

TÉCNICAS DE AVALIAÇÃO DA BOLSA DE VALORES E ANÁLISE DE SEU COMPORTAMENTO COM A INSERÇÃO DE ALGORITMOS INTELIGENTES

Trabalho de Conclusão de Curso

Engenharia da Computação

Rodrigo Santiago Borges

Orientador: Prof. Dr. Sérgio Murilo Maciel Fernandes





Universidade de Pernambuco
Escola Politécnica de Pernambuco
Graduação em Engenharia de Computação

Rodrigo Santiago Borges

**TÉCNICAS DE AVALIAÇÃO DA BOLSA
DE VALORES E ANÁLISE DE SEU
COMPORTAMENTO COM A INSERÇÃO
DE ALGORITMOS INTELIGENTES**

Monografia apresentada como requisito parcial para obtenção do diploma de
Bacharel em Engenharia de Computação pela Escola Politécnica de Pernambuco –
POLI da Universidade de Pernambuco.

Recife, dezembro de 2020

Rodrigo Santiago Borges

Técnicas de avaliação da bolsa de valores e análise de seu comportamento com a inserção de algoritmos inteligentes/ Rodrigo Santiago Borges – Recife – PE, Brasil, dezembro 2020

57p.

Orientador: Sergio Murilo Maciel Fernandes

Trabalho de Conclusão de Curso – Engenharia da Computação
Escola Politécnica de Pernambuco

Universidade de Pernambuco, dezembro de 2020

1. RNA 2. MLP com algoritmo de treinamento backpropagation 3. Mercado de Ações 4. Série Temporal I. Prof. Sergio Murilo Maciel Fernandes. II. Universidade de Pernambuco. III. Escola Politécnica. IV Técnicas de avaliação da bolsa de valores e análise de seu comportamento com a inserção de algoritmos inteligentes.

MONOGRAFIA DE FINAL DE CURSO

Avaliação Final (para o presidente da banca)*

No dia 1/12/2020, às 14h00min, reuniu-se para deliberar sobre a defesa da monografia de conclusão de curso do(a) discente **RODRIGO SANTIAGO BORGES**, orientado(a) pelo(a) professor(a) **SÉRGIO MURILO MACIEL FERNANDES**, sob título **TÉCNICAS DE AVALIAÇÃO DA BOLSA DE VALORES E ANÁLISE DE SEU COMPORTAMENTO COM A INSERÇÃO DE ALGORITMOS INTELIGENTES**, a banca composta pelos professores:

DANIEL AUGUSTO RIBEIRO CHAVES (PRESIDENTE)
SÉRGIO MURILO MACIEL FERNANDES (ORIENTADOR)

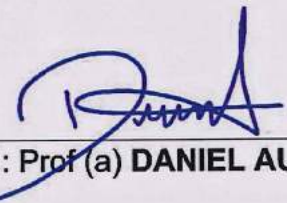
Após a apresentação da monografia e discussão entre os membros da Banca, a mesma foi considerada:

Aprovada Aprovada com Restrições* Reprovada


e foi-lhe atribuída nota: 8,5 (Oito, Cinco)

*(Obrigatório o preenchimento do campo abaixo com comentários para o autor)

O(A) discente terá 30 dias para entrega da versão final da monografia a contar da data deste documento.



AVALIADOR 1: Prof (a) **DANIEL AUGUSTO RIBEIRO CHAVES**



AVALIADOR 2: Prof (a) **SÉRGIO MURILO MACIEL FERNANDES**

AVALIADOR 3: Prof (a)

* Este documento deverá ser encadernado juntamente com a monografia em versão final.

Dedico este trabalho ao meu filho Arthur e ao meu pai Arlindo Borges (in memoriam) que sempre foi exemplo de generosidade e honestidade.

Agradecimentos

Agradeço à minha esposa que sempre me incentivou a buscar o novo e a enfrentar os desafios, estando sempre ao meu lado, inclusive nessa árdua jornada de conclusão acadêmica.

Agradeço à minha mãe e minha irmã pelo apoio de sempre.

Agradeço aos amigos e professores da Escola Politécnica de Pernambuco - POLI que fizeram parte da minha formação e evolução pessoal me encorajando no alcance de novos horizontes, em especial, à Jonathan Bandeira e ao meu orientador, Prof. Sérgio Murilo.

Autorização de publicação de PFC

Eu, **Rodrigo Santiago Borges** autor(a) do projeto de final de curso intitulado: **TÉCNICAS DE AVALIAÇÃO DA BOLSA DE VALORES E ANÁLISE DE SEU COMPORTAMENTO COM A INSERÇÃO DE ALGORITMOS INTELIGENTES**; autorizo a publicação de seu conteúdo na internet nos portais da Escola Politécnica de Pernambuco e Universidade de Pernambuco.

O conteúdo do projeto de final de curso é de responsabilidade do autor.

Rodrigo Santiago Borges

Rodrigo Santiago Borges

Sérgio Murilo Maciel Fernandes

Orientador(a): **Sérgio Murilo Maciel Fernandes**

Coorientador(a):



Prof, de TCC: **Daniel Augusto Ribeiro Chaves**

Data: 1/12/2020

Resumo

O Mercado de ações opera na negociação de títulos e é organizado por agentes que realizam avaliações através de análises técnicas/gráficas dos preços futuros dos papéis. A inteligência artificial vem ganhando espaço já que a análise técnica se torna difusa e de difícil interpretação pelo ser humano. Este trabalho tem por objetivo realizar a predição de um índice de referência de uma série temporal da Bolsa de valores italiana utilizando uma rede neural *feedforward*, multicamada, com aprendizado *backpropagation*. O algoritmo foi implementado através da linguagem Python utilizando recursos do Google Colab. Os dados foram normalizados em um intervalo de 0 a 1 para uso da Rede Neural Artificial (RNA). A acurácia final apresentada após o treinamento e simulação da rede foi de 56,41%. Com os resultados apresentados o objetivo da pesquisa foi atingido satisfatoriamente mediante comparação com outros trabalhos desenvolvidos com algoritmos inteligentes.

Palavras chave: Mercado de Ações, MLP, RNA, *backpropagation*, Série Temporal.

Abstract

Stock market operates in the trading of securities and is organized by agents who perform valuation through technical / graphical analysis of the future prices of securities. Artificial Intelligence (AI) has been gaining space in reason of the technical analysis has become diffuse and difficult for human to analyse. This Project aims to project a reference index of an Italian Stock Exchange time series using a feedforward neural network, multilayer with backpropagation learning. The algorithm was implemented through Python language using the Google Colab resources. The data was normalized in a range of 0 to 1 for Artificial Neural Network (ANN) use. The final accuracy presented after network training and simulation was 56,41 %. With the results obtained, the research objective was satisfactorily achieved by comparison with other works developed with intelligent algorithms.

Keywords: Stock Market, MLP, ANN *backpropagation*, Time Series.

Sumário

Capítulo 1 Introdução	13
1.1 Motivação	13
1.2 Objetivos	14
1.2.1 Objetivo Geral	14
1.2.2 Objetivos Específicos	14
1.3 Organização	15
Capítulo 2 Fundamentação Teórica	16
2.1 Bolsa de Valores e Análise de Investimentos	16
2.1.1 Análise Fundamentalista	18
2.1.2 Análise Técnica	19
2.2 Previsão de séries temporais	21
2.2.1 Séries temporais	21
2.2.2 Rastreador de tendência.	21
2.2.3 Osciladores Estocásticos	23
2.2.4 MACD – Média móvel de convergência/divergência	24
2.2.5 Osciladores Índice de Força Relativa (IFR ou RSI)	25
2.2.6 Oscilador Williams %R	26
2.3 Redes Neurais	28

2.3.1	Neurônio Artificial	29
2.3.2	Elementos de um neurônio artificial	31
2.3.2.1	Sinais de entrada	31
2.3.2.2	A entrada líquida "net"	31
2.3.2.3	Função de Ativação (fa):	32
2.4	Treinamento da rede	34
2.4.1	Algoritmo de treinamento	35
2.4.2	Normalização de Dados	35
2.4.3	Crítérios de parada	36
2.5	Classificação da RNA	38
2.5.1	Função Custo	38
2.5.2	Otimizadores	38
2.5.3	Métricas	39
	Capítulo 3 Trabalhos relacionados	41
	Capítulo 4 Metodologia e Resultados	41
4.1	Tendência e sinal de negociação	43
4.2	Decisão de Negociação	44
4.3	Indicadores técnicos escolhidos	44
4.4	Processamento dos dados	45
4.5	Configurações da rede	46
4.5.1	Treinamento	47

4.6	Tratamento e normalização dos dados	48
4.7	Resultados	49
	Capítulo 5 Considerações Finais	52
5.1	Conclusão	52
5.2	Sugestões de trabalhos futuros	53
	Capítulo 6 Referências Bibliográficas	54
	Apêndice A - Código Fonte da RNA	58

Índice de Figuras

Figura 1 - Tipos de Gráficos para Análise Técnica	20
Figura 2 - Gráfico para identificação de tendências	22
Figura 3 - Gráfico de Osciladores Estocásticos	23
Figura 4 - Gráfico de Média Móvel Convergência/Divergência	25
Figura 5 - Gráfico de Índice de Força Relativa	26
Figura 6 - Gráfico Oscilador Williams %R	27
Figura 7 - Célula neural biológica com a sequência de propagação do sinal	29
Figura 8 - Modelo matemático de uma Rede Perceptron de uma única camada	31
Figura 9 - Modelo de neurônio artificial com função de ativação em destaque	32
Figura 10 - Gráfico Função Linear	33
Figura 11 - Gráfico Função Sigmoidal	33
Figura 12 - Gráfico Função Tangente Hiperbólica	34
Figura 13 - Critério de Parada	37
Figura 14 – Cross Validation	47

Tabela de Símbolos e Siglas

Adam	- Adaptive moment estimation
BP	- Neural Network BackPropagation
CART	- Classification And Regression Trees
CBR	- Case-Based Reasoning
COVID19	- Corona Virus Disease 2019
GRU	- Gated Recurrent Unit
IFR	- Índice de Força Relativa
LDA	- Linear Discriminant Analysis
LSTM	- Long Short Term Memory
MACD	- Moving Average of Convergence / Divergence
MLP	- Multilayer Perceptron
MMA	- Médias Móveis Aritméticas
MME	- Médias Móveis Exponenciais
OBV	- On Balance Volume
QDA	- Quadratic Discriminant Analysis
RNA	- Rede Neural Artificial
SMA	- Simple Moving Average
SVM	- Support Vector Machines

Capítulo 1

Introdução

1.1 Motivação

Com a evolução dos mercados financeiros e com as transformações nos valores das ações é perceptível a necessidade de aplicação de técnicas eficientes para analisar o comportamento de ativos visando uma melhor previsibilidade de investimentos.

A expectativa de lucro nesse tipo de aplicação é uma possibilidade e não uma certeza, e nem sempre é possível identificar os riscos que podem influenciar o valor de uma ação.

Prever o comportamento futuro de uma ação não é uma tarefa fácil, sendo importante buscar meios que auxiliem o entendimento de estratégias e providências para obter êxito neste mercado.

No caso específico de previsão de séries temporais relacionadas ao mercado acionário, é importante prever com maior precisão a tendência do movimento (subida ou descida) da série, visto que tais características podem auxiliar na estratégia de compra e venda da mesma. Preços de ativos financeiros dependem, em geral, de outros preços do mercado. A volatilidade também pode depender de outros eventos, como anúncios governamentais de taxas de desemprego, de inflação, PIB trimestral, taxa de juros, entre outros (Amorim, 2008).

Uma ação varia com a demanda de oferta/procura de aplicadores o que aponta a imprevisibilidade dos preços incentivando a descoberta de meios consideráveis para obtenção de uma boa predição desses valores.

Não há um indicador que seja capaz de sinalizar sempre, e de forma consistente, os momentos para tomar uma decisão, eles têm uma característica que os fazem melhores em determinadas situações de mercado (Sachetini, 2006).

Nessas condições, surge a oportunidade de abordar a Rede Neural Artificial que é uma das habituais técnicas da inteligência artificial aplicadas na previsão de séries temporais, que pode produzir com sucesso um modelo eficiente na previsão do mercado de ações.

1.2 Objetivos

1.2.1 Objetivo Geral

Analisar a previsão de preço futuro de fechamento do índice de referência do mercado de ações usando Redes Neurais Artificiais.

1.2.2 Objetivos Específicos

1. Estudar as análises técnica e fundamentalista para compreender melhor o escopo de investimentos e criar modelo para entrada de dados.
2. Demonstrar o uso de algoritmos inteligentes para entendimento do comportamento do mercado;
3. Apresentar um modelo de algoritmo RNA Multilayer Perceptron (MLP) no auxílio de decisão para compra/manutenção/venda de uma ação;
4. Realizar a comparação de pesquisas que utilizaram técnicas de Inteligência Artificial visando previsão futura do valor de um título financeiro.

1.3 Organização

O presente trabalho está organizado da seguinte forma:

- **CAPÍTULO 1** – Neste capítulo é realizada uma breve introdução apresentando motivação e objetivos do presente trabalho.
- **CAPÍTULO 2** – Neste capítulo são tratados os conceitos e fundamentos referentes à bolsa de valores e análise de investimentos, estocástica e previsão de séries temporais, redes neurais e treinamento de uma RNA.
- **CAPÍTULO 3** – Apresentação de análise comparativa dos resultados de trabalhos de pesquisa que utilizaram técnicas de Inteligência Artificial para predição futura do valor de uma ação.
- **CAPÍTULO 4** – Neste capítulo são apresentados os aspectos e resultados da metodologia proposta.
- **CAPÍTULO 5** – Este último capítulo apresenta as considerações finais e sugestões de trabalhos possíveis de serem desenvolvidos posteriormente.
- **APÊNDICE** – São apresentados os algoritmos propostos na metodologia.

Capítulo 2

Fundamentação Teórica

2.1 Bolsa de Valores e Análise de Investimentos

A bolsa de valores é o mercado onde são negociadas ações, títulos e outros valores mobiliários, podendo ser organizado por uma sociedade anônima que visa lucro ou por uma sociedade sem fins lucrativos.

A primeira bolsa de valores do mundo surgiu por volta de 1487, em Bruges, na Bélgica, com a expansão comercial. Mais tarde, em 1531, seria criada a bolsa de Antuérpia, também na Bélgica, baseada na negociação de empréstimos e considerada a primeira bolsa oficial.

As primeiras ações de que se tem notícia foram emitidas em 1602, na bolsa de Amsterdã, pela Companhia Holandesa das Índias Orientais, que na época monopolizava a colonização na Ásia. No Brasil, a primeira foi instalada no Rio de Janeiro, em 1845.

Ao adquirir uma ação a pessoa torna-se sócia da companhia, dividindo lucros e prejuízos. Para a empresa, o maior objetivo de emitir ações é obter mais dinheiro, a um custo baixo, para poder crescer e, por consequência, lucrar mais.

Existem dois tipos de investimentos, aqueles com rentabilidade fixa, geralmente praticado por quem não possui uma grande experiência com compra ou venda de ações ou quem não quer arriscar no mercado e o investimento que não se tem previsibilidade sobre o retorno, onde é preciso ter uma noção sobre o histórico da empresa, e o macro da economia, podendo, porém, ter altos desempenhos no que diz respeito ao investimento inicial.

A B3 SA (estilizado como [B]³ em referência às letras iniciais de Brasil, Bolsa, Balcão) é uma das maiores empresas de infraestrutura do mercado financeiro do

mundo, prestando serviços de negociação em ambiente de câmbio e balcão sendo uma das maiores empresas em valor de mercado, mantendo uma posição global de destaque no mercado de ações. Sediada na cidade de São Paulo surgiu sob o formato atual após a fusão da Bolsa de Valores, Mercadorias e Futuros de São Paulo (BM&FBOVESPA) com a Central de Custódia e de Liquidação Financeira de Títulos (CETIP).

A BM&FBOVESPA havia surgido em 8 de maio de 2008, quando houve a fusão da Bolsa de Valores de São Paulo (Bovespa) e a Bolsa de Mercadorias e Futuros (BM&F).

O Ibovespa é o principal indicador de desempenho das ações negociadas em B3 e lista as principais empresas do mercado de capitais brasileiro. Foi criada em 1968 e, nos últimos 50 anos, estabeleceu uma referência para investidores em todo o mundo.

O índice é reavaliado a cada quatro meses e resulta de uma carteira teórica de ações. É composto por ações e unidades de empresas listadas na B3 que atendem aos critérios descritos em sua metodologia, representando cerca de 80% do número de negócios e do volume financeiro de nossos mercados de capitais.

Alguns eventos avassaladores como a Grande Depressão de 1929 (Bolsa de Valores de Nova Iorque), e a crise mundial do *subprime* (2008) e atualmente a pandemia da Covid-19 acentuou a instabilidade nos mercados financeiros globais e redobrou a aposta de queda em bolsas de valores de todo o mundo.

A crise do Coronavírus causou uma volatilidade muito grande e um pânico generalizado no mercado financeiro e é justamente em momentos como esse, que investidores e especuladores buscam tentar se capitalizar, diante da expectativa de queda. Assim, pode ser perceptível que o uso de algoritmo de análise técnica para tomar decisões apenas baseado em estatísticas poderá ter um desempenho inferior ao da análise humana, pois são operações que exigem um certo refinamento estratégico para não esbarrar em impactos negativos diante do atual cenário econômico de crise mundial.

Diante de um cenário com a maior inconstância do século: isso significa que a instabilidade nunca foi tão grande, o risco de se fazer cálculo é bastante elevado e envolve diversas variáveis. Dessa forma, a Inteligência artificial pode levar vantagem em cima de técnicas como no caso da estocásticas e outras usadas no mercado financeiro mediante a sua grande capacidade analítica uma vez que em nenhuma outra Crise Mundial se observou um cenário tão confuso.

Baldwi e Weder Di Mauro (2020) destacam que alguns efeitos podem ser mais persistentes, principalmente no se refere às interrupções que as empresas, os indivíduos e os governos estão experimentando, o que implicará riscos para a globalização e para a integração mundial. As empresas e as cadeias globais de suprimentos podem ser abruptamente quebradas por um choque na saúde. Assim, a economia global vem hoje sendo fortemente afetada por um evento raro e de fortíssimo impacto, comumente chamado de balck swan (IMAI et al., 2020), que tem provocado um lockdown (LIN et al., 2020) nas cadeias globais de valor e pronunciado retração na demanda de bens e serviços em virtude das políticas de isolamento social (quarentenas).

Para operações em mercados de bolsa de valores a análise de investimentos pode ser associada a correntes de pensamento que seguem duas escolas: a escola fundamentalista pertinente à análise da empresa e da economia e a escola técnica atinente com a análise do comportamento dos preços das ações.

2.1.1 Análise Fundamentalista

A análise fundamentalista é uma dentre um conjunto de estratégias que têm sido empregadas há muito tempo com o objetivo de aumentar os lucros decorrentes do mercado de ações.

Na realidade, o indivíduo que emprega a análise fundamentalista não é um investidor, mas antes de tudo um especulador. Contudo, devido aos efeitos positivos desse tipo particular de especulação e em virtude da conotação negativa associada àquela palavra, a maioria dos indivíduos que empregam esta técnica preferem intitular-se investidores (Walter, 1974).

Essa análise utiliza estudos de históricos financeiro dos investimentos com a finalidade de prever o comportamento futuro do preço de ativos e, consiste não apenas na elaboração do modelo matemático correto, mas principalmente das premissas bem utilizadas e da análise qualitativa, em que são apresentados os pontos positivos e negativos para o investimento.

A justificativa desse tipo de análise é antecipar o comportamento futuro de uma determinada empresa no mercado, isto é, adiantar-se ao mercado. Para que isso seja certo, tem que partir de uma hipótese básica: o mercado não é eficiente no curto prazo. Se não fosse assim, não seria possível adiantar-se ao mercado. Hoje o preço de uma ação não reflete o verdadeiro valor, mas existe uma tendência de que isso ocorra em um futuro próximo. O analista fundamentalista trata o tempo todo de descobrir supervalorizações ou subvalorizações, com base em determinada informação ainda não negociada pelo mercado. (Pinheiro, 2009).

2.1.2 Análise Técnica

A análise técnica consiste no estudo do comportamento histórico do mercado, observando suas tendências e reações, com vistas à determinação de seu estado atual e das suas condições futuras. Uma das suas principais premissas é a de que o mercado apresenta repetições de padrões de comportamento, ou seja, que o seu comportamento futuro é consistente com o passado. Assim, a principal característica desse tipo de análise é a predição do momento em que os preços irão subir ou cair, indicando o instante para entrar ou sair do mercado (CHAVES e ROCHA, 2004).

A leitura de gráficos é um componente essencial da Análise Técnica, pois permite estudar em detalhes as movimentações do mercado e estipular flutuações de preço futuras das ações, conforme Figura 1 que apresenta a série temporal através de vários tipos de gráficos, fornecendo diferentes visões como o valor de abertura/fechamento/máximo/mínimo e volume negociado.



Figura 1. Tipos de gráficos para análise técnica
Fonte: Metastock

2.2 Previsão de séries temporais

2.2.1 Séries temporais

Uma série temporal é uma sequência ordenada de valores de uma variável em intervalos de tempo igualmente espaçados, sendo considerada univariada quando consiste de observações únicas (Morettin e Toloí, 2006).

A análise de séries temporais leva em conta o fato de que os pontos de dados tomados no decorrer do tempo possam ter uma estrutura interna, tal como correlação, tendência ou variações sazonais que deverá ser considerada. As séries são muito utilizadas para entender as forças e estruturas que geraram os dados observados, assim como ajustar um modelo matemático que permita o monitoramento e a previsão da variável de interesse (Morettin e Toloí, 2006).

Os modelos estatísticos para séries temporais utilizam o passado histórico da variável para projetar observações futuras. Dessa forma, se pode ter uma ideia, em média, de como a variável se comportará nos próximos períodos.

2.2.2 Rastreador de tendência.

As médias móveis são precisamente o tipo de indicador mais popular e também o mais antigo da análise técnica, onde sua principal utilidade é de indicar, através da inclinação de uma média de preço, a direção em que ocorre a movimentação de valores em diferentes prazos de tempo e a partir disso indicar a direção da força do mercado em diferentes períodos. Daí é que esse tipo de indicador é classificado como rastreador de tendência ou "*trend following*" (Moreno et. al., 2008).

Os valores calculados são representados por pontos e situados no gráfico conforme a escala de preço do ativo, os quais são unidos através de uma linha que se torna a média móvel do ativo na determinada amostragem.

O cálculo aritmético e o exponencial são dois métodos mais simples diante de vários métodos diferentes para se calcular o valor da média.

As Médias Móveis Aritméticas (MMA) também chamadas de médias simples utilizam o método de cálculo mais simples, o qual pode ser expresso pela fórmula:

$$MMA(n) = \frac{V1 + V2 + \dots + Vn}{n} \quad (1)$$

Cada V representa o valor do preço da série defasada de n termos.

As Médias Móveis Exponenciais (MME) dão mais ênfase para os valores mais recentes no tempo e possuem um cálculo um pouco mais refinado. Esse tipo de cálculo pode seguir diversas linhas, mas o mais utilizado na análise técnica segue a fórmula:

$$MME(n) = (V(n) * k) + (MME(n - 1) * (1 - K)) \quad (2)$$

$$K = \frac{2}{n + 1} \quad (3)$$

O uso da MME associado a MMA possibilita a identificação de tendência pela simples observação de suas inclinações e da associação das médias móveis para decisão de compra/venda, conforme observado na Figura 2.



Figura 2. Gráfico para identificação de tendências

Fonte: Bússola do Investidor.

2.2.3 Osciladores Estocásticos

O Indicador Oscilador Estocástico revela em termos percentuais a relação da cotação atual de um ativo com seu maior ou menor preço em um determinado período de tempo (Morettin e Toloj, 2006).

A linha principal é chamada de %K normalmente apresentada na cor azul e a segunda linha chamada de %D é uma Média Móvel da %K, normalmente representada na cor vermelha, conforme Figura 3.



Figura 3: Gráfico de Osciladores Estocásticos
Fonte: Bússola do Investidor

Dentre várias maneiras de interpretar um Oscilador Estocástico os três métodos mais comuns no mercado são:

1. Compre quando o oscilador (tanto a %K quanto a %D) cai abaixo de um determinado nível e, em seguida, sobe acima desse nível. Venda quando o oscilador sobe acima de um determinado nível e, em seguida, cai abaixo desse nível.

2. Compre quando a linha %K sobe acima da %D. Venda quando a %K cai abaixo da %D.
3. Busque por divergências. A série pode fazer novos picos e o Oscilador Estocástico não está conseguindo atingir seus topos anteriores.

Quando o estocástico já se encontra acima dos 80% e corta este valor na descendente, temos o sinal de venda. Quando o indicador já está abaixo dos 20% e corta este percentual na ascendente, temos o sinal de compra (PERSON, J. L., 2004).

2.2.4 MACD – Média móvel de convergência/divergência

O Moving Average Convergence/Divergence (MACD) é um oscilador que deve revelar mudanças na força, direção, momento e duração de uma tendência no preço de uma ação (Morettin e Tolo, 2006).

A versão mais simples do MACD é a diferença entre duas médias móveis, uma em um período mais curto (n) e outra por um longo período (m).

$$MACD(n, m) = MME(n) - MME(m) \quad (4)$$

Informações adicionais podem ser obtidas usando uma terceira média móvel da $MACD(n, m)$ durante um período (s), chamada "linha de sinal" $SL(s)$.

Quando o $MACD(n, m)$ aumenta e cruza a linha de sinal é um momento de alta; quando diminui e cruza a linha de sinal, é um momento de baixa.

$$MACD(n, m, s) = MME(n) - MME(m) - SL(s) \quad (5)$$

O MACD pode assumir qualquer valor real, positivo ou negativo onde os valores positivos denotam que a tendência do índice está aumentando e vice-versa, conforme pode ser observado na Figura 4.

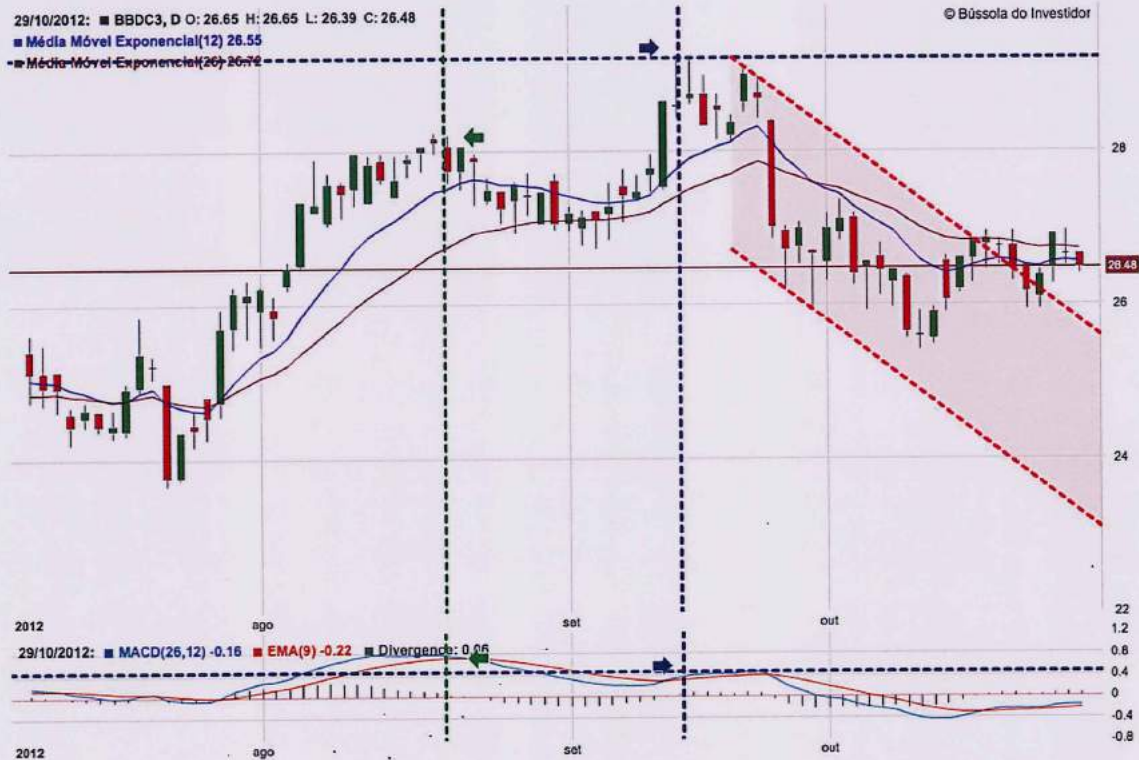


Figura 4: Gráfico de Média Móvel Convergência/Divergência
Fonte: Bússola do Investidor

2.2.5 Osciladores Índice de Força Relativa (IFR ou RSI)

O Índice de Força Relativa é um oscilador de momento que compara a magnitude dos ganhos e perdas recentes em um período de tempo especificado para medir a velocidade e a alteração dos movimentos de preços de um título (Morettin e Tolo, 2006).

O IFR pode variar entre 0 e 100 e funciona como um termômetro do cansaço do mercado, medindo se ele já subiu ou caiu muito e precisa tomar fôlego antes de continuar na mesma tendência como pode ser visualizado na Figura 5.

$$IFR = 100 - (1 + FR) \quad (6)$$

Para fazer o cálculo do IFR de um ativo tem-se que $IFR = \text{Índice de Força Relativa}$ e $FR = \text{Ganho médio} \div \text{Perda média}$.

Para simplificar a explicação de cálculo, o IFR tem sido dividido em seus componentes básicos: FR, Ganho Médio e Perda Média e este cálculo do IFR é baseado em uma quantidade de períodos onde as perdas são expressas em valores positivos.



Figura 5: Gráfico de Índice de Força Relativa
Fonte: Bússola do Investidor

2.2.6 Oscilador Williams %R

O Williams %R é um oscilador que analisa se um mercado de ações ou de commodities está sendo negociado próximo à alta ou à baixa, ou em algum lugar intermediário, de sua recente faixa de negociação, conforme Figura 6.

$$\%R(n) = -100 * \frac{\max(\max(t), \max(t-1), \dots, \max(t-n+1)) - c(t)}{\max(\max(t), \max(t-1), \dots, \max(t-n+1)) - \min(\min(t), \min(t-1), \dots, \min(t-n+1))} \quad (7)$$



Figura 6: Gráfico Oscilador Williams %R
Fonte: Bússola do Investidor

2.3 Redes Neurais

Uma RNA é um modelo computacional/matemático inspirado no sistema nervoso dos seres vivos que adquire conhecimento através da experiência. É formada basicamente por um conjunto de unidades de processamento (neurônios artificiais) que são interligados por um enorme número de interconexões (sinapses artificiais) representadas por vetores/matrizes de peso sináptico (Haykin, 2001).

A grande vantagem das redes neurais para solução de problemas complexos provém de algumas propriedades:

1. **Adaptabilidade:** capacidade da RNA se adaptar aos pesos sinápticos em face a modificações no meio ambiente, onde ela pode ser treinada a operar em um ambiente e depois ser retreinada para absorver pequenas modificações desse ambiente;
2. **Aprendizagem:** habilidade da RNA aprender automaticamente, o mapeamento desejado entre a entrada e a saída, através de um processo iterativo de ajustes aplicados aos seus parâmetros livres
3. **Generalização:** capacidade da RNA apresentar uma saída adequada para uma entrada não presente no processo de aprendizagem;
4. **Não-Linearidade:** uma RNA é não linear se for constituída de neurônios artificiais também não lineares. Esta é uma característica relevante porque a maioria dos sistemas físicos responsáveis pela geração do mapeamento entre os sinais de entrada e saída desejada são não-lineares;
5. **Resposta a Evidências:** uma RNA pode gerar um indicador sobre a sua confiança e pode classificar de acordo com os sinais de entrada e de retroalimentação;
6. **Tolerância a Falhas:** a informação é distribuída por toda RNA. Mesmo que parte das conexões estejam comprometidas não haverá mudança significativa no desempenho da RNA.

Os diferentes modelos de RNAs podem ser diferenciados pelo seu tipo de treinamento que pode ser supervisionado ou não supervisionado e também pelo tipo de arquitetura (Redes Alimentadas Adiante com Camada Única, Redes Alimentadas Diretamente com Múltiplas Camadas e Redes Recorrentes), (Haykin, 2001).

As redes MLPs são redes supervisionadas. O sinal de entrada se propaga para frente através da rede camada por camada. Apresenta um processo de aprendizagem supervisionado, no qual os padrões de treinamento são apresentados à rede e com base nos erros obtidos são realizados ajustes nos pesos sinápticos, cuja finalidade é diminuir os erros nas próximas iterações. O principal algoritmo de treinamento das redes MLPs é o algoritmo de retropropagação (back-propagation) proposto no início por (Werbos, 1974).

2.3.1 Neurônio Artificial

O neurônio artificial é um modelo simplificado e simulado do neurônio real e suas características básicas são a adaptação e a representação de conhecimentos baseada em conexões, conforme Figura 7.

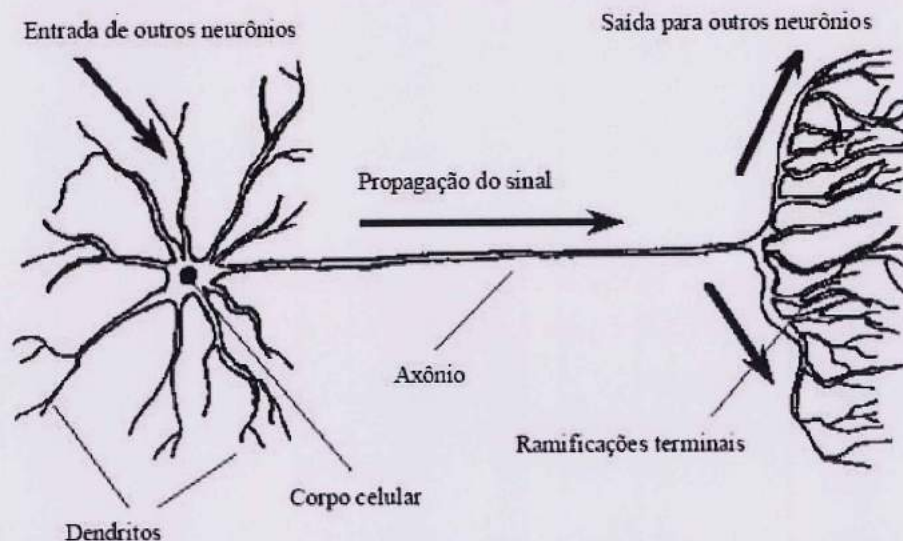


Figura 7. Célula neural biológica com a sequência de propagação do sinal.
Fonte: Silva (1998).

Em 1943, McCulloch & Pitts criaram um modelo matemático onde viu-se o cérebro como um sistema computacional. Este modelo é uma simplificação do que se sabia até então a respeito do neurônio biológico (Haykin, 2001).

Suposições básicas do modelo:

- A atividade de um neurônio é um processo tudo ou nada.
- Um certo número fixo (>1) de entradas devem ser excitadas dentro de um período de adição latente para excitar um neurônio.
- Único atraso significativo é o atraso sináptico.
- A atividade de qualquer sinapse inibitória previne absolutamente a excitação do neurônio.
- A estrutura das interconexões não muda com o tempo.

Descrição matemática do modelo:

- Tem-se o neurônio com n terminais de entrada x_1, x_2, \dots, x_n (dendritos) e apenas um terminal de saída y (axônio).
- O comportamento das sinapses é emulado através de pesos w_1, w_2, \dots, w_n acoplados as entradas dos neurônios, cujo valor pode ser excitatório ou inibitório.
- O efeito de uma sinapse i no neurônio pós-sináptico é dado por $w_i * x_i$.

2.3.2 Elementos de um neurônio artificial

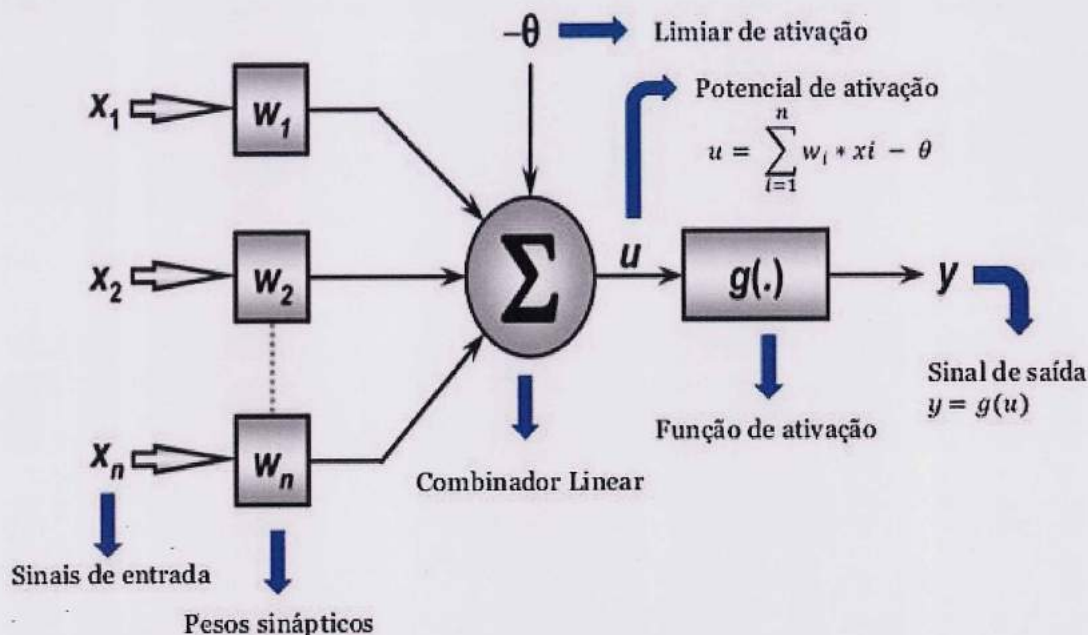


Figura 8: Modelo matemático de uma Rede Perceptron de uma única camada
Fonte: <https://www.embarcados.com.br/rede-perceptron-de-uma-unica-camada/>

2.3.2.1 Sinais de entrada

- As entradas podem ser as saídas de outros neurônios, entradas externas, bias ou qualquer combinação destes elementos.
- Bias (Tendência), “- θ ” é um erro sistemático diferentemente de erro aleatório. Um ou mais componentes do erro sistemático podem contribuir para a tendência. Uma grande diferença sistemática em relação ao valor de referência aceito reflete-se em um grande valor de tendência.

2.3.2.2 A entrada líquida “net”

O somatório de todas estas entradas, multiplicadas por suas respectivas forças de conexão sináptica, os pesos, dá origem ao chamado "net" de um neurônio.

$$\text{net}(t) = \sum_{j=1}^N w_{ij} * x_j \quad (8)$$

O w_{ij} é um número real que representa a conexão sináptica da entrada i -ésimo neurônio. A conexão sináptica é conhecida como excitatória se $w_{ij} > 0$ ou inibitória caso $w_{ij} < 0$.

2.3.2.3 Função de Ativação (fa):

Após a determinação do net o valor do neurônio é atualizado através da função de ativação e finalmente, o valor de saída do neurônio é produzido através da função de saída.

A função de ativação é uma função que calcula a saída de um neurônio. A entrada que ela recebe representa a soma de todos os produtos das entradas e seus respectivos pesos doravante denominada "soma ponderada".

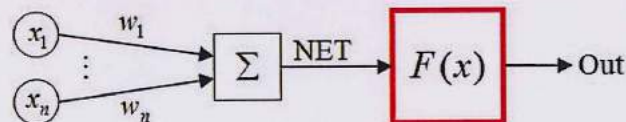


Figura 9: Modelo de neurônio artificial com função de ativação em destaque

Fonte: <https://www.mql5.com/pt/articles/497>

Os estados futuros de um neurônio são afetados pelo estado atual e pelo valor da rede de entrada. Este tipo de neurônio que possui "memória" é conhecido como neurônio dinâmico.

$$x_{t+1} = fa(x_t, net_t) \quad (9)$$

Por outro lado, considerando a função como constante, teremos neurônios que não possuem "memória", ou seja, o estado atual é igual aos estados anteriores e, portanto, o neurônio é considerado como "neurônio estático".

$$x_{t+1} = fa(net_t) \quad (10)$$

Essencialmente, qualquer função contínua e monotônica crescente, tal que $x \in R$ e $y(x) \in [-1, 1]$, pode ser utilizada como função de saída na modelagem neural. Existem uma série de funções mais comumente utilizadas como funções de saída dos neurônios. Estas funções são:

- A Função Degrau;

$$Out = \begin{cases} 0, & NET < \theta \\ 1, & NET \geq \theta \end{cases} \quad (11)$$

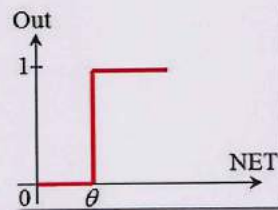


Figura 10: Gráfico Função Degrau

Fonte: <https://www.mql5.com/pt/articles/497>

- A Função Sigmoidal ou Logística;

$$Out(Net) = \frac{1}{1 + e^{-Net}} \quad (12)$$

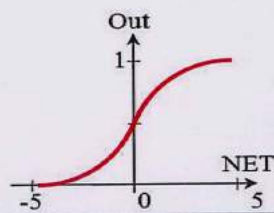


Figura 11: Gráfico Função Sigmoidal

Fonte: <https://www.mql5.com/pt/articles/497>

- A Função Tangente Hiperbólica;

$$Out(Net) = \tanh(kNet) = \frac{(1 - e^{-kNet})}{(1 + e^{-kNet})} \quad (13)$$

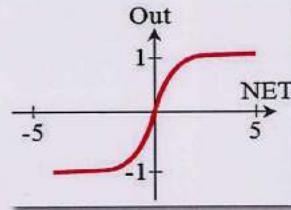


Figura 12: Gráfico Função Tangente Hiperbólica

Fonte: <https://www.mql5.com/pt/articles/497>

2.4 Treinamento da rede

Uma das habilidades mais importantes das redes neurais e que as torna tão interessantes é a capacidade de melhorar seu desempenho por meio da retroalimentação de seus sinais de resposta permitindo escolher a melhor combinação a um conjunto de entrada. Isso ocorre através do treinamento da rede que consiste no ajuste dos pesos da RNA.

O aprendizado das redes neurais utilizada neste trabalho é do tipo supervisionado em que cada resposta da rede neural é oferecida como feedback do valor que ela deveria ter dado como resposta. Desse modo, a rede pode ajustar seus pesos para aproximar-se mais da resposta esperada.

O treinamento de uma RNA é composto por vários ciclos ou iterações onde um ciclo ocorre quando todos os exemplos de treinamento foram mostrados a ela.

O ajuste dos pesos da rede neural pode ocorrer de forma online quando um exemplo é calculado por ela ou é feito quando um ciclo termina sendo assim classificado como batch, também conhecido como em lote.

Durante o treinamento o aprendizado da rede é guiado por uma função objetivo que geralmente utiliza a função entropia cruzada binária. A cada interação o algoritmo de treinamento ajustará os pesos da rede neural para se adequar a função objetivo.

2.4.1 Algoritmo de treinamento

Existem diversos algoritmos de treinamento para uma RNA que se diferenciam apenas no critério de atualização de pesos. Em linhas gerais, as entradas são processadas normalmente pela RNA e seu resultado é comparado com o resultado esperado e o erro cometido pela rede na fase *forward*. Após isso, o gradiente do erro é calculado. O valor do gradiente é medido como o produto do valor da derivada da função de ativação multiplicado pelo erro, e isso é feito no sentido contrário ao usual da propagação do sinal ocorrendo como fase *backward*. Esse valor é então usado de algum modo para ajustar os pesos do algoritmo de treinamento.

- Algoritmo *backpropagation*

Um dos mais antigos e tradicionais métodos de treinamento para redes *feedforward* que utiliza em conjunto com o gradiente um parâmetro nomeado taxa de aprendizado que indicará quanto o gradiente vai afetar a atualização do peso.

Um dos problemas desse algoritmo é a possibilidade de ficar preso em mínimos locais o que pode levar a convergência prematura. Por esse motivo é utilizado um outro parâmetro chamado de momento cuja função é colocar inercia na atualização dos pesos para evitar esse tipo de problema.

2.4.2 Normalização de Dados

A normalização de dados é o processo pelo qual todos os dados são normalizados sendo reduzido aos limites $[0,1]$ ou $[-1,1]$, garantindo que dados de ordem de grandezas diferentes possam ser comparados.

Caso a normalização não seja realizada os dados de entrada terão um efeito adicional sobre o neurônio, levando a decisões erradas.

A normalização pode ser realizada pela seguinte fórmula:

$$y(x) = \frac{(b - a)(x - x_{min})}{x_{max} - x_{min}} + a \quad (14)$$

Onde: y é o valor normalizado; x o valor a ser normalizado; x_{min} e x_{max} o valor mínimo e máximo do conjunto, respectivamente; a e b o menor e o maior valor possível para y , respectivamente.

2.4.3 Critérios de parada

A cada repetição o resultado calculado pela RNA vai se aproximando dos valores do conjunto de treinamento. Portanto, enquanto o treinamento continuar é provável que o erro do conjunto de treinamento continue a diminuir, mas isso pode afetar a capacidade de generalização da RNA. Então, precisa-se estabelecer um ou vários critérios de parada para a fase de treinamento.

Dois critérios de paradas:

1. Taxa de erro desejada: a taxa de erro desejada é definida e quando é alcançada o treinamento finaliza e a rede está treinada.
2. Número máximo de iterações: é definida por quantas iterações a rede deve ser treinada. Quando chegar na iteração definida, o treinamento acaba.

Uma combinação dos dois casos pode ser aplicada para ter um critério mais robusto, mas existe uma desvantagem do treinamento parar antes ou depois do ideal. Prolongar demais o treinamento poderá provocar um super ajustamento da rede chamado *overfitting* que a fará ter um desempenho satisfatório apenas sobre o conjunto de treinamento, pois ela memorizará esses dados. Por outro lado, caso o treinamento pare antes do ideal, a RNA terá um desempenho abaixo do que poderia ser obtido que é conhecido como *underfitting*.

Para evitar esses problemas existe o critério de parada conhecido como validação cruzada onde além dos dados de treinamento é considerado um outro conjunto de dados, onde as informações devem ser estatisticamente representativas, conforme pode ser analisado na Figura 13.



Figura 13: Critério de Parada

Fonte: <http://eventos.abrh.org.br/xivsrhne/apresentacoes/10266.pdf>

Critérios de funcionamento:

1. No final de cada iteração é calculado o erro do conjunto de treinamento e do conjunto de validação cruzada.
2. Os valores dos erros de ambos os conjuntos são comparados. Enquanto a rede estiver aprendendo corretamente o valor das taxas de erro decrescerão juntas e o treinamento deve continuar.
3. Quando o erro do conjunto de validação cruzada passar a aumentar enquanto a do conjunto de treinamento continuar a diminuir, é o sinal de que a rede agora está sofrendo um super ajustamento e o treinamento deve ser parado.

2.5 Classificação da RNA

Classificação é um tipo de aprendizagem supervisionada onde a saída é um valor discreto podendo ser binário quando o conjunto de resultados possíveis são compostos pelo par de valores {0, 1}.

2.5.1 Função Custo

A função de perda/custo (loss) será a entropia cruzada binária (Binary Cross-Entropy), dada pela seguinte fórmula:

$$J(z) = -\frac{1}{N} \sum_{i=1}^N y_i * \log(\omega_i) + (1 - y_i) * \log(1 - \omega_i) \quad (15)$$

Onde y representa a saída real e ω representa a saída predita pela rede.

2.5.2 Otimizadores

Um fator importante a ser escolhido numa rede neural é o otimizador, Adaptive moment estimation (Adam) é um método para ajuste de parâmetros proposto por Kingma and Ba (2014). Esse método utiliza as médias móveis dos gradientes e dos quadrados dos gradientes para atualizar os parâmetros. Essas médias são inicializadas como zero, e são calculadas como:

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * g_t \quad (16)$$

$$v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * g_t^2 \quad (17)$$

Onde m_t é o vetor das médias dos gradientes no tempo t , v_t é o vetor das médias dos quadrados dos gradientes no tempo t , g_t é o gradiente da função de custo, β_1 é o peso atribuído à média dos gradientes e β_2 é o peso atribuído à média dos quadrados dos gradientes.

Os parâmetros padrão para o otimizador pode ser descrito conforme código a seguir:

```
# pass optimizer by name: default parameters will be used
model.compile(loss='categorical_crossentropy', optimizer='adam')
```

A otimização de Adam é um método de descida gradiente estocástico que se baseia na estimativa adaptativa de momentos de primeira e segunda ordem, conforme código abaixo:

```
tf.keras.optimizers.Adam(
    learning_rate=0.001,
    beta_1=0.9,
    beta_2=0.999,
    epsilon=1e-07,
    amsgrad=False,
    name="Adam",
    **kwargs
)
```

De acordo com Kingma et al. 2014, o método é " computacionalmente eficiente, tem pouca necessidade de memória, invariante para o reescalonamento diagonal de gradientes e é adequado para problemas que são grandes em termos de dados/parâmetros ".

2.5.3 Métricas

Após o treinamento da rede os diferentes métodos de aprendizagem deverão ser avaliados para que possam ser comparados.

A acurácia representa o índice de acerto no conjunto de dados. Ou seja, ela verifica onde o algoritmo acertou na avaliação: Verdadeiros Positivos (TP) e Verdadeiros Negativos (TN) dividido pelo somatório dos Falsos Positivos (FP), Falsos Negativos (FN), Verdadeiros Positivos (TP) e Verdadeiros Negativos (TN).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (18)$$

A acurácia só pode ser considerada uma métrica válida quando o banco de dados da série temporal for balanceado. Ou seja, quando a distribuição das classes (sejam 2 ou mais) for consideravelmente uniforme.

Capítulo 3

Trabalhos relacionados

(Renato Bruni, 2017), apresentou classificação binária do índice da bolsa de valores italiana com algoritmo de inteligência artificial utilizando como rótulos os indicadores técnicos. A base de dados deste artigo despertou o interesse em quantificar resultados já que o autor não fez a aplicação da rede neural. Após o tratamento da RNA MLP utilizando o algoritmo *Backpropagation* foi obtido um resultado de acurácia de 52,50% e um erro médio de 3,93%.

(KIM, 2003), realizou previsão do índice de preços de ações aplicando a técnica de Inteligência Artificial Suporte Vector Machine (SVM), referenciando uma comparação entre os algoritmos SVM com Algoritmos BackPropagation (BP) e Resolução de Casos Baseados em vizinhança (CBR), tendo alcançado acurácia de 57,83%, 54,73% e 51,97%, respectivamente. O resultado alcançado pelo SVM foi similar ao resultado da rede neural mlp de duas camadas aplicada neste trabalho.

(Soni, & Sneha, & Shailendra, Shrivastava, 2010), utilizou a combinação de três algoritmos de aprendizado de máquina supervisionado, Árvore de Decisão de Classificação/Regressão (CART), Análise Linear Discriminante (LDA) e Análise Quadrática Discriminante (QDA). Os resultados experimentais e a seção de comparação de desempenho mostram que a taxa de acurácia da árvore de classificação e regressão é de apenas 56,11%, enquanto a LDA e a QDA mostram 74,26% e 76,57%, respectivamente. A árvore de classificação e regressão - CART teve um resultado similar ao aplicado neste trabalho com ligeira vantagem para a RNA MLP enquanto com a LDA e a QDA obtiveram melhores desempenhos do que a técnica praticada neste trabalho.

(Sezer, Omer & Ozbayoglu, Murat, 2018), executou a conversão de dados tabulares convencionais ou de séries temporais em imagem e treinamento de um modelo de classificação, atingindo acurácia de 87,33% sugerindo melhorias com

otimizações do resultado. O trabalho referenciado utilizou a junção de duas áreas da computação moderna quando aplicou o processamento digital de imagens e *Deep Learning*. A técnica realizada obteve melhor desempenho do que a praticada neste trabalho já que não foi possível aplicar essas mesmas técnicas devido ao tempo de aplicação e indisponibilidade do algoritmo de processamento digital de imagem.

(Penteado, 2003), Contribui com evidências que a análise gráfica pode ser aplicada para previsão de preços de ações no mercado financeiro brasileiro. Em seu estudo concluiu que a análise gráfica apresentou uma acurácia de 75,2% verdadeiros contra 24,8% de detecção de sinais falsos.

(Lima, 2016), contribuiu com o objetivo a utilização de Mineração de Dados (Mineração de Opinião) para previsão de um preço futuro na bolsa de valores. No estudo foi apresentado o teste dos algoritmos de Naive Bayes e SVM que tiveram instâncias corretamente classificadas num percentual de 35,29% e 47,05% respectivamente. Quando associado ao sentimento extraído do twitter vemos que a SVM sobe para 70,58% de acurácia, tornando-se importante fator a ser considerado nas previsões de ações.

Capítulo 4

Metodologia e Resultados

Para criar os sinais de entrada numa rede RNA é necessário criar algum parâmetro de saída para o treinamento supervisionado.

4.1 Tendência e sinal de negociação

Para o desenvolvimento desse trabalho foram criadas categorias de compra, venda e retenção sendo utilizada a Média Móvel Simples nas últimas 15 etapas (SMA15) para classificar o movimento do mercado de ações como ascendente (tendência de alta) ou descendente (tendência de baixa) da seguinte forma:

- Tendência = 'para cima': se o preço de fechamento > SMA15 e SMA15 estiver subindo nos últimos 5 dias.
- Tendência = 'baixa': se o preço de fechamento < SMA15 e SMA15 estiver caindo nos últimos 5 dias.
- Tendência = 'retenção': Caso contrário. A tendência é usada apenas para poder calcular o sinal de negociação (TS).

Um sinal de negociação foi usado como saída de treinamento para a RNA.

- TS = TSup: quando a última ocorrência da tendência foi 'up'
- TS = TSdown: quando a última ocorrência da tendência foi 'inativa'

TSup e TSdown são dados por:

$$TSup = \frac{(close(t) - closeMin)}{(closeMax - closeMin) * 0,5} + 0,5 \quad (19)$$

$$TS_{down} = \frac{(close(t) - closeMin)}{(closeMax - closeMin) * 0,5} \quad (20)$$

Onde:

$$closeMin = Min(close(t), close(t + 1), close(t + 2)) \quad (21)$$

$$closeMax = Max(close(t), close(t + 1), close(t + 2)) \quad (22)$$

4.2 Decisão de Negociação

Momento de determinar a tendência prevista (Trendpred) com base no sinal de negociação previsto (TSpred):

- Trendpred = 'up': se TSpred > 0.5.
- Trendpred = 'down': se TSpred ≤ 0.5.

Foi determinado comprar, manter ou vender de acordo com:

- COMPRAR: se a tendência do dia seguinte = 'up'
- VENDER: se a tendência do dia seguinte = 'down'
- HOLD: se não existir decisão de compra ou venda

4.3 Indicadores técnicos escolhidos

Os indicadores técnicos utilizados foram:

- SMA [15] – Média móvel simples
- MACD [26, 12, 9] – Média móvel convergente/divergente
- %K [15, 3] – Estocástico
- %D [15, 3] – Estocástico

- RSI [15] – Índice de força relativa
- %R [15] – De Larry William

A escolha destes indicadores se deu por conseguir respostas computacionais mais rápidas além de evitar sombreamento dos resultados. Após algumas simulações foi possível verificar que outros indicadores não trariam resultados mais precisos dessa forma apenas aumentaria a complexidade de dados a serem analisados.

4.4 Processamento dos dados

A base de dados utilizada nos experimentos foi retirada de uma série temporal do índice da bolsa de Valores do Financial Times (FTSE MIB) que é o principal indicador para os mercados acionários italianos.

É composto pelas 40 classes de ações mais negociadas na bolsa e captura aproximadamente 80% da capitalização do mercado doméstico.

Com esses índices, para cada dia de negociação entre 20 de abril de 2010 e 12 de julho de 2016 foi fornecido o preço de abertura, máximo, mínimo e fechamento e a classificação de acordo com o resultado do TS (compra, venda e mantém).

Cada registro de dados corresponde a um dia de negociação das ações na bolsa.

A base de dados utilizada foi capturada do artigo (*Stock Market Index Data and indicators for Day Trading as a Binary Classification problem*) de Renato Bruni, tendo sido escolhida após identificação de tendência de queda, ou seja, que o valor do primeiro fechamento é superior ao do último fechamento para simular momentos de crise globais como os vividos durante a atual crise do COVID19.

O algoritmo da RNA tem como objetivo atuar como método de segurança do investimento aplicado ainda que o momento seja adverso.

Os conjuntos de dados são fornecidos no formato CSV que pode ser aberto com o MS Excel ou como arquivo de texto.

4.5 Configurações da rede

Para esse experimento apresenta-se uma rede neural de classificação, tendo em vista que o resultado esperado de saída da rede será classificador de apoio à decisão de negociação de ações nas bolsas de valores.

O problema da classificação dos dados é a atribuição de rótulos aos registros de acordo com um critério aprendido automaticamente de um conjunto de treinamento, ou seja, um conjunto de registros que já possuem uma classe.

A classificação é uma tarefa muito importante de mineração de dados e muitos algoritmos de classificação estão disponíveis. Foi atribuído a classe a cada registro para que qualquer parte do conjunto de dados possa ser usada como conjunto de treinamento.

Após essa fase de aprendizado, o algoritmo de classificação será capaz de prever a classe para o restante dos registros. A precisão dessa previsão pode ser calculada comparando-a com a classe real que é a fornecida no conjunto de dados.

A classe atribuída a cada registro diário é 1 se o dia subsequente for favorável à negociação intra-dia e 0, caso contrário.

Favorável à negociação intra-dia significa que o aumento entre o preço de abertura e o preço de fechamento do mesmo dia é grande o suficiente para obter lucro comprando pelo preço de abertura e vendendo pelo preço de fechamento.

Foi selecionado um limiar de 0,3% para previsão do dia seguinte o que deve proporcionar uma oportunidade razoável de lucro.

Dessa forma a previsão de compra e venda permitiria realizar negociações intra-dia no dia seguinte ou poderia ser usada para definir estratégias de negociação entre dias.

Os dados foram tratados de forma que os atributos nulos fossem substituídos pelos dados do dia seguinte.

4.5.1 Treinamento

Os dados são um ativo valioso e fazem parte de toda a RNA. Se houver divisão dos dados usando como parâmetro a função *train_test_split*, só poderá ser treinado o modelo com a parte reservada para o treinamento a cada ciclo, conforme Figura 14.

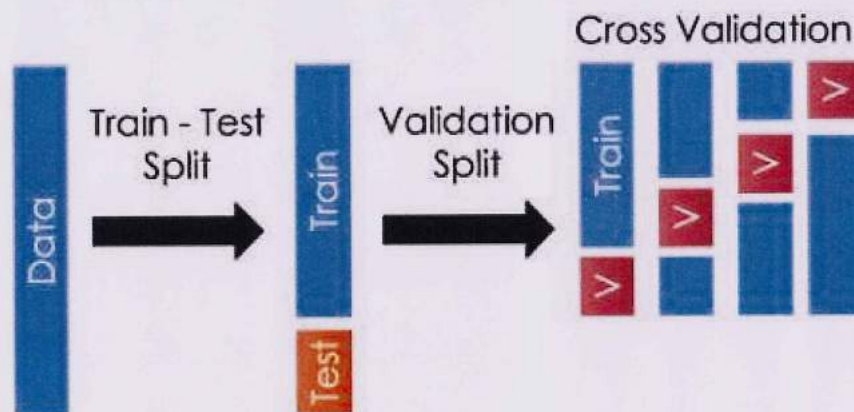


Figura 14: Cross Validation

Fonte: <https://medium.com/@hi.martinez/train-test-split-cross-validation-you-b87f662445e1>

A RNA melhora sua performance com o aumento de dados no treinamento e uma solução para superar esse problema é a validação cruzada que também é útil para medir o desempenho de um modelo com mais precisão.

Com a validação cruzada o conjunto de dados é dividido em n divisões. A divisão $N-1$ é usada para treinamento e a divisão restante é usada para teste. O modelo é executado em todo o conjunto de dados n vezes e, a cada vez, uma divisão diferente é usada para o teste. Dessa forma, são utilizados todos os pontos de dados para treinamento e teste.

A RNA utilizada possui a entrada dos 6 indicadores técnicos (rótulos) com treinamento de configuração:

```
# evaluate model with Baseline dataset
estimator = KerasClassifier(build_fn=create_baseline, epochs=100, batch_size=1, verbose=0)
kfold = StratifiedKFold(n_splits=2, shuffle=True)
results = cross_val_score(estimator, X_std, encoded_Y, cv=kfold)
print("Baseline: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))
Baseline: 56.92% (0.81%)
```


- KFold divide o conjunto de dados em k dobras e se *shuffle* for definido como *False* as dobras consecutivas serão a versão deslocada da dobra anterior. Esse modelo divide os dados em "lotes" do tamanho "*batch_size*" e repetidamente iterando em todo o conjunto de dados por um determinado número de "épocas".
- StratifiedKFold leva a validação cruzada um passo adiante. A distribuição de classes no conjunto de dados é preservada nas divisões de treinamento e teste.
- Cross_val_score pega o conjunto de dados e aplica validação cruzada para dividir os dados. Em seguida, treina um modelo usando o estimador especificado.

Neste trabalho a métrica utilizada foi a acurácia por ser a única métrica possível de ser comparada nos demais trabalhos relacionados no capítulo 3.

4.6 Tratamento e normalização dos dados

Os dados foram filtrados para verificar e corrigir as informações ausentes ou imprecisas.

Os Indicadores que são calculados na rede neural utilizam o histórico da série temporal e só estarão disponíveis quando observados na série temporal seguinte.

A classe não está disponível para o último registro. Esses dados ausentes são codificados como 'N'. Nenhum outro dado ausente aparece no conjunto de dados. A limpeza de dados é realmente uma questão importante para dados semelhantes (R. Bruni, 2005).

Os dados fornecidos podem ser usados para testar a eficácia da análise técnica na previsão da tendência ou para testar a precisão dos algoritmos de classificação.

As técnicas de normalização utilizadas foram MinMaxScaler, StandardScaler.

```
sklearn.preprocessing import StandardScaler, MinMaxScaler
```

4.7 Resultados

O resultado da MLP deste trabalho foi comparada a outros trabalhos e demonstrou sua relevância conforme quadro abaixo:

Tabela 1 – Resultados – Vantagens e Desvantagens

MODELO	VANTAGEM	DESVANTAGEM	RESULTADO ACURÁCIA
MLP implementada	Grande capacidade de hibridização com os demais modelos	Não é rede recorrente (neurônios servindo como entradas)	56,41%
Algoritmo BP (Renato Bruni, 2017)	Implementação simplificada da MLP	Resultado inferior a MLP apresentada neste trabalho. O autor não implementou, sendo necessário simular a rede.	52,5%
Algoritmo SVM (Kim, 2003)	Busca melhor classificação, utilizando Inversão de Matrizes	Algoritmo não supervisionado, Alta complexidade, não funciona em conjunto de dados com ruídos	57,83%
Algoritmo BP (Kim, 2003)	Implementação simplificada da MLP	Resultado inferior a MLP implementada neste trabalho	54,73%

MODELO	VANTAGEM	DESVANTAGEM	RESULTADO ACURÁCIA
Algoritmo CBR (Kim, 2003)	Algoritmo de análise de vizinhança.	Processo não parametrizável	51,97%
Algoritmo CART	Fácil implementação	limitações na resolução de processos não lineares.	56,11%
Algoritmo LDA	Utiliza estocástica e processos não lineares	Somente estatístico, não é escalável o resultado (não existe uma padronização de aprendizagem)	74,26%
Algoritmo QDA	Utiliza estocástica e processos não lineares	Somente estatístico, não é escalável o resultado (não existe uma padronização de aprendizagem)	76,57%
Rede Neural convoluída em Imagem	Rede profunda, melhor precisão e acurácia dentre os trabalhos	Alta complexidade, Necessita de Processamento Digital de Imagens	87,33%
Naive Bayes (Lima, 2016)	-	Resultado inferior a MLP apresentada neste trabalho.	35,29%

MODELO	VANTAGEM	DESVANTAGEM	RESULTADO ACURÁCIA
SVM, (Lima, 2016)	busca melhor classificação, utiliza Inversão de Matrizes	Resultado inferior a MLP apresentada neste trabalho.	47,05%
SVM com Análise de Sentimento, (Lima, 2016)	Utiliza técnica de mineração de dados para realizar análise de sentimento do mercado.	Tempo de processamento	70,58%

Fonte: Elaborada pelo Autor

Capítulo 5

Considerações Finais

5.1 Conclusão

Um dos propósitos do presente trabalho foi investigar possíveis algoritmos que pudessem auxiliar na predição de preços no mercado de ações.

A pesquisa inicialmente mostrou a importância na definição dos indicadores que formaram a base de dados inicial. É considerável destacar que a base de dados deve ter uma distribuição cronológica, pois esta organização é incisiva para a previsibilidade da série.

Foi proposto uma MLP de treinamento com um algoritmo *backpropagation*, utilizando o otimizador Adam, implementado no padrão default do Keras, com função de *loss* “*Binary Cross-Entropy*”, contendo a função de ativação “*relu*” e na camada de saída “*sigmóide*”,

Os dados foram normalizados em um intervalo de 0 a 1 para uso da RNA. A acurácia final apresentada após o treinamento e simulação da rede foi de 56,41% e erro de 0,09%, apresentando um resultado no intervalo esperado para a rede comparando com outros trabalhos relacionados.

Com o resultado entre as aplicações com MLP, SVM e CART foi perceptível a relevância do algoritmo MLP com duas camadas escondidas que demonstrou um desempenho em nível igual ou superior as demais técnicas tradicionais.

Cabe ressaltar que as séries temporais podem ser delimitadas pelas técnicas de LDA, QDA e RNA Profundas Convoluídas, já que possuem maior precisão quando acontece grande variação de dados em curto espaço de tempo. Assim, apresentam um resultado superior para previsão de preços no mercado de ações.

Por fim, a pesquisa possibilitou a visibilidade de que a RNA MLP *Backpropagation* pode auxiliar na tomada de decisão na previsão de preços futuros de ações em bolsa de valores tendo apenas menor precisão quando comparada com àquelas que dispõem de outros recursos computacionais tal como processamento digital de imagens que abrange outra área da computação.

5.2 Sugestões de trabalhos futuros

1. Realizar abordagem sobre Rede Neural Recorrente como: Gated Recurrent Unit (GRU) e Long Short Term Memory (LSTM);
2. Utilizar a Rede Neural Convoluída em imagem;
3. Estudar os indicadores mais influentes (pesos sinápticos) para otimização da Rede;
4. Utilizar novas configurações de uma MLP;
5. Utilizar outras bases de dados e realizar testes estatísticos;
6. Realizar estudo de uma nova RNA de regressão para prever o preço de um ativo considerando a sistemática de stop utilizada pelo mercado de ações.
7. Utilizar OBV (On Balance Volume) para verificar a correlação do volume de negócios com os indicadores Wilians %R e IFR nos rótulos da RNA proposta.

Capítulo 6

Referências Bibliográficas

ABRANSOM, N. Information Theory and Coding. McGraw-Hill Book Company, 1963. 300 p.

ALEKSANDER, I. e MORTON, H. An Introduction to Neural Computing. International Thomson Computer Press, 1995. 490 p.7

AMORIM, M. C. Previsão de séries temporais usando séries exógenas e combinação de redes neurais aplicadas ao mercado financeiro. Dissertação de Mestrado. Centro de Informática, Universidade Federal de Pernambuco, 2008.

BAIRD, H.S. Document Image Defect Models. Structured Document Image Analysis, vol. 2, n. 3, p. 546-556, 1992.

BALDWIN, R.; WEDER DI MAURO, B. Economics in the Time of COVID-19. A VoxEU.org. CEPR Press Book, 2020.

BOVESPA e MB ASSOCIADOS. Desafios e Oportunidades para o Mercado de Capitais Brasileiro. São Paulo: Bovespa, 2000.

HAYKIN, S. Redes Neurais – 2ed.[S.1]: BOOKMAN COMPANHIA ED, 2001. ISBN 9788573077186.

HEATON, J. Programming Neural Networks With Encog 2. St. Louis, MO USA: Heaton Reseach, Inc, 2010.

H. M. SACHETIM. Análise técnica: Estudo da confiabilidade dos principais indicadores de análise técnica, aplicado as ações mais negociadas na bovespa no período de

1995 a 2005. Dissertação de Mestrado. Universidade Federal do Paraná, Setor de Ciências Sociais Aplicadas. Curitiba, 2006.

IMAI, N. et al. Report 3: transmissibility of 2019-nCoV. British Medical Journal Publishing Group, [S.I.], 2020.

Disponível em: <https://www.imperial.ac.uk/media/imperialcollege/medicine/sph/ide/gidafellowships/Imperial-2019-nCoV-transmissibility.pdf>. Acesso em: 2 abr. 2020.

KIM, K. jae. Financial time series forecasting using support vector machines. Neurocomputing, v. 55, n. 1, p. 307 – 319, 2003. Support Vector Machines.

Kingma, PD; Ba, J, “Adam: A Method for Stochastic Optimization”. arXiv 2014, arXiv: 1412.6980v9.

-LIMA, MILSON. Um Modelo para Predição de Bolsa de Valores Baseado em Mineração de Opinião. 2016. Programa de Pós Graduação em Engenharia de Eletricidade/ccet, Universidade Federal do Maranhão, São Luiz-MA, 2016.

Disponível:

https://tedebc.ufma.br/jspui/bitstream/tede/297/1/Dissertacao_MilsonLouseiroLima.pdf

LIN, Q. et al. A conceptual model for the coronavirus disease 2019 (COVID-19) outbreak in Wuhan, China with individual reaction and governmental action. International Journal of Infectious Diseases, [S.I.], v. 93, p. 211-216, 2020.

MELLO, C.A.B. Synthesis of Images of Historical Documents for Web Visualization, Proceedings of IEEE International Multi-Media Modelling Conference, 2004, Brisbane, AU.

MORENO, B. M. S.; SANA, C. P.; SILVA, M. R. Estudo da eficiência de indicadores de análise técnica: o uso de médias móveis (moving average) e estocástico. 2008. Faculdade de Ciências Econômicas e Administrativas de Presidente Prudente. 67. p. (Finanças) Presidente Prudente, 2008.

Disponível: <https://arxiv.org/abs/1412.6980v9> (acessado em 28 de setembro de 2020).

MORETTIN, P. A. e TOLOI, C. M. C. Análise de séries temporais, 2. ed. São Paulo: Edgar Blücher / ABE – Projeto Fisher, 2006.

OLIVEIRA, E. M. de J. Previsão da Série Temporal do Índice da Bolsa de Valores de São Paulo Usando Redes Neurais e Estatística. Dissertação (Dissertação de Mestrado) – Centro de Informática – Universidade Federal de Pernambuco, Recife-PE, 2001.

Penteado, M. A. de B. Uma avaliação estatística da análise gráfica no mercado de ações brasileiro à luz da teoria dos mercados eficientes e das finanças comportamentais. Dissertação (Dissertação de Mestrado) – Faculdade de Economia, Administração e Contabilidade – Departamento de Administração - Universidade de São Paulo, São Paulo-SP, 2003.

RIEDMILLER, M.; BRAUN, H. A direct adaptive method for faster backpropagation learning: the rprop algorithm. In: Neural Networks, 1993. IEEE International Conference on. [S.l.: s.n], 1993. P. 586-591 vol. 1.

Renato Bruni, Stock Market Index Data and indicators for Day Trading as a Binary Classification problem, Data in Brief, Volume 10, 2017, Pages 569-575, ISSN 2352-3409, <https://doi.org/10.1016/j.dib.2016.12.044>.

(<http://www.sciencedirect.com/science/article/pii/S2352340916308058>)

Sezer, Omer & Ozbayoglu, Murat. (2018). Algorithmic Financial Trading with Deep Convolutional Neural Networks: Time Series to Image Conversion Approach. Applied Soft Computing. 70. 10.1016/j.asoc.2018.04.024.

Silva, L. N. C. Análise e síntese de estratégias de aprendizado para redes neurais artificiais. 1998. 210 p. Dissertação (Mestrado em Engenharia Elétrica) – Faculdade de Engenharia Elétrica e Computação. Universidade Estadual de Campinas, Campinas, 1998.

Soni, & Sneha, & Shailendra, Shrivastava. (2010). Classification of Indian Stock Market Data Using Machine Learning Algorithms. International Journal on Computer Science and Engineering. 2.

UNIVERSIDADE DE SÃO PAULO. Instituto Astronômico e Geográfico. Anuário Astronômico, São Paulo, 1988. 279 p.

VALENÇA, M. Fundamentos das Redes Neurais. 2. Ed. Olinda – PE: Livro Rápido, 2010, 310p.

WALTER, Richard G .. Análise fundamentalista e avaliação de títulos: aspectos teóricos. Rev. adm. empres. , São Paulo, v. 14, n. 1, pág. 15-32, fevereiro de 1974. Disponível em <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0034-75901974000100003&lng=en&nrm=iso>. acesso em 02 de dezembro de 2020. <https://doi.org/10.1590/S0034-75901974000100003>.

WERBOS, P., Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. Ph.D. Thesis, Applied Mathematics, Howard University, Nov. 1974.

Apêndice A - Código Fonte da RNA

Código Inicial, sem escolha de um conjunto de indicadores.

```
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import StratifiedKFold
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
```

```
#load dataset
dataset = pd.read_csv("FtseMIB - Modif.csv", sep = ";", header = 0)
dataset = dataset.values
dataset.head()
```

	Class	Date	Opening Price	Closing Price	Maximum	Minimum	Var. %	Return	MOMENTUM5	EMA12	EMA26	MACD 12-26	EMA9 of MACD 12-26	MACD 12-26-9	ROI10	ROI20	RO
0	0	20100420	22924.41	23271.68	23271.68	22831.36	2.130	-1.000000	-1.0	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.0	-1.0	
1	0	20100421	23354.58	23045.05	23363.83	23000.61	-0.010	-0.009738	-1.0	23236.81385	23027.61692	209.196923	-1.000000	-1.000000	-1.0	-1.0	
2	1	20100422	23018.57	22616.50	23171.61	22547.24	-0.019	-0.018596	-1.0	23141.38095	22584.87562	556.505325	278.658604	277.846722	-1.0	-1.0	
3	0	20100423	22660.93	22726.00	22814.05	22469.54	0.005	0.004842	-1.0	23077.47619	22736.85572	340.620464	291.050976	49.569489	-1.0	-1.0	
4	0	20100426	22962.51	22783.27	23053.51	22714.55	0.003	0.002520	-1.0	23032.21370	22786.84033	245.373367	281.915454	-36.542087	-1.0	-1.0	

```
# Split inputs (X) and outputs (Y)
X = dataset[:,2:28].astype(float)
Y = dataset[:,0]
print(X)
print(Y)
```

```
[[ 2.29244100e+04  2.32716800e+04  2.32716800e+04 ... -1.00000000e+00
 -1.00000000e+00 -1.00000000e+00]
 [ 2.33545800e+04  2.30450500e+04  2.33638300e+04 ... -1.00000000e+00
 -1.00000000e+00 -1.00000000e+00]
 [ 2.30185700e+04  2.26165000e+04  2.31716100e+04 ... -1.00000000e+00
 -1.00000000e+00 -1.00000000e+00]
 ...
 [ 1.60663800e+04  1.60663800e+04  1.60876700e+04 ... -3.63734400e+01
 5.75620004e+02  5.05422854e+01]
 [ 1.62606400e+04  1.62606400e+04  1.62637400e+04 ... 1.61893678e+00
 5.60002861e+02  4.85865250e+01]
 [ 1.67210000e+04  1.67210000e+04  1.67678300e+04 ... 5.19712022e+01
 5.56230514e+02  5.31830167e+01]]
[0. 0. 1. ... 1. 1. 0.]
```

```
# Encode values as Integer
encoder = LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)
```

```
# baseline model
def create_baseline():
    # create model
    model = Sequential()
    model.add(Dense(26, input_dim=26, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    # Compile model
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model
```

```
# evaluate model with Baseline dataset
estimator = KerasClassifier(build_fn=create_baseline, epochs=100, batch_size=5, verbose=0)
kfold = StratifiedKFold(n_splits=10, shuffle=True)
results = cross_val_score(estimator, X, encoded_Y, cv=kfold)
print("Baseline: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))
```

Baseline: 48.06% (6.92%)

```
# evaluate baseline model with standardized dataset
estimators = []
estimators.append(('standardize', StandardScaler()))
estimators.append(('mlp', KerasClassifier(build_fn=create_baseline, epochs=100, batch_size=5, verbose=0)))
pipeline = Pipeline(estimators)
kfold = StratifiedKFold(n_splits=10, shuffle=True)
results = cross_val_score(pipeline, X, encoded_Y, cv=kfold)
print("Standardized: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))
```

```
# larger model
def create_larger():
    # create model
    model = Sequential()
    model.add(Dense(26, input_dim=26, activation='relu'))
    model.add(Dense(13, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    # Compile model
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model
estimators = []
estimators.append(('standardize', StandardScaler()))
estimators.append(('mlp', KerasClassifier(build_fn=create_larger, epochs=100, batch_size=5, verbose=0)))
pipeline = Pipeline(estimators)
kfold = StratifiedKFold(n_splits=10, shuffle=True)
results = cross_val_score(pipeline, X, encoded_Y, cv=kfold)
print("Larger: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))
```

Larger: 52.50% (3.93%)

Código Final, com escolha de um conjunto de indicadores, utilização do otimizador Adam e avaliação do grau de compra/manutenção/venda ao invés de compra/venda.

```
!pip install ta
import pandas as pd
import numpy as np
import math
import random
import os
import matplotlib.pyplot as plt
import copy
import ta
import pandas as pd
import numpy as np
import sklearn
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import StratifiedKFold
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.pipeline import Pipeline
```

Collecting ta

Downloading <https://files.pythonhosted.org/packages/90/ec/e4f5aea8c7f0f55f92b52ffbafa389ea82f3a10d9cab2760e40af34c5b3f/ta-0.5.25.tar.gz>
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from ta) (1.18.5)
Requirement already satisfied: pandas in /usr/local/lib/python3.6/dist-packages (from ta) (1.0.5)

Requirement already satisfied: python-dateutil<=2.6.1 in /usr/local/lib/python3.6/dist-packages (from pandas->ta) (2.8.1)

Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages (from pandas->ta) (2018.9)

Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (from python-dateutil>=2.6.1->pandas->ta) (1.15.0)

Building wheels for collected packages: ta

Building wheel for ta (setup.py) ... done

Created wheel for ta: filename=ta-0.5.25-cp36-none-any.whl size=24880 sha256=cda7ad8df3fbbec343bf3e1e9912c70347a59cf4b86b12f8d8fea18f5d5beb39
Stored in directory: /root/.cache/pip/wheels/2e/93/b7/cf649194508e53cee4145ffb949e9f26877a5a8dd12db9ed5b

Successfully built ta

Installing collected packages: ta

Successfully installed ta-0.5.25

```
df = pd.read_csv('FtseMIB - Modif2.csv', sep = ';', header = 0)
df.head()
```

	0	Date	Opening Price	Closing Price	Maximum	Minimum	Var. %	Return
0	0.0	20100623	20666.47	20609.03	20854.92	20372.10	-0.010	-0.009678315
1	0.0	20100624	20371.90	20358.92	20659.26	20205.37	-0.012	-0.012135942
2	1.0	20100625	20468.32	19942.39	20491.93	19825.97	-0.021	-0.020459337
3	0.0	20100628	19979.29	19961.74	20121.43	19734.59	0.001	0.000970295
4	1.0	20100629	20063.25	20130.61	20282.08	19790.33	0.009	0.008459683

```
ta.utils.dropna(df)
```

	0	Date	Opening Price	Closing Price	Maximum	Minimum	Var. %	Return
2	1.0	20100625	20468.32	19942.39	20491.93	19825.97	-0.021	-0.020459337
4	1.0	20100629	20063.25	20130.61	20282.08	19790.33	0.009	0.008459683
6	1.0	20100701	19297.85	19311.75	19562.34	19111.09	0.004	0.00392436
8	1.0	20100706	19114.29	19073.85	19256.91	18926.85	0.007	0.006867169
9	1.0	20100707	19126.59	18848.56	19126.59	18842.49	-0.012	-0.011811459
...
1542	1.0	20160628	15601.62	15601.62	15799.21	15286.86	-0.039	0.032974964
1543	1.0	20160629	15946.93	15946.93	16019.31	15754.04	0.033	0.022132958
1544	1.0	20160630	16197.78	16197.78	16250.33	15629.01	0.022	0.0157303
1549	1.0	20160707	15436.16	15436.16	15710.05	15391.02	-0.023	0.000802008
1550	1.0	20160708	16066.38	16066.38	16087.67	15356.04	0.001	0.040827512

636 rows x 8 columns

```
indicator_sma = ta.trend.SMAIndicator(close = df['Closing Price'], n = 9, fillna = True)
df['SMA'] = indicator_sma.sma_indicator()
df.head()
```

	0	Date	Opening Price	Closing Price	Maximum	Minimum	Var. %	Return	SMA
0	0.0	20100623	20666.47	20609.03	20854.92	20372.10	-0.010	-0.009678315	20609.030000
1	0.0	20100624	20371.90	20358.92	20659.26	20205.37	-0.012	-0.012135942	20483.975000
2	1.0	20100625	20468.32	19942.39	20491.93	19825.97	-0.021	-0.020459337	20303.446667
3	0.0	20100628	19979.29	19961.74	20121.43	19734.59	0.001	0.000970295	20218.020000
4	1.0	20100629	20063.25	20130.61	20282.08	19790.33	0.009	0.008459683	20200.538000

```
indicator_macd = ta.trend.MACD(close = df['Closing Price'], n_slow = 26, n_fast = 12, n_sign = 9, fillna = True)
df['MACD'] = indicator_macd.macd()
df.head()
```

	0	Date	Opening Price	Closing Price	Maximum	Minimum	Var. %	Return	SMA	MACD
0	0.0	20100623	20666.47	20609.03	20854.92	20372.10	-0.010	-0.009678315	20609.030000	0.000000
1	0.0	20100624	20371.90	20358.92	20659.26	20205.37	-0.012	-0.012135942	20483.975000	-19.951795
2	1.0	20100625	20468.32	19942.39	20491.93	19825.97	-0.021	-0.020459337	20303.446667	-68.583636
3	0.0	20100628	19979.29	19961.74	20121.43	19734.59	0.001	0.000970295	20218.020000	-104.360337
4	1.0	20100629	20063.25	20130.61	20282.08	19790.33	0.009	0.008459683	20200.538000	-117.730113

```
indicator_rsi = ta.momentum.RSIIndicator(close = df['Closing Price'], n = 15, fillna = True)
df['RSI'] = indicator_rsi.rsi()
df.head()
```

	0	Date	Opening Price	Closing Price	Maximum	Minimum	Var. %	Return	SMA	MACD	RSI
0	0.0	20100623	20666.47	20609.03	20854.92	20372.10	-0.010	-0.009678315	20609.030000	0.000000	100.000000
1	0.0	20100624	20371.90	20358.92	20659.26	20205.37	-0.012	-0.012135942	20483.975000	-19.951795	0.000000
2	1.0	20100625	20468.32	19942.39	20491.93	19825.97	-0.021	-0.020459337	20303.446667	-68.583636	0.000000
3	0.0	20100628	19979.29	19961.74	20121.43	19734.59	0.001	0.000970295	20218.020000	-104.360337	3.091129
4	1.0	20100629	20063.25	20130.61	20282.08	19790.33	0.009	0.008459683	20200.538000	-117.730113	24.820660


```
indicator_R = ta.momentum.WilliamsRIndicator(high = df['Maximum'], low = df['Minimum'], close = df['Closing Price'], lbp = 15, fillna = True)
df['%R'] = indicator_R.wr()
df.head()
```

	0	Date	Opening Price	Closing Price	Maximum	Minimum	Var. %	Return	SMA	MACD	RSI	%R
0	0.0	20100623	20666.47	20609.03	20854.92	20372.10	-0.010	-0.009678315	20609.030000	0.000000	100.000000	-50.927882
1	0.0	20100624	20371.90	20358.92	20659.26	20205.37	-0.012	-0.012135942	20483.975000	-19.951795	0.000000	-76.360557
2	1.0	20100625	20468.32	19942.39	20491.93	19825.97	-0.021	-0.020459337	20303.446667	-68.583636	0.000000	-88.685553
3	0.0	20100628	19979.29	19961.74	20121.43	19734.59	0.001	0.000970295	20218.020000	-104.360337	3.091129	-79.724724
4	1.0	20100629	20063.25	20130.61	20282.08	19790.33	0.009	0.008459683	20200.538000	-117.730113	24.820660	-64.651487

```
indicator_stochastic = ta.momentum.StochasticOscillator(high = df['Maximum'], low = df['Minimum'], close = df['Closing Price'], n = 15, d_n = 3, fillna = True)
df['%K'] = indicator_stochastic.stoch()
df['%D'] = indicator_stochastic.stoch_signal()
df.head()
```

	0	Date	Opening Price	Closing Price	Maximum	Minimum	Var. %	Return	SMA	MACD	RSI	%R	%K	%D
0	0.0	20100623	20666.47	20609.03	20854.92	20372.10	-0.010	-0.009678315	20609.030000	0.000000	100.000000	-50.927882	49.072118	49.072118
1	0.0	20100624	20371.90	20358.92	20659.26	20205.37	-0.012	-0.012135942	20483.975000	-19.951795	0.000000	-76.360557	23.639443	36.355780
2	1.0	20100625	20468.32	19942.39	20491.93	19825.97	-0.021	-0.020459337	20303.446667	-68.583636	0.000000	-88.685553	11.314447	28.008669
3	0.0	20100628	19979.29	19961.74	20121.43	19734.59	0.001	0.000970295	20218.020000	-104.360337	3.091129	-79.724724	20.275276	18.409722
4	1.0	20100629	20063.25	20130.61	20282.08	19790.33	0.009	0.008459683	20200.538000	-117.730113	24.820660	-64.651487	35.348513	22.312745

```
dataset = df.values
X = dataset[:,8:15].astype(float)
Y = dataset[:,0]
print(X)
print(Y)
```

```
[[ 2.06090300e+04  0.00000000e+00  1.00000000e+02 -5.09278820e+01
  4.90721180e+01  4.90721180e+01]
 [ 2.04839750e+04 -1.99517949e+01  0.00000000e+00 -7.63605573e+01
  2.36394427e+01  3.63557803e+01]
 [ 2.03034467e+04 -6.85836365e+01  0.00000000e+00 -8.86855532e+01
  1.13144468e+01  2.80086691e+01]
 ...
 [ 1.58623233e+04 -2.48028137e+02  4.84652521e+01 -6.44269606e+01
  3.55730394e+01  2.11849089e+01]
 [ 1.59355478e+04 -2.04439071e+02  5.01242702e+01 -5.78390844e+01
  4.21609156e+01  3.06448495e+01]
 [ 1.60215556e+04 -1.31234429e+02  5.38930020e+01 -4.22270454e+01
  5.77729546e+01  4.51689699e+01]]
[0.0 0.0 1.0 ... 1.0 0.0 nan]
```

```
# Encode values as Integer
encoder = LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)
```

```
scaler = MinMaxScaler()
scaler.fit(X)
X_std = scaler.transform(X)
print(X_std)
```

```
[[0.70904825 0.64974348 1.          0.49072118 0.49072118 0.48980076]
 [0.69743463 0.63782645 0.          0.23639443 0.23639443 0.36168624]
 [0.68066931 0.60877908 0.          0.11314447 0.11314447 0.27759079]
 ...
 [0.26823075 0.50159846 0.48465252 0.35573039 0.35573039 0.20884279]
 [0.27503096 0.52763382 0.5012427  0.42160916 0.42160916 0.30414977]
 [0.28301834 0.57135831 0.53893002 0.57772955 0.57772955 0.45047733]]
```

```
# baseline model
def create_baseline():
    # create model
    model = Sequential()
    model.add(Dense(2, input_dim=6, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    # Compile model
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model
```

```
# evaluate model with Baseline dataset
estimator = KerasClassifier(build_fn=create_baseline, epochs=100, batch_size=1, verbose=0)
kfold = StratifiedKFold(n_splits=2, shuffle=True)
results = cross_val_score(estimator, X_std, encoded_Y, cv=kfold)
print("Baseline: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_split.py:667: UserWarning:
  % (min_groups, self.n_splits)), UserWarning)
Baseline: 56.92% (0.81%)
```

```
# evaluate baseline model with standardized dataset
estimators = []
estimators.append(('standardize', StandardScaler()))
estimators.append(('mlp', KerasClassifier(build_fn=create_baseline, epochs=100, batch_size=1, verbose=0)))
pipeline = Pipeline(estimators)
kfold = StratifiedKFold(n_splits=2, shuffle=True)
results = cross_val_score(pipeline, X_std, encoded_Y, cv=kfold)
print("Standardized: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_split.py:667: UserWarning: The least populated class
  % (min_groups, self.n_splits)), UserWarning)
Standardized: 57.89% (0.16%)
```



```
# smaller model
def create_smaller():
    # create model
    model = Sequential()
    model.add(Dense(9, input_dim=6, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    # Compile model
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model
estimators = []
estimators.append(('standardize', StandardScaler()))
estimators.append(('mlp', KerasClassifier(build_fn=create_smaller, epochs=100, batch_size=1, verbose=0)))
pipeline = Pipeline(estimators)
kfold = StratifiedKFold(n_splits=2, shuffle=True)
results = cross_val_score(pipeline, X_std, encoded_Y, cv=kfold)
print("Smaller: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_split.py:667: UserWarning: The least populated
% (min_groups, self.n_splits)), UserWarning)
Smaller: 56.41% (0.09%)
```

```
# larger model
def create_larger():
    # create model
    model = Sequential()
    model.add(Dense(9, input_dim=6, activation='relu'))
    model.add(Dense(3, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    # Compile model
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model
estimators = []
estimators.append(('standardize', StandardScaler()))
estimators.append(('mlp', KerasClassifier(build_fn=create_larger, epochs=100, batch_size=5, verbose=0)))
pipeline = Pipeline(estimators)
kfold = StratifiedKFold(n_splits=2, shuffle=True)
results = cross_val_score(pipeline, X, encoded_Y, cv=kfold)
print("Larger: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_split.py:667: UserWarning: The least populated
% (min_groups, self.n_splits)), UserWarning)
Larger: 54.41% (0.42%)
```