operar comprado e operar vendido, respectivamente, do inglês *long position* e *short position*. Na operação comprada

tem-se o objetivo de comprar uma ação por um preço baixo para ganhar com a sua possível valorização em um momento posterior. Logo, o objetivo é comprar uma ação a um certo preço e vendê-la após o seu valor superar o preço da compra na porcentagem definida ou desejada pelo investidor. Operar vendido é quando o investidor visa ganhar com a baixa do mercado. Dessa forma, a intenção desse tipo de operação é obter lucro com a queda de um ativo (16).

Dentro do mercado de ações, a análise técnica é amplamente utilizada pois, permite aos agentes de mercado prever as tendências dos preços das ações, interpretando os resultados de diferentes formas a fim de encontrar os momentos certos de se operar comprado ou vendido (17). A análise técnica surgiu no século XV através dos especuladores que se voltavam para as oscilações decorrentes da oferta e da demanda do arroz na China. Posteriormente, surgiram estudos no Japão que comprovaram a sua eficiência em produzir ganhos no mercado de renda variável. Como a análise técnica é muito útil em mercado voláteis, é possível interpretar qualquer mercado do mundo, como por exemplo, ações, derivativos, câmbios entre outros (18).

Este tipo de análise é muito utilizado no mercado de ações, principalmente no gráfico de velas, do inglês *candlestick*. Este tipo de gráfico é o mais utilizado por investidores pelo fato de permitir o reconhecimento de padrões visuais de forma que possam ter mais informações ao prever movimentos de preços de produtos financeiros (19). Porém, há dois outros tipos de gráficos utilizados, que são o gráfico de barra e o gráfico de linha. Abaixo são explicados cada tipo de gráfico e suas características:

- **Gráfico de velas**, do inglês ***Candlestick chart***: Representa os preços das ações em forma de barras verticais por um determinado período de tempo. Porém, essas barras verticais podem ter aparências diferentes conforme os preços praticados durante o pregão (18). Conforme pode ser visto na Figura 2.1, a barra marca o espaço entre o preço de abertura e o de fechamento, com o seu corpo. Já a distância entre tais preços e a máxima ou minima, é marcada com um traço fino. Quando se junta cada barra vertical (chamada de *candle*) em uma linha temporal, é formado o gráfico de velas (vide Figura 2.2).

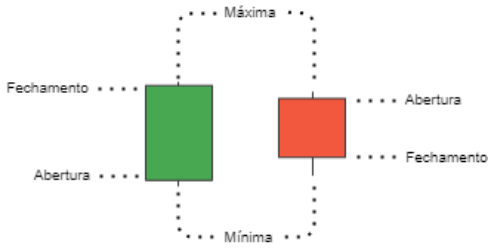

Figura 2.1: Imagem explicativa de *candles*

Existem diversos padrões de *candles* que podem indicar tanto continuação quanto reversão da tendência, como pode ser visto nos 2 exemplos abaixo:

  - **Estrela Cadente**, do inglês ***Shooting Star***: É um padrão de *candle* de uma barra, no qual recebe esse nome porque lembra uma estrela no horizonte com a cauda para cima.

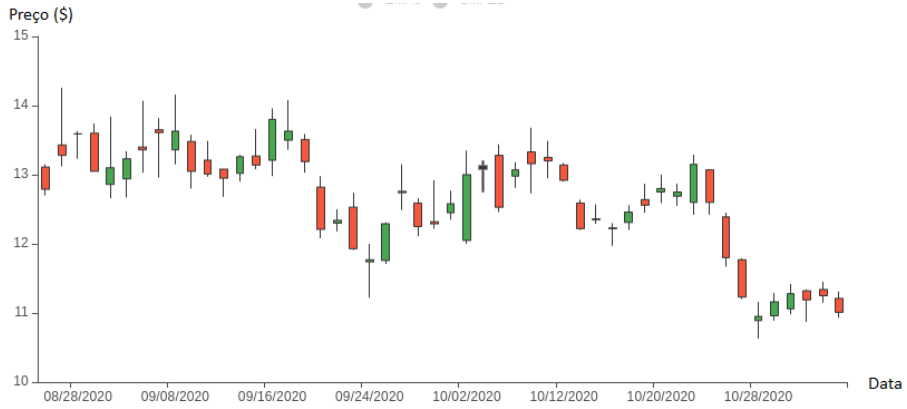

Figura 2.2: Demonstração do gráfico de velas

Ela é formada por um pequeno corpo localizado na parte inferior do candle com uma grande sombra (linha) na parte superior, como pode ser visto na Figura 2.3. Ela é encontrada após um movimento de alta e pode indicar uma reversão do movimento, isto é, que o próximo movimento será de baixa.

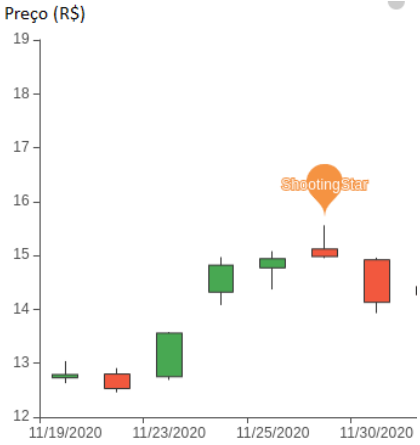

Figura 2.3: Exemplo prático do padrão estrela cadente

– **Doji**: O doji é, também, um padrão de *candle* de uma barra, que representa um momento de indecisão no mercado. Ele é caracterizado por apresentar o preço de fechamento igual ao de abertura. Logo, o seu corpo é representado por uma linha horizontal, como demonstrado na Figura 2.4. Sua importância para uma estrátegia de compra ou de venda é porque ele pode significar uma reversão dos preços das ações, isto é, se uma ação está subindo, após um doji, ela começará a cair.

- **Gráfico de linha**, do inglês ***Line chart***: Este tipo de gráfico consiste na marcação de apenas um ponto de valor para cada período de tempo. Os pontos marcados em cada pregão, por exemplo, são unidos uns aos outros de forma adjcante, formando uma linha que se estende ao longo da escala horizontal (18), como pode ser visualizado na Figura 2.5.
- **Gráfico de barra**, do inglês ***Bar chart***: Este tipo de gráfico é utilizado para representar, na forma de uma barra vertical, os preços praticados no mercado para cada período temporal

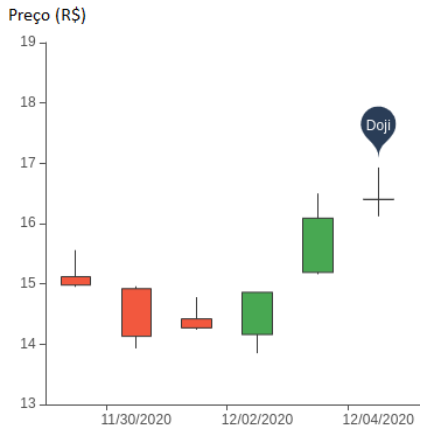

Figura 2.4: Exemplo prático do padrão doji

Figura 2.5: Demonstração do gráfico de linha

escolhido. Cada barra é formado por um traço vertical que indica um período de tempo, no qual a ponta superior significa o preço máximo e ponta inferior o preço mínimo. O traço horizontal esquerdo representa o preço de abertura e o horizontal direito, o preço de fechamento (18). A junção de cada barra ao longo do eixo temporal forma o gráfico de barra (haja vista Figura 2.6).

Dentro da análise técnica, existem os indicadores técnicos que são primordiais para esse tipo de análise. Esses indicadores são cálculos matemáticos baseados no preço histórico que visa prever a direção do mercado financeiro (20). Portanto, eles podem fornecer um sinal de negociação (compra ou venda) aos investidores. E uma estratégia de posicionamento é uma estratégia de entrada que ajuda o investidor a identificar esses sinais, tendências e pontos de entrada. No ramo de indicadores técnicos, existe 4 principais grupos, que são (18):

1. ***Overlap***: Este grupo de indicador técnico pode ser traduzido como grupo de sobreposição, pois com base nos valores de preços anteriores e atuais das ações, é elaborado um elemento gráfico que sobrepõe o gráfico de velas. Dentro deste grupo, os principais indicadores são:
    - **Média móvel aritmética**, do inglês ***Simple Moving Average (SMA)***: A média móvel

Figura 2.6: Demonstração do gráfico de barra

aritmética é um indicador seguidor de tendência, muito utilizado na análise técnica (18). A média móvel é o cálculo do preço médio de um ação sobre um número específico de períodos. A fórmula da SMA é definida em (2.1).

$$\text{SMA} = \frac{1}{n} \sum_{i=t-n}^{t} P(i), \tag{2.1}$$

onde P e n são, respectivamente, o atual preço da ação e o número de períodos. A linha amarela na Figura 2.7, mostra um exemplo da média móvel de 21 períodos.

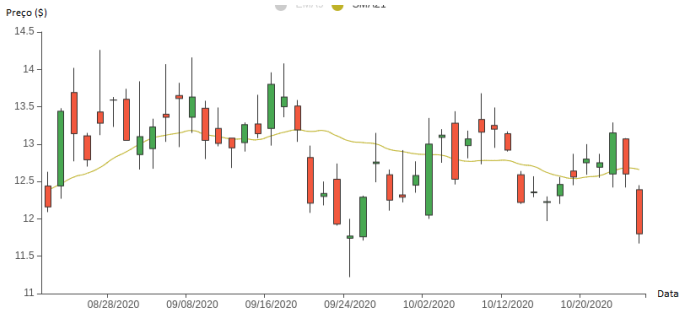

Figura 2.7: SMA de 21 períodos (dias)

- **Média móvel exponencial**, do inglês ***Exponential Moving Average (EMA)***: A média móvel exponencial, assim como a SMA, é um indicador de tendência muito utilizado na análise técnica (18). Este indicador dá um peso maior para as oscilações dos preços mais recentes, diferentemente da SMA que tem um peso igual a todos os períodos. A fórmula da EMA é definida em (2.2) e (2.3).

$$\text{EMA}_t = P_t \cdot \mu + (1 - \mu) \cdot \text{EMA}_{t-1} \tag{2.2}$$

$$\mu = \frac{2}{(n+1)}, \tag{2.3}$$

onde P_t and n são, respectivamente, o atual preço da ação e o número de períodos. Na Figura 2.8, a linha rosa mostra exemplo da EMA de 9 períodos.

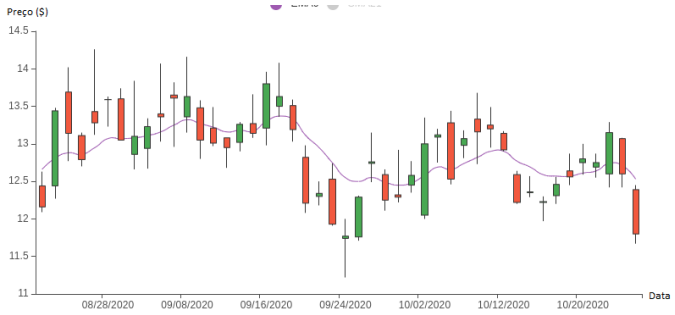

Figura 2.8: EMA de 9 períodos (dias)

- **Bandas de Bollinger**, do inglês ***Bollinger Bands (BB)***: O estudo "Bollinger Bands", criado por John Bollinger, traça linhas (bandas) superior e inferior em torno do preço da ação. A largura das bandas é baseada no desvio padrão dos preços de fechamento de uma média móvel de preço. Além da linha superior e inferior, existe uma linha intermediária (banda do meio) composta por uma SMA. Sua fórmula é definida por (2.4), (2.5) e (2.6).

$$\text{BOLU} = \text{SMA}(\text{TP}, n) + m \cdot \sigma[\text{TP}, n], \tag{2.4}$$

$$\text{BOLD} = \text{SMA}(\text{TP}, n) - m \cdot \sigma[\text{TP}, n], \tag{2.5}$$

$$\text{TP} = (\text{Maxima} + \text{Minima} + \text{Fechamento})/3 \tag{2.6}$$

onde BOLU é a banda de cima, BOLD é a banda de baixo, n é o número de périodos, m é o número de desvios padrão e σ[TP, n] é o desvio padrão dos últimos n períodos dos valores de TP. A Figura 2.9 mostra um exemplo deste indicador.

Figura 2.9: Exemplo do indicador "Bollinger Bands"

2. ***Momentum***: O segundo é o grupo de impulso, que mostra os movimentos do preço das

ações ao longo do tempo e quão fortes estão os movimentos. Os indicadores primordias para este grupo são:

- **Média Móvel Convergente e Divergente**, do inglês ***Moving Average Convergence Divergence (MACD)***:

  A Média Móvel Convergente e Divergente é um indicador de acompanhamento de tendências, que mostra a relação entre duas médias móveis de um preço de uma ação. O MACD é calculado subtraindo a EMA de 26 períodos da EMA de 12 períodos. A EMA de 9 períodos é chamada de linha de sinal. Já o $\mathrm{EMA}_{\mathrm{MACD}}$ é o cálculo da média móvel exponencial dos valores obtidos pela fórmula (2.7). A MACD é frequentemente exibido com um histograma (Figura 4.5) que representa graficamente a distância entre o MACD e sua linha de sinal. As fórmulas (2.7) e (2.8) definem matematicamente a MACD.

$$\mathrm{MACD} = \mathrm{EMA}_{26} - \mathrm{EMA}_{12} \tag{2.7}$$

$$\mathrm{MACD}_{\mathrm{Histogram}} = \mathrm{MACD} - \mathrm{EMA}_{\mathrm{MACD}} \tag{2.8}$$

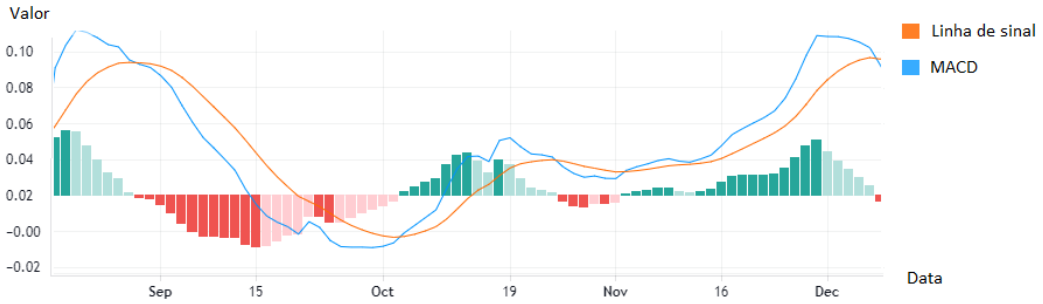

Figura 2.10: Exemplo do indicador "MACD"

- **Índice de força relativa** , do inglês ***Relative Strength Index (RSI)***:

  O índice de força relativa (RSI) é um indicador que mede a magnitude das mudanças recentes de preços para avaliar as condições de sobrecompra ou sobrevenda no preço de uma ação. O RSI é exibido como um oscilador, um gráfico de linha que se move entre dois extremos, e que varia de 0 a 100. Um ativo é considerado sobrecomprado quando o RSI está acima de 70, isto é, está num momento de alta forte e o próximo movimento provável é de uma queda. Da mesma forma, uma ação é considerada sobrevendida quando o RSI está abaixo de 30, no qual o próximo movimento provável é de uma alta pois, está num momento de queda forte (18). A fórmula do RSI é formada por (2.9) e (2.10):

$$\mathrm{RSI(n)} = 100 - \frac{100}{1 + \mathrm{RS(n)}} \tag{2.9}$$

$$\mathrm{RS(n)} = \frac{1}{n} \sum_{i=1}^{n} \frac{|G_i|}{|L_i|}, \tag{2.10}$$

onde G_i e L_i são, respectivamente, o ganho e a perda no momento i. A Figura 2.11 demonstra de forma gráfica este indicador. Note que o somatório dos valores de L_i tem que, obrigatoriamente, ser diferente de zero para o cálculo desse indicador. Se for igual a zero, o indicador não poderá ser calculado.

Figura 2.11: Exemplo do indicador "RSI"

- **Oscilador de Preço Absoluto** , do inglês ***Absolute Price Oscillator (APO)***:
  O oscilador de preço absoluto exibe a diferença entre duas médias móveis exponenciais do preço de uma ação e é expresso como um valor total dessa diferença. E essa diferença é sempre expressada entre uma EMA de período curto, necessariamente maior, e uma EMA de período longo. Um exemplo seria a diferença entre uma EMA de 12 e uma EMA de 26 períodos. A fórmula para este indicador é dada em 2.11 e na Figura 2.12, é possível visualizar este indicador.

$$\text{APO} = EMA \text{ de periodo curto} - EMA \text{ de periodo longo}, \tag{2.11}$$

Figura 2.12: Exemplo do indicador "APO"

3. **Volume**: O terceiro é o grupo de volume, que consiste no volume de negócios feitos pelos investidores no mercado. Os indicadores fundamentais deste grupo são:

   - **Saldo de Volume**, do inglês ***On Balance Volume (OBV)***: O saldo de volme é um indicador que calcula o valor contínuo do volume de negociação de um ativo e indica se esse volume está fluindo para dentro ou para fora dele. O OBV é um total cumulativo de volume (positivo e negativo) até aquele período, conforme definido em 2.12.

$$\text{OBV}_\text{t} = \text{OBV}_{\text{t}-1} + \begin{cases} \text{volume}, & \text{se fechamento}_\text{t} > \text{fechamento}_{\text{t}-1} \\ 0, & \text{se fechamento}_\text{t} = \text{fechamento}_{\text{t}-1} \\ \text{-volume}, & \text{se fechamento}_\text{t} < \text{fechamento}_{\text{t}-1} \end{cases} \tag{2.12}$$

onde $\mathrm{OBV_t}$ é igual valor atual do saldo de volume, $\mathrm{OBV_{t-1}}$ é o valor do último período do saldo de volume e volume é o número de negociações realizadas naquele momento. Na figura 2.13, este indicador é demonstrado de forma gráfica.

Figura 2.13: Exemplo do indicador "OBV"

- ***Chaikin A/D Line* (AD)**: Este indicador é baseado no volume desenvolvido por Marc Chaikin, que mede o fluxo cumulativo de dinheiro para dentro e para fora de uma ação. O A/D Line é calculado multiplicando o volume do período específico por um multiplicador que se baseia na relação do preço de fechamento com a faixa de alta-baixa. Logo, o *A/D Line* é formado pelo total do volume atual de fluxo de dinheiro. As fórmulas 2.13, 2.14 e 2.15 definem este indicador. O *Chaikin A/D Line* pode ser usado para afirmar uma tendência subjacente ou para prever reversões.

$$\text{A/D Line}_t = \text{A/D Line}_{t-1} - \text{Volume de fluxo de dinheiro}_t, \tag{2.13}$$

$$\text{Volume de fluxo de dinheiro}_t = \text{volume}_t \cdot \mathrm{N}, \tag{2.14}$$

$$\mathrm{N} = \frac{(\text{fechamento} - \text{minima}) - (\text{maxima} - \text{fechamento})}{\text{maxima} - \text{minima}}, \tag{2.15}$$

onde N é o multiplicador de fluxo de dinheiro.

- **Oscilador de Chaikin**, do inglês ***Chaikin A/D Oscillator (ADOSC)***: Este indicador mede a linha de distribuição de acumulação MACD. Para calcular o oscilador de Chaikin, é necessário subtrair uma EMA de 3 dias da linha de distribuição acumulada (LDA) de uma EMA de 10 dias da LDA. Isso mede o momento previsto por oscilações em torno da LDA, o que indica se os agentes de mercado estão acumulando as ações, isto é, se o mercado está otimista. As fórmulas 2.16, 2.17, 2.18 e 2.19 definem este indicador.

$$\mathrm{N} = \frac{(\text{fechamento} - \text{minima}) - (\text{maxima} - \text{fechamento})}{\text{maxima} - \text{minima}}, \tag{2.16}$$

$$\mathrm{M} = \mathrm{N} \cdot \mathrm{Volume_t}, \tag{2.17}$$

$$\mathrm{LDA} = \mathrm{M_{t-1}} + \mathrm{M_t}, \tag{2.18}$$

$$\text{OC} = \text{EMA}_3 \text{ de ADL} - \text{EMA}_{10} \text{ de ADL}, \tag{2.19}$$

onde N é o multiplicador de fluxo de dinheiro, N é o volume de fluxo de dinheiro e OC é o oscilador de Chaikin.

4. **Operações Matemáticas**, do inglês ***Math Operator***: O último é o grupo de operadores matemáticos, que visa fazer cálculos e relacioná-los com o gráfico de ações. Os indicadores essenciais deste grupo são:

   - **Mínimo**: Este indicador estima o preço de mínimo de uma ação dentre um número n de períodos, conforme sua definição em 2.20.

   $$\text{Minimo} = \min(\text{preco}_1, ..., \text{preco}_\text{n}), \tag{2.20}$$

   - **Máximo**: Este indicador calcula o preço de máximo de uma ação dentre um número n de períodos, de acordo com sua Fórmula 2.21.

   $$\text{Maximo} = \max(\text{preco}_1, ..., \text{preco}_\text{n}) \tag{2.21}$$

   - **Soma**: Este indicador mede a soma de todos os valores de fechamento de uma ação dentre um número n de períodos, de acordo com sua Fórmula 2.22.

   $$\text{Soma} = \sum_{\text{i=0}}^{\text{n}} \text{fechamento}_\text{i}, \tag{2.22}$$

As estratégias de posicionamento, do inglês *setups*, são estratégias de compra ou venda de ações, que são definidas a partir de particularidades (padrões) que acontecem no gráfico de vela ou nos indicadores técnicos, geralmente com uma ou duas outras condições de confirmação. Logo, essas estratégias visam facilitar o investidor a identificar padrões ou momentos ideais de se operar comprado ou vendido em uma ação a fim de obter lucros sobre ela (16). Abaixo, são explicadas as principais estratégias de posicionamento no mercado de ações (18):

- **Estratégia RSI sobrevendido/sobrecomprado**, do inglês ***RSI Overbought/Oversold setup***:

  Essa estratégia é baseada nos níveis de sobrecompra e sobrevenda do indicador técnico RSI, respectivamente, 70 e 30 (vide Figura 2.11). O RSI compara a magnitude dos ganhos recentes com as perdas recentes e gera um número que varia de 0 a 100, como explicado acima. Assim, uma operação de compra é recomendada quando o valor do RSI chega a mais de 30. E a saída dessa operação é quando atingido o valor de 70. Uma operação vendida é recomendada quando o valor de RSI está abaixo de 70 e a saída de venda começa quando o valor atinge o valor de 30.

- **Estratégia 9.1**, do inglês ***9.1 Setup***:

Essa estratégia é fundamentada na EMA de 9 períodos. Nesta estratégia, uma entrada de compra é recomendada no *candle* que torna a EMA em uma curva crescente, ao invés de continuar no movimento de descida, como estava anteriormente (a exemplo de Figura 2.14). De forma oposta, a saída da operação é realizada quando a EMA que estava crescente, começa a descer. A entrada na operação vendida é ativada no candle que torna a EMA em uma linha descrecente, que anteriormente estava crescente. A saída de compra começa quando a EMA crescente desce.

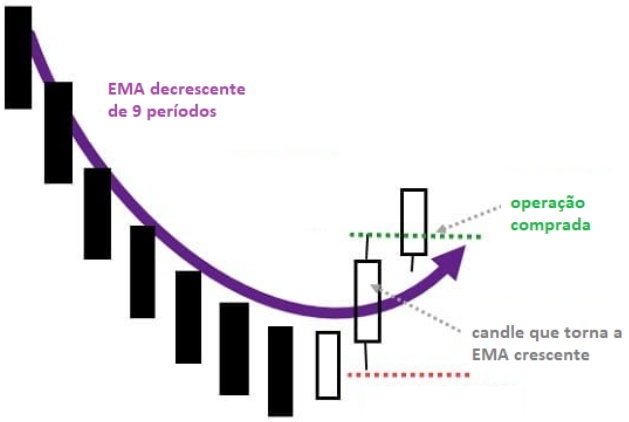

Figura 2.14: Estratégia 9.1

- **Estratégia da MACD padronizada**, do inglês ***MACD Standardization setup***:

  Esta configuração é baseada na padronização do histograma MACD para valores entre -1 e 1. E essa padronização é feita de acordo com a Fórmula 2.23. A partir disso, uma operação de compra é recomendada quando o valor do histograma fica menor que -0,45, e a saída de compra é quando o valor supera o valor de -0,05 (Figura 2.15). A entrada da operação vendida começa quando o valor do histograma está acima de 0,45 e a saída de venda é acionada quando esse valor está abaixo de 0,05.

$$\text{Histograma Padronizado} = 2 \cdot \frac{\text{MACD}_{\text{Histogram}} - min(\text{MACD}_{\text{Histogram}})}{max(\text{MACD}_{\text{Histogram}}) - min(\text{MACD}_{\text{Histogram}})} - 1, \quad (2.23)$$

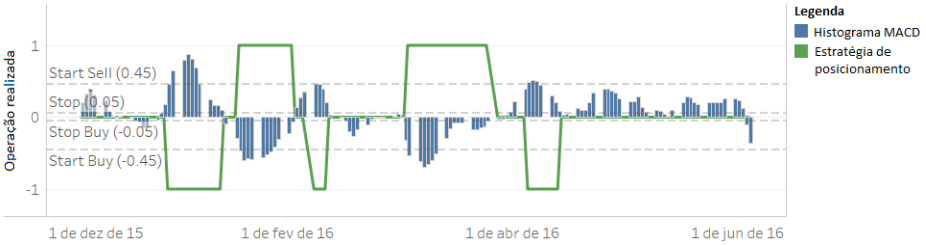

Figura 2.15: Estratégia da MACD padronizada

- **Estratégia do Ponto Contínuo**, do inglês ***Continuous Point (CP) setup***: Esta estratégia de entrada tem seu alicerce na SMA de 21 períodos. Para a entrada de compra, a operação é definida no candle que toca a SMA, e esta média móvel está crescendo. Na operação vendida, é necessário uma SMA decrescente e um candle que toca na SMA (Figura 2.16).

Figura 2.16: Estratégia do Ponto Contínuo

2.1.2 Aprendizado de Máquina

O AM, do inglês *Machine Learning*, é definido em (21), como um procedimento de análise que automatiza o desenvolvimento de modelos analíticos usando algoritmos, que aprendem interativamente a partir de dados de entrada. A característica interativa do AM é essencial pois, conforme os modelos são expostos a novos dados, eles se tornam capazes de se adaptarem de forma independente, isto é, aprendem com os modelos e cálculos feitos anteriormente para produzir decisões e resultados confiáveis sobre esse novo conjunto de dados (22).

Inicialmente, para resolver um problema em um computador (máquina), é preciso de um algoritmo. E um algoritmo é uma seqüência de instruções que devem ser realizadas para transformar a entrada em saída. Porém, existem muitas aplicações para as quais não se tem um algoritmo, mas existem dados de exemplo. E com o avanço da tecnologia, atualmente tem-se a capacidade de armazenar e processar grandes quantidades de dados, bem como acessá-los de locais fisicamente distantes em uma rede de computadores. Os dados armazenados só se tornam úteis quando analisados e transformados em informações que podem ser usadas, por exemplo, para fazer previsões.

É muito improvável identificar o processo no qual o dado foi concebido pelas diferentes características de interações dos usuários, mas é possível construir uma boa e útil aproximação desse processo. Dessa forma, é possível detectar certos padrões ou regularidades nestes dados. E é exatamente este o nicho do AM (22). Mas, para isso, é necessário o pré-processamento dos dados, isto é, transformar os dados brutos de entrada em um formato apropriado para uma análise subsequente. Ademais, o pré-processamento é importante para reduzir a dimensionalidade e remover dados inconsistentes a fim de produzir resultados mais consistentes e significativos (23).

Porém, o AM não é somente um problema de dados, é também uma parte do ramo da inteligência artificial. Para ser inteligente, um sistema que esteja em um ambiente em constante mudança deve ter a capacidade de aprender a se adequar. E se o sistema pode aprender e se adaptar a essas mudanças, o projetista do sistema não precisará prever nem fornecer soluções para todas as situações possíveis. Logo, o AM é bastante essencial para achar soluções para vários problemas em diversas áreas além do mercado de ações, como por exemplo, problemas de visão computacional, reconhecimento de fala e robótica (24).

O pré-processamento de dados e suas particularidades são explicadas na seção 2.1.2.1. E segundo (22), o AM pode ser divido em 3 tipos de aprendizado: o aprendizado supervisionado (seção 2.1.2.2), o aprendizado não-supervisionado (seção 2.1.2.3) e o aprendizado por reforço (seção 2.1.2.4). Na seção 2.1.2.5, é explicado como é feita a junção das técnicas de aprendizado por reforço com as do aprendizado profundo, formando o aprendizado por reforço profundo.

2.1.2.1 Pré-processamento de dados

O pré-processamento de dados é uma área ampla e consiste em várias estratégias e técnicas diferentes, inter-relacionadas de maneiras complexas. Sumariamente, as técnicas de pré-processamento de dados se enquadram em duas categorias: selecionar objetos de dados e atributos para a análise ou criar e alterar os atributos. Em ambos os casos, o objetivo é melhorar a análise de AM em relação a tempo, custo e qualidade (23).

Os conjuntos de dados podem ter um grande número de variáveis, o que resulta em um conjunto com um alto grau de dimensionalidade. Um exemplo para tal é: um conjunto de séries temporais que contêm o preço do fechamento diário de várias ações ao longo de um período de 30 anos. Diante deste cenário, existem vários benefícios para redução da dimensionalidade e de seleção de variáveis. Um benefício importante é que muitos algoritmos de AM funcionam melhor se a dimensionalidade e o número de atributos nos dados for menor (23). Isso ocorre porque a redução de dimensionalidade pode eliminar recursos irrelevantes, além de melhorar a precisão preditiva dos algoritmos, aumentando a compreensibilidade dos modelos construídos e produzindo resultados mais consistentes (25).

Algumas das abordagens mais comuns para redução de dimensionalidade, particularmente para dados contínuos, usam técnicas de álgebra linear para projetar os dados de um espaço de alta dimensão em um espaço de dimensão inferior. A Análise de Componentes Principais , do inglês *Principal Components Analysis (PCA)*, é uma técnica de álgebra linear para variáveis contínuas que encontra novos atributos (componentes principais) que (1) são combinações lineares dos atributos originais, (2) são ortogonais (perpendiculares) entre si e (3) capturar a quantidade máxima de variação nos dados. Por exemplo, os dois primeiros componentes principais capturam o máximo possível da variação nos dados com dois atributos ortogonais que são combinações lineares dos atributos originais (23).

A seguir, é explicado brevemente a matemática base para o PCA: os estatísticos resumem a

variabilidade de uma coleção de dados multivariados, isto é, dados que têm múltiplos atributos contínuos, calculando a matriz de covariância S dos dados (23). Dada uma matriz de dados D m por n, cujo m são as linhas que contém dados dos objetos e n colunas de atributos, a matriz de covariância de D é a matriz S, que tem entradas s_{ij} definidas como:

$$s_{ij} : covariance(d_{*i}, d_{*j}). \tag{2.24}$$

Em outras palavras, s_{ij} é a covariância dos atributos i^{th} e j^{th} (colunas) dos dados. A covariância de dois atributos é uma medida de quão fortemente os atributos variam juntos. Se $i = j$, isto é, os atributos forem iguais, a covariância é a variância do atributo. Se a matriz de dados D é pré-processado de modo que a média de cada atributo seja 0, então $S = D^T D$. Um objetivo do APC é encontrar uma transformação dos dados que satisfaça as seguintes propriedades:

1. Cada par de novos atributos tem 0 covariância (para atributos distintos).
2. Os atributos são ordenados em relação a quanto da variação dos dados cada atributo captura.
3. O primeiro atributo captura o máximo possível da variação dos dados.
4. Sujeito ao requisito de ortogonalidade, cada atributo sucessivo captura o máximo possível da variância restante.

E uma das abordagens mais utilizadas para seleção de variáveis é o limiar de variância, no qual é removida as variáveis cuja sua variância não atende a razão de variância mais bem explicada, que se dá pela fórmula 2.25.

$$\mathrm{Var}[x] = p(1-p), \tag{2.25}$$

onde p é a probabilidade da variável estar presente no conjunto de dados. Logo, esta técnica de seleção de variáveis visa a remoção de todas as variáveis de baixa variância e mantém as principais características desse conjunto.

2.1.2.2 Aprendizado supervisionado

A aprendizagem supervisionada, do inglês *supervised learning*, é uma técnica de AM com o objetivo de aprender a mapear a partir de dados de entrada, uma saída, cujos os valores corretos do mapeamento são fornecidos por um supervisor. A saída deste mapeamento pode prever um rótulo de classe do objeto de entrada, em problemas de classificação, ou pode ser um valor contínuo, em problemas de regressão (26).

O aprendizado supervisionado é caracterizado pela capacidade de construir modelos que aprendem com base em observações ou conjunto de dados existentes, o que permite a replicação desse aprendizado na previsão de futuros cenários. Neste tipo de aprendizado, os algoritmos possuem

a capacidade de generalizar a partir de regularidades constatadas apoiado em um determinado conjunto de treinamento, isto é, eles utilizam um conhecimento prévio do domínio, previamente estabelecido, para orientar a generalização de situações futuras (22).

Dentro do aprendizado supervisionado, existem duas tarefas principais, que são: a classificação e a regressão. Ambas estão relacionados aos tipos de dados utilizados e, conforme pode ser visto na Figura 2.17, adaptada de (23), as duas tarefas possuem a função de mapear um atributo de entrada x para um dos rótulos de classe predefinido, y. Porém, a característica principal que distingue classificação de regressão é que na regressão, uma tarefa de modelagem preditiva y é um atributo contínuo, enquanto que na classificação é um atributo discreto (23).

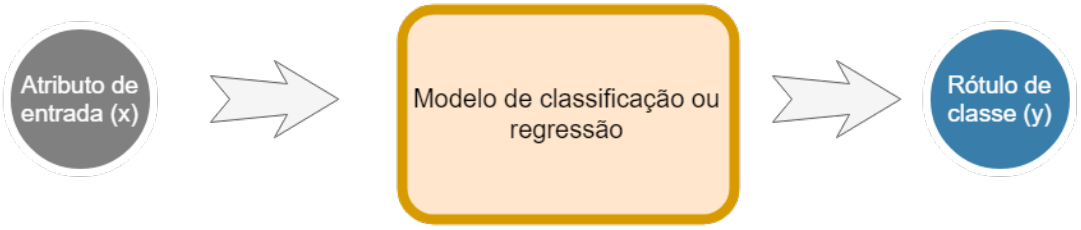

Figura 2.17: Classificação ou regressão como tarefa de mapear um atributo x em um rótulo de classe y

Um modelo de classificação pode servir como uma ferramenta explicativa para distinguir entre objetos de diferentes classes e também pode ser usado para prever o rótulo da classe de registros desconhecidos. Para isso, tem-se os atributos preditivos e o atributo-alvo. Os atributos preditivos designam as características que serão determinantes para que se classique um dado em um atríbuto-alvo. Logo, o atríbuto-alvo é a característica que se deseja predizer com base nas características dos atributos preditivos e ele é categórico com os rótulos que representam as classes dos dados.

Os exemplos de classificadores, ou técnica de classificação, incluem classificadores baseados em árvore de decisão, classificadores baseados em regras, redes neurais, máquinas de vetores de suporte e classificadores *naive Bayes*. Cada técnica emprega um algoritmo de aprendizado para identificar um modelo que melhor se ajusta ao relacionamento entre o conjunto de atributos e o rótulo de classe dos dados de entrada. O modelo gerado por um algoritmo de aprendizado deve ajustar os dados de entrada bem e prever corretamente os rótulos de classe de registros que ele nunca viu antes. Portanto, um objetivo chave do algoritmo de aprendizado é construir modelos com boa capacidade de generalização, isto é, modelos que prevêem com precisão os rótulos de classe de registros anteriormente desconhecidos (27).

Uma abordagem geral para resolver problemas de classificação é definir, primeiramente, um conjunto de treinamento, que consiste em registros cujos rótulos de classe são conhecidos. Após isso, o conjunto de treinamento é usado para criar um modelo de classificação, que é aplicado posteriormente ao conjunto de testes, que consiste em registros com rótulos de classe desconhecidos (22). Entre as tarefas de classificação, a técnica baseada na construção de redes neurais artificiais, do inglês *artificial neural networks* (ANN) ou de redes neurais profundas, do inglês *deep neural networks* (DNN), se destacam quando é necessária a identificação de padrões des-

critivos e preditivos, além de ser uma técnica muito utilizada e ter um potencial de performance significativo (28).

O estudo de ANN foi inspirado em tentativas de simular sistemas neurais biológicos. O cérebro humano consiste principalmente de células nervosas, chamadas neurônios, ligadas entre si com outros neurônios por meio de fios de fibra chamados axônios. Os axônios são usados para transmitir impulsos nervosos de um neurônio para outro sempre que os neurônios são estimulados. Um neurônio é conectado aos axônios de outros neurônios por meio de dendritos, que são extensões do corpo celular do neurônio. O ponto de contato entre um dendrito e um axônio é chamada de sinapse. Análogo à estrutura do cérebro humano, uma ANN é composta de um conjunto interconectado de nós e elos direcionados (23).

A Figura 2.18 ilustra uma rede neural simples conhecida como *perceptron*. O perceptron consiste em dois tipos de nós: nós de entrada, que são usados para representar os atributos de entrada e um nó de saída, que é usado para representar a saída do modelo. Os nós em uma rede neural são comumente conhecidos como neurônios. Em um *perceptron*, cada nó de entrada é conectado por meio de um elo com um peso ao nó de saída. O elo ponderado é usado para emular a força da conexão sináptica entre os neurônios. Como nos sistemas neurais biológicos, treinar um modelo *perceptron* equivale a adaptar os pesos dos elos até que eles se ajustem às relações de entrada-saída dos dados subjacentes (29).

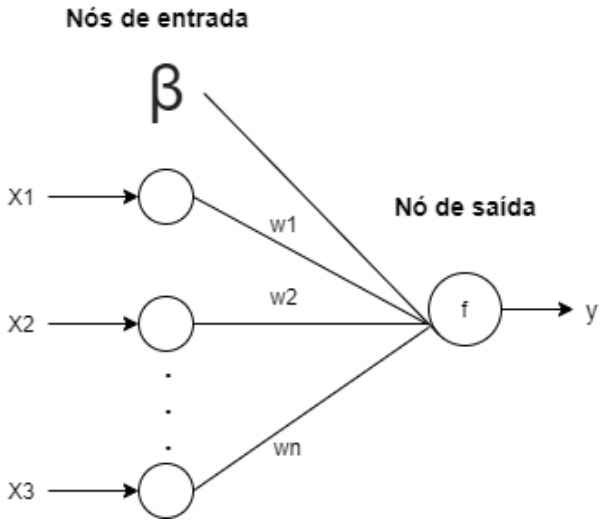

Figura 2.18: A estrutura de um perceptron

Um *perceptron* calcula seu valor de saída, y, realizando uma soma ponderada em suas entradas, somando um fator de viés (*bias*) β da soma e, em seguida, examinando o sinal do resultado (vide Fórmula 2.26).

$$y = f\left(\beta + \sum_{i=1}^{n} w_i x_i\right), \tag{2.26}$$

Durante a fase de treinamento de um modelo *perceptron*, os parâmetros de peso w são ajusta-

dos até que as saídas do *perceptron* se tornem consistentes com as saídas reais dos exemplos de treinamento. O ajuste desses pesos é dado pela Fórmula 2.27.

$$w_j^{(k+1)} = w_j^{(k)} + \lambda(y_i - \hat{y}_i^{(k)})x_{ij}, \tag{2.27}$$

onde $w^{(k)}$ é o parâmetro de peso associado ao i-ésimo elo de entrada após a k-ésima iteração, λ é um parâmetro conhecido como taxa de aprendizagem (*learning rate*), e x_{ij} é o j-ésimo valor do atributo do exemplo de treino x_i.

Na fórmula de atualização de peso, os elos que mais contribuem para o termo de erro são os que requerem o maior ajuste. No entanto, os pesos não devem ser alterados muito drasticamente porque o termo de erro é calculado apenas para o exemplo de treinamento atual. Caso contrário, os ajustes feitos anteriormente as iterações serão desfeitas. A taxa de aprendizagem λ, um parâmetro cujo valor está entre 0 e 1, pode ser usado para controlar a quantidade de ajustes feitos em cada iteração. Se, λ está próximo de 0, então o novo peso é muito influenciado pelo valor do peso antigo. Por outro lado, se λ for próximo a 1, então o novo peso é bastante sensível à quantidade de ajuste realizado na iteração atual (23).

Uma ANN ou DNN tem uma estrutura mais complexa do que a de um modelo *perceptron*. As complexidades adicionais podem surgir de várias maneiras:

- A rede pode conter várias camadas intermediárias entre suas camadas de entrada e saída. Essas camadas intermediárias são chamadas de camadas ocultas e os nós embutidos nessas camadas são chamados de nós ocultos. A estrutura resultante é conhecida como rede neural multicamadas (haja vista Figura 2.19). Em uma rede neural *feed-forward*, os nós em uma camada são conectados apenas aos nós da próxima camada. O *perceptron* é uma rede neural *feed-forward* de camada única porque tem apenas uma camada de nós: a camada de saída que executa operações matemáticas complexas. Em uma rede neural recorrente, os links podem conectar nós dentro da mesma camada ou nós de uma camada para as camadas anteriores (23).
- A rede pode usar diversos tipos de funções de ativação. Essas funções de ativação permitem que os nós ocultos e de saída produzam valores de saída que são não lineares em seus parâmetros de entrada. Exemplos de funções de ativação incluem funções lineares, sigmóides (logísticas) e tangentes hiperbólicas, conforme explicitado abaixo:
  - **Função Linear (Linear function)**: Essa função pega as entradas, multiplicadas pelos pesos de cada neurônio, e cria um sinal de saída proporcional à entrada. Como seu próprio nome diz, é uma função que só pode ser utilizada para classificar dados lineares. Sua fórmula é dada por 2.28 e pode ser vista na Figura 2.20.

    $$f(x) = c \cdot x, \tag{2.28}$$

    onde c é uma constante.

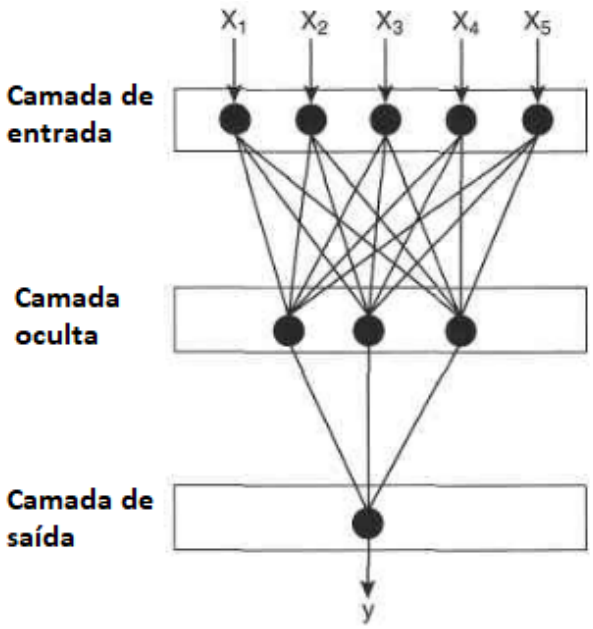

Figura 2.19: A estrutura de uma rede neural de múltiplas camadas

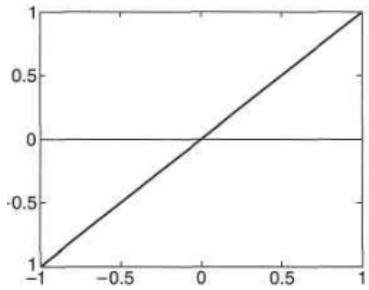

Figura 2.20: Função linear

– **Função Sigmoid (*Sigmoid function*)**: Também conhecida como função logística por se parece com esse tipo de regressão. Sua vantagem é que tem um gradiente suave, evitando saltos nos valores de saída. Porém, essa função exige muito recurso computacional para ser calculada. Sua fórmula é definida por 2.29 e pode ser visualizada na Figura 2.21.

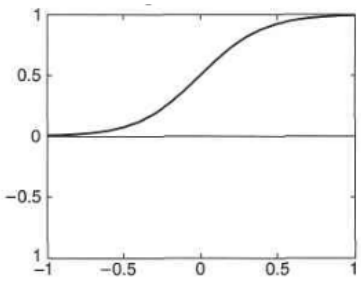

Figura 2.21: Função Sigmoid

$$\sigma(x) = \frac{1}{1 + e^x} \tag{2.29}$$

– **Função Tangente Hiperbólica (TanH)**: Essa função tem um formato bem parecida com a sigmoid, mas varia de -1 a 1, em vez de 0 a 1 como na sigmoide. Logo, tem sua vantagem de ter o zero centralizado, o que facilita para modelar entradas com valores fortemente negativos, neutros e fortemente positivos. Sua fórmula é definida em 2.30 e pode ser visualizada na Figura 2.22.

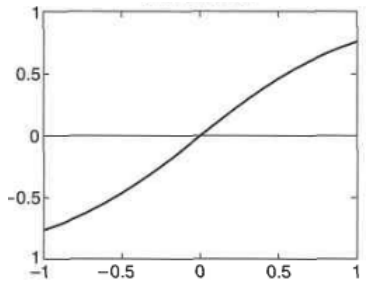

Figura 2.22: Função Tanh

$$tanh(x) = 2\sigma(2x) - 1 \tag{2.30}$$

– **Função degrau (*Step function*)**: Uma função de ativação baseada em limite. Se o valor de entrada estiver acima ou abaixo de um certo limite, o neurônio é ativado e envia exatamente o mesmo sinal para a próxima camada. Sua fórmula é encontrada em 2.31 e visualização na Figura 2.23.

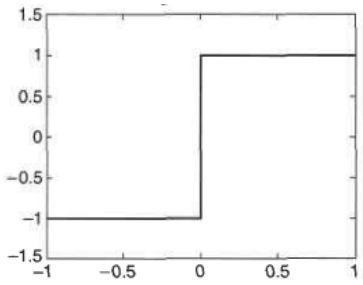

Figura 2.23: Função Degrau

$$f(x) := \begin{cases} \text{-1}, & \text{se } x < 0 \\ 0, & \text{se } x = 0 \\ 1, & \text{se } x > 0 \end{cases} \tag{2.31}$$

– **Função Unidade Linear Retificada ou ReLU (*Rectified Linear Unit*)**:

Redes com a função ReLU são fáceis de otimizar, já que a ReLU é extremamente parecida com a função identidade, além de favorecer uma convergência rápida da rede.

Por este motivo, a função é ReLU é bastante utilizada (29). Sua fórmula é definida por 2.32 e um exemplo de seu formato é demonstrado na Figura 2.24.

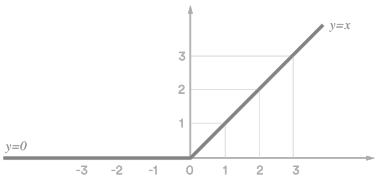

Figura 2.24: Função ReLU

$$ReLU(x) = max\{0, x\} \tag{2.32}$$

Essas complexidades adicionais permitem que as ANN ou DNN modelem relacionamentos mais complexos entre as variáveis de entrada e saída. E a diferença entre uma ANN e uma DNN é pelo fato da ANN conter até no máximo 3 camadas ocultas, enquanto a DNN contém 3 ou mais camadas ocultas.

O objetivo do algoritmo de aprendizagem da ANN ou DNN é determinar um conjunto de pesos que minimiza a soma total dos erros quadráticos (vide Fórmula 2.33). Note, pela fórmula, que a soma dos erros quadráticos depende de w pois, a classe $\hat{y}$ prevista é uma função dos pesos atribuídos aos nós ocultos e de saída.

$$E(w) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.33}$$

Na maioria dos casos, a saída de uma ANN ou DNN é uma função não linear de seus parâmetros por causa da escolha de suas funções de ativação, por exemplo, função sigmóide ou ReLU. Como resultado, não é simples derivar uma solução para w que seja globalmente ótima. Algoritmos ambiciosos, como os baseados no método de gradiente descendente, foram desenvolvidos para resolver o problema de otimização de forma eficiente. A fórmula de atualização de peso usada pelo método de gradiente descendente pode ser escrita conforme a Fórmula 2.34 (23).

$$w_j \leftarrow w_j - \lambda\frac{\partial E(w)}{\partial w_j} \tag{2.34}$$

onde λ é a taxa de aprendizado. O segundo termo afirma que o peso deve ser aumentado em uma direção que reduza o termo de erro geral. No entanto, como a função de erro não é linear, é possível que o método de descida de gradiente fique preso em um mínimo local.

Uma técnica conhecida como retropropagação, do inglês *back-propagation*, foi desenvolvida para resolver esse problema. Existem duas fases em cada iteração do algoritmo: a fase de avanço e a fase de retrocesso. Durante a fase de avanço, os pesos obtidos na iteração anterior são usados para calcular o valor de saída de cada neurônio na rede. Durante a fase de retrocesso, a fórmula de atualização de peso é aplicada na direção inversa (23).

Durante a fase de treinamento da rede, existem dois termos essenciais: época e mini-lote. Época é o número de vezes que um algoritmo de AM treina com o conjunto de dados completo (lote). E um mini-lote, do inglês *mini-batch*, é uma fração desse conjunto completo de dados. A utilização do mini-lote permitirá manter dois objetivos: ajustar dados suficientes na memória do computador e manter a filosofia de vetorização ao mesmo tempo. O que acarreta em um custo computacional mais baixo e um tempo de processamento da rede menor (29).

2.1.2.3 Aprendizado não-supervisionado

Na aprendizagem não-supervisionada, do inglês *unsupervised learning*, o objetivo é aprender um mapeamento de entrada para uma saída sem um supervisor, diferentemente dos algoritmos com aprendizagem supervisionada que assumem a existência de um supervisor ou de uma medida de adequação para classificação de exemplos de treinamento. Este tipo de aprendizado não possui a vantagem de um ambiente de treinamento com casos para calibração de um modelo de classificação (23).

Ao invés disso, os algoritmos não-supervisionados propõem hipóteses para explicar as observações a partir de informações encontradas nos dados que descrevem os objetos e seus relacionamentos. Logo, o objetivo é que os objetos dentro de um grupo sejam similares (ou relacionados) uns aos outros e diferentes (ou não relacionados a) os objetos em outros grupos (23).

2.1.2.4 Aprendizado por reforço

A aprendizagem por reforço, do inglês *reinforcement learning* (RL), se preocupa com a forma como os modelos gerados, chamados de agentes autônomos, devem tomar ações em um ambiente, de modo a escolher as melhores ações para atingir seu objetivo. O problema, devido à sua generalidade, é estudado em muitas outras disciplinas, como teoria dos jogos, para o modelo aprender a jogar jogos de tabuleiros, robótica, para o algoritmo aprender a controlar um robô móvel, entre outras disciplinas. Cada vez que o agente executa uma ação em seu ambiente, o algoritmo tenta maximizar alguma noção de recompensa acumulativa para as ações corretamente feitas. Porém, há a penalização para ações feitas erroneamente. Dessa forma, é possível indicar a conveniência do estado resultante para o agente (modelo) (30).

Neste aprendizado, o conceito de maximizar a recompensa através do desempenho é feito por um processo de ajuste dos parâmetros feitos pela interação contínua com o ambiente. Dessa forma, ao invés de haver um supervisor que indique o resultado esperado a cada estímulo fornecido como entrada, existe um procedimento que atribui uma nota para a resposta da máquina de aprendizado ao estímulo, com o objetivo de alcançar o nível máximo de sucesso no seu funcionamento com base em um índice estabelecido (30).

Logo, na aprendizagem por reforço, o agente aprende com uma série de reforços, que são punições ou recompensas. Um exemplo seria o ponto para uma vitória no final de um jogo de

damas, que significa que o agente que executou certas ações que culminaram em um resultado final correto. Porém, a falta de gorjeta no final da viagem, dá ao agente de táxi uma indicação de que ele fez algo de errado. Cabe ao agente decidir quais das ações anteriores ao reforço foram as mais indicadoras para que ele alcance o sucesso. O seu principal objetivo da aprendizagem por reforço é escolher uma sequência de ações de forma a maximizar a recompensa a longo prazo (31).

O aprendizado por reforço é modelado por uma série de componentes, no qual o diagrama é apresentado na Figura 2.25. Esses componentes são explicitados abaixo:

- **Agente**: um agente executa ações, isto é, um agente aprende a atingir uma meta em um ambiente incerto e potencialmente complexo. Logo, o algoritmo é considerado o agente.
- **Ações (A)**: É o conjunto de todos os movimentos possíveis que o agente pode fazer. Uma ação é quase autoexplicativa, mas deve-se notar que os agentes geralmente escolhem de uma lista de ações possíveis e discretas. Nos mercados de ações, a lista pode incluir a compra, venda ou manutenção de qualquer um de uma matriz de ativos financeiros e seus derivativos.
- **Interpretador**: É o usuário, em muitos casos o cientista de dados, que define as regras de recompensas, ambiente e ações do agente.
- **Ambiente**: O mundo pelo qual o agente se move e que responde ao agente. O ambiente toma o estado atual e a ação do agente como entrada e retorna como saída a recompensa do agente e seu próximo estado.
- **Estado (S)**: Um estado é uma situação concreta e imediata em que o agente se encontra, isto é, um local e momento específico, uma configuração instantânea que coloca o agente em relação a outras coisas importantes, como ferramentas, obstáculos, inimigos ou prêmios.
- **Recompensa (R)**: Uma recompensa é o retorno pelo qual medimos o sucesso ou o fracasso das ações de um agente em determinados estados durante uma janela de tempo.
- **Política (π)**: A política é a estratégia que o agente emprega para determinar a próxima ação com base no estado atual. Logo, ela mapeia estados para ações, no qual são escolhidas as ações que prometem a maior recompensa.

Com base nos componentes explicados acima, tem-se a equação de Bellman (32), que calcula o valor esperado a longo prazo da recompensa total com desconto, a partir ações executadas pelo agente, em oposição à recompensa de curto prazo R. Dessa forma, a fórmula mede para cada estado s, qual ação retorna a melhor recompensa dentro do conjunto de ações a a partir de um estado s, em contraste do valor de recompensa futuro $V(s')$ (Fórmula 2.35).

$$V(s) = \max_a(R(s,a) + \gamma V(s')), \tag{2.35}$$

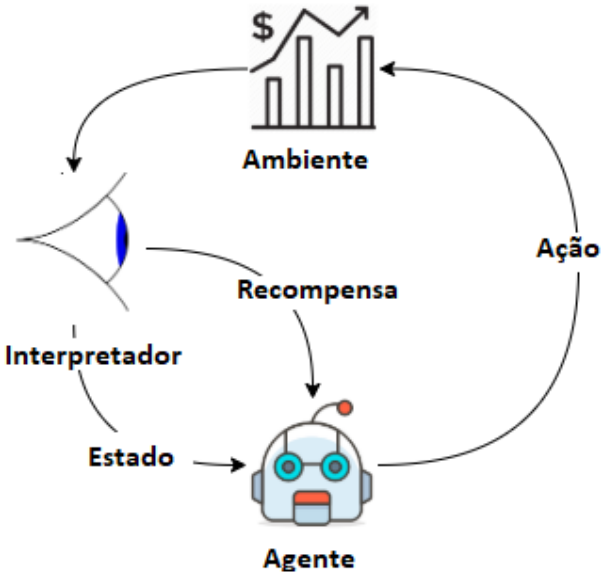

Figura 2.25: Estrutura do aprendizado por reforço

onde γ é uma constante chamada de fator de desconto, que tem o objetivo de amortecer o efeito das recompensas futuras na escolha de ação do agente.

O RL usa a estrutura formal do Processo de Decisão de Markov, do inglês *Markov Decision Process* (MDP), para definir a interação entre um agente de aprendizagem e seu ambiente em termos de estados, ações e recompensas. Essa estrutura é uma maneira simples de representar características essenciais dos problemas de IA. Esses recursos incluem uma sensação de causa e efeito, um senso de incerteza e não-determinismo, e a existência de metas explícitas, o que é algo muito similar ao aprendizado humano.

O MDP é um processo estocástico em que a distribuição de probabilidade condicional para os estados futuros do processo depende apenas do estado atual. Além disso, fornece uma estrutura matemática para modelar a tomada de decisão em que os resultados são parcialmente aleatórios e parcialmente sob o controle de um tomador de decisão, conforme pode ser vista na Figura 2.26. Essa figura demonstra o cenário do mundo real, no qual o processo de escolha de ações do agente é estocástico, isto é, não se tem a clareza de que o agente comprará aquele ativo ou se vai outro tipo de ação (vender ou manter o ativo), no exemplo do mercado de ações.

Logo, neste cenário de exemplo, há a probabilidade de 80 % do agente executar a ação esperada (ideal) e outros 20 % distribuídos para não agir de forma esperada. Enquanto que no cenário determínistico, existe a certeza de qual ação será tomada, isto é, a probabilidade de ele executar a ação ideal é de 100 %. Porém, este cenário não se adequa ao mundo real e não pode ser considerado para análises e modelos de IA (32).

A fórmula do DMP é dado por 2.36, onde é adicionado a probabilidade do agente tomar aquela ação multiplicado pelo valor do estado futuro. No final, há um somatório dessas probabilidades, o que permite o cálculo do valor esperado do estado no cenário estocático pois, não se sabe ao

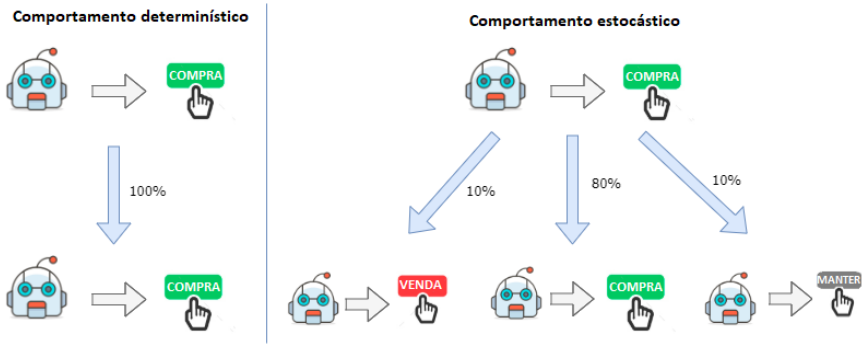

Figura 2.26: Processo Deterministico e Estocástico

certo qual ação o agente executará. (Fórmula 2.36).

$$V(s) = \max_{a}(R(s,a) + \gamma \sum_{s'} P(s,a,s')V(s')), \tag{2.36}$$

Dentro dos algoritmos do RL, um dos mais utilizados é o *Q-Learning*. O *Q-learning* é um algoritmo que busca encontrar a melhor ação a ser tomada, dado o estado atual. É considerado sem política, do inglês *off-policy*, porque a função *Q-learning* aprende com ações que estão fora da política atual, como executar ações aleatórias (33). Dessa forma, o *Q-learning* busca aprender uma política que maximize a recompensa total. O "Q"significa qualidade que, neste caso, representa a utilidade de uma determinada ação para obter alguma recompensa futura.

O *Q-learning* aprende a política ideal, mesmo quando as ações são selecionadas de acordo com uma política mais exploratória ou até aleatória. A fórmula do *Q-learning* é definida em 2.37, no qual é bem parecida com a equação do MDP. Porém, nessa fórmula é utilizado o conceito do "Q", que fornece um cálculo da qualidade daquela ação ser tomada ao estar em um estado s. Logo, a fórmula se preocupa mais com as recompensas dadas ao agente, gerando a qualidade da ação com o objetivo de executar as ações de maior qualidade e obter um somatório de recompensas maior possível.

$$Q(s,a) = R(s,a) + \gamma \sum_{s'}(P(s,a,s') \max_{a'} Q(s',a')) \tag{2.37}$$

A diferença temporal, do inglês *Temporal Difference (TD)*, é extramente essencial para o algoritmo de *Q-Learning* (33). Essa diferença é o cálculo do valor de "Q", porém em períodos diferentes. Logo, o objetivo é encontrar os melhores valores de "Q" com base nos valores já calculados previamente. E por este motivo, a diferença temporal é muito importante para uma performance significativa de um modelo de *Q-Learning*. A fórmula do TD é descrita em 2.38, no qual é possível ver que é calculado o valor de "Q" como na Fórmula 2.37, porém deste valor é substraído o valor de "Q" obtido em um momento prévio. Logo, isso faz com que as ações realizadas em momentos anteriores tenham influência nas novas ações a serem executadas pelo agente.

$$TD(a,s) = R(s,a) + \gamma \max_{a'} Q(s',a') - Q_{t-1}(s,a) \tag{2.38}$$

Ao incluir a fórmula da diferença temporal na fórmula do *Q-Learning*, obtém-se a Fórmula final definida em 2.39

$$Q_t(s,a) = Q_{t-1}(s,a) + \alpha(R(s,a) + \gamma \max_{a'} Q(s',a') - Q_{t-1}(s,a)), \tag{2.39}$$

onde α é a taxa de aprendizado, que varia de 0 a 1. Essa taxa é extremamente importante pois, se o valor for próximo de 0, a decisão da próxima ação terá um viés muito grande da ação anterior. Por outro lado, se for próximo de 1, o agente considerará de forma leve as ações e aprendizados obtidos no passado para tomar a próxima ação.

2.1.2.5 Aprendizado por reforço profundo

O DL consiste em várias camadas de redes neurais, projetadas para executar tarefas mais sofisticadas. A construção de modelos de DL foi inspirada no design do cérebro humano, mas simplificada. Os modelos de DL consistem em algumas camadas de rede neural que são, em princípio, responsáveis por aprender gradualmente recursos mais abstratos sobre dados específicos.

Já o RL, como explicado na seção 2.1.2.4, emprega um sistema de recompensas e penalidades para obrigar o computador a resolver um problema sozinho. O envolvimento humano é limitado à mudança do ambiente e ao ajuste do sistema de recompensas e penalidades. Como o computador maximiza a recompensa, ele está propenso a procurar maneiras inesperadas de fazê-lo. O envolvimento humano é focado em impedir que ele explore o sistema e motive a máquina a executar a tarefa da maneira esperada. O RL é útil quando não existe uma maneira fixa ou ideal de executar uma tarefa, mas existem regras que o modelo deve seguir para desempenhar corretamente suas tarefas.

O RL foi combinado com o DL para aprender recursos complexos e regras de negociação de dados de ações. A combinação desses dois conceitos é chamado de aprendizado por reforço profundo, do inglês *deep reinforcement learning* (DRL). O DRL está prestes a revolucionar a inteligência artificial (IA) e representa um passo em direção à construção de sistemas autônomos com uma compreensão de alto nível do mundo visual (34). Dentre os algoritmos RL disponíveis, o *Q-learning* é uma abordagem amplamente utilizada que não requer um modelo definido do ambiente (34), conforme explicado acima.

Assim, o DRL visa estimar os valores "Q" para cada ação com base em uma determinada situação e é treinado com base na equação de atualização do *Q-learning*. Ele permite que o modelo extraia as principais características de cada circunstância para prever a recompensa e a melhor ação, mesmo em situações não reveladas. Esta arquitetura DRL é baseada no estado da arte do DRL proposto pela DeepMind e é o método mais adotado para permitir que os modelos

aprendam políticas de controle complexas em diversas áreas, uma vez que os resultados superam significativamente a linha de base dos modelos de RL (35).

Dentro desta arquitetura, os estados do ambiente formam um vetor de estados, que por sua vez, é passado como entrada na rede neural. Em seguida, a ANN ou DNN tentará prever qual ação deve ser executada, retornando como saídas um valor Q para cada uma das ações possíveis. Ao final, a função de perda é retropropagada na rede neural para atualizar os pesos no processo de aprendizagem com base, no qual é comparado o valor previsto de "Q" com o valor esperado de "Q". O valor esperado de "Q" é calculado no momento anterior que o agente estava naquele mesmo estado com base na fórmula do *Q-Learning*. Nesta parte, o agente estará aprendendo com o processo de aprendizado profundo em conjunto com o aprendizado por reforço. A Figura 2.27 mostra o diagrama deste processo e a fórmula 2.40 mostra a função de erro utilizada, que é propagada de volta para a rede neural para atualizar os seus pesos.

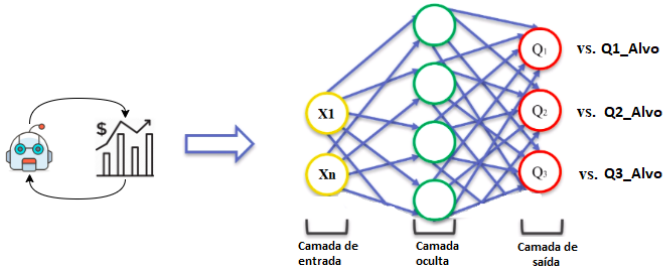

Figura 2.27: Processo de aprendizagem do DRL

$$L = \frac{1}{n}\sum_{i=1}^{n}(Q\_Alvo - Q)^2 \tag{2.40}$$

Para o processo de tomada de decisões (ações), é necessário utilizar o método *Epsilon-greedy*. Este método escolhe entre a a ações de melhor qualidade e ações aleatórias, pois no início das interações, o modelo não sabe quais são as melhores ações e, portanto, ele deve explorar o ambiente para conhecer as ações que resultarão na maior recompensa possível. Dessa forma, o agente no início explorará o ambiente e, posteriormente, agirá conforme as ações de maior qualidade. A fórmula do *Epsilon-greedy* é definida em 2.41.

$$\text{Ação} = \begin{cases} \text{ação aleatória}, & \text{se } rand < \epsilon \\ argmax(Q(s,a)), & \text{senão}, \end{cases} \tag{2.41}$$

onde ϵ é um valor que, após um número aleatório ($rand$) for escolhido entre 0 e 1, define se naquele momento o agente executará uma ação aleatória ou uma ação que tenha a maior qualidade entre todas as ações possíveis. No início das interações, o valor de ϵ é igual a 1 e, após cada interação, esse valor é decaído em 5 %. O que faz com que após um certo número de interações, o modelo comece a escolher mais ações de qualidade do que ações aleatórias.

2.2 MÉTRICAS DE AVALIAÇÃO E PERFORMANCE

Um modelo gerado por um algoritmo de AM deve ajustar bem os dados de entrada e prever corretamente os rótulos de classe de registros que não vistos anteriormente. Portanto, o objetivo primordial dos algoritmos de AM é construir modelos com boa capacidade de generalização, isto é, modelos que preveem com precisão os rótulos de classe de registros anteriormente desconhecidos. Essa precisão ou a avaliação do desempenho de um modelo é baseada nas contagens de registros de teste, correta e incorretamente, previstas pelo modelo. Essas contagens são tabuladas em uma tabela conhecida como matriz de confusão, do inglês *confusion matrix* (36).

A seção 2.1 descreve a matriz de confusão para um problema de classificação binária. Cada entrada f_{ij} nesta tabela denota o número de registros da classe i previstos como da classe j. Por exemplo, f_{11} é o número de registros da classe 1 previstos corretamente como a classe 1 e cada registro previsto de forma correta é chamado de verdadeiro positivo, do inglês true positive (TP). E o f_{00} é o número de registros da classe 0 previstos corretamente como a classe 0 e cada registro previsto é chamado de verdadeiro negativos, do inglês "True negative" (TN). Já o f_{10} é o número de registros da classe 1 previstos incorretamente como a classe 0, e cada registro previsto é chamado de falso negativo, do inglês "False negative" (FN). E, por fim, o f_{01} é o número de registros da classe 0 previstos incorretamente como classe 1, onde cada registro previsto é chamado de falso positivo, do inglês "False positive" (FP) (23).

Embora uma matriz de confusão forneça as informações necessárias para determinar o desempenho de um modelo de classificação, resumir essas informações com um único número tornaria mais conveniente comparar o desempenho de diferentes modelos. Isso pode ser feito usando uma métrica de desempenho, como a acurácia, que é definida da seguinte maneira (36):

$$\text{Acurácia} = \frac{\text{Número de predições corretas}}{\text{Número total de predições}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \tag{2.42}$$

Equivalentemente, o desempenho de um modelo pode ser expresso por sua taxa de erro, que é dada pela seguinte equação (36):

$$\text{Taxa de erro} = \frac{\text{Número de predições erradas}}{\text{Número total de predições}} = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \tag{2.43}$$

Tabela 2.1: Matriz de confusão de exemplo para um problema de duas classes

		Previsto	
		Classe = 1	Classe = 0
Real	Classe = 1	f_{11} (TP)	f_{10} (FN)
	Classe = 0	f_{01} (FP)	f_{00} (TN)

Porém, a acurácia trata todas as classes como igualmente importantes, e ela não é adequada para analisar conjuntos de dados desequilibrados, em que a classe rara é considerada mais inte-

ressante do que a classe majoritária. Para isso, foram desenvolvidas duas novas métricas chamadas de precisão e revocação, do inglês "precision" e "recall", respectivamente. A revocação e a precisão são duas métricas amplamente utilizadas em aplicações em que a determinação bem-sucedida de uma das classes é considerada mais significativa do que a detecção das outras classes (23). Uma definição formal dessas métricas é fornecida abaixo (36):

$$\text{Precisão} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{2.44}$$

$$\text{Revocação} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{2.45}$$

A precisão determina a fração de registros que realmente são de classes positivas no grupo em que o classificador declarou como uma classe positiva. Quanto maior a precisão, menor o número de erros falsos positivos cometidos pelo classificador. Revocação mede a fração de exemplos positivos corretamente previstos pelo classificador. Classificadores com grande valor de revocação têm poucos exemplos positivos classificados erroneamente como de classe negativa (23).

Geralmente, é possível construir modelos de linha de base que maximizam uma métrica, mas não a outra. Por exemplo, um modelo que declara que cada registro é da classe positiva terá uma recordação perfeita, mas uma precisão muito baixa. Então, construir um modelo que maximize a precisão e a revocação é o principal desafio dos algoritmos de classificação. Essas duas métricas podem ser resumidas em outra conhecida como medida de F_1, do inglês "F_1 measure", conforme a seguir (36):

$$F_1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}, \tag{2.46}$$

Dessa forma, conseguir um valor alto da medida de F_1 é o objetivo dos algoritmos classificadores, pois essa métrica é a relação entre as duas principais métricas de desempenho dos modelos classificatórios. Logo, a partir do momento em que F_1 tem um valor alto, os valores de precisão e de revocação também são igualmente altos (37).

2.3 ESTADO DA ARTE

Em (1), é proposto um sistema de suporte à decisão para investimentos no mercado de ações utilizando Redes Neurais (RN) *One-Against-All* (OAA). Ainda, em (1), o sistema proposto investigou modelos de aprendizado de máquina para predição de estoque, como Algoritmos Genéticos (GAs), Máquinas de Vetores de Suporte (SVMs) e RNs. O estudo teve como base o diagrama apresentado na Figura 2.28. Os resultados experimentais mostram que a classificação multi-binária usando a técnica OAA supera as técnicas tradicionais com a maior taxa de retorno de

57,67 % em um período de 3 anos, de 31/10/2007 a 20/12/2010.

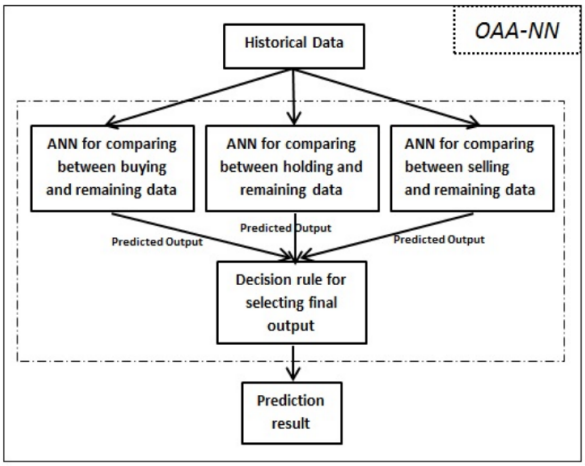

Figura 2.28: Diagrama base do estado da arte (1)

Em (2), uma nova estratégia de investimento é proposta para ganhos otimizados em investimentos no mercado de ações com base em RN. Os resultados indicam que a estrutura RN proposta pode ajudar os investidores a tomar decisões de investimento, obtendo 78 % do F1-Score nas operações de compra e 60 % nas operações de venda. Sem esta estrutura NN proposta, a pontuação F1 para operações de compra é 59 % e 45 % para operações de venda.

Foi proposto uma framework de inteligência de negócio, do inglês *business intelligence* (BI), em (12), que visa ajudar os investidores na tomada de decisões comerciais mais eficientes. O estudo propôs técnicas e ferramentas para coletar dados, transformar, armazenar, analisá-los e apresentá-los ao usuário final, no caso para o investidor.

Neste trabalho, é proposto uma estrutura de IA baseada em DRL que aprende regras de negociação para reconhecer padrões e maximizar o lucro do investidor. Ademais, para melhorar a usabilidade e interatividade dos investidores com a plataforma, foi criada uma camada de visualização sugerindo a ação a ser executada em cada momento.

3 METODOLOGIA

Este capítulo apresenta a *framework* de DRL proposta, que tem como premissa criar um modelo de AM a partir dos dados históricos das ações, cujo o objetivo é desenvolver um modelo que reconheça padrões e preveja a melhor ação a ser tomada em cada momento, permitindo que a *framework* seja utilizada como um sistema de apoio a tomada de decisões dos investidores. Além disso, foi criada uma camada de visualização da framework, para que ela seja intuitiva e útil para o usuário final. Para atingir este objetivo, foi feito um diagrama de blocos com as etapas necessárias para chegar no resultado final da *framework* proposta.

Conforme pode ser visto na Figura 3.1, a primeira etapa (Bloco A) é a extração, transformação e carga dos dados, no qual o objetivo é extrair os dados do mercado de ações das plataformas, transformá-los em um formato apropriado para análises subsequentes e, a partir disso, guardá-los em uma base de dados. O segundo passo (Bloco B) é o processo de realizar os cálculos dos indicadores técnicos e das estratégias de posicionamento a partir dos dados das ações, obtidos no primeiro bloco. Todo o processo do Bloco B é explicado na seção 3.2.

A terceira etapa (Bloco C), explicada na seção 3.3 é a etapa da seleção de *features* e redução de dimensionalidade, uma vez que há muitas variáveis no mercado de ações, devido à sua alta complexidade. O quarto passo (Bloco D) é a utilização do DRL para classificar e predizer a partir dos dados da bolsa de valores, que é explicado na seção 3.4. A última etapa (Bloco E), explicada na seção 3.5, é a camada de visualização, no qual o investidor poderá interagir e visualizar as recomendações dadas pelo modelo proposto.

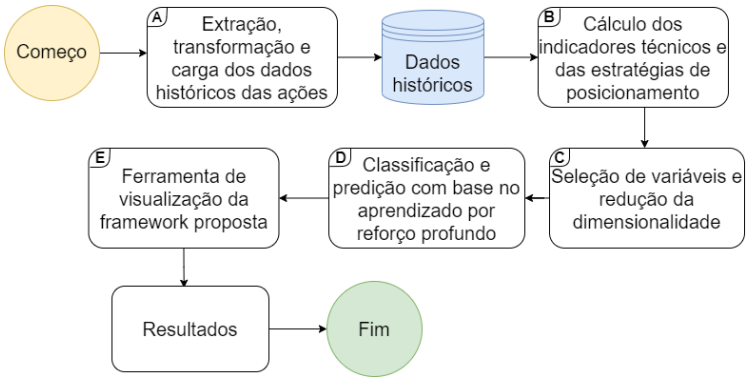

Figura 3.1: Diagrama de blocos do framework de DRL proposto.

3.1 EXTRAÇÃO, TRANSFORMAÇÃO E CARGAS DOS DADOS HISTÓRICOS DAS AÇÕES

Esta seção explica todo o processo de Extração, Transformação e Carga, do inglês *extract, transform and load* (ETL), dos dados históricos das ações. Este passo é fundamental pois, como visto na seção 2.1.2.1, o pré-processamento de dados é responsável por transformar os dados brutos de entrada em um formato apropriado para utilizar análises de AM e para remover dados inconsistentes, a fim de produzir resultados mais significativos (38). A figura 3.2 apresenta o diagrama contendo a estratégia utilizada para alcançar o objetivo de extrair, transformar e carregar os dados históricos das ações.

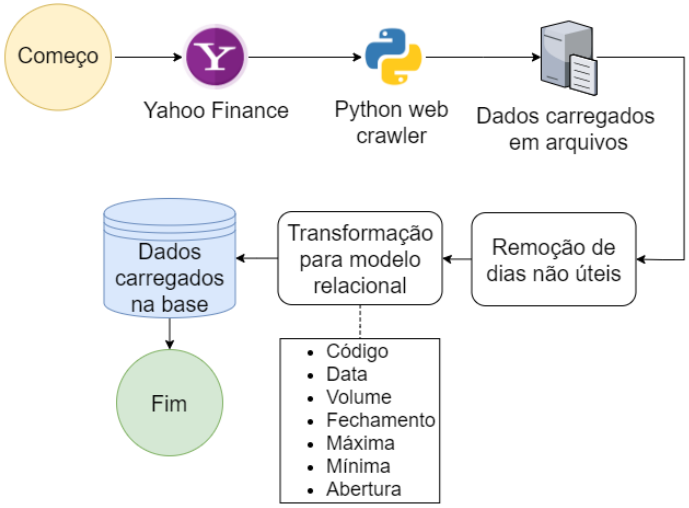

Figura 3.2: Diagrama de blocos do ETL dos dados históricos das ações

ETL é o processo de sistematização do tratamento e limpeza dos dados oriundos dos diversos sistemas organizacionais para a inserção, posteriormente, em um banco de dados ou *data warehouse* (39). Neste caso, é aplicado o ETL para tratar e transformar os dados do mercado de ações oriundos da plataforma *Yahoo Finance*. Entre as principais plataformas financeiras, o *Yahoo Finance* (40) possui dados robustos e recursos de conteúdo que podem auxiliar os usuários na tomada de decisões no mercado financeiro. Além disso, a plataforma fornece dados atuais e históricos de ações de maneira eficiente.

Para extrair os dados do site *Yahoo Finance*, foi um criado *Web Crawler* na linguagem de programação *Python*. O *Web Crawler*, também conhecido como *Bot*, é um robô usado para encontrar e recuperar informações de um site de forma autonônoma. Ele captura informações das páginas e cadastra os links encontrados, possibilitando encontrar outras páginas e mantendo sua base de dados atualizada. Logo, a criação de um *Web Crawler* permite a extração dos dados históricos das ações do *Yahoo Finance*. O *Python* foi escolhido pois, tem uma sintaxe simples, prática e uma grande comunidade que apoia e ajuda os desenvolvedores que atuam com essa linguagem.

A partir disso, foram criados arquivos com os dados das ações para mantê-los no disco local e

facilitar seu controle. Após analisar os dados obtidos, foi notado que haviam dados das ações em dias que não são realizados pregões, como sábado, domingo e feriados, por alguma incosistência da plataforma. Dessa forma, foi preciso remover essas incosistências para que não prejudicasse o desenvolvimento do modelo de AM.

A etapa subsequente foi transformar os dados não relacionais para a forma relacional, isto é, para o formato de tabelas. Este processo resultou em sete colunas de dados, que são: o código da ação, a data de operação, o volume de negociações daquele dia, o seu preço de fechamento e abertura do dia, maior e menor preço do dia. Por fim, esses dados foram carregados numa base de dados local para facilitar as buscas deles.

Como viés do estudo deste trabalho, quinhentas (500) ações dos Estados Unidos da América (EUA) foram investigadas com base em seus preços diários históricos de 2014 a 2019. Os dados abrangem 1503 dias úteis para cada símbolo de ação das quinhentas ações dos EUA investigadas. Além disso, foram utilizados os dados do mercado de ações da Tailândia de 2005 a 2006, com o objetivo de comparar, de forma justa, a performance do modelo proposto com o do estado da arte. O código para este processo de ETL, feito em *python*, se encontra no Apêndice A.

3.2 CÁLCULO DOS INDICADORES TÉCNICOS E DAS ESTRATÉGIAS DE POSICIONAMENTO

Esta seção esclarece o processo do cálculo dos indicadores técnicos e das estratégias de posicionamento, uma vez que esta etapa permite que os dados sejam rotulados e os dados de entrada sejam preparados para o modelo de AM. A Tabela 3.1 mostra os indicadores técnicos que foram calculados para servir de variáveis de entrada para o modelo de AM.

Tabela 3.1: Indicadores técnicos usados como variáveis de entrada

Grupo	Indicador técnico
Overlap	BB
Overlap	SMA
Overlap	EMA
Momentum	MACD
Momentum	RSI
Momentum	APO
Volume	OBV
Volume	AD
Volume	ADOSC
Operações Matemáticas	Mínimo
Operações Matemáticas	Máximo
Operações Matemáticas	Soma

Como os indicadores técnicos são séries de valores derivados da aplicação de uma fórmula sobre a série de preços de uma ação para identificar o movimento de alta ou de baixa de um ativo,

eles foram utilizados como variáveis de entrada para o modelo de DRL. Dessa forma, com base nesses importantes fatores técnicos, o modelo deverá aprender e reconhecer quando uma ação está em um movimento de alta ou de queda em um certo período.

Cada estratégia de posicionamento é baseada em um indicador técnico e existem três tipos de entradas ou posições: operar comprado, operar vendido e manter a posição. As operações compradas significam que o investidor está comprando uma ação, esperando que seu preço suba. A posição vendida é quando o investidor vende uma ação para recomprá-la ou cobri-la posteriormente por um preço inferior. A manutenção da posição significa que o investidor decidiu não realizar nenhuma ação de compra nem venda. E as estratégias de posicionamento levam em consideração esses três tipos de entrada, conforme explicado na seção 2.1.1.3.

Como as estratégias de posicionamento indicam qual o ideal momento e qual tipo de operação deverá ser realizada em determinada ação, as seguintes estratégias foram escolhidas para que o modelo de DRL aprenda e identifique: Estratégia RSI sobrevendido/sobrecomprado, estratégia 9.1, estratégia da MACD padronizada e estratégia do PC. Logo, a partir dos indicadores técnicos, o modelo de DRL deverá reconhecer e identificar as operações corretas com base nas estratégias de posicionamento. Além disso, foi proposto para o modelo também dar importância para o lucro e prejuízo obtidos nas operações, a fim de obter um modelo que realize operações que retornem lucro para o investidor. O código para esta etapa, desenvolvido em *python*, se encontra no Apêndice B.

3.3 SELEÇÃO DE VARIÁVEIS E REDUÇÃO DA DIMENSIONALIDADE

Esta seção explica todo o processo da seleção de variáveis (*features*) e redução da dimensionalidade dos dados de entrada obtidos na seção 3.2. Esta etapa é essencial pois, o propósito inclui remover recursos irrelevantes e redundantes, melhorar a acurácia preditiva dos algoritmos, aumentar a compreensibilidade dos modelos construídos e, por consequência, produzir resultados mais consistentes (38).

A redução da dimensionalidade pode ser vista como uma transformação de uma alta dimensão de dados para uma de baixa dimensão, com o objetivo de eliminar a redundância de dados (41), melhorar a eficiência dos modelos ao ser utilizado dados de baixa dimensão (42). O PCA é uma ferramenta poderosa na redução dimensional de dados altamente correlacionados (43). Assim, o PCA foi aplicado para reduzir a dimensionalidade nas 12 variáveis da Tabela I, mas com o objetivo de manter as mesmas informações importantes para o modelo.

Portanto, foi possível encontrar a razão de variância mais bem explicada (*best-explained variance ratio*) para essas variáveis e, como pode ser visto na Figura 3.3, os dois primeiros componentes mantiveram a variância mais importante (0.824) das 12 *features*. Este valor é também conhecido como o limiar da variância dos dados, isto é, as principais informações estão nos componentes até este limiar, a partir dele os dados dos componentes vão variar muito pouco e sua

relevância é mínima para o modelo. Dessa forma, basta a utilização desses dois componentes como atributos de entrada para o modelo pois, eles contêm as informações mais relevantes dentre todas as 12 variáveis.

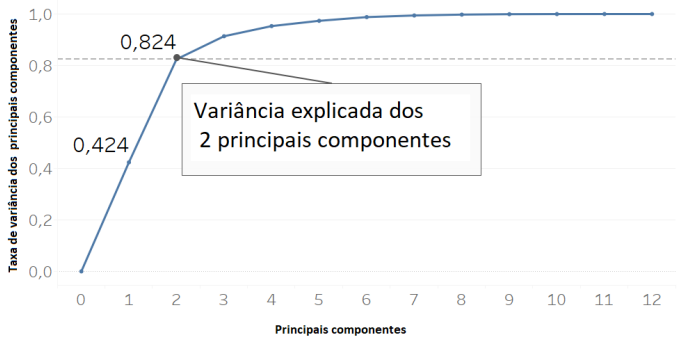

Figura 3.3: Variância explicada dos principais componentes do PCA

Dessa forma, foi aplicada uma seleção de variáveis que remove todos os atributos cuja variância não atende ao limiar da variância dos dados (0.824), conforme a Fórmula 2.25 explicada na seção 2.1.2.1. Com a aplicação desse método de seleção de váriaveis, foram selecionados 6 variáveis de 12, conforme mostrado na Tabela 3.2.

Tabela 3.2: 6 variáveis selecionadas

Grupo	Indicador técnico
Overlap	SMA
Overlap	EMA
Momentum	MACD
Momentum	RSI
Volume	OBV
Operações Matemáticas	Máximo

Logo, a seleção de *features* permitiu que fossem removidos 6 variáveis que não eram de extrema importância, reduzindo a complexidade do modelo e facilitando a sua compreensão. Da mesma forma, redução de dimensionalidade permitiu a utilização de apenas 2 atributos de entrada para o modelo, reduzindo ainda mais o número de variáveis e, por consequência, reduzindo a carga de utilização de recursos computacionais complexos. O código desenvolvido em *python* para esta etapa se encontra no Apêndice C.

3.4 CLASSIFICAÇÃO E PREDIÇÃO COM BASE NO APRENDIZADO POR REFORÇO PROFUNDO

Esta seção explica todo o algoritmo e processo de construção do modelo baseado em aprendizado por reforço profundo. Esta etapa é a que permite que o modelo aprenda as regras das

estratégias de posicionamento a partir dos dados de entrada, no caso os indicadores técnicos. A figura 3.4 apresenta o diagrama contendo a estratégia utilizada para alcançar o objetivo de construir e desenvolver o modelo de DRL para o mercado de ações.

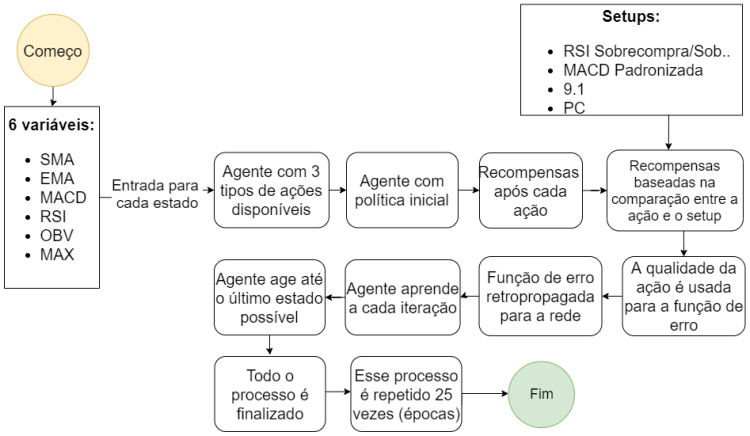

Figura 3.4: Diagrama de blocos da classificação e predição com base no DRL

A técnica de aprendizado profundo usada neste framework proposto inclui a criação de um *perceptron* de multicamadas, que considera, para cada período, os seis recursos selecionados na Tabela 3.2 como entrada e retorna os valores "Q" para cada ação como saída. Dessa forma, é retornado qual ação o agente deve considerar como a melhor naquele instante dentro do espectro de ações disponíveis (posição de compra, posição vendida e manter posição). Como pode ser vista na Figura 3.5, a rede é construída com 1 camada de entrada com 6 nós, cada um correspondente aos 6 atributos selecionados. Além disso, há três camadas ocultas e cada camada considera a função ReLU como a função de ativação.

Posteriormente, é adicionada uma camada achatada, do inglês *flatten layer*, com a finalidade de reduzir o número de dimensões e transferir os dados para a camada de saída. Uma vez que existem três ações possíveis, a camada de saída contém três neurônios usando a função de ativação linear. Assim, a rede neural visa estimar os valores "Q" para cada ação com base em uma determinada situação e é treinado com base na equação de atualização do *Q-learning*, conforme explicado na seção 2.1.2.5. O *Q-learning* permite que a rede extraia as principais características de cada circunstância para prever a recompensa e a melhor ação, mesmo em situações não reveladas anteriormente.

Para ajustar e escolher os melhores hiperparamêtros da rede neural, foi utilizado o algoritmo *Grid Search*. O *Grid Search* é uma técnica de ajuste que tenta calcular os valores ideais de hiperparâmetros. É uma pesquisa exaustiva realizada sobre os valores de parâmetros específicos de um modelo (44). Dessa forma, com este algoritmo foi possível melhorar a performance da rede neural após ser selecionados os melhores hiperparâmetros para cada camada.

No início, temos os seis atributos selecionados, conforme descrito na seção 3.2. Como explicado anteriormente, esses recursos são as entradas para o modelo e, para cada estado, existem três tipos de ações para o agente de DRL realizar. Assim, o agente começa com uma política de

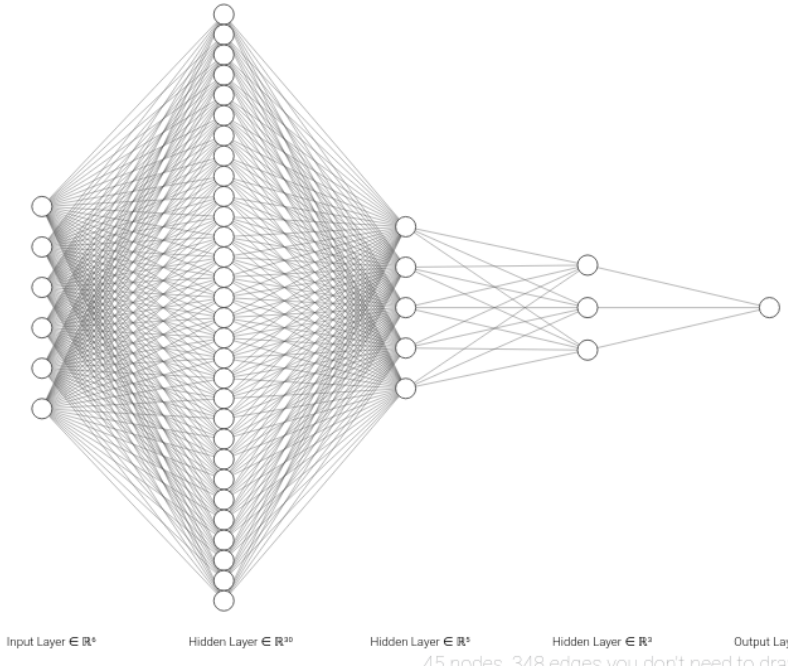

Figura 3.5: Estrutura da rede neural multicamadas utilizada

comportamento inicial de qual ação deve ser executada para dar a recompensa máxima entre as três ações possíveis em um determinado estado. Dessa forma, o agente observa um estado do ambiente, prevê a melhor ação recompensada com base nesta política e a executa. Após a ação ser realizada, o agente recebe a recompensa de acordo com a ação desejada, este que é definada de acordo com as 4 estratégias de posicionamento selecionadas na seção 3.2.

A recompensa é positiva (+1) quando a ação prevista coincide com a posição da estratégia, o que significa que o agente fez o movimento certo e merece uma recompensa por isso. A recompensa é neutra (0) quando a ação executada é manter a posição e a posição estratégica indica que é um momento de operação comprada ou vendida. Nessa situação, o agente não realiza nenhuma ação de comprar ou vender e, por isso, não pode gerar perdas ou lucros. Portanto, a recompensa é neutra. A recompensa é negativa (-1) quando a ação realizada é uma operação comprada ou vendida e a estratégia de posicionamento indica o inverso, o que significa que o agente realizou a ação errada e merece uma recompensa negativa. A recompensa é, também negativa, quando o agente fecha uma posição, ou seja, executa sua posição, e todo esse processo gerou prejuízo ao invés de lucro.

A qualidade de cada ação realizada pelo agente em um determinado estado é usada para atualizar o comportamento da política do agente com o cálculo e a retropropagação da função de perda baseada em *Q-learning* para o DRL, conforme explicado na seção 2.1.2.5. Dessa forma, o agente aprende cada vez mais após cada interação e interage com o ambiente com o objetivo de maximizar a recompensa futura cumulativa. Todo esse processo descrito se repete até o último estado, terminando uma época. Os resultados experimentais, explicitados no próximo capítulo, mostraram que o agente precisa de 25 épocas para atingir a maior recompensa cumulativa, a fim

de identificar os padrões de negociação e dar recomendações sobre qualquer ação. O código feito em *python* para esta etapa se encontra no Apêndice D.

3.5 FERRAMENTA DE VISUALIZAÇÃO DA *FRAMEWORK* PROPOSTA

Esta seção explica todo o processo de concepção e estrutura da ferramenta de visualização da *framework* proposta. Esta etapa permite que a interação entre o usuário final, neste caso investidores, e o modelo de DRL de forma prática e intuitiva através da plataforma criada. A figura 3.6 demonstra a interface da plataforma desenvolvida, no qual, neste exemplo, está sendo analisada a ação da empresa *Twitter*, de código de negociação TWTR.

Figura 3.6: Demonstração da ferramenta de visualização da *framework* proposta

O desenvolvimento de uma ferramenta de visualização, ou interface, é a prática responsável pelo planejamento, desenvolvimento e aplicação de uma solução com o objetivo de facilitar a experiência do usuário e estimular sua interação com alguma plataforma (45). O desenvolvimento de uma interface tem o papel fundamental de oferecer soluções amigáveis e intuitivas para o usuário final. Este desenvolvimento diz respeito à parte visual, à usabilidade, arquitetura da informação, navegação e transição de telas.

Portanto, uma ferramenta de visualização é uma peça fundamental para o entendimento das análises por parte do usuário final, que neste caso são os investidores. Por isso, é necessário desenvolver uma interface nos padrões recomendados por especialistas desta área (45). Uma das principais recomendações é a criação de uma aplicação de página simples, do inglês *Single-Page Application* (SPA), pois, consiste na criação de um *site* que interage com o usuário atualizando dinamicamente a página com novos dados do servidor, ao invés do método tradicional em que o navegador carrega novas páginas inteiras. Este método padrão faz com que o usuário final espere por uma quantidade de tempo maior para carregar a página do que o método SPA, o que piora significativamente sua experiência com a plataforma.

Pelo fato do método SPA ter seu desenvolvimento simplificado e otimizado, este recurso é utilizado por várias plataformas mundialmente utilizadas, como *Facebook*, *Gmail*, *Google Maps* entre outros (46). Por esses motivos, foi utilizado este tipo de arquitetura de interface, no qual foi escolhida a framework *React* (47) para desenvolvê-la. O *React* é desenvolvido e mantido pela *Facebook*, no qual se destaca por ter um modelo de objeto de documento, do inglês *document object model* (DOM), virtual, tornando a experiência do usuário melhor e agilizando o trabalho dos desenvolvedores.

Como o React é apropriado para a construção de uma plataforma de visualização para o usuário final, foi necessário o desenvolvimento de um servidor que enviassem os dados do modelo de AM para ele. Como o modelo foi desenvolvido em *Python* e como arquiteturas SPA se comunicam por meio do protocolo *Representational state transfer (REST)*, foi escolhida a framework *Flask* (48), que utiliza a linguagem *Python* e tem bibliotecas de comunicação via REST já implementadas.

Flask é uma das frameworks mais utilizadas por se prático e ágil o seu desenvolvimento (49). Dessa forma, a ferramenta de visualização da *framework* proposta teve sua camada de *front-end* desenvolvida em *React* e a camada de *back-end* em *Flask*, no qual a comunicação entre eles foi através do REST. Essa ferramenta foi desenvolvida com toda essa estrutura com o objetivo de ser intuitiva, leve, rápida e, principalmente, para ser útil como um sistema de apoio a tomada de decisões por parte dos investidores.

4 RESULTADOS

Este capítulo visa apresentar a validação dos resultados obtidos pela *framework* de DRL proposta e a comparação deles com o estado da arte. Além disso, é feita a visualização desses resultados de forma gráfica, do retorno sobre o investimento ao se comparar o modelo com índices do mercado e, por fim, é apresentado todos os módulos disponíveis na ferramenta de visualização.

4.1 AVALIAÇÃO DA PERFORMANCE DOS ALGORITMOS

As performances do modelo baseado em aprendizado por reforço profundo, em cada uma das estratégias de posicionamento, foram avaliadas por meio das métricas de precisão, revocação e medida de F1-Score, equações de 2.44 a 2.46, respectivamente. Isso foi feito a partir da matriz de confusão gerada por cada estratégia, após a aplicação de conjunto de testes nos classificadores gerados por eles.

Para validação do modelo, foi utilizado o método de validação cruzada *k-fold* para séries temporais (50), do inglês *Time-Series K-fold cross validation*, o que dá uma melhor indicação de quão bem o modelo irá executar em dados não vistos e de acordo com (51), pode de fato ser usado ao invez do método padrão *holdout*. Neste método padrão, o conjunto de dados em dois subconjuntos mutuamente exclusivos, um para treinamento e outro para teste (validação).

Já no método *k-fold*, é divido o conjunto total de dados em k subconjuntos mutuamente exclusivos do mesmo tamanho e, a partir daí, um subconjunto é utilizado para teste e os k-1 restantes são utilizados para estimação dos parâmetros, fazendo-se o cálculo da acurácia do modelo. Este processo é realizado k vezes alternando de forma circular o subconjunto de teste. No *k-fold* de séries temporais, os subconjuntos são testados de forma sequencial para obedecer a ordem temporal. A Figura 4.1 mostra um exemplo do método de validação cruzada *k-fold* para séries temporais.

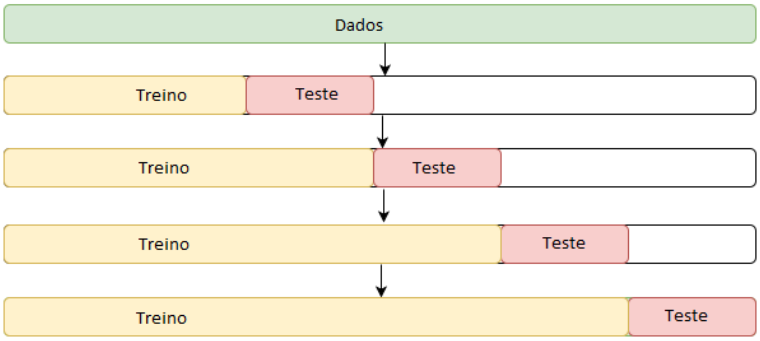

Figura 4.1: Validação cruzada *k-fold* para séries temporais

Dessa forma, este método se destaca entre outros métodos pois, nele é possível validar um conjunto de dados específico ou que tem certas particularidades de forma contínua, simulando o que acontece com dados reais (51). Neste trabalho, todo o conjunto de dados foi dividido em dez subconjuntos para treinar o algoritmo DRL, onde cada vez são selecionados três subconjuntos subsequentes para treinamento e o próximo subconjunto para teste.

O objetivo de um modelo de DRL, é maximizar a sua recompensa durante uma sequência de ações. Neste sentido, a meta é alcançar um número de épocas ideal que maximize a recompensa do modelo, tornando-o um modelo de boa performance e alta acurácia. Conforme mencionado na seção 3.4, na Figura 4.2 é possível ver que a recompensa chega ao seu ideal em torno de 25 épocas. Isto é, por volta de 25 épocas, a recompensa obtida passa dos 90 % (1082) da recompensa máxima e chega próximo a recompensa total de 100 % (1203).

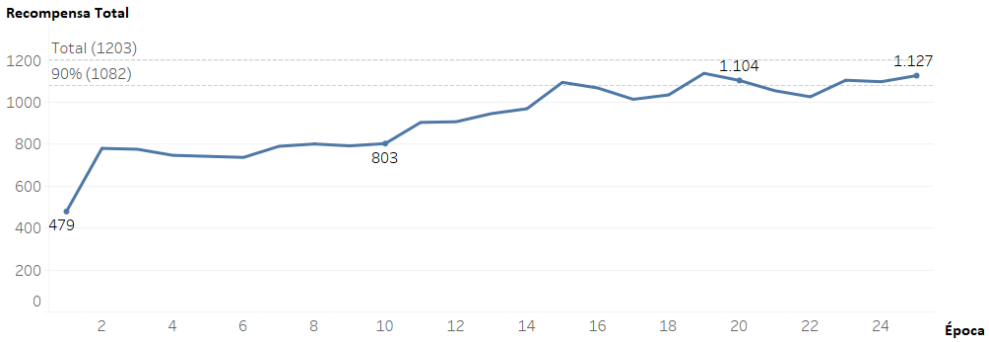

Figura 4.2: Recompensa do modelo ao longo de 25 épocas

Após 25 épocas, como pode ser visto na Figura 4.3, a recompensa pouco varia e essa variação não afeta o desempenho do modelo. Ademais, o tempo de processamento e os recursos computacionais aumentam significativamente, o que leva a conclusão de que 25 épocas é o número ideal para este modelo.

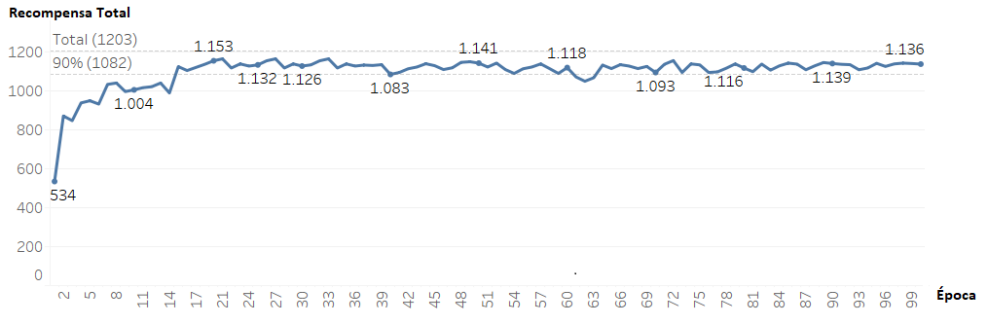

Figura 4.3: Recompensa do modelo ao longo de 100 épocas

O resultado da performance do modelo ao ser aplicado nas 4 estratégias de posicionamento, pode ser visto na tabela 4.1. De acordo com essa tabela, a estratégia RSI tem a melhor performance de 0.88 de medida de *F1-Score* para operações compradas, 0.86 para operações vendidas e 0.89 para operações em que se tinha que manter a posição. O resultado do MACD foi bastante similar ao do RSI, ficando com a segunda melhor performance seguida da estratégia 9.1 e PC,

respectivamente.

As performances do modelo, em geral, foram bastante significativas e se mostraram muito eficientes em dados de ações reais, pelo fato de todos terem uma medida maior que 80 % para todas as métricas, principalmente a métrica *F1-Score*.

Tabela 4.1: Performance do modelo ao ser aplicado nas 4 estratégias de posicionamento

Estratégia	Operação	Precisão	Revocação	F1-Score
RSI Sobrecomprado/Sobrevendido	comprada	0,90	0,83	0,86
RSI Sobrecomprado/Sobrevendido	vendida	0,87	0,88	0,88
RSI Sobrecomprado/Sobrevendido	mantida	0,88	0,9	0,89
Padronização MACD	comprada	0,86	0,84	0,85
Padronização MACD	vendida	0,87	0,86	0,86
Padronização MACD	mantida	0,86	0,84	0,85
9.1	comprada	0,84	0,86	0,85
9.1	vendida	0,82	0,85	0,83
9.1	mantida	0,83	0,84	0,84
PC	comprada	0,8	0,84	0,82
PC	vendida	0,83	0,84	0,83
PC	mantida	0,82	0,84	0,83

A Tabela 4.2 compara o melhor resultado de operações, seja comprada, vendida ou mantida, do modelo proposto com o estado da arte (2). A efeito de comparação, o modelo proposto foi testado com a mesma base de dados, no qual é utilizado um fundo de índice comercializado como ação, do inglês *Exchange-traded fund* (ETF). Logo, foi utilizado o ETF *XLF: SPDR Consumer Staples* da bolsa de valores dos EUA entre 31/10/2007 a 20/12/2010.

Tabela 4.2: Comparação entre a performance do modelo proposto e o estado da arte (2)

Estratégia	Operação	Precisão	Revocação	F1-Score
Modelo proposto	comprada	0,9	0,84	0,87
Modelo proposto	vendida	0,86	0,87	0,86
Modelo proposto	mantida	0,86	0,88	0,87
Estado da arte (2)	comprada	0,75	0,83	0,78
Estado da arte (2)	vendida	0,88	0,45	0,6
Estado da arte (2)	mantida	0,81	0,63	0,71

Nessa comparação, é possível notar que houve uma melhora significativa nas operações comprada, vendida e mantida, no qual o modelo proposto novamente superou o estado da arte em todas as métricas. Principalmente na métrica F1-Score, onde no estado da arte foi obtido um valor de 0,82 para operações de compra, enquanto que no modelo proposto foi obtido o valor de 0,87. Nas operações vendidas, o estado da arte obteve um F1-Score de 0,79 e o modelo proposto obteve um 0,86. Nas operações de manter a posição, o estado da arte conseguiu um valor de 0,87 para o F1-Score e o modelo um valor de 0,71. Portanto, o modelo proposto se adequou a dados em que não haviam sido revelados anteriormente, isto é, não foram utilizados estes dados para o seu treino, e mostrou uma performance significativamente maior que o estado da arte (1).

Da mesma forma, a Tabela 4.3 compara o melhor resultado de operações, seja comprada, vendida ou mantida, do modelo proposto com o estado da arte (1). Para uma comparação justa, o modelo proposto foi testado com a mesma base de dados, no qual é utilizado dados das ações do mercado de ações da Tailândia, do inglês *stock exchange of Thailand* (SET) de 01/03/2005 a 03/12/2006.

Tabela 4.3: Comparação entre a performance do modelo proposto e o estado da arte (1)

Estratégia	Operação	Precisão	Revocação	F1-Score
Modelo proposto	comprada	0,91	0,89	0,9
Modelo proposto	vendida	0,85	0,88	0,86
Modelo proposto	mantida	0,89	0,87	0,88
Estado da arte (1)	comprada	0,81	0,83	0,82
Estado da arte (1)	vendida	0,83	0,76	0,79
Estado da arte (1)	mantida	0,71	0,49	0,58

Nessa comparação, é possível notar, também, que houve uma melhora significativa nas operações comprada, vendida e mantida, no qual o modelo proposto novamente superou o estado da arte em todas as métricas. Especialmente na métrica F1-Score, onde no estado da arte foi obtido um valor de 0,82 para operações de compra, enquanto que no modelo proposto foi obtido o valor de 0,9. Nas operações vendidas, o estado da arte obteve um F1-Score de 0,79 e o modelo proposto obteve um 0,86. Dessa forma, foram utilizados dados que o modelo proposto ainda não havia conhecido (treinado). E, mesmo assim, mostrou uma performance, de modo considerável, maior que o estado da arte (1).

4.2 VISUALIZAÇÃO DOS RESULTADOS DE FORMA GRÁFICA

A visualização de dados é a exibição de informações em um formato gráfico ou tabular. Para ela ser bem-sucedida, é necessário que as informações sejam convertidas em um formato visual para que as características dos dados e as relações entre eles possam ser analisadas ou relatadas. Portanto, uma forma bastante utilizada há muito tempo, por ser muito eficiente, é a visualização de forma gráfica (12). Neste sentido, foram feitos gráficos dos resultados de cada uma das estratégias de posicionamento para uma melhor análise e entendimento da performance do modelo.

A Figura 4.4 mostra um gráfico de linha com série temporal, onde a linha em azul é cada operação de acordo com a estratégia de posicionamento RSI e a linha hachurada é a ação prevista e realizada pelo modelo. O valor “1” no eixo y significa que é uma operação comprada, já o valor “0” significa manter a posição e “-1” uma operação vendida. É possível notar uma ligeira diferença entre as ações definidas pelo setup (linha azul) através do eixo temporal e as ações previstas pelo algoritmo (linha laranja pontilhada). Dessa forma, o algoritmo está agindo corretamente muitas vezes e, em poucas situações, cometendo alguns erros neste período, o que ratifica a performance apresentada na Tabela 4.1.

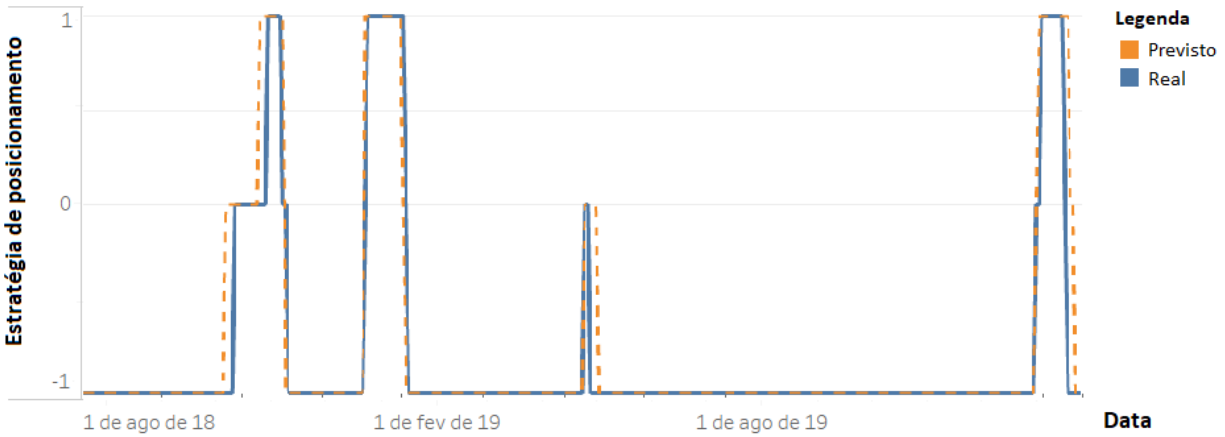

Figura 4.4: Ações previstas do modelo em constrate com a estratégia de posicionamento RSI

Da mesma forma, foram feitos gráficos para as outras três estratégias de posicionamento, no qual a Figura 4.5 compara a performance do modelo com a estratégia de padronização MACD, a Figura 4.6 com a estratégia 9.1 e, por fim, a Figura 4.7 com a estratégia PC.

Neste contexto, é possível notar que, conforme mostrado na Tabela 4.1, o gráfico com a estratégia MACD Padronizada teve mais diferenças entre as ações preditas e as corretas quando se comparado com o gráfico da estratégia do RSI. E assim por diante para o restante das estratégias quando se comparadas entre si, até chegar na estratégia de pior perfomance, que é o setup PC.

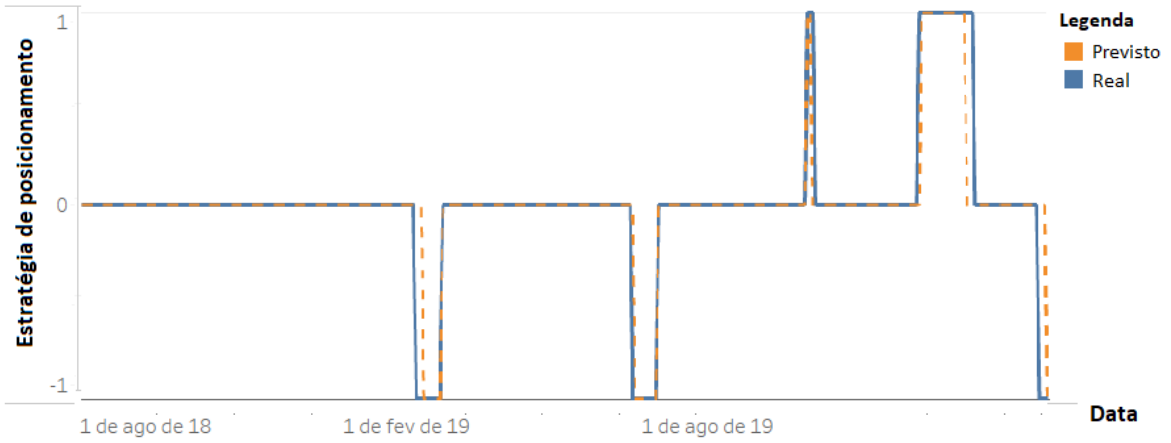

Figura 4.5: Ações previstas do modelo em constrate com a estratégia de posicionamento MACD Padronizada

4.3 RETORNO SOBRE O INVESTIMENTO

O retorno sobre o investimento é algo essencial para o mercado de ações e seus investidores pois, é justamente o quanto ele obteve de retribuição (lucro ou prejuízo) após investir em determinada ação. Logo, ao investir em um ativo, o investidor tem um percentual de lucro ou prejuízo sobre o que foi investido. Neste contexto, em um modelo de AM desenvolvido para a

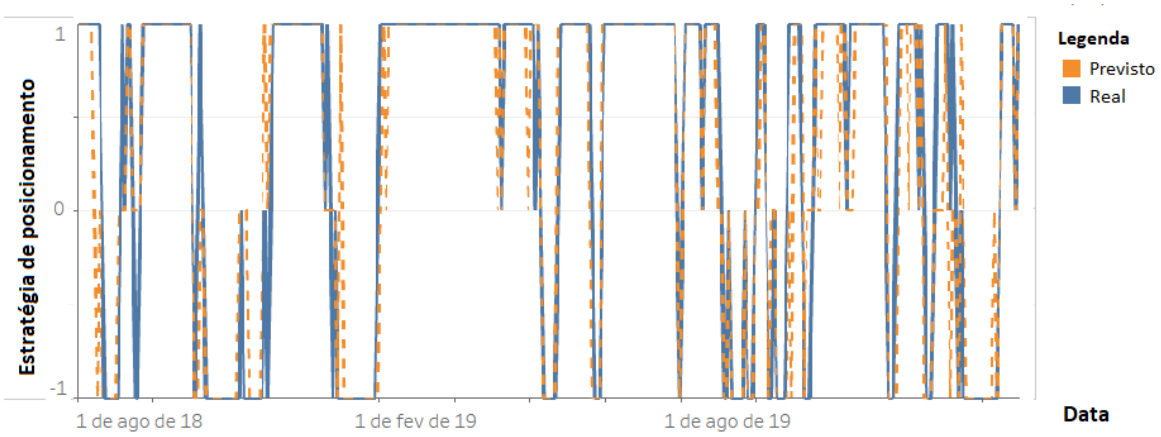

Figura 4.6: Ações previstas do modelo em constrate com a estratégia de posicionamento 9.1

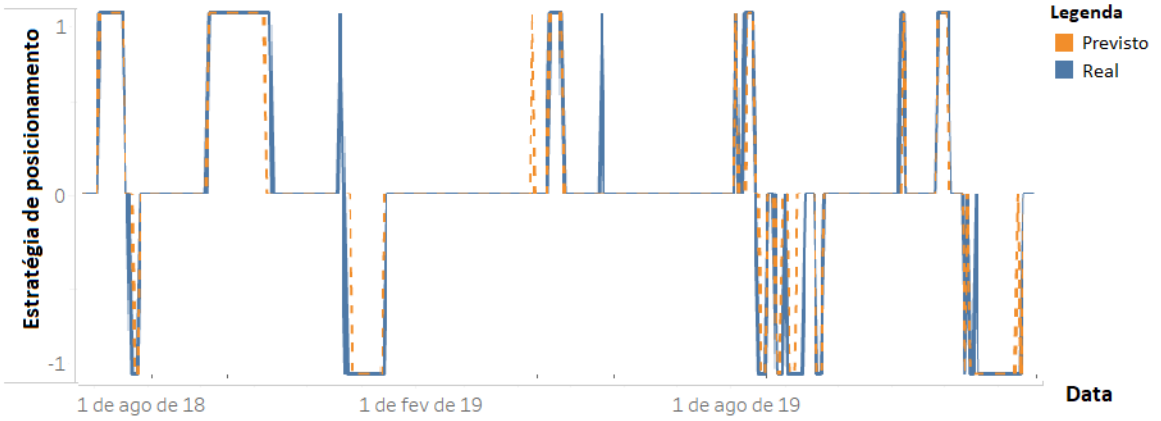

Figura 4.7: Ações previstas do modelo em constrate com a estratégia de posicionamento PC

bolsa de valores é imprescindível que ele obtenha lucros e que, no mínimo, não gere perdas para o investidor.

Sendo assim, é fundamental avaliar a performance do retorno sobre o investimento do modelo proposto e compará-lo com os índices e ativos do próprio mercado de ações, além da perfomance do estado da arte. Na Tabela 4.4, é demonstrado a comparação da performance do retorno sobre o investimento do modelo com o estado da arte (1). Para comparar esses dados de forma justa, foi utilizado a mesma base de dados e o mesmo período de (1), conforme explicado na seção 4.1.

Nesta tabela, é possível notar um aumento no retorno sobre o investimento, neste caso lucro, de 11,73 %. O que é um aumento expressivo pois, dependendo da quantidade monetária que o indivíduo tenha investido, pode haver uma grande diferença entre os dois retornos.

Tabela 4.4: Comparação entre o retorno sobre investimento do modelo proposto e o estado da arte (1)

Algoritmo	Porcentagem de retorno (31/10/2007 a 20/12/2010)
Modelo proposto	69.4 %
Estado da arte (1)	57.67 %

Além da comparação com o estado da arte, foi realizada a comparação do retorno sobre o investimento do modelo proposto com 4 ações dos EUA, que estão no índice S&P500. O S&P500 é um dos um índices mais famosos pois, é através dele que é medido o desempenho das ações das 500 maiores empresas listadas na bolsas de valores dos EUA. Logo, as ações escolhidas estão entre as mais relevantes de todo o país e do mundo pois, a bolsa de valores dos EUA (NYSE) é a maior do planeta (16). E, também, foi comparada a performance do modelo com o próprio índice S&P500. A Figura 4.8 mostra um gráfico de linha temporal, no qual o período analisado é do começo do ano de 2014 ao final de 2019, totalizando 5 anos. Esse gráfico demonstra o lucro diário obtido por 4 ações americanas, além do S&P500. Ações essas que são: Google de código de negociação GOOGL, Twitter de código TWTR, Disney (DIS) e Ford (F).

A legenda demonstra qual cor corresponde ao código de ativação entre os 5 ativos descritos acima, sendo laranja a curva de retorno do modelo proposto. Na figura, nota-se que a ação da Google foi a que mais trouxe lucro durante este período com 199 %. Em seguida, vem o modelo proposto com 178,9 % de lucro e bem próximo a ele, a ação da Disney com 165,5 %. Após isso, vem o S&P500, Twitter e Ford com, respectivamente, 50,6 %, 33,2 %, -61,1 % de retorno sobre o investimento. A ação da Ford foi a única que retornou prejuízo ao final de todo o período.

A partir destes dados, conclui-se que o modelo, além de ter uma boa performance em relação a classificação e predição, também apresenta resultados expressivos quando o assunto é retorno sobre o investimento em dados reais do mercado financeiro. Isto porque o lucro obtido pelo modelo ficou abaixo apenas do retorno da Google, que também foi altamente expressivo com a multiplicação em quase duas vezes do capital investido nela.

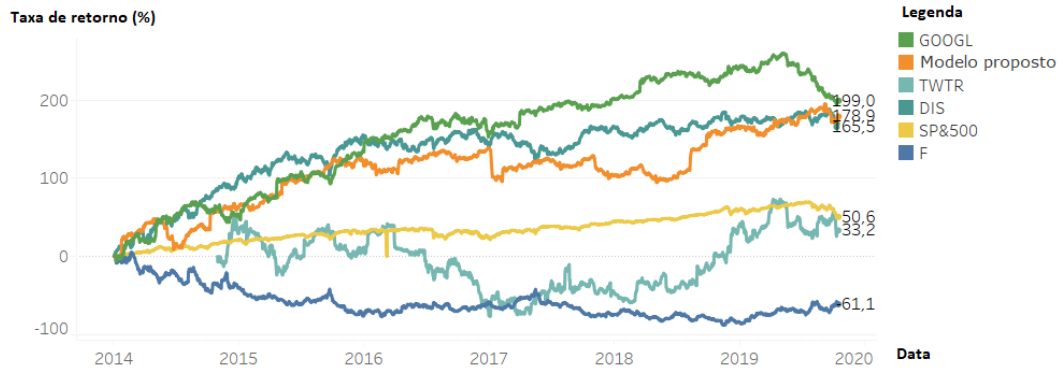

Figura 4.8: Retorno sobre o investimento diário do modelo proposto e de 5 ativos dos EUA

4.4 MÓDULOS DA FERRAMENTA DE VISUALIZAÇÃO

A ferramenta de visualização foi criada para permitir que haja interação entre os investidores e o modelo de DRL, de forma prática e intuitiva. Esta ferramenta contém 2 módulos principais: módulo de aprendizado e detecção de padrões; módulo das predições e apoio à tomada de decisões. Todos os módulos possuem o gráfico de *candlestick* como elemento gráfico principal e

filtro seletor de ações, para que o usuário possa escolher qual ativo quer analisar.

A Figura 4.9 mostra um exemplo do módulo de aprendizado e detecção de padrões. No gráfico de *candlestick*, é possível notar balões azuis e laranjas, que significam uma detecção de um padrão de mercado. No qual, azul são padrões de alta e laranja, padrões de baixa. O padrão que o usuário deseja que a ferramenta detecte é definido, no canto direito da tela, no seletor embaixo do nome "Setup". O que deixa a ferramenta bem customizável e auxilia o investidor a identificar estes padrões.

Logo abaixo, há uma descrição do padrão escolhido e como ele ocorre, para que o usuário aprenda ou entenda melhor como encontrá-lo. Da mesma forma, é possível detectar padrões de *candles*, clicando em "Candlestick", onde o usuário também poderá selecionar qual padrão desejado para a *framework* detectar.

Figura 4.9: Módulo de detecção de padrões

A Figura 4.10 mostra um exemplo do módulo de predição e apoio à tomada de decisão. No gráfico de *candlestick*, é possível notar balões azuis e laranjas, que significam uma operação comprada ou vendida sugerida pelo modelo. No qual, azul são operações de compra e laranja, operações vendidas.

No canto direito, é possível visualizar a performance do retorno sobre investimento do agente sobre essa ação, que no exemplo é uma ação do Twitter. Percebe-se que, para este exemplo, o retorno diário acumulado do agente desde o início dessa ação é de aproximadamente 180 %. O que é 97.49 % maior que a valorização deste ativo e 124.44 % maior que a performance do índice S&P500 neste período.

Logo abaixo, ainda no canto direito, é exposto a recomendação do modelo de qual ação deve ser tomada naquele momento. E, para este exemplo, a ação que o modelo recomenda é de operação de compra, isto é, o modelo acredita, com base em seus treinos e experiências, que essa ação sofrerá uma valorização, fazendo com que seu preço suba e o investidor tenha lucro ao vender posteriormente esta ação.

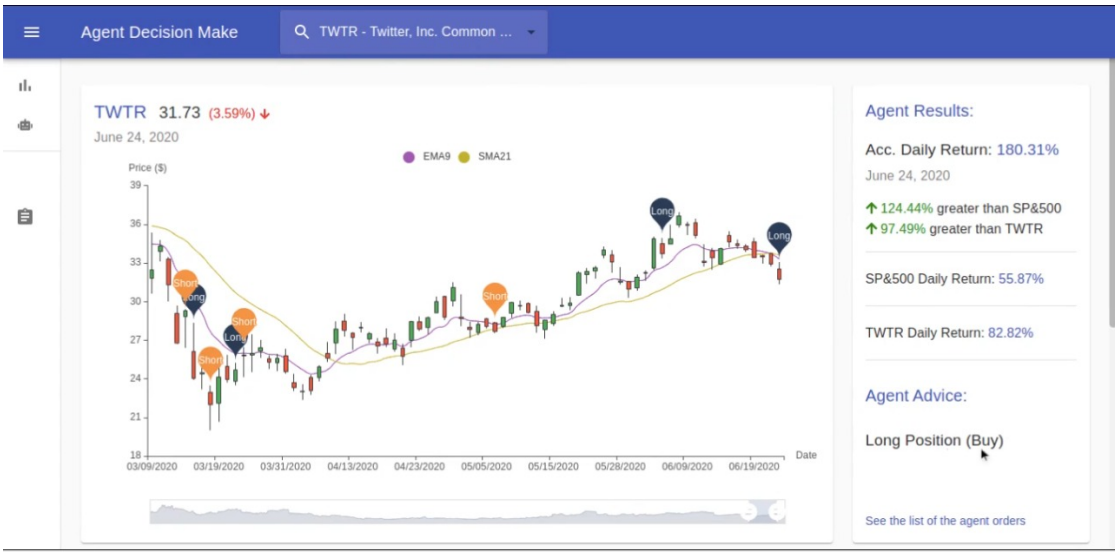

Figura 4.10: Módulo de predição e apoio à tomada de decisões

5 CONCLUSÃO E TRABALHOS FUTUROS

Esse trabalho consistiu na construção de uma *framework* de apoio à tomada de decisão para o mercado de ações através da aplição de técnicas de aprendizado por reforço profundo. Em meio a um mercado tão complexo como a bolsa de valores, este trabalho foi desenvolvido com o intuito de se obter um melhor entendimento sobre o mercado ações e gerar análises significativas para o ramo acadêmico.

A eficácia da ferramenta proposta foi validada nos mercados de ações dos EUA e da Tailândia. O modelo de DRL obtido supera a *framework* do estado da arte em termos de avaliação da estratégia de posicionamento, com valores de 0,86 % na métrica F1-Score para operações compradas e 0,88 % nas operações vendidas. A melhor abordagem do estado da arte apresenta um valor de F1-Score de 0,82 % (2) para operações compradas e 0,79 % para operações vendidas. Além disso, o modelo proposto supera a taxa de retorno sobre investimento mais alta do estado da arte com 69,4 % contra 57,67% em (1).

Em constrate com o estado da arte que utilizou somente técnicas de DL, a *framework* desenvolvida propôs o uso do DRL. O uso deste aprendizado permite que o modelo seja atualizado de forma constante pois, o seu objetivo é escolher uma sequência de ações para maximizar as recompensas de longo prazo.

Por essa razão, o DRL é naturalmente adequado para aprender regras de negociação, e sua combinação com DL permite resolver problemas de previsão de memória longa e curta simultaneamente. Portanto, os resultados demonstram um potencial significativo da *framework* de DRL proposta no mercado de ações, uma vez que supera consideravelmente as abordagens de AM do estado da arte (1) e (2).

A principal deficiência da ferramenta proposta é não permitir que as negociações de ações sejam feitas de forma automatizada, isto é, essa ferramenta somente propõe o que deveria ser feito em determinado momento. Logo, um investidor que não tenha tempo hábil para investir no mercado de ações ou que não tenha confiança suficiente para investir, não seria beneficiado e ajudado pela plataforma.

5.1 TRABALHOS FUTUROS

Como perspectiva de trabalho futuro e, a partir da principal deficiência do trabalho, é sugerida a integração dessa *framework* com o módulo de *Home Broker*. O *Home Broker* é um sistema que permite a negociação de ações e outros ativos financeiros por meio da internet, garantindo um processo rápido e automatizado para o investidor. A integração da *framework* com este módulo, permitiria que o modelo operasse de forma autônoma para qualquer investidor que tivesse inte-

resse.

Dessa forma, essa *framework* teria um módulo que opera na bolsa de valores, com base nos treinamentos e experiências do modelo que atualizam seu conhecimento diariamente, para obter de forma autonôma lucros aos investidores que têm dificuldade para operar ou, principalmente, que não têm tempo hábil para tal.

REFERÊNCIAS BIBLIOGRÁFICAS

1 Boonpeng, S.; Jeatrakul, P. Decision support system for investing in stock market by using oaa-neural network. In: *2016 Eighth International Conference on Advanced Computational Intelligence*. [S.l.: s.n.], 2016. p. 1–6.

2 Liu, C.; Malik, H. A new investment strategy based on data mining and neural networks. In: *2014 International Joint Conference on Neural Networks*. [S.l.: s.n.], 2014. p. 3094–3099.

3 Iacomin, R. Stock market prediction. In: *2015 19th International Conference on System Theory, Control and Computing*. [S.l.: s.n.], 2015. p. 200–205.

4 Umm-e-Laila; Bukhari, S. F. A.; Raees, A.; Umm-e-Farwa. Investment supervision in stock market. In: *2010 International Conference on Computer Applications and Industrial Electronics*. [S.l.: s.n.], 2010. p. 294–296.

5 Srivastava, R. P. Decision table based analysis of trading models. In: *Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*. [S.l.: s.n.], 2006. p. 3–8.

6 RAMALINGAM S.; POTHULA, S. Decision table based analysis of trading models. In: *Survey on Machine Learning Techniques Used for Stock Market Prediction*. [S.l.: s.n.], 2018. v. 4, n. 1.

7 Idrees, S. M.; Alam, M. A.; Agarwal, P. A prediction approach for stock market volatility based on time series data. *IEEE Access*, v. 7, p. 17287–17298, 2019.

8 Ray, R.; Khandelwal, P.; Baranidharan, B. A survey on stock market prediction using artificial intelligence techniques. In: *2018 International Conference on Smart Systems and Inventive Technology*. [S.l.: s.n.], 2018. p. 594–598.

9 Sable, R.; Goel, S.; Chatterjee, P. Empirical study on stock market prediction using machine learning. In: *2019 International Conference on Advances in Computing, Communication and Control*. [S.l.: s.n.], 2019. p. 1–5.

10 Parmar, I.; Agarwal, N.; Saxena, S.; Arora, R.; Gupta, S.; Dhiman, H.; Chouhan, L. Stock market prediction using machine learning. In: *2018 First International Conference on Secure Cyber Computing and Communication*. [S.l.: s.n.], 2018. p. 574–576.

11 Dang, M.; Duong, D. Improvement methods for stock market prediction using financial news articles. In: *2016 3rd National Foundation for Science and Technology Development Conference on Information and Computer Science*. [S.l.: s.n.], 2016. p. 125–129.

12 AlArmouty, B.; Fraihat, S. Data analytics and business intelligence framework for stock market trading. In: *2019 2nd International Conference on new Trends in Computing Sciences*. [S.l.: s.n.], 2019. p. 1–6.

13 LEVINSON, M. *Guide to Financial Markets*. [S.l.]: The Economist, 2014.

14 GRAHAM, B. *The Intelligent Investor*. [S.l.]: Harper, 1949.

15 FABOZZI, F. J. *Fixed Income Securities*. [S.l.]: John Willey Sons, Inc., 1983.

16 SIEGEL, J. *Stocks for the Long Run*. [S.l.]: McGraw–Hill Education, 1994. v. 4.

17 Teixeira, L. A.; de Oliveira, A. L. I. Predicting stock trends through technical analysis and nearest neighbor classification. In: *2009 IEEE International Conference on Systems, Man and Cybernetics*. [S.l.: s.n.], 2009. p. 3094–3099.

18 PALEX. *Fundamentos de Análise Técnica de Ações*. [S.l.]: Alexandre Fernandes, 2014. v. 1.

19 Guo, S.; Hsu, F.; Hung, C. Deep candlestick predictor: A framework toward forecasting the price movement from candlestick charts. In: *2018 9th International Symposium on Parallel Architectures, Algorithms and Programming*. [S.l.: s.n.], 2018. p. 219–226.

20 Oriani, F. B.; Coelho, G. P. Evaluating the impact of technical indicators on stock forecasting. In: *2016 IEEE Symposium Series on Computational Intelligence*. [S.l.: s.n.], 2016. p. 1–8.

21 ANGRA, S.; AHUJA, S. Machine learning and its applications - a review. *IEEE International Conference on Big Data Analytics and Computational Intelligence*, 2017.

22 ALPAYDıN, E. Introduction to machine learning. *The MIT Press*, v. 3, 2014.

23 TAN, P. N.; STEINBACH, M.; KUMAR, V. *Introduction to data mining*. [S.l.]: Addison-Wesley Longman Publishing Co. Inc, Boston, MA, 2005. v. 1.

24 ANTONY, P. J.; MANUJESH, P.; JNANESH, N. A. Data mining and machine learning approaches on engineering materials — a review. *IEEE International Conference on Recent Trends in Electronics, Information Communication Technology*, 2016.

25 Liu, H.; Dougherty, E. R.; Dy, J. G.; Torkkola, K.; Tuv, E.; Peng, H.; Ding, C.; Long, F.; Berens, M.; Parsons, L.; Yu, L.; Zhao, Z.; Forman, G. Evolving feature selection. *IEEE Intelligent Systems*, v. 20, n. 6, p. 64–76, 2005.

26 SINGH, A.; THAKUR, N.; SHARMA, A. A review of supervised machine learning algorithms. *IEEE International Conference on Computing for Sustainable Global Development*, 2016.

27 LI, X.; YE, N. A supervised clustering and classification algorithm for mining data with mixed variables. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, v. 36, n. 2, p. 396–406, 2006.

28 Uhrig, R. E. Introduction to artificial neural networks. In: *Proceedings of IECON '95 - 21st Annual Conference on IEEE Industrial Electronics*. [S.l.: s.n.], 1995. v. 1, p. 33–37 vol.1.

29 HAYKIN, S. S. *Neural Networks: a comprehensive foundation*. [S.l.]: Prentice Hall, 1994. v. 2.

30 MITCHELL, T. *Machine Learning*. [S.l.]: McGraw-Hill Science, 1997. v. 1.

31 NORVIG, P.; RUSSELL, S. J. *Artificial Intelligence: A Modern Approach*. [S.l.]: Prentice Hall, 1994. v. 3.

32 SUTTON, R. S.; G., A. *Reinforcement Learning: An Introduction*. [S.l.]: The MIT Press, 2015. v. 2.

33 Jang, B.; Kim, M.; Harerimana, G.; Kim, J. W. Q-learning algorithms: A comprehensive classification and applications. *IEEE Access*, v. 7, p. 133653–133667, 2019.

34 Arulkumaran, K.; Deisenroth, M. P.; Brundage, M.; Bharath, A. A. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, v. 34, n. 6, p. 26–38, 2017.

35 Wei, Z.; Wang, D.; Zhang, M.; Tan, A.; Miao, C.; Zhou, Y. Autonomous agents in snake game via deep reinforcement learning. In: *2018 IEEE International Conference on Agents*. [S.l.: s.n.], 2018. p. 20–25.

36 POWERS, D. M. W. Evaluation: From precision, recall and f-factor to roc, informedness, markedness correlation. *Journal of Machine Learning Technologies*, v. 2, n. 1, p. 37–63, 2011.

37 MAROM, N. D.; ROKACH, L.; SHMILOVICI, A. Using the confusion matrix for improving ensemble classifiers. *IEEE Convention of Electrical and Electronics Engineers in Israel*, 2010.

38 Brandão, I. V.; da Costa, J. P. C. L.; Santos, G. A.; Praciano, B. J. G.; Júnior, F. C. M. D.; de S. Júnior, R. T. Classification and predictive analysis of educational data to improve the quality of distance learning courses. In: *2019 Workshop on Communication Networks and Power Systems*. [S.l.: s.n.], 2019. p. 1–6.

39 Hanlin, Q.; Xianzhen, J.; Xianrong, Z. Research on extract, transform and load(etl) in land and resources star schema data warehouse. In: *2012 Fifth International Symposium on Computational Intelligence and Design*. [S.l.: s.n.], 2012. v. 1, p. 120–123.

40 YAHOO! Finance. Disponível em: https://finance.yahoo.com/.

41 Agarwal, A.; El-Ghazawi, T.; El-Askary, H.; Le-Moigne, J. Efficient hierarchical-pca dimension reduction for hyperspectral imagery. In: *2007 IEEE International Symposium on Signal Processing and Information Technology*. [S.l.: s.n.], 2007. p. 353–356.

42 Dai, Y.; Guan, J.; Quan, W.; Xu, C.; Zhang, H. Pca-based dimensionality reduction method for user information in universal network. In: *2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems*. [S.l.: s.n.], 2012. v. 01, p. 70–74.

43 Zhang, T.; Yang, B. Big data dimension reduction using pca. In: *2016 IEEE International Conference on Smart Cloud*. [S.l.: s.n.], 2016. p. 152–157.

44 Shekar, B. H.; Dagnew, G. Grid search-based hyperparameter tuning and classification of microarray cancer data. In: *2019 Second International Conference on Advanced Computational and Communication Paradigms*. [S.l.: s.n.], 2019. p. 1–8.

45 Badashian, A. S.; Mahdavi, M.; Pourshirmohammadi, A.; nejad, M. M. Fundamental usability guidelines for user interface design. In: *2008 International Conference on Computational Sciences and Its Applications*. [S.l.: s.n.], 2008. p. 106–113.

46 Tesarik, J.; Dolezal, L.; Kollmann, C. User interface design practices in simple single page web applications. In: *2008 First International Conference on the Applications of Digital Information and Web Technologies*. [S.l.: s.n.], 2008. p. 223–228.

47 REACT. Disponível em: https://reactjs.org/.

48 FLASK. Disponível em: https://flask.palletsprojects.com/en/1.1.x/.

49 Vogel, P.; Klooster, T.; Andrikopoulos, V.; Lungu, M. A low-effort analytics platform for visualizing evolving flask-based python web services. In: *2017 IEEE Working Conference on Software Visualization*. [S.l.: s.n.], 2017. p. 109–113.

50 Wong, T.; Yeh, P. Reliable accuracy estimates from k-fold cross validation. *IEEE Transactions on Knowledge and Data Engineering*, v. 32, n. 8, p. 1586–1594, 2020.

51 Yadav, S.; Shukla, S. Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification. In: *2016 IEEE 6th International Conference on Advanced Computing*. [S.l.: s.n.], 2016. p. 78–83.

APÊNDICES

A CÓDIGO DO ETL DOS DADOS HISTÓRICOS DAS AÇÕES

```
import pandas as pd
import talib
import numpy as np
import matplotlib.pyplot as plt

from bs4 import BeautifulSoup
from scipy.ndimage.interpolation import shift
from bokeh.plotting import figure, output_file, show
from sklearn import preprocessing
from sqlalchemy import create_engine


def get_stock_data(start_date, end_date, stock_name):
    datelist = pd.date_range(start_date, end_date,freq='d')
    stock_data = []
    for date in datelist:
        r = requests.get(f"https://api.scrapingdog.com/scrape?api_key=<your-api-
            key>&url=https://finance.yahoo.com/quote/{stock_name}?p={stock_name}&d
            ={date}&.tsrc=fin-srch").text
        soup = BeautifulSoup(r, html .p a r s e r )
        alldata = soup.find_all(   tbody   )
        try:
         table1 = alldata[0].find_all(   tr   )
        except:
         table1=None
        try:
         table2 = alldata[1].find_all(   tr   )
        except:
         table2 = None

        l={}
        u=list()
        for i in range(0,len(table1)):
         try:
           table1_td = table1[i].find_all(   td   )
         except:
           table1_td = None
         l[table1_td[0].text] = table1_td[1].text
         u.append(l)
         l={}

        for i in range(0,len(table2)):
         try:
```

```
            table2_td = table2[i].find_all(   td   )
          except:
            table2_td = None
          l[table2_td[0].text] = table2_td[1].text
          u.append(l)

        stock_data.append(u)

    return stock_data

if __name__ == '__main__':
    start_date = '01-01-2013'
    end_date = '12-31-2019'
    stock_name = 'GOOGL'
    data = get_stock_data(start_date, end_date, stock_name)
    df = pd.DataFrame(data)
    engine = create_engine('sqlite://', echo=False)
    df.to_sql('stock_data', con=engine)
```

B CÓDIGO DO CÁLCULO DOS INDICADORES TÉCNICOS E DAS ESTRATÉGIAS DE POSICIONAMENTO

```
import pandas as pd
import talib
import numpy as np
import matplotlib.pyplot as plt

from scipy.ndimage.interpolation import shift
from bokeh.plotting import figure, output_file, show
from pandas_datareader import data
from sklearn import preprocessing
from etl import get_stock_data


def prepare_dataset_for_pca(data, stock_name):
    close = data['Close'].values
    # Generate input data - technical indicators
    _, ind1, __ = talib.BBANDS(close)  # Overlap: Bbands
    ind2 = talib.SMA(close)  # Overlap: SMA
    ind3 = talib.EMA(close)  # Overlap: EMA

    ind4 = talib.RSI(close)  # Momentum: RSI
    ind5 = talib.APO(close)  # Momentum: APO
    _, ind6, __ = talib.MACD(close)  # Momentum: MACD

    ind7 = talib.ADOSC(close)  # Volume: ADOSC
    ind8 = talib.OBV(close)    # Volume: OVV
    ind9 = talib.AD(close)     # Volume: AD

    ind10 = talib.MIN(close)  # Math Operator: Lowest value over a specified period
    ind11 = talib.MAX(close)  # Math Operator: Highest value over a specified period
    ind12 = talib.SUM(close)  # Math Operator: Summation over a specified period

    indicators, y, len_without_nan = macd_setup(close, stock_name, k_fold=True)
    len_without_nan = 63

    x = np.vstack((ind1[-len_without_nan:], ind2[-len_without_nan:], ind3[-len_without_nan:],
                   ind4[-len_without_nan:], ind5[-len_without_nan:], ind6[-len_without_nan:],
                   ind7[-len_without_nan:], ind8[-len_without_nan:], ind9[-len_without_nan:],
```

```
                    ind10[-len_without_nan:], ind11[-len_without_nan:], ind12[-len_without_nan:]
                    ))
    x = x.T

    return x, y


def prepare_dataset(data, stock_name, setup='macd', k_fold=False):
    """
    Function prepare_dataset to generate input data and trading strategy from stock close prices
    """

    close = data['Close'].values
    # Generate input data - technical indicators
    ind1 = talib.SMA(close)
    ind2 = talib.EMA(close)
    ind3 = talib.MACD(close)
    ind4 = talib.RSI(close)
    ind5 = talib.OBV(close)
    ind6 = talib.MAX(close)

    if setup == 'macd':
        indicators, y, len_without_nan = macd_setup(close, stock_name, k_fold)
    elif setup == '9_1':
        indicators, y, len_without_nan = nine_one_setup(close, stock_name, k_fold)
    elif setup == 'rsi':
        indicators, y, len_without_nan = rsi_setup(close, stock_name, k_fold)
    elif setup == 'pc':
        indicators, y, len_without_nan = pc_setup(close, stock_name, k_fold)

    x = np.vstack(indicators+(ind1[-len_without_nan:], ind2[-len_without_nan:],
                              ind3[-len_without_nan:], ind4[-len_without_nan:],
                                  ind5[-len_without_nan:], ind6[-len_without_nan:])
                                  )
    x = x.T

    return x, y


def check_pnl(df, set_index=True):
    """
    Function to return accumulated daily return based on dataframe with 'Position' and 'Close' columns
    :param df:
    :return:
    """

    for index, row in df.iterrows():
        if index != 0 and index != len(df):
            if row['Position'] == 0:
```

```
                last_operation = df.loc[index - 1]
                last_df = df[:index][::-1][df.Position == 0]
                if last_df.empty:
                    last_df = last_operation
                    last_price = last_df['Close']
                else:
                    last_price = last_df.iloc[0]['Close']

                pnl = (row['Close'] / last_price) - 1
                if last_operation['Position'] == -1 and row['Close']:
                    pnl *= -1

                df.loc[index, 'Pnl'] = pnl

    df['acc_daily_return'] = df['Pnl'].cumsum()
    df['acc_daily_return'] = df['acc_daily_return'] * 100
    df.dropna(inplace=True)
    if set_index:
        df.index = df.Date


def macd_setup(close, stock_name, k_fold):
    """
    Function to prepare dataset with macd setup
    :return:
    """
    macd, macdsignal, macdhist = talib.MACD(close, fastperiod=12, slowperiod=26,
        signalperiod=9)

    # Cancel NaN values
    macdhist = macdhist[~np.isnan(macdhist)]
    len_without_nan = len(macdhist)
    macd = macd[-len(macdhist):]
    macdsignal = macdsignal[-len(macdhist):]

    # Scaling features to a range [0, 1]
    min_max_scaler = preprocessing.MinMaxScaler(feature_range=(-1, 1))
    macdhist_norm = min_max_scaler.fit_transform(np.expand_dims(macdhist, axis=1))

    # Implement strategy
    start_sell = 0.45
    stop_sell = 0.05
    start_buy = -0.45
    stop_buy = -0.05

    y = np.full(len(macdhist), np.nan)
    y[0] = 0

    for i in range(1, len(macdhist)):

        if y[i - 1] == 0:
            if (macdhist_norm[i] >= start_sell):
```

```
                # Enter sell position
                y[i] = -1
            elif (macdhist_norm[i] <= start_buy):
                # Enter buy position
                y[i] = 1
            else:
                y[i] = 0
        elif y[i - 1] == -1:
            if macdhist_norm[i] > stop_sell:
                # Stay in sell position
                y[i] = -1
            else:
                # Leave sell position
                y[i] = 0
        else:
            if macdhist_norm[i] < stop_buy:
                # Stay in buy position
                y[i] = 1
            else:
                # Leave buy position
                y[i] = 0

    # Plot strategy
    dates = np.arange(len(macdhist))
    plt.plot(dates, y, 'g', label='Strategy Positions')
    plt.bar(dates, macdhist_norm[:, 0], width=1, color='blue', label='MACD
        histogram')
    plt.plot(dates, start_sell * np.ones(len(macdhist)), 'k--', lw=1)
    plt.plot(dates, stop_sell * np.ones(len(macdhist)), 'k--', lw=1)
    plt.plot(dates, start_buy * np.ones(len(macdhist)), 'k--', lw=1)
    plt.plot(dates, stop_buy * np.ones(len(macdhist)), 'k--', lw=1)
    plt.xlabel('Days')
    plt.xlim((300, 600))
    plt.legend()
    try:
        plt.savefig(f'./images/macd/{stock_name}.png', bbox_inches='tight')
    except Exception as e:
        print('fail on trying to save setup')
    plt.show()
    try:
        file_model = './model/'
        if k_fold:
            file_model = './k-fold/'
        pd.DataFrame({'y': y, 'macdhist_norm': macdhist_norm[:,0]}).to_csv(f'{
            file_model}macd/setup_{stock_name}.csv')
    except Exception as e:
        print('fail on trying to save setup')

    indicators = (macdhist, macd, macdsignal)

    return indicators, y, len_without_nan

```

```
186 def nine_one_setup(close, stock_name, k_fold):
187     """
188     Function to prepare dataset with 9.1 setup
189     :return:
190     """
191     macd, macdsignal, macdhist = talib.MACD(close, fastperiod=12, slowperiod=26,
            signalperiod=9)
192     ema_9 = talib.EMA(close, 9)
193 
194     # Cancel NaN values
195     macdhist = macdhist[~np.isnan(macdhist)]
196     len_without_nan = len(macdhist)
197     macd = macd[-len(macdhist):]
198     macdsignal = macdsignal[-len(macdhist):]
199     ema_9 = ema_9[-len(macdhist):]
200     close_values = close[-len(macdhist):]
201 
202 
203     y = np.full(len(ema_9), np.nan)
204     y[0] = 0
205 
206     for i in range(1, len(ema_9)):
207 
208         if y[i - 1] == 0:
209             if (ema_9[i] <= ema_9[i-1]):
210                 # Enter sell position
211                 y[i] = -1
212             elif (ema_9[i] >= ema_9[i-1]):
213                 # Enter buy position
214                 y[i] = 1
215             else:
216                 y[i] = 0
217         elif y[i - 1] == -1:
218             if ema_9[i] > close_values[i]:
219                 # Stay in sell position
220                 y[i] = -1
221             else:
222                 # Leave sell position
223                 y[i] = 0
224         else:
225             if ema_9[i] < close_values[i]:
226                 # Stay in buy position
227                 y[i] = 1
228             else:
229                 # Leave buy position
230                 y[i] = 0
231 
232     # Plot strategy
233     dates = np.arange(len(macdhist))
234     test_index = len(y)*0.8
235     plt.plot(dates, y, 'g', label='Strategy Positions')
236     plt.xlabel('Days')
```

```
237     plt.xlim((test_index, len(y) ))
238     plt.legend()
239     try:
240         plt.savefig(f'./images/9_1/{stock_name}.png', bbox_inches='tight')
241     except:
242         print('fail on saving image')
243 
244     plt.show()
245     try:
246         file_model = './model/'
247         if k_fold:
248             file_model = './k-fold/'
249         pd.DataFrame({'y': y, 'ema_9': ema_9}).to_csv(f'{file_model}9_1/setup_{
                stock_name}.csv')
250     except Exception:
251         print('fail on trying to save setup')
252 
253     indicators = (macdhist, macd, macdsignal, ema_9, close_values)
254 
255     return indicators, y, len_without_nan
256 
257 
258 def rsi_setup(close, stock_name, k_fold):
259     """
260     Function to prepare dataset with rsi setup
261     :return:
262     """
263     macd, macdsignal, macdhist = talib.MACD(close, fastperiod=12, slowperiod=26,
            signalperiod=9)
264     rsi = talib.RSI(close, 9)
265 
266     # Cancel NaN values
267     macdhist = macdhist[~np.isnan(macdhist)]
268     len_without_nan = len(macdhist)
269     macd = macd[-len(macdhist):]
270     macdsignal = macdsignal[-len(macdhist):]
271 
272     rsi_values = rsi[-len(macdhist):]
273     # Implement strategy
274     start_sell = 50
275     stop_sell = 30
276     start_buy = 30
277     stop_buy = 50
278 
279     y = np.full(len(rsi_values), np.nan)
280     y[0] = 0
281 
282     for i in range(1, len(rsi_values)):
283 
284         if y[i - 1] == 0:
285             if (rsi_values[i] >= start_sell):
286                 # Enter sell position
```

```
                y[i] = -1
            elif (rsi_values[i] <= start_buy):
                # Enter buy position
                y[i] = 1
            else:
                y[i] = 0
        elif y[i - 1] == -1:
            if rsi_values[i] > stop_sell:
                # Stay in sell position
                y[i] = -1
            else:
                # Leave sell position
                y[i] = 0
        else:
            if rsi_values[i] < stop_buy:
                # Stay in buy position
                y[i] = 1
            else:
                # Leave buy position
                y[i] = 0

    # Plot strategy
    dates = np.arange(len(rsi_values))
    test_index = len(y) * 0.8

    plt.plot(dates, y, 'g', label='Strategy Positions')
    plt.xlabel('Days')
    plt.xlim((test_index, len(y)))
    plt.legend()
    try:
        plt.savefig(f'./images/rsi/{stock_name}.png', bbox_inches='tight')
    except:
        print('error on saving image')
    plt.show()

    try:
        file_model = './model/'
        if k_fold:
            file_model = './k-fold/'
        pd.DataFrame({'y': y, 'rsi': rsi_values}).to_csv(f'{file_model}rsi/setup_{
            stock_name}.csv')
    except Exception as e:
        print('fail on trying to save setup')

    indicators = (rsi_values, macdhist, macd, macdsignal)

    return indicators, y, len_without_nan


def pc_setup(close, stock_name, k_fold):
    """
    Function to prepare dataset with pc setup
```

```
    :return:
    """
    macd, macdsignal, macdhist = talib.MACD(close, fastperiod=12, slowperiod=26,
        signalperiod=9)
    sma_21 = talib.SMA(close, 21)

    # Cancel NaN values
    macdhist = macdhist[~np.isnan(macdhist)]
    len_without_nan = len(macdhist)
    macd = macd[-len(macdhist):]
    macdsignal = macdsignal[-len(macdhist):]

    sma_21 = sma_21[-len(macdhist):]

    close_values = close[-len(macdhist):]
    sma_21_shift = shift(sma_21, 1, cval=sma_21[0])
    difference_sma_21 = sma_21 - sma_21_shift
    ratio_close_sma = close_values/sma_21

    # (sma_21 - sma_21[i-1]) (high_values - sma_21)
    y = np.full(len(sma_21), np.nan)
    y[0] = 0
    print('adding ratio')

    for i in range(1, len(sma_21)):

        if y[i - 1] == 0:
            # if (difference_sma_21[i] < 0 and difference_high[i] > 0):
            if (difference_sma_21[i] < 0 and ratio_close_sma[i] < 1):
                # Enter sell position
                y[i] = -1

            # elif (difference_sma_21[i] > 0 and difference_low[i] > 0):
            elif (difference_sma_21[i] > 0 and ratio_close_sma[i] < 1):
                # Enter buy position
                y[i] = 1
            else:
                y[i] = 0
        elif y[i - 1] == -1:
            if ratio_close_sma[i] < 1:
                # Stay in sell position
                y[i] = -1
            else:
                # Leave sell position
                y[i] = 0
        else:
            if ratio_close_sma[i] < 1:
                # Stay in buy position
                y[i] = 1
            else:
                # Leave buy position
                y[i] = 0
```

```
389
390     # Plot strategy
391     dates = np.arange(len(macdhist))
392     test_index = len(y)*0.8
393
394     plt.plot(dates, y, 'g', label='Strategy Positions')
395     plt.xlabel('Days')
396     plt.xlim((test_index, len(y)))
397     plt.legend()
398     try:
399         plt.savefig(f'./images/pc/{stock_name}.png', bbox_inches='tight')
400     except:
401         print('failed on try saving image')
402     plt.show()
403     try:
404         file_model = './model/'
405         if k_fold:
406             file_model = './k-fold/'
407         pd.DataFrame({'y': y, 'sma_21': sma_21}).to_csv(f'{file_model}pc/setup_{
                stock_name}.csv')
408     except Exception:
409         print('fail on trying to save setup')
410
411     indicators = (sma_21, close_values, ratio_close_sma, macdhist, macd,
            macdsignal)
412
413     return indicators, y, len_without_nan
414
415
416
417 if __name__ == '__main__':
418     start_date = '01-01-2013'
419     end_date = '12-31-2018'
420     stock_name = 'GOOGL'
421     data = get_stock_data(start_date, end_date, stock_name)
422     x, Y = prepare_dataset(data, stock_name, setup='pc')
423     # x, Y = prepare_dataset_for_pca(data, stock_name)
424     test_index = int(len(Y)*0.2)
425     test_index = len(Y)
426     df = pd.DataFrame({'Position': Y[-test_index:], 'Close': data.Close[-
            test_index:]})
427     df.reset_index(inplace=True)
428     check_pnl(df)
429     test_pnl = df['acc_daily_return']
430     fig = plt.figure()
431     ax1 = fig.add_subplot(111, ylabel='Accumulated Daily Return (%)')
432     test_pnl.plot(ax=ax1, color='y', lw=2., legend=True, label='Strategy return')
433     plt.show()
```

C CÓDIGO DA SELEÇÃO DE VARIÁVEIS E REDUÇÃO DA DIMENSIONALIDADE

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


# Import libraries
import pandas as pd
import talib
import timeit
import tensorflow as tf
import pandas_datareader.data as web
import scikitplot as skplt
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn import preprocessing
from get_stock_data import get_stock_data, prepare_dataset_for_pca
from sklearn.decomposition import PCA
from sklearn.feature_selection import VarianceThreshold
from sklearn.preprocessing import StandardScaler


# In[2]:


# Stock symbol
symbol = 'AAL'
# Load close prices of to NumPy array
# start_date = '01-01-2013'
# end_date = '01-01-2018'
start_date='01-01-2013'
end_date='12-31-2018'
data = get_stock_data(start_date=start_date,end_date=end_date, stock_name=symbol)
data


# In[3]:


close = data['Close'].values
close
```

```
44
45
46  # In[4]:
47
48
49  X, y = prepare_dataset_for_pca(data, symbol)
50
51
52  # In[5]:
53
54
55  # Split dataset
56  n_train = int(X.shape[0] * 0.8)
57  X_train, y_train = X[:n_train], y[:n_train]
58  X_test, y_test = X[n_train:], y[n_train:]
59
60
61  # In[6]:
62
63
64  # Normalize data
65  stdsc = StandardScaler()
66  # scaler = preprocessing.MinMaxScaler()
67
68  X_train = stdsc.fit_transform(X_train)
69  X_test = stdsc.transform(X_test)
70
71
72  # In[7]:
73
74
75  X_train.shape
76
77
78  # In[ ]:
79
80
81  X_train.std()
82
83
84  # In[35]:
85
86
87  # Applying PCA
88  n_components=12
89  pca = PCA(n_components=n_components)
90  X_train_pca = pca.fit_transform(X_train)
91  X_test_pca = pca.transform(X_test)
92
93
94  # In[36]:
95
```

```
96
97  pca.explained_variance_ratio_
98
99
100 # In[37]:
101
102
103 pca.get_covariance()
104
105
106 # In[44]:
107
108
109 plt.figure(figsize=(8,8))
110 sns.set(font_scale=1.2)
111 n_features = 12
112 cols= [i for i in range(1, n_features+1)]
113 print(cols)
114 hm = sns.heatmap(pca.get_covariance(),
115                  cbar=True,
116                  annot=True,
117                  square=True,
118                  fmt='.2f',
119                  annot_kws={'size': 12},
120                  yticklabels=cols,
121                  xticklabels=cols)
122 plt.title('Covariance matrix showing correlation coefficients')
123 plt.tight_layout()
124 plt.savefig('covariance_matrix.png')
125 plt.show()
126
127
128 # In[15]:
129
130
131 fig = skplt.decomposition.plot_pca_component_variance(pca,
        target_explained_variance=0.75)
132
133
134 # In[65]:
135
136
137 pca_data = {
138     'components': [i for i in range(1, n_components+1)],
139     'explained_variance_ratio': pca.explained_variance_ratio_
140 }
141 pd.DataFrame(data=pca_data).to_excel('pca_components.xlsx', index=False)
142
143
144 # In[68]:
145
146
```

```
var_list = pca.explained_variance_ratio_
pca_data = {
    'components': [i for i in range(1,n_components+1)],
    'explained_variance_ratio': [np.sum(var_list[0:index]) for index in range(1,
        n_components+1)]
}
pd.DataFrame(data=pca_data).to_excel('pca_explained.xlsx', index=False)


# In[33]:


# Aplying feature selection
p = 0.874
variance = (p * (1 - p))
sel = VarianceThreshold(threshold=variance)
sel.fit_transform(X_train)


# In[34]:


sel
```

D CÓDIGO DA CLASSIFICAÇÃO E PREDIÇÃO COM USO DE DRL

```
import pickle
import numpy as np
import random

from collections import deque

from keras.models import Sequential
from keras.layers import Dense, Flatten
from keras.layers import Dropout


# Deep Q-learning Agent
class Agent:

    def __init__(self, look_back, action_size, n_features):
        self.look_back = look_back
        self.action_size = action_size
        self.n_features = n_features

        self.memory = deque(maxlen=3000)

        self.gamma = 0.95  # discount rate
        self.epsilon = 1.0  # exploration rate
        self.epsilon_min = 0.01
        self.epsilon_decay = 0.995

        self.model = self.create_DQN()

    def create_DQN(self, optimizer='adam', loss='mse', activation='linear'):
        model = Sequential()
        model.add(Dense(30, input_shape=(self.look_back, self.n_features),
            activation='relu'))
        model.add(Dense(15, activation='relu'), )
        model.add(Dense(5, activation='relu'))
        model.add(Flatten())
        model.add(Dense(self.action_size, activation=activation))

        model.compile(optimizer=optimizer, loss=loss, metrics=['accuracy'])

        return model

    def remember(self, state, action, reward, next_state, done):
        self.memory.append((state, action, reward, next_state, done))
```

```
43
44      def replay(self, batch_size):
45          mini_batch = random.sample(self.memory, batch_size)
46
47          for state, action, reward, next_state, done in mini_batch:
48
49              if done:
50                  target = reward
51              else:
52                  target = reward + self.gamma * np.amax(self.model.predict(
                        next_state)[0])
53
54              target_f = self.model.predict(state)
55              target_f[0][action] = target
56
57              self.model.fit(state, target_f, epochs=1, verbose=0)
58
59          if self.epsilon > self.epsilon_min:
60              self.epsilon *= self.epsilon_decay
61
62      def act(self, state):
63          if np.random.rand() <= self.epsilon:
64              return np.random.randint(self.action_size)
65
66          act_values = self.model.predict(state)
67
68          return np.argmax(act_values[0])
69
70      def save_checkpoint(self, symbol, ep, setup, k_fold=False):
71          file = f'./checkpoints/{setup}/{symbol}_robot_checkpoint{ep}'
72          if k_fold:
73              file = f'./k-fold/{setup}/{symbol}_robot_checkpoint{ep}'
74
75          # Serialize weights to HDF5
76          self.model.save_weights(file + ".h5")
77          # Save epsilon
78          pickle.dump(self.epsilon, open(file + "_epsilon.pickle", "wb"))
79
80      def run(self, dataX, dataY, episodes, batch_size, symbol, setup, k_fold=False)
            :
81          times = len(dataX)
82
83          total_reward_list = []
84          max_reward = (times - self.look_back + 1)
85          optimal_reward = max_reward * 0.92
86          print(f'max reward: {max_reward}')
87          print(f'optimal reward: {optimal_reward}')
88          file_model = './model/'
89          file_checkpoint = './checkpoints/'
90          file = ''
91          ep = 0
92
```

```
93          for ep in range(episodes):
94              state = dataX[:self.look_back, :][np.newaxis, :, :]
95              pos = dataY[self.look_back - 1]
96
97              done = False
98              total_reward = 0
99
100             for t in range(1, times - self.look_back + 1):
101
102                 action = self.act(state)
103
104                 if action == pos:  # 0:-1      1:0     2:1
105                     reward = +1
106
107                 elif (pos == 0 or pos == 2):
108                     if action == 1:
109                         reward = 0
110                     else:
111                         reward = -1
112                 else:
113                     reward = -1
114
115                 total_reward += reward
116
117                 if t == times - self.look_back:
118                     done = True
119
120                 next_state = dataX[t:t + self.look_back, :][np.newaxis, :, :]
121                 next_pos = dataY[t + self.look_back - 1]
122
123                 self.remember(state, action, reward, next_state, done)
124
125                 state = next_state
126                 pos = next_pos
127
128                 if done:
129                     print('Episode: %i ---> Total Reward: %i' % (ep, total_reward)
                            )
130                     total_reward_list.append(total_reward)
131
132                 if len(self.memory) > batch_size:
133                     self.replay(batch_size)
134
135             if (ep + 1) % 5 == 0 and ep > 0:
136                 self.save_checkpoint(symbol, ep + 1, setup, k_fold)
137
138             if total_reward >= optimal_reward:
139                 print(f'Stop training because it reached the goal of {round(
                        total_reward / max_reward, 2)} '
140                       f'to the total of {max_reward}...')
141                 self.save_checkpoint(symbol, ep + 1, setup, k_fold)
142                 break
```

```
143
144         if k_fold:
145             file_model = './k-fold/'
146             file_checkpoint = './k-fold/'
147
148         self.model.save_weights(f"{file_model}{setup}/{symbol}_final_weights_agent
                .h5")
149         model_json = self.model.to_json()
150         with open(f"{file_model}{setup}/{symbol}_final_model.json", "w") as
                json_file:
151             json_file.write(model_json)
152
153         np.savetxt(f'{file_checkpoint}{setup}/{symbol}_robot_checkpoint{ep + 1}
                _total_reward.txt', total_reward_list)
154
155         return ep + 1
156
157     def evaluate(self, dataX, dataY):
158
159         times = len(dataX)
160
161         state = dataX[:self.look_back, :][np.newaxis, :, :]
162         pos = dataY[self.look_back - 1]
163         pos_list = []
164
165         done = False
166         total_reward = 0
167
168         for t in range(1, times - self.look_back + 1):
169
170             action = self.act(state)
171
172             if action == pos:
173                 reward = +1
174
175             elif (pos == 0 or pos == 2):
176                 if action == 1:
177                     reward = 0
178                 else:
179                     reward = -1
180             else:
181                 reward = -1
182
183             pos_list.append(action)
184             total_reward += reward
185
186             if t == times - self.look_back:
187                 done = True
188
189             next_state = dataX[t:t + self.look_back, :][np.newaxis, :, :]
190             next_pos = dataY[t + self.look_back - 1]
191             self.remember(state, action, reward, next_state, done)
```

```
192
193             state = next_state
194             pos = next_pos
195
196             if done:
197                 print(f'Total Reward: {total_reward}/{times - self.look_back + 1}'
                        )
198
199         return np.array(pos_list)
```

```
1  #!/usr/bin/env python
2  # coding: utf-8
3
4  # In[2]:
5
6
7  # Import libraries
8  import pandas as pd
9  import numpy as np
10 import matplotlib.pyplot as plt
11 import pickle
12 import talib
13 import timeit
14 import tensorflow as tf
15 import os
16 import keras
17 import pandas_datareader.data as web
18
19 from keras.models import model_from_json
20 from sklearn import preprocessing
21 from sklearn.metrics import accuracy_score
22 from sklearn.metrics import precision_recall_fscore_support
23 from get_stock_data import get_stock_data, prepare_dataset, check_pnl
24 from sklearn.metrics import plot_confusion_matrix
25 from sklearn.model_selection import TimeSeriesSplit
26
27
28 from Agent import Agent
29
30
31 # In[2]:
32
33
34 #GPU
35 from tensorflow.python.client import device_lib
36 # get the number of cpus availables: grep -c ^processor /proc/cpuinfo
37 sess = tf.compat.v1.Session(config=tf.compat.v1.ConfigProto(log_device_placement=
       True))
38 config = tf.compat.v1.ConfigProto( device_count = {'GPU': 1 , 'CPU': 4} )
39 sess = tf.compat.v1.Session(config=config)
```

```
tf.compat.v1.keras.backend.set_session(sess)


# In[3]:


# Stock symbol
symbol = 'TSLA'
# Load close prices of to NumPy array
start_date='01-01-2013'
end_date='12-31-2019'
data = get_stock_data(start_date=start_date,end_date=end_date, stock_name=symbol)
data


# In[4]:


close = data['Close'].values
close


# In[5]:


# Implement RSI trading strategy
# Setup options:
# - rsi [default]
# - 9_1
# - macd
# - pc
setup='rsi'
X, y = prepare_dataset(data, symbol, setup=setup)


# In[6]:


print(set(y))
print(X.shape)


# In[6]:


## Create and train the Agent
# Variable definiton
episodes = 25
look_back = 20
batch_size = 32


```

```
# In[8]:


# Split dataset
n_train = int(X.shape[0] * 0.8)
X_train, y_train = X[:n_train], y[:n_train]
X_test, y_test = X[n_train:], y[n_train:]


# In[7]:


# NO SPLIT dataset when it aims to test with some stock and validate with another
symbol_another = symbol
# symbol_another = 'AAL'
data_another = get_stock_data(start_date=start_date,end_date=end_date, stock_name=
    symbol_another)
close_another = data_another['Close'].values
X_another, y_another = prepare_dataset(data_another, symbol_another, setup=setup)

X_train, y_train = X, y
X_test, y_test = X_another, y_another


# In[9]:


## Run with TimeSeries k-fold
n_features = X_train.shape[1]
agent_kfold = Agent(look_back, 3, n_features)
k = 1
tscv = TimeSeriesSplit(n_splits=3)
for train_index, test_index in tscv.split(X):
    X_train_kfold, X_test_kfold = X[train_index], X[test_index]
    y_train_kfold, y_test_kfold = y[train_index], y[test_index]

    scaler = preprocessing.MinMaxScaler()
    X_train_kfold = scaler.fit_transform(X_train_kfold)
    X_test_kfold = scaler.transform(X_test_kfold)

    le = preprocessing.LabelEncoder()

    y_train_kfold = le.fit_transform(y_train_kfold)
    y_test_kfold = le.transform(y_test_kfold)


    start = timeit.default_timer()
    episode = agent_kfold.run(X_train_kfold, y_train_kfold, episodes, batch_size,
        symbol, setup, k_fold=True)
    stop = timeit.default_timer()
    print(f'stop training in {(stop - start)/60} min')
    y_pred_test_kfold = agent_kfold.evaluate(X_test_kfold, y_test_kfold)
```

```
142
143     # Calculate and print accuracy
144     acc = accuracy_score(y_test_kfold[look_back-1:-1], y_pred_test_kfold)
145
146     print(f'Accuracy: {round(acc*100,2)} %')
147
148     p, r, f1, s = precision_recall_fscore_support(y_test_kfold[look_back-1:-1],
            y_pred_test_kfold, average=None)
149     try:
150         results = pd.DataFrame({'1-Precision': p, '2-Recall': r, '3-F1 score': f1,
                '4-Support': s},
151                                index=le.classes_).round(decimals=3)
152         print(results)
153         results.to_csv(f'./k-fold/{setup}/{symbol}_fscore_{k}.csv')
154         k = k+1
155     except Exception as e:
156         print('exception: ', e)
157
158
159 # In[8]:
160
161
162 # Normalize data
163 scaler = preprocessing.MinMaxScaler()
164
165 X_train = scaler.fit_transform(X_train)
166 X_test = scaler.transform(X_test)
167
168
169 # In[9]:
170
171
172 # Encode trading signal with integers between 0 and n-1 classes
173 le = preprocessing.LabelEncoder()
174
175 y_train = le.fit_transform(y_train)
176 y_test = le.transform(y_test)
177
178 print(le.classes_)
179
180
181 # In[10]:
182
183
184 # Create Agent
185 n_features = X_train.shape[1]
186 action_size = len(le.classes_)
187 agent = Agent(look_back, action_size, n_features)
188
189
190 # In[11]:
191
```

```
192
193 # Train Agent
194 start = timeit.default_timer()
195 episode = agent.run(X_train, y_train, episodes, batch_size, symbol, setup)
196 stop = timeit.default_timer()
197 print(f'stop training in {(stop - start)/60} min')
198
199
200 # In[12]:
201
202
203 # Load rewards
204 total_reward_list = np.loadtxt(f'./checkpoints/{setup}/{symbol}_robot_checkpoint{
        episode}_total_reward.txt')
205 # Plot
206 plt.figure()
207 plt.plot(np.arange(1, episode+1), total_reward_list)
208 plt.xlabel('Episodes')
209 plt.ylabel('Total Reward')
210 plt.savefig(f'./images/{setup}/total_reward.png')
211 plt.show()
212
213
214 # In[11]:
215
216
217 #Load some agent saved and its weights
218 # load json and create model
219 action_size = 3
220 n_features = X_train.shape[1]
221 # json_file = open(f'./model/{setup}/{symbol}_final_model.json', 'r')
222 # loaded_model_json = json_file.read()
223 # json_file.close()
224 loaded_agent = Agent(look_back, action_size, n_features)
225 loaded_agent.epsilon = pickle.load( open(f'./checkpoints/{setup}/{symbol}
        _robot_checkpoint25_epsilon.pickle', "rb"))
226 # loaded_agent.model = model_from_json(loaded_model_json)
227 # load weights into new model
228 loaded_agent.model.load_weights(f"./checkpoints/{setup}/{symbol}
        _robot_checkpoint25.h5")
229
230
231 # In[12]:
232
233
234 # When it aims to test with saved agents
235 agent = loaded_agent
236
237
238 # In[13]:
239
240
```

```
# Evaluate the model
# Make predictions
y_pred_test = agent.evaluate(X_test, y_test)

# Calculate and print accuracy
acc = accuracy_score(y_test[look_back-1:-1], y_pred_test)

print(f'Accuracy: {round(acc*100,2)} %')


# In[14]:


# Calculate and print precision, recall, f1 score and support
p, r, f1, s = precision_recall_fscore_support(y_test[look_back-1:-1], y_pred_test,
     average=None)
results = pd.DataFrame({'1-Precision': p, '2-Recall': r, '3-F1 score': f1, '4-
    Support': s},
                       index=le.classes_).round(decimals=3)
print(results)
results.to_csv(f'./model/{setup}/{symbol}_fscore.csv')


# In[15]:


# Decodificate labels
y_true_test = le.inverse_transform(y_test[look_back-1:-1])
y_pred_test = le.inverse_transform(y_pred_test)


# In[16]:


# Plot strategy
plt.figure()
plt.plot(y_true_test, label='true')
plt.plot(y_pred_test, label='pred')
pd.DataFrame({'y_true_test': y_true_test, 'y_pred_test': y_pred_test}).to_csv(f'./
    model/{setup}/{symbol}_true_pred.csv')
plt.savefig(f'./images/{setup}/plot_strategy.png')
plt.legend()
plt.show()


# In[18]:


#PnL strategy
SP500 = web.DataReader(['sp500'], 'fred', start_date, end_date)
test_index = len(y) - int(len(y)*0.8) - look_back
SP500 = SP500[-test_index:]
```

```
SP500['daily_return'] = (SP500['sp500']/ SP500['sp500'].shift(1)) -1
SP500['acc_daily_return'] = SP500['daily_return'].cumsum()
SP500['acc_daily_return'] = SP500['acc_daily_return']*100

data_symbol = data[-test_index:]
data_symbol['daily_return'] = (data_symbol['Close']/ data_symbol['Close'].shift(1)) -1
data_symbol['acc_daily_return'] = data_symbol['daily_return'].cumsum()
data_symbol['acc_daily_return'] = data_symbol['acc_daily_return']*100

#Drop all Not a number values using drop method.
data_symbol.dropna(inplace=True)
SP500.dropna(inplace=True)

prediction = pd.DataFrame({'Position': y_pred_test[-test_index:], 'Close': data.Close[-test_index:]})
prediction.reset_index(inplace=True)
check_pnl(prediction)

real = pd.DataFrame({'Position': y_true_test[-test_index:], 'Close': data.Close[-test_index:]})
real.reset_index(inplace=True)
check_pnl(real)

prediction['acc_daily_return'] = prediction['Pnl'].cumsum()
prediction['acc_daily_return'] = prediction['acc_daily_return']*100
prediction.dropna(inplace = True)
prediction.index = prediction.Date


fig = plt.figure()
ax1 = fig.add_subplot(111, ylabel='Accumulated Daily Return (%)')

sp_500 = SP500['acc_daily_return']
trading_agent = prediction['acc_daily_return']
data_symbol_chart = data_symbol['acc_daily_return']
real_trading = real['acc_daily_return']

if 'SA' in symbol:
    symbol = symbol[:-3]
data_symbol_chart.to_csv(f'./pnl/{setup}/{symbol}_pnl.csv')
prediction.to_csv(f'./pnl/{setup}/trading_agent_pnl.csv')
real_trading.to_csv(f'./pnl/{setup}/real_trading_pnl.csv')
SP500.to_csv(f'./pnl/{setup}/sp500_pnl.csv')


sp_500.plot(ax=ax1, color='y', lw=2., legend=True, label='S&P 500')
trading_agent.plot(ax=ax1, color='b', lw=2., legend=True, label='Trading Agent')
real_trading.plot(ax=ax1, color='r', lw=2., legend=True, label='Real Trading')
data_symbol_chart.plot(ax=ax1, color='g', lw=2., legend=True, label=symbol)
plt.title(f'Agent Trading vs SP&500 vs {symbol} accumulated daily returns')
plt.savefig(f'./images/{setup}/trading vs sp500 vs {symbol} index')
```

```
plt.show()


# In[3]:


#PnL strategy with entire stock
SP500 = web.DataReader(['sp500'], 'fred', start_date, end_date)
entire_index = len(y) - look_back
SP500 = SP500[-entire_index:]
SP500['daily_return'] = (SP500['sp500']/ SP500['sp500'].shift(1)) -1
SP500['acc_daily_return'] = SP500['daily_return'].cumsum()
SP500['acc_daily_return'] = SP500['acc_daily_return']*100

data_entire = data_another[-entire_index:]
data_entire['daily_return'] = (data_entire['Close']/ data_entire['Close'].shift(1)
    ) -1
data_entire['acc_daily_return'] = data_entire['daily_return'].cumsum()
data_entire['acc_daily_return'] = data_entire['acc_daily_return']*100

#Drop all Not a number values using drop method.
data_another.dropna(inplace=True)
SP500.dropna(inplace=True)

prediction = pd.DataFrame({'Position': y_pred_test[-entire_index:], 'Close':
    data_entire.Close[-entire_index:]})
prediction.reset_index(inplace=True)
check_pnl(prediction)

real = pd.DataFrame({'Position': y_true_test[-entire_index:], 'Close': data_entire
    .Close[-entire_index:]})
real.reset_index(inplace=True)
check_pnl(real)

prediction['acc_daily_return'] = prediction['Pnl'].cumsum()
prediction['acc_daily_return'] = prediction['acc_daily_return']*100
prediction.dropna(inplace = True)
prediction.index = prediction.Date


fig = plt.figure()
ax1 = fig.add_subplot(111, ylabel='Accumulated Daily Return (%)')

sp_500 = SP500['acc_daily_return']
trading_agent = prediction['acc_daily_return']
data_entire_chart = data_entire['acc_daily_return']
real_trading = real['acc_daily_return']

if 'SA' in symbol:
    symbol = symbol[:-3]

if 'SA' in symbol_another:
```

```
388     symbol_another = symbol_another[:-3]
389
390 data_entire_chart.to_csv(f'./entire_stock/{setup}/pnl_{symbol_another}_pnl.csv')
391 prediction.to_csv(f'./entire_stock/{setup}/pnl_{symbol_another}_trading_agent_pnl.
        csv')
392 real_trading.to_csv(f'./entire_stock/{setup}/pnl_{symbol_another}_real_trading_pnl
        .csv')
393 SP500.to_csv(f'./entire_stock/{setup}/pnl_sp500_pnl.csv')
394
395
396 sp_500.plot(ax=ax1, color='y', lw=2., legend=True, label='S&P 500')
397 trading_agent.plot(ax=ax1, color='b', lw=2., legend=True, label='Trading Agent')
398 real_trading.plot(ax=ax1, color='r', lw=2., legend=True, label='Real Trading')
399 data_entire_chart.plot(ax=ax1, color='g', lw=2., legend=True, label=symbol_another
        )
400 plt.title(f'Agent Trading vs SP&500 vs {symbol_another} accumulated daily returns'
        )
401 plt.savefig(f'./entire_stock/{setup}/pnl_trading vs sp500 vs {symbol_another}
        index')
402 plt.show()
```