

**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**  
**NÚCLEO DE EDUCAÇÃO A DISTÂNCIA**  
**Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data**

**Rafael Silva Pinto**

**MACHINE LEARNING PARA**  
**RECOMENDAÇÃO DE COMPRA DE AÇÕES NA BOLSA DE VALORES**

Belo Horizonte  
2021

**Rafael Silva Pinto**

**MACHINE LEARNING PARA  
RECOMENDAÇÃO DE COMPRA DE AÇÕES NA BOLSA DE VALORES**

Trabalho de Conclusão de Curso apresentado  
ao Curso de Especialização em Ciência de  
Dados e Big Data como requisito parcial à  
obtenção do título de especialista.

Belo Horizonte  
2021

## **AGRADECIMENTO**

Agradeço a Deus pela saúde e oportunidade de concluir esse trabalho.

A minha esposa Élide e minhas enteadas Ana Clara e Rafaela pelo apoio e momentos de lazer.

Aos meus pais e irmãos, Agostinho, Beatriz, Lúcia e Luciano por me acompanharem em muitos momentos importantes da minha vida.

Aos professores e a PUC Minas por todo o conhecimento compartilhado durante todo o curso.

## SUMÁRIO

1. Introdução.....	5
1.1. Contextualização.....	5
1.2. O problema proposto.....	6
2. Coleta de Dados .....	7
2.1 Dados históricos dos valores das ações .....	7
2.2 Dados históricos do valor do Dólar .....	10
3. Processamento/Tratamento de Dados .....	12
3.1 Dados históricos dos valores das ações .....	12
3.2 Dados históricos do valor do Dólar .....	14
3.3 Merge, junção dos dados .....	15
4. Análise e Exploração dos Dados .....	17
4.1 Feature Engineering.....	21
4.2 Normalização dos dados .....	24
4.3 Coluna Target .....	25
4.4 Exploração dos dados .....	27
4.5 Conversão de dados categóricos .....	29
4.6 Frequência de acerto .....	30
5. Criação de Modelos de Machine Learning .....	32
5.1 Introdução e métricas .....	32
5.2 Regressão Logística .....	35
5.3 Árvore de Decisão .....	39
5.4 Redes Neurais Artificiais .....	42
5.5 XGBoost.....	45
6. Apresentação dos Resultados .....	47
6.1 Introdução.....	47
6.2 Resultados para MGLU3, VALE3 e PETR4, ganho de 10% em até 15 dias...47	
6.3 Resultados para outras ações .....	48
7. Considerações finais .....	52
8. Links.....	54
REFERÊNCIAS.....	55

## 1. Introdução

### 1.1. Contextualização

Com a recente baixa no valor da taxa Selic (Sistema Especial de Liquidação e de Custódia), investimentos de renda fixa, como a poupança, deixaram de ser atrativos e muitos brasileiros iniciaram a jornada em investimentos de maior risco.

Um dos investimentos mais atrativos e comentados recentemente é ações na bolsa de valores. O assunto vem se destacando principalmente pela supervalorização de algumas empresas como Magazine Luiza, que valorizou quase 800 vezes nos últimos 5 anos, indo de R\$ 0,03 para R\$ 23,88 (em valores atuais, após desdobramento das ações) entre 01/12/2015 e 01/12/2020 (INFOMONEY, 2020).

Muitos dos novos investidores na bolsa de valores não buscam o ganho de longo prazo, mas sim retornos rápidos. Ao investir em uma empresa buscando o longo prazo, significa que o investidor acredita no setor que aquela companhia atua e em seu desempenho. Porém, retornos de curto prazo estão muito relacionados a acontecimentos políticos, econômicos e especulações recentes. Um exemplo é a queda no valor de todas as ações em março de 2020, a incerteza do que ocorreria com o mercado com o surgimento do novo corona vírus provocou uma redução significativa em todas as ações do mundo inteiro.

Além de acontecimentos e especulações, o “efeito manada” é muito observado na compra e venda de ações. Quando o valor de uma ação começa a crescer rapidamente, diversos investidores compram essa ação com medo de não aproveitarem a possível oportunidade, e o mesmo ocorre quando uma ação começa a perder seu valor rapidamente.

Mesmo com a supervalorização de algumas empresas, o investimento em ações na bolsa de valores é de alto risco e, ao se buscar retornos de curto prazo, pode ser uma armadilha para investidores recém chegados.

Diante desse contexto, esse trabalho busca o desenvolvimento de um algoritmo para auxiliar na recomendação de compra de ações que tendem a crescer no curto-médio prazo. Todo o trabalho foi desenvolvido usando *Python* e bibliotecas de *Machine Learning* no ambiente de desenvolvimento *Jupyter Lab*.

## 1.2. O problema proposto

O problema proposto consiste na construção de um algoritmo para auxiliar na tomada de decisão de compra de ações na bolsa de valores B3. O trabalho não deve ser visto como a construção de uma metodologia para definir a compra, mas sim uma ferramenta que apoia a decisão baseada em dados históricos da ação.

Para facilitar o entendimento do problema e da solução a ser proposta utilizamos a técnica do 5Ws, que consiste em responder as seguintes perguntas:

**Why?** Reduzir os riscos em investimentos na bolsa de valores.

**Who?** Novos investidores que buscam ganho de curto-médio prazo.

**What?** Gerar um algoritmo que seja capaz de recomendar a compra de ações.

**Where?** Todos os dados e aplicações da metodologia apresentados nesse trabalho são feitos pela internet.

**When?** O prazo para retorno do investimento é de 15 a 30 dias e o horizonte do histórico utilizado é desde o IPO (Oferta Pública Inicial) da ação.

Na bolsa de valores existem mais de 350 empresas listadas, e as ações MGLU3 (Magazine Luiza), VALE3 (Vale) e PETR4 (Petrobras) foram definidas como o caso de estudo dentro desse trabalho, com o intuito de delimitar o escopo e viabilizar as análises.

A maior parte dos trabalhos que envolvem ações da bolsa de valores tenta prever o valor futuro dos ativos. Diferentemente, o método aqui proposto busca a recomendação da compra da ação, na qual a decisão é baseada num ganho esperado dentro de um horizonte de tempo determinado. A recomendação de compra é uma variável binária, na qual (1) representa a recomendação de compra, e (0) a não recomendação.

## 2. Coleta de Dados

Neste trabalho foram usadas duas fontes de dados distintas, uma contendo dados históricos dos valores das ações e outra contendo dados históricos do valor do dólar.

A seguir são apresentados os métodos de captura da informação e uma breve descrição dos dados obtidos.

### 2.1 Dados históricos dos valores das ações

Para o desenvolvimento deste trabalho foi utilizada a biblioteca `pandas_datareader` e a API do Yahoo Finance (YAHOO, 2020). Dessa forma, um dataset contendo o histórico do valor das ações pode ser obtido com facilidade.

O uso da API Yahoo Finance pode ser feito conforme o código a seguir, aplicado à ação MGLU3 para o período 01/01/2019 a 01/01/2020.

```
from pandas_datareader import data
df = data.DataReader('MGLU3.SA', 'yahoo', '2019-01-01', '2020-01-01')
df.head()
```

	High	Low	Open	Close	Volume	Adj Close
Date						
2019-01-02	5.81219	5.59469	5.63812	5.81219	27017600.0	5.549253
2019-01-03	5.79656	5.60500	5.78094	5.75000	26342400.0	5.489877
2019-01-04	5.76094	5.57344	5.75156	5.57500	27721600.0	5.322793
2019-01-07	5.70250	5.53969	5.56250	5.66156	21174400.0	5.405438
2019-01-08	5.70281	5.59375	5.67344	5.61594	19398400.0	5.361881

Portanto, três datasets foram gerados, um para cada ação estudada (MGLU3, VALE3 e PETR4). Porém, não é possível a junção deles, uma vez que denotam informações de ações distintas.

A seguir é apresentado um resumo dos dados obtidos nesses datasets.

Nome da coluna	Descrição	Tipo
Date	Data	Datetimeindex
High	Máximo valor da ação na data	Float
Low	Mínimo valor da ação na data	Float
Open	Valor de abertura da ação na data	Float
Close	Valor de fechamento da ação na data	Float
Volume	Volume total negociado da ação na data	Float
Adj Close	Valor de fechamento ajustado ação na data. Este valor considera desconto de dividendos, desdobramentos e outros	Float

Os dados obtidos através da API apresentam resumos diários dos valores e volume das ações. Esses dados são suficientes para uma análise primária, uma vez que são suficientes para construção dos gráficos mais comumente utilizados por investidores na bolsa de curto-médio prazo.

O gráfico de velas, ou candlestick, associado a médias móveis de 9 e 21 períodos, é o mais utilizado entre os investidores. Ele apresenta variações diárias com os valores de abertura, fechamento, máximo e mínimo, assim como o comportamento histórico do valor da ação.

A figura a seguir mostra um exemplo desse gráfico para a ação MGLU3 entre agosto e dezembro de 2020. Sendo a linha branca a média de 9 períodos e a linha cinza a média de 21 períodos.

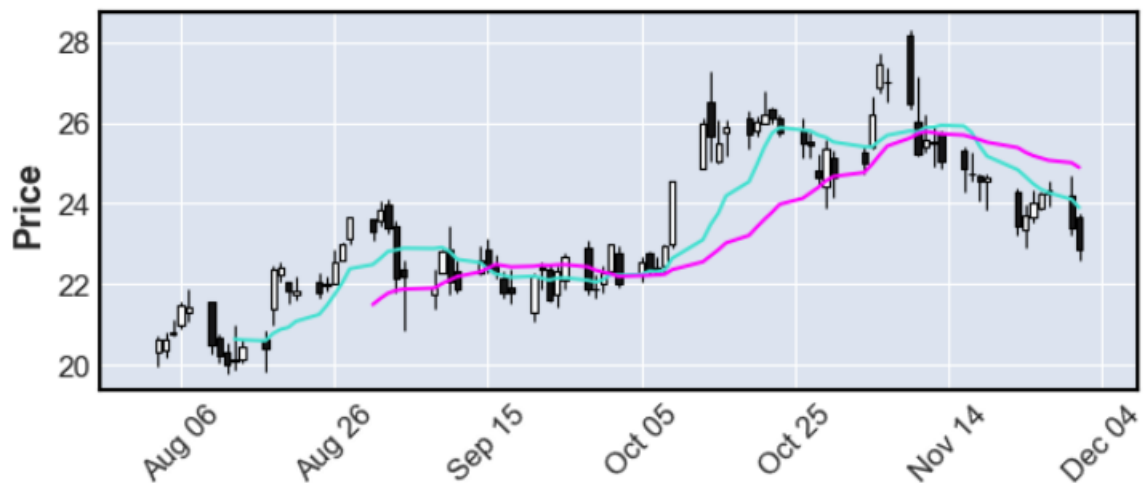


**Fig. 1** - Exemplo de gráfico candlestick com médias móveis (MGLU3 Ago-Dez 2020)

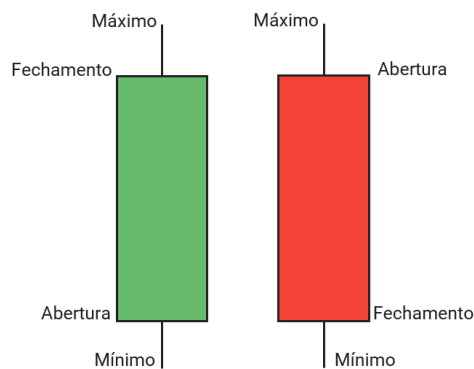


O gráfico apresentado na Fig. 1 foi obtido utilizando a plataforma <https://br.tradingview.com/>, mas pode, também, ser gerado dentro de um algoritmo em Python utilizando a biblioteca mplfinance, conforme código a seguir.

```
import mplfinance as mpf
plot_begin = '2020-08-01'
df_plot = data.DataReader(param.stock_list[stock], param.source, plot_begin, param.dt_end)
mpf.plot(df_plot, type='candle', mav=(9,21), volume=False, show_nontrading=True, figsize=(8,3))
```



A figura abaixo mostra um detalhamento dos valores de máximo, mínimo, abertura e fechamento para o gráfico candlestick. Sendo que velas verdes denotam que o valor de fechamento é superior ao de abertura e velas vermelhas o oposto.



**Fig. 2** – Explicação de valores no candlestick

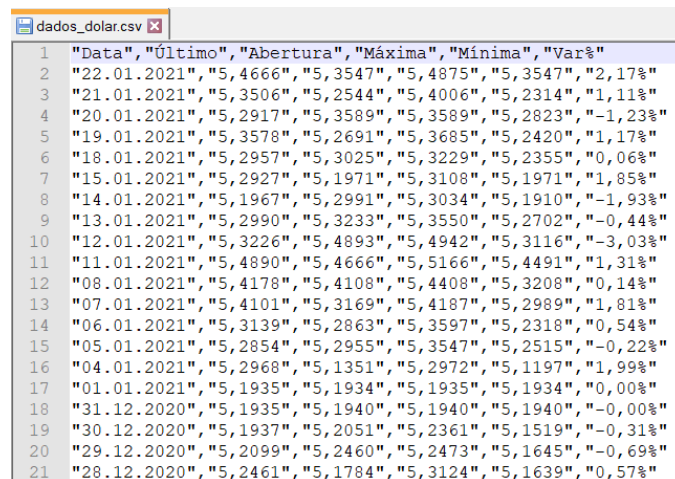
A metodologia proposta nesse trabalho não considera especulações e notícias em sua concepção, portanto não são utilizadas APIs para consumir dados de notícias e redes sociais, como Twitter. O algoritmo aqui proposto baseia-se no

comportamento histórico dos valores das ações e tem como princípio a tomada de decisão utilizada por diversos investidores de curto prazo, chamada de “análise de gráfico”.

## 2.2 Dados históricos do valor do Dólar

O dólar é usado como referências em diversas transações e análises financeiras, e pode influenciar os custos operacionais das empresas e o valor de suas ações. Assim, uma das fontes de dados desde trabalho é o valor histórico do dólar em relação ao real.

Em Investing (2020) é possível realizar o download desse dado histórico, o qual gera um CSV com os dados diários, uma amostra desse CSV é apresentada logo abaixo.



```

1 "Data","Último","Abertura","Máxima","Mínima","Var%"
2 "22.01.2021","5,4666","5,3547","5,4875","5,3547","2,17%"
3 "21.01.2021","5,3506","5,2544","5,4006","5,2314","1,11%"
4 "20.01.2021","5,2917","5,3589","5,3589","5,2823","-1,23%"
5 "19.01.2021","5,3578","5,2691","5,3685","5,2420","1,17%"
6 "18.01.2021","5,2957","5,3025","5,3229","5,2355","0,06%"
7 "15.01.2021","5,2927","5,1971","5,3108","5,1971","1,85%"
8 "14.01.2021","5,1967","5,2991","5,3034","5,1910","-1,93%"
9 "13.01.2021","5,2990","5,3233","5,3550","5,2702","-0,44%"
10 "12.01.2021","5,3226","5,4893","5,4942","5,3116","-3,03%"
11 "11.01.2021","5,4890","5,4666","5,5166","5,4491","1,31%"
12 "08.01.2021","5,4178","5,4108","5,4408","5,3208","0,14%"
13 "07.01.2021","5,4101","5,3169","5,4187","5,2989","1,81%"
14 "06.01.2021","5,3139","5,2863","5,3597","5,2318","0,54%"
15 "05.01.2021","5,2854","5,2955","5,3547","5,2515","-0,22%"
16 "04.01.2021","5,2968","5,1351","5,2972","5,1197","1,99%"
17 "01.01.2021","5,1935","5,1934","5,1935","5,1934","0,00%"
18 "31.12.2020","5,1935","5,1940","5,1940","5,1940","-0,00%"
19 "30.12.2020","5,1937","5,2051","5,2361","5,1519","-0,31%"
20 "29.12.2020","5,2099","5,2460","5,2473","5,1645","-0,69%"
21 "28.12.2020","5,2461","5,1784","5,3124","5,1639","0,57%"

```

A leitura desse CSV é realizada usando o comando **read\_csv** do Pandas, o qual gera o DataFrame **df\_dolar**.

```

df_dolar = pd.read_csv('dados_dolar.csv')
df_dolar.head(5)

```

	Data	Último	Abertura	Máxima	Mínima	Var%
0	22.01.2021	5,4666	5,3547	5,4875	5,3547	2,17%
1	21.01.2021	5,3506	5,2544	5,4006	5,2314	1,11%
2	20.01.2021	5,2917	5,3589	5,3589	5,2823	-1,23%
3	19.01.2021	5,3578	5,2691	5,3685	5,2420	1,17%
4	18.01.2021	5,2957	5,3025	5,3229	5,2355	0,06%

Logo abaixo é apresentada uma descrição das colunas do DataFrame **df\_dolar**.

Nome da coluna	Descrição	Tipo
Data	Data dos valores	String
Última	Valor de fechamento do Dólar no dia	String
Abertura	Valor de abertura do Dólar no dia	String
Máxima	Valor máximo do Dólar no dia	String
Mínima	Valor mínimo do Dólar no dia	String
Var%	Variação do valor entre fechamento e abertura	String

Já observa-se a necessidade de tratamento dos dados, uma vez que as colunas do DataFrame são geradas como *strings*, mesmo representando valores numéricos.

### 3. Processamento/Tratamento de Dados

As bases coletadas foram analisadas e realizada uma análise prévia para entendimento dos dados obtidos.

#### 3.1 Dados históricos dos valores das ações

O volume de dados obtido para uma única ação possibilita a aquisição dos dados durante o processamento do código, pois a API em questão retorna os valores máximo (High), mínimo (Low), abertura (Open), fechamento (Close), volume (Volume) e fechamento ajustado (Adj Close) para cada dia do horizonte desejado. Portanto, os dados são obtidos no momento da execução do código.

A fim de facilitar a entrada de dados e definir os parâmetros utilizados no algoritmo, foi implementado o código a seguir.

```
import pandas as pd, numpy as np, seaborn as sns, matplotlib.pyplot as plt, warnings
from datetime import date
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
class NClass(): pass
param = NClass()
```

```
from pandas_datareader import data
param.gain = 0.1
param.DaysAhead = 15
stock = 0
param.stock_list = ['MGLU3.SA', 'VALE3.SA', 'PETR4.SA']
param.dt_begin = ['2011-05-01', '2000-01-02', '2000-01-02']
param.dt_end = '2020-12-01'
param.dt_check = '2020-11-17'
param.source = 'yahoo'
df = data.DataReader(param.stock_list[stock], param.source, param.dt_begin[stock], param.dt_end)
```

```
df.head()
```

	High	Low	Open	Close	Volume	Adj Close
Date						
2011-05-02	0.518750	0.503125	0.503125	0.514062	280003200.0	-0.008697
2011-05-03	0.521562	0.506250	0.515625	0.509375	33670400.0	-0.008618
2011-05-04	0.515000	0.510000	0.510000	0.515000	39203200.0	-0.008713
2011-05-05	0.512812	0.510000	0.512812	0.511875	35097600.0	-0.008660
2011-05-06	0.511875	0.500000	0.510937	0.508125	38672000.0	-0.008596

Assim, o dataframe **df** contém todos dos dados diários da ação definida em **param.stock\_list[stock]** para o período entre **param.dt\_begin[stock]** e **param.dt\_end**. Os parâmetros **param.gain** e **param.DaysAhead** definem o ganho desejado e o limite de dias para esse ganho, respectivamente, que são 10% e 15 dias para o exemplo apresentado.

Os dados não requerem um tratamento especial pois não possuem registros duplicados ou que não parecem fazer parte do dataset. Existe ainda uma pequena parcela de dados nulos para a coluna Volume, porém é uma porcentagem inferior a 1% dos dados, conforme apresentado a seguir.

```
df.index.duplicated().sum()
```

```
0
```

```
(df==0).astype(int).sum()
```

```
High      0
Low       0
Open      0
Close     0
Volume    22
Adj Close 0
dtype: int64
```

```
df[df.Volume==0].head(5)
```

	High	Low	Open	Close	Volume	Adj Close
Date						
2011-05-10	0.503125	0.503125	0.503125	0.503125	0.0	-0.008512
2011-05-16	0.510937	0.510937	0.510937	0.510937	0.0	-0.008644
2012-01-16	0.295625	0.295625	0.295625	0.295625	0.0	-0.005001
2012-02-13	0.312187	0.312187	0.312187	0.312187	0.0	-0.005282
2012-03-19	0.359375	0.359375	0.359375	0.359375	0.0	-0.006080

```
print("Porcentagem de dados nulos: %.2f%%" % (df.Volume[df.Volume==0].count()/df.Volume.count()*100))
```

```
Porcentagem de dados nulos: 0.92%
```

Dessa forma, definiu-se que os dados nulos podem ser ignorados, pois a mesma proporção de dados nulos se repete para as demais ações estudadas. Sendo 0.92% para MGLU3, 0.98% para VALE3 e 0.63% para PETR4.

O volume de dados obtidos depende da ação escolhida, pois são utilizados dados desde o IPO da empresa, limitado a 01/01/2000. Abaixo é apresentado a quantidade de dados para cada uma das três ações estudadas nesse trabalho.

Ação	MGLU3	VALE3	PETR4
Quantidade de registros	2.379	5.233	5.146

### 3.2 Dados históricos do valor do Dólar

No processo de coleta dos dados, observamos que os dados referentes ao histórico do dólar são carregados como *strings*, mesmo representando valores numéricos e datas. Assim, o código a seguir realiza os tratamentos necessários no DataFrame **df\_dolar**.

```
df_dolar['Último'] = df_dolar['Último'].apply(lambda x: x.replace(',', '.')).astype(float)
df_dolar['Abertura'] = df_dolar['Abertura'].apply(lambda x: x.replace(',', '.')).astype(float)
df_dolar['Máxima'] = df_dolar['Máxima'].apply(lambda x: x.replace(',', '.')).astype(float)
df_dolar['Mínima'] = df_dolar['Mínima'].apply(lambda x: x.replace(',', '.')).astype(float)
df_dolar['Data'] = pd.to_datetime(df_dolar.Data, format='%d.%m.%Y')
df_dolar = df_dolar.drop('Var%', axis=1)
df_dolar = df_dolar.rename(columns={"Último": "dolar_close",
                                   "Abertura": "dolar_open",
                                   "Máxima": "dolar_max",
                                   "Mínima": "dolar_min"})
df_dolar = df_dolar.set_index('Data')

df_dolar.head(5)
```

	dolar_close	dolar_open	dolar_max	dolar_min
Data				
2021-01-22	5.4666	5.3547	5.4875	5.3547
2021-01-21	5.3506	5.2544	5.4006	5.2314
2021-01-20	5.2917	5.3589	5.3589	5.2823
2021-01-19	5.3578	5.2691	5.3685	5.2420
2021-01-18	5.2957	5.3025	5.3229	5.2355

As colunas numéricas foram convertidas em *float*, também substituindo o caracter “,” por “.” através da função **lambda**. A coluna Data foi convertida como index do DataFrame (para facilitar a junção dos DataFrames). E as colunas foram renomeadas.

Observa-se, logo a seguir, que **df\_dolar** não possui dados iguais a 0.

```
df_dolar[df_dolar==0].count()

dolar_close    0
dolar_open      0
dolar_max       0
dolar_min       0
dtype: int64
```

Aplicando o comando *describe()* em **df\_dolar**, verificamos que o DataFrame possui mais de 5.000 registros, considerando dados desde o ano de 2000. E ainda, os valores máximo, mínimos, média e demais apresentados não apresentam nenhuma anomalia.

```
df_dolar.describe()
```

	dolar_close	dolar_open	dolar_max	dolar_min
count	5489.000000	5489.000000	5489.000000	5489.000000
mean	2.681737	2.680768	2.701580	2.661896
std	0.913963	0.913118	0.923199	0.903861
min	1.538300	1.538600	1.542000	1.528000
25%	1.964400	1.963500	1.972900	1.952700
50%	2.364600	2.364500	2.386000	2.346800
75%	3.218100	3.218300	3.242600	3.195700
max	5.887000	5.991700	5.992200	5.817600

### 3.3 Merge, junção dos dados

Uma etapa fundamental na construção de um modelo de *machine learning* é a junção de diversas bases em um único dataset. a seguir é apresentada a junção dos DataFrames **df** e **df\_dolar** usando o comando **join**.

```
df = df.join(df_dolar)
df.head(5)
```

	High	Low	Open	Close	Volume	Adj Close	dolar_close	dolar_open	dolar_max	dolar_min
Date										
2017-05-02	0.903085	0.867226	0.890585	0.894453	18406400.0	0.858490	3.1517	3.1757	3.1959	3.1460
2017-05-03	0.909960	0.873945	0.896484	0.898437	18662400.0	0.862314	3.1652	3.1515	3.1679	3.1408
2017-05-04	0.904687	0.878593	0.890664	0.886718	29798400.0	0.851066	3.1885	3.1663	3.1956	3.1614
2017-05-05	1.066406	0.984375	0.996093	1.011718	68300800.0	0.971040	3.1774	3.1902	3.1960	3.1682
2017-05-08	1.038671	0.961875	0.992148	1.027343	22963200.0	0.986037	3.1980	3.1753	3.2077	3.1753

Após a junção dos dados é importante verificar se existe algum dado nulo, pois os datasets iniciais podem não serem totalmente complementares. A seguir, verificamos que não existem dados nulos no novo DataFrame.

```
df.isnull().sum()
```

```
High      0
Low       0
Open      0
Close     0
Volume    0
Adj Close 0
dolar_close 0
dolar_open 0
dolar_max 0
dolar_min 0
dtype: int64
```



#### 4. Análise e Exploração dos Dados

Após a captura de dados, a fase de exploração de dados foi desenvolvida focando na ação MGLU3, e posteriormente as demais ações foram analisadas.

Essa abordagem se mostrou suficiente para as ações estudadas e garante um estudo mais detalhado do processo de exploração de dados.

O primeiro passo na análise exploratória de dados foi entender algumas informações das colunas do dataset **df**.

```
df.describe()
```

	High	Low	Open	Close	Volume	Adj Close	dolar_close	dolar_open	dolar_max	dolar_min
count	2387.000000	2387.000000	2387.000000	2387.000000	2.387000e+03	2387.000000	2387.000000	2387.000000	2387.000000	2387.000000
mean	3.301892	3.166546	3.236255	3.235523	3.667301e+07	3.170768	3.129108	3.127568	3.152559	3.105403
std	5.772843	5.542120	5.661031	5.658132	3.126611e+07	5.621065	1.020603	1.020327	1.031895	1.008830
min	0.031757	0.030390	0.031445	0.030585	0.000000e+00	0.029213	1.538300	1.538600	1.542000	1.528000
25%	0.248437	0.239004	0.243750	0.243594	1.735840e+07	0.224298	2.221000	2.221400	2.233700	2.211750
50%	0.386210	0.370625	0.378437	0.379062	2.946560e+07	0.344207	3.186200	3.183900	3.204300	3.163000
75%	3.902656	3.718906	3.843593	3.812656	4.555060e+07	3.691302	3.810950	3.807300	3.842750	3.777750
max	28.309999	26.740000	28.150000	27.450001	4.304640e+08	27.421442	5.887000	5.991700	5.992200	5.817600

Observamos que as médias dos valores das colunas High, Low, Open e Close são próximas, assim como o desvio padrão, mínimo, máximo e quartis. O que demonstra um bom equilíbrio entre seus valores. Também vemos que o desvio padrão dessas colunas é superior ao valor da média, o que já demonstra uma variação considerável ao longo do tempo.

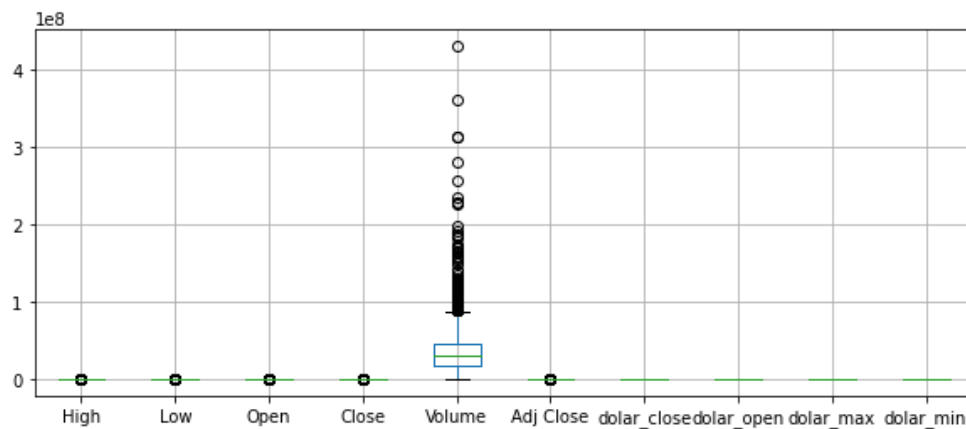
Já a coluna Volume pode variar consideravelmente, pois possui um alto desvio padrão e valores muito distante das demais colunas. Dessa forma, já se identifica a necessidade de normalizar os dados.

As colunas referentes ao dólar também apresentam dados consistentes, sem grandes variações e desvio padrão inferior à média.

A seguir é apresentado um gráfico boxplot feito a partir das colunas do dataset **df**.

```
df.boxplot(figsize=(10,4))
```

<AxesSubplot:>



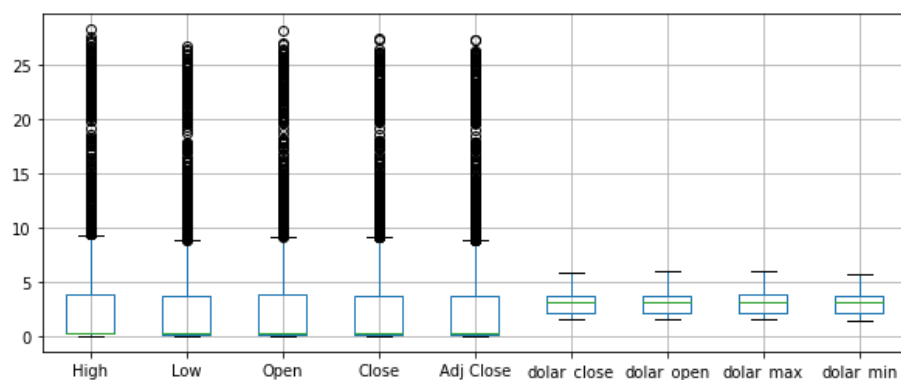
**Fig. 3** – Boxplot para todas as colunas do dataset para MGLU3

Novamente, a grande diferença na dimensão entre Volume e as demais colunas se mostra evidente. E essa diferença de valores impossibilita uma análise coerente.

Assim, um novo boxplot foi gerado retirando a coluna Volume, conforme apresentado a seguir.

```
df.drop('Volume', axis=1).boxplot(figsize=(10,4))
```

<AxesSubplot:>



**Fig. 4** – Boxplot removendo a coluna Volume do dataset para MGLU3

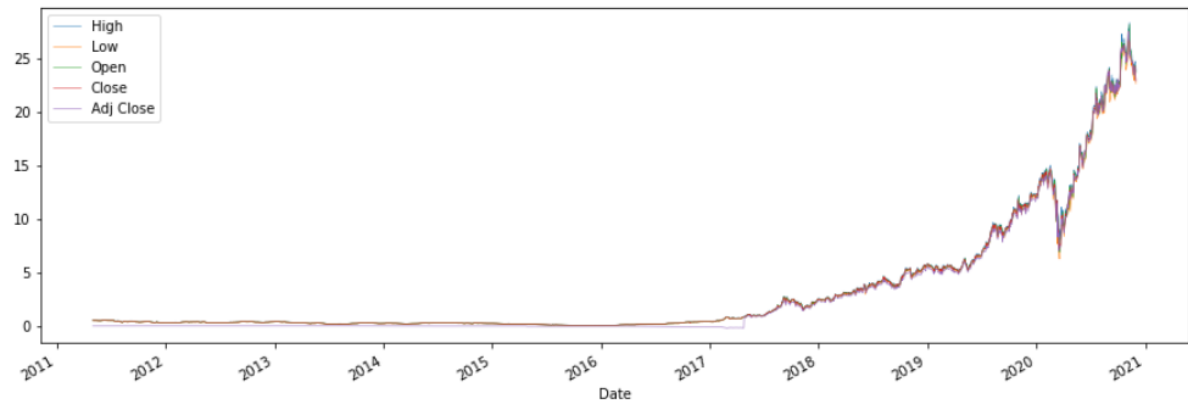
Observamos um padrão saudável para os dados referente ao dólar, porém, uma quantidade considerável de outliers nos dados referentes aos valores das

ações, isso ocorre provavelmente pelo fato da série histórica se comportar de forma diferente em intervalos de tempo distintos.

O gráfico a seguir mostra o comportamento histórico da ação.

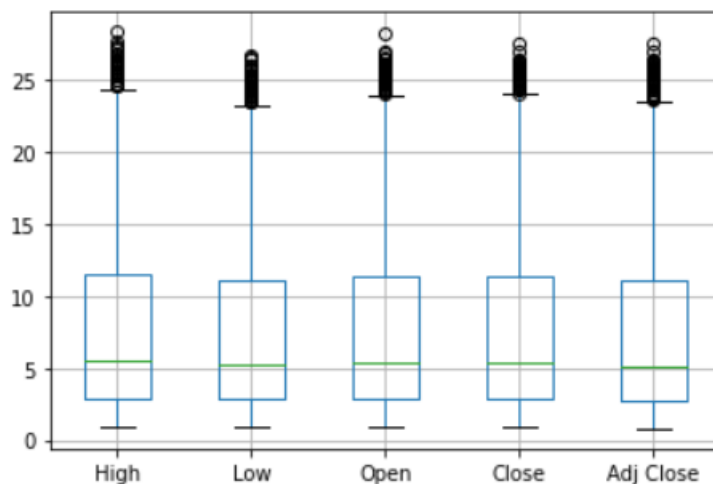
```
df[['High', 'Low', 'Open', 'Close', 'Adj Close']].plot(figsize=(15,5), linewidth=0.5)
```

```
<AxesSubplot:xlabel='Date'>
```



**Fig. 5** – Histórico da ação MGLU3

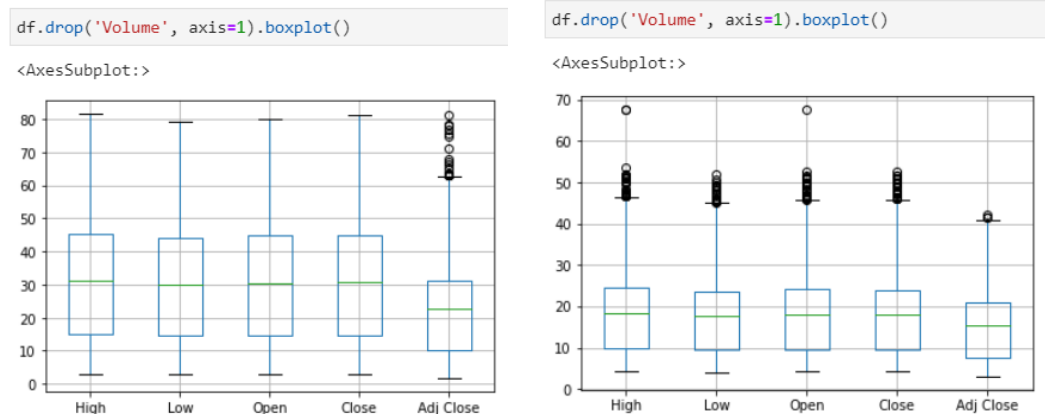
Observamos que existe um comportamento muito diferente antes de maio de 2017 e após essa data. E por isso muito dos dados recentes são considerados como outliers. Como essa ação mudou seu comportamento drasticamente, utilizar o histórico anterior a 2017 poderá trazer resultados que não representam cenários recentes da mesma. Sendo assim, um corte na data 01/05/2017 foi criado na entrada de dados. E o novo boxplot considerando apenas as colunas High, Low, Open, Close e Adj Close se torna como a seguir.



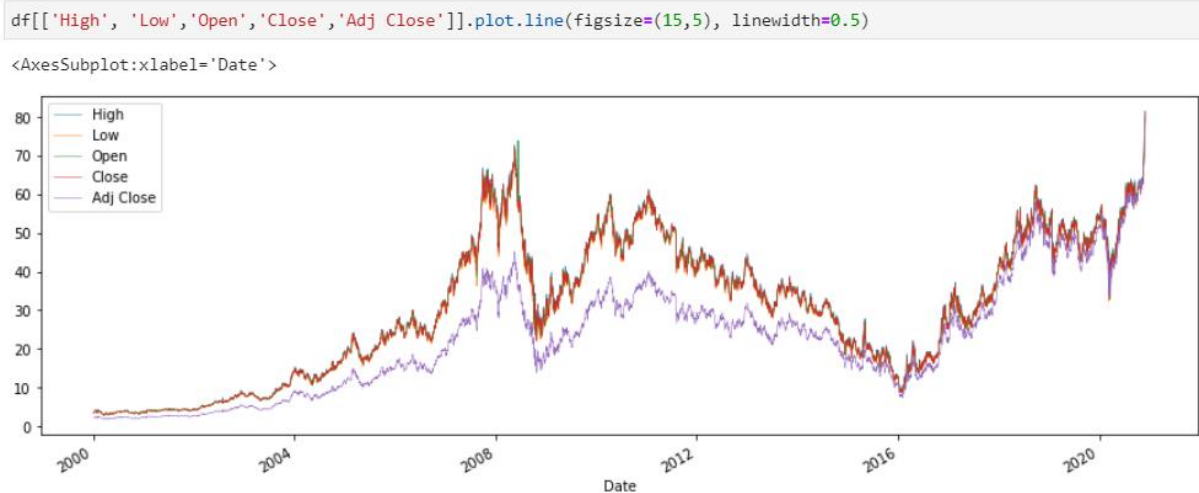
**Fig. 6** – Boxplot do dataset para MGLU3 a partir de 01/05/2017

O novo gráfico para MGLU3 com corte em 01/05/2017 ainda possui outliers, porém, estes são menos significativos pois possuem uma amplitude menor que o interquartil do boxplot.

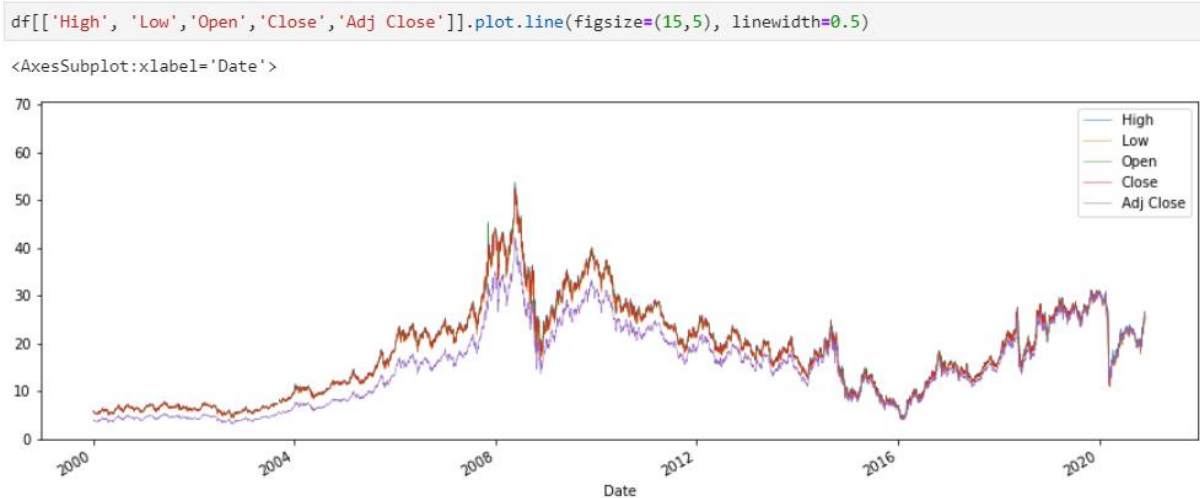
Esse comportamento não foi identificado nas ações VALE3 e PETR4, conforme gráficos a seguir.



**Fig. 7** – Boxplot para VALE3 (à esquerda) e PETR4 (à direita)



**Fig. 8** – Histórico da ação VALE3



**Fig. 9** – Histórico da ação PETR4

Sendo assim, a entrada de dados foi alterada para:

```
from pandas_datareader import data
param.gain = 0.1
param.DaysAhead = 15
stock = 0
param.stock_list = ['MGLU3.SA', 'VALE3.SA', 'PETR4.SA']
param.dt_begin = ['2017-05-01', '2000-01-02', '2000-01-02'] #nova entrada
#param.dt_begin = ['2011-05-01', '2000-01-02', '2000-01-02'] #IPO ou limite
param.dt_end = '2020-12-01'
param.dt_check = '2020-11-17'
param.source = 'yahoo'
df = data.DataReader(param.stock_list[stock], param.source, param.dt_begin[stock], param.dt_end)
```

## 4.1 Feature Engineering

O processo de *feature engineering* é um dos mais importantes na construção de modelos de *machine learning*. Nesse processo, novas variáveis são criadas e as existentes podem ser transformadas, a fim de representar características importantes do problema a ser tratado.

Para avaliar se o valor de uma ação possui tendência de subir nos próximos dias, o gráfico apresentado na Fig. 1 é comumente utilizado por especialistas que buscam investimento “baseado em gráfico” (ou não fundamentalistas). A seguir são nomeadas novas variáveis de interesse, que são comumente observadas por especialistas em investimento “baseado em gráfico”:

- **varOpenClose**: variação entre o valor de abertura e fechamento
- **varHighLow**: variação entre valor máximo e mínimo

- **mean9** e **mean21**: médias móveis de 9 e 21 períodos
- **closeGraterM9** e **closeGraterM21**: se o valor de fechamento do dia é superior às médias móveis de 9 e 21 períodos.
- **M9GreaterM21**: se a média móvel de 9 períodos está acima da média móvel de 21 períodos
- **varCloseM9** e **varCloseM21**: diferença entre valor de fechamento e as médias móveis de 9 e 21 períodos.
- **corr[X]**: tendência de alta ou baixa nos últimos [X] dias (limitado a 1)
- **dolar\_varOpenClose**: variação entre valores de fechamento e abertura do dólar.
- **dolar\_varMaxMin**: variação entre valores de máximo e mínimo do dólar.
- **varHighDm[X]**: variação do valor atual frente ao máximo histórico do dia [X].
- **varCloseDm[X]**: variação do valor atual frente ao fechamento histórico do dia [X].
- **dolar\_varHighDm[X]**: variação do dólar valor atual frente ao máximo histórico do dia [X].
- **dolar\_varCloseDm[X]**: variação do dólar valor atual frente ao fechamento histórico do dia [X].

Essas novas variáveis são criadas com o código a seguir.

```

#novas variáveis
df['seq'] = range(0, len(df['Open']))
df['varOpenClose'] = df.Close - df.Open
df['varHighLow'] = df.High - df.Low
df['mean9'] = df.Close.rolling(9).mean()
df['mean21'] = df.Close.rolling(21).mean()
df['closeGreaterM9'] = (df.Close > df.mean9)
df['closeGreaterM21'] = (df.Close > df.mean21)
df['M9GreaterM21'] = (df.mean9 > df.mean21)
df['varCloseM9'] = df.Close - df.mean9
df['varCloseM21'] = df.Close - df.mean21
df['varM9M21'] = df.mean9 - df.mean21
df['corr9'] = np.array(df.Close.rolling(9).corr(df.seq))
df['corr21'] = np.array(df.Close.rolling(21).corr(df.seq))
df['corr45'] = np.array(df.Close.rolling(45).corr(df.seq))
df['corr60'] = np.array(df.Close.rolling(60).corr(df.seq))
df['corr90'] = np.array(df.Close.rolling(90).corr(df.seq))
df['corr180'] = np.array(df.Close.rolling(180).corr(df.seq))
df['dolar_varOpenClose'] = df.dolar_close - df.dolar_open
df['dolar_varMaxMin'] = df.dolar_max - df.dolar_min
for i in range(1, 90):
    df['varHighDm'+str(i)] = df.Close - df.High.shift(periods=i)
    df['varCloseDm'+str(i)] = df.Close - df.Close.shift(periods=i)
    df['dolar_varHighDm'+str(i)] = df.dolar_close - df.dolar_max.shift(periods=i)
    df['dolar_varCloseDm'+str(i)] = df.dolar_close - df.dolar_close.shift(periods=i)

```

A coluna **varHighDm[x]** representa a variação entre o valor de fechamento da ação no dia em análise e o valor máximo de **[x]** dias anteriores, o que é realizado para até 90 dias que antecedem cada registro. O mesmo ocorre para **varCloseDm[X]**, **dolar\_varHighDm[X]** e **dolar\_varCloseDm[X]**.

Como exemplo, o código abaixo mostra os últimos 3 registros do dataset com as colunas Close, High, varHighDm1.

```
df[['Close', 'High', 'varHighDm1']].tail(3)
```

	Close	High	varHighDm1
Date			
2020-11-27	24.250000	24.549999	-0.049999
2020-11-30	23.379999	24.690001	-1.170000
2020-12-01	22.879999	23.770000	-1.810001

As colunas **corr9**, **corr21**, **corr45**, **corr60**, **corr90** e **corr180** apresentam a tendência de alta ou baixa da ação para um período de 9, 21, 45, 60, 90 e 180 dias. Para representar essa tendência, foi utilizada a correção entre a média móvel e um número sequencial, representado pela coluna **seq**. Sendo assim, se a média móvel

está crescendo no mesmo sentido do número sequencial **seq**, a correlação entre ambos é positiva.

Analisando a **corr60** com o valor histórico da ação MGLU3, no gráfico a seguir, verificamos que **corr60** consegue representar os momentos de tendência de alta e baixa para o valor da ação.



**Fig. 10** – Valor histórico da ação MGLU3 (em azul) e nova variável corr60 (em cinza)

Diversas outras variáveis também podem ser monitoradas na decisão de compra de ativos de empresas, porém decidiu-se seguir com essas variáveis e avaliar se são suficientes para representar o problema. Exemplos de outras variáveis são valores de suporte e resistência de compra/venda, outros valores de médias móveis, se historicamente a ação vem sofrendo suporte ou resistência em relação ao valor da média móvel, entre outros.

## 4.2 Normalização dos dados

Todas variáveis dentro do dataset **df**, incluindo as que foram criadas nessa etapa, precisam ser normalizadas, para que sua real contribuição ao problema seja coerente. Essa etapa é apresentada no código a seguir.



```

#normalização
dfP = pd.DataFrame()
dfP['High'] = df['High']/df['Close'] - 1
dfP['Low'] = df['Low']/df['Close'] - 1
dfP['Open'] = df['Open']/df['Close'] - 1
dfP['Close'] = df['Close']/df['Close'] - 1
dfP['Volume'] = df['Volume']/df['Volume'].rolling(9).mean() - 1
dfP['varOpenClose'] = df['varOpenClose']/df['Close']
dfP['varHighLow'] = df['varHighLow']/df['Close']
dfP['mean9'] = df['mean9']/df['Close'] - 1
dfP['mean21'] = df['mean21']/df['Close'] - 1
dfP['closeGreaterM9'] = df['closeGreaterM9']
dfP['closeGreaterM21'] = df['closeGreaterM21']
dfP['M9GreaterM21'] = df['M9GreaterM21']
dfP['varCloseM9'] = df['varCloseM9']/df['Close']
dfP['varCloseM21'] = df['varCloseM21']/df['Close']
dfP['varM9M21'] = df['varM9M21']/df['Close']
dfP['corr9'] = df['corr9']
dfP['corr21'] = df['corr21']
dfP['corr45'] = df['corr45']
dfP['corr60'] = df['corr60']
dfP['corr90'] = df['corr90']
dfP['corr180'] = df['corr180']
dfP['dayweek'] = df.index.dayofweek
dfP['dolar_varOpenClose'] = df['dolar_varOpenClose']/df.dolar_max.max()
dfP['dolar_varMaxMin'] = df['dolar_varMaxMin']/df.dolar_max.max()
for i in range(1,90):
    dfP['varHighDm'+str(i)] = df['varHighDm'+str(i)]/df['Close']
    dfP['varCloseDm'+str(i)] = df['varCloseDm'+str(i)]/df['Close']
    dfP['dolar_varHighDm'+str(i)] = df['dolar_varHighDm'+str(i)]/df.dolar_max.max()
    dfP['dolar_varCloseDm'+str(i)] = df['dolar_varCloseDm'+str(i)]/df.dolar_max.max()

```

Dessa forma, um novo dataframe **dfP** é criado. Os DataFrames **df** e **dfP** são relacionados entre si, um contem valores absolutos, outro valores normalizados, respectivamente.

### 4.3 Coluna Target

Outra etapa fundamental para modelos de *machine learning* supervisionados é a criação de uma coluna target no dataset. Em problemas que isso é possível, faz com que os modelos e resultados obtidos possam ser explorados com maior facilidade. Assim, o seguinte código cria a coluna **targetH**, na qual o valor 1 representa registros que historicamente se comportaram como o que se deseja prever nesse trabalho.

```

#target
targetH = np.array(range(0,len(dfP.index)))
i = 0
for idfP in dfP.index:
    targetH[i] = 0
    count = 1
    while count <= param.DaysAhead:
        i2 = i + count
        if i2 >= len(dfP.index): break
        if df.iloc[i2]['High']/df.iloc[i]['Close'] >= 1+param.gain: targetH[i] = 1
        count += 1
    i += 1
dfP['targetH'] = targetH

```

A coluna **targetH** é definida como atingimento do valor **param.gain** (em porcentagem) através do máximo valor da ação para os próximos **param.DaysAhead** dias, ou seja, caso a ação em algum momento atinja o ganho esperado dentro do prazo definido, **targetH** será 1, caso contrário será 0.

Com isso, um novo datafarme (**dfP**) surge, contendo todas as informações necessárias para aplicação de modelos de *machine learning*. Uma amostragem do novo dataframe **dfP** é apresentado logo a seguir. A coluna Close é ignorada, pois dentro de **dfP** todos dados estão normalizados em relação a Close.

```
dfP.sample(10).drop('Close', axis=1)
```

Date	High	Low	Open	Volume	varOpenClose	...	varHighDm86	varHighDm87	varHighDm88	varHighDm89	targetH
2018-03-12	0.009937	-0.015990	-0.004106	-0.148353	0.004106	...	0.322747	0.271011	0.276410	0.249297	0
2018-10-03	0.008325	-0.016792	-0.016074	1.103360	0.016074	...	0.167719	0.210563	0.208483	0.202454	1
2020-10-26	0.024735	-0.014134	0.012956	-0.261027	-0.012956	...	0.297409	0.292403	0.289949	0.295838	1
2018-07-05	0.041667	-0.005333	0.041667	-0.033165	-0.041667	...	0.225000	0.226667	0.225000	0.210000	1
2017-09-12	0.049697	-0.078238	-0.000287	1.262239	0.000287	...	0.506484	0.511009	0.506322	0.492899	1
2019-09-16	0.000000	-0.060224	-0.030685	-0.116640	0.030685	...	0.368225	0.336822	0.335927	0.323487	1
2020-11-26	0.002062	-0.017732	-0.014433	-0.420190	0.014433	...	0.119588	0.112784	0.077320	0.101443	0
2020-05-15	0.020316	-0.015055	-0.002358	-0.406165	0.002358	...	0.086704	0.107564	0.105569	0.105206	1
2019-10-10	0.026708	-0.003595	0.019517	-0.127181	-0.019517	...	0.343862	0.362962	0.377247	0.366975	1
2019-05-10	0.011904	-0.029169	0.002843	-0.298323	-0.002843	...	0.002733	0.029114	0.039141	0.023807	0

10 rows × 111 columns

As linhas que possuem **targetH=1** representam dias em que o valor de fechamento foi superado em 10% em até 15 dias posteriores. Como exemplo, tomando a segunda linha do dataset acima, no dia 03/10/2018 o valor da ação no fim do dia foi superado em pelo menos 10% até o dia 18/10/2018. Essa abordagem

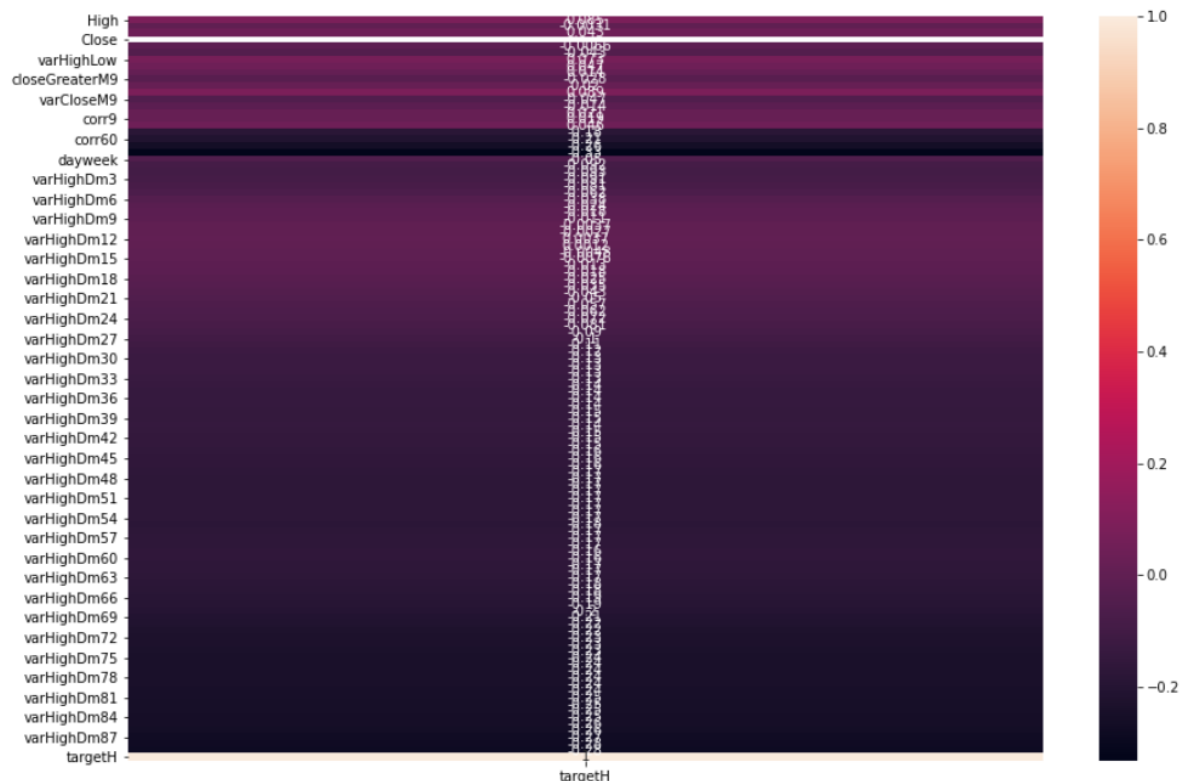
não define quando nem o valor exato do valor futuro da ação, o que é coerente, uma vez que o objetivo desse trabalho é recomendar a compra de ações que possuem potencial e não a definição exata de valores futuros. Portanto, `targetH=1` mostra dias em que essa ação deveria ter sido comprada no passado, pois em até 15 dias futuros, ela cresceu pelo menos 10% em seu valor.

#### 4.4 Exploração dos dados adicional

O gráfico a seguir mostra um mapa de calor (heatmap) entre a nova coluna **targetH** e as demais colunas do dataset **dfP** para a ação MGLU3, excluindo variáveis com correlações inferiores a 0.1.

```
corrdfP = dfP.corr(method='pearson', min_periods=1)[['targetH']]
plt.figure(figsize=(15, 10))
sns.heatmap(corrdfP, annot=True)
```

<AxesSubplot:>



**Fig. 11** – HeatMap da correlação entre a coluna **targetH** do dataset **dfP** e suas outras colunas para ação MGLU3

Observamos que não existe uma única variável que explica o comportamento da coluna **targetH**, porém as variáveis **varHighDm[x]** mais próximas a 9 dias do

instante de tempo de **Date** possuem maior correção que as demais. Também as variáveis **varHighLow**, **M9GreaterM21** e **Open**, possuem uma correlação um pouco acima das demais.

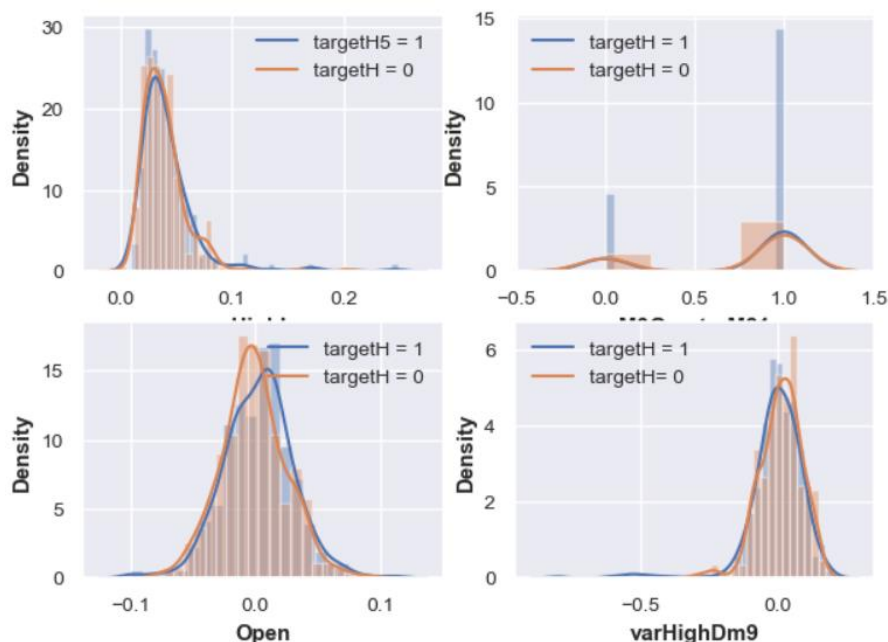
O código a seguir gera um gráfico que separa o conjunto de dados com **targetH=0** e **targetH=1**, para os quais, curvas de densidade são plotadas sobrepostas com o objetivo de entender se essas variáveis citadas podem ter alta significância para o problema.

```
plt.figure(figsize=(7, 5))
sns.set(font_scale=0.9)
plt1 = plt.subplot(2,2,1)
sns.distplot(dfP.varHighLow[dfP.targetH==1])
sns.distplot(dfP.varHighLow[dfP.targetH==0])
plt1.legend(['targetH = 1', 'targetH = 0'])

plt2 = plt.subplot(2,2,2)
sns.distplot(dfP.M9GreaterM21[dfP.targetH==1])
sns.distplot(dfP.M9GreaterM21[dfP.targetH==0])
plt2.legend(['targetH = 1', 'targetH = 0'])

plt3 = plt.subplot(2,2,3)
sns.distplot(dfP.Open[dfP.targetH==1])
sns.distplot(dfP.Open[dfP.targetH==0])
plt3.legend(['targetH = 1', 'targetH = 0'])

plt4 = plt.subplot(2,2,4)
sns.distplot(dfP.varHighDm9[dfP.targetH==1])
sns.distplot(dfP.varHighDm9[dfP.targetH==0])
plt4.legend(['targetH = 1', 'targetH = 0'])
```



**Fig. 12** – Curvas de densidade para algumas variáveis do dataset, usando a ação MGLU3

Verificamos que isoladamente nenhuma das variáveis explica significativamente o problema exposto. A variável **Open** mostra uma leve distorção para cada um dos grupos (targetH=0 e targetH=1), porém não os separa com significância.

Como novas variáveis foram criadas e algumas dependem de dados históricos, como **corr[x]** e **varHighDm[x]**. Logo abaixo é feita uma análise dos dados que possuem NaN e estes são retirados. A quantidade de dados NaN depende exclusivamente das variáveis **corr[x]** e **varHighDm[x]**.

```
pd.DataFrame({'Count': dfP.isna().count(), 'NaN': dfP.isna().sum()})
```

	Count	NaN
High	894	0
Low	894	0
Open	894	0
Close	894	0
Volume	894	8
...	...	...
varHighDm86	894	86
varHighDm87	894	87
varHighDm88	894	88
varHighDm89	894	89
targetH	894	0

```
dfP = dfP.dropna()
```

## 4.5 Conversão de dados categóricos

Colunas que representam categorias, como dia da semana, sexo, cores, ou qualquer tipo de informação que, mesmo numérica, represente grupos e não ordem, devem ser tratadas de forma especial para que os algoritmos de *machine learning* possam funcionar adequadamente.

O código abaixo mostra a transformação de algumas colunas em categorias, uma vez que isso é importante para um correto funcionamento dos algoritmos de *machine learning*.

```
dfP[['dayweek', 'closeGreaterM9', 'closeGreaterM21', 'M9GreaterM21']] = dfP[['dayweek', 'closeGreaterM9', 'closeGreaterM21', 'M9GreaterM21']].astype('category')
dfDum = pd.get_dummies(dfP[['dayweek', 'closeGreaterM9', 'closeGreaterM21', 'M9GreaterM21']], drop_first=False)
dfP = pd.concat([dfP, dfDum], axis=1)

dfP.tail(2)
```

	High	Low	Open	Close	Volume	varOpenClose	varHighLow	mean9	mean21	closeGreaterM9	closeGreaterM21	M9GreaterM21	varCloseM9	varCloseM21	var
2020-11-30	0.056031	-0.006843	0.035073	0.0	0.559725	-0.035073	0.062874	0.030843	0.069575	False	False	False	-0.030843	-0.069575	-0.030843
2020-12-01	0.038899	-0.012238	0.033654	0.0	0.096157	-0.033654	0.051136	0.044386	0.087787	False	False	False	-0.044386	-0.087787	-0.044386

O código acima transforma um dado categórico em diversas colunas, conforme o exemplo abaixo, o que garante que não exista alguma relação de ordem na informação.

Data	Dia da Semana	→ Transformação	Data	Dia da semana 1	Dia da semana 2
06/12/20	1		06/12/20	1	0
07/12/20	2		07/12/20	0	1

## 4.6 Frequência de acerto

Por fim, uma análise da frequência do valor target é observada.

```
freq = pd.DataFrame(columns=['metric', '0', '1'])
freq = freq.append({'metric': 'targetH', '0': dfP.targetH[dfP.targetH==0].count(), '1': dfP.targetH[dfP.targetH==1].count(), ignore_index=True})
freq_p = freq[freq.metric=='targetH']['1'].values/np.sum(freq[freq.metric=='targetH']['0', '1'].values)
print(freq)
print('Frequência targetH: %.2f%%' % (100*freq_p))
```

```
metric    0    1
0 targetH  331  390
Frequência targetH: 54.09%
```

Abaixo é apresentado um resumo da frequência para cada uma das ações estudadas, buscando um ganho de 10% em até 15 dias.

Ação	MGLU3	VALE3	PETR4
Frequência TargetH para ganho 10% e prazo 15 dias	54%	28%	28%

A frequência de **TargetH=1** mostra que o evento esperado ocorre 54% para MGLU3, 28% para VALE3 e 28% para PETR4. Essas porcentagens são importantes para se garantir um dataset balanceado. Se essas porcentagens forem muito baixas, seria necessário a rebalanceamento do dataset usando técnicas como a geração de dados sintéticos ou a redefinição dos parâmetros de entrada, alterando o ganho desejado e o prazo para o mesmo, pois cada ação tem um comportamento histórico diferente.

Caso o ganho esperado fosse reduzido para 7% e seu prazo mantido 15 dias, por exemplo, a nova frequência de **TargetH** para cada ação (MGLU3, VALE3 e PETR4), conforme apresentado abaixo.

Ação	MGLU3	VALE3	PETR4
Frequência TargetH para ganho 7% e prazo 15 dias	68%	45%	43%

O valor do ganho esperado (**param.gain**) pode ser personalizado para cada ação, uma vez que o crescimento histórico de cada uma é muito particular. Na busca por uma comparação honesta, o ganho fica definido como 10% para as 3 ações. Pois o valor de 28% de frequência para VALE3 e PETR4 mostra um desbalanceamento fraco, que não afeta significativamente o problema a ser resolvido. O que será investigado na sessão de resultados.

O valor da frequência de **TargetH=1** também mostra o acerto em caso de compras das ações de forma aleatória, sem uma estratégia definida. Ou seja, os resultados dos modelos de *machine learning* precisam ser consideravelmente melhores que essa frequência.

## 5. Criação de Modelos de Machine Learning

Antes de buscar modelos de *machine learning* para resolução do problema, precisamos entender bem as métricas que devem ser avaliadas para o trabalho proposto.

### 5.1 Introdução e métricas

Uma das principais ferramentas para se avaliar os resultados de um modelo de *machine learning* é a matriz de confusão. Nela são representados os verdadeiros positivos (TP), falsos positivos (FP), falsos negativos (FN) e verdadeiros negativos (TN) gerados pelo modelo desenvolvido. Uma representação genérica é mostrada na figura abaixo.

		Valor Predito	
		Sim	Não
Real	Sim	Verdadeiro Positivo (TP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (TN)

**Fig. 13** – Matriz de confusão, fonte: Nogare (2020)

Normalmente o dataset utilizado em modelos de *machine learning* são separados em uma proporção, normalmente de 80/20% ou 70/30%, na qual a maior parcela é usada para treinamento do modelo (dataset de treinamento) e a menor parcela para testar o modelo (dataset de teste). O dataset de teste é importante para garantir que não ocorreu *overfit* durante o treinamento do modelo.

*Overfit*, ocorre quando um modelo de *machine learning* consegue uma boa métrica durante o treinamento, porém não performa adequadamente durante seu uso em novos dados. Assim, o dataset de teste é um conjunto de dados ainda não observado pelo modelo, ou seja, é usado para simular o uso do algoritmo em novos dados.

Para criação dos datasets de treinamento e testes, foi utilizada a biblioteca **sklearn**, conforme código a seguir.



```

from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score, roc_curve, classification_report, accuracy_score
from sklearn.metrics import recall_score, precision_score, f1_score, confusion_matrix, auc

dfP = dfP.dropna()
for col in dfP.columns: dfP[col] = dfP[col].astype(float)
y = dfP.targetH
X = dfP.drop(['targetH'], axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

X_new = dfP.drop(['targetH'], axis=1).tail(1)
for col in X_new.columns: X_new[col] = X_new[col].astype(float)

```

Com isso, os seguintes datasets são gerados:

- **X\_train**: dados de entrada para treinamento
- **X\_test**: dados de entrada para testes
- **y\_train**: dados de saída para treinamento
- **y\_test**: dados de saída para testes
- **X\_new**: último registro do dataset, o qual deve ser usado para gerar a predição desejada.

A matriz de confusão pode ser criada tanto para o dataset de treinamento quanto para o dataset de teste.

As principais métricas para se medir a performance de um modelo de machine learning são (NOGARE, 2020):

- **Acurácia** (Acuracy): Quantidade classificada como Positivos e Negativos corretamente, e pode ser formalizada em  $(TP + TN) / (TP + TN + FP + FN)$
- **Precisão** (Precision): Quantidade Positiva classificada corretamente. E é calculada por  $TP / (TP + FP)$
- **Recall**: Taxa de valores classificada como Positivo, comparada com quantos deveriam ser. E pode ser calculada como  $TP / (TP + FN)$
- **F1 SCORE**: É calculado como a média harmônica entre Precisão e Recall, sendo sua formulação matemática representada por  $(2 * TP) / (2 * TP + FP + FN)$

Nesse trabalho a principal métrica é a Precisão, na qual mostra a quantidade de previsões que foram corretamente feitas. Um alto número dessa métrica indica

que os investimentos feitos utilizando a recomendação do modelo trouxeram o retorno esperado.

A métrica secundária é o Recall, pois esse indica a quantidade de eventos que ocorreram e que foram capturados pelo modelo. Para o presente trabalho não se busca um alto número de Recall, mas sim de Precisão, uma vez que se deseja reduzir os riscos do investimento. Um alto número de Recall, indicaria que não se deseja perder oportunidades de investimento, ou seja, deseja-se capturar a maior quantidade possível de oportunidades de ganho.

Mesmo assim, o Recall precisa ser monitorado, pois valores muito baixos podem mostrar uma ineficiência do modelo de *machine learning*.

A seguir são exploradas quatro técnicas de *machine learning*, as quais podem ser usadas para resolver o problema apresentado. As técnicas são regressão logística, árvore de decisão, redes neurais artificiais e XGBoost.

Para todas as técnicas apresentadas, a ação MGLU3 foi utilizada na primeira análise, a qual possui uma frequência de 54% para **targetH=1**, buscando um ganho de 10% em até 15 dias. E para comparar o resultado obtido com o uso dessas técnicas, o seguinte DataFrame foi criado.

```
y_pred = pd.DataFrame(columns=('stock', 'freq', 'model', 'precTest', 'recallTest', 'precAll', 'recallAll', 'targetH'))
y_pred['model'] = ['Regressao', 'Arvore', 'RNA', 'XGBoost']
y_pred['freq'] = float(freq_p)
y_pred['stock'] = param.stock_list[stock][:5]
```

A partir do DataFrame **y\_pred** será possível comparar os valores da Precisão e Recall das diferentes técnicas. A coluna **stock** representa o nome da ação, **freq** a frequência de **targetH=1** no dataset completo, **model** o nome do modelo, **precTest** a precisão do modelo no dataset de teste, **recallTest** o recall do modelo no dataset de teste, **precAll** a precisão do modelo no dataset completo, **recallAll** o recall do modelo no dataset completo.

A coluna **targetH** indica o valor da predição do modelo para a entrada de dados **X\_new**, a qual representa o último registro da ação e **targetH** representa uma previsão se essa ação irá crescer nos próximos dias. A principal saída do modelo é **targetH**, valor igual a 1 representa a recomendação de compra da ação e 0 a não recomendação.

## 5.2 Regressão Logística

As técnicas de regressão estão dentre as mais utilizadas quando se deseja entender as correlações entre variáveis de entrada e saída de um modelo, normalmente são as primeiras a serem testadas em modelos de *machine learning* (HOSMER et al., 2013).

A regressão logística pode ser aplicada ao problema aqui proposto, uma vez que se busca uma variável categórica como saída, **targetH=1** ou **targetH=0**, ou seja, um novo conjunto de dados faz parte ou não do conjunto observado e desejado.

Para aplicar a regressão logística, foi utilizada a biblioteca **sklearn**, na qual foram usadas inicialmente todas as colunas como dados de entrada. O algoritmo a seguir usa a biblioteca **statsmodels** para investigação das variáveis mais significativas ao modelo.

```
import statsmodels.api as sm
import statsmodels.formula.api as smf
from sklearn.linear_model import LogisticRegression

all_col_with_plus = ' + '.join(dfP.drop(['targetH'], axis=1).columns)
modelo = smf.glm(formula='targetH ~ ' + all_col_with_plus, data=dfP,
                  family = sm.families.Binomial()).fit()
print(modelo.summary())
```

O resultado logo abaixo apresenta uma amostragem do sumário do modelo obtido. É apresentado apenas o sumário, pois o modelo possui mais de 100 variáveis de entrada.

Generalized Linear Model Regression Results						
=====						
Dep. Variable:	targetH	No. Observations:	715			
Model:	GLM	Df Residuals:	606			
Model Family:	Binomial	Df Model:	108			
Link Function:	logit	Scale:	1.0000			
Method:	IRLS	Log-Likelihood:	-378.85			
Date:	Sun, 06 Dec 2020	Deviance:	757.70			
Time:	17:59:14	Pearson chi2:	667.			
No. Iterations:	100					
Covariance Type:	nonrobust					
=====						
	coef	std err	z	P> z	[0.025	0.975]
-----						
Intercept	2.0745	0.323	6.430	0.000	1.442	2.707
High	-0.7074	8.181	-0.086	0.931	-16.742	15.327
Low	7.7166	6.909	1.117	0.264	-5.825	21.258
Open	-1.1691	4.463	-0.262	0.793	-9.917	7.579
Close	-5.966e-14	5.38e-14	-1.108	0.268	-1.65e-13	4.58e-14
Volume	-0.1414	0.299	-0.473	0.636	-0.728	0.445
varOpenClose	1.1691	4.463	0.262	0.793	-7.579	9.917
varHighLow	-8.4240	4.626	-1.821	0.069	-17.492	0.644
mean9	6.1448	8.825	0.696	0.486	-11.151	23.441
mean21	-28.1691	12.520	-2.250	0.024	-52.708	-3.631

Buscando variáveis de maior significância para o problema, observa-se que apenas as seguintes variáveis possuem *p* valor menor que 5%:

- mean21
- varCloseM21
- varM9M21
- corr60
- corr180
- varHighDm7
- varHighDm9
- varHighLow (próximo a 5%)

Mesmo que essas variáveis sejam as que melhor contribuem para a regressão logística, observou-se que ao usar apenas essas variáveis como entrada, a Precisão e Recall ficaram inferiores daqueles obtidos com o uso de todas as variáveis. Portanto, optou-se por manter todas as variáveis no modelo.

A seguir é apresentado o código para treinamento da regressão logística usando a biblioteca **sklearn**.

```

model = LogisticRegression(penalty='none', solver='newton-cg')

model.fit(X_train, y_train)
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                    intercept_scaling=1, l1_ratio=None, max_iter=100,
                    multi_class='warn', n_jobs=None, penalty='none',
                    random_state=None, solver='newton-cg', tol=0.0001, verbose=0,
                    warm_start=False)
print(model.coef_)

```

Os diversos parâmetros da regressão logística foram variados a fim de se obter aqueles que melhor geravam resultados nos testes. Logo abaixo são apresentados os coeficientes das variáveis de entrada gerado pelo modelo.

```

[[ 7.99070294e+00  1.63697459e+01 -2.21506565e+00  0.00000000e+00
 -3.54325844e-02  2.21506565e+00 -8.37904299e+00  9.51544661e+00
 -3.98272851e+01  6.88563124e-01  9.57810642e-01  8.04841759e-01
 -9.51544661e+00  3.98272851e+01  4.93427317e+01  1.68648505e-01
 1.09165543e+00  1.53919572e+00 -1.83642986e-01 -6.73137081e-01
 -8.98156320e+00  2.69253116e-01  1.31324721e+01 -1.25389816e+01
 2.05683050e+00  3.62450516e+00 -2.26405385e+00  8.18301517e+00
 5.88499878e+00 -4.53381969e+00  3.10778488e+00 -1.90846847e+01
 -8.29718454e+00 -9.24123174e+00 -4.66580526e+00  3.34502621e+00
 -4.77114901e+00 -2.09758584e+01  1.17425112e+01 -1.45167233e+01
 3.20108338e+00 -2.25463859e+01  8.46846452e+00 -2.24958321e+00
 7.07241131e+00 -1.11317239e+01  8.58710562e+00 -2.78079567e+00
 1.04481638e+01 -7.23737293e+00 -1.00043704e+01  1.73837272e+00
 -7.89301735e+00 -2.27776340e+00  1.79538493e+01 -1.70344685e+01
 5.94949030e+00 -4.83725920e+00  3.17593392e+00 -4.51454500e+00
 1.92605638e+00  2.57701867e+00  1.67905034e+00 -9.28782434e+00
 1.21675568e+01 -1.65047848e+01  5.32807660e+00  1.03072938e+01
 -5.18804121e+00  4.44694551e+00 -5.73750401e+00 -9.38404901e+00
 4.10623345e+00 -1.03247137e+01  1.31631105e+01 -2.58801454e+00
 -1.10192089e+01  1.43982779e+01 -6.34499522e+00 -9.61155431e+00
 1.56324788e+01 -7.94256260e-01  2.36443424e+00  3.25555801e+00
 -4.85128739e+00  6.29219986e+00 -1.08537540e+01  9.38711187e+00
 8.36000960e-01  6.72475661e+00 -3.51075042e+00  6.72251232e+00
 -9.49034785e+00  1.84753263e+00 -1.02139110e+01  4.73731852e+00
 -6.24949599e+00 -7.32753042e-01 -7.26858589e-01  9.32493683e+00
 -1.07218947e+01 -7.43873855e-01  5.94646989e+00  3.49698082e+00
 -1.05822121e+01  1.60808044e+01 -1.06661790e+01  1.12965126e+01
 -3.93274900e+00 -1.29353795e+01  6.93306441e+00  1.43752520e+00
 8.02502465e-01  6.53156526e-01 -1.74261024e-01 -3.29194832e-01
 1.70116521e+00  6.88563124e-01  1.43191769e+00  9.57810642e-01
 1.58488658e+00  8.04841759e-01]]

```

A discussão de quais variáveis mais contribuem não é o objetivo desse trabalho pois este não é um modelo fundamentalista, mas sim “baseado em gráfico”. Portanto, maiores discussões serão geradas comparando as diferentes técnicas e optou-se por explorar os coeficientes da regressão logística apenas se ela for a melhor técnica a ser aplicada.

A seguir é apresentada a matriz de confusão aplicando o modelo aos dados de testes, assim como as métricas de Precisão, Recall e F1-Score para os datasets de treinamento e teste.

```
print('- Matriz de Confusão')
print(confusion_matrix(y_test, model.predict(X_test)))
print('\n- Reporte completo')
print(classification_report(y, model.predict(X)))
print('\n- Reporte teste')
print(classification_report(y_test, model.predict(X_test)))
```

- Matriz de Confusão

```
[[49 49]
```

```
[46 71]]
```

- Reporte completo

	precision	recall	f1-score	support
0.0	0.68	0.68	0.68	325
1.0	0.73	0.74	0.73	390
accuracy			0.71	715
macro avg	0.71	0.71	0.71	715
weighted avg	0.71	0.71	0.71	715

- Reporte teste

	precision	recall	f1-score	support
0.0	0.52	0.50	0.51	98
1.0	0.59	0.61	0.60	117
accuracy			0.56	215
macro avg	0.55	0.55	0.55	215
weighted avg	0.56	0.56	0.56	215

Para o dataset de teste, observa-se uma precisão de 59% para **targetH=1**, levemente acima da frequência de **targetH=1** (54% para MGLU3), apresentada anteriormente, porém ainda não é possível avaliar se esse é um bom resultado, maiores comparações são necessárias e apresentadas no próximo capítulo.

O modelo treinado pode ser aplicado em um novo dataset usando o comando `model.predict(X)`. A seguir os resultados obtidos para predição do dataset **X\_new** são armazenados dentro do DataFrame **y\_pred**.

```

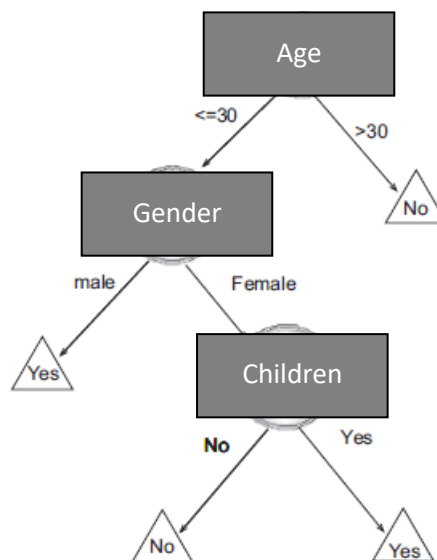
y_pred.targetH[y_pred.model=='Regressao'] = model.predict(X_new)
y_pred.H_precAll[y_pred.model=='Regressao'] = precision_score(y, model.predict(X))
y_pred.H_precTest[y_pred.model=='Regressao'] = precision_score(y_test, model.predict(X_test))
y_pred.H_recallAll[y_pred.model=='Regressao'] = recall_score(y, model.predict(X))
y_pred.H_recallTest[y_pred.model=='Regressao'] = recall_score(y_test, model.predict(X_test))

```

### 5.3 Árvore de Decisão

A árvore de decisão é um modelo que gera uma decisão baseada em cortes hierárquicos, onde uma decisão gera uma consequência (ROKACH and MAIMON, 2014). A hierarquia na tomada de decisão é muito baseada na forma como criamos procedimentos.

Fig. 14 mostra um exemplo de hierarquia na tomada de decisão para definir a urgência em um determinado exame médico.



**Fig. 14** – Exemplo de árvore de decisão para definir a urgência em um determinado exame médico.  
FONTE: Rokach; Maimon (2007)

Observamos na Fig. 14 que homens com menos de 30 anos de idade e mulheres abaixo dos 30 anos que possuem filhos são os casos de maior urgência para se realizar o exame.

E para aplicação da árvore de decisão no problema desse trabalho, foi utilizada a biblioteca **sklearn**. O código utilizado para treinamento do modelo de *machine learning* e a geração da figura que mostra a árvore gerada é apresentado

logo a seguir. Ademais, foram testados diversos parâmetros do modelo e a configuração que gerou o melhor resultado é o apresentado a seguir, parâmetros não apresentados indicam que o valor *default* obteve o melhor resultado.

```
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split
from sklearn import metrics, tree

model = DecisionTreeClassifier(criterion="entropy", max_depth=5)
model = model.fit(X_train,y_train)
fig = plt.figure(figsize=(20,10))
_ = tree.plot_tree(model,
                    feature_names=X.columns,
                    class_names=['targetNo', 'targetYes'],
                    filled=True)
```

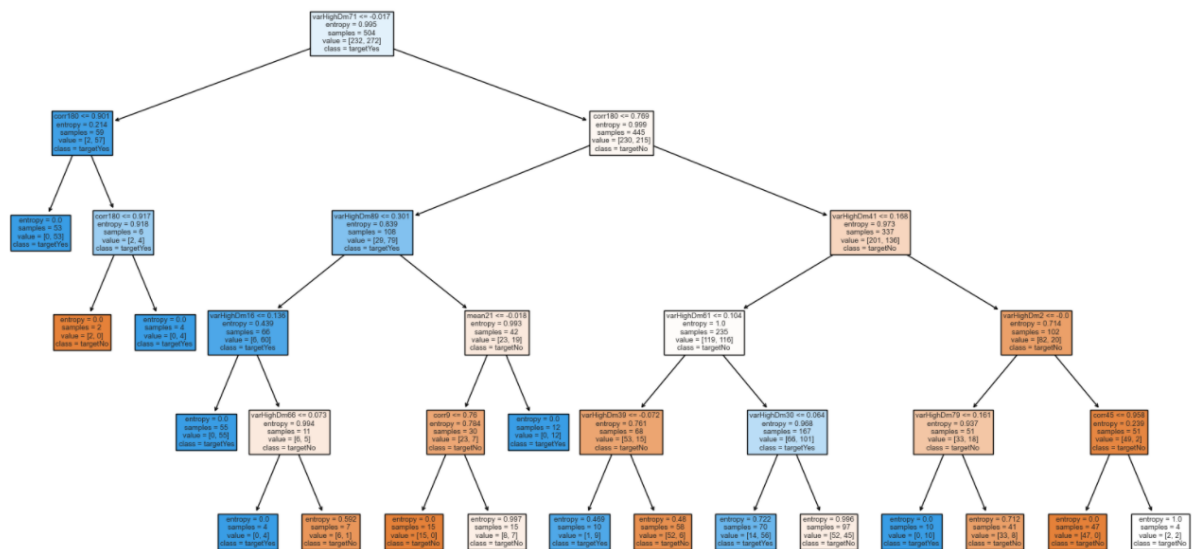


Fig. 15 – Árvore de decisão gerada para MGLU3

Observou-se que alguns parâmetros, como **max\_depth**, ao serem alterados podem gerar o *overfit* do modelo, o que significa que o algoritmo gera uma alta precisão para o dataset de treinamento e uma baixa aderência para o dataset de teste.

As discussões sobre a árvore de decisão gerada e seus cortes somente é feita nesse trabalho no caso em que esse método apresente o melhor resultado entre as técnicas utilizadas.



Para aplicação do modelo criado, podemos utilizar a função **model.predict()** e a seguir são apresentadas as matrizes de confusão para o dataset completo e o dataset de teste.

```
print('- Matriz de Confusão')
print(confusion_matrix(y_test, model.predict(X_test)))
print('\n- Reporte completo')
print(classification_report(y, model.predict(X)))
print('\n- Reporte teste')
print(classification_report(y_test, model.predict(X_test)))
```

- Matriz de Confusão

```
[[76 23]
 [45 73]]
```

- Reporte completo

	precision	recall	f1-score	support
0.0	0.72	0.89	0.80	331
1.0	0.88	0.71	0.79	390
accuracy			0.79	721
macro avg	0.80	0.80	0.79	721
weighted avg	0.81	0.79	0.79	721

- Reporte teste

	precision	recall	f1-score	support
0.0	0.63	0.77	0.69	99
1.0	0.76	0.62	0.68	118
accuracy			0.69	217
macro avg	0.69	0.69	0.69	217
weighted avg	0.70	0.69	0.69	217

Para o dataset de teste, observa-se uma precisão de 76% para **targetH=1**, a qual é acima da frequência de **targetH=1** (54% para MGLU3) e melhor que a obtida pela regressão logística (59%).

O modelo treinado pode ser aplicado em um novo dataset usando o comando **model.predict(X)**. A seguir os resultados obtidos para predição do dataset **X\_new** são armazenados dentro do DataFrame **y\_pred**.

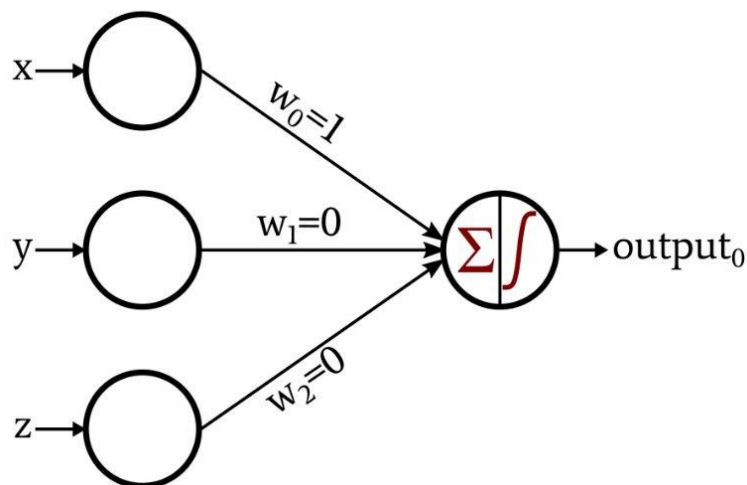
```
y_pred.targetH[y_pred.model=='Arvore'] = model.predict(X_new)
y_pred.H_precAll[y_pred.model=='Arvore'] = precision_score(y, model.predict(X))
y_pred.H_precTest[y_pred.model=='Arvore'] = precision_score(y_test, model.predict(X_test))
y_pred.H_recallAll[y_pred.model=='Arvore'] = recall_score(y, model.predict(X))
y_pred.H_recallTest[y_pred.model=='Arvore'] = recall_score(y_test, model.predict(X_test))
```

## 5.4 Redes Neurais Artificiais

As redes neurais artificiais (RNA) se tornaram a técnica de *machine learning* mais conhecida recentemente, seu princípio simula o mecanismo biológico de aprendizado no qual neurônios são conectados entre si por uma ligação que pode ser mais ou menos intensa (AGGARWAL, 2018).

Existem diversas topologias de RNA e algumas são mais bem aplicadas para determinados tipos de problemas, como as redes convolucionais que são muito aplicadas para problemas de visão computacional.

O funcionamento básico de uma RNA pode ser visto na figura a seguir, na qual um perceptron é apresentado.



**Fig. 16** – Ilustração de um perceptron, rede neural artificial de 1 neurônio e 1 camada

Pela Fig. 16 observamos que em uma RNA são definidas as variáveis de entrada ( $x$ ,  $y$  e  $z$ ), as quais são conectadas a um neurônio por ligações que possuem pesos ( $w_0$ ,  $w_1$  e  $w_2$ ), o neurônio soma os valores que chegam pelas ligações e aplica uma função de ativação, gerando assim a saída  $output_0$ .

Para o presente trabalho foi utilizada a biblioteca **keras** do **TensorFlow** e o código a seguir foi criado para resolução do problema. Foram testadas diversas configurações da rede neural e a apresentada foi a que obteve os melhores resultados. Nessa RNA foram utilizadas 5 camadas, sendo a primeira com 15 neurônios, a segunda com 7 neurônios, as terceira e quarta camadas com 3 neurônios e a última camada a que gera o resultado, o qual é binário.

```
from keras import Sequential
from keras.layers import Dense
import keras
```

```
model = Sequential()
model.add(Dense(15, activation='relu', kernel_initializer='random_normal', input_dim=len(X.columns)))
model.add(Dense(7, activation='relu', kernel_initializer='random_normal', input_dim=len(X.columns)))
model.add(Dense(3, activation='relu', kernel_initializer='random_normal', input_dim=len(X.columns)))
model.add(Dense(3, activation='relu', kernel_initializer='random_normal', input_dim=len(X.columns)))
model.add(Dense(1, activation='sigmoid', kernel_initializer='random_normal'))
```

```
model.compile(optimizer='adam', loss='mean_squared_error', metrics=['accuracy'])
model.fit(X_train, y_train, batch_size=128, epochs=300, verbose=False)
eval_model=model.evaluate(X_train, y_train)
eval_model
```

Para configurações com menos camadas e neurônios observou-se uma redução na precisão do modelo. Já para números maiores que os utilizados, o modelo apresentou considerável *overfit*. Para as funções de ativação e outros parâmetros, os utilizados apresentaram melhores resultados.

Redes neurais com mais de uma camada são complexas de serem avaliadas e a discussão sobre os pesos definidos pode ser uma difícil tarefa. Por esse motivo, modelos como regressão logística e árvore de decisão são normalmente avaliados antes do uso de RNA quando se deseja obter fundamentos e clareza quanto aos resultados obtidos pelo modelo de *machine learning*.

De forma generalista, em problemas de previsão envolvendo dados não estruturados, como imagens, textos e vídeos, as redes neurais artificiais tendem a superar todos os outros algoritmos ou frameworks, porém em dados estruturados árvores de decisão tendem a gerarem resultados melhores. Porém, essa afirmação não é uma regra, apenas uma observação (GOMES, 2019).

Para aplicação do modelo treinado, podemos usar a função **model.predict\_classes()** e a seguir são apresentadas as matrizes de confusão para o dataset completo e o dataset de teste.

```

print('- Matriz de Confusão')
print(confusion_matrix(y_test, model.predict_classes(X_test)))
print('\n- Reporte completo')
print(classification_report(y, model.predict_classes(X)))
print('\n- Reporte teste')
print(classification_report(y_test, model.predict_classes(X_test)))

```

```

- Matriz de Confusão
[[74 25]
 [52 66]]

- Reporte completo
              precision    recall  f1-score   support

     0.0         0.71      0.91      0.80       331
     1.0         0.90      0.69      0.78       390

 accuracy                   0.79       721
 macro avg              0.81      0.80      0.79       721
 weighted avg           0.81      0.79      0.79       721

- Reporte teste
              precision    recall  f1-score   support

     0.0         0.59      0.75      0.66        99
     1.0         0.73      0.56      0.63       118

 accuracy                   0.65       217
 macro avg              0.66      0.65      0.64       217
 weighted avg           0.66      0.65      0.64       217

```

Para o dataset de teste, observa-se uma precisão de 73% para **targetH=1**, a qual é acima da frequência de **targetH=1** (54% para MGLU3) e melhor que a obtida pela regressão logística (59%), porém levemente inferior à árvore de decisão (76%).

O modelo treinado pode ser aplicado em um novo dataset usando o comando `model.predict(X)`. A seguir os resultados obtidos para predição do dataset **X\_new** são armazenados dentro do DataFrame **y\_pred**.

```

y_pred.targetH[y_pred.model=='RNA'] = model.predict_classes(X_new)
y_pred.H_precAll[y_pred.model=='RNA'] = precision_score(y, model.predict_classes(X))
y_pred.H_precTest[y_pred.model=='RNA'] = precision_score(y_test, model.predict_classes(X_test))
y_pred.H_recallAll[y_pred.model=='RNA'] = recall_score(y, model.predict_classes(X))
y_pred.H_recallTest[y_pred.model=='RNA'] = recall_score(y_test, model.predict_classes(X_test))

```

## 5.5 XGBoost

A combinação de técnicas de *machine learning*, assim como a aplicação de vários pequenos modelos em regiões distintas do *dataset* podem gerar resultados surpreendentes em alguns problemas.

O XGBoost foi proposto por Chen e Guestrin (2016) e é um algoritmo baseado em árvore de decisão e que utiliza uma estrutura de Gradient boosting. O uso desse algoritmo por cientista de dados vem crescendo recentemente e gerando excelentes resultados (GOMES, 2019).

Usando a biblioteca **xgboost**, o código a seguir mostra o algoritmo para treinamento do modelo de *machine learning*.

```
from xgboost import XGBClassifier
```

```
model = XGBClassifier(max_depth=9, scale_pos_weight=0.2, objective='binary:logitraw')
model.fit(X_train, y_train)
y_pred_temp = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred_temp)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
Accuracy: 82.03%
```

Alguns dos parâmetros do XGBoost foram testados e os utilizados foram os que apresentaram melhores resultados.

Para aplicação do modelo treinado, podemos usar a função **model.predict()** e a seguir são apresentadas as matrizes de confusão para o dataset completo e o dataset de teste.

```

print('- Matriz de Confusão')
print(confusion_matrix(y_test, model.predict(X_test)))
print('\n- Reporte completo')
print(classification_report(y, model.predict(X)))
print('\n- Reporte teste')
print(classification_report(y_test, model.predict(X_test)))

```

- Matriz de Confusão

```
[[94  5]
 [34 84]]
```

- Reporte completo

	precision	recall	f1-score	support
0.0	0.90	0.98	0.94	331
1.0	0.99	0.91	0.95	390
accuracy			0.94	721
macro avg	0.94	0.95	0.94	721
weighted avg	0.95	0.94	0.94	721

- Reporte teste

	precision	recall	f1-score	support
0.0	0.73	0.95	0.83	99
1.0	0.94	0.71	0.81	118
accuracy			0.82	217
macro avg	0.84	0.83	0.82	217
weighted avg	0.85	0.82	0.82	217

Para o dataset de teste, observa-se uma precisão de 94% para **targetH=1**, a qual é superior à frequência de **targetH=1** (54% para MGLU3) e melhor que a obtida pela regressão logística (59%), árvore de decisão (76%) e RNA (73%).

O modelo treinado pode ser aplicado em um novo dataset usando o comando `model.predict(X)`. A seguir os resultados obtidos para predição do dataset **X\_new** são armazenados dentro do DataFrame **y\_pred**.

```

y_pred.targetH[y_pred.model=='XGBoost'] = model.predict(X_new)
y_pred.H_precAll[y_pred.model=='XGBoost'] = precision_score(y, model.predict(X))
y_pred.H_precTest[y_pred.model=='XGBoost'] = precision_score(y_test, model.predict(X_test))
y_pred.H_recallAll[y_pred.model=='XGBoost'] = recall_score(y, model.predict(X))
y_pred.H_recallTest[y_pred.model=='XGBoost'] = recall_score(y_test, model.predict(X_test))

```

Os resultados preliminares mostram que o XGBoost gerou uma boa precisão e um bom recall. Porém, é necessário ainda explorar o resultado obtido para outras ações, além da MGLU3, e comparar os resultados obtidos para os diferentes modelos testados. Essa discussão é apresentada no próximo capítulo.

## 6. Apresentação dos Resultados

### 6.1 Introdução

No capítulo anterior foram apresentados os algoritmos para criação, treinamento e execução dos modelos de *machine learning* testados nesse trabalho, assim como resultados prévios obtidos para a ação MGLU3.

Nesse capítulo são apresentados os resultados obtidos para outras ações da bolsa de valores e a comparação ao utilizar os quatro modelos criados. O ganho esperado continuou mantido como pelo menos 10% em até 15 dias.

### 6.2 Resultados para MGLU3, VALE3 e PETR4, ganho de 10% em até 15 dias.

Durante a construção dos modelos de *machine learning*, o DataFrame **y\_pred** foi criado, ele armazena os resultados obtidos pelos algoritmos para a ação da bolsa de valores definida na entrada de dados.

No DataFrame **y\_pred**, a coluna **stock** representa o nome da ação, **freq** a frequência de **targetH=1** no dataset completo, **model** o nome do modelo, **precTest** a precisão do modelo no dataset de teste, **recallTest** o **recall** do modelo no dataset de teste, **precAll** a precisão do modelo no dataset completo, **recallAll** o **recall** do modelo no dataset completo. E a coluna **targetH** indica o valor da predição do modelo para a entrada de dados **X\_new**, a qual representa o último registro da ação e **targetH** representa uma previsão se essa ação irá crescer nos próximos dias. **targetH** possui valor 1 para recomendação de compra e 0 para não recomendação

Logo abaixo são apresentados os DataFrames para as ações MGLU3, VALE3 e PETR4, considerando um ganho de pelo menos 10% em até 15 dias após o investimento.

stock	freq	model	precTest	recallTest	precAll	recallAll	targetH
MGLU3	0.540915	Regressao	0.570312	0.618644	0.714286	0.74359	0
MGLU3	0.540915	Arvore	0.752475	0.644068	0.874608	0.715385	0
MGLU3	0.540915	RNA	0.725275	0.559322	0.89701	0.692308	0
MGLU3	0.540915	XGBoost	0.94382	0.711864	0.986111	0.910256	0

stock	freq	model	precTest	recallTest	precAll	recallAll	targetH
VALE3	0.28186	Regressao	0.403941	0.207071	0.605105	0.29676	0
VALE3	0.28186	Arvore	0.609756	0.189394	0.724806	0.275405	0
VALE3	0.28186	RNA	0.568966	0.5	0.800167	0.707658	0
VALE3	0.28186	XGBoost	0.928571	0.361111	0.990143	0.813697	0

stock	freq	model	precTest	recallTest	precAll	recallAll	targetH
PETR4	0.280341	Regressao	0.502538	0.240876	0.646789	0.313566	0
PETR4	0.280341	Arvore	0.6875	0.294404	0.769772	0.324685	0
PETR4	0.280341	RNA	0.638095	0.489051	0.825323	0.661972	0
PETR4	0.280341	XGBoost	0.926316	0.428224	0.987589	0.825797	0

Mesmo que o modelo apresentado nesse trabalho não seja fundamentalista, ou seja, o algoritmo apenas analisa dados históricos similar à decisão “baseada em gráfico”, é possível avaliar o motivo da recomendação de não compra das ações (**targetH=0**). Todas as simulações foram realizadas no dia 11/12/2020, dia em que várias ações da bolsa de valores estão caindo frente ao aumento de casos de covid-19 pelo mundo.

A coluna **freq** representa a porcentagem de vezes que essa ação cresceu 10% ou mais em até 15 dias. Ou seja, caso o investidor comprasse aleatoriamente a ação MGLU3, teria uma chance de 54% de ganho. Como a coluna **precTest** indica a precisão do modelo usando dados de testes, podemos afirmar que a diferença entre **precTest** e **freq** indica a real performance do modelo, pois mostra o quanto aumentamos a chance de ganho para novos casos.

### 6.3 Resultados para outras ações

As ações MGLU3, VALE3 e PETR4 foram usadas na construção e validação do modelo proposto, pois a análise detalhada de grande quantidade de ações inviabilizaria a construção desse trabalho.

Com o objetivo de validar a metodologia proposta, as seguintes ações foram avaliadas, buscando um ganho de pelo menos 10% em até 15 dias após o investi-



mento. Sendo assim, a quantidade de dataset explorados nesse trabalho é ampliada de 3 para 10, todos contendo dados históricos das ações.

```
param.stock_list = ['EMBR3.SA', 'RAIL3.SA', 'USIM5.SA', 'BOVA11.SA', 'CMIG4.SA', 'GOLL4.SA']
param.dt_begin   = ['2000-01-02', '2015-04-01', '2000-01-02', '2008-12-02', '2000-01-02', '2004-06-23']
```

A seguir são apresentados os resultados para essas ações. A definição do ganho esperado deve ser escolhida com cautela, pois a ação em questão pode nunca ter performado o valor especificado. Então a coluna **freq** nos informa a recorrência histórica desse ganho.

stock	freq	model	precTest	recallTest	precAll	recallAll
EMBR3	0.242424	Regressao	0.427536	0.162088	0.631111	0.243151
EMBR3	0.242424	Arvore	0.654321	0.291209	0.739292	0.339897
EMBR3	0.242424	RNA	0.568106	0.46978	0.772308	0.644692
EMBR3	0.242424	XGBoost	0.89404	0.370879	0.983246	0.803938

stock	freq	model	precTest	recallTest	precAll	recallAll
RAIL3	0.341424	Regressao	0.432836	0.464	0.823666	0.841232
RAIL3	0.341424	Arvore	0.73913	0.544	0.863636	0.675355
RAIL3	0.341424	RNA	0.725352	0.824	0.784274	0.921801
RAIL3	0.341424	XGBoost	0.975	0.624	0.994695	0.888626

stock	freq	model	precTest	recallTest	precAll	recallAll
USIM5	0.418167	Regressao	0.510135	0.240064	0.558681	0.275598
USIM5	0.418167	Arvore	0.681287	0.370429	0.738786	0.401914
USIM5	0.418167	RNA	0.616495	0.475358	0.739677	0.591388
USIM5	0.418167	XGBoost	0.895706	0.464229	0.980952	0.837799

stock	freq	model	precTest	recallTest	precAll	recallAll
BOVA1	0.076523	Regressao	0.272727	0.122449	0.66129	0.278912
BOVA1	0.076523	Arvore	0.777778	0.285714	0.906667	0.462585
BOVA1	0.076523	RNA	0.75	0.122449	0.958333	0.312925
BOVA1	0.076523	XGBoost	0.888889	0.163265	0.990476	0.707483

stock	freq	model	precTest	recallTest	precAll	recallAll
CMIG4	0.278498	Regressao	0.607843	0.0693512	0.697674	0.084686
CMIG4	0.278498	Arvore	0.496154	0.288591	0.569544	0.335215
CMIG4	0.278498	RNA	0.625874	0.400447	0.698364	0.482004
CMIG4	0.278498	XGBoost	0.890244	0.326622	0.984113	0.786874

stock	freq	model	precTest	recallTest	precAll	recallAll
GOLL4	0.452743	Regressao	0.553616	0.426104	0.605243	0.459613
GOLL4	0.452743	Arvore	0.63198	0.477927	0.684628	0.491468
GOLL4	0.452743	RNA	0.654135	0.667946	0.747417	0.740614
GOLL4	0.452743	XGBoost	0.915942	0.606526	0.981669	0.88339

Observamos que para todas as ações o algoritmo obteve um desempenho satisfatório, resultando em uma precisão superior a 88% e um recall superior a 20% aplicando-se o modelo XGBoost no dataset de teste. Ressaltando que o principal objetivo desse trabalho é garantir uma melhoria na precisão e manter um recall não muito baixo (<10%).

A coluna **freq** da ação BOVA11 possui valor muito baixo (<10%), isso indica que um ganho de 10% num período de 15 dias é um evento raro para essa ação. Mesmo assim o algoritmo XGBoost alcançou uma precisão de 88,9% nos dados de testes.

O algoritmo proposto nesse trabalho é genérico e gera a recomendação de compra de qualquer ação, para qualquer ganho em um determinado período. Dessa forma, a ação BOVA11 também foi avaliada buscando-se um ganho de 5% em até 15 dias e os resultados obtidos são apresentados a seguir.

stock	freq	model	precTest	recallTest	precAll	recallAll
BOVA1	0.385216	Regressao	0.571429	0.32	0.656388	0.402703
BOVA1	0.385216	Arvore	0.654822	0.573333	0.680187	0.589189
BOVA1	0.385216	RNA	0.785714	0.537778	0.86553	0.617568
BOVA1	0.385216	XGBoost	0.921875	0.524444	0.984448	0.855405

Com a nova métrica, o valor de **freq** aumenta para 38%, mostrando que um ganho de 5% em até 15 dias é algo mais recorrente nessa ação. Um ganho esperado coerente é importante para que o ganho do investimento tenha maior probabilidade de ocorrer.

A coluna **targetH** representa a recomendação de compra, sendo 1 recomendada a compra, 0 não recomendada. Para os testes realizados, o algoritmo XGBoost não recomendou a compra de nenhuma das ações avaliadas para o dia 11/12/2020.

Dessa forma, verificamos que o algoritmo XGBoost proporcionou boas métricas para precisão e recall do conjunto de dados de testes para todas as ações analisadas.

## 7. Considerações finais

Esse trabalho propõe um modelo de *machine learning* para recomendação de compra de ações na bolsa de valores. Os dados de entrada são capturados por uma API do site Yahoo Finance e também são usados dados históricos do dólar. Foram realizadas etapas de limpeza/tratamento dos dados, *feature engineering* e junção dos dados.

A análise e exploração de dados mostra que as bases obtidas são consistentes e confiáveis. Também, observamos que o valor de algumas ações, como MGLU3, possui uma mudança considerável em seu comportamento, ou seja, existe uma mudança na variação desses valores após algum período. E para se obter resultados mais precisos, dados anteriores a esse período de mudança podem ser retirados da análise, pois não representam o comportamento atual da ação.










Quatro modelos de machine learning foram avaliados, sendo eles, Regressão Logística, Árvore de Decisão, Redes Neurais Artificiais e o algoritmo XGBoost. O XGBoost apresentou os melhores resultados de precisão e recall para o dataset de teste e se tornou a referência para a recomendação de compra de uma ação.

Considerando ganhos de 10% em até 15 dias nos investimentos em ações, os resultados mostram uma precisão do algoritmo superior a 88% para os testes realizados, um ótimo resultado comparado à frequência de menos de 50% de ocorrência desse evento para a maioria das ações.

A abordagem feita nesse trabalho não é fundamentalista, portanto, discussões do motivo da recomendação não são viáveis. Ademais, o algoritmo proposto não visa a definição de compra de uma ação, mas sim uma recomendação para auxiliar o investidor e aumentar as chances de ganho. Portanto, uma análise pessoal sempre deve ser feita antes de se realizar uma compra de um ativo.

Para trabalhos futuros recomenda-se a inclusão de informações e notícias de veículos de comunicação e redes sociais, como reportagens e tweets.

Logo abaixo é apresentado o quadro “*The Machine Learning Canvas*”, proposto por Dorard (2014), o qual resume o trabalho criado.

The Machine Learning Canvas (v0.4)				
Designed for:		Designed by: Rafael Silva Pinto		Date: 12/11/2020
Iteration: 1				
<b>Decisions</b>  <p>How are predictions used to make decisions that provide the proposed value to the end-user?</p> <p>Apoio na análise se uma ação da bolsa de valores possui alta chance de se valorizar nos próximos dias</p>	<b>ML task</b>  <p>Input, output to predict, type of problem.</p> <p>Como entrada de dados: dados históricos das ações, com valores de abertura, fechamento, máximo, mínimo e volume de negociações. E dados referente ao dólar</p> <p>Dados de saída: avaliação se a ação analisada irá se valorizar X% nos próximos Z dias. (X e Z são dados de entrada)</p>	<b>Value Propositions</b>  <p>What are we trying to do for the end-user(s) of the predictive system? What objectives are we serving?</p> <p>Com o baixo valor da taxa SELIC, diversos novos investidores surgiram tentando gerar lucros no curto prazo.</p> <p>O algoritmo desenvolvido propõe um sistema de recomendação na compra de ações, avaliando se historicamente o valor da ação de valorizou em cenários similares ao atual.</p>	<b>Data Sources</b>  <p>Which raw data sources can we use (internal and external)?</p> <p>A principal fonte de informação é o histórico do valor das ações. Informações externas.</p> <p>Também foi utilizado dados históricos do dólar como uma segunda fonte de dados.</p>	<b>Collecting Data</b>  <p>How do we get new data to learn from (inputs and outputs)?</p> <p>Os dados são obtidos através da API YahooFinance para valores de ações e do site investing.com.br para valores do dólar.</p>
<b>Making Predictions</b>  <p>When do we make predictions on new inputs? How long do we have to featurize a new input and make a prediction?</p> <p>O algoritmo foi construído para ser usado próximo ao horário de fechamento das negociações, entre 16h e 17h de cada dia.</p>	<b>Offline Evaluation</b>  <p>Methods and metrics to evaluate the system before deployment.</p> <p>A principal métrica é a Precisão.</p> <p>Como métrica secundária temos o Recall.</p>		<b>Features</b>  <p>Input representations extracted from raw data sources.</p> <p>Além das informações básicas contidas no histórico de ações (valores de abertura, fechamento, máximo, mínimo e volume de negociações), diversas outras features foram criadas a partir desses dados, como amplitude (diferença entre valores máximos e mínimos, de abertura e fechamento), variações do valor atual em relação a valores históricos, informação sobre a correlação positiva e negativa quanto ao (de)crecimento histórico do valor da ação.</p> <p>O mesmo se aplica aos dados do dólar.</p>	<b>Building Models</b>  <p>When do we create/update models with new training data? How long do we have to featurize training inputs and create a model?</p> <p>O algoritmo foi criado de forma que o treinamento é executado todos os dias, no mesmo momento da geração da predição do novo dado.</p>

## 8. Links

Link para o vídeo: <https://youtu.be/ESrLHLjl36I>

Link para o repositório: <https://github.com/rafaelcxc/recomendacaocompraacao>

## REFERÊNCIAS

AGGARWAL, C. C. **Neural Networks and Deep Learning**. Cham: Springer International Publishing, 2018.

CHEN, T.; GUESTRIN, C. XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. **Anais...** . p.785–794, 2016. New York, NY, USA: ACM. Disponível em: <<https://dl.acm.org/doi/10.1145/2939672.2939785>>. .

DORARD, L. The Machine Learning Canvas. Disponível em: <<https://www.louisdorard.com/machine-learning-canvas>>. Acesso em: 12/12/2020.

GOMES, P. C. T. Conheça o algoritmo XGBoost. Disponível em: <<https://www.datageeks.com.br/xgboost/>>. Acesso em: 13/12/2020.

HOSMER, D. W.; LEMESHOW, S.; STURDIVANT, R. X. **Applied Logistic Regression**. Wiley, 2013.

INFOMONEY. InfoMoney Magazine Luiza (MGLU3). Disponível em: <<https://www.infomoney.com.br/cotacoes/magazine-luiza-mglu3/historico/>>. Acesso em: 1/12/2020.

INVESTING. USD/BRL - Dólar Americano Real Brasileiro. Disponível em: <<https://br.investing.com/currencies/usd-brl-historical-data>>. Acesso em: 20/12/2020.

NOGARE, D. Performance de Machine Learning – Matriz de Confusão. Disponível em: <<http://diegonogare.net/2020/04/performance-de-machine-learning-matriz-de-confusao/>>. Acesso em: 6/12/2020.

ROKACH, L.; MAIMON, O. **Data Mining with Decision Trees**. New York, NY: WORLD SCIENTIFIC, 2007.

YAHOO. Yahoo Finance: API. Disponível em: <<https://finance.yahoo.com/quotes/API,Documentation/view/v1/>>. Acesso em: 27/12/2020.