Aplicando Algoritmos de Aprendizado de Máquina em Python no Mercado Financeiro

1st Guido Margonar Moreira
Engenharia de Computação
Universidade Tecnológica Federal do Paraná (UTFPR)
Apucarana, Brasil
guidomoreira@alunos.utfpr.edu.br

2st João Pedro Padoan
Engenharia de Computação
Universidade Tecnológica Federal do Paraná (UTFPR)
Apucarana, Brasil
padoan@alunos.utfpr.edu.br

3st Cristian Andre Sanches
Engenharia de Computação
Universidade Tecnológica Federal do Paraná (UTFPR)
Apucarana, Brasil
cristiansanches@alunos.utfpr.edu.br

***Resumo*—Operações envolvendo a compra e venda de ações no mercado financeiro são cada vez mais recorrentes em nosso mundo moderno. Desde trabalhadores comuns até grandes empresas investem no mercado financeiro e podem lucrar ou perder dinheiro nessas transações, podendo - em casos extremos - até levá-los à falência ou ao enriquecimento rápido. Neste trabalho, aplicamos técnicas de Machine Learning para a avaliação de operações de compra e venda de ações no mercado financeiro. Desta forma esperamos obter resultados melhores (isto é, lucros maiores e perdas menores) comparados à operações envolvendo apenas a intuição e raciocínio humanos.**

Operations involving the purchase and sale of shares in the financial market are increasingly recurrent in our modern world. Ordinary workers and large companies invest in the financial market and can profit or lose money in these transactions, sometimes - in extreme cases - leading them to bankruptcy or, in the luckier cases, to quick enrichment. In this work, we apply Machine Learning techniques to evaluate stock purchase and sale operations in the financial market. In this way we expect to obtain better results (that is, greater profits and smaller losses) compared to operations involving only human intuition and reasoning.

***Index Terms*—metatrader5, python, estatística, tensorflow**

I. Introdução

Operações envolvendo a compra e venda de ações no mercado financeiro são cada vez mais recorrentes em nosso mundo moderno. Desde trabalhadores comuns até grandes empresas investem no mercado financeiro e podem lucrar ou perder dinheiro nessas transações, podendo - em casos extremos - até levá-los à falência ou ao enriquecimento rápido. Neste trabalho, aplicamos técnicas de *Machine Learning* para a avaliação de operações de compra e venda de ações no mercado financeiro. Desta forma esperamos obter resultados melhores (isto é, lucros maiores e perdas menores) comparados à operações envolvendo apenas a intuição e raciocínio humanos.

II. Objetivo

O objetivo deste trabalho é aplicar as técnicas de inteligência artificial que aprendemos ao longo da disciplina de Sistemas Inteligentes 1 para a avaliação da assertividade de operações de compra e venda de ações no mercado financeiro. Para isso, aplicamos algoritmos de *Machine Learning* - e especificamente algoritmos envolvendo redes neurais - para a realização de operações financeiras, usando como fonte de aprendizado o histórico de ações de empresas de grande porte atuantes no mercado, e comparamos os diferentes algoritmos quanto ao seu desempenho e testamos a sua eficiência em relação aos comportamentos mais comuns no mercado (como o *Buy and Hold*, o comprar e segurar de ações), tanto a longo quanto a curto e médio prazo.

Para a implementação de nossa proposta usaremos a plataforma de negociação *Meta Trader 5* como base de dados. A plataforma permite a análise de dados de ativos financeiros e a simulação de transações no mercado. Ela também permite que importemos os seus dados para que os usemos em nosso próprio programa. Como linguagem de programação para a implementação dos algoritmos usaremos o *Python*.

III. Metodologia

Para a implementação do algoritmo capaz de realizar previsões no preço de ações foram escolhidos o programa *Metatrader 5*, software dedicado a esse tipo de atividade contando com bancos de dados e ferramentas, que será usado em conjunto da linguagem de programação *Python*, com os módulos necessários para obtenção, visualização e análise dos dados(como *matplotlib*, *pandas*, *scikit* e *tensorflow*). Para o treinamento e teste do desempenho do futuro modelo serão utilizadas os bancos de dados disponíveis diretamente no programa *MetaTrader 5* bem como as disponíveis de graça por grandes empresas como *Yahoo Finance*, *Facebook*, *Amazon*, *Google Finance*, etc.

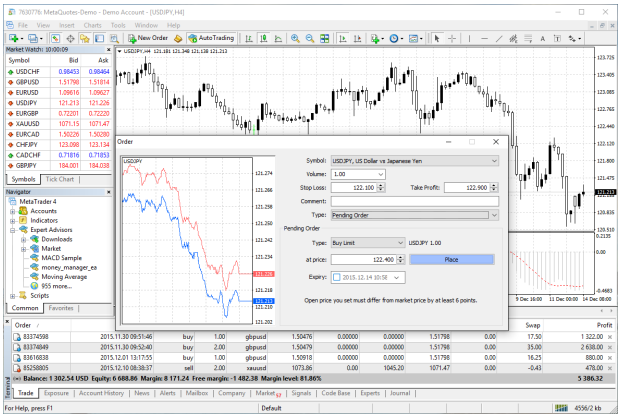

Figura 1: Interface da plataforma de negociação *Meta Trader 5*, que usaremos como base de dados do mercado financeiro. Fonte: https://www.metatrader4.com/pt/trading-platform.

O desenvolvimento do algoritmo capaz de analisar o dado e gerar uma predição serão feitos por meio das ferramentas do *Meta Trader 5* para o *Python* em conjunto com as bibliotecas *Scikit* e *TensorFlow*, na aplicação de redes neurais.

```
pd.options.mode.chained_assignment = None

#Pegar dados do Metatrader
if (mt5.initialize()):
  print("ok coneccao")
else:
  print("falha")
a = mt5.terminal_info()

#Ativo do dolar e yen
ativo = "USDJPY"
mt5.symbol_select(ativo, True)
f = mt5.copy_rates_from_pos(ativo, mt5.TIMEFRAME_M5,
      0, 10000)
g = pd.DataFrame(f)
print(g)
g['time'] = pd.to_datetime(g['time'], unit='s')
```

Figura 2: Aqui inicializamos a base de dados do *Meta Trader 5*. Analisaremos a cotação do dólar em relação ao yen.

A. Estudos Relacionados

Trabalhos acadêmicos que envolvem a aplicação de técnicas de sistemas inteligentes na análise de operações financeiras são abundantes. Mattos Braga (2022), no artigo "*Machine Learning* aplicado em ações no mercado financeiro B3" [1], aplica os algoritmos de *Machine Learning* de Regressão Linear, *Support Vector Machine* (SVM), *K Nearest Neighbor* (KNN), Floresta Aleatória e Árvores de Decisão treinados com dados históricos da Petrobrás, Itaú, Bradesco, Vale e Ambev para avaliar a assertividade de operações financeiras. O autor compara o desempenho de diversos algoritmos de *Machine Learning* à estratégia de *Buy And Hold* (comprar e segurar as ações), obtendo resultados significativamente melhores quando os algoritmos controlam a compra e venda de ações.

```
#Tamanho de cada input para a rede neural
Mrange = 10

#Modelar os dados para treinamento
threshold = 0.7
data = g.filter(['close'])
dataset = data.values
training_data_len = math.ceil(len(dataset)*threshold
    )

scaler = MinMaxScaler(feature_range=(-0.5,0.5))

scale_data = scaler.fit_transform(dataset)

#Criar dado de treino
train_data = scale_data[0:training_data_len,:]
x_train = []
y_train = []

for i in range(Mrange,len(train_data)):
  x_train.append(train_data[i-Mrange:i, 0])
  y_train.append(train_data[i, 0])
```

Figura 3: Aqui separamos um subconjunto do vetor de *data frame* para treinar a rede neural e escalonamos os dados para a escala de 0,5 até -0,5.

```
model = Sequential()
model.add(LSTM(100, return_sequences=True,
    input_shape=(x_train.shape[1], 1)))
model.add(LSTM(100, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))

model.compile(optimizer= 'adam',loss='
    mean_squared_error')
model.fit(x_train,y_train,batch_size=1,epochs=1)
```

Figura 4: Aqui criamos e treinamos o modelo da rede neural.

Uma revisão sistemática de artigos sobre robôs do mercado financeiro e seus benefícios e limitações é oferecida por Gotardo et al. (2022) [2]. O artigo aponta que quatro principais robôs brasileiros (o *Magnetis, Monetus, Vérios* e *Warren*) fazem a gestão de mais de 1.130 bilhões e 160 mil contas abertas. Comparado com os principais robôs dos EUA, o *Wealthfront* e o *Betterment*, que administram um patrimônio de cerca 26 bilhões, o porte dos principais robôs brasileiros ainda é relativamente pequeno.

B. Bases de dados

1) Yahoo Finance: O *Yahoo Finance* é um serviço oferecido pelo *Yahoo* que oferece informações relacionadas ao mercado financeiro. Ele é um dos recursos preferidos para obter informações sobre cotações de ações, análises, gráficos, notícias sobre o mercado, rastreamento de portfólio, entre outros. Por meio do *Yahoo Finance*, temos acesso às ações, índices financeiros, relatórios de ganhos, notícias relevantes e análises de mercado de empresas específicas, além de cotações em tempo real e históricas de ações. Utilizamos a base de dados do *Yahoo Finance* para alguns testes em um

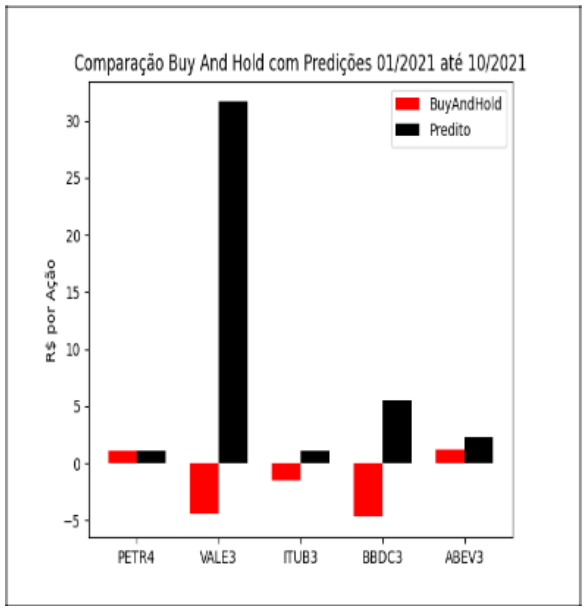

Figura 5: Demonstração da aplicação do método *Support Vector Machine* (SVM) para os históricos de empresas selecionadas. Na empresa VALE3, ao longo de três meses, a predição obteve um lucro de $R\$31,65$ por ação, enquanto a estratégia de *Buy And Hold* teve um prejuízo de $R\$4,45$ por ação. Fonte: Braga (2022).

Figura 6: Cotação do dólar em relação ao euro entre 2 e 6 de Janeiro de 2023. Fonte: Os autores.

```python
from datetime import datetime
import MetaTrader5 as mt5
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

if (mt5.initialize()):  #inicializador
  print("ok conexão")
else:
  print("falha")
a = mt5.terminal_info()
ativo = "PETR4"  #Código da B3 para a petrobras
mt5.symbol_select(ativo, True)
f = mt5.copy_rates_from_pos(ativo, mt5.TIMEFRAME_M5, 0, 40)
g = pd.DataFrame(f)
g['time'] = pd.to_datetime(g['time'], unit='s')
g['close'].plot

date = datetime(2023, 5, 22)  # tempo para analise
flag = mt5.COPY_TICKS_ALL  #base de dados
dados = mt5.copy_ticks_from(ativo, date, 10,flag)  #analise de dados referente ao ativo no tempo determidado
df = pd.DataFrame(dados)
mt5.shutdown()  # fecha o sistema
```

Figura 7: Código desenvolvido para plotar as ações da Petrobras no dia 22 de maio de 2023. Fonte: Os autores

algoritmo de *Random Forest*.

2) Meta Trader 5: O Meta Trader 5 (MT5) é a última versão disponível da plataforma de negociação eletrônica desenvolvida pela empresa *MetaQuotes Software*. A *MetaQuotes* fornece uma biblioteca *Python* que permite a comunicação direta com a plataforma MT5 a partir de programas em *Python*. Com essa biblioteca temos acesso a funcionalidades para obter dados de mercado, enviar ordens de negociação e realizar outras operações de negociação por meio do MT5.
O *Meta Trader 5* utiliza uma base de dados interna conhecida como *MetaTrader History Center* para gerenciar dados relacionados a histórico de preços, indicadores econômicos, operações de negociação e outras informações do mercado. Os dados históricos disponíveis no MT5 são provenientes de fornecedores de dados de mercado, como corretoras e provedores de serviços financeiros. Esses dados incluem informações como preços de abertura, fechamento, máxima, mínima e volume de negociação em diferentes intervalos de tempo.

C. Modelos de Machine Learning

1) Random Forest Regression: O *random forest regression* é um algoritmo de aprendizado de máquina que combina os princípios de aprendizagem em conjunto (do inglês *ensemble learning*) e árvores de decisão para tarefas de regressão. Ele é uma extensão do algoritmo *random forest*, que é usado principalmente em tarefas de classificação.
No *random forest regression* uma coleção de árvores de decisão é criada, onde cada árvore é treinada com um subconjunto aleatório dos dados de treinamento e utiliza um subconjunto aleatório de entradas. A aleatoriedade introduzida no treinamento ajuda a reduzir o sobreajuste (*overfitting*) e aumenta a capacidade de generalização do modelo. Durante cada treinamento cada árvore de decisão aprende independentemente a predizer a variável alvo baseada em diferentes subconjuntos dos dados. A predição final do *random forest regression* é obtida pela média das saídas ou pela predição majoritária das árvores de decisão individuais.
Ainda que *random forests* são poderosas e versáteis, elas podem não ser a escolha ótima para todo problema de regressão. A performance e adequação do algoritmo depende das características específicas do problema dado.

2) Rede Neural: Uma rede neural, também conhecida como rede neural artificial, é um modelo computacional inspirado na estrutura e funcionamento das redes neurais biológicas no cérebro humano. Ela é um modelo matemático composto por nós interconectados, chamados de neurônios artificiais, organizados em camadas. A unidade básica de uma rede neural é um neurônio artificial que recebe entradas, realiza operações com essas entradas e, por fim, produz uma saída. Cada entrada é multiplicado por um peso correspondente, e essas entradas são somadas. A soma é então passada por uma função de

ativação, que introduz não-linearidade e produz a saída do neurônio.

Neurônios são organizados em camadas, consistindo em uma camada de entrada, uma ou mais camadas escondidas, e uma camada de saída. A camada de entrada recebe as entradas, que são propagadas pelas entradas intermediárias até chegarem à camada de saída, produzindo uma saída final da rede neural. Durante o processo de treinamento, a rede neural aprende por meio do ajustamento dos pesos associados a cada conexão entre neurônios. Esse ajuste é tipicamente realizado usando a técnica de *backpropagation*, que compara a saída da rede à saída desejada e atualiza os pesos de modo a minimizar a diferença entre elas. Este processo iterativo continua até que o nível de acurácia se torne satisfatório.

Redes neurais tem a capacidade de aprender e generalizar a partir de padrões nos dados, tornando-as ferramentas poderosas no reconhecimento de padrões, classificação, regressão e otimização.

D. Bibliotecas

Utilizamos as seguintes bibliotecas *Python* para o desenvolvimento do nosso projeto:

1) Scikit-learn: O *scikit-learn* [3], também conhecido como *sklearn*, é uma biblioteca *Python* de código aberto construída sobre os pacotes *Numpy*, *SciPy* e *matplotlib* que fornece ferramentas simples, flexíveis e eficientes para a análise de dados. O *sklearn* fornece métodos para os diferentes estágios da análise de dados, isto é: o pré-processamento e tratamento dos dados, a sua classificação, regressão, clusterização, redução de dimensionalidade e ajuste de parâmetros.

2) TensorFlow: O *TensorFlow* [4] é uma biblioteca open source para aplicações de *machine learning* e *deep learning* desenvolvida pela *Google Brain* amplamente usada nas pesquisas e na indústria. O *TensorFlow* representa computações como gráficos computacionais. Esses grafos consistem de uma série de nós interconectados, onde cada nó representa uma operação matemática e os vértices representam o fluxo de dados (*tensor*) entre os nós. Essa computação baseada em grafos permite execução paralela eficiente e otimização computacional, tornando-a útil para aplicações de *machine learning* de larga escala.

3) Keras: O Keras [5] é uma API de redes neurais de código aberto de alto nível escrita em *Python*. Ela é desenvolvida a partir do *TensorFlow* e fornece uma interface amigável e interativa para desenvolver e treinar redes neurais.

4) Matplotlib: O *Matplotlib* [6] é uma biblioteca para a criação de visualizações de dados estáticas, animadas e interativas facilmente customizáveis. Com ele podemos criar gráficos interativos com opções de *zoom*, *pan*, *et cetera*.

IV. Resultados

Nesta seção apresentaremos a saída de dados de nossos modelos de *random forest* e rede neural de modo que possamos comparar as curvas preditas dos dados com a curva real histórica no geral. A rede neural foi bem melhor em prever os valores precisos, como podemos observar comparando as figuras **??**, 10 com a 9, onde o *random forest* não foi capaz de prever valores acima dos para os quais ele foi treinado. Ainda assim, o *random forest* obteve uma taxa de acertos melhor do que as redes neurais, como podemos notar nas tabelas I e II. No entanto essa taxa se trata de uma análise de acerto quanto a decisão tomada de dar um lance ou não, então o *random forest* acerta parte das cessões em que ele acha que não vai acontecer nada pois caso elas estejam caindo de preço é considerada a decisão certa não investir. Para melhorar o *random forest* seria necessário modificá-lo para que ele preveja valores acima do limite dos dados de treino, enquanto para melhorar as redes neurais seria bom fornecer mais dados externos a serem considerados e combinação com outros modelos que possam otimizar ou fazer previsões mais apropriadas para certos dados.

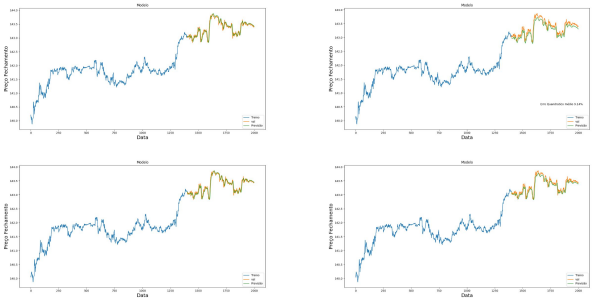

Figura 8: Nesta figura mostramos quatro exemplos de predição da cotação do dólar referente ao yen, comparando a predição a partir da aplicação de redes neurais e o valor histórico real. Obtemos porcentagens de erro tão baixas quanto 0.5%. Fonte: Os autores.

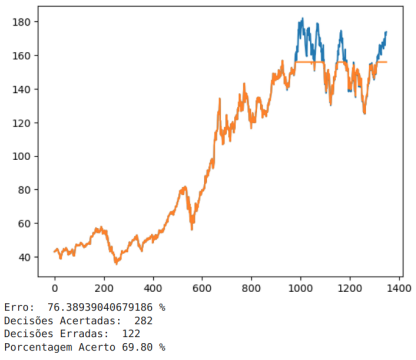

Figura 9: Exemplo gráfico com resultado do Random Forest. Com 69% de decisões certas e o erro quadrático é de 76%. Fonte: Os autores

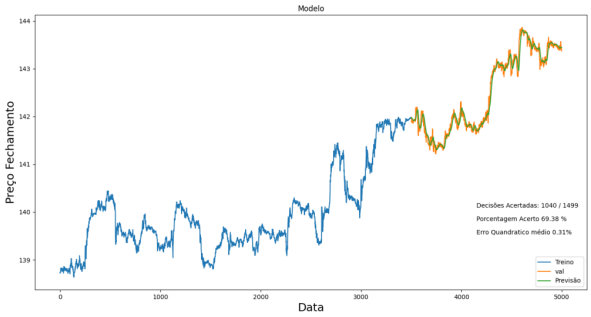

Figura 10: Exemplo de gráfico de uma saída do modelo de nossa rede neural. Observamos que a porcentagem de acertos de decisão é de 70% e o erro quadrático é de 1%, o que são ótimos indicadores. Fonte: Os autores

					Média
Erro quadrático	5.52%	3.16%	4.84%	2.09%	3.9%
Taxa de acertos	50.13%	49.54%	50.04%	49.87%	49.9%

Tabela I: Tabela com os erros quadráticos e as taxas de acertos de quatro iterações de saída do modelo de rede neural. Fonte: Os autores.

					Média
Erro quadrático	76.38%	76.36%	75.94%	79.18%	76.06%
Taxa de acertos	69.80 %	70.30 %	70.05%	70.30%	70.54 %

Tabela II: Tabela com os erros quadráticos e as taxas de acertos de quatro iterações de saída do algoritmo *random forest regression*, utilizando diferentes árvores aleatórias. Fonte: Os autores.

V. Conclusão

Concluímos que o nosso modelo de predição produziu resultados bons para o que foi proposto, apresentando erros quadráticos entre as curvas preditas pelo modelo de rede neural e a curva real relativamente baixos, porém as taxas de predição tanto para o algoritmo *random forest regression* quanto para a rede neural ficaram com uma média em torno de 50%, como mostramos na seção de resultados. Dessa forma os modelos estão longe de uma versão utilizável na prática, o que está dentro do esperado visto que nosso parâmetros de previsão foram apenas referentes a variação do valor no tempo. Para um modelo mais eficaz, algumas possíveis soluções incluem levar em conta o valor de concorrentes, utilizar relatórios de planos semanais fornecidos pela empresa, levar em conta a variação no valor de produtos ou empresas ligadas(por exemplo fornecedora de matéria prima), desenvolvimento de uma rede neural mais customizada para este cenário não ficando limitado ao modelo sequencial utilizado neste projeto. No geral, os resultados foram satisfatórios para uma prototipação inicial mas ainda está bem longe da versão final sendo necessária a análise de mais áreas, paradigmas e fatores para cada dado previsto.

Referências

[1] BRAGA, B. M. et al. Machine Learning aplicado em ações no mercado financeiro b3. Colloquium Exactarum, v. 14, n. 1, p. 57–66, 2022. ISSN 2178-8332. Acesso em: 24 de Abril de 2023. Disponível em: https://revistas.unoeste.br/index.php/ce/article/view/4281.

[2] GOTARDO, D. L. F. Robôs no mercado de ações: Benefícios e limitações. Convibra, 2020. Acessado em 24 de abril de 2023. Disponível em: https://convibra.org/congresso/res/uploads/pdf/artigo19813_20201248.pdf.

[3] PEDREGOSA, F., VAROQUAUX, Gael, GRAMFOX, A., MICHEL, V., THIRION, B., GRISEL, O., et al. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12 de outubro, 2825–2830.

[4] ABADI, Martin, BARHAM, P., CHEN, J., CHEN, Z., DAVIS, A., DEAN, J.,et al. (2016). Tensorflow: A system for large-scale machine learning. In 12th Symposium on Operating Systems Design and Implementation (pp. 265–283).

[5] CHOLLET, F., et al. (2015). Keras. GitHub. Disponível em: https://github.com/fchollet/keras.

[6] HUNTER, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90–95.