UNIVERSIDADE FEDERAL DO PAMPA

MARCO ANTONIO JORGE TICONA

MODELO DA ESTRATÉGIA INNER CIRCLE TRADE USANDO INTELIGÊNCIA ARTIFICIAL

Bagé
2023

MARCO ANTONIO JORGE TICONA

MODELO DA ESTRATÉGIA INNER CIRCLE TRADE USANDO INTELIGÊNCIA ARTIFICIAL

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Engenharia de Computação como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Milton Roberto Heinen

Bagé
2023

Ficha catalográfica elaborada automaticamente com os dados fornecidos
pelo(a) autor(a) através do Módulo de Biblioteca do
Sistema GURI (Gestão Unificada de Recursos Institucionais) .

T555m TIcona, Marco Antonio Jorge
Modelo da estratégia Inner Circle Trade usando Inteligência Artificial / Marco Antonio Jorge TIcona.
119 p.

Trabalho de Conclusão de Curso(Graduação)-- Universidade Federal do Pampa, ENGENHARIA DE COMPUTAÇÃO, 2023.
"Orientação: Milton Roberto Heinen".

1. Inteligência Artificial. 2. Mercado Financeiro. 3. Aprendizado de Máquina . 4. Forex. 5. Matriz de Confusão. I. Título.

SERVIÇO PÚBLICO FEDERAL

MINISTÉRIO DA EDUCAÇÃO

Universidade Federal do Pampa

MARCO ANTONIO JORGE TICONA

MODELO DA ESTRATÉGIA INNER CIRCLE TRADE USANDO INTELIGÊNCIA ARTIFICIAL

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Engenharia de Computação como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação.

Trabalho de Conclusão de Curso defendido e aprovado em: 13 de julho de 2023.

Banca examinadora:

_____

Prof. Dr. Milton Roberto Heinen
Orientador
UNIPAMPA

_____

Prof. Dr. Érico Marcelo Hoff do Amaral
UNIPAMPA

_____

Prof. Dr. Fábio Luis Livi Ramos
UNIPAMPA

Assinado eletronicamente por **MILTON ROBERTO HEINEN**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 18/07/2023, às 11:06, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **FABIO LUIS LIVI RAMOS**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 18/07/2023, às 13:22, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **ERICO MARCELO HOFF DO AMARAL**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 18/07/2023, às 23:10, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

A autenticidade deste documento pode ser conferida no site https://sei.unipampa.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1187884** e o código CRC **807C7EA9**.

Referência: Processo nº 23100.010919/2023-83 SEI nº 1187884

Este trabalho é todo dedicado aos meus pais, pois é graças ao seu esforço que hoje posso concluir o meu curso.

AGRADECIMENTO

Aos meus pais, que me incentivaram nos momentos difíceis e compreenderam a minha ausência enquanto eu me dedicava à realização deste trabalho.

“A persistência é o caminho do êxito.”

— Charles Chaplin

RESUMO

A aplicação da Inteligência Artificial (IA) no setor financeiro, especialmente nas áreas de gestão de ativos e finanças, tem se tornado cada vez mais comum, devido à vasta disponibilidade de dados e ao acesso a poder computacional. No entanto, a falta de prática ou conhecimento nesse campo pode resultar em prejuízos para os envolvidos. O objetivo deste estudo é desenvolver um modelo, chamado de TradeClassifier, capaz de tomar decisões sobre o momento adequado para operar em *Long* ou *Short* em um ativo do mercado *Forex*, utilizando a estratégia *Inner Circle Trade* (ICT) e técnicas de Inteligência Artificial. O modelo tem a finalidade de auxiliar tanto os operadores inexperientes quanto os experientes. As técnicas de Aprendizado de Máquina (*Machine Learning*) utilizadas para classificação incluem *Suport Vector Machine* (SVM), Árvore de Decisão, Floresta Aleatória e Redes Neurais, que são amplamente empregadas em trabalhos acadêmicos para previsão e classificação. Optou-se pelo uso da linguagem de programação Python devido às bibliotecas disponíveis e à sua popularidade no setor financeiro e no desenvolvimento de projetos de IA. Foi adotada a abordagem de aprendizado supervisionado para a classificação, uma vez que o modelo recebeu uma base de dados do ativo EUR/USD (Euro x Dólar Americano) e classifica os momentos de operar. Os resultados de acurácia dos modelos obtiveram cerca de 55% de acurácia utilizando a técnica *GridSearchCV()* para otimização dos parâmetros. A matriz de confusão foi utilizada para analisar as classificações realizadas pelos modelos. Além disso, foram realizadas simulações de negociações com base nas previsões geradas por modelos que apresentaram resultados lucrativos, os modelos SVM e Redes Neurais. O SVM obteve 79,235.23% de lucro e o modelo de Rede Neural acumulou 63,450.64%.

Palavras-chave: Aprendizagem de Máquina. Inteligência Artificial. Mercado Financeiro. Forex. Matriz de Confusão.

ABSTRACT

The application of Artificial Intelligence (AI) in the financial sector, especially in asset management and finance, has become increasingly common due to the vast availability of data and access to computational power. However, lack of practice or knowledge in this field can result in losses for those involved. The objective of this study is to develop a model called TradeClassifier capable of making decisions about the appropriate time to operate in Long or Short on a Forex market asset, using the Inner Circle Trade (ICT) strategy and Artificial Intelligence techniques. The model aims to assist both inexperienced and experienced traders. The Machine Learning techniques used for classification include Support Vector Machine (SVM), Decision Tree, Random Forest, and Neural Networks, which are widely employed in academic works for prediction and classification. Python programming language was chosen due to the available libraries and its popularity in the finance sector and AI project development. A supervised learning approach was adopted for classification, as the model was given a database of the EUR/USD asset (Euro vs. US Dollar) and classifies the moments to operate. The model accuracy results reached approximately 55% accuracy using the GridSearchCV() technique for parameter optimization. The confusion matrix was used to analyze the classifications made by the models. Additionally, trading simulations were conducted based on the predictions generated by models that showed profitable results, namely the SVM and Neural Networks models. The SVM model achieved a profit of 79,235.23%, and the Neural Network model accumulated 63,450.64% profit.

Keywords: Machine Learning, Artificial Intelligence, financial market, Forex, confusion matrix.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
ACM	Association for Computing Machinery
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
RTP	Real-Time Protocol
RSSF	Rede de Sensores sem Fio
SIMD	Single Instruction Multiple Data
UNIPAMPA	Universidade Federal do Pampa
IA	Inteligência Artificial
ML	*Machine Learning*
LSTM	Long ShortTerm Memory
SVM	*Support Vector Machine*
RNA	Rede Neural Artificial
ICT	Inner Circle Trade
FOREX	Foreign Exchange Market
UFRGS	Universidade Federal do Rio Grande do Sul

SUMÁRIO

1 INTRODUÇÃO

Entre os países membros do G20, grupo que reúne as 20 maiores economias do mundo, o Brasil é o país que possui o quarto maior índice de inflação (BUSS; FERRARI, 2022). Investir em renda variável tornou-se uma forma de proteger o seu capital contra o processo inflacionário, entretanto com a falta de experiência e conhecimento, os operadores iniciantes podem perder o seu dinheiro.

O mercado financeiro é o ambiente onde ocorre a negociação de ativos, como ações, títulos, moedas, etc. As pessoas que operam naquele espaço são chamados de *traders*, que têm o objetivo de buscar alguma rentabilidade ou proteger o seu capital da inflação, fazendo a compra e venda de ativos, negociados na bolsa de valores (INFOMONEY, 2020). Para a decisão das compras e das vendas, utilizam-se várias ferramentas, técnicas e indicadores para buscar o lucro desejado pelo investidor (MARTINS, 2010). O Forex (*Foreign Exchange Market*) é uma opção de mercado onde são negociados derivados de moedas com a finalidade de proteger o seu capital, sendo um dos maiores mercados do mundo. Segundo Maskey (2021), estima-se que sejam transacionados, diariamente, contratos representando volume total entre 6,6 trilhões de dólares (FOREX, 2022).

Atualmente o uso de métodos computacionais para *traders* é indispensável, por reduzir o tempo de leitura dos dados e também pela facilidade de combinação de estratégias, tornando as operações possíveis de serem realizadas rapidamente.

Com o crescimento do uso de Inteligência Artificial (IA) em vários setores, como saúde, agropecuária, ensino e gestão, esta tecnologia, no mercado financeiro, foi adquirida para auxiliar as tomadas de decisão dos *traders* na compra e venda de ativos e também no uso de previsões. A aplicação de IA na área de finanças tem como finalidade auxiliar o investidor nas operações. Neste sentido pode-se apontar o serviço 3Commas[1] como um exemplo, visto que apresenta estratégias e *bots* automatizados para investimento em criptomoedas. Este trabalho terá como enfoque o uso de técnicas de IA para classificar os melhores períodos da compra e a venda dos ativos, visando a obtenção de lucro.

Para analisar momentos de compra e venda, os *traders* usam indicadores para auxiliar na operação (MARTINS, 2010). Os Indicadores, por sua vez, são dados indiretos calculados a partir do histórico de preços, metainformações utilizadas para fundamentar a decisão das operações. EMA (média móvel exponencial) é um exemplo de indicador

[1] https://3commas.io/

simples, que considera a média das últimas entradas. Porém, essa estratégia não é a mais usada. O sucesso das operações pode sofrer influência, dependendo do indicador empregado.

Existem vários tipos de estratégias direcionadas para as operações. Neste trabalho usaremos a técnica chamada ICT[2] (*Inner Circle Trade*), desenvolvida pelo Michael J. Huddleston. A referência da técnica e do desenvolvedor foi citada pelo site onde são apresentados vídeos didáticos sobre a estratégia. Essa técnica visa mapear os movimentos de grandes instituições bancárias no mercado. Para determinar o melhor momento para a compra e a venda dos ativos, esse método é a base do modelo, desenvolvido ao longo deste trabalho.

1.1 Problema de Pesquisa

Este trabalho visa apresentar uma alternativa para auxiliar na negociação no mercado Forex, destinada a operadores inexperientes, por meio da utilização de inteligência artificial e estratégias do mercado financeiro.

1.2 Objetivos

Este trabalho tem como objetivo geral desenvolver o estudo e a implementação de um modelo que classifique as operações no mercado financeiro, utilizando alguns conceitos baseados no ICT (*Inner Circle Trade*), além de técnicas de IA.

Pode-se apontar como objetivos específicos deste trabalho:

1. Estudar os conceitos e abordagens do mercado financeiro.
2. Revisar a literatura sobre o mercado financeiro utilizando IA.
3. Estudar a estratégia e os conceitos do ICT.
4. Estudar técnicas de IA que possam ser utilizadas para operar.
5. Definir as ferramentas que serão utilizadas para o desenvolvimento do modelo.

[2]http://www.theinnercircletrader.com/

1.3 Organização do trabalho

No Capítulo 1 são apresentados o uso da inteligência artificial em vários setores, e a demonstração de IA aplicado no mercado financeiro. O Capítulo 2 apresenta as fases do trabalho e as *Strings* que foram definidas para a revisão bibliográfica.

O Capítulo 3 traz a descrição dos conceitos sobre o mercado financeiro, onde são apresentados os conceitos sobre o ICT, as técnicas de IA que foram escolhidas e os trabalhos correlatados.

No Capítulo 4 são apresentadas a linguagem e as bibliotecas para o desenvolvimento do modelo proposto e a descrição do modelo final e a forma de calcular a precisão do modelo. O Capítulo 5 são descritos os testes realizados, simulações *trades* e os seus resultados.

Por fim, o Capítulo 6 traz as conclusões finais e perspectivas futuras do trabalho.

2 MATERIAL E MÉTODOS

Com os objetivos definidos, foi realizada uma busca de trabalhos relacionados para que, assim, haja aprofundamento dos conhecimentos sobre como o mercado financeiro funciona. O levantamento bibliográfico exploratório foi a metodologia adotada neste trabalho, utilizando o livro de Prodanov e Freitas (2013), definindo as *Strings* para serem utilizadas nos repositórios escolhidos de artigos online utilizando o *proxy* da Unipampa. Após o processo de seleção dos artigos, houve a filtragem da quantidade de material com base nos critérios de inclusão e exclusão.

Segundo os métodos do trabalho que foram selecionados, o trabalho está estruturado nas seguintes atividades:

1. Revisão sistemática da literatura a respeito do mercado financeiro e sobre técnicas de Inteligência Artificial que serão implementadas;
2. Entendimento sobre técnicas/configurações que os investidores aplicam no mercado financeiro;
3. Entendimento do modelo e das principais configurações do ICT;
4. Realização do planejamento do modelo, como as técnicas de IA utilizadas, a linguagem de programação e os pacotes/bibliotecas;
5. Desenvolvimento do modelo utilizando dados dos preços Forex;
6. Validação do modelo por meio de experimentos com dados históricos;
7. Escrita do texto do TCC I e TCC II.

Para auxiliar a construção das strings, foram formuladas questões relacionadas à pesquisa. As questões são referentes aos artigos que desejamos encontrar para responder às questões:

Q1 Quais *setups*/estratégias ou indicadores, aplicados no mercado Forex (Foreign Exchange Market)?

Q2 Quais técnicas/estratégias de IA são aplicadas no mercado financeiro?

Para a construção utilizamos dois termos, mercado financeiro e inteligência artificial. Para o mercado financeiro empregamos os termos *Smart Money*, *Order Blocks*, Forex, *Foreing Exchange*, *Trading*, *financial markets*, *imbalance forex trading*, *breaker blocks*, *supply and demand*, *technical analysis*, *fundamental analysis*, *price action* e *Inner Circle Trader*. Para inteligência artificial empregamos os termos *artificial intelligence*,

Tabela 1 – Termos relacionados

Termos	Português	Inglês
Mercado Financeiro	Mercado de câmbio; mercados financeiros; Negociação financeira.	Foreign Exchange Market; financial markets; Financial trading.
Inteligência Artificial	Inteligência Artificial; Aprendizado de Máquina; Redes neurais.	Artificial Intelligence; Machine Learning; Neural networks.
Estratégias de mercado	Setups; Smart Money; Análise Técnica; Análise fundamentalista.	Setups; Smart Money; Technical analysis; fundamental analysis.

Fonte: Autor (2022)

Tabela 2 – String de busca

String 1	(Foreign Exchange Market OR financial markets OR Financial trading) AND (Artificial Intelligence OR Machine Learning OR Neural networks) AND (setups OR Smart Money OR Technical analysis OR fundamental analysis)
String 2	(Mercado de câmbio OR mercados financeiros OR Negociação financeira) AND (Inteligência Artificial OR Aprendizado de Máquina OR Redes neurais) AND (setups OR Smart Money OR Análise Técnica OR Análise fundamentalista)
String 3	(Foreign Exchange Market OR financial markets OR Financial trading) AND (setups OR Smart Money OR Technical analysis OR fundamental analysis)
String 4	(Mercado de câmbio OR mercados financeiros OR Negociação financeira) AND (setups OR Smart Money OR Análise Técnica OR Análise fundamentalista)

Fonte: Autor (2022)

machine learning, *deep learning*, *neural networks* e *genetic algorithms*.

Na Tabela 1 são apresentados os termos em português e em inglês para serem aplicados em repositórios estrangeiros. O termo "estratégias de mercado" foi criado para encontrar trabalhos sobre os conceitos do ICT. Os termos *setups* e *Smart Money* são relacionados a estratégias de investimento, sendo palavras que não são usadas de forma traduzidas.

Na Tabela 2 são apresentadas as *strings* de busca, usadas nos repositórios de artigos Science Direct, IEEE Xplore e Google Scholar, usando o *proxy* da Unipampa (Universidade Federal do Pampa).

Para a primeira e a segunda *String*, temos como objetivo encontrar trabalhos, que estejam relacionados ao mercado financeiro e ao uso de IA nesta área. No

repositório Science Direct, colocamos as *Strings* de busca somente nos títulos, resumo e palavras-chave, e filtramos somente com trabalhos recentes desde 2010 e que fossem ligados à área de Ciência de Computação. Para a terceira e a quarta *String* temos como objetivo encontrar trabalhos que estejam relacionados ao mercado financeiro e às técnicas de investimento. No repositório Science Direct colocamos a *String* para selecionar no título, resumo e palavras-chaves. Como houve o retorno de 13 trabalhos, não foi necessário filtrar.

2.1 Critérios de inclusão e exclusão

Com as *strings* construídas e aplicadas nos repositórios online, com o intuito de cumprir os objetivos do trabalho, foram definidos os critérios de inclusão e exclusão: segundo a metodologia, servem para filtrar somente os artigos, que estão relacionados ao trabalho.
Critérios de inclusão:

- Serão inclusos os trabalhos sobre mercado financeiro.
- Serão inclusos os trabalhos desde o ano 2010, exceto caso tenha valor para o desenvolvimento do trabalho.

Critérios de exclusão:

- Serão excluídos todos os artigos encontrados que não forem escritos em idiomas português ou inglês.
- Serão excluídos os trabalhos repetidos.
- Serão excluídos os trabalhos anteriores dos mesmos autores.

Após definir os critérios de inclusão e aplicá-los, foi retornado uma grande quantidade de trabalhos científicos retornou para o repositório de IEE Xplore onde foi utilizado o filtro para os trabalhos desde o ano de 2015, para coletar trabalhos mais recentes. Porém, nos repositórios do Google Schoolar e ScienceDirect, aplicando somente as *Strings* foi feita a coleta desde o ano 2010 devido à quantidade de trabalhos retornados. Assim obtivemos trabalhos correlatos e artigos que auxiliam sobre os conceitos do mercado financeiro.

3 REVISÃO BIBLIOGRÁFICA

Para estudar as abordagens e os conceitos empregados no mercado financeiro, utilizamos artigos e livros selecionados a partir de uma pesquisa realizada em materiais especializados, que serviram para auxiliar no entendimento sobre as técnicas/estratégias e métodos usados pelos operadores (*traders*) profissionais.

3.1 Conceitos do mercado financeiro

Para haver melhor entendimento sobre o trabalho, esta seção descreverá os conceitos sobre o mercado financeiro. A palavra *trader* é o nome dado para o investidor, operador do mercado financeiro. O objetivo do *trader* é obter lucro em curto prazo, utilizando conhecimentos matemáticos e ferramentas computacionais. Os operadores que utilizam ferramentas computadorizadas obtêm uma grande vantagem, podendo processar mais informações e aplicando variedades de indicadores nos dados (ELDER, 2004).

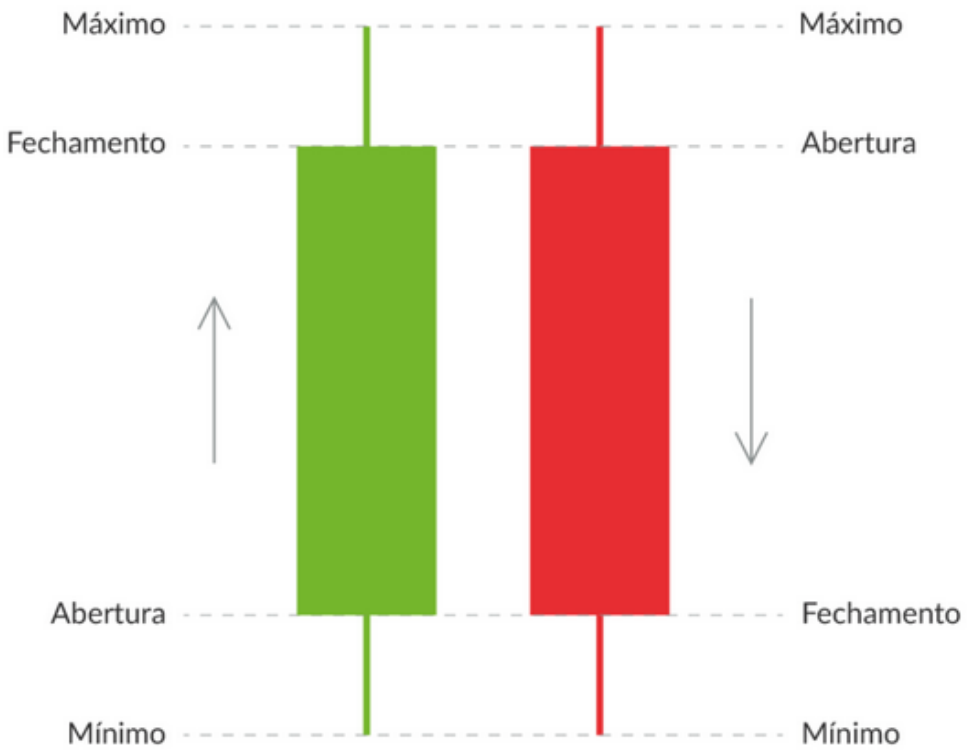
Existem vários tipos de estratégias em que os *traders* podem usar, das quais são utilizadas em períodos de curto, médio e longo prazos, dependendo do objetivo do indivíduo. O *Day trade* é uma negociação realizada em períodos curtos. Por isso, exige-se mais conhecimento e cuidado para o operador para concluir várias operações e obter lucros na soma. O *Swing trading* é uma estratégia de negociação de médio prazo, variando em semanas ou meses de negociação com um ativo. Esta técnica se diferencia pela menor quantidade de operações realizadas em comparação com o método *Day Trade*. A estratégia *buy and hold* (comprar e manter na tradução em português) se refere à compra de um ativo e sua respectiva venda depois de meses, ou até anos, em períodos de longo prazo. Durante esse tempo podem ter ocorrido flutuações do mercado, que vão ser ignoradas pelo *trader*, exceto quanto seja atingido o lucro desejado para a operação de venda do ativo.

Os preços dos ativos no mercado financeiro são geralmente visualizados com gráficos de velas ou em inglês chamados *candlestick*. Trata-se de um instrumento muito usado no mercado financeiro para acompanhar suas respectivas tendências, além de facilitar o reconhecimento do comportamento de um certo ativo.

Existem dois tipos de velas: quando o corpo é verde, o preço do ativo sobe, então a abertura – que seria o preço inicial – apresentaria menor valor situando-se na parte inferior da vela; e o fechamento (maior valor) é situado na parte superior da vela. Quando o preço

do ativo baixou, o corpo é da cor vermelha. Neste caso, a abertura e o fechamento são invertidos; os pavios representam os valores mínimo e máximo que os preços atingiram. Os dois tipos de velas e os seus componentes são apresentados nas Figura 1 e 2, onde há a exibição do gráfico de velas de Euro para Dólar do mercado Forex.

Figura 1 – Tipos de velas



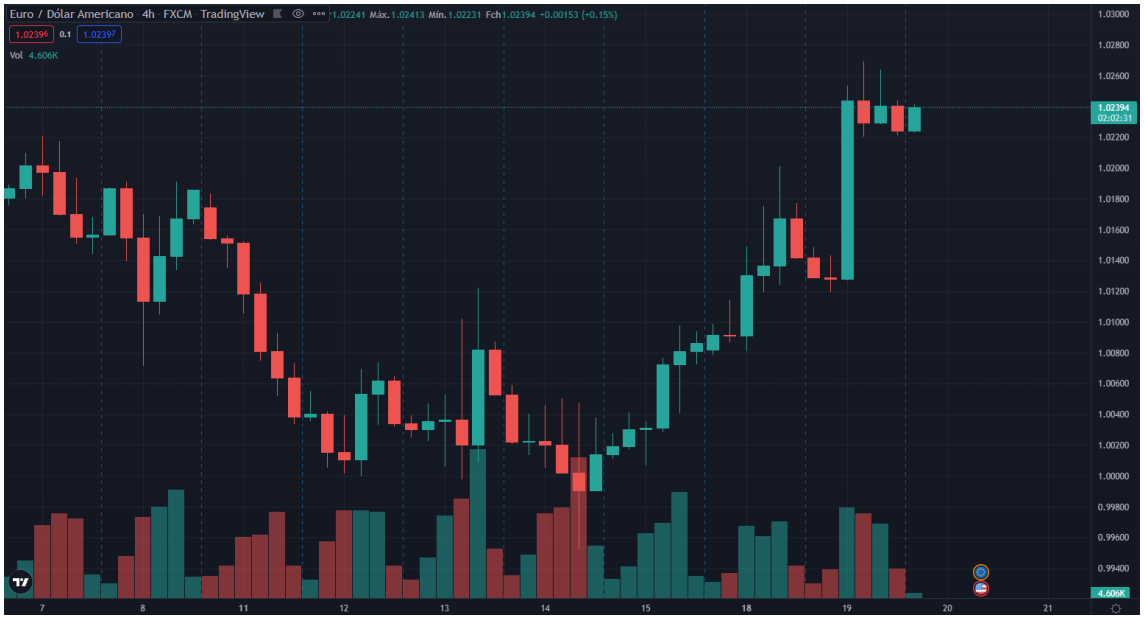
Fonte: (BIANCA, 2021)

Com a descrição dos conceitos básicos do mercado financeiro, sobre os tipos de operadores, e principalmente sobre o gráfico de velas, os dados que foram importados, contém os parâmetros do gráfico de *candlestick* da moeda EUR/USD (Euro x Dólar Americano), negociados no mercado Forex.

3.1.1 Forex

O Forex é umas dos mercados financeiros mais importantes do mundo, destinado a transações de câmbio, contendo uma movimentação de negociação de 6,6 trilhões de dólares diários (MASKEY, 2021). A negociação dos ativos é efetuado em pares,

Figura 2 – Gráfico de velas de Euro para Dólar



Fonte: Autor (2022)

observando o valor da moeda comparada com a outra. Por exemplo, na negociação de EUR/USD, é realizado a compra do euro, e para a venda, é esperado que o preço suba em relação ao dólar.

Para os *traders* operarem no mercado Forex, necessita-se de um capital mínimo para negociar. A Alavancagem, por sua vez, diz respeito a um tipo de empréstimo, possibilitando o *trader* negociar lotes de grande quantidade. Uma alavancagem de 1:10, por exemplo, significa que o *trader* precisa entrar com somente 10% do valor da operação como margem, e os 90% restantes são emprestados pela corretora. Em geral, altas alavancagens envolvem risco maiores, e portanto, aconselha-se usar alavancagem abaixo de 1:100.

Quando ocorre a operação de uma compra de ativo no mercado Forex, o valor é comprado com um preço mais alto e na hora da venda com o preço mais baixo. Isso ocorre para que os corretores ganhem a comissão. Esses dois tipos de preço praticados no Forex são o *Bid* e *Ask*. *Bid* seria a oferta destinada para o preço da compra, enquanto o *Ask* para o preço da venda. A tarifa que a corretora (*Broker*) ganha é calculada a partir da diferença entre o *Ask* e o *Bid*, que é chamada de *Spread*. Este valor é variável dependendo da corretora e da demanda e a oferta. Entretanto existe o Spread fixo, que é usado mais para operadores iniciantes.

Pips são usados para medir o movimento no mercado Forex, como, por exemplo,

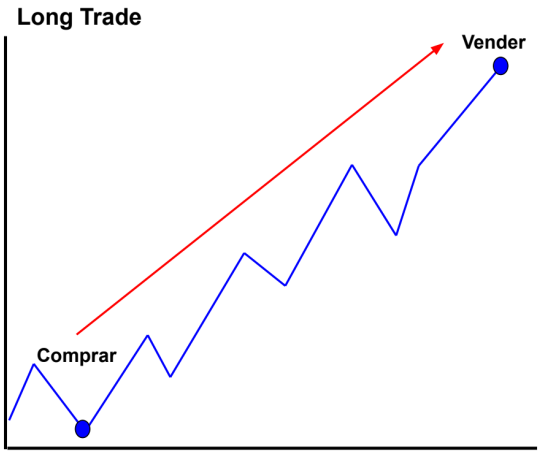
com o uso padrão EUR/USD. Com esse ativo, imagine que o preço mais recente seja 1,2052 insinuando que, com 1 Euro podemos comprar 1,2052 dólares, se o *trader* conseguir prever que o Euro ficará mais forte e vender por 1,2080, então ele obterá um lucro de 28 *pips*, mas se a previsão for errada e o preço cair para 1,2022, logo, terá prejuízo de 30 *pips*.

Tamanho do lote é a unidade monetária ou posição executada na negociação. Diferente no mercado acionário, cuja venda se dá por meio de ações, no mercado Forex a compra é realizada por lote, normalmente, um lote contém 100.000 unidades da moeda escolhida. Existem outros lotes, tais como minilotes, com 10.000 unidades, o microlote, com 1.000 e nano-lote, com 100 unidades.

A obtenção se lucro nem sempre é garantida quando se investe em renda variável, podendo gerar prejuízo ao operador. Para evitar esse risco, existem mecanismos para ajudar os *traders*, como o *Stop Loss* que consiste em uma ferramenta configurada pelo *trader* para programar um limite de perda caso haja desvalorização do ativo selecionado, quando o ativo atingir o limite, é vendido automaticamente, evitando assim um prejuízo maior. O *Take Profit* é outra ferramenta utilizada com o objetivo de estabelecer um limite para o lucro, caso haja valorização do ativo negociado. Assim, quando atingir o limite, o ativo é vendido automaticamente e garante-se o lucro antes de ocorrer uma desvalorização.

Na área financeira, os termos *long trade* e *short trade* são utilizados para descrever as posições que os investidores assumem em relação a um ativo específico (TORO, 2023).

Figura 3 – Long Trade



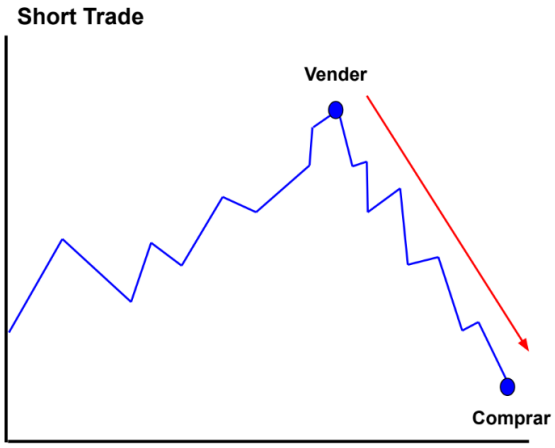
Fonte: Autor (2022)

De acordo com Toro (2023), quando um investidor toma uma posição *long trade*,

significa que ele adquire o ativo com a expectativa de que seu valor aumente ao longo do tempo. Em outras palavras, o investidor está comprando o ativo com a intenção de vendê-lo posteriormente a um preço mais alto, buscando obter lucro com a valorização conforme a Figura 3. Por exemplo, se um investidor está *long trade* em ações de uma determinada empresa, ele é proprietário dessas ações e espera que o valor delas se valorize no futuro.

Por outro lado, uma posição *short trade* envolve a venda de um ativo que o investidor não possui, com a perspectiva de que o seu valor diminua (TORO, 2023). Nesse caso, o investidor está "apostando" que o preço do ativo irá cair. Posteriormente, ele recomprará o ativo a um preço mais baixo para lucrar com a diferença entre o preço de venda inicial e o preço de recompra, conforme a Figura 4. Por exemplo, se um investidor está *short trade* em ações de uma determinada empresa, ele vendeu essas ações mesmo sem possuí-las, antecipando-se a uma queda de preço para comprá-las de volta por um valor menor e obter lucro.

Figura 4 – Short Trade



Fonte: Autor (2022)

Em resumo, a posição *long* envolve a compra de um ativo com a expectativa de valorização, enquanto a posição *short* envolve a venda de um ativo com a expectativa de desvalorização.

Na Tabela 3 é descrita os termos apresentados nesta seção sobre o mercado Forex.

Tabela 3 – Termos Forex

Termos	Descrição
Ask	é o preço pelo qual um vendedor está disposto a vender um ativo financeiro
Bid	é o preço pelo qual os compradores estão dispostos a comprar o ativo.
Spread	é uma medida da liquidez e da eficiência do mercado financeiro, sendo a diferença entre o preço de compra (*bid*) e o preço de venda (*ask*)
Take Profit	é um termo utilizado no mercado financeiro, para se referir a uma ordem pré-definida para fechar uma posição aberta quando o preço atinge um determinado nível de lucro desejado.
Stop Loss	é um termo utilizado no mercado financeiro, para se referir a uma ordem pré-definida para fechar uma posição aberta quando o preço atinge um determinado nível de perda aceitável.
Long	é uma estratégia utilizada no mercado financeiro para descrever uma posição em que um *trader* compra um ativo com a expectativa de que seu preço aumente ao longo do tempo. É uma estratégia na qual o *trader* busca lucrar com a valorização do ativo.
Short	é uma estratégia utilizada no mercado financeiro em que um trader vende um ativo que não possui (vendendo emprestado) com a expectativa de que seu preço diminua no futuro. É uma maneira de lucrar com a queda do valor de um ativo.
Alavancagem	é o uso de capital emprestado para aumentar o potencial de retorno de um investimento. É uma estratégia que permite que os investidores operem com um montante maior de recursos do que o capital próprio disponível.
Lotes	é uma unidade de medida padronizada para transações de ativos, como moedas, ações, commodities e outros instrumentos financeiros. Um lote representa uma quantidade específica de um determinado ativo que está sendo negociado.
Pip	é uma unidade de medida comumente utilizada no mercado Forex para representar a variação mínima no preço de um par de moedas.

Fonte: Autor (2023)

3.1.2 Análise técnica

A análise técnica corresponde ao estudo sobre o gráfico dos valores de um ativo, tentando encontrar padrões e tendências, sendo desse modo, a forma de análise mais comum quando há operação pelo *trader* no mercado Forex. As tendências de aumento e queda dos preços são indicadas por padrões. Segundo o livro Martins (2010), para ocorrer o estudo do gráfico, é necessário estabelecer um das cotações, como meses, semanas,

dias ou até em horas. Após estabelecer o período, monta-se o gráfico do tipo *candle*, atualmente o mais usado, que será analisado. A base da análise técnica, foi iniciada com a Teoria de Dow.

3.1.3 Teoria de Dow

A análise técnica moderna se iniciou com Charles Henry Dow, chamada de Teoria de Dow, que analisa os preços históricos de um ativo (MARTINS, 2010). Essa teoria estabelece períodos de longo prazo, como semanal ou mensal, para identificar tendências. Segundo Martins (2010), a teoria possui algumas regras:

1. Os índices e preços já descontam tudo: O mais elementar princípio é que os índices e preços já descontam tudo, isto é, já refletem que o mercado tem conhecimento, como notícias, decisões das empresas, resultados contábeis, etc. Além disso, os novos eventos são rapidamente incorporados, tais como os preços, que influenciam as tendências do ativo analisado.
2. O mercado se move em ondas e apresenta três tendências principais. O estudo das tendências é um dos pontos principais da teoria. Segundo Dow, o mercado sempre apresenta três tipos de tendências principais: a primária (como as marés, as mais duradouras); a secundária (como as ondas, de médio prazo); e as terciárias (marolas, de curto prazo).Falaremos sobre elas com exemplos mais adiante.
3. O mercado sempre está classificado com uma tendência, seja ela alta ou baixa.
4. A tendência deve ser confirmada por dois índices: quando se analisa ações de uma empresa, deve ser utilizada as ações das maiores empresas do mesmo setor e a confirmação da tendência utilizando as ações da empresa.
5. Os volumes influenciam nas tendências dos ativos: o volume e nada , quando a tendência está de alta, o volume é alta, e quando a tendência é de baixa, o volume é baixa.
6. A tendência será mantida até que os sinais de reversão sejam confirmados: A fim de fugir dos ruídos do mercado acionário, Dow considerou alguns princípios para definir uma mudança concreta de tendência. Segundo os estudos, somente o fechamento do preço acima de um topo ou fundo anterior caracterizaria uma mudança de tendência.

Conforme o livro Martins (2010), as tendências de alta e de baixa são classificadas

em: Fases do mercado de alta:

Acumulação: é a fase depois do fim da tendência de baixa, formando uma consolidação, não ocorrendo nenhuma nova miníma.

Alta Sensível: Os investidores percebem uma inversão de tendência, provocando grande quantidade de compra, fazendo assim o preço do ativo crescer.

Euforia: nesta fase é a última fase de tendência de alta, normalmente, os novos investidores, compram esses ativos acreditando que o preço irá continuar a subir.

Fases do mercado de baixa:

Distribuição: está fase ocorre pelo final de tendência de alta, ocorrendo vendas constantes.

Baixa Sensível: está fase ocorre a inversão da tendência de alta, fazendo com que ocorra vendas dos ativos, desvalorizando as cotações, gerando novas mínimas.

Desespero: Após a fase de Baixa Sensível, desvalorização do ativo, os demais operadores começam a vender visando minimizar o prejuízo.

Figura 5 – Fases das tendências

Fonte: Autor (2022)

Na Figura 5 é apresentado um gráfico semanal, podendo identificar as fases da Teoria de Dow. As primeiras fases são identificadas a tendência de alta, com fases de acumulação, onde as cotações são consolidadas, onde anteriormente, estava tendo uma tendência de baixa. Em seguida ocorre a Alta Sensível, ocorrendo a valorização do preço do ativo e por fim a fase da Euforia, onde ocorre a última tendência de alta. As fases de baixa começa na Distribuição, equivalente à fase de acumulação, mas focado em área de vendas dos ativos. A fase Baixa Sensível, quando ocorre o começo da desvalorização do ativo, ocorre a venda dos ativos, evitando prejuízo maior, e por fim a fase de Desespero, onde há uma grande desvalorização pela quantidade de vendas dos ativos.

Existem padrões nos gráficos em que são negociados e conceitos principais da análise técnica que serão apresentados a seguir:

Utilizado por *traders* no mercado cambial, Suporte e Resistência é o conceito de análise técnica considerado mais comum utilizado por *traders* no mercado cambial. Os operadores usam essa observação do gráfico para identificar o nível desses elementos para estudar e compreender o comportamento do preço dos ativos. A zona de suporte normalmente ocorre quando o ativo é apresentado com o valor mais baixo. Em geral, costuma ser mais procurado pela tentativa de comprar e vender mais caro. Entretanto, torna-se um problema a partir do momento em que o suporte corre o risco de ser quebrado, prejudicando os operadores que compraram na zona de suporte. A zona de resistência é localizada no novo topo, onde normalmente não ocorre compra do ativo e sim a venda dele, proporcionando em seguida uma tendência de queda. Entretanto, zonas de resistência não são definitivas iguais ao suporte, se o ativo romper a resistência devido à alta do preço, o nível da resistência torna-se o nível do suporte. Então o suporte é o inverso da resistência, se ocorre o rompimento, o suporte se torna a nova resistência ou a resistência torna-se o novo suporte. Na Figura 6 é apresentado um ativo com uma tendência de alta, e com o passar do tempo, o nível de resistência é rompida, tornando-se um nível de suporte.

As linhas de tendência são traçadas para indicar em um gráfico a tendência de alta ou de baixa, o preço atinge um fundo e um topo, com comportamento de zigue-zague. Traça-se a linha de tendência de alta sempre na zona de suporte no gráfico, conforme apresentado na Figura 7, enquanto a de baixa é traçada na zona de resistência.

O padrão *Head and Shoulder*, ou em português "cabeça e ombro", é visualizado no gráfico quando existe uma quebra de tendência (MASKEY, 2021). O gráfico apresenta três picos, o meio é o pico mais alto que é chamado de *Head* (cabeça), e o primeiro

Figura 6 – Resistência e Suporte

Fonte: (SANTOS, 2020)

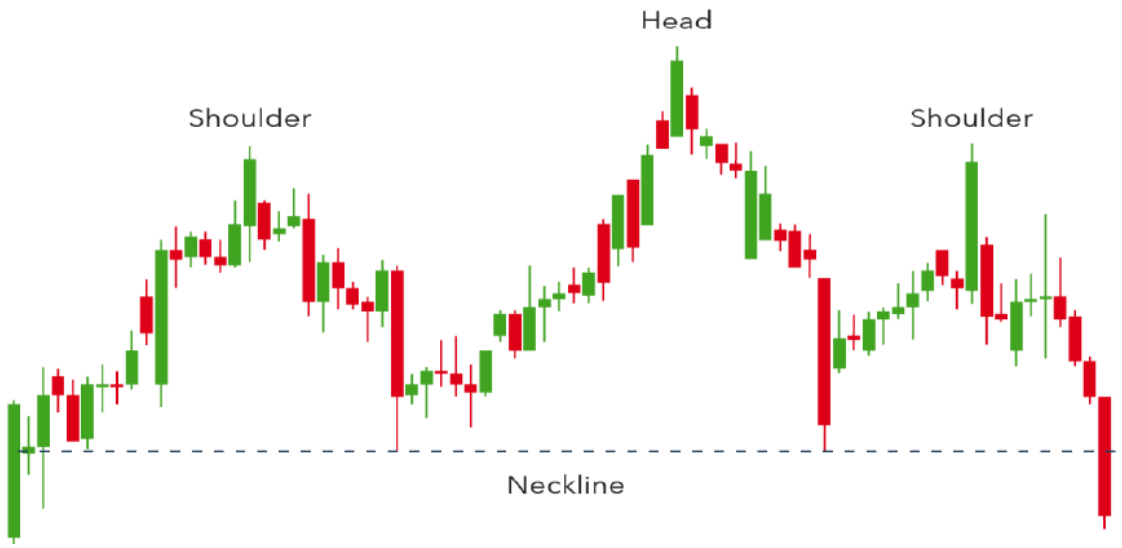
Figura 7 – Linha de Tendência

Fonte: (MUNIZ, 2019)

e o último, que são semelhantes, chamados de *Shoulder* (ombros). Visualmente é uma representação de reversão de tendência de alta para baixa, quando o terceiro pico passa do *neckline*, à qual se trata de uma linha que indica a inversão, sinalizando o fim da tendência

de alta, apresentada na Figura 8.

Figura 8 – Gráfico de Cabeça e Ombro



Fonte: (MASKEY, 2021)

3.1.4 Indicadores

Os Indicadores são outra forma de analisar gráficos do mercado financeiro usando técnicas matemáticas, verificando as chances de lucratividade. Os principais indicadores/ferramentas usados no mercado financeiro são Médias Móveis, MACD (*Moving Average Convergence/Divergence*), RSI (*Relative Strenght Index*), Estocástico e Bandas de Bolliger.

3.1.4.1 Médias móveis

A Média móvel é um indicador usado para medir o valor médio do preço, representado por uma linha no gráfico do ativo. Usado para facilitar a visualização dos movimentos do preço, retirando os ruídos, representados pelas variações mais fortes. Com isso o comportamento do preço de um ativo torna-se mais simples e claro de entender. As médias são calculadas com base nos preços de fechamento (*Close*) do ativo, podendo assim, facilitar a observação das tendências em períodos curtos, médios ou longos. As principais Médias móveis são a Simples (MMS) e a Exponencial (MME). A Equação 1 apresenta o cálculo da média móvel simples, e a Equação 2 representa o cálculo da média

móvel exponencial.

$$MMS = \frac{P_1 + P_2 + ... + P_N}{N} \tag{1}$$

Em que:

P = Preço do que será calculado a média.

N = Número de dias da média móvel.

$$MME = P_{hoje} \times K + MME_{ontem} \times (1 - K) \tag{2}$$

Em que:

$K = \frac{2}{N+1}$

$P_{hoje} =$ Preço de hoje.

$MME_{ontem} =$ MME de ontem.

Segundo Martins (2010), embora o resultado das duas médias sejam semelhantes, a média móvel exponencial se aproxima mais do último preço do gráfico do ativo, conforme é apresentado na Figura 9, sendo que a linha pontilhada é representada pela média exponencial. Como a média móvel apresenta um tempo de resposta menor, torna a média móvel mais utilizada quando o objetivo é gerar um peso maior nos últimos dias de um período. Contudo, isso não indica que uma média é melhor do que a outra. Dependendo do gráfico onde o indicador for aplicado, pode-se apresentar resultados melhores com um tipo da média móvel, por isso, recomenda-se utilizar os dois tipos para concluir qual indicador deve ser usado.

Existem situações em que são abordadas médias móveis de diferentes períodos, e seus respectivos cruzamentos indicam possíveis direções das determinadas tendências, o que para Martins (2010) nos leva a concluir duas regras de operação: a primeira nos diz que quando uma média móvel mais curta cruza a mais longa para baixo, ocorre um sinal de tendência de baixa e quando a média móvel mais curta cruza a mais longa para cima, é tendência de alta. Na Figura 10 podemos observar essas regras, depois do dia 19/04, quando as médias cruzam, é indicado uma tendência de baixa, o que acaba acontecendo.

Com as tendências identificadas, também podemos determinar os momentos de compra e de venda dos ativos, quando as linhas cruzam e sinalizam uma tendência de alta, podemos realizar a compra, e quando as linhas cruzam sinalizando uma tendência de baixa, é realizada a venda.

Figura 9 – Média Móvel Simples e Exponencial

Figura 10 – Média Móvel tendências.

Quando determinamos a quantidade de períodos/*candles* (velas) que será utilizada no indicador, devemos considerar os objetivos e finalidades das médias. Uma média móvel com mais períodos/*candles* é mais distante quando comparada com uma média

móvel que envolva períodos/*candles*, porém os objetivos das mesmas podem demandar períodos mais longos, por exemplo. Lemos (2018) é indicado que as médias móveis curtas são mais sensíveis ao preço quando comparadas com as longas, mas isso não torna uma mais eficaz que a outra, onde mercados possuem demandas e eficiências diferentes.

Por padrão de literatura, são geralmente escolhidos no mínimo duas velas/*candles* para os cálculos das médias, onde um é mais curto do que o outro para realizar as estratégias de cruzamento entre ambas as linhas. Alguns padrões de acompanhamento de tendência classificam os períodos de ajuste das médias móveis conforme a tendência a ser analisada no movimento, são alguns padrões apontados pelo autor supracitado:

- Curtíssimo prazo: a quantidade de velas/*candles* varia entre 5 e 20.
- Curto prazo: a quantidade de velas/*candles* varia entre 10 e 30.
- Médio prazo: a quantidade de velas/*candles* varia entre 15 e 60.
- Médio-longo prazo: a quantidade de velas/*candles* varia entre 45 e 120.
- Longo prazo: a quantidade de velas/*candles* varia entre 120 e 250.

Além das médias comuns, o indicador também conta com algumas médias especiais que acabam envolvendo parâmetros diferentes, e uma delas e a Média móvel ponderada por volume (MMPV). A média MMPV, ou em inglês, *Volume Weighted Average Price* (VWAP), é utilizada para ponderar os preços pelos volumes negociados, podendo obter respostas mais rápidas, sinais mais confiáveis e retorno melhores conforme Lemos (2018). Para se obter a MMPV é preciso dos dados dos preços e dos volumes, conforme é apresentado na Equação 3.

$$MMPV = \frac{\sum(P_i \times V_i)}{V_t} \tag{3}$$

- $MMPV =$ Média móvel ponderada por volume.
- $P_i =$ Preço do *Close* da posição i.
- $V_i =$ Volume da posição i.
- $V_t =$ Volume total.

3.1.4.2 MACD

De acordo com Martins (2010), o indicador MACD (Moving Average Convergence/Divergence) utiliza duas linhas. Para formar a primeira linha é usada a Equação 4, sendo a diferença entre duas médias móveis exponenciais (MME), uma rápida

com 12 períodos/*candles* e a outra lenta de 26 períodos/*candles*.

$$MACD = MME_{12} - MME_{26} \tag{4}$$

Em que:

- $MACD = Moving\ Average\ Converge/Divergence$.
- MME_{12} = Média Móvel Exponencial de 12 períodos do valor de fechamento.
- MME_{26} = Média Móvel Exponencial de 26 períodos do valor de fechamento.

Por fim, para formar a segunda linha, chamada de Linha de Sinal, é realizada uma Média Móvel Exponencial de nove períodos da linha MACD. Conforme Martins (2010), o valor de nove períodos é o mais utilizado, porém, assim como os valores das médias, pode ser alterado segundo a abordagem/estratégia utilizada. A Equação 5 representa a fórmula do sinal utilizada no MACD.

$$Sinal = MME_{9doMACD} \tag{5}$$

Em que:

- $Sinal$ = Linha Sinal.
- $MME_{9doMACD}$ = Média Móvel Exponencial de 9 períodos da linha MACD.

A Figura 11 apresenta as linhas do MACD. No gráfico, observa-se que as MMEs de 12 períodos/*candles*, representado na linha contínua, e 26 períodos/*candles*, representado com a linha pontilhada. Não é necessário projetar essas duas linhas no gráfico, pois elas foram traçadas apenas para o entendimento do estudo. No gráfico da parte inferior, estão traçados o MACD e seu Histograma, formado pelas barras pretas, a linha MACD contínua, enquanto a linha Sinal é a pontilhada. Pode-se verificar também as tomadas de decisão para as operações, como o momento de compra e de venda.

Como o MACD identifica as mudanças de tendências de um ativo, Martins (2010), descreve 3 formas de como operar com esse indicador:

- Quando a linha MACD cruza a linha zero para cima ou para baixo, a resolução se assemelha ao comportamento de corte de uma MME mais curta em uma MME mais longa.
- A regra de operação mais comum do MACD ocorre quando a linha MACD cruza a linha de sinal, onde caso a linha MACD cruze a linha de sinal para cima, temos um

Figura 11 – MACD

indicativo de compra. Caso a linha MACD cruze a linha de sinal para baixo, temos uma venda.

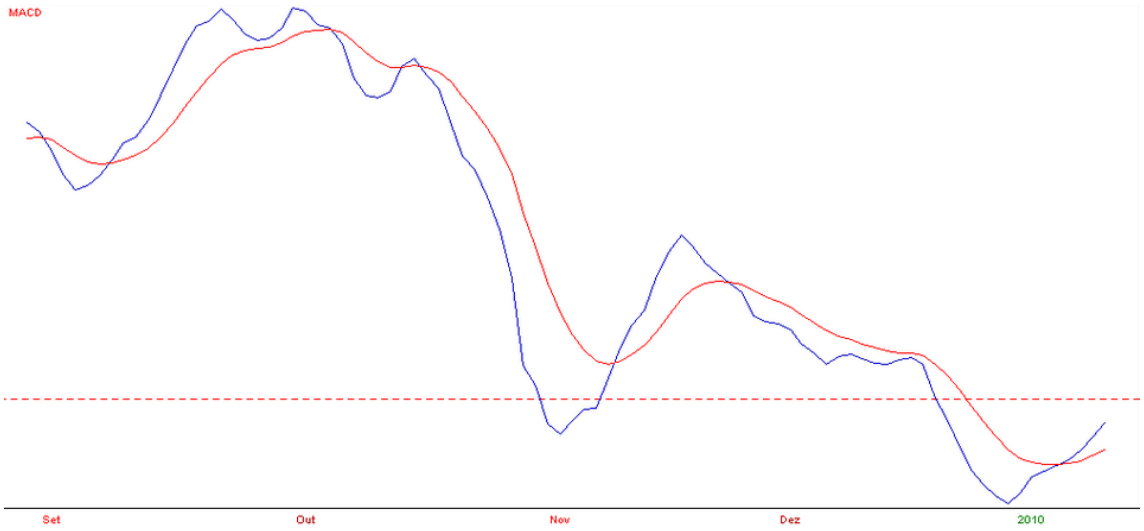
- O terceiro modo de operação envolve o Histograma MACD, que se caracteriza por ser a diferença entre a linha MACD e a linha de sinal conhecido como um dos sinais mais fortes de compra e venda da análise técnica, o que o transforma em um indicador raro.

Com essas regras basta analisar o gráfico apresentado pela Figura 12, a linha vermelha é a Linha de sinal e a azul é o resultado das diferenças das médias. Quando a linha de sinal ultrapassa a linha de MACD (linha azul) para cima no gráfico, é uma indicação de que o preço terá uma tendência de alta, apropriado para a compra.

3.1.4.3 Histograma MACD

O cálculo do Histograma MACD é resultado da diferença entre as linhas MACD e Sinal. Quando as linhas cruzam, o Histograma é zero; quando a Linha MACD está acima da Sinal, ele é formado por barras positivas; quando as linhas se invertem, ele é formado por barras negativas (MARTINS, 2010). A fórmula do indicador é apresentada conforme a Equação 6:

Figura 12 – Indicador MACD



Fonte: (XPE, 2022)

$$H_{MACD} = MACD - Sinal \tag{6}$$

Em que:

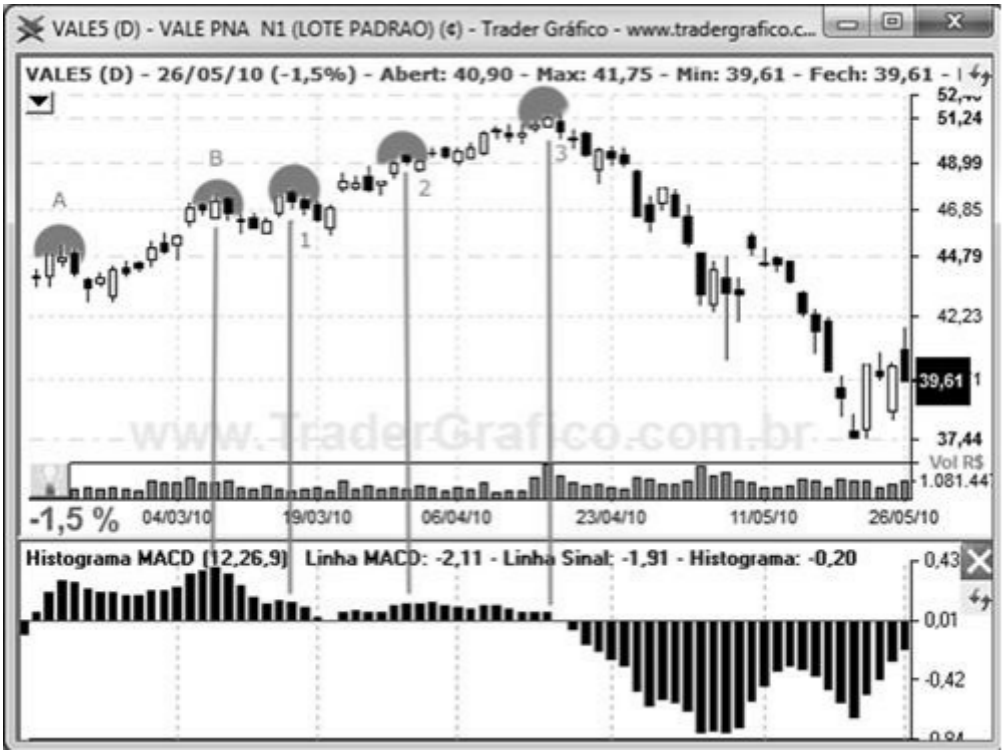
- H_{MACD} = Histograma MACD
- $MACD$ = Moving Average Convergence/Divergence
- $Sinal$ = Linha de Sinal

No Histograma MACD conseguimos operar encontrando divergências entre os preços e o estudo, possibilitando o operador de identificar fortes sinais de possíveis reversões de tendência. Podemos notar um exemplo na Figura 13, em que os topos A e B têm Histograma de MACD mais altos, porém nos 3 topos a seguir, representados por 1, 2 e 3, têm o histograma mais baixo. Analisando o gráfico de cima, nos topos cada vez mais altos, formaria uma oportunidade de compra, porém analisando o gráfico de Histograma MACD, observa-se uma perda de força nos topos 1,2 e 3, advertindo uma reversão de tendência próxima. Depois do topo 3, é confirmada a reversão de tendência.

3.1.4.4 Índice de Força Relativa

O RSI (*Relative Strength Index*) ou em português IFR (Índice de Força Relativa) é um indicador para rastrear a tendência de preço interpretando a força de um ativo, que seria a relação de compra e vendas. A função desse indicador é avaliar o enfraquecimento

Figura 13 – Histograma MACD



Fonte: (MARTINS, 2010)

de tendência, inversões e taxas de variação do preço. O indicador mede a aceleração do movimento dos preços de determinado ativo e mostra quando a velocidade de uma tendência começa a diminuir até mudar de direção. Medida pelo conceito de Força Relativa, esta aceleração monitora as mudanças nos preços de fechamento. Seu acompanhamento muitas vezes possibilita observar o enfraquecimento de uma tendência, bem como os rompimentos de suportes e resistências antes de se tornarem visíveis nos gráficos de preço. O IFR pode ser calculado com a Equação 7, o resultado varia de 0 a 100

$$IFR = 100 - (\frac{100}{1 + (\frac{U}{D})} \tag{7}$$

Em que:

- $IFR =$ Índice de Força Relativa.
- $U =$ Média de todas as variações positivas no preço no período em estudo.
- $D =$ Média de todas as variações negativas no preço no período em estudo.

De acordo com Martins (2010), se o resultado for mais de 80%, a ação entra em uma área de risco, chamada de sobre-compra, ela pode sinalizar reversão da tendência

em curso, a reversão ocorre quando a ação sai da área de sobre-compra e não quando ela entra. Outra zona de análise está na faixa abaixo de 20%, chamada de sobre-venda, onde o sinal é oposto, sinalizando reversão para alta dos preços. Na Figura 14, observa-se o IFR traçado embaixo do gráfico, com linhas indicado as áreas de risco, a vermelha para sobre-compra e a verde para sobre-venda.

Figura 14 – Índice de Força Relativa

Fonte: (MARTINS, 2010)

3.1.4.5 Estocástico

O Indicador é similar ao IFR, porém o estocástico tem 2 linhas: estocástica rápida e lenta. O resultado dela varia entre 0 e 100% , as áreas de sobrevenda é entre 0 a 30%, e a de sobrecompra e de 70% a 100% (MARTINS, 2010). A fórmula que representa o indicador estocástico rápido é a Equação 8.

$$\%K = \frac{(P_{FechaHoje} - Min_n)}{(Max_n - Min_n)} \times 100 \tag{8}$$

Em que:

- $\%K =$ Estocástico rápido.

- $n =$ Número de dias/barras do estocástico
- $P_{FechaHoje} =$ Último valor de fechamento (por padrão do dia de hoje).
- $Max_n =$ A máxima mais alta em um número predeterminado de barras.
- $Min_n =$ A mínima mais baixa em número predeterminado de barras.

O estocástico lento é aplicado em ser mais voltado para uma largura mais curta, reduzindo o estocástico rápido pelo padrão das últimas três barras do gráfico, como pode ser visto na Equação 9.

$$\%D = \frac{\sum_3(P_{FechaHoje} - Min_n)}{\sum_3(Max_n - Min_n)} \times 100 \tag{9}$$

Em que:

- $\%D =$ Estocástico lento.
- $\sum_3 =$ Soma das últimas três barras.
- $n =$ Número de dias/barras do estocástico
- $P_{FechaHoje} =$ Último valor de fechamento (por padrão do dia).
- $Max_n =$ A máxima mais alta em um número predeterminado de barras.
- $Min_n =$ A mínima mais baixa em número predeterminado de barras.

Na Figura 15, são apresentados os dois gráficos estocásticos, a rápida e a lenta, com áreas de sobrevenda acima de 75% e sobrecompra abaixo de 25%. Pode ser observado que o estocástico rápido, em comparação com a lenta, é mais sensível às variações dos preços do ativo. Como o indicador é uma rastreador de tendências, as operações podem ser classificadas, quando a linha cruza a reta de sobrecompra de baixo para cima, definido como momento de compra, enquanto cruza a reta de sobrevenda de cima para baixo, classificado como momento de venda (MARTINS, 2010).

3.1.4.6 Bandas de Bollinger

As Bandas de Bollinger são classificadas como um indicador avançado por Martins (2010). Para acompanhar os preços, são usadas 2 fórmulas estatísticas: a média (simples ou exponencial) e desvio padrão. Para obter o resultado do indicador, necessita-se obter a média móvel simples dos preços de fechamento do gráfico e o desvio padrão de uma certa quantidade de velas/*candles*. O valor encontrado deve ser multiplicado por dois e somado com a média móvel, formando uma banda para cima. Então, faz-se uma subtração da média móvel, formando uma banda para baixo.

Figura 15 – Estocástico

Fonte: (MARTINS, 2010)

Com essas duas bandas e a média móvel simples são formadas as Bandas de Bollinger. Martins (2010) diz que é esperado que 95% do preço mantenha-se na faixa estipulada, por questão estatística. Na Figura 16, a linha escura é a média móvel simples de 20 velas/*candles*, e as linhas claras são as faixas das Bandas de Bollinger. Quando ocorre o estreitamento das Bandas, mudanças bruscas tendem a ocorrer, quando os preços saem das bandas (faixas), a tendência atual é mantida e quando os topos são fundos ocorrem fora das bandas, seguido por topos e fundos internos, as bandas, indicadas por reversões de tendência.

3.1.4.7 Average True Range

ATR (*Average True Range*) é um indicador técnico usado no mercado financeiro para medir a volatilidade de um ativo financeiro. Ele foi desenvolvido por J. Welles Wilder Jr. e é amplamente utilizado por *traders* e analistas para auxiliar na tomada de decisões de investimento (RESEARCH, 2023).

O calculo do indicadoe é medindo a variação entre os preços máximos e mínimos de um ativo ao longo de um determinado período. Geralmente, é usado um período de 14 dias, mas esse valor pode ser ajustado de acordo com as preferências do analista.

Figura 16 – Bandas de Bollinger

A Equação 10 apresenta o cálculo do TR (*True Range*), que é a medida da variação máxima entre três valores: o valor máximo (*High*) menos o valor mínimo (*Low*) do período atual, o valor máximo (*High*) menos o fechamento (*Close*) anterior e o valor mínimo (*Low*) menos o fechamento (*Close*) anterior

$$TR = Max[(H-L), Abs(H-Cp), (Cp-L)] \tag{10}$$

O ATR é apresentado na Equação 11, que é uma média móvel do TR calculado em um determinado número de períodos. A média móvel suaviza os valores do TR para fornecer uma visão mais geral da volatilidade ao longo do tempo.

$$ATR = MME(TR) \tag{11}$$

O valor do ATR é expresso na mesma unidade de preço do ativo e representa a média das variações de preço observadas durante o período escolhido. Quanto maior o valor do ATR, maior é a volatilidade do ativo.

Os *traders* usam o ATR de várias maneiras. Neste trabalhos foi usado para determinar níveis de *stop loss* (limite de perda) e *take profit* (limite de ganho), levando em consideração a volatilidade atual do ativo. Além disso, o ATR também pode ser usado

como base para desenvolver estratégias de negociação, como a definição de níveis de entrada e saída do mercado.

3.1.5 Análise Fundamentalista

A análise fundamentalista é um estudo de informações sobre um ativo para projetar expectativas para o futuro e avaliar a compra do ativo, diferente da análise técnica que estuda os padrões históricos do preço (MASKEY, 2021). Usando este parâmetro com abordagem Forex, estudar-se as informações sobre a moeda que será negociada, como noticias de economia, fatores políticos, sociais e econômicos. Então, quando usar o método de análise fundamentalista, deve-se estar informado a respeito das notícias do país que adota aquela unidade monetária. Um exemplo é a desvalorização do EURO devido à Guerra da Ucrânia e as incertezas no fornecimento de energia para o continente, que geram temores de uma recessão europeia nos mercados.

3.1.6 *Smart Money*

O *Smart Money* é o dinheiro investido por bancos centrais, grandes bancos, instituições, especialistas de mercado, fundos de cobertura, empresas de gestão de investimentos e outros profissionais. *Smart Money* é a palavra derivada de termos de jogos de azar, onde é frequentemente referido como o dinheiro lucrado por jogadores com um bom histórico de sucesso. Era frequentemente associado a jogadores que costumavam ter uma ideia aprofundada e informações privilegiadas do jogo (BANTON, 2022).

O conceito é semelhante no mundo dos investimentos. *Smart Money* no Forex é muitas vezes referido como o capital investido com profundo conhecimento do mercado, que os *traders* operadores do mercado não podem abordar. Assim, considera-se que tem muito mais chance de sucesso do que a análise técnica de operadores do mercado (BANTON, 2022).

O *Smart Money* refere-se aos investidores mais inteligentes com uma enorme força econômica que pode causar impacto no mercado e mover o preço. Eles também são descritos como formadores de mercado. Esses *traders* têm volumes e dinheiro suficientes para causar uma mudança no mercado ao vivo. Bancos Centrais, grandes seguradoras e empresas globais são alguns dos exemplos por trás do*Smart Money*. Eles têm enorme

influência e são sempre poderosos no mercado financeiro (FOREXLENS, 2022).

Os bancos são os *traders* de *Smart Money* com volume maior. Os recursos que os bancos investem no mercado Forex geralmente representam mais de 60% do volume total do mercado. Eles têm informações detalhadas, conhecimento e uma equipe profissional que os ajuda a lucrar persistentemente com seus negócios. O *Smart Money* tem uma enorme quantidade de capital para negociar. Devido a seus grandes pedidos, suas ações não passam despercebidas (MASKEY, 2021).

Bancos e instituições são os principais *traders* do mercado Forex. A quantidade de bancos e instituições que negociam Forex é muito maior em comparação com os *traders*. Seu enorme volume de negociações pode forçar a mudança de direção no mercado. O *Smart Money* é totalmente diferente dos *traders*. Em quando os *traders* operam nas tendências, o *Smart Money* tem em vista vender no topo e comprar no fundo. O comércio de *Smart Money* significa negociar com as estratégias institucionais, que estão alinhadas com os aspectos do *Smart Money*. Este método de negociação é mais poderoso do que o uso de qualquer análise técnica usada por operadores do mercado. (FOREXLENS, 2022).

Os operadores do mercado não podem movimentar o mercado, pois o *Smart Money* controla mais de 60% do volume total do mercado. Negociar com Smart Money exige muita prática e tempo. A negociação de Smart Money é muito confidencial e é uma das estratégias mais ocultas no mundo das moedas. Grandes instituições e bancos nunca revelam suas estratégias (FOREXLENS, 2022). A negociação de *Smart Money* de maneira institucional é focada na estrutura do mercado e nos conceitos de oferta e demanda (*Order Blocks*). Portanto, é muito importante ter um conhecimento profundo sobre os conceitos de estruturas de mercado e negociação de oferta e demanda e a análise do fluxo de ordens com as quais o banco negocia.

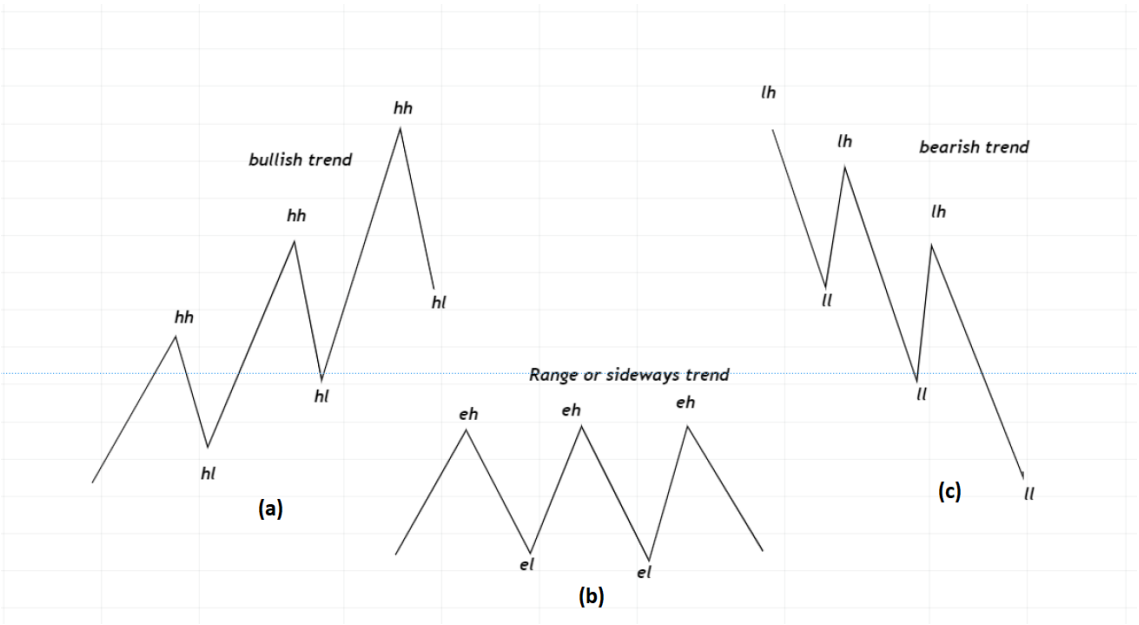
3.1.7 *Inner Circle Trade*

O ICT é uma estratégia que visa seguir as movimentações de mercado causadas pelas grandes instituições. Essa estratégia visa operar com as instituições, podendo assim lucrar junto com elas. Os conceitos principais do *Inner Circle Trade* são Estrutura do Mercado, Imbalance e Order Blocks. Esses conceitos serão descritos a seguir:

A Estrutura de mercado é o elemento mais importante do Forex, que consiste sobre os movimentos básicos do mercado: *Higher high* (hh), *higher low* (hl), *lower high* (lh) e *lower low* (ll) (MASKEY, 2021). A estrutura de mercado é formado por 3 tendências,

Bullish, *Bearish* e *Sideways*. A *Bullish* ocorre quando há tendência de alta, portanto o gráfico conterá *Higher High* e *higher low* apresentado na Figura 17 (a). Enquanto *Bearish* tem tendência de baixa e forma-se por *lower high* e *lower low* apresentados na Figura 17 (c). E por fim tem a *Sideways* que forma um *equal higher* e *equal low* apresentada na Figura 17 (b), descrevendo uma consolidação até que a faixa seja quebrada formando uma tendência de alta ou de baixa. Na Figura 18 é observado no início do gráfico um *Bullish trend* contento *hh* e *hl* e depois uma *Bearish trend* apresentando *lh* e *ll*.

Figura 17 – Tipos de Tendência



Fonte: (MASKEY, 2021)

O momento de quebra de estrutura provoca uma reversão, isto é, se o mercado está em tendência de alta, agora irá ter uma tendência de baixa e vice-versa. Percebe-se momento de quebra quando se tem uma oscilação de *higher high* e *higher low* e de repente ocorre *lower low*, mudando assim, a direção de tendência. Na Figura 19, *x* representa a quebra de estrutura (*break of structure*), depois da quebra de estrutura, iniciando-se uma tendência de baixa.

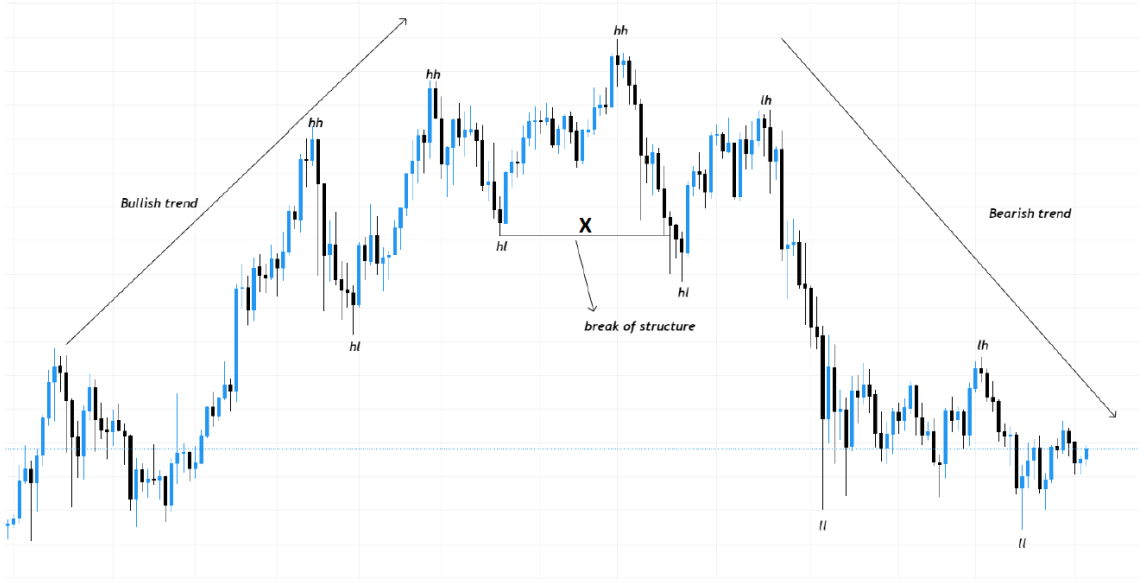
O *Imbalance*, ou "desequilíbrio", traduzido para português, trata-se de uma reação do gráfico de velas quando ocorre uma grande quantidade de compra ou de venda (MASKEY, 2021). Na Figura 20 é apresentada visualmente como o *Imbalance* é descrito em um gráfico de velas, fazendo o preço cair rapidamente, deixando um desequilíbrio devido à grande quantidade de venda do ativo.

Order Blocks no mercado Forex referem-se à coleção de ordens de grandes bancos

Figura 18 – Estrutura de mercado no gráfico de velas

Fonte: (MASKEY, 2021)

Figura 19 – Quebra de Estrutura de Mercado



Fonte: (MASKEY, 2021)

e instituições na negociação (MASKEY, 2021). Os grandes bancos não apenas realizam a compra ou a venda, mas distribuem uma única operação para maximizar o potencial de lucro.

Na análise técnica, quando uma estrutura de mercado variada ou o preço se move na forma de um bloco horizontal, as operações são acumuladas nessa área. Uma onda de impulso se formará depois que os pedidos se acumularem e quebrarem o bloco ou a faixa

Figura 20 – Imbalance

Fonte: (MASKEY, 2021)

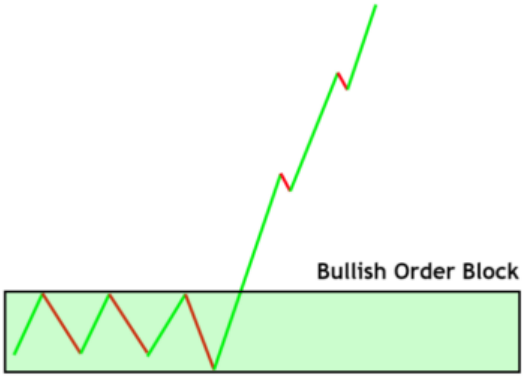
de preço. Essa onda impulsiva mostra o desequilíbrio e a tendência de preços feita por *traders* institucionais e grandes bancos. Porque a maioria das grandes movimentações do mercado são realizadas pelos bancos.

Para identificar *Order Block* no gráfico é necessário traçar duas retas horizontais, uma passando no ponto mais alto e o outro no ponto mais baixo formando uma zona de *Order Block*. São nas zonas em que são realizadas as operações, mas dependendo do que ocorrer depois da consolidação, poderá ser identificado o tipo dele. Existe 2 tipos de *Order Blocks*: *Bullish Order Block* e *Bearish Order Block*.

Bullish Order Block, descrita na Figura 21, ocorre uma quebra de estrutura, indicando uma tendência de alta devido à grande quantidade de compras efetuadas pelas instituições. Depois que ocorrer uma inversão da tendência e o ativo voltar para zona de *Bullish Order Block* recomenda-se a compra, em razão dessa área ser protegida pelas instituições, assim tende de repetir o movimento anterior do mercado.

Na *Bearish Order Block* descrita na Figura 22, ocorre uma quebra de estrutura indicando uma tendência de baixa devido à grande quantidade de vendas realizadas pelas instituições. Depois que o ativo alcançar a zona novamente recomenda-se a venda do ativo, porque o mercado tende a repetir o mesmo movimento do mercado devido às

Figura 21 – *Bullish Order Block*



Fonte: (MUHAMMAD, 2022)

operações de vendas das instituições.

Figura 22 – *Bearish Order Block*

Fonte: (MUHAMMAD, 2022)

3.2 Acurácia

A acurácia é uma medida de desempenho comumente utilizada para avaliar a precisão de um modelo de classificação (BISHOP, 2006). Ela representa a proporção de predições corretas em relação ao total de predições feitas pelo modelo.

De acordo com Bishop (2006), a acurácia é definida como a taxa de acertos do modelo em relação ao número total de amostras avaliadas. Essa medida é amplamente utilizada em diferentes áreas da ciência de dados e aprendizado de máquina, pois é uma forma direta de quantificar a precisão global do modelo.

Matematicamente, a acurácia é calculada dividindo o número de predições corretas pelo número total de amostras avaliadas e multiplicando por 100 para obter o

resultado em porcentagem, conforme é apresentado na Equação 12.

$$\text{Acurácia} = \left(\frac{\text{Predições corretas}}{\text{Total de amostras}} \right) \times 100 \tag{12}$$

Uma acurácia alta indica que o modelo está fazendo um bom trabalho em classificar corretamente as amostras, enquanto uma acurácia baixa indica que o modelo está tendo dificuldades em fazer predições precisas.

3.3 Matriz de Confusão

A matriz de confusão é uma métrica utilizada para avaliar o desempenho de um modelo de classificação (GÉRON, 2019). Ela é especialmente útil quando se trabalha com problemas de classificação binária, embora também possa ser adaptada para problemas de classificação multiclasse.

De acordo com Géron (2019), matriz de confusão é uma tabela que mostra a contagem dos acertos e erros do modelo em relação às classes reais do conjunto de dados. Ela é construída com base nas previsões do modelo e nas classes verdadeiras dos exemplos. A estrutura da matriz de confusão é apresentado na Figura 23.

Figura 23 – Matriz de confusão

		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP)	False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

Fonte: (THARWAT, 2018)

Onde:

- TP (*True Positive*): representa a quantidade de exemplos que foram corretamente classificados como positivos.

- FN (*False Negative*): representa a quantidade de exemplos que foram erroneamente classificados como negativos.
- FP (*False Positive*): representa a quantidade de exemplos que foram erroneamente classificados como positivos.
- TN (*True Negative*): representa a quantidade de exemplos que foram corretamente classificados como negativos.

Esses valores são úteis para calcular diversas métricas de avaliação do modelo, como a acurácia, precisão, *recall*, *F1-score*, entre outras. A partir da matriz de confusão, é possível analisar o desempenho do modelo em relação a cada classe e identificar possíveis problemas, como viés de classificação ou desequilíbrio entre as classes.

3.4 Inteligência Artificial

Segundo Russell e Norvig (2009) a área de IA busca a construção de máquinas e softwares que imitem a inteligência humana, podendo realizar tarefas complexas com base a informações (dados) que são coletados. Existem quatro categorias de IA, Sistemas que pensam como humanos, racionalmente, agem como humanos e agem racionalmente (RUSSELL; NORVIG, 2009).

3.4.1 Aprendizado de Máquina

Aprendizado de Máquina ou em inglês *Machine Learning* (ML) é uma sub-área da IA, com a função de aprender padrões através de dados para melhorar processos, produtos e serviços. Existem três tipos principais de Aprendizado de Máquina: Supervisionado, Não Supervisionado e por Reforço.

- A Aprendizagem Supervisionada ocorre por meio treinamento que recebe um conjunto de dados classificados. Exemplo disso é o reconhecimento de uma bicicleta, pois para isso, o algoritmo receberá dados que classifiquem o que tem aquele veículo. Então a máquina ira saber classificar o objeto. Esse é o modelo menos complexo, porque tenta simular o aprendizado humano.
- Aprendizagem não supervisionada recebe dados, porém não são classificados, a máquina visa identificar padrões. Um exemplo são as sugestões de conexões do

Linkedln.

- O aprendizado reforço é com base a experiência, onde o algoritmo deve lidar com o que errou antes e procurar a resposta correta. Um exemplo é o Youtube, após assistir um vídeo, a plataforma recomenda títulos semelhantes.

3.4.1.1 Support Vector Machines

Support Vector Machines (SVM) é um algoritmo de ML, cuja técnica de aprendizagem supervisiona e usa dados para classificação ou regressão. Segundo Russell e Norvig (2009) as vantagens do SVM são:

1. Os SVMs constroem um limite de decisão com maior distância possível a pontos de exemplo.
2. Os SVMs criam uma separação linear em hiperplano, mas tem a capacidade de incorporar os dados em um espaço de dimensão superior.
3. Os SVMs são um método não paramétrico.

3.4.1.2 Floresta Aleatória

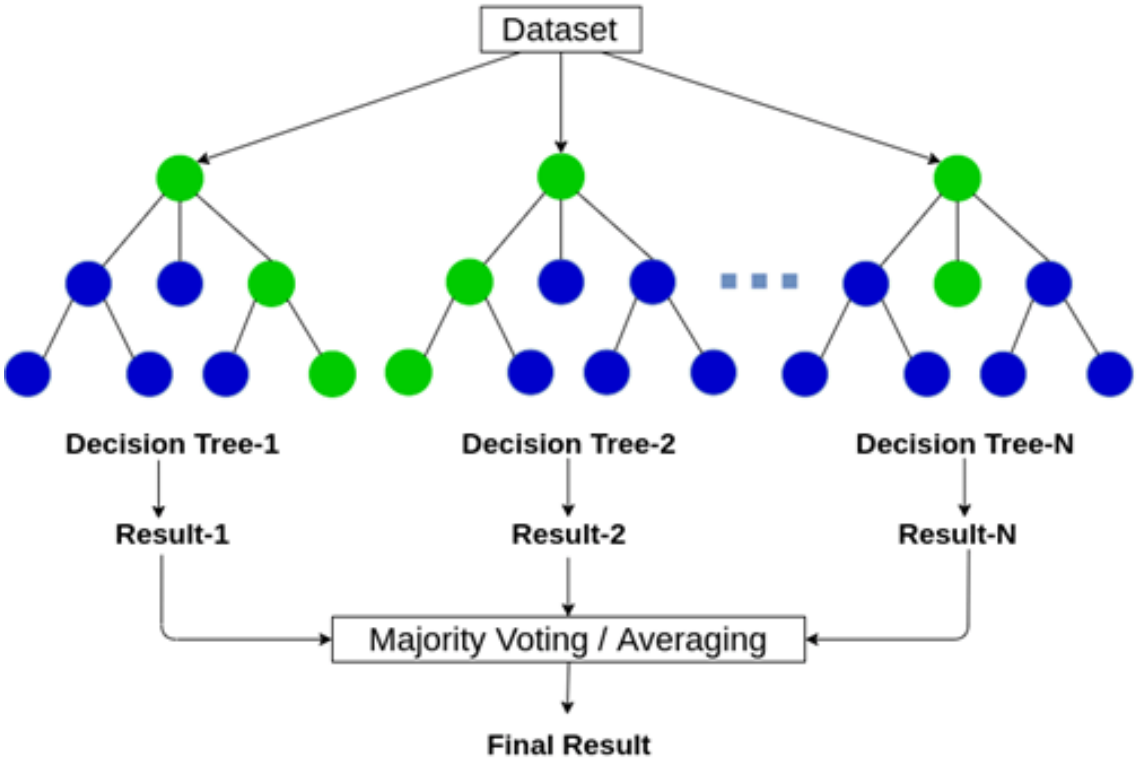
Floresta Aleatória, ou também chamada em inglês de *Random Forest*, é um algoritmo de ML supervisionado, um dos mais usados devido à sua precisão e simplicidade, e bastante utilizado para classificação e regressão. Deu-se o nome de floresta pelo fato de ser formada por várias árvores de decisão, como apresentada na Figura 24.

De acordo com Russell e Norvig (2009), a Árvore de decisão é uma função que recebe atributos e retorna um valor de saída. A saída é uma variável booleana, verdadeiro (positivo) ou falso (negativo). A Árvore de Decisão chega ao resultado com processos de testes em sequência, cada nó da árvore é um teste dos atributos de estrada e as ramificações dos nós são os valores do atributo. Cada nó de folha da árvore informa o valor que foi que será retornado pela função (teste).

Um exemplo de atributos de uma árvore de decisão para determinar se deve-se esperar ou não por uma mesa em um restaurante, são apresentados na Tabela 4.

Cada variável tem um pequeno conjuntos de valores, como o valor de Tempo de espera, não é um número inteiro, são 4 valores, 0-10, 10-30, 30-60 ou >60 minutos. Na Figura 25 é a árvore de decisão não e incluído os atributos Preço e Tipo, se o exemplos são processados a partir da raiz e seguindo as ramificações até alcançar uma folha (resultado).

Figura 24 – Diagrama da Floresta Aleatória



Fonte: (HAO et al., 2021)

Tabela 4 – Atributos para árvore de decisão

Atributo	**Descrição**
Alternativa	Restaurante alternativo por perto.
Bar	Se o restaurante tem um bar onde se pode esperar.
Sex/Sáb	Esperar às sextas e sábados.
Famintos	Se estão com fome.
Clientes	Quantas pessoas estão no restaurante, com valores nenhum, alguns e cheio.
Preço	A faixa de preços do restaurante.
Chovendo	Se tiver chovendo o resultado e negativo para espera.
Reserva	Se tiver feito reserva.
Tipo	O tipo do restaurante.
Tempo de espera	O tempo de espera para ter uma mesa vazia.

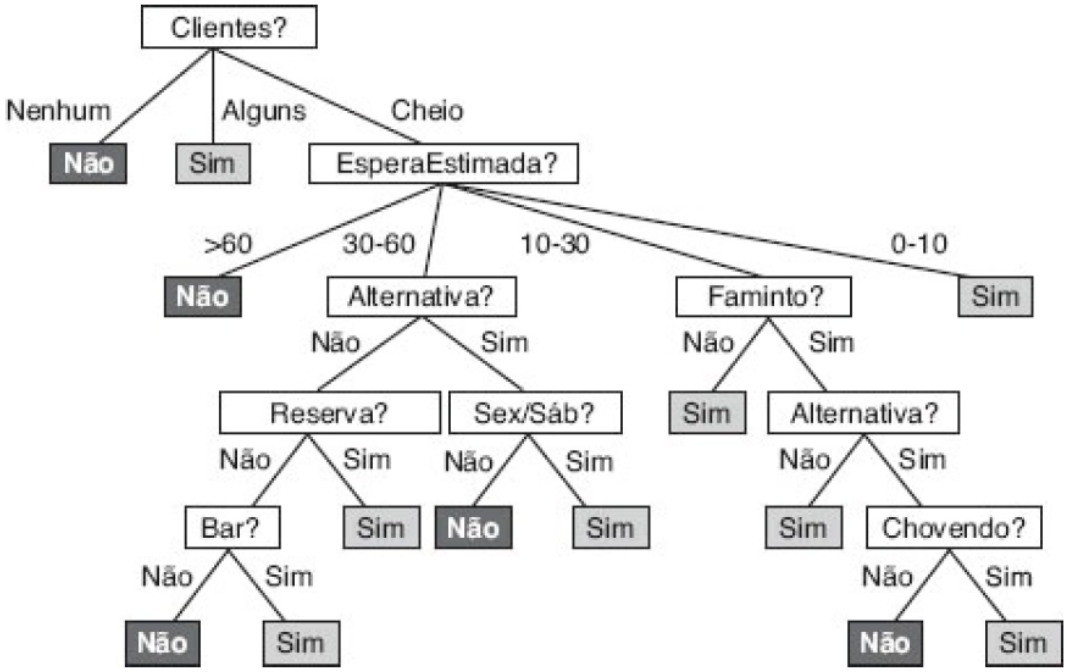
Fonte: Autor (2022)

Se a situação seria Clientes cheio e EsperaEstimada de 0-10, o resultado seria positiva (sim para esperar uma mesa).

As vantagens da floresta aleatória são:

- Sem overfitting significando o treino tem um desempenho excelente enquanto os dados de teste têm resultado ruim.

Figura 25 – Árvore de Decisão para decidir a espera ou não de uma mesa



Fonte: (RUSSELL; NORVIG, 2009)

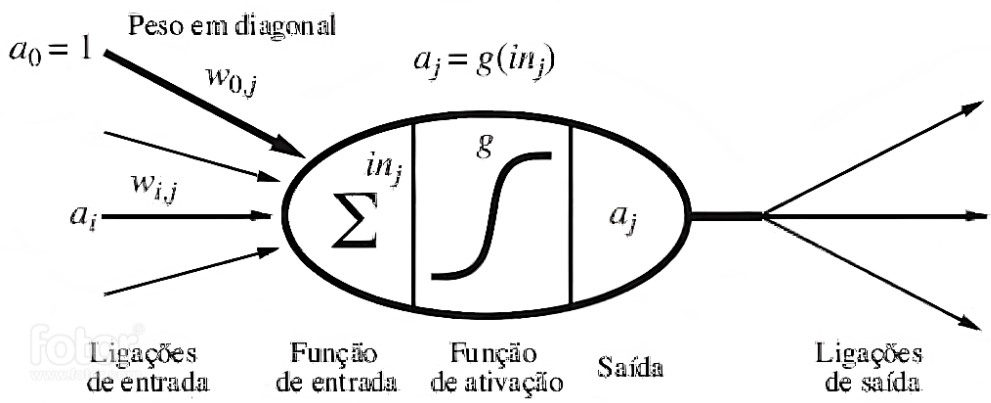
- A árvore de decisão é versátil por causa que pode ser usada para regressão e classificação.
- Altamente precisa por utilizar carias arvores de decisão.
- Reduz o tempo gasto no gerenciamento de dados

3.4.1.3 Redes Neurais Artificiais

De acordo Russell e Norvig (2009) Redes Neurais Artificiais (RNAs) é constituído por neurônios artificiais que visa simular as funções de um neurônio biológico. Por isso os dendritos são associados a entrada que tem ligações com o corpo artificial sendo associados a um determinado peso, recebidos pelo símbolo soma onde são processados e por fim a saída, relacionada com o axônio. As redes neurais, é uma das formas mais populares e eficazes de aprendizagem do sistema de acordo Russell e Norvig (2009). Na Figura 26 é descrita o modelo matemático de um neurônio e a Equação 13 apresenta matematicamente.

$$a_j = g(\sum_{i=0}^{n} w_i, a_{ij}) \tag{13}$$

Figura 26 – Modelo matemático de um neurônio



Fonte: (RUSSELL; NORVIG, 2009)

Com a Figura 26 e a Equação 13 a variável j é a unidade atual, a_i é a ativação de saída da unidade i e w_{ij} é o peso na ligação da unidade i com a unidade atual j. A saída da unidade é dada pela Equação 13.

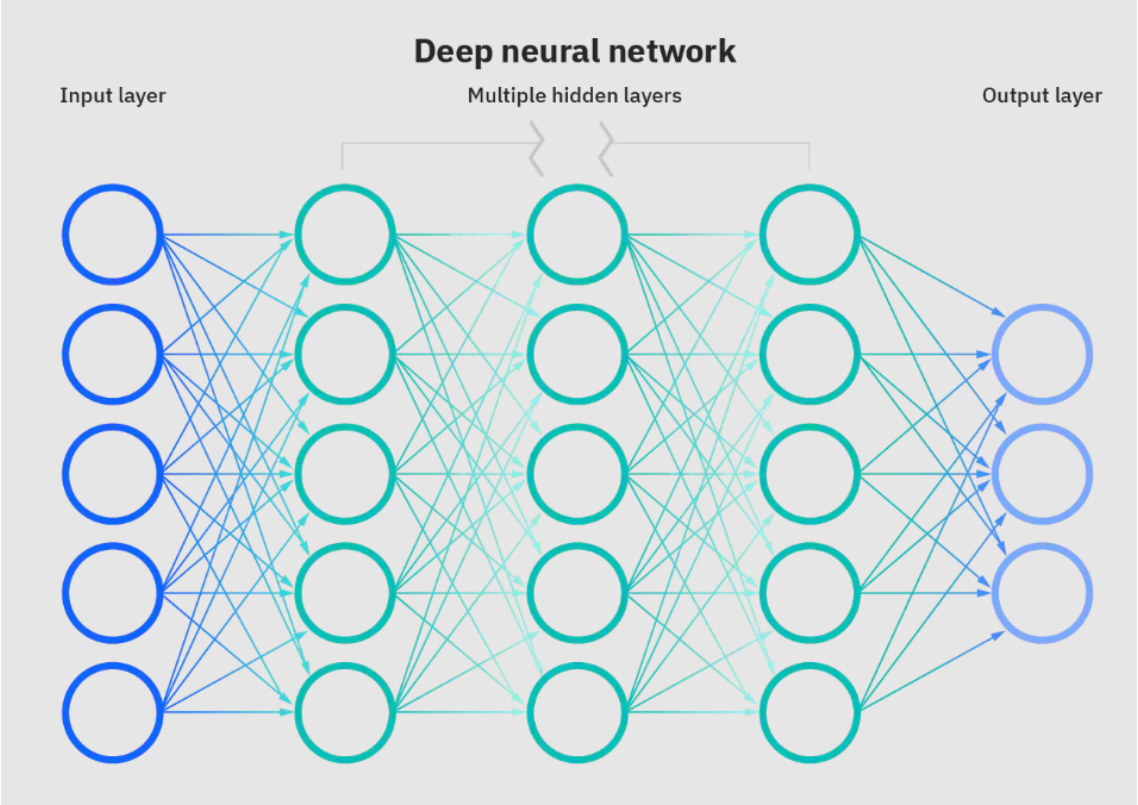
De acordo com Russell e Norvig (2009) cada ligação entre os nós tem o peso de w_{ij}. A entrada fictícia a_0 recebe o valor de 1 e um peso de $w0,j$. Após serem realizadas as somas ponderadas das entradas de cada unidade, é aplicada a função de ativação g para obter a saída.

A Rede Neural Artificial é composta por camadas de um nó (neurônio artificial), a estrutura da rede é uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Cada neurônio artificial se conecta com outro e tem um peso e um limite associados. Se a saída do nó individual estiver acima do valor do limite especificado, esse nó será ativado, enviando para a próxima camada da rede, senão nenhum dado será transmitido para a próxima rede. As redes neurais são treinados com dados para melhorar a precisão ao longo do tempo.

Os tipos de Redes Neurais Artificiais são classificadas com finalidades diferentes são: Perceptron, Redes Neurais Feedforward (Perceptrons Multicamadas), Redes Neurais Convolucionais e Redes Neurais Recorrentes. A **Rede Neural Perceptron** é a mais antiga, somente contém um neurônio artificial. As **Redes Neurais Feedforward** ela e composta por uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Essa rede é composta por neurônios sigmóides e não perceptrons.

As **Redes Neurais Convolucionais**, ou em inglês *Convolutional Neural Network* (CNN), similares a Feedforward, é geralmente usado para reconhecimento de imagens, porém apresentam um maior número de camadas e, consequentemente, de operações.

Figura 27 – Estrutura da Rede Neural Artificial



Fonte: IBM (2022)

Em uma Rede Convolucional cada camada é responsável por extrair determinadas informações dos dados de entrada. A informação flui através de cada camada da rede, com a saída da camada anterior fornecendo a entrada para a camada seguinte da rede.

As **Redes Neurais Recorrentes** ou em inglês *Recurrent neural network* (RNN), diferenciam-se pela presença de ao menos uma conexão de feedback e loops entre os neurônios. Este loop permite que o resultado processado no passo anterior componha e contribua com o resultado seguinte, por isso são consideradas redes com memória. Então o RNR utilizam os dados de entrada atual, mas também o que foram recebidos anteriormente.

A *Long Short Term Memory* (LSTM) é uma arquitetura que pertence a RNN que usa valores em intervalos arbitrários. A LSTM é bem adequada para classificar, processar e prever séries temporais com intervalos de tempo de duração desconhecida.

Esses algoritmos de aprendizado são potencializados principalmente ao usar dados de séries temporais para fazer previsões sobre resultados futuros, como previsões de mercado de ações ou previsão de vendas.

As vantagens das Redes Neurais artificiais são:

- Tem a capacidade de fornecer os dados a derem processados em paralelo, então podem lidar com mais de uma tarefa ao mesmo tempo.
- As rede neurais então em resistência significando que a perda de uma ou mais célula influencia o desempenho das redes neurais artificiais.
- São utilizadas para armazenar informações na rede de forma que, mesmo na falta de um par de dados, isso não signifique que a rede não esteja gerando resultados.
- Somos capazes de treinar RNAs para que essas redes aprendam com eventos passados e tomem decisões.

3.5 Trabalhos correlatos

Nesta seção foi feita as descrições dos trabalhos encontrados que usam o IA no mercado financeiro, no total são 9 trabalhos, serão colocados os objetivos, mercado de aplicação e as técnicas de IA utilizadas nos trabalhos correlatos.

3.5.1 Avaliação de um *Algotrading* baseado em *Deep Learning* para o Mercado de Capitais utilizando Gerenciamento de Risco

O trabalho de Silva et al. (2021) tem como objetivo avaliar um sistema de negociação algorítmica (*algotrading*) que utiliza técnicas de *Deep Learning* no mercado de capitais, com foco na incorporação de estratégias de gerenciamento de risco.

O estudo começa descrevendo a implementação do algoritmo de negociação, que utiliza modelos de Deep Learning para analisar dados do mercado financeiro, como preços de ativos, volumes de negociação e outros indicadores relevantes. Esses modelos são projetados para identificar padrões complexos nos dados e gerar sinais de compra ou venda com base em análises preditivas.

O aspecto inovador do trabalho é a integração de estratégias de gerenciamento de risco no algoritmo. Isso significa que o sistema não se concentra apenas em obter lucros, mas também em controlar as perdas potenciais e proteger o capital investido. As estratégias de gerenciamento de risco ajudam a reduzir a exposição a riscos desnecessários e a evitar grandes perdas financeiras.

Os resultados são discutidos em detalhes, destacando a eficácia do algoritmo de

Deep Learning em gerar sinais de negociação precisos e a importância das estratégias de gerenciamento de risco na preservação do capital investido.

3.5.2 *Machine Learning* e Análise Técnica como Ferramentas para Construção de Portfólios de Renda Variável no Mercado Brasileiro

O artigo tem como objetivo explorar a aplicação de técnicas de *Machine Learning* e análise técnica na construção de portfólios de investimento em renda variável no mercado brasileiro.

O objetivo do estudo é investigar como essas abordagens podem ser combinadas para melhorar a seleção de ativos e a alocação de recursos em portfólios de ações. A análise técnica é usada para identificar padrões históricos nos preços e volumes de negociação dos ativos, enquanto o *Machine Learning* é aplicado para gerar previsões e tomar decisões baseadas em modelos estatísticos.

O estudo utiliza dados históricos do mercado brasileiro, como preços de ações, volumes de negociação e indicadores técnicos relevantes. Algoritmos de *Machine Learning*, como redes neurais e árvores de decisão, são treinados com esses dados para prever o desempenho futuro dos ativos e identificar oportunidades de investimento.

Os resultados do estudo são analisados com base em métricas de desempenho financeiro, como retorno do investimento, volatilidade e risco ajustado. Além disso, o artigo discute a importância da gestão de riscos e diversificação na construção de portfólios de renda variável.

3.5.3 um agrupamento de modelos conexonistas por meio de sinapses artificiais e suas aplicações no mercado de criptomoedas

No trabalho Júnior (2020) é construído um algoritmo chamado de CMEAS, que é feito para agrupar duas redes neurais convolucionais. As propostas para o treinamento do algoritmo foram com aprendizado supervisionado e com aprendizado de reforço. Com os resultados comprovados pelos testes de Wilcoxon, apresentaram que o CMEAS obteve fator de lucro e índice *sharpe* superior , também foi superior em todas as métricas da estratégia compra e retem (*buy and hold*).

3.5.4 Desenvolvimento de um método híbrido para negociações de ações na bolsa de valores brasileira

No trabalho Eberman (2018) visa construir um algoritmo que selecione uma ação e invista nele, porque a maioria dos trabalhos sobre mercado financeiro, é selecionado uma ação e depois aplicado um método estratégico. Neste trabalho, para diferenciar aos outros trabalhos, é utilizado um método de tomada de decisão multicritério, em cada dia são ranqueadas por esse método utilizando critérios de análise técnica e selecionada para a compra. Portanto, para aumentar a confiabilidade do método de tomada de decisão multicritério, é utilizado um método híbrido composto por decomposição de modo empírico e máquina de aprendizado extremo. Quando o TOPSIS selecionava uma ação e era confirmada pelo modelo híbrido, conseguiu aumentar o percentual de negociações com lucro.

3.5.5 Avaliando a classificação de aprendizado de máquina para negociação financeira: Uma abordagem empírica

O artigo "Avaliação de classificação de aprendizado de máquina para negociação financeira: Uma abordagem empírica"em português, de Gerlein et al. (2016), apresenta uma avaliação do uso de algoritmos de aprendizado de máquina para realizar classificação em operações financeiras.

O objetivo do estudo é investigar a eficácia de diferentes técnicas de aprendizado de máquina na previsão de tendências e tomada de decisão em negociações financeiras. O autor conduz experimentos empíricos usando diferentes algoritmos de classificação, como redes neurais, máquinas de vetor de suporte (SVM) e árvores de decisão, aplicados a conjuntos de dados financeiros.

Os resultados mostram que as técnicas de aprendizado de máquina podem fornecer previsões úteis para negociação financeira. Alguns algoritmos apresentam melhor desempenho do que outros, dependendo das características dos dados e do tipo de instrumento financeiro analisado. Além disso, o estudo destaca a importância da seleção adequada de recursos e parâmetros para melhorar a precisão das previsões.

3.5.6 Uma abordagem quantitativa para criar uma plataforma híbrida de filtragem de ações

Conforme o trabalho de Nawani et al. (2020), tem como objetivo o desenvolvimento de uma plataforma de investimento usando *Machine Learning*, para investidores iniciantes e experientes. Os usuários podem selecionar a partir de uma lista de estratégias hibridas que ajudam na hora da pre-seleção de ações, resumindo seria um sistema para filtragem de ações para investidores. No trabalho são filtrados trabalhos ações do *S&P* 500 depois de aplicar estrategias fundamenta e técnicas. Foi usado LSTM, que é uma técnica de aprendizado de maquina, para prever a tendencias das ações que foram filtradas e foi aplicada analise de sentimento para verificar a opinião publica sobre os ativos que foram filtradas.

3.5.7 Análise e modelagem das mudanças de preço no mercado de câmbio com base em dados estruturais de mercado

O trabalho de Bebeshko, Khorolska e Desiatko (2021) é construído um sistema teórico que tente determinar o preço futuro de ativos, para obter lucro para os *traders*. São usados informações sobre essas empresas, e existem três métodos principais para prever preços de ações: análise fundamental, análise técnica e métodos tecnológicos. A previsão utilizada neste trabalho foi o uso de redes neurais e regressão logística. Quando usado a regressão logística, era demonstrado os melhores indicadores de qualidade

3.5.8 Sistema automatizado de negociação para previsão do mercado de câmbio usando indicadores de análise técnica e redes neurais artificiais

No trabalho de Ismail et al. (2022) discutido um sistema de negociação automatizada para previsão de mercado Forex utilizando Indicadores de Análise Técnica e Redes Neurais. É utilizado o câmbio Libra-Dólar(GBP/USD) como experimento, utilizando a combinação de Indicadores de Análise Técnica e Redes Neurais. Os Indicadores selecionado são Média Móvel, *Relative Strength Index* (RSI), *Candle Pattern Classification Method* e *Output Normalization*. O ambiente experimental foi construído com linguagem C# e bibliotecas. Neste trabalho comentou que os indicadores ideais são

a Média Móvel e o RSI.

3.5.9 Determinação dos momentos de compra e venda de *bitcoins* usando técnicas de inteligência artificial

Sobre o trabalho Filho (2022), tem como objetivo determinar a compra e a venda de Bitcoin usando técnicas de inteligencia artificial para auxiliar investidores no mercado de criptomoedas. Para a análise técnica foram utilizados os indicadores mias populares e mais consolidados que são usado no mercado. As técnicas de Inteligencia Artificial, foram utilizadas árvore de decisão, floresta aleatória e rede neural artificial. Para o desenvolvimento do modelo foi utilizado a linguagem Python com as bibliotecas que são oferecidos pela linguagem para a execução de *Machine Learning* e analise de dados. Em conclusão o modelo obteve resultados positivos de lucros em longo prazo, utilizando as 3 técnicas de IA que foram propostas. E como indicadores foi usado RSI, MACD, Histograma MACD, bandas de Bollinger e o Estocástico.

3.5.10 Resultado dos trabalhos correlatos

Para melhor descrição dos trabalhos encontrados, foi realizada a construção da Tabela 5 com os objetivos do trabalho e as técnicas de IA que foram utilizadas para o objetivo do trabalho.

Podemos perceber a quantidade de trabalhos que utilizam Redes Neurais Artificiais, tornando assim essa técnica importante a ser aplicada neste trabalho. Porém, podemos perceber que a maioria de trabalhos utiliza a previsão do mercado financeiro para a negociação. O trabalho de Filho (2022) é o que se aproxima mais a proposta do trabalho, sendo o principal artigo a ser seguido para a construção do modelo.

Tabela 5 – Descrição dos trabalhos correlatos

Autor	**Objetivo**	**Técnicas de IA**	**Resultados**
Silva et al. (2021)	Desenvolver um sistema de negociação autônoma com previsão	LSTM	Maior Lucro anual foi de 208.23%
Coelho (2020)	Aplicar LSTM no mercado de ações utilizando indicadores de análise técnica para prever o retorno da negociação	LSTM	Entre 01/2019 até 09/2020 teve retorno de 120% de lucro
Júnior (2020)	Aplicação de Redes Neurais para previsão de séries temporais no mercado de criptomoedas	Redes Neurais	8.1% de lucro por mês
Eberman (2018)	Construção de um algoritmo que selecione uma ação e invista nele de forma automática	Rede Neural Artificial	Período de 50 dias teve com o melhor retorno teve de 326.77%
Gerlein et al. (2016)	Encontrar o melhor momento para negociação no mercado Forex utilizando previsões	Rede Neural Artificial; SVM; Árvore de Decisão	Período de 6 anos obtém 142.89% de lucro.
Nawani et al. (2020)	Construção de uma plataforma de investimentos utilizando previsão	LSTM	Trabalho analisando os resultados de previsão.
Bebeshko, Khorolska e Desiatko (2021)	Construção de um sistema teórico para previsão do valor dos ativos	Regressão logística; Redes Neurais Artificiais	O modelo de Regressão Logística teve melhor desempenho de previsão, com precisão de 90%.
Ismail et al. (2022)	A previsão do mercado financeiro para negociação utilizando indicadores	Redes Neurais Artificiais.	Período de 5 anos têm 1754.3% de lucro acumulado.
Filho (2022)	Determinar a compra e a venda utilizando indicadores.	Árvore de Decisão; Floresta Aleatória; Redes Neurais Artificiais.	Entre 2015 até 2022 obteve o lucro de 1,194.69% com os modelos combinados.

Fonte: Autor (2022)

4 PLANEJAMENTO E DESENVOLVIMENTO DO TRADECLASSIFIER

Neste capítulo é apresentado o planejamento para o desenvolvimento do TradeClassifier, descrevendo as abordagens utilizadas e os métodos de execução escolhidos. Para a implementação do modelo, optamos pela linguagem Python, sendo mais adequada devido às bibliotecas disponíveis para o mercado financeiro e inteligência artificial.

Foram encontrados diversos trabalhos correlatos que abordam a previsão da movimentação do mercado, mas poucos se focam na classificação dos melhores momentos de compra e venda. Além disso, apenas alguns estudos utilizam técnicas específicas do mercado financeiro para analisar esses momentos de negociação, sendo que a maioria se baseia apenas em dados sobre o movimento do mercado e técnicas de inteligência artificial para aprender padrões em gráficos.

O desenvolvimento do TradeClassifier foi inspirado no trabalho de Filho (2022), que possui o mesmo objetivo de identificar os momentos mais favoráveis para operações no mercado financeiro, mas se diferencia na estratégia do mercado financeiro. Outras diferenças deste projeto é a adoção de dois tipos de negociação: *Long* e *Short*. Esses termos são comumente utilizados no mercado financeiro para descrever ações de compra e venda de ativos. A negociação *Long* refere-se à compra de um ativo com a expectativa de que seu valor aumente, proporcionando um lucro quando o ativo é vendido posteriormente. Já a negociação *Short* envolve a venda de um ativo emprestado, com a intenção de recomprá-lo posteriormente a um preço mais baixo, lucrando assim com a queda do valor do ativo (BODIE; KANE; MARCUS, 2014).

Além disso, a estratégia adotada neste projeto envolve a utilização de *Take Profit* e *Stop Loss* nas operações. O *Take Profit* é um nível predefinido em que o *trader* decide encerrar a operação e realizar o lucro antes que o preço do ativo se reverta. Por outro lado, o *Stop Loss* é um nível predefinido em que o *trader* decide encerrar a operação e limitar as perdas caso o preço do ativo se mova contra sua posição. Essa estratégia foi inspirado do trabalho de Silva et al. (2021), para controlar as perdas potenciais e proteger o capital investido.

As técnicas de inteligência artificial que foram implementadas são, SVM (*Support Vector Machine*), Floresta Aleatória, Árvore de decisão e Redes Neurais.

4.1 Ferramentas

As ferramentas adotadas para o desenvolvimento do projeto são: Python[1] e suas bibliotecas e a plataforma Visual Studio Code[2]. A escolha da linguagem Python foi feita devido à sua simplicidade, as bibliotecas de Inteligência Artificial, manipulação de dados e por ser de código aberto. A linguagem também é muito utilizadas em ciência de dados e aprendizado de máquina, que são áreas fundamentais para a inteligência artificial AWS (2023).

A linguagem Python permite a manipulação de dados em larga escala, a criação de modelos preditivos e de aprendizado de máquina, além da visualização de dados, que são cruciais para a tomada de decisões de investimento.

As bibliotecas que serão usadas para o desenvolvimento do modelo são *Pandas*[3], *Numpy*[4], *Matplotlib*[5], *Scikit-learn*[6]. *Pandas* é uma biblioteca para manipulação de dados, usada principalmente na área de análise de dados. A biblioteca *Numpy* serve para usar várias funções e operações estatísticas, bastante usadas no mercado financeiro. *Matplotlib* é para a visualização de gráficos e figuras. A *Scikit-learn* é uma biblioteca que contém técnicas de IA, oferecendo modelos como *SVM* e *Random Forest*, facilitando o desenvolvimento do trabalho.

O modelo será aplicado no mercado Forex, destinado para câmbio de moedas, e o ativo escolhido é o Euro para Dólar (EUR/USD), as moedas mais negociáveis. Os dados, são do formato CSV (*Comma-separated values*) que foram obtidos diretamente do *MetaTrader 4*[7], a principal plataforma para negociação utilizada pelos *traders* no mercado Forex. O DataFrame contém os dados dos preços em períodos de cada 4 horas do ativo. Na Figura 28 é apresentado os dados do ativo EUR/USD, com parâmetros do gráfico de velas (Data, Time, Abertura, Alta, Baixa, Fechamento e Volume), facilitando a descrição dos dados.

[1] https://python.org.br/
[2] https://code.visualstudio.com/
[3] https://pandas.pydata.org/
[4] https://numpy.org/
[5] https://matplotlib.org/
[6] https://scikit-learn.org/stable/
[7] https://www.metatrader4.com/pt

Figura 28 – Dados do Ativo EUR/USD

	Data	Time	Abertura	Alta	Baixa	Fechamento	Volume
2	1999.01.04	08:00	1.18010	1.18190	1.17690	1.17910	878
3	1999.01.04	12:00	1.17910	1.18250	1.17750	1.18080	1609
4	1999.01.04	16:00	1.18050	1.18620	1.17820	1.18220	1329
5	1999.01.04	20:00	1.18210	1.18420	1.18110	1.18150	312
6	1999.01.05	00:00	1.18150	1.18350	1.18050	1.18080	344
7	1999.01.05	04:00	1.18110	1.18250	1.17950	1.18160	344
8	1999.01.05	08:00	1.18210	1.18330	1.18080	1.18200	1319
9	1999.01.05	12:00	1.18180	1.18230	1.17710	1.17830	1520
10	1999.01.05	16:00	1.17810	1.17930	1.17500	1.17670	1373
11	1999.01.05	20:00	1.17690	1.17850	1.17560	1.17580	457
12	1999.01.06	00:00	1.17610	1.17610	1.17180	1.17260	332
13	1999.01.06	04:00	1.17280	1.17470	1.17250	1.17420	103
14	1999.01.06	08:00	1.17440	1.17730	1.17330	1.17530	1555
15	1999.01.06	12:00	1.17540	1.17640	1.17170	1.17190	1516
16	1999.01.06	16:00	1.17190	1.17290	1.15530	1.16040	1417
17	1999.01.06	20:00	1.16000	1.16360	1.15890	1.16230	365
18	1999.01.07	00:00	1.16230	1.16410	1.16120	1.16340	431
19	1999.01.07	04:00	1.16350	1.16480	1.16350	1.16410	48
20	1999.01.07	08:00	1.16570	1.16790	1.16510	1.16570	955

Fonte: Autor (2022)

4.2 Idealização do modelo

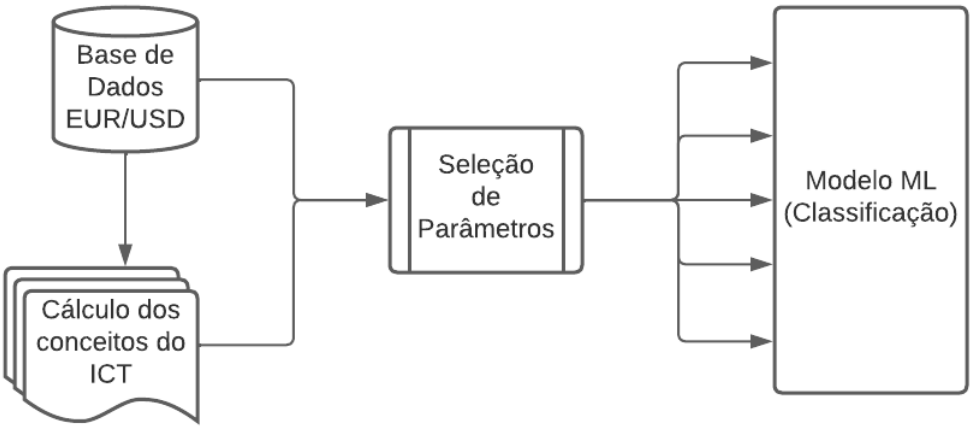
As técnicas de IA que foram utilizadas na forma de classificação do ativo EUR/USD com dados desde 2000 até 2020. Para o processo de desenvolvimento do modelo, foi feito o levantamento de estruturas fundamentais. A Figura 29 é descrita um diagrama do modelo generalizado do processo dos dados.

A execução é dividida em duas etapas, o treinamento do modelo e o teste do modelo. Para isso se deve separar a base de dados para treinamento e outro para teste, assim podendo calcular a acurácia do modelo, utilizando o 'train test split', com 25% para teste.

O diagrama da Figura 29 apresenta os componentes necessários para o treinamento do Modelo. Para a entrada é usada a Base de Dados do ativo EUR/USD, contendo dados históricos do ativo. A ligação da Base de Dados com cálculo dos conceitos do ICT é realizado os cálculos do dos conceitos do ICT conforme os parâmetros de escrita, ou seja, a partir dos dados históricos serão realizados diversos cálculos. Depois dos cálculos são localizadas as regiões de interesse, ou seja, o ICT indica em quais zonas que poderá ser realizada as operações.

As informações das regiões de interesse do ICT, juntamente com o histórico de

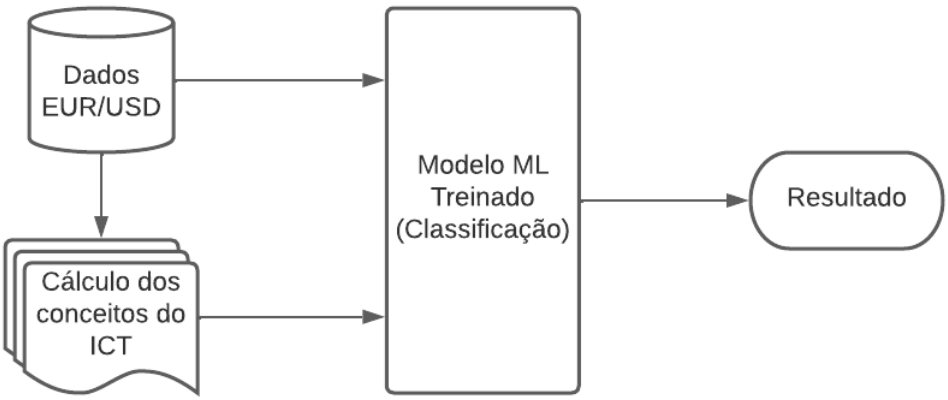
Figura 29 – Diagrama do Treinamento do Modelo



Fonte: Autor (2022)

dados do ativo, são enviadas para o módulo de seleção de parâmetros, que irá selecionar quais as informações mais relevantes a serem utilizadas pelo modelo. A seleção de parâmetros irá descobrir quais são as regiões mais importantes, e combinar os valores com os dados históricos para a realizar uma classificação com os dados de entrada. O módulo de classificação será treinado a partir com os parâmetros selecionados.

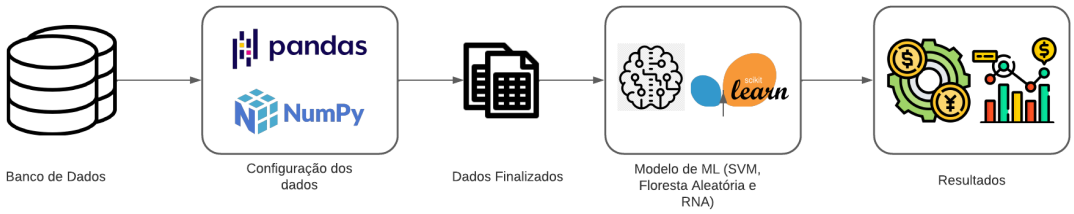
Figura 30 – Diagrama do Teste do modelo



Fonte: Autor (2022)

Na Figura 31 é descrito o modelo final que será desenvolvido, onde será processado os dados de entrada de uma forma que o modelo de ML construído com a biblioteca *Scikit-learn* gere um resultado visualmente de onde comprar e vender, igual como realizado no trabalho de Filho (2022).

Figura 31 – Arquitetura do modelo



Fonte: Autor (2022)

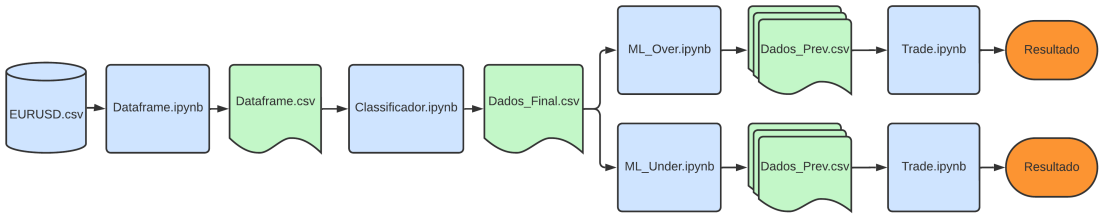
O resultado é a classificação dada com base no treinamento realizado. A resposta é dada a cada vela processada, com classificação de operações no Long e Short.

4.3 Desenvolvimento do TradeClassifier

Nesta seção será descrito todo o processo de desenvolvimento do modelo computacional proposto. Na Subseção 4.3.1 é descrita como o *DataFrame* é organizado para ser processado pelas técnicas de IA, também é onde foi adicionados os valores do ICT conforme os dados iniciais (*Open*, *High*, *Low*, *Close* e *Volume*).

O Diagrama do Algoritmo é apresentado na Figura 32 para melhor visualização da organização do código.

Figura 32 – Diagrama do Algoritmo



Fonte: Autor (2022)

O arquivo "dataframe.ipynb"é responsável por importar os dados históricos do par de moedas EURUSD e realizar as transformações necessárias para criar o *dataframe* contendo os dados históricos do ativo e os valores dos conceitos do ICT. Esse arquivo exporta o *dataframe* resultante, que será utilizado nos próximos passos do processo.

O arquivo "Classificador.ipynb"importa o *dataframe* criado pelo arquivo anterior e aplica o classificador para categorizar os dados em *Long* e *Short*, com base em determinados critérios ou regras estabelecidas. Os dados classificados são exportados

como resultado desse arquivo.

O arquivo "Machine_Learning.ipynb"importa os dados classificados pelo arquivo anterior e realiza o treinamento e teste dos modelos de aprendizado de máquina. Ele exporta o *dataframe* com os dados já processados e testados pelo modelo, incluindo as classificações realizadas pelos modelos.

Por fim, o arquivo "Trade.ipynb"utiliza o *dataframe* resultante do arquivo anterior para realizar as operações de trade com base nas classificações dos modelos. Ele calcula o lucro final obtido com as operações e também gera gráficos que mostram o comportamento do lucro ao longo do tempo para cada modelo.

Essa organização dos arquivos permite uma divisão clara das tarefas e facilita a compreensão do fluxo de execução do código como um todo. Além disso, cada arquivo pode ser executado de forma independente, o que proporciona flexibilidade e agilidade no desenvolvimento e nos testes.

O algoritmo desenvolvido importa os dados históricos do mercado e testa os dados com base dos conceitos do ICT. Depois são processados os dados nas técnicas de ML e assim são classificadas as operações com base aos valores dos conceitos de modo que o modelo aprenda a obter lucro a partir dos dados dos testes.

Para utilizar a biblioteca *scikit-learn* para classificação financeira com os modelos selecionados em Python, foram seguidos os seguintes passos:

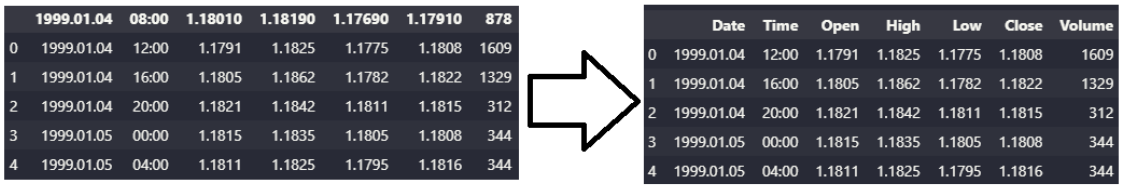
1. Importar e limpar os dados financeiros para treinar e testar os modelo. Os dados têm informações históricos de preços do ativo EURUSD.
2. Cálculo dos conceitos do ICT.
3. Dividir os dados em conjuntos de treinamento e teste. Para avaliar o desempenho dos modelos em dados que ele não viu antes.
4. Instancie o classificador dos modelos do *scikit-learn*, podendo escolher os parâmetros de cada modelo e ajustar os parâmetros para obter mais acertos na classificação.
5. Treine os modelos usando o conjunto de treinamento de dados.
6. Classificar usando os modelos treinados no conjunto de teste de dados.
7. Avaliar o desempenho dos modelos usando métricas de classificação, como acurácia, matriz de confusão, precisão, revocação e pontuação F1.

4.3.1 Configuração da base de dados

A base de dados utilizada neste projeto consiste em informações do mercado financeiro, mais especificamente do par de moedas EUR/USD (Euro para o Dólar americano). Esses dados são armazenados em um arquivo CSV (*Comma-Separated Values*), que é um formato comumente utilizado para armazenar dados tabulares.

Na Figura 33 é apresentado o processo de carregamento dos dados e a importação deles utilizando a biblioteca *Pandas*, que é uma biblioteca amplamente utilizada para análise de dados em Python. No lado esquerdo da figura, é observado os dados puros do arquivo CSV, que são carregados em um objeto *Dataframe* do *Pandas*. No lado direito da figura, é mostrado o *DataFrame* resultante, onde foram adicionados nomes às colunas para facilitar a identificação dos valores.

Figura 33 – Dados carregados



	1999.01.04	08:00	1.18010	1.18190	1.17690	1.17910	878
0	1999.01.04	12:00	1.1791	1.1825	1.1775	1.1808	1609
1	1999.01.04	16:00	1.1805	1.1862	1.1782	1.1822	1329
2	1999.01.04	20:00	1.1821	1.1842	1.1811	1.1815	312
3	1999.01.05	00:00	1.1815	1.1835	1.1805	1.1808	344
4	1999.01.05	04:00	1.1811	1.1825	1.1795	1.1816	344

	Date	Time	Open	High	Low	Close	Volume
0	1999.01.04	12:00	1.1791	1.1825	1.1775	1.1808	1609
1	1999.01.04	16:00	1.1805	1.1862	1.1782	1.1822	1329
2	1999.01.04	20:00	1.1821	1.1842	1.1811	1.1815	312
3	1999.01.05	00:00	1.1815	1.1835	1.1805	1.1808	344
4	1999.01.05	04:00	1.1811	1.1825	1.1795	1.1816	344

Fonte: Autor (2023)

Após a nomeação das colunas no *Dataframe*, os parâmetros "Data"e "Time"são combinados para formar uma única coluna representando a data e hora das observações. Em seguida, utilizando a biblioteca *Pandas*, essa coluna é convertida para o formato de objeto de data, permitindo realizar operações relacionadas a datas e horários. Após a conversão, a nova coluna "Datetime"é definida como o índice do *Dataframe*, para facilitar o acesso aos dados com base em datas específicas, permitindo a realização de análises temporais mais precisas.

Para ajustar o período de dados a ser utilizado no trabalho, foram criadas duas variáveis que indicam a data de início e a data de término desejadas. Essas variáveis permitem selecionar um intervalo específico dentro do conjunto de dados históricos do mercado financeiro. Após definir as datas de 2000 até 2020, foi realizada a renderização dos valores do ativo EUR/USD. A visualização desses valores é apresentada na figura Figura 34. Essa representação gráfica permite observar a variação dos preços ao longo do tempo.

Inicialmente foi realizada a classificação dos *candles* (velas). Para classificar

Figura 34 – Gráfico do EUR/USD de 2000 até 2020

Fonte: Autor (2022)

os *candles* (velas) em *Bullish* (Alta) e *Bearish* (Baixa), foi realizada a comparação dos parâmetros *Open* e *Close*. Se o *Close* for maior que o *Open*, então ele é *True* caso contrário, ele é *false*.

Após exportar e carregar os dados no algoritmo, iniciou-se o cálculo dos conceitos do ICT. Esses conceitos são adicionados ao *Dataframe*, permitindo uma análise mais detalhada e a utilização dessas informações nos modelos de *machine learning*. Esses conceitos são indicadores técnicos, padrões gráficos e métricas que auxiliam na identificação de oportunidades de negociação no mercado financeiro.

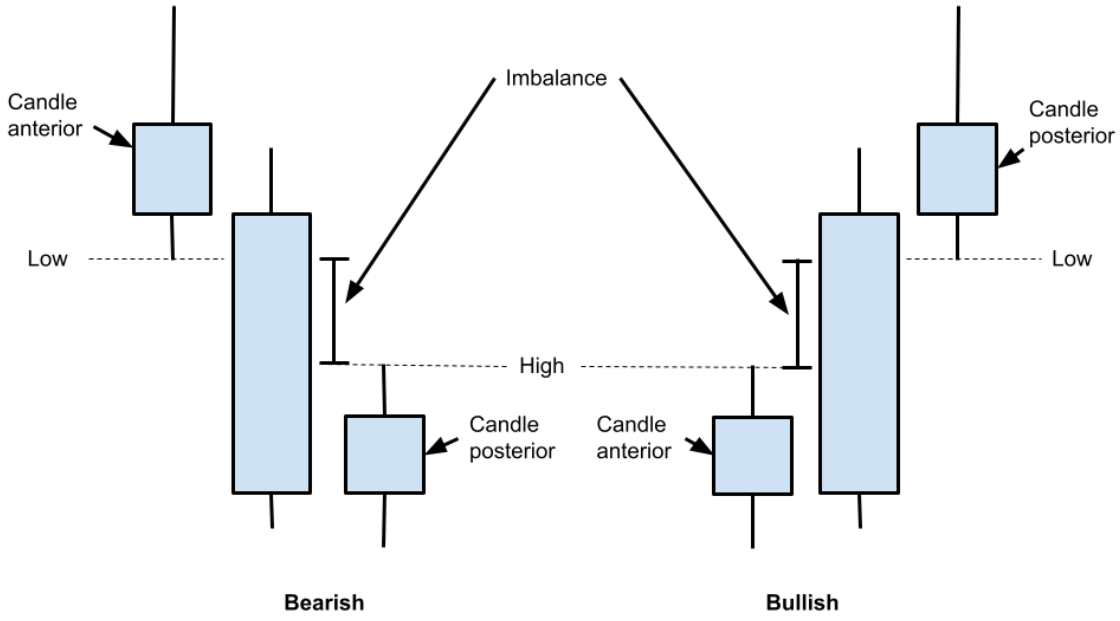
4.3.1.1 Imbalance

Na etapa de cálculo do Imbalance, primeiro é necessário identificar se o *candle* é *Bullish* (alta) ou *Bearish* (baixa), para determinar se existe um desequilíbrio no *candle* atual. Se o *candle* for *Bearish*, um desequilíbrio é identificado se o valor mínimo (*Low*) do candle anterior não alcançar o valor máximo (*High*) do *candle* posterior. Se o *candle* for *Bullish*, um desequilíbrio é identificado se o valor máximo (*High*) não alcançar o valor mínimo (*Low*) do *candle* posterior.

Após identificar o desequilíbrio, são atribuídos valores para os parâmetros "ImbLow"e "ImbHigh". No caso de um *candle Bearish*, o valor ImbLow será o valor máximo (*High*) do *candle* anterior, e o valor "ImbHigh"será o valor mínimo (*Low*) do *candle* posterior. No caso de um *candle Bullish*, o valor "ImbHigh"será o valor mínimo (*Low*) do *candle* anterior, e o valor "ImbLow"será o valor máximo (*High*) do *candle* posterior.

A Figura 35 apresenta um exemplo de como o desequilíbrio é identificado e de onde são extraídos os valores para o cálculo do Imbalance, conforme descrito anteriormente.

Figura 35 – Representação do cálculo do Imbalance



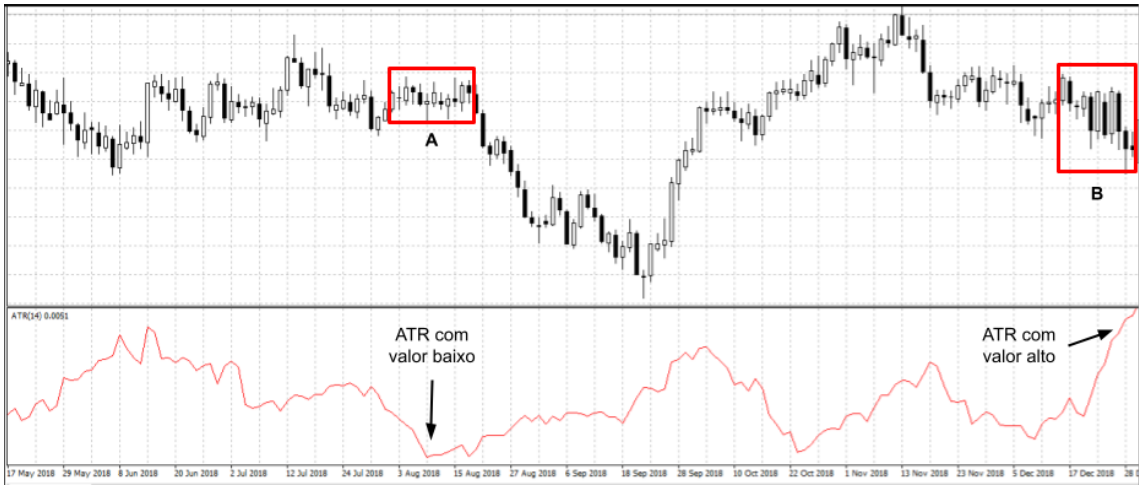
Fonte: Autor (2023).

4.3.1.2 Average True Range

O cálculo do *Average True Range* (ATR) pode ser realizado utilizando a biblioteca *pandas_ta*, que fornece diversas análises técnicas financeiras, incluindo o ATR. Para calcular o ATR, basta passar os dados dos *candles* para a função *pandas_ta.atr()* e informar o período desejado.

Na Figura 36, os *candles* destacados pelo retângulo **A** apresentam corpos pequenos. Na parte inferior do gráfico, é possível observar que o valor do ATR é baixo. Por outro lado, o retângulo **B** destaca *candles* com corpos maiores, resultando em um valor mais alto para o ATR.

Figura 36 – Identificação dos valores do ATR



Fonte: Autor (2023)

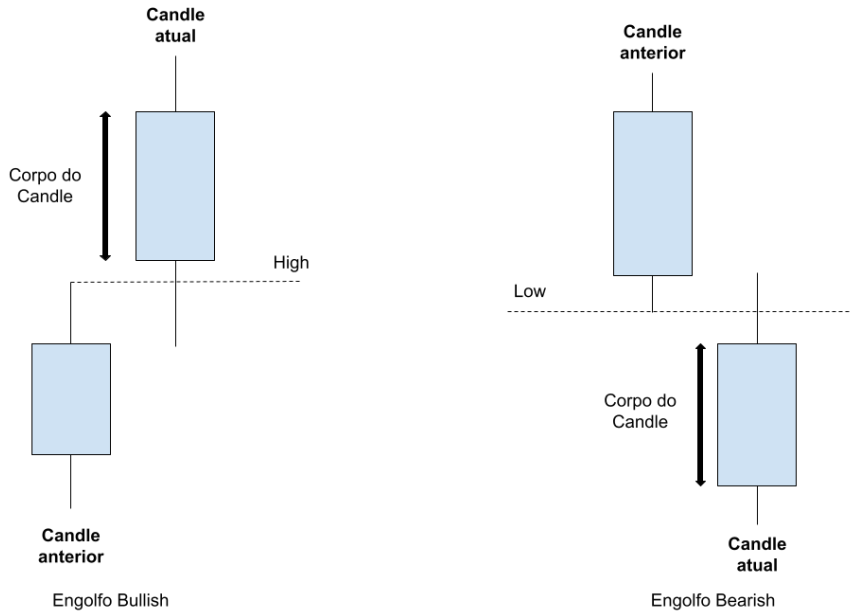
Nesse exemplo, os dados dos *candles* são carregados de um arquivo CSV chamado 'dados_candles.csv'. A função *ta.atr()* é utilizada para calcular o ATR, passando as colunas dos preços de alta (*High*), baixa (*Low*) e fechamento (*Close*) dos *candles*. O parâmetro *period* é usado para definir o período do ATR, que nesse caso é 14. Os valores calculados do ATR são adicionados como uma nova coluna chamada 'ATR' no *dataframe*.

4.3.1.3 Engolfo

O padrão Engolfo é um comportamento observado em gráficos, no qual o próximo *candle* ultrapassa os extremos do *candle* anterior. Quando o início do corpo do *candle* atual é maior que o valor máximo do *candle* anterior, é chamado de Engolfo *Bullish*. Por outro lado, quando o início do corpo do *candle* atual é menor que o valor mínimo do *candle* anterior, é chamado de Engolfo *Bearish*, esses padrões podem ser observados Figura 37.

No *dataframe*, a coluna denominada "Engolfo"representa o padrão Engolfo *Bullish* e *Bearish* por meio dos valores 1 e -1, respectivamente. Para os *candles* em que esse comportamento não ocorre, é atribuído o valor 0. Essa representação numérica facilita a identificação e análise dos padrões de Engolfo nos dados.

Figura 37 – Identificação de Engolfo



Fonte: Autor (2023)

4.3.1.4 FuCandle

O FuCandle é um *candle* que apresenta um pavio bastante alongado e um corpo pequeno, sendo que esse pavio é 61.8% maior em comparação ao tamanho médio dos corpos de "candles". O tamanho médio dos corpos é determinado através do cálculo do ATR.

Figura 38 – Identificação de FuCandle

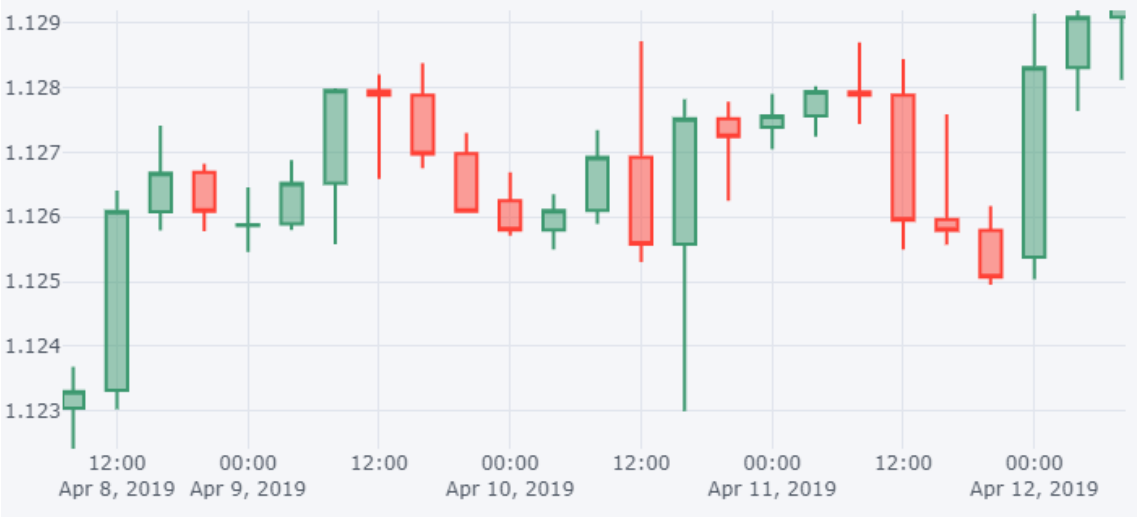
Fonte: Autor (2023)

Na Figura 38, é possível identificar os FuCandle que estão destacados por meio de um retângulo vermelho. Observa-se que os pavios são maiores em comprimento do que os corpos dos *candles*.

4.3.1.5 Vela Bull

A criação da coluna "VelaBull"consiste em um valor *booleano* que indica o tipo de *candle* representado, sendo *True* para um *Candle Bullish*, indicando valorização, e *False* para um *Candle Bearish*, indicando desvalorização. Essa coluna auxilia na categorização e análise dos diferentes tipos de *candles* presentes nos dados.

Figura 39 – Identificação de Candles



Fonte: Autor (2023)

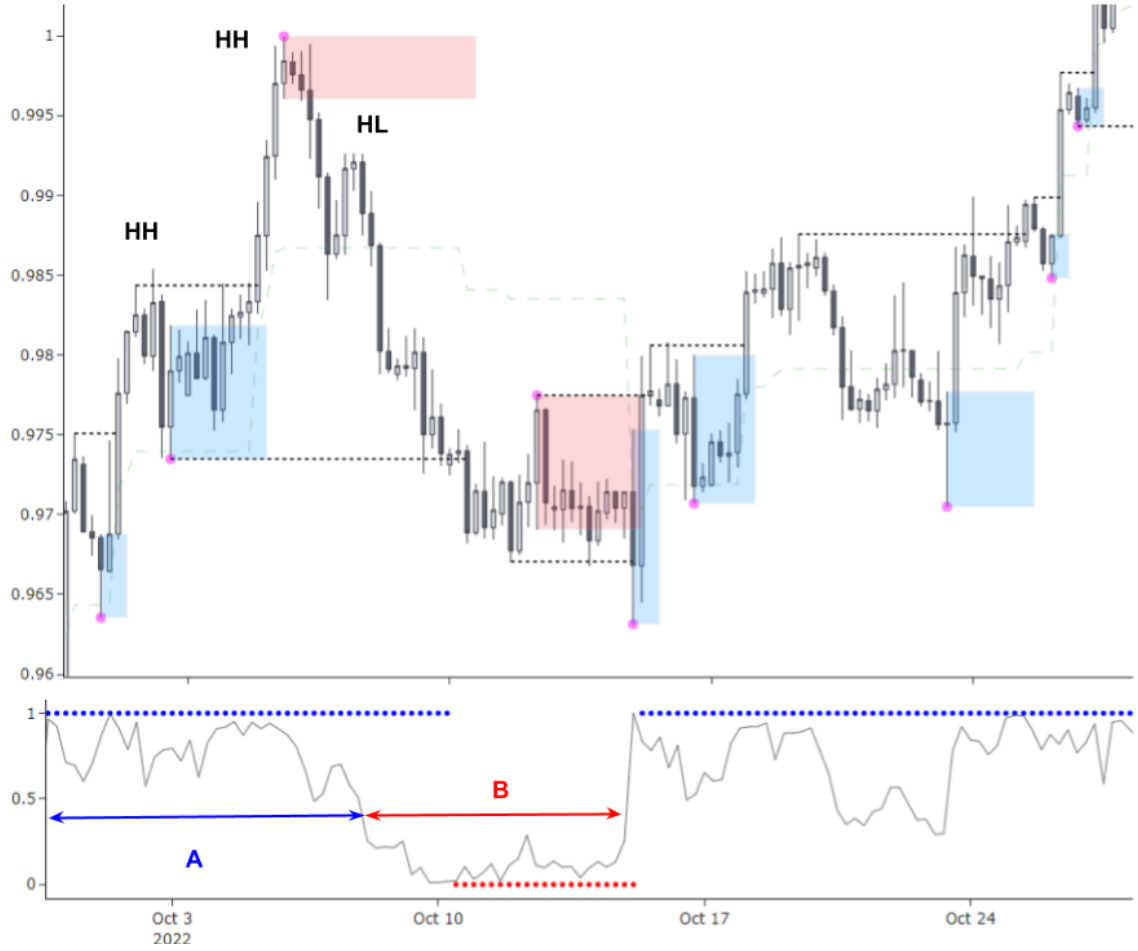
Na Figura 39, é possível identificar os diferentes tipos de *Candle*. Os *candles* de cor verde representam *Candle Bullish*, enquanto os de cor vermelha representam *Candle Bearish*.

4.3.1.6 Structure

No cálculo da Estrutura do Mercado, são identificados os topos e fundos do gráfico. Um topo é considerado consolidado quando o fechamento do candle é abaixo da mínima do candle anterior, indicando uma reversão na tendência. O engolfo ocorre quando um candle "engole"o anterior, ou seja, o corpo do candle atual é maior que o corpo do candle anterior, e esse padrão é considerado uma consolidação do topo.

As colunas "Structure", "StructureTop", "StructureBottom"e "StructPD"foram criadas para capturar informações sobre a estrutura dos dados do ativo Forex (EURUSD). Essas colunas são usadas para analisar os topos e fundos do gráfico, a fim de determinar se o período em questão é *Bullish* (valorização) ou *Bearish* (desvalorização).

Figura 40 – Estrutura do mercado



Fonte: Autor (2023)

O rompimento do topo é identificado quando os próximos *candles* ultrapassam o valor máximo do último topo, indicando uma possível continuação da tendência de alta. Da mesma forma, o rompimento do fundo é identificado quando os próximos *candles* ultrapassam o valor mínimo do último fundo, indicando uma possível continuação da tendência de baixa.

A tendência é considerada *Bullish* quando o último rompimento foi para cima, ou seja, ocorreu o rompimento do topo. Por outro lado, a tendência é considerada *Bearish* quando o último rompimento foi do fundo, indicando uma reversão na tendência de alta.

Em resumo, o cálculo da Estrutura do Mercado busca identificar os padrões de rompimento de topos e fundos, permitindo classificar a tendência do gráfico como *Bullish* ou *Bearish*.

Na Figura 40, é possível observar a tendência do mercado. No início do gráfico, são identificados *Higher High* (HH), indicando uma tendência de valorização. Em

seguida, é identificado um *Higher Low*, indicando uma inversão na tendência. Na parte inferior do gráfico, é apresentado o *Price Discount*, que indica se o valor do ativo está alto em comparação com as informações anteriores. No período A, quando a estrutura apresenta uma tendência de valorização, é observado que o *Price Discount* é alto. No período B, é observado que o valor do *Price Discount* está baixo, indicando que o ativo está mais barato. Isso ocorre porque a estrutura está em uma tendência de baixa.

No *Dataframe* a coluna "Structure"representa a estrutura geral do período, indicando se é *Bullish*, *Bearish* ou neutro. As colunas "StructureTop"e "StructureBottom"registram os valores dos topos e fundos encontrados no período, respectivamente. Por fim, a coluna "StructPD", que representa o *Price Discount*, é responsável por indicar a estrutura do gráfico com base na análise dos topos e fundos se o ativo esta com valor alto ou baixo, com parando com os valores anteriores.

Essas colunas são úteis para identificar padrões e tendências nos dados de mercado, auxiliando na tomada de decisões de negociação.

4.3.2 Classificação das operações

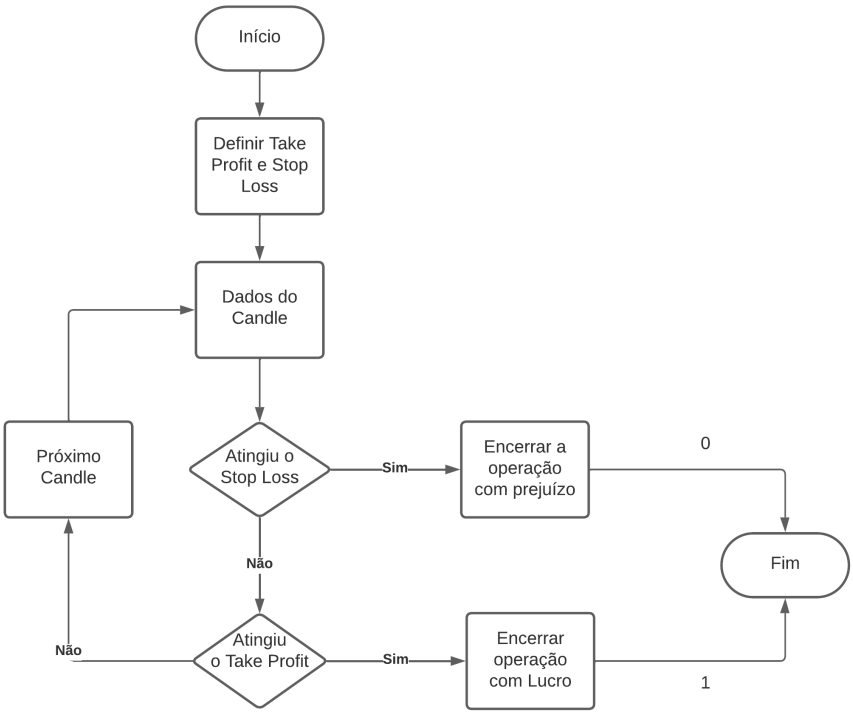
As classificações das operações foram definidas como 0 para "não opera"e 1 para "opera". As trades são realizadas em dois tipos: *Long* (compra) e *Short* (venda a descoberto).

Foram configurados níveis de *Take Profit* (TP) e *Stop Loss* (SL) para as operações. Quando uma operação atinge o nível de *Take Profit*, é classificada como 1, indicando que foi bem-sucedida. Por outro lado, se a operação atinge o nível de *Stop Loss*, é classificada como 0, indicando que foi encerrada com prejuízo. Caso o candle atual não atinja nenhum valor de Take Profit ou Stop Loss, é analisado o próximo candle até que um desses valores seja alcançado. Esse processo de análise é repetido até que uma das condições de encerramento da operação seja satisfeita, conforme é apresentado na Figura 41.

Essas configurações permitem avaliar o desempenho das operações e tomar decisões com base nos níveis de TP e SL estabelecidos, visando maximizar os lucros e controlar as perdas.

No Fluxograma mencionado acima, é necessário definir o tamanho do *stop* e o risco-retorno. Após definir os valores do *stop* e do tamanho do *stop* são identificados no *Dataframe*, e estabelecido que o *take profit* será três vezes o tamanho do *stop*, conforme é apresentado na Figura 42. É estabelecido que o lucro deve ser três vezes o tamanho

Figura 41 – Fluxograma do *Take Profit* e *Stop Loss*



Fonte: Autor (2023)

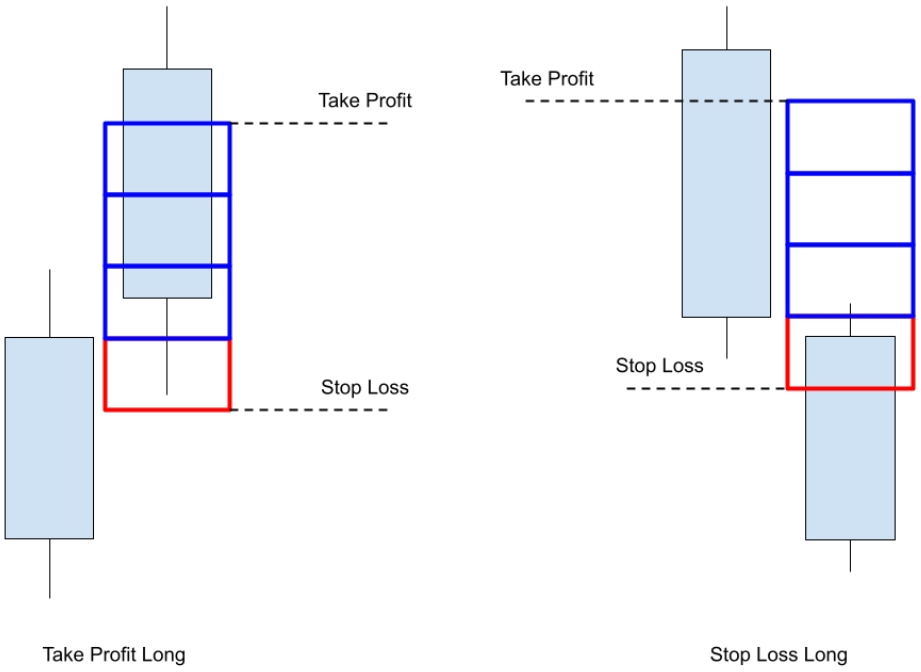
do *Stop loss* para que o lucro possa cobrir o prejuízo. Essa proporção é utilizada porque, se aumentarmos a quantidade, pode ocorrer menos vezes o *Take profit* (momento em que se encerra a operação com lucro). Portanto, utilizar o valor de 3, que é mais utilizada pelos *traders*, é uma quantidade que tende a ocorrer mais vezes, garantindo uma relação favorável entre risco e recompensa nas operações.

Em seguida, o código percorre todo o *Dataframe* para identificar quando ocorrem os eventos de *stop* e *take profit*. Esses eventos são então codificados com 0 e 1 nas colunas "Long"e "Short", respectivamente, completando assim o *Dataframe* com as informações necessárias.

Essa abordagem permite rastrear e registrar as operações de acordo com os valores definidos para o *stop*, tamanho do *stop* e *take profit*, facilitando a análise posterior do desempenho das trades.

Por fim, os dados foram finalizados e estão prontos para serem analisados e processados pelos modelos de Machine Learning. Na Figura 43, são apresentados os dados iniciais provenientes do Banco de Dados, os valores do ICT que foram gerados e, por fim, as duas colunas "Long"e "Short"que representam a classificação de operações e não operações.

Figura 42 – Classificação Long



Fonte: Autor (2023)

Figura 43 – Dados Finais

Open time	Open	High	Low	Close	ATR	EngolfO	FuCandle	Structure	StructTop	StructBottom	StructPD	VelaBull	ImbLow	ImbHigh	Long	Short
2000-02-17 04:00:00+03:00	0.98530	0.98740	0.98490	0.98610	0.00380	0	0	1	0.99470	0.96670	0.692857	True	0.00000	0.00000	0	0
2000-02-17 08:00:00+03:00	0.98630	0.99580	0.98510	0.99380	0.00429	1	0	1	0.99470	0.96670	0.967857	True	0.98740	0.98840	0	0
2000-02-17 12:00:00+03:00	0.99370	0.99440	0.98840	0.98980	0.00441	0	0	1	0.99470	0.96670	0.825000	False	0.00000	0.00000	0	0
2000-02-17 16:00:00+03:00	0.98990	0.99270	0.98370	0.98830	0.00474	-1	0	1	0.99470	0.96670	0.771429	False	0.00000	0.00000	0	0
2000-02-17 20:00:00+03:00	0.98850	0.98920	0.98680	0.98800	0.00457	0	0	1	0.99470	0.96670	0.760714	False	0.00000	0.00000	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2022-08-31 08:00:00+03:00	1.00403	1.00460	0.99741	0.99793	0.00411	-1	0	-1	1.00955	0.99004	0.404408	False	0.00000	0.00000	0	0
2022-08-31 12:00:00+03:00	0.99794	1.00184	0.99714	1.00084	0.00416	0	0	-1	1.00955	0.99004	0.553562	True	0.00000	0.00000	0	0
2022-08-31 16:00:00+03:00	1.00085	1.00790	0.99952	1.00535	0.00446	1	0	-1	1.00955	0.99004	0.784726	True	1.00184	1.00355	0	0
2022-08-31 20:00:00+03:00	1.00534	1.00574	1.00355	1.00545	0.00430	0	0	-1	1.00955	0.99004	0.789851	True	0.00000	0.00000	0	0
2022-09-01 00:00:00+03:00	1.00510	1.00545	1.00200	1.00234	0.00423	-1	0	-1	1.00955	0.99004	0.630446	False	0.00000	0.00000	0	0

Dados iniciais | Valores do ICT | Classificação

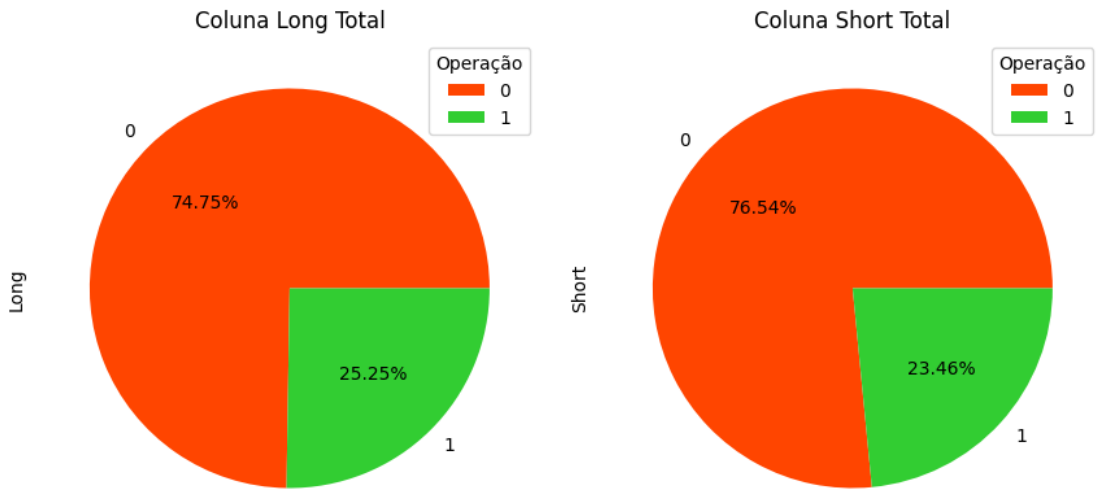
Fonte: Autor (2023)

4.3.3 Análise dos dados

Após a formação do *dataframe* final, que abrange 35.098 registros, constata-se uma tendência de maior predominância do valor zero em relação ao valor um nas colunas *Long* e *Short*. Mais especificamente, o *dataframe* apresenta 26.237 ocorrências de zero e 8.861 ocorrências de um na coluna *Long*, enquanto na coluna *Short* registra-se 26.863

ocorrências de zero e 8.235 ocorrências de um. A distribuição percentual das operações é ilustrada na Figura 44.

Figura 44 – Proporção dos registros do dataframe



Fonte: Autor (2023)

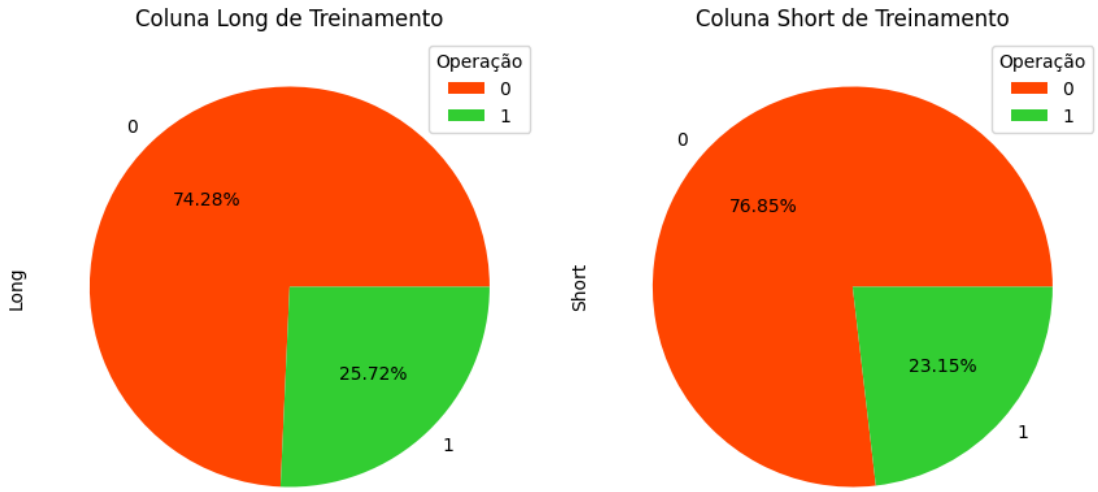
Após a análise dos dados, procedeu-se à divisão dos mesmos em conjuntos de treinamento e teste, utilizando uma proporção de 75% para treinamento e 25% para teste. Os valores da variável ICT foram atribuídos à variável x, enquanto as operações *Long* e *Short* foram representadas pelas variáveis y e z, respectivamente.

Após a divisão dos dados em conjuntos de treinamento e teste, treino são dados de 2000 até 2017, e dados de testes são de 2017 até 2022, foram obtidos 26.323 registros para o conjunto de treinamento e 8.775 registros para o conjunto de teste. Os dados foram então utilizados para treinar o modelo e posteriormente classificá-los. No entanto, ao realizar a classificação, os modelos apresentaram uma acurácia de 75% e retornaram apenas valores de zero, indicando a ausência de operações nos dados de teste.

Essa situação ocorreu devido ao desbalanceamento dos dados de treinamento, como pode ser observado na Figura 45. Os dados de treinamento apresentam uma maior proporção de não operações, indicadas pelo número 0, em relação às operações, indicadas pelo número 1, em termos percentuais.

Para solucionar esse problema, foram utilizadas técnicas de *Oversampling* e *Undersampling*. O *Undersampling* consiste na redução dos registros da classe majoritária (no nosso caso, eliminaríamos a quantidade de classificações 0), enquanto o *Oversampling* envolve o aumento dos registros da classe minoritária (no nosso caso,

Figura 45 – Proporção dos registros de Treinamento



Fonte: Autor (2023)

aumentaríamos a quantidade de classificações 1). Essas técnicas foram aplicadas aos dados de treinamento após a utilização da função de divisão dos dados em treinamento e teste (*split_train_test*).

Após a aplicação dessas técnicas, a proporção dos registros nos dados de treinamento é igual a 50% para a classificação 0 (não operar) e 50% para a classificação 1 (operar).

No caso do *Undersampling*, os registros com a classificação 0 foram removidos para igualar a quantidade de registros com a classificação 1. Antes da aplicação do *Undersampling*, havia 6.771 registros com a classificação 1 na coluna *Long* e 6.093 registros com a classificação 1 na coluna *Short*. Portanto, após a aplicação do *Undersampling*, restaram 13.542 linhas na coluna *Long* e 12.186 linhas na coluna *Short* para o treinamento.

Antes da aplicação do *Oversampling*, havia 19.552 registros com a classificação 0 na coluna *Long* e 20.230 registros com a classificação 0 na coluna *Short*. Após a aplicação do *Oversampling*, os dados totais de treinamento passaram a ter 39.104 linhas na coluna *Long* e 40.460 linhas na coluna *Short*.

Após a finalização do processo de preparação dos dados de treinamento, o próximo passo é treinar os modelos utilizando os dados gerados pelo *Undersampling* e *Oversampling*. Esses dados balanceados serão fornecidos aos modelos para que eles possam aprender e se ajustar aos padrões e características dos dados. O treinamento

dos modelos permite que eles capturem os padrões e relacionamentos nos dados do ICT, possibilitando a realização das classificações de onde operar com *Long* e *Short*.

4.3.4 Parâmetros dos modelos de *Machine Learnig*

Com os dados de treinamento e teste prontos para o processo de modelagem, foi tomada a decisão de utilizar modelos com parâmetros padrão e também modelos com parâmetros modificados.

A utilização de modelos com parâmetros padrão é útil para fins de comparação e referência, pois representa a configuração básica do algoritmo sem qualquer ajuste personalizado. Isso nos permite avaliar o desempenho dos modelos com os parâmetros padrão e compará-lo com os resultados obtidos após a modificação dos parâmetros.

Durante os testes preliminares do modelo, foi observado que as execuções produziam resultados promissores com pontuações ótimas, mesmo ao utilizar os algoritmos de aprendizado de máquina em suas configurações padrão. Foram realizados testes abrangentes por meio de tentativa e erro para selecionar os parâmetros mais adequados. No entanto, esse processo pode se tornar exaustivo e resultar em trabalho desnecessário, uma vez que uma pequena alteração nos períodos ou nos valores de entrada pode invalidar toda a seleção de parâmetros realizada anteriormente.

Para contornar essas situações, optou-se por realizar uma busca exaustiva de hiperparâmetros de forma automatizada. Os hiperparâmetros são as variáveis responsáveis por controlar o fluxo do processo de treinamento dos modelos. A solução adotada foi a inclusão do algoritmo *GridSearchCV* no modelo. Esse algoritmo realiza uma busca exaustiva automatizada, testando todas as combinações possíveis dos parâmetros especificados para cada estimador (algoritmo de classificação e/ou regressão utilizado para realizar previsões). Ele identifica os melhores parâmetros que maximizam o desempenho do modelo. Dessa forma, o *GridSearchCV* permite encontrar de maneira eficiente os conjuntos de parâmetros mais adequados para obter um desempenho otimizado dos modelos.

Nos testes preliminares realizados, observou-se que a busca exaustiva utilizando a função *GridSearchCV* resulta em um tempo de execução praticamente maior em comparação com os modelos com hiperparâmetros padrão. Além disso, o processamento dos dados de treinamento gerados pelo *Oversampling* levou mais tempo em comparação com os dados de *Undersampling*, devido à maior quantidade de dados.

Especificamente, ao utilizar os dados de treinamento do *Oversampling* na modelagem do SVM com *GridSearchCV*, o tempo de processamento foi de aproximadamente 176 minutos. Por outro lado, com os dados de treinamento do *Undersampling*, o tempo de processamento foi de cerca de 18 minutos. Essa diferença de tempo se deve ao fato de que o *Oversampling* gera um maior número de registros de treinamento, o que demanda mais tempo para processar e treinar o modelo.

É correto afirmar que o tempo de execução do algoritmo *GridSearchCV* é proporcional à quantidade de hiperparâmetros recebidos para cada modelo. O algoritmo realiza uma busca exaustiva em todas as combinações possíveis dos parâmetros especificados e avalia o desempenho do estimador com cada conjunto de parâmetros. Portanto, o tempo de execução pode ser maior quando há um grande número de parâmetros a serem testados.

Apesar do tempo de execução elevado em algumas buscas de melhora de parâmetros, esse esforço é necessário para obter melhores resultados e melhorar a efetividade do modelo. O algoritmo avalia todos os scores do estimador, buscando a combinação ideal de parâmetros que maximize o desempenho. Embora isso possa tornar o processo um pouco mais lento, é uma etapa importante para atingir resultados otimizados e obter um modelo mais eficaz.

Para cada modelo, foram selecionados parâmetros estratégicos que apresentaram mudanças significativas durante os testes de tentativa e erro. A seguir está a lista de parâmetros escolhidos para o algoritmo de busca exaustiva e descritas de acordo com *SKlearn*:

1. SVM:
    - C: parâmetro de regularização
    - kernel: tipo de *kernel* a ser utilizado
    - gamma: parâmetro para *kernels*
2. Árvore de decisão:
    - criterion: A função para medir a qualidade de uma divisão
    - splitter: A estratégia usada para escolher a divisão em cada nó
    - max_depth: profundidade máxima da árvore
    - random_state: Controla a aleatoriedade do estimador
3. Floresta Aleatória:

- n_estimators: número de árvores no conjunto
- criterion: A função para medir a qualidade de uma divisão
- max_depth: profundidade máxima das árvores

4. Rede Neural:

- max_inter: Número máximo de iterações
- hidden_layer_sizes: O i-ésimo elemento representa o número de neurônios na i-ésima camada oculta
- alpha: Força do termo de regularização
- learning_rate: Cronograma de taxa de aprendizado para atualizações de peso
- momentum: Momentum para atualização do gradiente descendente

Esses parâmetros foram selecionados com base na experiência e conhecimento prévio sobre os modelos, bem como em testes preliminares que mostraram sua influência significativa no desempenho dos modelos. Durante a busca exaustiva, o algoritmo *GridSearchCV()* avaliará diferentes combinações desses parâmetros para encontrar a melhor configuração para cada modelo. Com o *dataframe* finalizado e os parâmetro a serem modificados, foi iniciado o treinamento e os testes dos modelos de ML.

O código do TradeClassifier é disponibilizado nos apêndices do trabalho. No Apêndice A, encontra-se o código para os cálculos do ICT. No Apêndice B, está o código para a classificação das operações *Long* e *Short*. O Apêndice C contém o código do Balanceamento de Dados e das técnicas de *Machine Learning*. Por fim, no Apêndice D, é apresentado o código que realizou a simulação com as classificações feitas pelos modelos.

Também é possível acessar o código no repositório do Github no link fornecido: https://github.com/marcoajt/Project_ML_FM.

5 RESULTADOS E DISCUSSÕES

Este capítulo visa abordar os experimentos realizados e os resultados encontrados durante a implementação e realização dos experimentos. A Seção 5.1 descreve a situação atual do trabalho quanto à finalização e possíveis testes. A Seção 5.2 descreve todo o processo de experimentos e testes realizados, seus respectivos resultados e análises obtidas do processo de validação e simulação (com cálculos e lógicas) e os objetos gráficos para a visualização do lucro ao decorrer do período de testes.

5.1 Situação atual

O modelo está atualmente totalmente funcional e pronto para ser testado em uma execução completa. Durante o desenvolvimento, foram obtidos resultados consistentes e não foram identificados erros de compilação. As visualizações gráficas forneceram uma análise detalhada do lucro ao utilizar diferentes modelos de aprendizado de máquina, possibilitando uma melhor compreensão dos momentos de perdas e ganhos.

Em termos de desempenho, o tempo de execução foi considerado satisfatório quando os dados são processados em modelos com parâmetros padrão. Entretanto, quando é utilizado o *GridSearchCV()* para a procura dos melhores parâmetros, o tempo de execução aumenta drasticamente. A utilização do Visual Studio Code, juntamente com a extensão do Jupyter, mostrou-se uma plataforma adequada para o desenvolvimento, organização e a execução do modelo.

5.2 Testes realizados

Esta seção visa abordar todos os resultados obtidos nos testes realizados. A seção de testes foi dividida em 2 subseções, uma abordando os *scores* com os parâmetros padrões de cada um dos modelos de machine learning (Subseção 5.2.1), a segunda utilizando a configuração permitida pela biblioteca (Seção 5.3).

5.2.1 Testes dos modelos com configurações padrões

O objetivo das execuções com parâmetros *default* é observar a capacidade e eficiência do código em seu fluxo de execução com os algoritmos de *machine learning* com configuração padrão (*default*).

Foram conduzidos três testes utilizando os modelos em sua configuração padrão, sem a realização de uma busca intensiva pelos melhores parâmetros. O propósito dessas execuções com parâmetros *default* (padrão) é observar a habilidade e eficiência do script em seu fluxo de execução com os algoritmos de *machine learning* em sua forma original. Além disso, uma vantagem adicional da utilização sem a busca intensiva é a notável redução no tempo de execução do código.

A Tabela 6 apresenta o desempenho de generalização de cada um dos métodos, na busca por uma acurácia das operações *Long* e *Short*, utilizando os parâmetros padrões de cada modelo e os dados *Undersampling* e *Oversampling*.

Tabela 6 – Resultados da acurácia dos modelos

Testes	Modelo	Under Long	Under Short	Over Long	Over Short
1	SVM	53.03%	48.88%	54.21%	53.90%
1	Árvore de decisão	58.47%	57.64%	61.98%	63.70%
1	Floresta Aleatória	58.46%	60.84%	68.44%	69.19%
1	Redes Neurais	44.48%	58.14%	50.36%	69.04%
2	SVM	53.03%	48.88%	54.21%	53.90%
2	Árvore de decisão	57.68%	57.93%	62.35%	64.39%
2	Floresta Aleatória	58.88%	59.81%	68.14%	69.19%
2	Redes Neurais	42.79%	60.42%	54.74%	70.31%
3	SVM	53.03%	48.88%	54.21%	53.90%
3	Árvore de decisão	58.59%	58.28%	62.51%	63.17%
3	Floresta Aleatória	59.36%	60.15%	68.76%	68.93%
3	Redes Neurais	50.95%	56.44%	45.77%	68.96%

Fonte: Autor (2023)

A coluna Modelo representa o algoritmo responsável pela execução das operações. O lado direito da tabela são os resultados da acurácia dos modelos dos tipos de operação financeira *Long* e *Short*, com os dados utilizando as técnicas de balanceamento dos dados de treino, *Undersampling* e *Oversampling*, que na tabela é representada por *Under* e *Over*.

Os resultados que utilizaram a técnica de *Oversampling* apresentaram uma melhora na acurácia em comparação com o *Undersampling*. Isso ocorre porque o *Undersampling* reduz a quantidade de dados da classe majoritária, o que pode levar à

perda de informações relevantes para o treinamento do modelo.

No entanto, a acurácia pode não ser a métrica mais adequada para avaliar o desempenho e maximizar o lucro no mercado financeiro. Isso porque na Tabela 6 os modelos Árvore de Decisão e Floresta Aleatória foi relativamente alta em comparação com os outros modelos. No entanto, ao realizar a simulação de trades, esses modelos não conseguiram gerar um lucro significativo.

Isso destaca uma importante distinção entre a acurácia do modelo e sua capacidade de gerar resultados financeiros favoráveis. A acurácia mede a taxa de acertos do modelo nas previsões, mas não leva em consideração a magnitude dos ganhos ou perdas nas operações. Portanto, é possível ter um modelo com alta acurácia, mas que não seja lucrativo em termos financeiros.

Nesse contexto, foi aplicado a métrica matriz de confusão. A matriz de confusão fornece uma visão mais completa do desempenho do modelo, permitindo analisar os verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos. Essa informação é valiosa para entender como o modelo está classificando corretamente e incorretamente os dados.

Os verdadeiros positivos representam os casos em que o modelo previu corretamente uma operação lucrativa (por exemplo, prevendo uma operação *Long* que realmente gerou lucro). Ao aumentar a quantidade de verdadeiros positivos, o modelo está acertando mais vezes em identificar as oportunidades lucrativas.

Por outro lado, os falsos negativos representam os casos em que o modelo classificou erroneamente uma operação como "não operar", quando, na verdade, deveria ter operado e gerado lucro. Ao diminuir a quantidade de falsos negativos, o modelo está reduzindo os casos em que perdeu oportunidades de lucro.

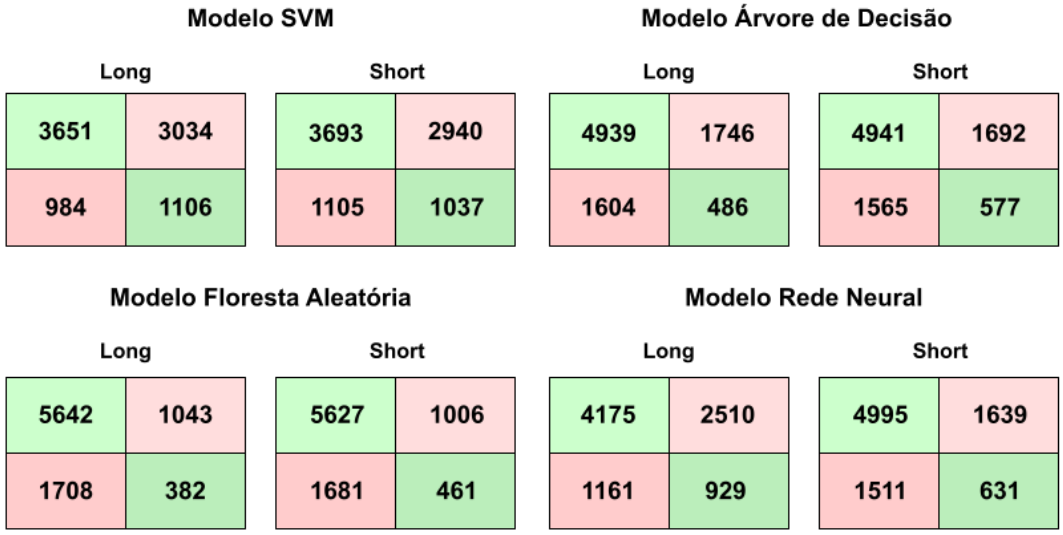
A quantidade de falsos positivos é um aspecto importante a ser considerado. Esses são os casos em que o modelo classifica erroneamente uma operação como "operar"quando, na verdade, deveria ter sido classificada como "não operar". Isso pode levar a prejuízos e acionamento de *stop loss* para limitar as perdas de capital.

Por fim os verdadeiros negativos representam os casos em que o modelo classificou corretamente uma operação como "não operar"quando, de fato, não deveria ter ocorrido uma operação. Essas são as situações em que o modelo evitou operações desnecessárias e potencialmente prejudiciais, protegendo o capital de investimento.

A ocorrência de falsos positivos indica que o modelo está identificando oportunidades de negociação que não são realmente lucrativas, resultando em operações

que podem levar a perdas financeiras. É importante acompanhar e analisar a taxa de falsos positivos, pois ela pode afetar significativamente o desempenho geral do modelo.

Figura 46 – Matriz de confusão com dados Oversampling



Fonte: Autor (2023)

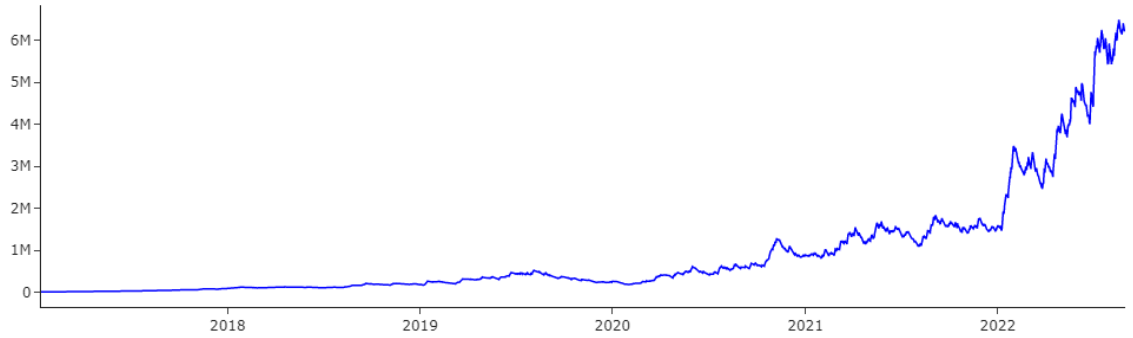
A matriz de confusão dos modelos que utilizaram os dados *Oversampling* é apresentado na Figura 46. É observado que na matriz de confusão dos modelos de Árvore de Decisão e Floresta Aleatória, houve um número significativo de falsos negativos, ou seja, casos em que o modelo classificou erroneamente a operação como "não operar"quando, na verdade, deveria ter operado. Isso resultou em uma perda de oportunidades de lucro e contribuiu para a queda do capital inicial.

5.2.2 Trading com modelos utilizando parâmetros padrão

Os resultados são divididos em duas partes, utilizando tanto o *Under Sampling* quanto o *Over Sampling*, ambos com modelos que possuem parâmetros padrões. A configuração da simulação são capital inicial de U$10,000.00 e a compra de 100,000 lotes em cada operação.

Na Figura 47, é possível observar um gráfico crescente do ganho ao utilizar o modelo SVM. O capital final alcançado foi de 6,216,854.01. Esses resultados indicam que, mesmo utilizando os parâmetros padrões do modelo SVM, é possível obter um lucro significativo.

Figura 47 – Trade SVM com parâmetros padrões com dados Undersampling



Fonte: Autor (2023)

Essa visualização demonstra a capacidade do modelo SVM em identificar oportunidades lucrativas no mercado financeiro, mesmo sem uma otimização dos parâmetros. Essa descoberta encorajadora sugere que o modelo SVM, com os parâmetros padrões, pode ser uma opção promissora para realizar operações no mercado financeiro com potencial de obter retornos positivos.

Por outro lado, na Figura 49, observa-se que o modelo de Árvore de Decisão apresentou pequenas perdas no início das operações, seguidas de um período de lucro até 2018. No entanto, a partir desse ponto, o modelo não classificou adequadamente as demais operações, resultando em perdas e mantendo o capital abaixo do valor inicial.

Figura 48 – Trade Árvore de Decisão com parâmetros padrões com dados Undersampling

Fonte: Autor (2023)

Essa observação indica que o desempenho do modelo de Árvore de Decisão pode ser instável e inconsistente ao longo do tempo. É possível que o modelo tenha dificuldade em capturar as mudanças e tendências mais recentes do mercado, levando a resultados menos favoráveis.

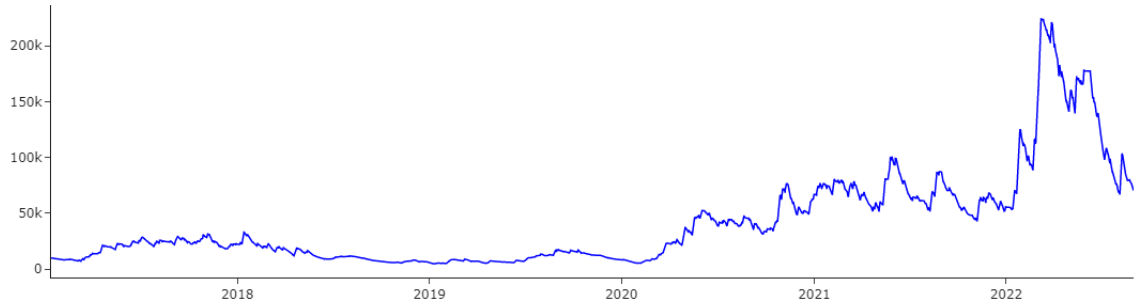
Esses resultados ressaltam a importância de monitorar e avaliar o desempenho dos

modelos ao longo do tempo, especialmente em cenários financeiros, onde a precisão e a consistência das previsões são essenciais.

Em resumo, enquanto o modelo de Árvore de Decisão mostrou um início promissor com lucros, seu desempenho posterior foi afetado negativamente, resultando em perdas e um capital final de 1,585.95, abaixo do valor inicial.

O modelo de Floresta Aleatória apresentou uma classificação melhor em comparação com o modelo de Árvore de Decisão. No entanto, é importante observar que o modelo de Floresta Aleatória teve um bom desempenho no ano de 2022, apos de atingir o topo acima de 200,000.00, o modelo não conseguiu assegurar o lucro obtido, resultando em uma queda no capital acumulado. Apesar dessa queda, o trade acumulou um valor final de 70,294.12.

Figura 49 – Trade Floresta Aleatória com parâmetros padrões com dados Undersampling



Fonte: Autor (2023)

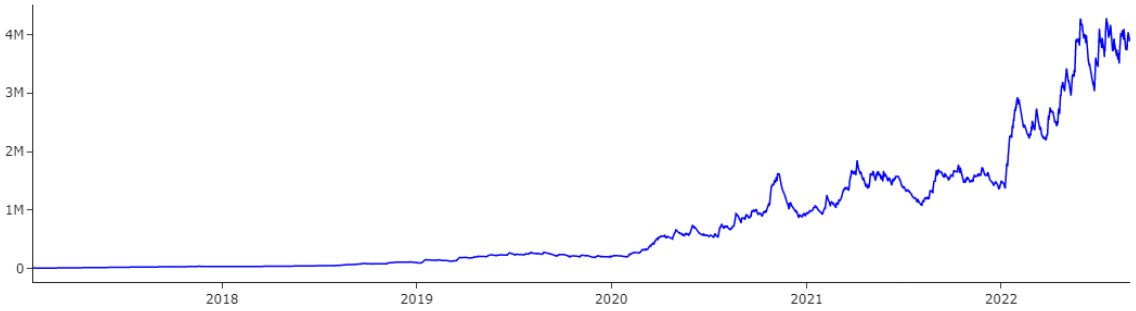
Essa análise indica que o modelo de Floresta Aleatória foi capaz de obter resultados melhores em comparação com a Árvore de Decisão na maior parte do período avaliado. No entanto, a má classificação no ano de 2022 sugere que o modelo pode ter tido dificuldades em lidar com as mudanças e tendências específicas desse período, levando a perdas no capital.

Apesar da queda no valor acumulado, é importante ressaltar que o trade ainda obteve um resultado positivo, alcançando um valor final de 70,294.12. Isso indica que o modelo de Floresta Aleatória conseguiu capturar oportunidades lucrativas ao longo do tempo, mesmo que tenha enfrentado dificuldades em determinados períodos.

Na Figura 50, é apresentado o gráfico do acúmulo de capital utilizando o modelo de Rede Neural com os parâmetros padrões. Assim como o modelo SVM, esse modelo também apresentou um comportamento crescente no acúmulo de capital ao longo do tempo.

É interessante observar que, apesar de ter um comportamento similar ao do modelo

Figura 50 – Trade Rede Neural com parâmetros padrões



Fonte: Autor (2023)

SVM, o modelo de Rede Neural obteve um capital final de 3,896,678.12, que é inferior ao alcançado pelo modelo SVM. Isso indica que, mesmo com um desempenho positivo, o modelo de Rede Neural pode não ter sido tão eficaz quanto o SVM na maximização do lucro no mercado financeiro.

É importante ressaltar que os resultados podem variar dependendo do conjunto de dados, das condições de mercado e de outros fatores. No entanto, essa comparação sugere que, considerando os parâmetros padrões, o modelo SVM pode ter uma vantagem em termos de lucratividade em relação ao modelo de Rede Neural.

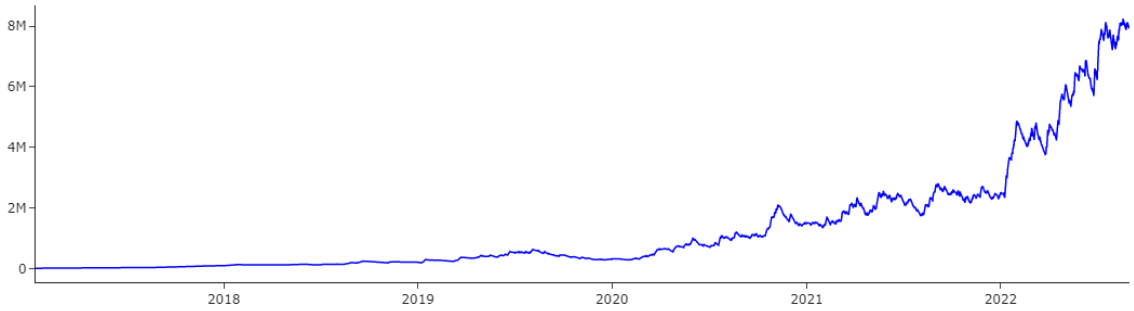
Em conclusão, ao analisar os modelos com parâmetros padrões utilizando os dados com *Undersampling*, foi observado um alto lucro, com exceção do modelo de Árvore de Decisão. No entanto, é importante ressaltar que o uso do *Undersampling* pode resultar na perda de informações essenciais, uma vez que reduz a quantidade de exemplos da classe majoritária.

Para uma comparação mais abrangente, foram realizadas simulações adicionais utilizando os dados com *Oversampling*. O *Oversampling* é uma técnica que visa equilibrar as classes, aumentando a quantidade de exemplos da classe minoritária. Essa abordagem permite que o modelo aprenda de forma mais equilibrada e obtenha classificações mais precisos.

Ao analisar os modelos utilizando dados *Oversampling*, é observado que o modelo SVM continua sendo o melhor classificador, assim como foi observado com os dados *Undersampling*. Além disso, é importante notar que o gráfico do acúmulo de capital do modelo SVM com dados *Oversampling* mantém um padrão similar ao observado com os dados *Undersampling*.

No entanto, a diferença significativa está no montante de arrecadação alcançado. Com o uso de dados *Oversampling*, o modelo SVM obteve um capital final de

Figura 51 – Trade SVM com parâmetros padrões



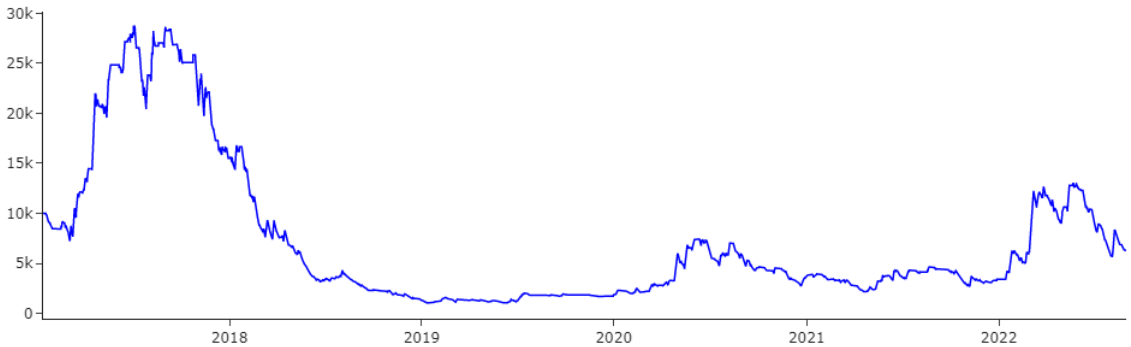
Fonte: Autor (2023)

7,933,523.59, o que representa um aumento substancial em comparação com os resultados obtidos com dados *Undersampling*.

Esses resultados destacam a importância do tratamento adequado do desbalanceamento de classes, como é realizado pelo *Oversampling*. Ao equilibrar a distribuição das classes, o modelo é capaz de aprender de forma mais precisa e capturar padrões relevantes, resultando em uma maior capacidade de identificar oportunidades lucrativas no mercado financeiro.

Ao avaliar o desempenho do modelo de Árvore de Decisão utilizando dados *Oversampling*, é importante destacar que ele continua apresentando um resultado inferior em termos de arrecadação em comparação com os outros modelos. No entanto, é positivo observar que, mesmo com esse desempenho inferior, o modelo não terminou com um prejuízo maior em relação ao modelo que utilizou os dados *Undersampling*. Na Figura 52 é apresentado o gráfico do capital durante o período de testes.

Figura 52 – Trade Árvore de Decisão com parâmetros padrões



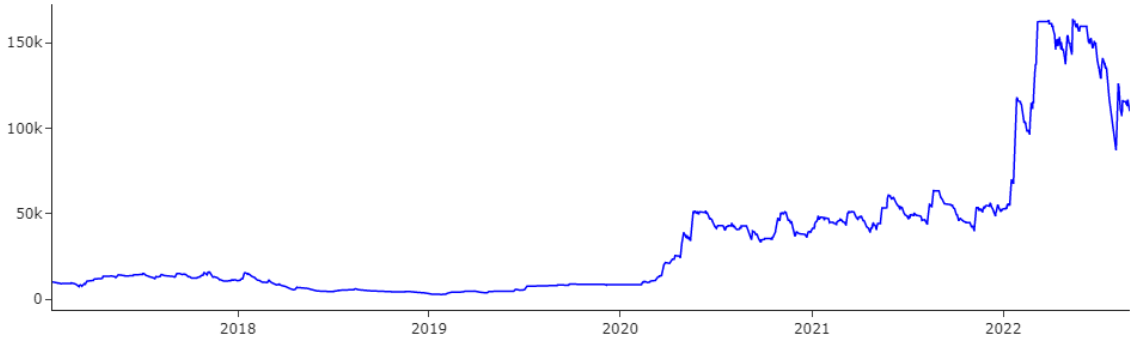
Fonte: Autor (2023)

Com um capital final de 6,277.48, o modelo de Árvore de Decisão utilizando dados *Oversampling* conseguiu minimizar as perdas e encerrar o período de negociação

sem um prejuízo maior. Embora não tenha alcançado um lucro tão expressivo quanto os outros modelos, é importante destacar que não ocorreu uma perda substancial de capital.

Ao analisar o desempenho da Floresta Aleatória utilizando dados *Oversampling*, pode-se observar que o comportamento do gráfico do acúmulo de capital foi semelhante ao observado com dados *Undersampling*. No entanto, é notável que o capital final alcançado com dados *Oversampling* foi de 109,877.04, o que representa um aumento significativo em comparação com o uso de dados *Undersampling*. Entretanto na Figura 53, pode-se observar que o gráfico do capital acumulado gerado pelo modelo de Floresta Aleatória com *Oversampling* apresenta uma queda no final do período de teste.

Figura 53 – Trade Floresta Aleatória com parâmetros padrões



Fonte: Autor (2023)

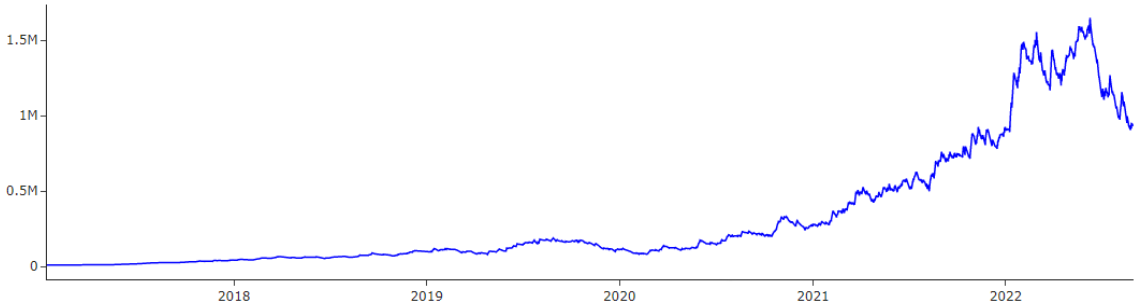
Esse resultado indica que o uso de dados *Oversampling*, que equilibra a distribuição das classes, permitiu que o modelo de Floresta Aleatória capturasse com mais eficiência as tendências e padrões do mercado financeiro, resultando em um maior acúmulo de capital ao final do período de negociação.

Ao avaliar o desempenho do modelo de Rede Neural utilizando dados *Oversampling*, foi observado que o rendimento foi inferior em comparação com o modelo que utilizou dados *Undersampling*. O capital final alcançado pelo modelo de Rede Neural com dados *Oversampling* foi de U$933,711.83, o que representa um valor mais baixo em comparação com o uso de dados *Undersampling* conforme é apresentado na Tabela 8.

Na Tabela 8 demonstra que utilizando dados *OverSampling*, o capital final é maior do que usar dados *Undersampling*. Porém o modelo de Rede Neural não teve prejuízo mas ficou abaixo da capital final Under.

Essa diferença no desempenho pode ser atribuída a várias razões. Pode ser que o modelo de Rede Neural não tenha sido capaz de capturar adequadamente os padrões e tendências presentes nos dados com *Oversampling*. Por isso ajustar os parâmetros

Figura 54 – Trade Rede Neural com parâmetros padrões



Fonte: Autor (2023)

Tabela 7 – Resultados da simulação *trade* de modelos *default Undersampling*

Modelo	Capital inicial	Capital final	Lucro %
SVM	U$10,000.00	U$6,216,854.01	62,068.54%
Árvore de decisão	U$10,000.00	U$1,585.95	-84.14%
Floresta Aleatória	U$10,000.00	U$70,294.12	602.94%
Redes Neurais	U$10,000.00	U$3,896,678.12	38,866.78%

Fonte: Autor (2023)

Tabela 8 – Resultados da simulação *trade* de modelos *default Oversampling*

Modelo	Capital inicial	Capital final	Lucro %
SVM	U$10,000.00	U$7,933,523.59	79,235.23%
Árvore de decisão	U$10,000.00	U$6,277.48	-37.22%
Floresta Aleatória	U$10,000.00	U$109,877.04	998.77%
Redes Neurais	U$10,000.00	U$933,711.83	9,237.11

Fonte: Autor (2023)

do modelo de Rede Neural, melhora o desempenho na simulação *trading*, para isso a importância de utilizar o *GridSearchCV()*.

5.3 Testes com modelos utilizando parâmetros melhorados

Ao executar o código com a função *GridSearchCV()* para realizar uma busca abrangente de parâmetros, o tempo de execução do modelo aumentou consideravelmente devido à necessidade de avaliar várias combinações de parâmetros. Essa abordagem visa encontrar os melhores parâmetros para o modelo, buscando uma maior capacidade de generalização e melhor desempenho.

Conforme observado na Tabela 9, todos os modelos apresentaram pequenas

melhoras em suas métricas ao utilizar dados de *Oversampling* em comparação com os modelos com parâmetros *default*.

Tabela 9 – Resultados da acurácia dos modelos

Testes	Modelo	Over Long	Over Short
1	SVM	50.19%	50.66%
1	Árvore de decisão	61.40%	64.58%
1	Floresta Aleatória	63.41%	65.63%
1	Redes Neurais	59.19%	56.13%
2	SVM	54.21%	53.90%
2	Árvore de decisão	62.35%	64.39%
2	Floresta Aleatória	67.12%	68.02%
2	Redes Neurais	54.71%	54.34%
3	SVM	53.01%	54.02%
3	Árvore de decisão	60.43%	62.13%
3	Floresta Aleatória	67.06%	64.92%
3	Redes Neurais	55.77%	59.23%

Fonte: Autor (2023)

Observa-se que os resultados obtidos com a busca abrangente de parâmetros não apresentaram diferenças significativas em termos de capacidade de generalização quando comparados com os modelo utilizou parâmetros padrões. É possível que os parâmetros padrões já estejam adequados para o conjunto de dados específico, resultando em um desempenho semelhante ao encontrado após a busca abrangente.

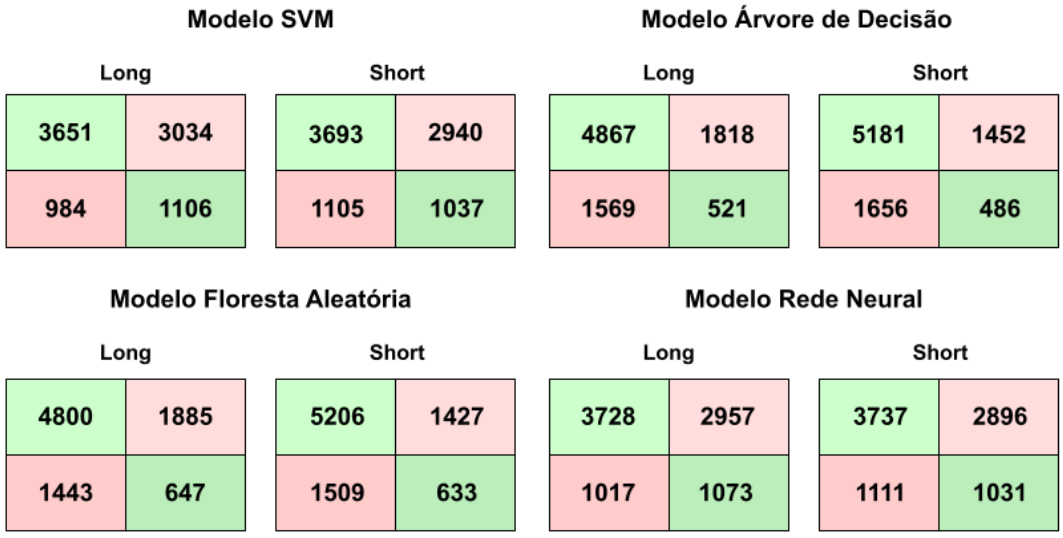
Esses resultados reforçam a importância de considerar técnicas de rebalanceamento de classes, como *Oversampling*, ao treinar modelos de aprendizado de máquina em conjuntos de dados desbalanceados. Essas técnicas podem ajudar a evitar o viés dos modelos em direção à classe majoritária e a melhorar o desempenho geral, permitindo uma tomada de decisão mais eficaz no contexto do mercado financeiro.

A matriz de confusão de modelos com parâmetros melhorados apresentados na Figura 55 foram treinados dados *Oversampling*. Pode ser observado que a matriz ce confusão não mudou muito comparado com a matriz com modelos *default*, somente a classificação

É interessante notar que o aumento na quantidade de verdadeiros positivos e a diminuição dos falsos negativos na matriz de confusão indicam uma melhoria na capacidade dos modelos de identificar corretamente as operações que deveriam ser executadas. Isso resultou em um aumento razoável no lucro durante a simulação.

No entanto, é importante ressaltar que, apesar das melhorias observadas, é

Figura 55 – Matriz de confusão de modelos com parâmetros melhorados



Fonte: Autor (2023)

recomendado realizar uma análise abrangente dos resultados, considerando diferentes métricas e aspectos específicos do problema em questão. Além disso, é importante estar ciente das limitações e das possíveis variações nos resultados ao aplicar diferentes técnicas de rebalanceamento de classes.

Após a conclusão do processo de pré-processamento dos dados nos modelos, o script prossegue para a execução dos resultados, simulações, cálculos finais e visualizações gráficas. A seção a seguir apresenta detalhadamente os resultados obtidos durante a execução dos seis testes selecionados como exemplo para o projeto.

5.3.1 Simulação de *trading* com parâmetros modificados

Com um capital inicial de U$10,000.00 e a compra de 100,000 lotes em cada operação, podemos ter uma ideia melhor do desempenho dos modelos. Após executar as simulações com os parâmetros modificados e treinados com os dados de *Oversampling*, os resultados confirmam que os modelos SVM e Rede Neural continuam apresentando um desempenho superior em relação à cobertura de prejuízo, em comparação com o modelo de Árvore de Decisão e Floresta Aleatória.

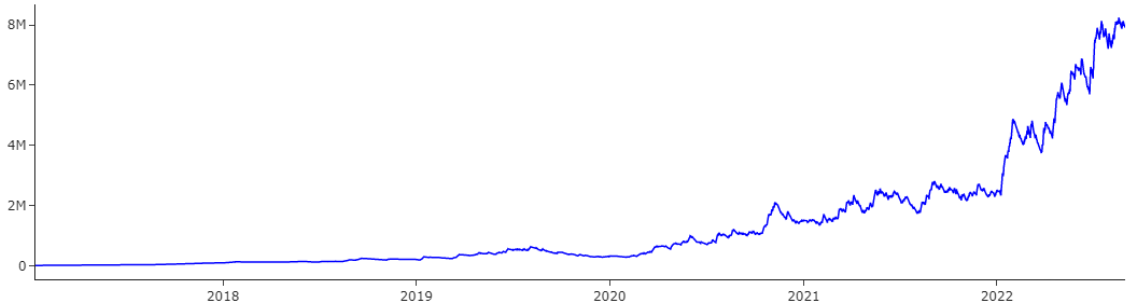
Essa constatação reforça a conclusão de que os modelos SVM e Rede Neural são mais adequados para lidar com a complexidade dos dados financeiros e identificar os

melhores momentos para operar no mercado. Esses modelos possuem maior flexibilidade e capacidade de aprendizado, permitindo capturar relações não lineares e padrões mais sutis nos dados.

Por outro lado, o modelo de Árvore de Decisão continua mostrando limitações na capacidade de generalização e na cobertura de prejuízo. Essa diferença de desempenho pode ser atribuída à natureza mais simplista do modelo de Árvore de Decisão, que pode não ser capaz de capturar as nuances e os padrões complexos dos dados financeiros.

Na Figura 56 é possível observar o gráfico do ganho ao utilizar o modelo SVM com os parâmetros otimizados. O resultado mostra um crescimento consistente do capital ao longo do tempo, culminando em um capital final de 7,933,523.59. Essa melhoria no resultado em comparação com o modelo de parâmetros padrão indica que a busca pelos melhores parâmetros foi eficaz e contribuiu para aumentar o desempenho do modelo.

Figura 56 – Trade SMV com parâmetros modificados



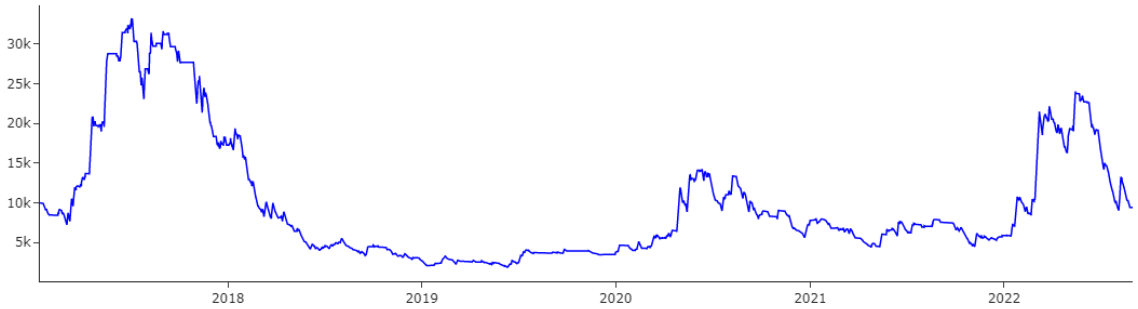
Fonte: Autor (2023)

Na simulação utilizando o modelo de Árvore de Decisão com parâmetros modificados, ainda foi observado um desempenho inferior em comparação com outros modelos. Na análise do gráfico com os parâmetros padrão, houve três picos em que o capital alcançou valores superiores ao inicial conforme é apresentado na Figura 57. No entanto, mesmo com esses picos, o modelo não conseguiu sustentar esses ganhos e, no final, ainda resultou em prejuízo.

Esses resultados sugerem que o modelo de Árvore de Decisão pode não ser a melhor opção para essa tarefa específica de previsão de momentos de negociação. É possível que a natureza mais simplista e linear do modelo de Árvore de Decisão não seja capaz de capturar as complexidades e padrões sutis presentes nos dados financeiros.

O modelo de Floresta Aleatória com parâmetros modificados mostrou uma melhora em relação ao modelo com parâmetros padrão, resultando em um capital final maior. No entanto, é importante observar que o modelo alcançou um pico próximo a

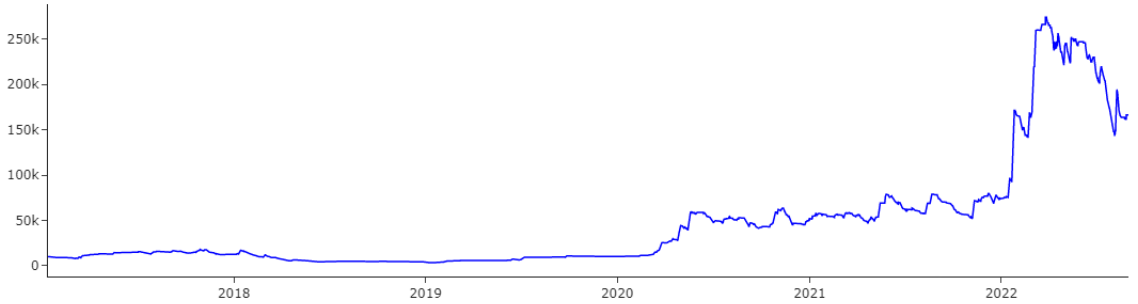
Figura 57 – Trade Árvore de Decisão com parâmetros modificados



Fonte: Autor (2023)

U$250,000.00, mas acabou perdendo rendimento e terminou com um capital final de U$166,426.59 como é observado na Figura 58.

Figura 58 – Trade Floresta Aleatória com parâmetros modificados



Fonte: Autor (2023)

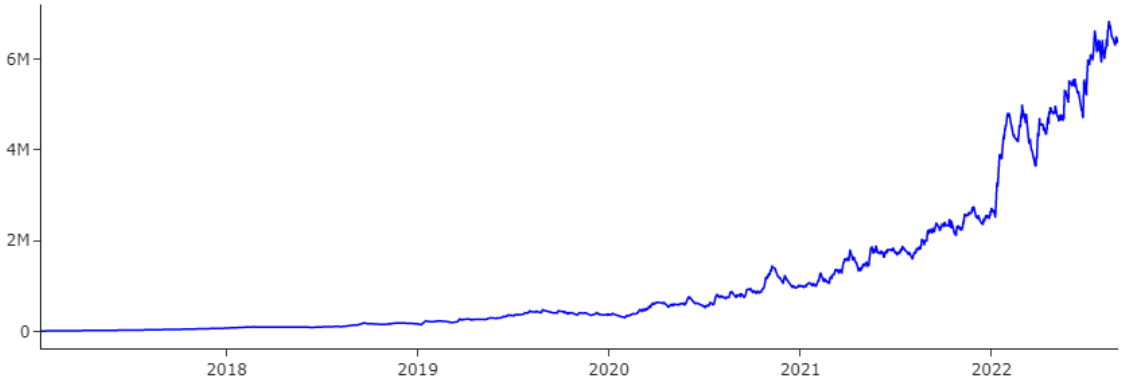
Essa diminuição no rendimento próximo ao final do período pode indicar que o modelo de Floresta Aleatória também pode ter dificuldades em capturar nuances e padrões complexos do mercado.

No caso da Rede Neural, é interessante notar que, apesar de ter obtido lucro, o desempenho não foi superior em comparação com a utilização dos parâmetros padrão e *Oversampling*, como é observado na Figura 59. Isso pode indicar que os parâmetros modificados não contribuíram significativamente para melhorar o desempenho do modelo nesse contexto específico.

Portanto, os resultados obtidos confirmam a recomendação de priorizar os modelos SVM e Rede Neural como opções mais promissoras para a classificação dos melhores momentos de negociação no mercado financeiro, considerando as particularidades e os objetivos específicos do sistema de negociação. Na Tabela 10 é apresentado o capital inicial e o final dos modelos.

Observa-se que o SVM, Redes Neurais e Floresta Aleatória são modelos

Figura 59 – Trade Rede Neural com parâmetros modificados



Fonte: Autor (2023)

Tabela 10 – Resultados da simulação *trade*

Modelo	**Capital inicial**	**Capital final**	**Lucro %**
SVM	U$10,000.00	U$7,933,523.59	79,235.23%
Árvore de decisão	U$10,000.00	U$9,346.82	-6.53%
Floresta Aleatória	U$10,000.00	U$166,426.59	1,564.26%
Redes Neurais	U$10,000.00	U$6,355,064.24	63, 450.64%

Fonte: Autor (2023)

promissores para serem utilizados para negociar no mercado Forex. Enquanto os modelos de Árvore de Decisão não conseguiu se sustentar ao passar do tempo, com capital final abaixo do capital inicial.

Tabela 11 – Comparação de projeto

Modelos	Quantidade de Trades	Período	Lucro %
TradeClassifier (SVM)	4635	5 anos	79,235.23%
(FILHO, 2022) (Combinado)	5	7 anos	1,194.69%

Fonte: Autor (2023)

Ao comparar os resultados obtidos no trabalho de Filho (2022), nota-se que o capital inicial de 500 alcançou um melhor resultado de 6473. As operações foram classificadas utilizando modelos de Árvore de Decisão, Floresta Aleatória e Redes Neurais combinadas, visando otimizar a análise e maximizar os retornos financeiros. O lucro obtido foi de 1,194.69%. No entanto, vale ressaltar que essa conquista foi alcançada com um número reduzido de entradas de operação no período de 2015 a 2022.

O TradeClassifier opera agressivamente analisando todos os *Candles*, buscando operar para atingir o *Take Profit*, por isso se deve do alto lucro acumulado com a técnica SVM conforme pode ser observado na Tabela 11.

6 CONSIDERAÇÕES FINAIS

Esta pesquisa visou detalhar o processo de desenvolvimento de um modelo para auxiliar investidores novatos em um mercado de moedas (Forex) por técnicas de inteligência artificial. Através de pesquisas bibliográficas, análises e leituras em trabalhos correlatos e estudo de manipulação de dados na linguagem Python e suas bibliotecas, foi possível realizar o desenvolvimento total do modelo.

Para se alcançar um patamar de desenvolvimento passível de demonstrar a estrutura e o funcionamento proposto do modelo, a pesquisa foi dividida em três objetivos específicos de estudos e abordagens. O primeiro enfatiza a tecnologia da do mercado financeiro e suas características. O segundo objetivo é voltado para um estudo dos principais conceitos teóricos e abordagens técnicas do mercado financeiro. Enquanto o terceiro objetivo aborda as SMV, árvores de decisão, florestas aleatórias e redes neurais artificiais, técnicas de inteligência artificial em que foram utilizadas durante a pesquisa e o desenvolvimento. Por fim, o quinto objetivo aborda a estrutura e algumas bibliotecas da linguagem Python utilizadas para a análise de dados e utilização de algoritmos de aprendizado de máquina durante o desenvolvimento do trabalho. Com tais estudos e implementações realizadas, podemos concluir que, com base nos resultados obtidos dos testes realizados, o modelo apresenta seu funcionamento correto dentro do esperado, conseguindo gerar uma resposta positiva de aprendizagem para a base de dados em que o processo é executado, além de implementar saídas simuladas satisfatórias.

Sendo assim, foi possível realizar uma implementação funcional de um modelo capaz de auxiliar investidores novatos com um *feedback* de saída simplificado de duas respostas ("operar"e "não opera"), com dois tipos de entrada, *Long* e *Short*, para os dados recebidos através de uma integração em Python que utilizou estudos de padrões de operações das instituições, com dados históricos do ativo EURxUSD.

Do ponto de vista de possíveis perspectivas futuras, o trabalho possui um potencial expansivo extremamente vasto, com a possibilidade de aprimoramentos na lógica de aprendizado dos modelos, nos algoritmos de aprendizado de máquina, nos cálculos lógicos e visualizações gráficas, nos filtros lógicos de classificação, automatização de recebimento dos dados do ativo, podendo aplicar os modelos de aprendizagem em cada momento real e entre diversas outras possibilidades. Com a computação, também é possível adicionar novos conceitos do ICT e combinações com a leitura de tendências, principalmente as estratégias de indicadores. O modelo não é exclusivo do mercado

Forex, os padrões do mercado influenciado pelas instituições ocorre em todos os mercados financeiros, portanto este trabalho pode se estender em outros mercados.

REFERÊNCIAS

AWS. aws, 2023. Disponível em: https://aws.amazon.com/pt/what-is/python/.

BANTON, C. **Smart Money**. Investopedia, 2022. Disponível em: https://www.investopedia.com/terms/s/smart-money.asp.

BEBESHKO, B.; KHOROLSKA, K.; DESIATKO, A. Analysis and modeling of price changes on the exchange market based on structural market data. In: IEEE. **2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T)**. [S.l.], 2021. p. 151–156.

BIANCA, A. **Grafico Candlestick: saiba o que é e como analisar**. Gorila, 2021. Disponível em: https://gorila.com.br/blog/grafico-candlestick/.

BISHOP, C. M. **Pattern Recognition and Machine Learning**. [S.l.]: Springer, 2006.

BODIE, Z.; KANE, A.; MARCUS, A. J. **Investments**. [S.l.]: McGraw-Hill Education, 2014.

BUSS, G.; FERRARI, H. **Inflação do Brasil é a quarta maior do G20**. Poder 360, 2022. Disponível em: https://www.poder360.com.br/economia/inflacao-do-brasil-e-a-4a-maior-do-g20/.

COELHO, F. F. **Machine learning e análise técnica como ferramentas para construção de portfólios de renda variável no mercado brasileiro**. Tese (Doutorado), 2020.

EBERMAN, E. Desenvolvimento de um método híbrido para negociações de ações na bolsa de valores brasileira. Universidade Federal do Espírito Santo, 2018.

ELDER, A. **Como se transformar em um operador e investidor de sucesso**. [S.l.]: Elsevier, 2004.

FILHO, M. H. I. DeterminaÇÃo dos momentos de compra e venda de bitcoins usando tÉcnicas de inteligÊncia artificial. 2022.

FOREX. [S.l.]: Portal do Investidor, 2022. <https://www.investidor.gov.br/menu/Menu_Investidor/Old/Evitando_Fraudes/Forex.html>. Accessed: 2022-04-27.

FOREXLENS. Forexlens, 2022. Disponível em: https://www.forexlens.com/how-smart-money-trading-works/.

GERLEIN, E. A. et al. Evaluating machine learning classification for financial trading: An empirical approach. **Expert Systems with Applications**, Elsevier, v. 54, p. 193–207, 2016.

GÉRON, A. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems**. [S.l.]: O'Reilly Media, 2019.

HAO, L. et al. Deep learning-based survival analysis for high-dimensional survival data. **Mathematics**, v. 9, p. 1244, 05 2021.

INFOMONEY. **O que faz um trader?** 2020. <https://www.infomoney.com.br/guias/trader/>. Accessed: 2022-04-26.

ISMAIL, M. A. H. et al. Automated trading system for forecasting the foreign exchange market using technical analysis indicators and artificial neural network. In: IEEE. **2022 IEEE 18th International Colloquium on Signal Processing & Applications (CSPA)**. [S.l.], 2022. p. 63–68.

JÚNIOR, C. V. Um agrupamento de modelos conexonistas por meio de sinapses artificiais e suas aplicações no mercado de criptomoedas. Centro Universitário FEI, São Bernardo do Campo, 2020.

LEMOS, F. Análise técnica dos mercados financeiros: um guia completo e definitivo dos métodos de negociação de ativos. **São Paulo: Saraiva**, 2018.

MARTINS, C. **Os supersinais da análise técnica: guia para investimentos lucrativos na bolsa**. [S.l.]: Elsevier, 2010.

MASKEY, B. Smart money concepts in the forex market. Centria University of Applied Sciences, 2021.

MUHAMMAD, A. **What are Order Blocks in forex**. Forex Bee, 2022. Disponível em: https://forexbee.co/order-blocks-forex/.

MUNIZ, G. **Linha de Tendência de Alta e Baixa**. Muniz Trader, 2019. Disponível em: https://muniztrader.com/blog/2019/09/17/linha-de-tendencia-de-alta-e-baixa/.

NAWANI, J. et al. A quantitative approach to create a hybrid stock filtering platform. In: IEEE. **2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)**. [S.l.], 2020. p. 1495–1499.

PRODANOV, C. C.; FREITAS, E. C. D. **Metodologia do trabalho científico: métodos e técnicas da pesquisa e do trabalho acadêmico-2ª Edição**. [S.l.]: Editora Feevale, 2013.

RESEARCH, S. **O que é ATR (Average True Range)?** 2023. <https://www.suno.com.br/artigos/atr/>.

RUSSELL, S. J.; NORVIG, P. **Artificial Intelligence: a modern approach**. 3. ed. [S.l.]: Pearson, 2009.

SANTOS, F. **Suporte e Resistência**. Meu Day Trade, 2020. Disponível em: https://meudaytrade.com/suporte-e-resistencia/.

SILVA, T. R. d. et al. Avaliação de um algotrading baseado em deep learning para o mercado de capitais utilizando gerenciamento de risco. Universidade Federal de Itajubá, 2021.

THARWAT, A. Classification assessment methods. **Journal Name**, Volume, n. Number, p. Páginas, 2018. Disponível em: URL.

TORO. Toro Blog, 2023. Disponível em: https://blog.toroinvestimentos.com.br/trading/long-short?utm_source=googleutm_medium=cpcutm_campa

XPE. **O que é indicador MACD e como usá-lo nos seus investimentos?** XP Educação, 2022. Disponível em: https://blog.xpeducacao.com.br/indicador-macd/?gclid=Cj0KCQjw2_OWBhDqARIsAAUNTTH_cB0QkF

APÊNDICE A – CÓDIGO DO TRADECLASSIFIER

```
#Instala os pacotes b sicos
import pandas as pd
try:
    import pandas_ta as ta
except:
    !pip install pandas_ta
    import pandas_ta as ta
import datetime as dt
import math
import matplotlib
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import numpy as np

from IPython.display import display

#Constantes e par metros
DIRETORIO_DADOS = './data/'          #Diret rio dos arquivos de dados (leitura)
DIRETORIO_SAIDA = './output/'        #Diret rio dos arquivos de sa da (escrita)
TICKET = 'EURUSD'                    #Moeda (par)
TIMEFRAME = 240                      #Tempo gr fico
INICIO = '2015-01-01'                #Data inicial
FINAL = '2024-04-01'                 #Data final

ATR_PERIOD = 14                      #Per odo do ATR
EMA_FAST = 50                        #Per odo da EMA r pida
EMA_MEAN = 144                       #Per odo da EMA m dia
EMA_SLOW = 200                       #Per odo da EMA lenta
RSI_PERIOD = 14                      #Per odo do RSI

TAM_STOP = 1                         #Tamanho do stop loss em ATRs
RISCO_RETORNO = 3                    #Rela  o risco-retorno
TOTAL_TAXAS = 0.0001                 #Soma da taxa, spread e slipagge

#Carrega a base de dados, define o nome das colunas, cria o  ndcie  e remove os registros
que n o  est o no per odo solicitado
#Reforna o dataframe com os dados carregados no formato exigido pelas rotinas do sistema
def carregarArquivo(DIRETORIO, TICKET, TIMEFRAME, INICIO, FINAL):
    df_full = pd.read_csv(DIRETORIO+TICKET+str(TIMEFRAME)+'.csv')
    df_full.dropna(axis=0, how='all', inplace=True) #Remove todos os registros com null em algum campo
    df_full.columns=(['Date', 'Time', 'Open', 'High', 'Low', 'Close', 'Volume']) #Seta os cabe alhos dos campos
    df_full = df_full[(df_full['High'] > df_full['Low']) & (df_full['Volume'] > 1)] #Remove candles sem negocia  o
    df_full['Open time'] = pd.to_datetime(df_full['Date'] + ' ' + df_full['Time'] +
    ':00+0300', format='%Y.%m.%d %H:%M:%S%z')
    df_full.drop(df_full.iloc[:, 0:2], axis = 1, inplace=True) #Remove os campos date e time
    df_full.set_index('Open time', inplace = True) #Cria o  ndice
    df = df_full[(df_full.index >= pd.to_datetime(INICIO+
    ' 00:00:00+0300')) & (df_full.index <= pd.to_datetime(FINAL+' 00:00:00+0300'))]
    del[df_full] #Deleta a base full
    return df #Retorna a base com os registros do per odo selecionado
# #end def

#Calcula o ATR desde o primeiro ao  ltimo  registro e insere no dataframe. Recebe o dataframe e adiona fisicamente a coluna 'ATR' nele
#Pressup e que o dataframe est  no formato padr o com as colunas 'Open', 'High', 'Low' e 'Close'
def calcularIndicadorATR(df, period):
    df['ATR'] = ta.atr(open=df['Open'], high=df['High'], low=df['Low'], close=df['Close'], length=period)
    df['ATR'].fillna(ta.true_range(open=df['Open'], high=df['High'], low=df['Low'], close=df['Close']).ewm(span=period,
                adjust=False).mean(), inplace=True) #Preenche os primeiros period registros
    df['ATR'].fillna((df['High']-df['Low']), inplace=True) #Preenche algum residual (em geral o primeiro registro)
    df['ATR'] = round(df['ATR'], 5)
#end def

#Calcula a EMA desde o primeiro ao  ltimo  registro e insere no dataframe. Recebe o dataframe e adiona fisicamente a
coluna 'EMA+period' nele
#Pressup e que o dataframe est  no formato padr o com as colunas 'Open', 'High', 'Low' e 'Close'
def calcularIndicadorEMA(df, period):
    df['EMA'+str(period)] = ta.ema(open=df['Open'], high=df['High'], low=df['Low'], close=df['Close'], length=period)
```

```
    df['EMA'+str(period)].fillna(df['Close'].ewm(span=period, adjust=False).mean(), inplace=True)
    df['EMA'+str(period)] = round(df['EMA'+str(period)], 5)
#end def

#Calcula o oscilador de Bolllinger e insere ele no dataframe. Recebe o dataframe e adiona fisicamente
a coluna
'Boll+period' nele
#Pressup e que o dataframe est  no formato padr o com as colunas 'Open', 'High', 'Low' e 'Close'
def calcularIndicadorBollingerOscilattor(df, period):
    df['Boll'+str(period)] = (df['Close'] - ta.ema(close=df['Close'],length=period)) /
    ta.stdev(close=df['Close'], length=period)
    df['Boll'+str(period)] = round(df['Boll'+str(period)], 5)
#end def

#Cacula a inclina  o (derivada) da EMA e insere ela no dataframe. Recebe o dataframe e
adiona fisicamente a coluna 'Deriv+period' nele
#Pressup e que o dataframe est  no formato padr o e possui a coluna 'EMA+period' nele
def calcularIndicadorROC(df, period):
    df['ROC'+str(period)] = ta.roc(df['EMA'+str(period)], max(1, math.trunc(period/4.236+0.5)))
    df['ROC'+str(period)] = round(df['ROC'+str(period)], 5)
#end def

#Carrega o arquivo e calcula os indicadores
df = carregarArquivo(DIRETORIO_DADOS, TICKET, TIMEFRAME, INICIO, FINAL)
calcularIndicadorATR(df, ATR_PERIOD)
calcularIndicadorEMA(df, EMA_FAST)
calcularIndicadorEMA(df, EMA_MEAN) # Nova parametro
calcularIndicadorEMA(df, EMA_SLOW)

calcularIndicadorBollingerOscilattor(df, EMA_FAST)
calcularIndicadorBollingerOscilattor(df, EMA_MEAN)
calcularIndicadorBollingerOscilattor(df, EMA_SLOW)

calcularIndicadorROC(df, EMA_FAST)
calcularIndicadorROC(df, EMA_MEAN)
calcularIndicadorROC(df, EMA_SLOW)

#Cruzamento de m dias
df['CrxFM'] = (df['EMA'+str(EMA_FAST)] - df['EMA'+str(EMA_MEAN)]) / df['Close']
df['CrxFS'] = (df['EMA'+str(EMA_FAST)] - df['EMA'+str(EMA_SLOW)]) / df['Close']
df['CrxMS'] = (df['EMA'+str(EMA_MEAN)] - df['EMA'+str(EMA_SLOW)]) / df['Close']

#RSI, engolfos bullish (+1), bearish (-1) e FU candles
df['RSI'] = ta.rsi(close=df['Close'], length=RSI_PERIOD)
df['Engolfo'] = np.where(df['Close'] > df['Open'], (np.where(df['Close'] >
df['High'].shift(+1), 1, 0)), (np.where(df['Close'] < df['Low'].shift(+1), -1, 0)))
df['FuCandle'] = np.where(df['High']-df['Low'] < 1.618*df['ATR'], 0, np.where((df['Open']-
df['Low'] > (df['High']-df['Low'])*0.618) & (df['Close']-df['Low'] > (df['High']-
df['Low'])*0.618), +1,
         np.where((df['High']-df['Close'] > (df['High']-df['Low'])*0.618) & (df['High']-
         df['Open'] > (df['High']-df['Low'])*0.618), -1, 0))) #Pavio > 61.8% do tamanho do candle (size) e size > 1.618*ATR

#Remove as primeiras velas onde n o foi poss vel calcular os indicadores
# df.dropna(axis=0, how='any', inplace=True)

#Verifica se o topo mais recente j  foi consolidado
def topoConsolidado(df, extremo, pos_atual):
    for j in range(df.index.get_loc(extremo), pos_atual, 1):
        if df['Close'].iloc[j+1] < df['Low'].iloc[j]:
            return True #Se engolfou retorna true
    return False
#end def


#Verifica se o fundo mais recente j  foi consolidado
def fundoConsolidado(df, extremo, pos_atual):
    for j in range(df.index.get_loc(extremo), pos_atual, 1):
        if df['Close'].iloc[j+1] > df['High'].iloc[j]:
            return True #Se engolfou retorna true
    return False
```

```
#end def

#Vari veis globais usadas daqui em diante
StructTopIdx = StructBottomIdx = df.index[0]

#Cria e inicializa novos campos no dataframe
df['Structure'] = 0
df['StructTop'] = df['High'].loc[StructTopIdx]
df['StructBottom'] = df['Low'].loc[StructBottomIdx]

#Listas e vari veis locais
lista_swings = []
lista_rompimentos = []
lista_demands = []
lista_supplys = []
structure = 0
steadyTop = steadyBottom = False

#Agora faz toda a m gica
for i in range(1, len(df)):
    #Verifica se os topos e fundos foram consolidados
    if not steadyTop:
        steadyTop = topoConsolidado(df, StructTopIdx, i)
    if not steadyBottom:
        steadyBottom = fundoConsolidado(df, StructBottomIdx, i)

    #Verifica se rompeu o topo
    if df['High'].iloc[i] > df['High'].loc[StructTopIdx]: #Violou o extremo
        if steadyTop: #Se o extremo  j   est  consolidado
            if df['Close'].iloc[i] > df['High'].loc[StructTopIdx]: #Fechou acima do extremo
                #Salva o rompimento
                lista_rompimentos.append([StructTopIdx, df.index[i], df['High'].loc[StructTopIdx]])
                #Atualiza o fundo (forte)
                StructBottomIdx = df['Low'].iloc[df.index.get_loc(StructTopIdx)+1:i+1].idxmin()
                steadyBottom = True #O fundo    consolidado no rompimento do topo
                lista_swings.append([StructBottomIdx, df['Low'].loc[StructBottomIdx]])
                lista_demands.append([StructBottomIdx, df['Low'].loc[StructBottomIdx], df.index[i+1], df['High'].loc[StructBottomIdx]])
                #Guarda o novo topo, marca ele como  n  o  consolidado e registra a estrutura como bullish
                StructTopIdx = df.index[i]
                steadyTop = False
                structure = 1 #Bullish
        else:
            StructTopIdx = df.index[i] #Extremo  n  o  consolidado - atualiza mas  n  o  registra rompimento

    #Verifica se rompeu o fundo
    if df['Low'].iloc[i] < df['Low'].loc[StructBottomIdx]: #Violou o extremo
        if steadyBottom: #Se o extremo  j   est  consolidado
            if df['Close'].iloc[i] < df['Low'].loc[StructBottomIdx]: #Fechou abaixo do extremo
                #Salva o rompimento
                lista_rompimentos.append([StructBottomIdx, df.index[i], df['Low'].loc[StructBottomIdx]])
                #Atualiza o topo (forte)
                StructTopIdx = df['High'].iloc[df.index.get_loc(StructBottomIdx)+1:i+1].idxmax()
                steadyTop = True #O topo forte    consolidado no rompimento do fundo
                lista_swings.append([StructTopIdx, df['High'].loc[StructTopIdx]])
                lista_supplys.append([StructTopIdx, df['Low'].loc[StructTopIdx], df.index[i+1], df['High'].loc[StructTopIdx]])
                #Guarda o novo fundo, marca ele como  n  o  consolidado e registra a estrutura como bearish
                StructBottomIdx = df.index[i]
                steadyBottom = False
                structure = -1 #Bearsih
        else:
            StructBottomIdx = df.index[i] #Extremo  n  o  consolidado - atualiza mas  n  o  registra rompimento

    #Salva a estrutura corrente no dataframe
    df.at[df.index[i], 'Structure'] = structure #Se estamos bullish ou bearish
    df.at[df.index[i], 'StructTop'] = df['High'].loc[StructTopIdx] #Topo atual da estrutra
    df.at[df.index[i], 'StructBottom'] = df['Low'].loc[StructBottomIdx] #Fundo atual da estrutra
#end for

#Premium ou discount? Guardamos em valores entre 0 e 1
df['StructPD'] = round((df['Close']-df['StructBottom']) / (df['StructTop'] - df['StructBottom']), 6)
```

```
#Agora criamos as listas de topos e fundos e supplys e demands
df_swings = pd.DataFrame(lista_swings, columns=['Time', 'Value'])
df_rompimentos = pd.DataFrame(lista_rompimentos, columns=['x0', 'x1', 'y'])
df_demands = pd.DataFrame(lista_demands, columns=['x0', 'y0', 'x1', 'y1'])
df_supplys = pd.DataFrame(lista_supplys, columns=['x0', 'y0', 'x1', 'y1'])

#Apaga as listas e vari veis locais
del[lista_swings, lista_rompimentos, lista_demands, lista_supplys]

Adiciona o campo VelaBull para simplificar os c lculos futuros (true se o candle    de alta)
df['VelaBull'] = df['Close'] > df['Open']

#C lculo dos imbalances
df['ImbLow'] = 0.0
df['ImbHigh'] = 0.0
for i in range(1, len(df)-1):
    if df['VelaBull'].iloc[i]: #Se o candle    bullish
        if df['High'].iloc[i-1] < df['Low'].iloc[i+1]:
            df.at[df.index[i], 'ImbLow'] = df['High'].iloc[i-1]
            df.at[df.index[i], 'ImbHigh'] = df['Low'].iloc[i+1]
    else: #Se o candle    bearish
        if df['Low'].iloc[i-1] > df['High'].iloc[i+1]:
            df.at[df.index[i], 'ImbLow'] = df['High'].iloc[i+1]
            df.at[df.index[i], 'ImbHigh'] = df['Low'].iloc[i-1]
#end for

df.dropna(axis=0, how='any', inplace=True)
df.to_csv('./output/Dataframe_2.csv')
```

APÊNDICE B – CÓDIGO DA CLASSIFICAÇÃO DO TRADECLASSIFIER

```
import pandas as pd
from IPython.display import display

dados = pd.read_csv('./output/Dataframe_2.csv')
dados.set_index('Open time', inplace = True) # Data time como index
dados

dados['MMS'] = dados['EMA50'] > dados['EMA200'] # Se EMA200 est  abaixo de EMA50
display(dados)

TAM_STOP = 1                        #Tamanho do stop loss em ATRs
RISCO_RETORNO = 3                   #Rela  o risco-retorno
TOTAL_TAXAS = 0.0001                #Soma da taxa, spread e slipagge


#Gera a saida desejada para treinar a rede neural
df['Long'] = 0
df['Short'] = 0

#Percorre o arquivo vendo em quais velas um long teria dado lucro
for i in range(0, len(df)):
    stop = df['Low'].iloc[i] - TAM_STOP * df['ATR'].iloc[i]
    size_stop = df['Close'].iloc[i] - stop + TOTAL_TAXAS
    take = df['Close'].iloc[i] + size_stop * RISCO_RETORNO + TOTAL_TAXAS
    for j in range(i+1, len(df)):
        if df['Low'].iloc[j] <= stop:
            df.at[df.index[i], 'Long'] = 0
            break
        elif df['High'].iloc[j] > take:
            df.at[df.index[i], 'Long'] = 1
            break

#Percorre o arquivo vendo em quais velas um short teria dado lucro
for i in range(0, len(df)):
    stop = df['High'].iloc[i] + TAM_STOP * df['ATR'].iloc[i]
    size_stop = stop - df['Close'].iloc[i] + TOTAL_TAXAS
    take = df['Close'].iloc[i] - size_stop * RISCO_RETORNO - TOTAL_TAXAS
    for j in range(i+1, len(df)):
        if df['High'].iloc[j] >= stop:
            df.at[df.index[i], 'Short'] = 0
            break
        elif df['Low'].iloc[j] < take:
            df.at[df.index[i], 'Short'] = 1
            break

df.to_csv('./output/Dados_Final.csv')
```

APÊNDICE C – CÓDIGO DOS MODELOS ML DO TRADECLASSIFIER

```
import pandas as pd
from IPython.display import display

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import numpy as np

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn import neural_network
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix

from sklearn.metrics import accuracy_score

hiper = False

dados = pd.read_csv('./output/Dados_Final.csv')
dados.set_index('Open time', inplace = True) # Data time como index
dados

x = dados.iloc[:, :-2]
y = dados.iloc[:, -2] #Long
z = dados.iloc[:, -1] #Short

# Visualiza o da propor o
print('----------- PROPOR O DO LONG ----------')
display(y.value_counts())
print('----------- PROPOR O DO SHORT ----------')
display(z.value_counts())

y.value_counts().plot.pie(autopct='%.2f')

z.value_counts().plot.pie(autopct='%.2f')

treino_x, teste_x, treino_y, teste_y, treino_z, teste_z =
train_test_split(x, y, z, test_size = 0.25, shuffle = False, random_state=42)

dados_open = teste_x['Open'].copy()
treino_x.drop(columns='Open' ,inplace=True)
teste_x.drop(columns='Open' ,inplace=True)

dados_close = teste_x['Close'].copy()
treino_x.drop(columns='Close' ,inplace=True)
teste_x.drop(columns='Close' ,inplace=True)

dados_low = teste_x['Low'].copy()
treino_x.drop(columns='Low' ,inplace=True)
teste_x.drop(columns='Low' ,inplace=True)

dados_high = teste_x['High'].copy()
treino_x.drop(columns='High' ,inplace=True)
teste_x.drop(columns='High' ,inplace=True)

dados_atr = teste_x['ATR'].copy()
treino_x.drop(columns='ATR' ,inplace=True)
teste_x.drop(columns='ATR' ,inplace=True)

display(treino_x)
display(teste_x)
print('Treino de y')
display(treino_y.value_counts())
print('Teste de y')
display(teste_y.value_counts())

print('Treino de z')
```

```
display(treino_z.value_counts())
print('Teste de z')
display(teste_z.value_counts())

treino_y.value_counts().plot.pie(autopct='%.2f')

teste_y.value_counts().plot.pie(autopct='%.2f')

treino_z.value_counts().plot.pie(autopct='%.2f')

teste_z.value_counts().plot.pie(autopct='%.2f')

#Balanceamento dos dados
from imblearn.over_sampling import RandomOverSampler

ros = RandomOverSampler(random_state=42)

X_resROS, y_resROS = ros.fit_resample(treino_x, treino_y)
X_resROS_s, z_resROS_s = ros.fit_resample(treino_x, treino_z)

y_resROS.value_counts().plot.pie(autopct='%.2f')

if(hiper == True):
    # Define the parameter grid
    param_grid = {
        'C': [0.1, 1, 10],
        'kernel': ['linear', 'rbf', 'sigmoid'],
        'gamma': [0.1, 0.5, 1]
    }
    svm_model = GridSearchCV(SVC(), param_grid,verbose = 2).fit(X_resROS,y_resROS)
    print('SVM: ', svm_model.best_params_)
    svm_model = SVC(C = svm_model.best_estimator_.C, kernel = svm_model.best_estimator_.kernel,
    gamma = svm_model.best_estimator_.gamma)

elif(hiper == False):
    svm_model = SVC()


if(hiper == True):
    # Define the parameter grid
    param_grid_s = {
        'C': [0.1, 1, 10],
        'kernel': ['linear', 'rbf', 'sigmoid'],
        'gamma': [0.1, 0.5, 1]
    }

    svm_model_s = GridSearchCV(SVC(), param_grid_s,verbose = 2).fit(X_resROS,y_resROS)
    print('SVM: ', svm_model_s.best_params_)
    svm_model_s = SVC(C = svm_model_s.best_estimator_.C, kernel =
    svm_model_s.best_estimator_.kernel, gamma = svm_model_s.best_estimator_.gamma)


elif(hiper == False):
    svm_model_s = SVC()

modelo_SVM_l = svm_model
modelo_SVM_s = svm_model_s

modelo_SVM_l.fit(X_resROS, y_resROS)
modelo_SVM_s.fit(X_resROS_s, z_resROS_s)

previsoes_SVM_l = modelo_SVM_l.predict(teste_x)
previsoes_SVM_s = modelo_SVM_s.predict(teste_x)

acuracia_SVM_l = accuracy_score(teste_y, previsoes_SVM_l) * 100 # taxa de acerto
acuracia_SVM_s = accuracy_score(teste_z, previsoes_SVM_s) * 100 # taxa de acerto

print("A acur cia do Long foi %.2f%%" % acuracia_SVM_l)
print("A acur cia do Short foi %.2f%%" % acuracia_SVM_s)
```

```
from sklearn.metrics import confusion_matrix
confusion_matrix(teste_y, previsoes_SVM_l)

confusion_matrix(teste_z, previsoes_SVM_s)

from sklearn.metrics import precision_score
precision_score(teste_y, previsoes_SVM_l)

dados_SVM = teste_x.copy()
dados_SVM['Long'] = previsoes_SVM_l
dados_SVM['Short'] = previsoes_SVM_s

display(dados_SVM)
display(dados_SVM['Long'].value_counts())
display(dados_SVM['Short'].value_counts())

if(hiper == True):
    parametros_arvore = {
        'criterion': ['entropy', 'gini'],
        'splitter': ['best', 'random'],
        'max_depth': [5, 10, 15, 20, 25, 30],
        'random_state': [10, 20, 30, 40, 50]
    }
    arvore = GridSearchCV(DecisionTreeClassifier(), parametros_arvore,verbose = 2).fit(X_resROS,y_resROS)
    print('Arvore: ', arvore.best_params_)
    arvore = DecisionTreeClassifier(criterion = arvore.best_estimator_.criterion, max_depth
    = arvore.best_estimator_.max_depth, random_state = arvore.best_estimator_.random_state)

elif(hiper == False):
    arvore = DecisionTreeClassifier()

if(hiper == True):
    parametros_arvore_s = {
        'criterion': ['entropy', 'gini'],
        'splitter': ['best', 'random'],
        'max_depth': [5, 10, 15, 20, 25, 30],
        'random_state': [10, 20, 30, 40, 50]
    }
    arvore_s = GridSearchCV(DecisionTreeClassifier(), parametros_arvore_s,verbose = 2).fit(X_resROS_s,z_resROS_s)
    print('Arvore: ', arvore_s.best_params_)
    arvore_s = DecisionTreeClassifier(criterion = arvore_s.best_estimator_.criterion,
    max_depth = arvore_s.best_estimator_.max_depth, random_state = arvore_s.best_estimator_.random_state)

elif(hiper == False):
    arvore_s = DecisionTreeClassifier()

modelo_DT_l = arvore
modelo_DT_s = arvore_s

modelo_DT_l.fit(X_resROS, y_resROS)
modelo_DT_s.fit(X_resROS_s, z_resROS_s)

previsoes_DT_l = modelo_DT_l.predict(teste_x)
previsoes_DT_s = modelo_DT_s.predict(teste_x)

acuracia_DT_l = accuracy_score(teste_y, previsoes_DT_l) * 100 # taxa de acerto
acuracia_DT_s = accuracy_score(teste_z, previsoes_DT_s) * 100 # taxa de acerto

print("A acur cia do Long foi %.2f%%" % acuracia_DT_l)
print("A acur cia do Short foi %.2f%%" % acuracia_DT_s)

dados_DT = teste_x.copy()
dados_DT['Long'] = previsoes_DT_l
dados_DT['Short'] = previsoes_DT_s

display(dados_DT)
display(dados_DT['Long'].value_counts())
display(dados_DT['Short'].value_counts())
```

```
if(hiper == True):
        parametros_floresta = {
                'n_estimators': [50, 100, 150, 200],
                'criterion': ['entropy', 'gini'],
                'max_depth': [5, 10, 15]
        }
        floresta = GridSearchCV(RandomForestClassifier(), parametros_floresta,verbose = 2).fit(X_resROS,y_resROS)
        print('Floresta: ', floresta.best_params_)
        floresta = RandomForestClassifier(criterion = floresta.best_estimator_.criterion,
        max_depth = floresta.best_estimator_.max_depth, n_estimators =
        floresta.best_estimator_.n_estimators, random_state=floresta.best_estimator_.random_state)

elif(hiper == False):
        floresta = RandomForestClassifier()

if(hiper == True):
        parametros_floresta_s = {
                'n_estimators': [50, 100, 150, 200],
                'criterion': ['entropy', 'gini'],
                'max_depth': [5, 10, 15]
        }
        floresta_s = GridSearchCV(RandomForestClassifier(), parametros_floresta_s,verbose = 2).fit(X_resROS_s,
        z_resROS_s)
        print('Floresta: ', floresta_s.best_params_)
        floresta_s = RandomForestClassifier(criterion =
        floresta_s.best_estimator_.criterion, max_depth =
        floresta_s.best_estimator_.max_depth, n_estimators =
        floresta_s.best_estimator_.n_estimators, random_state=floresta_s.best_estimator_.random_state)

elif(hiper == False):
        floresta_s = RandomForestClassifier()

modelo_RF_l = floresta
modelo_RF_s = floresta_s

modelo_RF_l.fit(X_resROS, y_resROS)
modelo_RF_s.fit(X_resROS_s, z_resROS_s)

previsoes_RF_l = modelo_RF_l.predict(teste_x)
previsoes_RF_s = modelo_RF_s.predict(teste_x)

acuracia_RF_l = accuracy_score(teste_y, previsoes_RF_l) * 100 # taxa de acerto
acuracia_RF_s = accuracy_score(teste_z, previsoes_RF_s) * 100 # taxa de acerto

dados_RF = teste_x.copy()
dados_RF['Long'] = previsoes_RF_l
dados_RF['Short'] = previsoes_RF_s

display(dados_RF)
display(dados_RF['Long'].value_counts())
display(dados_RF['Short'].value_counts())

if(hiper == True):
        parametros_rede = {
                'max_iter': [500, 1000, 5000, 10000],
                'hidden_layer_sizes': np.arange(10, 15),
                'alpha': [0.0001, 0.001, 0.01, 0.1],
                'learning_rate': ['constant', 'adaptative'],
                'momentum': [0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.0]
        }
        rede = GridSearchCV(neural_network.MLPClassifier(max_iter=500), parametros_rede,verbose = 2).fit(X_resROS,
        y_resROS)
        print('Rede: ', rede.best_params_)
        rede = neural_network.MLPClassifier(max_iter = rede.best_estimator_.max_iter,
        hidden_layer_sizes = rede.best_estimator_.hidden_layer_sizes, alpha =
        rede.best_estimator_.alpha, learning_rate = rede.best_estimator_.learning_rate,
        momentum = rede.best_estimator_.momentum, random_state=rede.best_estimator_.random_state)
elif(hiper == False):
        rede = MLPClassifier()
```

```
if(hiper == True):
        parametros_rede = {
                'max_iter': [500, 1000, 5000, 10000],
                'hidden_layer_sizes': np.arange(10, 15),
                'alpha': [0.0001, 0.001, 0.01, 0.1],
                'learning_rate': ['constant', 'adaptative'],
                'momentum': [0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.0]
        }
        rede_s = GridSearchCV(neural_network.MLPClassifier(max_iter=500), parametros_rede,verbose = 2).fit(X_resROS_s,
        z_resROS_s)
        print('Rede: ', rede_s.best_params_)
        rede_s = neural_network.MLPClassifier(max_iter = rede_s.best_estimator_.max_iter,
        hidden_layer_sizes = rede_s.best_estimator_.hidden_layer_sizes, alpha =
        rede_s.best_estimator_.alpha, learning_rate = rede_s.best_estimator_.learning_rate,
        momentum = rede_s.best_estimator_.momentum,
        random_state=rede_s.best_estimator_.random_state)
elif(hiper == False):
        rede_s = MLPClassifier()

modelo_RN_l = rede #0.001
#max_iter=numero_de_epocas,verbose=False,momentum=momento,learning_rate='constant',learning_
rate_init=taxa_de_aprendizado
modelo_RN_s = rede_s#rede 'alpha': 0.1, 'hidden_layer_sizes': 13, 'learning_rate':
'constant', 'max_iter': 5000, 'momentum': 0.6

# modelo_RN_l = MLPClassifier(max_iter=500)

modelo_RN_l.fit(X_resROS,y_resROS.values.ravel());
modelo_RN_s.fit(X_resROS_s,z_resROS_s.values.ravel());

previsoes_RN_l = modelo_RN_l.predict(teste_x)
previsoes_RN_s = modelo_RN_s.predict(teste_x)

acuracia_RN_l = accuracy_score(teste_y, previsoes_RN_l) * 100 # taxa de acerto
acuracia_RN_s = accuracy_score(teste_z, previsoes_RN_s) * 100 # taxa de acerto

dados_RN = teste_x.copy()
dados_RN['Long'] = previsoes_RN_l
dados_RN['Short'] = previsoes_RN_s

display(dados_RN)
display(dados_RN['Long'].value_counts())

print('-----MODELO SVM-----')
print("A acur cia do Long foi %.2f%%" % acuracia_SVM_l)
print("A acur cia do Short foi %.2f%%" % acuracia_SVM_s)
print('Matrix de Confus o do Long:')
print(confusion_matrix(teste_y, previsoes_SVM_l))
print('Precis o do Long')
print(precision_score(teste_y, previsoes_SVM_l))
print('Matrix de Confus o do Long:')
print(confusion_matrix(teste_z, previsoes_SVM_s))
print('Precis o do Short')
print(precision_score(teste_z, previsoes_SVM_s))
print('==================================')
print('-----MODELO  RVORE  DE DECIS O-----')
print("A acur cia do Long foi %.2f%%" % acuracia_DT_l)
print("A acur cia do Short foi %.2f%%" % acuracia_DT_s)
print('Matrix de Confus o do Long:')
print(confusion_matrix(teste_y, previsoes_DT_l))
print('Precis o do Long')
print(precision_score(teste_y, previsoes_DT_l))
print('Matrix de Confus o do Short:')
print(confusion_matrix(teste_z, previsoes_DT_s))
print('Precis o do Short')
print(precision_score(teste_z, previsoes_DT_s))
print('==================================')
print('-----MODELO FLORESTA ALEAT RIA-----')
print("A acur cia do Long foi %.2f%%" % acuracia_RF_l)
```

```
print("A acur cia do Short foi %.2f%%" % acuracia_RF_s)
print('Matrix de Confus o do Long:')
print(confusion_matrix(teste_y, previsoes_RF_l))
print('Precis o do Long')
print(precision_score(teste_y, previsoes_RF_l))
print('Matrix de Confus o do Short:')
print(confusion_matrix(teste_z, previsoes_RF_s))
print('Precis o do Short')
print(precision_score(teste_z, previsoes_RF_s))
print('================================')
print('-----MODELO REDE NEURAL-----')
print("A acur cia do Long foi %.2f%%" % acuracia_RN_l)
print("A acur cia do Short foi %.2f%%" % acuracia_RN_s)
print('Matrix de Confus o do Long:')
print(confusion_matrix(teste_y, previsoes_RN_l))
print('Precis o do Long')
print(precision_score(teste_y, previsoes_RN_l))
print('Matrix de Confus o do Short:')
print(confusion_matrix(teste_z, previsoes_RN_s))
print('Precis o do Short')
print(precision_score(teste_z, previsoes_RN_s))
print('================================')


if(hiper == False):
    # Dataframe SVM
    dados_SVM['Open'] = dados_open
    dados_SVM['Close'] = dados_close
    dados_SVM['Low'] = dados_low
    dados_SVM['High'] = dados_high
    dados_SVM['ATR'] = dados_atr
    dados_SVM.to_csv('./output/prev/Over_P/Dados_Prev_SVM.csv')

    # Dataframe Decision Tree
    dados_DT['Open'] = dados_open
    dados_DT['Close'] = dados_close
    dados_DT['Low'] = dados_low
    dados_DT['High'] = dados_high
    dados_DT['ATR'] = dados_atr
    dados_DT.to_csv('./output/prev/Over_P/Dados_Prev_DT.csv')

    # Dataframe Random Forest
    dados_RF['Open'] = dados_open
    dados_RF['Close'] = dados_close
    dados_RF['Low'] = dados_low
    dados_RF['High'] = dados_high
    dados_RF['ATR'] = dados_atr
    dados_RF.to_csv('./output/prev/Over_P/Dados_Prev_RF.csv')

    # Dataframe Neurakl Network
    dados_RN['Open'] = dados_open
    dados_RN['Close'] = dados_close
    dados_RN['Low'] = dados_low
    dados_RN['High'] = dados_high
    dados_RN['ATR'] = dados_atr
    dados_RN.to_csv('./output/prev/Over_P/Dados_Prev_RN.csv')
elif(hiper == True):
    # Dataframe SVM
    dados_SVM['Open'] = dados_open
    dados_SVM['Close'] = dados_close
    dados_SVM['Low'] = dados_low
    dados_SVM['High'] = dados_high
    dados_SVM['ATR'] = dados_atr
    dados_SVM.to_csv('./output/prev/Over_H/Dados_Prev_SVM.csv')

    # Dataframe Decision Tree
    dados_DT['Open'] = dados_open
    dados_DT['Close'] = dados_close
    dados_DT['Low'] = dados_low
    dados_DT['High'] = dados_high
```

```
dados_DT['ATR'] = dados_atr
dados_DT.to_csv('./output/prev/Over_H/Dados_Prev_DT.csv')

# Dataframe Random Forest
dados_RF['Open'] = dados_open
dados_RF['Close'] = dados_close
dados_RF['Low'] = dados_low
dados_RF['High'] = dados_high
dados_RF['ATR'] = dados_atr
dados_RF.to_csv('./output/prev/Over_H/Dados_Prev_RF.csv')

# Dataframe Neurakl Network
dados_RN['Open'] = dados_open
dados_RN['Close'] = dados_close
dados_RN['Low'] = dados_low
dados_RN['High'] = dados_high
dados_RN['ATR'] = dados_atr
dados_RN.to_csv('./output/prev/Over_H/Dados_Prev_RN.csv')
```

APÊNDICE D – CÓDIGO *TRADE* DO TRADECLASSIFIER

```
    import pandas as pd
from IPython.display import display

df_SVM = pd.read_csv('./output/prev/Under/Dados_Prev_SVM.csv')
df_DT = pd.read_csv('./output/prev/Under/Dados_Prev_DT.csv')
df_RF = pd.read_csv('./output/prev/Under/Dados_Prev_RF.csv')
df_RN = pd.read_csv('./output/prev/Under/Dados_Prev_RN.csv')

df_SVM.set_index('Open time', inplace = True) # Data time como index
df_DT.set_index('Open time', inplace = True) # Data time como index
df_RF.set_index('Open time', inplace = True) # Data time como index
df_RN.set_index('Open time', inplace = True) # Data time como index

display(df_SVM)
display(df_DT)
display(df_RF)
display(df_RN)


TAM_STOP =              1               #Tamanho do stop loss em ATRs
RISCO_RETORNO =         3               #Rela  o risco-retorno
TAMANHO_LOTE =     100000               #Tamanho do lote padr o
CAPITAL_INICIAL =   10000               #Capital inicial em USD
RISCO_POR_TRADE =   1/100               #Risco por trade em percentual/100
LOTE_MINIMO =        0.01               #Lote m nimo da corretora
LOTE_MAXIMO =         100               #Lote m ximo da corretora
FEE =                3.50               #Taxa (one-way) da corretora para um lote em USD
SPREAD =           0.1e-4               #Spread t pico em USD (pips/10000); default: 0.1e-4 (0.1 pips)
SLIPPAGE =         0.2e-4               #Slippage t pico em USD (pips/10000); default: 0.2e-4 (0.2 pips)
TOTAL_FEES = SPREAD + SLIPPAGE + (2*FEE)/TAMANHO_LOTE #Taxas totais da corretora

#Realiza o trade simulado e retorna o lucro que seria obtido, bem como o ponto onde o trade seria encerrado
def realizarLongTrade(df, timeEntry, money):

    #Calcula a posicao do stop, o 'custo' dele (tamanho em pips levando em conta as taxas)
    priceEntry = df['Close'].iloc[timeEntry]
    originalStop = round(df['Low'].iloc[timeEntry] - TAM_STOP * df['ATR'].iloc[timeEntry], 5)
    custoStop = round(abs(priceEntry - originalStop) + TOTAL_FEES, 5)

    #Calcula o tamanho da posicao e o take
    size = round(min(max(money/(custoStop*TAMANHO_LOTE), LOTE_MINIMO), LOTE_MAXIMO), 2)
    originalTake = round(df['Close'].iloc[timeEntry] + custoStop * RISCO_RETORNO + TOTAL_FEES, 5)

    #Avan a no futuro ajustando o stop (trailing) pelo ATR e verifica quando fechar o trade
    priceExit = priceEntry
    timeExit = timeEntry
    currentStop = originalStop
    for j in range(timeEntry+1, len(df)):
        if df['Low'].iloc[j] <= currentStop: #Fecha o trade se foi estopado
            priceExit = currentStop
            timeExit = j
            break
        elif df['High'].iloc[j] > originalTake: #Realiza o lucro e fecha o trade se atingiu o take
            priceExit = originalTake
            timeExit = j
            break
        else: #Ajusta o stop se poss vel levando em conta o fundo da vela atual
            newStop = round(df['Low'].iloc[j] - TAM_STOP * df['ATR'].iloc[j], 5)
            if currentStop < newStop:
                currentStop = newStop

    #Calcula o lucro do trade
    if timeExit > timeEntry:
        profit = round(priceExit*size*TAMANHO_LOTE - (priceEntry+TOTAL_FEES)*size*TAMANHO_LOTE, 2)
    else:
        profit = 0
```

```
    #Retorna uma lista com todos os parametros de plotagem
    return profit#[timeEntry, timeExit, priceEntry, originalStop, originalTake, currentStop, priceExit, custoStop, size, profit]


#Realiza o trade simulado e retorna o lucro que seria obtido, bem como o ponto onde o trade seria encerrado
def realizarShortTrade(df, timeEntry, money):

    #Calcula a posicao do stop, o 'custo' dele (tamanho em pips levando em conta as taxas)
    priceEntry = df['Close'].iloc[timeEntry]
    originalStop = round(df['High'].iloc[timeEntry] + TAM_STOP * df['ATR'].iloc[timeEntry], 5)
    custoStop = round(abs(priceEntry - originalStop) + TOTAL_FEES, 5)

    #Calcula o tamanho da posicao e o take
    size = round(min(max(money/(custoStop*TAMANHO_LOTE), LOTE_MINIMO), LOTE_MAXIMO), 2)
    originalTake = round(df['Close'].iloc[timeEntry] - custoStop * RISCO_RETORNO - TOTAL_FEES, 5)

    #Avan a no futuro ajustando o stop (trailing) pelo ATR e verifica quando fechar o trade
    priceExit = priceEntry
    timeExit = timeEntry
    currentStop = originalStop
    for j in range(timeEntry+1, len(df)):
        if df['High'].iloc[j] >= currentStop: #Fecha o trade se foi estopado
            priceExit = currentStop
            timeExit = j
            break
        elif df['Low'].iloc[j] < originalTake: #Realiza o lucro e fecha o trade se atingiu o take
            priceExit = originalTake
            timeExit = j
            break
        else: #Ajusta o stop se poss vel levando em conta o fundo da vela atual
            newStop = round(df['High'].iloc[j] + TAM_STOP * df['ATR'].iloc[j], 5)
            if currentStop > newStop:
                currentStop = newStop

    #Calcula o lucro do trade
    if timeExit > timeEntry:
        profit = round(priceEntry*size*TAMANHO_LOTE - (priceExit+TOTAL_FEES)*size*TAMANHO_LOTE, 2)
    else:
        profit = 0

    #Retorna uma lista com todos os parametros de plotagem
    return profit#[timeEntry, timeExit, priceEntry, originalStop, originalTake, currentStop, priceExit, custoStop, size, profit]

def realizarTodosTrades(df):
    granaAtual = CAPITAL_INICIAL
    df['Equity'] = CAPITAL_INICIAL
    df['Profit'] = 0

    #Geramos todos os trades
    for i in range(0, len(df)):
        if df['Long'].iloc[i] == 1:
            profit = realizarLongTrade(df, i, round(granaAtual*RISCO_POR_TRADE, 2))
            df.at[df.index[i], 'Profit'] = 1 if profit > 0 else 0
            granaAtual += profit
            if (granaAtual < 0):
                break
        elif df['Short'].iloc[i] == 1:
            profit = realizarShortTrade(df, i, round(granaAtual*RISCO_POR_TRADE, 2))
            df.at[df.index[i], 'Profit'] = 1 if profit > 0 else 0
            granaAtual += profit
            if (granaAtual < 0):
                break
        df.at[df.index[i], 'Equity'] = granaAtual
    return round(granaAtual, 2)

profit_SVM = realizarTodosTrades(df_SVM)
profit_DT = realizarTodosTrades(df_DT)
profit_RF = realizarTodosTrades(df_RF)
profit_RN = realizarTodosTrades(df_RN)
```

```
print('-----UTILIZANDO DADOS UNDER-----')
print(f'Profit do modelo SVM: {profit_SVM}')
print(f'Profit do modelo  rvore  de Decis o: {profit_DT}')
print(f'Profit do modelo Floresta Aleat ria: {profit_RF}')
print(f'Profit do modelo Rede Neural: {profit_RN}')

TICKET = 'EURUSD'                   #Moeda (par)
TIMEFRAME = 240                     #Tempo gr fico
INICIO = '1999-01-01'               #Data inicial
FINAL = '2024-04-01'                #Data final

fig = make_subplots(rows=2, cols=1, shared_xaxes=True, vertical_spacing=0.2, row_width=[0.2, 0.7])
fig.add_trace(go.Candlestick(x=df_SVM.index, open=df_SVM['Open'], high=df_SVM['High'],
low=df_SVM['Low'], close=df_SVM['Close'], showlegend=False, line=dict(width=1.0),
opacity=1.0,
            increasing_fillcolor='#dadce3', decreasing_fillcolor="#5d606b",
            increasing_line_color='#434651', decreasing_line_color='#434651'), row=1, col=1)


fig.add_trace(go.Scatter(x=df_SVM.index, y=df_SVM['Equity'], name='Equity',
showlegend=False, opacity=1.0, mode='lines', line=dict(color="#0000FF", width=1.5,
dash='solid')), row=2, col=1)

fig.update_yaxes(type='linear', row=1, col=1)
fig.update_xaxes(rangebreaks=[dict(bounds=["sat", "mon"])])
fig.update(layout_xaxis_rangeslider_visible=False)
fig.update_layout(title= 'SVM ' + TICKET + str(TIMEFRAME) + '   -   ' + INICIO + ' a ' +
FINAL, xaxis_tickfont_size=12, xaxis=dict(titlefont_size=14, tickfont_size=12),
yaxis=dict(title='Price',
            titlefont_size=14, tickfont_size=12), height=1500,width=1000, autosize=False,
            margin=dict(l=0,r=20, b=20, t=40, pad=0), paper_bgcolor='white',
            template='simple_white')
fig.show()


fig = make_subplots(rows=2, cols=1, shared_xaxes=True, vertical_spacing=0.2, row_width=
[0.2, 0.7])
fig.add_trace(go.Candlestick(x=df_DT.index, open=df_DT['Open'], high=df_DT['High'],
low=df_DT['Low'], close=df_DT['Close'], showlegend=False, line=dict(width=1.0),
opacity=1.0,
            increasing_fillcolor='#dadce3', decreasing_fillcolor="#5d606b",
            increasing_line_color='#434651', decreasing_line_color='#434651'), row=1,
            col=1)


fig.add_trace(go.Scatter(x=df_DT.index, y=df_DT['Equity'], name='Equity', showlegend=False,
opacity=1.0, mode='lines', line=dict(color="#0000FF", width=1.5, dash='solid')), row=2,
col=1)

fig.update_yaxes(type='linear', row=1, col=1)
fig.update_xaxes(rangebreaks=[dict(bounds=["sat", "mon"])])
fig.update(layout_xaxis_rangeslider_visible=False)
fig.update_layout(title= ' rvore  de Decis o ' + TICKET + str(TIMEFRAME) + '   -   ' +
INICIO + ' a ' + FINAL, xaxis_tickfont_size=12, xaxis=dict(titlefont_size=14,
tickfont_size=12), yaxis=dict(title='Price',
            titlefont_size=14, tickfont_size=12), height=1500,width=1000, autosize=False,
            margin=dict(l=0,r=20, b=20, t=40, pad=0), paper_bgcolor='white',
            template='simple_white')
fig.show()


fig = make_subplots(rows=2, cols=1, shared_xaxes=True, vertical_spacing=0.2, row_width=
[0.2, 0.7])
fig.add_trace(go.Candlestick(x=df_RN.index, open=df_RF['Open'], high=df_RF['High'],
low=df_RF['Low'], close=df_RF['Close'], showlegend=False, line=dict(width=1.0),
opacity=1.0,
            increasing_fillcolor='#dadce3', decreasing_fillcolor="#5d606b",
            increasing_line_color='#434651', decreasing_line_color='#434651'), row=1,
            col=1)
```

```
fig.add_trace(go.Scatter(x=df_RF.index, y=df_RF['Equity'], name='Equity', showlegend=False,
opacity=1.0, mode='lines', line=dict(color="#0000FF", width=1.5, dash='solid')), row=2,
col=1)

fig.update_yaxes(type='linear', row=1, col=1)
fig.update_xaxes(rangebreaks=[dict(bounds=["sat", "mon"])])
fig.update(layout_xaxis_rangeslider_visible=False)
fig.update_layout(title= 'Floresta Aleat ria ' + TICKET + str(TIMEFRAME) + '   -   ' +
INICIO + ' a ' + FINAL, xaxis_tickfont_size=12, xaxis=dict(titlefont_size=14,
tickfont_size=12), yaxis=dict(title='Price',
            titlefont_size=14, tickfont_size=12), height=1500, autosize=True,
            margin=dict(l=0,r=20, b=20, t=40, pad=0), paper_bgcolor='white',
            template='simple_white')
fig.show()


fig = make_subplots(rows=2, cols=1, shared_xaxes=True, vertical_spacing=0.2, row_width=
[0.2, 0.7])
fig.add_trace(go.Candlestick(x=df_RN.index, open=df_RN['Open'], high=df_RN['High'],
low=df_RN['Low'], close=df_RN['Close'], showlegend=False, line=dict(width=1.0),
opacity=1.0,
            increasing_fillcolor='#dadce3', decreasing_fillcolor="#5d606b",
            increasing_line_color='#434651', decreasing_line_color='#434651'), row=1,
            col=1)


fig.add_trace(go.Scatter(x=df_RN.index, y=df_RN['Equity'], name='Equity', showlegend=False,
opacity=1.0, mode='lines', line=dict(color="#0000FF", width=1.5, dash='solid')), row=2,
col=1)

fig.update_yaxes(type='linear', row=1, col=1)
fig.update_xaxes(rangebreaks=[dict(bounds=["sat", "mon"])])
fig.update(layout_xaxis_rangeslider_visible=False)
fig.update_layout(title= 'Rede Neural ' + TICKET + str(TIMEFRAME) + '   -   ' + INICIO + '
a ' + FINAL, xaxis_tickfont_size=12, xaxis=dict(titlefont_size=14, tickfont_size=12),
yaxis=dict(title='Price',
            titlefont_size=14, tickfont_size=12), height=1500, autosize=True,
            margin=dict(l=0,r=20, b=20, t=40, pad=0), paper_bgcolor='white',
            template='simple_white')
fig.show()
```