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

Renato Coelho

Rastreando a tendência das ações no mercado de bolsa, com uso de técnicas de ciência de dados.

Belo Horizonte
2020

Renato Coelho

TÍTULO DO PROJETO

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.

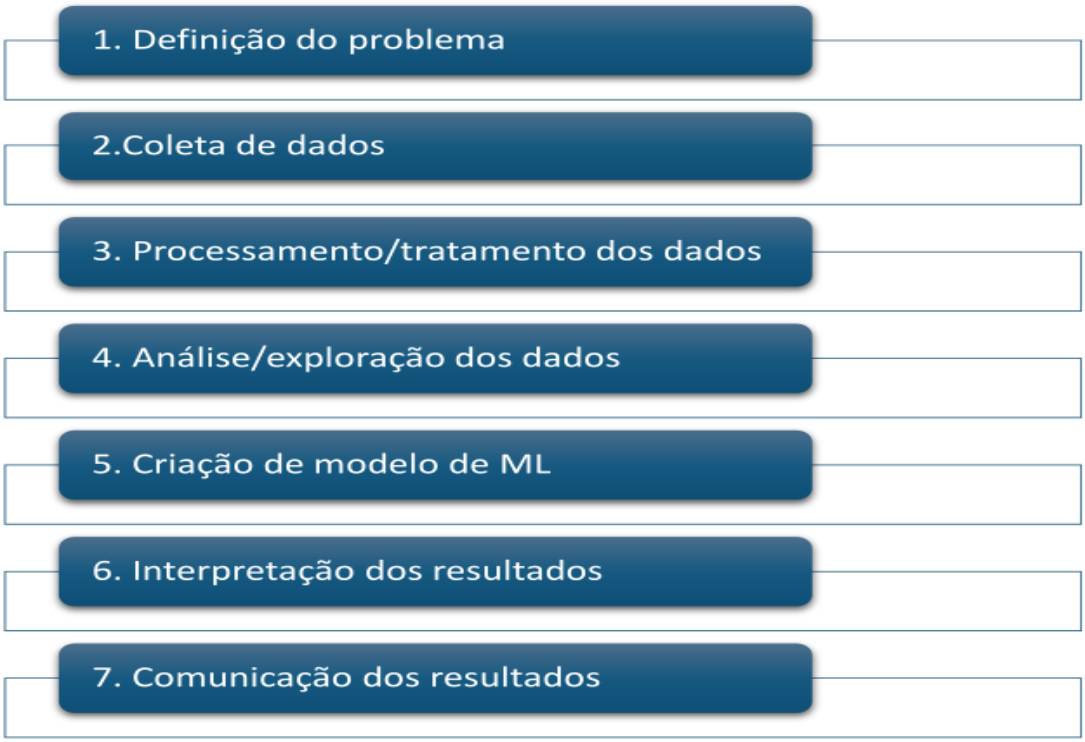
Belo Horizonte
2020

SUMÁRIO

1. Introdução

Este é um trabalho de conclusão do curso de Pós-Graduação em Ciência de Dados e BigData.

O diagrama a seguir, mostra as etapas do desenvolvimento que serão percorridos durante a construção deste trabalho.



A Ciência de Dados é um conjunto de estratégias, ferramentas e técnicas que busca reunir equipes multidisciplinares. Segundo o instituto Brasileiro de Pesquisa e Análise de Dados(IBPAD), a Ciência de Dados é uma atividade interdisciplinar que concilia principalmente duas grandes áreas: Ciência da Computação e Estatística, além de ser aplicada como apoio em diferentes áreas do conhecimento, tais como: Medicina, Biologia, Economia, Comunicação, Ciências Políticas, etc. Já o professor Luis Alfredo Vidal de Carvalho (Coppe/UFRJ – 2005), define: “Trata-se de um conjunto de técnicas reunidas da Estatística e da Inteligência Artificial com o objetivo específico de descobrir conhecimento novo que por ventura esteja escondido em grandes massas de dados(big data)”.

Big Data começa a receber atenção em 2007, com forte crescimento nos últimos anos. Big Data pode ser definido como um conjunto de técnicas capazes de se analisar grandes quantidades de dados, de diversas fontes de forma estruturada, ou não estruturada em grande volume e velocidade cada vez maior a serem processados e analisados para a geração de resultados importantes.

Machine Learning (ML) é uma importante área da inteligência artificial onde é possível criar algoritmos para ensinar uma determinada máquina a desempenhar tarefas. Um algoritmo de ML possibilita pegar um conjunto de dados de entrada e com base em determinados padrões encontrados gerar as saídas. Cada entrada desse conjunto de dados possuem suas próprias features, e ter um conjunto delas é o ponto inicial fundamental para qualquer algoritmo de ML.

O setor financeiro sempre foi no Brasil, um dos setores que mais investiu em tecnologia e inovação. E não seria neste momento que ficaria de fora. Oferecer produtos e serviços que possam alcançar uma maior rentabilidade ao seu cliente e de forma ágil e com o menor risco financeiro é um grande desafio. A descoberta da informação útil a partir de grandes bases de dados e a decisão orientada por dados, será determinante para o alcance da excelência.

Em 2002, no Brasil, tínhamos 85.249 investidores pessoas físicas na renda variável. No ano de 2008, na última grande crise mundial tínhamos no Brasil, 536.483 pessoas. Já no ano atual de 2020 temos 1.945.607 pessoas que buscam o mercado de renda variável como a possibilidade de se obter renda extra e/ou garantir uma aposentadoria.

Percebendo o tamanho deste mercado, e a tendência de se aumentar cada vez mais o número de investidores em renda variável, optei por realizar meus estudos em ciência de dados neste segmento da área financeira.

1.1. Contextualização

O mercado de capitais é um segmento do sistema financeiro responsável por intermediar negociações entre quem precisa captar recursos para financiar projetos e quem deseja investir. Isso ocorre, a partir da negociação de ativos, como por exemplo as ações e opções das empresas, títulos de dívida (as chamadas

debêntures), câmbio de moedas estrangeiras e de mercadorias como minérios e produtos agrícolas.

O mercado de bolsa, como é chamado onde se negocia as ações das empresas, é considerado um setor de risco e com muita volatilidade devido às grandes variações de valores que podem ocorrer com as ações, mesmo dentro de um único dia ou até mesmo em horas. Em primeiro lugar, não olhe para a bolsa de valores como um cassino, alerta o head de Renda Variável, Carlos Eduardo Daltozo. “Bolsa não é lugar para apostador. É necessário planejamento, estudo e dedicação”, acrescenta. (Valor investe, 02/2020)

Operar nessa área exige não apenas cautela e paciência, mas também muito estudo e conhecimento, além de informações confiáveis e ferramentas estatísticas capazes de extrair dos dados algumas medidas importantes para a tomada de decisões.

O objetivo deste trabalho, tem como ponto principal exercitar as atividades relacionadas de um cientista de dados, bem como a utilização de algumas ferramentas vistas durante o curso de pós-graduação em Ciência de Dados.

Para tal atividade principal, escolhi o tema análise de ações do mercado brasileiro, mais precisamente, um conjunto de empresas escolhidas ao acaso do setor de construção. A escolha deste setor de construção, deve-se pelo fato de ser um dos grandes setores para geração de emprego no Brasil.

Segundo o boletim Focus do Banco Central, o mercado espera que em 2020 a economia brasileira cresça 2,3%. Se confirmadas as estimativas, será o maior crescimento desde 2013, quando a economia cresceu 3,0%. Bem, e de onde, enfim, viria tamanho crescimento? O mercado tem uma aposta, a construção civil, mas o mercado está certo?

No mercado de ações, existem hoje dois principais estilos de análise, o fundamentalista e o grafista. O fundamentalista é o investidor que leva em consideração os dados macroeconômicos locais e internacionais que afetam o desempenho de setores da atividade e/ou das empresas. O objetivo é realizar projeções de resultados, estimar retornos para os próximos períodos, projetar crescimento futuro seja por crescimento orgânico ou por aquisições. Já o grafista, tentam rastrear a tendência do preço da ação, através da força entre os compradores e vendedores.

O perfil dos investidores também pode variar, vai desde os investidores de longo prazo, como os de curtíssimo prazo. Investidor de Longo Prazo (Buy and Hold) é um modo de se investir na bolsa onde o investidor adquire boas empresas, conforme sua análise, e enquanto elas mantiverem seu crescimento e números atraentes, elas permanecerão em sua carteira. Investidor de Curto Prazo, tem um perfil mais especulador e menos investidor, com operações chamadas de day trade (investidor que realiza operações em mercados financeiros, com o objetivo de obtenção de lucro, dentro da oscilação do dia) ou até mesmo scalper trade (investidor de day trading, que tem como intuito lucrar com os movimentos curtíssimos, rápidos e sequenciais que acontecem durante um dia de negociação).

Sabemos que o investidor que tiver a informação mais rápida e souber analisar esta, pode levar vantagens na montagem de sua estratégia operacional.

1.2. O problema proposto

Este trabalho, utilizando a técnica dos 5-Ws, elaboro meu plano de ação, procurando passar uma visão do entendimento, estratégia e dados necessários para apoiar o desenvolvimento do estudo.

Why?

Quem nunca sonhou em ter os números da mega-sena? Assim como diversos estudos foram realizados com o objetivo de se tentar prever os números da mega-sena, prever o preço exato da ação talvez seja algo tão difícil quanto, pois os preços das ações variam a todo momento e dependem de diversas variáveis (detalhadas mais a frente neste trabalho). Algumas destas variáveis são difíceis de se prever, e ainda, o reflexo do seu comportamento.

Existem diferentes definições para preço de uma ação, ele pode ser definido como o valor que alguém está disposto a pagar ou o que a última pessoa pagou por algo. Pode ser definido também como sendo a interseção das curvas de oferta e demanda. (ELDER, 2004).

Na visão do autor deste trabalho, o que determina o preço da ação é a força dos compradores versus a força dos vendedores, no sentido figurado da palavra, a

briga do touro contra o urso. Esta briga a qual me refiro é, na verdade, apostas dos “traders” (pessoa ou entidade, em finanças, que compram e vendem instrumentos financeiros como ações, títulos e derivativos na capacidade de agente, hedger, arbitragem ou o especulador) de prever o preço da ação.

A tendência, direção que caminhará a ação, pode ser determinada com base num conjunto de informações / variáveis, sendo que os diversos investidores através de análise destas informações, realizam suas operações na bolsa de valores.

A previsão do comportamento de uma ação ou do mercado como um todo, é tema de interesse de todos os investidores, pois as análises acuradas se traduzem em retorno financeiro.

E o que o investidor deve analisar antes de montar sua estratégia operacional:

- Variáveis macroeconômicas – Desenvolvimento do crescimento econômico, a taxa de empregos, a inflação, crises internacionais sejam estas políticas, econômicas e até de catástrofes / epidemias.

- Variáveis setoriais – É essencial analisar as expectativas do setor no qual a empresa está inserida, pois cada área se comporta de uma maneira diferente da outra. É importantíssimo ter uma noção de quais são as perspectivas de crescimento para o mercado de uma determinada companhia e o que pode afetar essa evolução.

- Variáveis de mercado – Como não poderia deixar de ser, tudo o que afeta especificamente o mercado de investimentos influenciará, consequentemente, o valor direto das ações. Como exemplo, podemos citar a alta de impostos e as alterações nas normas para aplicações como fatores que podem tanto motivar como desanimar o mercado de investimentos.

- Variável comportamento – É sempre importante estar atento aos elementos que influenciam as finanças de uma organização, tais como as mudanças no preço dos produtos, investimentos ou o endividamento, e também às mudanças na estrutura acionária.

Pela complexidade aqui exposta, vejo que a área de ciência de dados, será uma grande aliada na tomada de decisões das empresas financeiras e também dos investidores comuns que podem ser beneficiados por relatórios e direcionamentos realizadas por empresas especializadas.

Who?

Embora os dados oficiais é da B3, empresa responsável pela infraestrutura do mercado de ações no Brasil, por uma simples praticidade, utilizei a base disponível pela empresa Yahoo através da url https://finance.yahoo.com/quote/.

O site Yahoo já disponibiliza as informações agrupadas pelo período desejado, e somente informações de cotação da empresa desejada, além da possibilidade de gerar o arquivo no formato csv, ficando pronto para utilização pela biblioteca Pandas na linguagem de programação Python.

What?

Ao contrário da mega-sena que para ganhar você precisa ter os 06 números exatos do sorteio, no mercado de ações, utilizando técnicas de estatísticas, talvez seja possível prever a tendência da ação ou, pelo menos, o comportamento dos investidores, gerando assim uma probabilidade de ganhos ao investidor.

Pela probabilidade e estatística é possível utilizar um modelo matemático que pode antecipar os valores dos preços das ações de uma determinada empresa participante no B3(antiga BMF&Bovespa) a serem alcançados, em determinado período. Essa predição auxilia a pessoa ou as empresas a tomarem decisões de compras ou vendas de ações baseando-se em probabilidades. O foco principal é agregar valor aos investidores.

O objetivo principal deste trabalho consiste em criar modelos, através de técnicas de aprendizado de máquina, utilizando as séries temporais dos preços das ações.

Where?

Estas análises serão realizadas utilizando um notebook processador intel i5, utilizando o sistema operacional windows 8.1 64 bits. Utilizei a ferramenta Anaconda,

por ser uma distribuição gratuita e de código aberto da linguagem de programação Python. O Anaconda é uma iniciativa que tem como objetivo agregar todas as ferramentas para análise de dados, como o Jupyter Notebook, o qual utilizei para escrever meus códigos em Python, além de famosas bibliotecas, como Pandas, Matplotlib e Scikit-learn.

Neste trabalho, também foi utilizado o ambiente na nuvem do Google. O Google Colaboratory é um ambiente de notebooks Jupyter que não requer configuração e é executado na nuvem. Este ambiente foi necessário para a utilização das bibliotecas Keras e TensorFlow, na implementação de LSTM. Vou me permitir e detalhar apenas no momento oportuno, mais a frente neste trabalho. Por hora, fica aqui apenas o registro da utilização destas bibliotecas e ambiente.

When?

Os dados de histórico de cotação das empresas envolvidas nesta análise, compreendem o período de 02 de Janeiro de 2015 à 10 de Janeiro de 2020.

2. Coleta de Dados

O dataset utilizado para testes deste trabalho, foram obtidos através do site https://finance.yahoo.com. Entrando nesta url você deve clicar no link de cotação e digitar o código da empresa desejada.

A url final, como exemplo a empresa Gafisa, temos: https://finance.yahoo.com/quote/GFSA3.SA/history?p=GFSA3.SA .

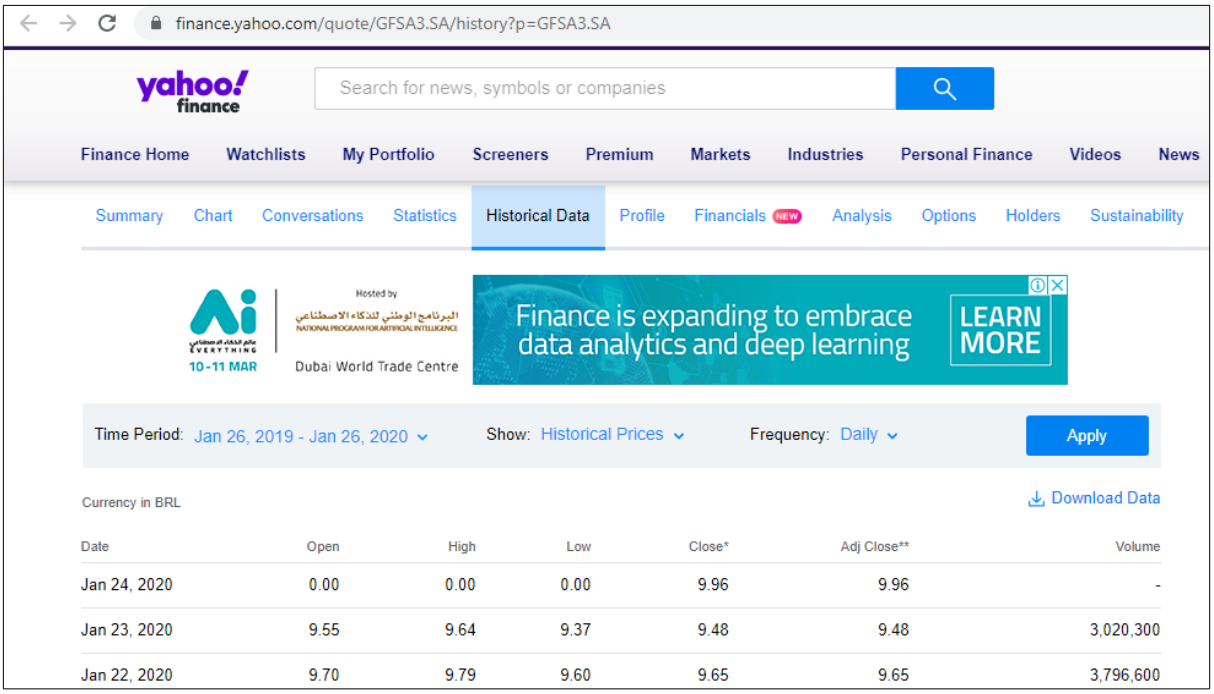


Figura 1 – Site Yahoo – Módulo Financeiro

O site do Yahoo, já disponibiliza as informações agrupadas pelo período desejado, você pode ainda escolher a frequência desejada, diária, semanal ou anual. Cada dataset gerado, conterá somente as informações de cotação da empresa desejada. Uma outra vantagem, é de gerar o arquivo no formato csv, ficando pronto para utilização pela biblioteca Pandas na linguagem de programação Python ou linguagem R.

Entendendo os dataset's. Os arquivos csv's baixados respeitam a mesma estrutura de dados, o qual descrevo abaixo:

Nome do Arquivo	Papel	Empresa	Período
GFSA35A.csv	GFSA3	Gafisa	02/01/2015 à 10/01/2020
HBOR35A.csv	HBOR3	Helbor	02/01/2015 à 10/01/2020
MRVE35A.csv	MRVE3	MRV	02/01/2015 à 10/01/2020
RSID35A.csv	RSID3	Rossi	02/01/2015 à 10/01/2020
TRIS35A	TRIS3	Trisul	02/01/2015 à 10/01/2010

Nome da coluna	Descrição	Tipo
Date	Campo data no formato YYYY-MM-DD. É a data correspondente ao dia de negociação.	Date
Open	Campo com a informação do preço de abertura do papel em determinado dia de negociação.	Numérico
High	Campo com a informação do preço máximo negociado na data.	Numérico
Low	Campo com a informação do preço minímo negociado do papel na data.	Numérico
Close	Campo com a informação do preço de fechamento negociado do papel na data.	Numérico
Adj Close	Campo com a informação de ajuste de fechamento negociado do papel na data.	Numérico
Volume	Campo com a informação do volume total de títulos negociados do papel realizado na data.	Numérico

3. Processamento/Tratamento de Dados

Para tratamento dos dataset's, utilizei as ferramentas Jupyter Notebook, linguagem de programação Python e biblioteca Pandas.

O Jupyter Notebook é um interfarce gráfica que permite a edição de notebooks em um navegador web, tais como Google Chrome ou Firefox.

Um documento do tipo notebook é um documento virtual que permite a execução de códigos de linguagem de programação. O notebook permite uma maneira interativa de programar, oferecendo ao usuário o output imediato do código, não havendo, assim, a necessidade de compilar ou executar todo o documento.

O primeiro passo é ler os arquivos .cvs, com o comando "read" do Pandas exemplo:

```
In [1]: import pandas as pd
        dataset = pd.read_csv('C:/Users/Lapin/Documents/Pytest/dados/GFSA35A.csv')
        dataset.head()
```

Out[1]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2015-01-02	15.8192	15.8192	14.3126	14.8507	14.812973	407867.0
1	2015-01-05	14.9583	14.9583	14.3126	14.5278	14.490892	928377.0
2	2015-01-06	14.5278	14.9583	14.3126	14.7431	14.705646	386949.0
3	2015-01-07	14.9583	15.3888	14.7431	15.1735	15.134953	743623.0
4	2015-01-08	15.3888	15.6040	14.8507	15.4964	15.457032	326855.0

Utilizando o comando dtypes, para identificar o tipo dos dados.

```
In [3]: dataset.dtypes

Out[3]: Date          object
        Open         float64
        High         float64
        Low          float64
        Close        float64
        Adj Close    float64
        Volume       float64
        dtype: object
```

Note no quadro acima, o campo Date o pandas classificou com o tipo “obtect”.

Desta forma, precisamos transformar este campo no tipo data. Para isso, executamos o comando to_datetime para converter em campo data.

```
In [4]: dataset['Date'] = pd.to_datetime(dataset['Date'])
        dataset.head()
```

Out[4]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2015-01-02	15.8192	15.8192	14.3126	14.8507	14.812973	407867.0
1	2015-01-05	14.9583	14.9583	14.3126	14.5278	14.490892	928377.0
2	2015-01-06	14.5278	14.9583	14.3126	14.7431	14.705646	386949.0
3	2015-01-07	14.9583	15.3888	14.7431	15.1735	15.134953	743623.0
4	2015-01-08	15.3888	15.6040	14.8507	15.4964	15.457032	326855.0

Olhando os dados do dataset, aparentemente não sofreu nenhuma alteração, mas ao executarmos novamente o comando dtypes, é possível visualizar a alteração.

```
In [5]: dataset.dtypes
Out[5]: Date          datetime64[ns]
        Open                 float64
        High                 float64
        Low                  float64
        Close                float64
        Adj Close            float64
        Volume               float64
        dtype: object
```

O comando shape apresenta a quantidade de linhas e colunas lidas.

```
In [7]: dataset.shape
Out[7]: (1250, 7)
```

Note que automaticamente ele desconsiderou o cabeçalho do arquivo. Estou abrindo o arquivo .csv num editor de planilhas, para mostrar detalhes do arquivo.

	A	B	C	D	E	F	G
1	Date	Open	High	Low	Close	Adj Close	Volume
2	2015-01-02	15.819200	15.819200	14.312600	14.850700	14.812973	407867
3	2015-01-05	14.958300	14.958300	14.312600	14.527800	14.490892	928377
1246	2020-01-03	9.530000	10.370000	9.350000	10.150000	10.150000	14827100
1247	2020-01-06	10.150000	10.550000	9.850000	9.900000	9.900000	8695100
1248	2020-01-07	10.000000	10.160000	9.420000	9.550000	9.550000	7715100
1249	2020-01-08	9.550000	9.570000	8.930000	9.090000	9.090000	9987100
1250	2020-01-09	9.180000	9.510000	9.030000	9.070000	9.070000	8374900
1251	2020-01-10	9.180000	9.320000	9.030000	9.190000	9.190000	6068300
1252							

Agora, precisamos identificar se existe algum valor nulo em nosso dataset. O pandas, ao carregar o dataset já identifica se existe valor nulo, e automaticamente insere o valor NAN, que significa já traduzindo Não é Número. No entanto, para alguns cálculos que realizaremos posteriormente neste trabalho, se não tratarmos essa informação, dará erro.

O comando é o isnull, e executamos conforme imagem:

```
In [8]: #Verificando de existe alguma informação nula.
        col_mask=dataset.isnull().any(axis=0)
        row_mask=dataset.isnull().any(axis=1)
        dataset.loc[row_mask,col_mask]
```

Out[8]:

	Open	High	Low	Close	Adj Close	Volume
778	NaN	NaN	NaN	NaN	NaN	NaN
1037	NaN	NaN	NaN	NaN	NaN	NaN

Veja, que foi identificado duas linhas no dataset 778 e 1037. Como é uma relação muito pequena em relação a quantidade de registros, estes registros não invalidam nossa massa de dados.

P = 2 defeitos / 1250 de total de registros = P = 0,0016 = P = 0,16%

Durante os primeiros testes realizados, eu apenas deletei estes dois registros do dataset, sem nenhuma perda ao resultado do teste. No entanto, para o teste final

deste trabalho, resolvi preencher os valores com a média do dia anterior com o dia posterior, para as duas datas com valores NAN:

14/02/2018

Dia Anterior	14,7488	14,7488	13,7836	13,9821	13,9821	1451106
Dia Posterior	13,9821	14,5143	13,7566	14,5143	14,5143	1408648
Data NAN	14,36545	14,63155	13,7701	14,2482	14,2482	1429877

06/03/2019

Dia Anterior	8,930500	9,065810	8,822250	8,822250	8,822250	1820035
Dia Posterior	8,840290	8,903430	8,714000	8,750080	8,750080	908355
Data NAN	8,885395	8,98462	8,768125	8,786165	8,786165	1364195

4. Análise e Exploração dos Dados

Dando início a análise dos dados, demonstrarei em forma de gráfico de linha, a evolução dos preços durante o período estudado. Neste gráfico, estou apresentando a evolução do preço dentro de todo o período pesquisado que compõe o dataset.

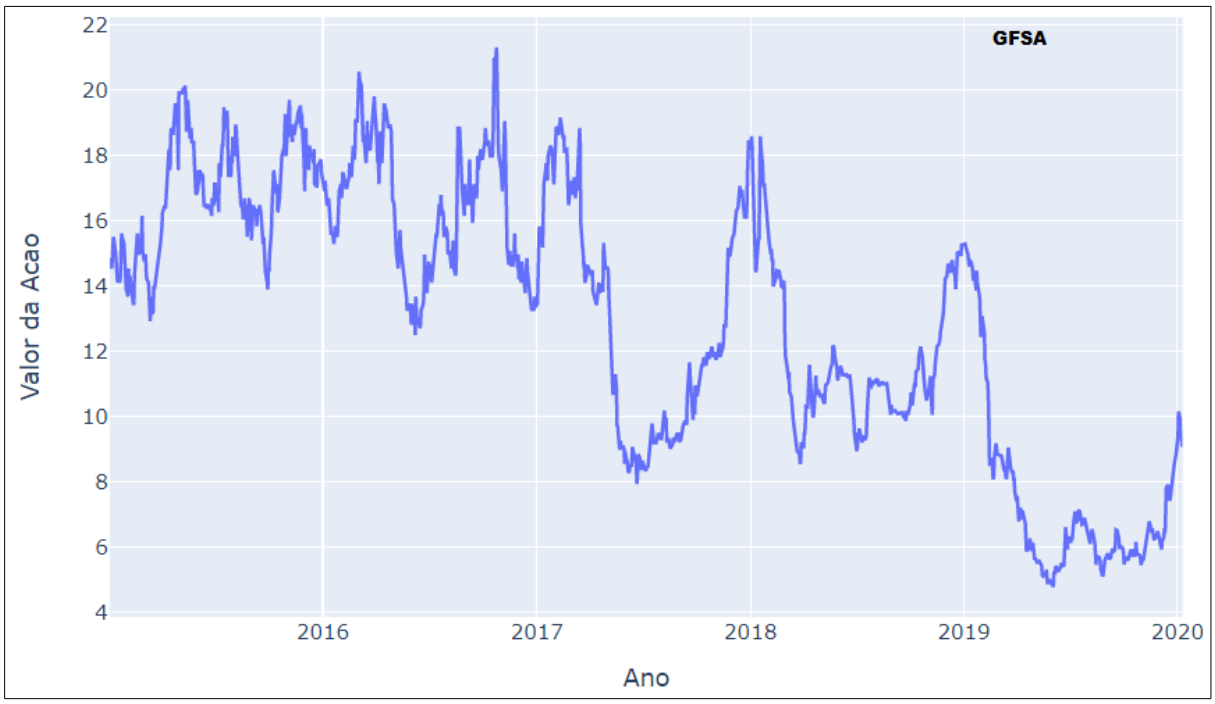


Figura 2 - Gráfico de linha – Evolução dos preços de fechamento da Gafisa

Para a composição do gráfico demonstrado acima, você deve importar as bibliotecas plotly. Esta biblioteca permite construir gráficos interativos.

Em tempo, vale deixar registrado que todas as bibliotecas que não fazem parte da instalação básica do Python, devem ser instaladas através do comando: **pip install plotly**, isto para os casos onde o sistema operacional é Windows.

```
In [9]: import plotly.offline as py
        import plotly.graph_objs as go
        py.init_notebook_mode(connected=True)
```

Este gráfico é composto por duas colunas do dataset, “date” e “Close”, correspondente ao campo data na negociação e o preço de fechamento desta data, adicionando nas variáveis x1 e y1 respectivamente.

Na figura a seguir é demonstrado todo o comando utilizado para plotar o gráfico.

```
In [10]: x1=dataset.Date
         y1=dataset.Close
         data = [go.Scatter(x=x1, y=y1)]
         layout = go.Layout(
            xaxis=dict(
                range=['02-01-2015','10-01-2020'],
                title='Ano'
            ),
            yaxis=dict(
                range=[min(x1), max(y1)],
                title='Valor da Acao'
            ))
         fig = go.Figure(data = data, layout = layout)
         py.iplot(fig)
```

O gráfico de linhas é o mais simples, mas é o que proporciona menos informações ao analista, já que não oferece nenhum tipo de informação sobre qual foi exatamente o comportamento do papel durante o dia de negociação. Esse tipo de gráfico apresenta simplesmente uma linha ligando os valores de fechamento dos períodos.

Um dos gráficos mais utilizados na análise de ações, é o gráfico candlestick, em português simplificamos para gráfico de velas. Ele consiste basicamente nas mesmas informações contidas no gráfico de barras, porém com algumas particularidades que facilitam a visualização da movimentação ocorrida dentro do período. São estas particularidades que fazem uma grande diferença na análise dos candles. Além das informações apresentadas nos gráficos de barras, a diferença crucial no gráfico de candles é o fato de cada barra apresentar o que chamamos de “pavios” (ou sombras), que são trechos mais finos da barra, visualizados no topo e/ou abaixo do corpo real da figura. Esse destaque pode ser verificado de duas formas: vazado (cor verde no meu exemplo), quando temos o preço de fechamento maior que o preço de abertura, ou preenchido (cor vermelha no meu exemplo), com o preço de fechamento abaixo do preço de abertura. (Ágora, 2014).

Para uma melhor visualização do gráfico, eu escolhi um período dos 21 dias mais recentes do arquivo.

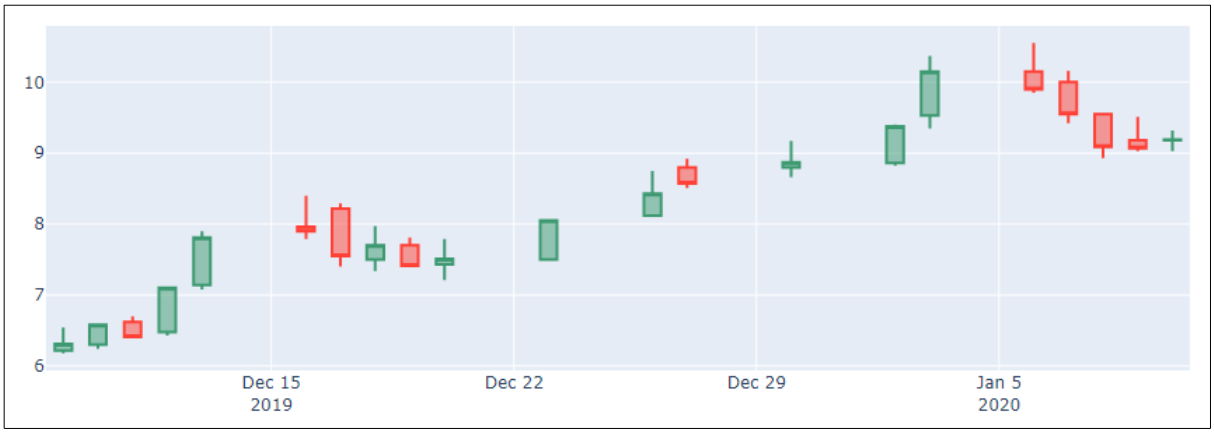


Figura 3 - Gráfico de candle – Evolução dos preços de fechamento da Gafisa

Estes espaços entre um candle e outro, representa o período onde não teve negociação, por ser um dia não útil. Embora eu não tenha no dataset estas datas, quando a biblioteca utilizada plotou o gráfico, automaticamente ela dividiu o gráfico em semanais, sendo 5 divisões ou seja, 5 semanas contempladas neste período de 21 dias. Cada candle representa um dia de movimento, e reforçando, o preenchido em verde o preço de fechamento foi maior que o preço de abertura. Enquanto os pintados em vermelho, são os dias em que o preço de fechamento foi menor que o preço de abertura.

Como o gráfico é interativo, ao passarmos o mouse em cima de um dos candles, é apresentado um quadro com o detalhe do preço no dia, mostrando o preço máximo do dia e o preço mínimo no dia. Todas estas informações, foram tiradas no dataset e pode ser visto no gráfico de candle devido ao seu desenho.

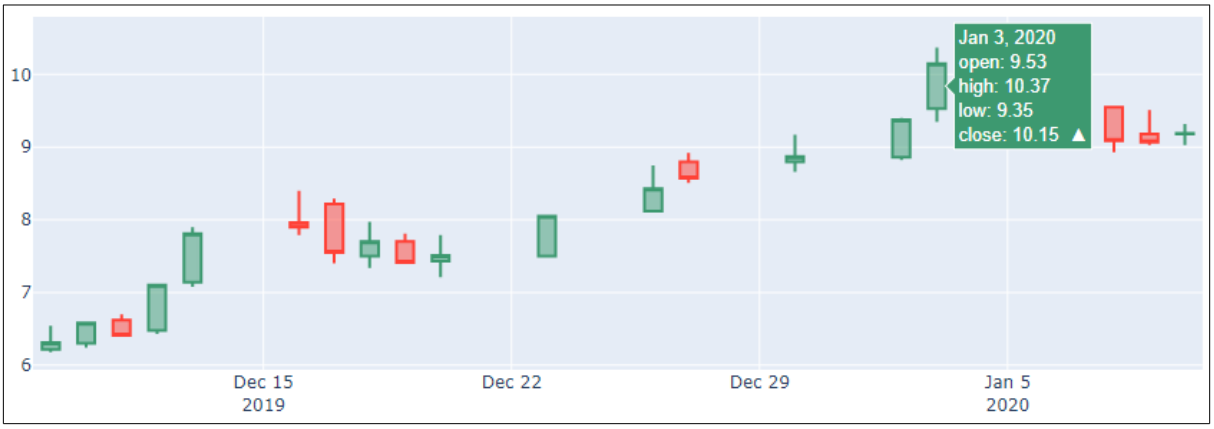


Figura 4 - Gráfico de candle com interatividade – Evolução dos preços de fechamento da Gafisa

E como eu faço tudo isso no código? É a mesma biblioteca importada anteriormente no gráfico de linha, utilizando o código Python abaixo.

```
In [15]: # Vizualizando através do gráfico candlestick os 21 dias mais recentes da massa de dados

dataset2 = dataset.tail(21)
dados = go.Candlestick(x=dataset2.Date,
                       open=dataset2.Open,
                       high=dataset2.High,
                       low=dataset2.Low,
                       close=dataset2.Close,
                       )
data=[dados]
py.offline.iplot(data,filename='grafico_candlestick')
```

Como eu desejava mostrar apenas os 21 dias mais recentes neste gráfico, eu passei o valor de 21 na função “tail” do dataset. Geralmente, utilizam a função “head”, no entanto, apresentaria os primeiros dias, e não os últimos como desejado nesta análise.

Voltando a análise utilizando a biblioteca pandas, uma das funções interessantes é a função “describe”. Esta função, retorna informações importantes sobre nossa massa de dados, que podem ser utilizados nas análises estatísticas. As informações são: count (quantidade de registros na massa de dados analisado), mean (média dos preços) , std ou standard deviation (desvio padrão dos preços), min (os valores mínimos daquela coluna) , max (os valores máximos daquela coluna), e os quartis 25%, 50% e 75% .

```
In [5]: dataset.describe()
```

Out[5]:

	Open	High	Low	Close	Adj Close	Volume
count	1250.000000	1250.000000	1250.000000	1250.000000	1250.000000	1.250000e+03
mean	13.002271	13.269651	12.684416	12.980676	12.969530	1.096188e+06
std	4.238084	4.315810	4.122763	4.239343	4.228237	1.401864e+06
min	4.780000	4.930000	4.650000	4.770000	4.770000	0.000000e+00
25%	9.535000	9.631855	9.325833	9.464975	9.464975	3.273352e+05
50%	13.959500	14.205000	13.630750	13.882200	13.882200	6.614425e+05
75%	16.572500	16.904600	16.249701	16.545600	16.514022	1.366899e+06
max	21.199900	21.845600	20.769400	21.307501	21.307501	1.482710e+07

Algumas informações que podem ser importantes num processo de análise. O quanto o preço variou entre o preço de abertura e o preço de fechamento? Ou, o quanto o preço variou entre o preço mínimo do dia e o preço máximo do dia.

Embora, não tenha essa informação no dataset, é possível calcular.

Segue os comandos e resultados.

```
In [18]: # Calcula a variação entre o preco de abertura e fechamento
         dataset['Variation'] = dataset['Close'].sub(dataset['Open'])

In [19]: # Calcula a variação entre o preço mínimo e preço máximo do dia
         dataset['Variation MM'] = dataset['High'].sub(dataset['Low'])

In [20]: dataset.head()
```

Out[20]:

	Date	Open	High	Low	Close	Adj Close	Volume	Variation	Variation MM
0	2015-01-02	15.8192	15.8192	14.3126	14.8507	14.812973	407867	-0.9685	1.5066
1	2015-01-05	14.9583	14.9583	14.3126	14.5278	14.490892	928377	-0.4305	0.6457
2	2015-01-06	14.5278	14.9583	14.3126	14.7431	14.705646	386949	0.2153	0.6457
3	2015-01-07	14.9583	15.3888	14.7431	15.1735	15.134953	743623	0.2152	0.6457
4	2015-01-08	15.3888	15.6040	14.8507	15.4964	15.457032	326855	0.1076	0.7533

Analisando todos os dias do dataset, pode ocorrer de encontrar o preço de abertura e preço de fechamento muito próximo um do outro, ou até mesmo, no mesmo preço, não sendo tão animador para os investidores de swing trader e position (investidores que operam num intervalo de tempo longo). No entanto, durante o dia, pode ter ocorrido uma grande variação nos preços da ação, gerando grandes oportunidades para os investidores de day trade e scalper trade.

Nos próximos gráficos, observaremos esta variação dos preços citada, ocorrido durante todo o pregão. Primeiro será apresentado a variação entre o preço de fechamento e preço de abertura. No segundo será apresentado a variação entre o preço mais alto e o preço mais baixo do dia.

Para gerar estes gráficos, basta importar as bibliotecas do matplotlib, no eixo x passar o campo “Date” do dataset e no eixo y informar a coluna “Variation” já calculada anteriormente. No segundo gráfico, no eixo y informar a coluna “Variation MM”.

A variação dos preços é algo interessante de plotar, veja como ficaram os gráficos e perceba como o preço dessa ação variou.

```
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import datetime as dt

x = dataset['Date']
y = dataset['Variation']

plt.plot_date(x,y, color='r',fmt="r-")
plt.xticks(rotation=30)
plt.suptitle('Gafisa variação dos preços entre Abertura e Fechamento')
plt.show()
```

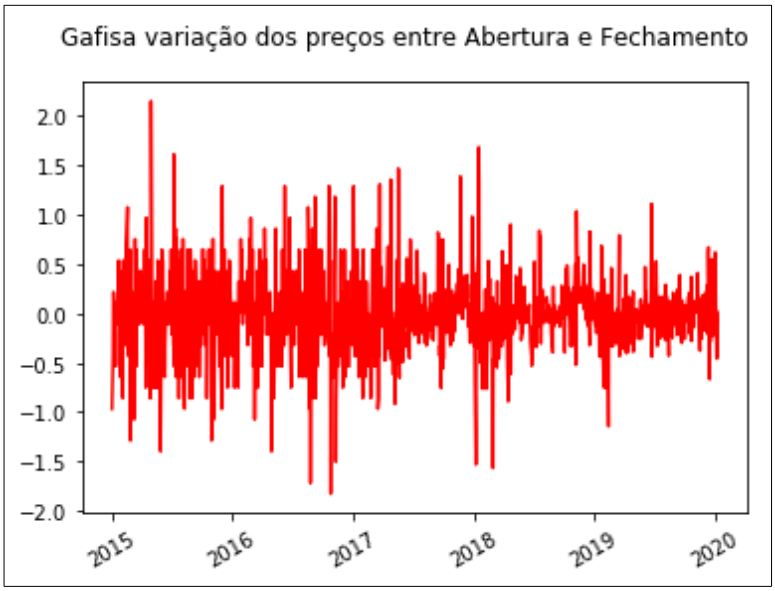


Figura 5 – Gráfico de variança dos preços da Gafisa

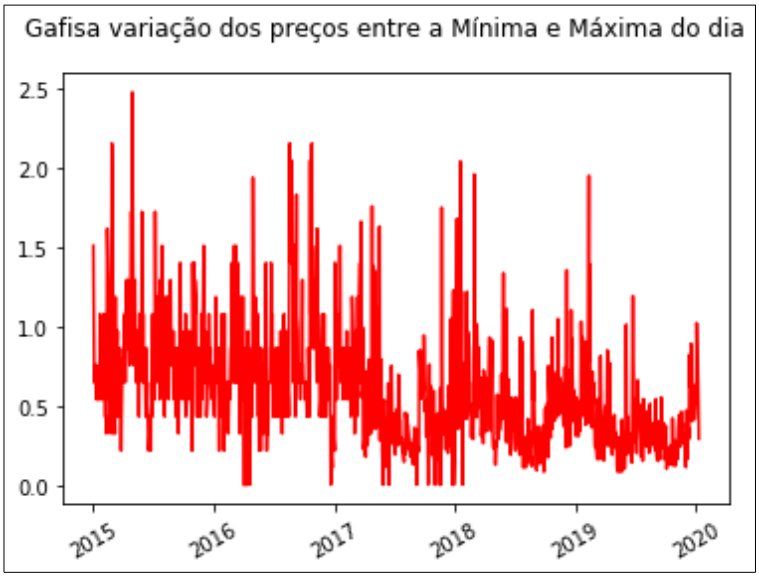


Figura 6 – Gráfico de variança de preços entre a mínima e máxima da Gafisa

Nas análises, também identificamos o aumento do volume ocorrido principalmente no ano de 2019 e início de 2020, tendo grandes picos em algumas datas. Isso, pode ser comprovado no gráfico de variação de volume.

```
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import datetime as dt


x = dataset['Date']
y = dataset['Volume']

plt.plot_date(x,y, color='b',fmt="b-")
plt.xticks(rotation=30)
plt.suptitle('Gafisa volume de negócios')
plt.show()
```

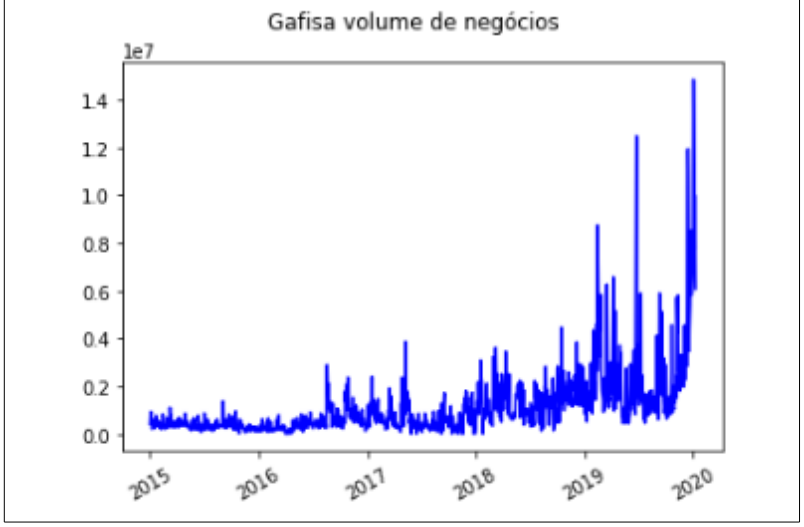


Figura 7 – Gráfico de volume de negócios da Gafisa

Com base nesta informação de aumento de volume, podemos supor duas hipóteses. Uma melhora na economia com a retomada do crédito para a população, e/ou a procura por ativos de renda variável para investimentos, uma vez que ocorreu uma diminuição da taxa selic, taxa básica de referência para remuneração de investimentos na renda fixa.

Em busca de dar mais subsídios a nossa análise, adiciono nesta pesquisa, ações de outras empresas do mesmo setor de atuação. As empresas também foram escolhidas ao acaso, e são elas: Helbor, MRV, Rossi e Trisul.

Evitando que o trabalho fique cansativo e repetitivo, todos os pontos de validação do dataset realizados nos dados da empresa Gafisa e demonstrados anteriormente neste documento, também foram realizados nestas novas empresas,

destacando os erros encontrados no dataset nas datas de 14/02/2018 e 06/03/2019, aplicando a mesma regra das médias.

O objetivo da inclusão destas novas empresas é comparar o comportamento dos preços se são semelhantes ou não para as empresas do mesmo setor.

O primeiro comando é o “describe”, apresentando as informações de count (quantidade de registros) demonstrando, a nível de comparação, a mesma quantidade de registros em todos os dataset’s. Outras informações importantes destas ações são: mean (média dos preços) , std ou standard deviation (desvio padrão dos preços), min (os valores mínimos daquela coluna) , max (os valores máximos daquela coluna), e os quartis 25%, 50% e 75%.

Dados das ações da empresa Helbor:

```
datasetHelbor.describe()
```

	Open	High	Low	Close	Adj Close	Volume	Variation
count	1250.000000	1250.000000	1250.000000	1250.000000	1250.000000	1.250000e+03	1250.000000
mean	2.031877	2.078336	1.979360	2.023773	1.970246	1.570134e+06	-0.008105
std	0.781334	0.797326	0.759528	0.779878	0.710242	1.775198e+06	0.070409
min	0.910000	0.920000	0.890000	0.910000	0.910000	0.000000e+00	-0.298240
25%	1.440000	1.480000	1.407975	1.436140	1.419670	4.901595e+05	-0.040000
50%	1.929220	1.966290	1.889780	1.919900	1.866876	9.838940e+05	-0.010000
75%	2.292410	2.368670	2.222862	2.283237	2.275376	1.942274e+06	0.020000
max	4.780000	4.880000	4.660000	4.820000	4.820000	1.799530e+07	0.320000

Dados das ações da empresa MRV:

```
datasetMRV.describe()
```

	Open	High	Low	Close	Adj Close	Volume	Variation
count	1250.000000	1250.000000	1250.000000	1250.000000	1250.000000	1.250000e+03	1250.000000
mean	11.864932	12.056703	11.668954	11.863321	10.548465	3.650483e+06	-0.001611
std	3.640252	3.687737	3.571581	3.635299	3.817606	2.158793e+06	0.266178
min	5.509160	5.590980	5.454610	5.500070	4.509130	0.000000e+00	-2.540000
25%	9.684202	9.829663	9.552388	9.693293	8.141259	2.284206e+06	-0.136367
50%	11.813750	11.972900	11.613750	11.786500	10.522151	3.158778e+06	0.000000
75%	13.752425	13.963800	13.544200	13.751025	12.249538	4.472712e+06	0.136400
max	22.379999	22.780001	21.860001	22.639999	22.350000	1.864060e+07	1.390002

Dados das ações da empresa Rossi:

```
datasetRossi.describe()
```

	Open	High	Low	Close	Adj Close	Volume	Variation
count	1250.000000	1250.000000	1250.000000	1250.000000	1250.000000	1.250000e+03	1250.000000
mean	5.840476	6.008028	5.657188	5.804308	5.804308	2.815340e+05	-0.036168
std	2.524478	2.613057	2.426137	2.504812	2.504812	4.278798e+05	0.302970
min	1.960000	1.980000	1.660000	1.960000	1.960000	0.000000e+00	-2.400000
25%	4.110000	4.230000	4.000000	4.092500	4.092500	7.432500e+04	-0.150000
50%	5.070000	5.200000	4.925000	5.030000	5.030000	1.686000e+05	-0.040000
75%	7.020000	7.120000	6.900000	7.000000	7.000000	3.417200e+05	0.050000
max	16.750000	17.950001	15.300000	16.750000	16.750000	7.158500e+06	2.650000

Dados das ações da empresa Trisul:

```
datasetTrisul.describe()
```

	Open	High	Low	Close	Adj Close	Volume	Variation
count	1250.000000	1250.000000	1250.000000	1250.000000	1250.000000	1.250000e+03	1250.000000
mean	3.069032	3.118138	3.014972	3.075080	2.878458	2.955535e+05	0.006048
std	3.075239	3.133648	3.011181	3.085749	3.172118	6.296572e+05	0.101886
min	1.125000	1.135000	1.050000	1.090000	0.818807	0.000000e+00	-0.730000
25%	1.395000	1.405000	1.375000	1.395000	1.099440	2.200000e+03	-0.020000
50%	1.500000	1.525000	1.490000	1.520000	1.288829	5.495000e+04	0.000000
75%	2.920000	2.950000	2.867500	2.907500	2.692446	3.093000e+05	0.030000
max	16.629999	16.850000	16.389999	16.570000	16.570000	8.160400e+06	0.989999

Agora, vamos analisar a evolução dos preços destas 5 ações durante o mesmo período de tempo analisado. Será que estas empresas sendo do mesmo setor tem a mesma evolução nos preços?

```
#Grafico mostra a evolução dos preços da ação.
plt.figure(figsize = (15,10))
plt.plot(datasetGafisa['Date'], datasetGafisa['Close'], label='Gafisa')
plt.plot(datasetHelbor['Date'], datasetHelbor['Close'], label='Helbor')
plt.plot(datasetMRV['Date'], datasetMRV['Close'], label='MRV')
plt.plot(datasetRossi['Date'], datasetRossi['Close'], label='Rossi')
plt.plot(datasetTrisul['Date'], datasetTrisul['Close'], label='Trisul')
plt.legend(loc='best')
plt.show()
```

Este código acima, plotará o gráfico de linhas, com a evolução dos preços de fechamento das 5 empresas a serem analisadas do setor de construção.

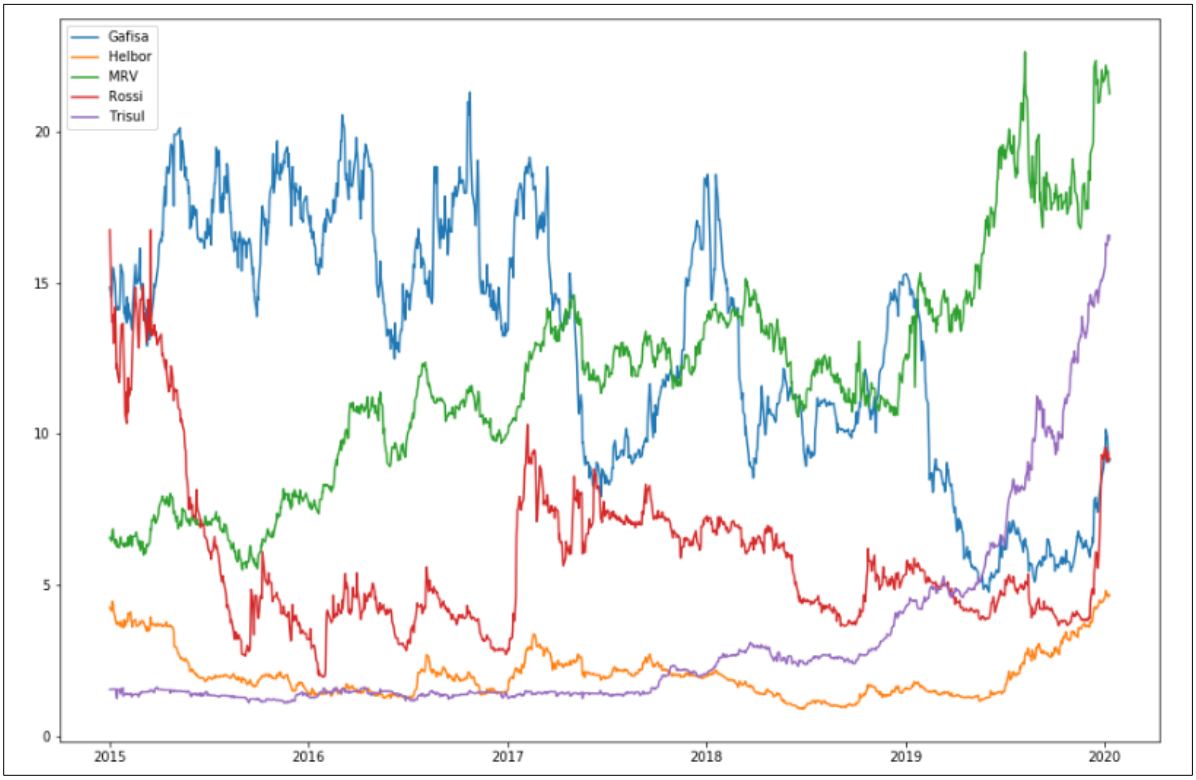


Figura 8 – Gráfico de linhas com a evolução dos preços das empresas do setor de construção

Com este gráfico, observamos, que mesmo sendo empresas do mesmo setor, os preços na sua linha do tempo, não possuem a mesma evolução, ou seja, não possuem uma relação direta.

Analisando o volume das 5 empresas em conjunto, chegamos a conclusão que o volume de negociação da empresa MRV foi maior em praticamente todo o período de análise.

Para construir este gráfico, utilizei os seguintes comandos.

```
#Grafico mostra o volume da ação em determinado período
plt.figure(figsize = (20,15))
plt.plot(datasetGafisa['Date'], datasetGafisa['Volume'], label='Gafisa')
plt.plot(datasetHelbor['Date'], datasetHelbor['Volume'], label='Helbor')
plt.plot(datasetMRV['Date'], datasetMRV['Volume'], label='MRV')
plt.plot(datasetRossi['Date'], datasetRossi['Volume'], label='Rossi')
plt.plot(datasetTrisul['Date'], datasetTrisul['Volume'], label='Trisul')
plt.legend(loc='best')
plt.show()
```

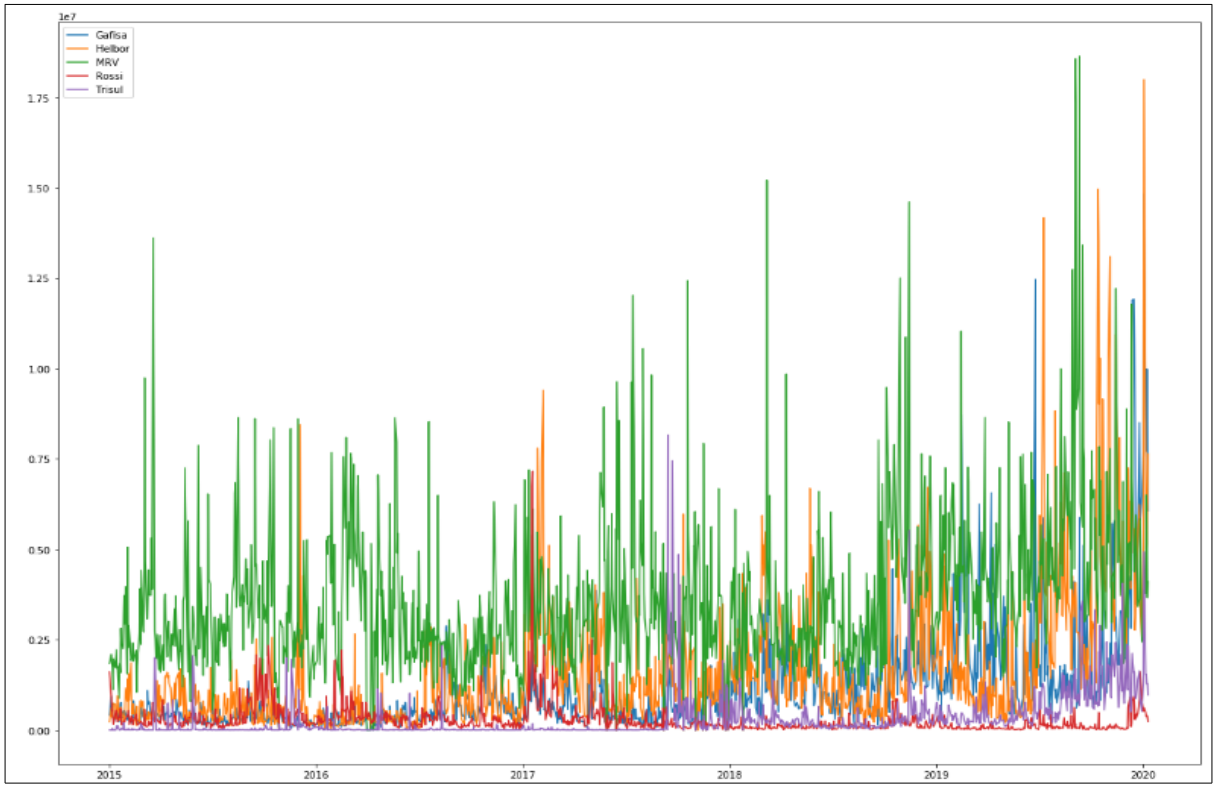


Figura 9 - Gráfico de volume das 5 empresas do setor de construção

Podemos analisar o volume de negócios em gráficos distintos. Utilizei uma outra forma de código, no entanto, utilizando a mesma biblioteca.

```
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import datetime as dt

x = datasetGafisa['Date']
y = datasetGafisa['Volume']

plt.plot_date(x,y, color='b',fmt="b-")
plt.xticks(rotation=30)
plt.suptitle('Gafisa volume de negócios')
plt.show()
```

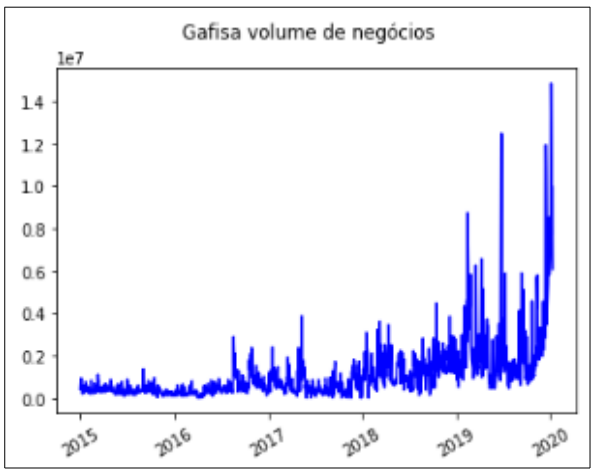


Figura 10 - Gráfico de volume de negócios da Gafisa

Utilizando o mesmo código, alterando o nome do dataset, gerei os gráficos abaixo.

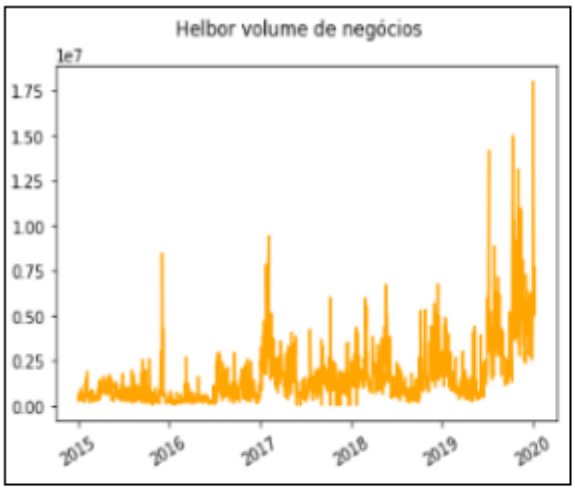


Figura 11 - Gráfico de volume de negócios da Helbor

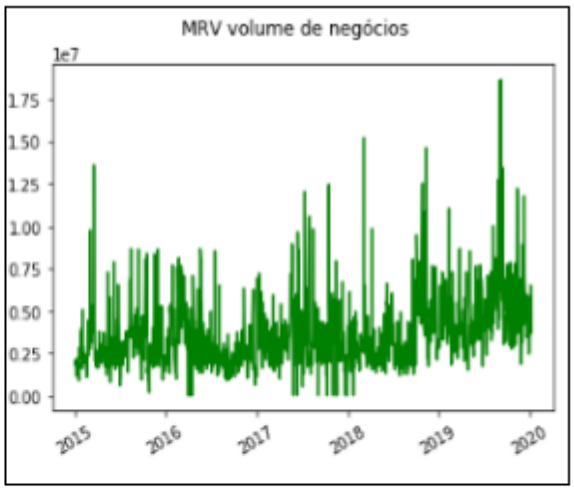


Figura 12 - Gráfico de volume de negócios da MRV

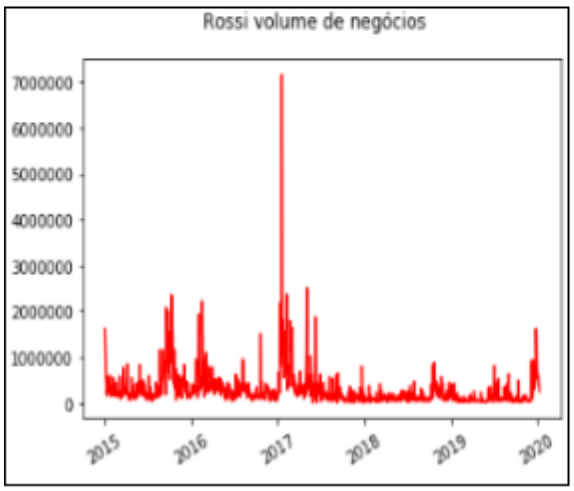


Figura 13- Gráfico de volume de negócios da Rossi

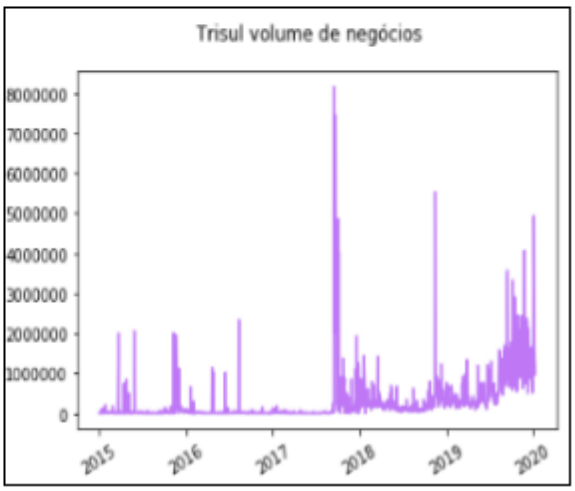


Figura 14- Gráfico de volume de negócios da Trisul

Analisando o volume de forma separada, chego a conclusão que realmente ocorreu um aumento do volume de negócios a partir de 2019, a qual pode ter sido influenciado pelos dois fatores apontados anteriormente, melhoria no setor de construção e a busca por ativos, que podem proporcionar ao investidor, um rendimento maior que ativos de renda fixa, uma vez que a diminuição da taxa selic tem diminuído neste período.

A seguir, observaremos a distribuição de frequência dos preços e sua concentração. Este gráfico é chamado de histograma.

Um histograma é um gráfico de frequência que tem como objetivo ilustrar como uma determinada amostra ou população de dados está distribuída. Ele, mede quantas vezes temos determinado valor dentro dessa nossa distribuição de dados.

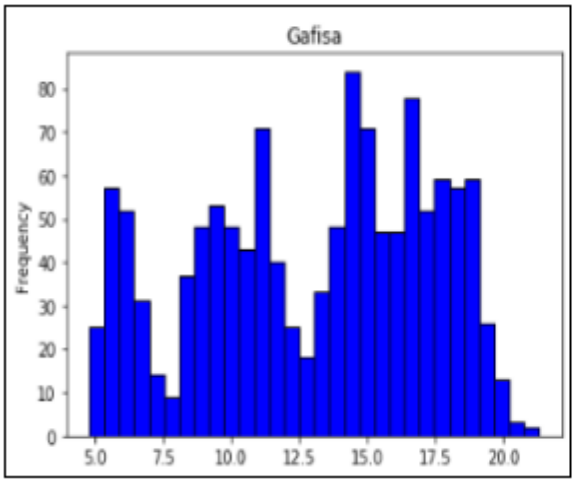


Figura 15 – Histograma dos preços Gafisa

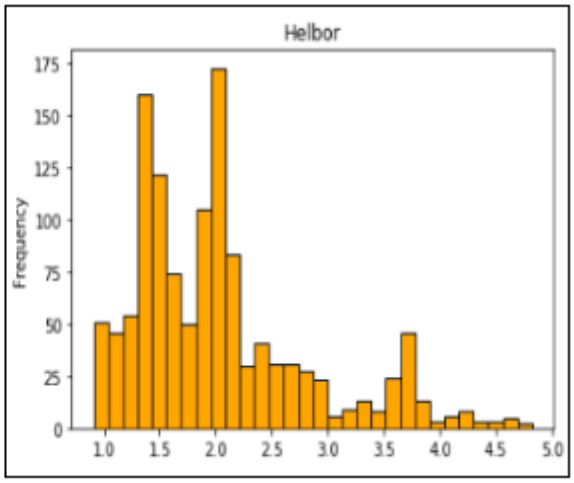


Figura 16 - Histograma dos preços Helbor

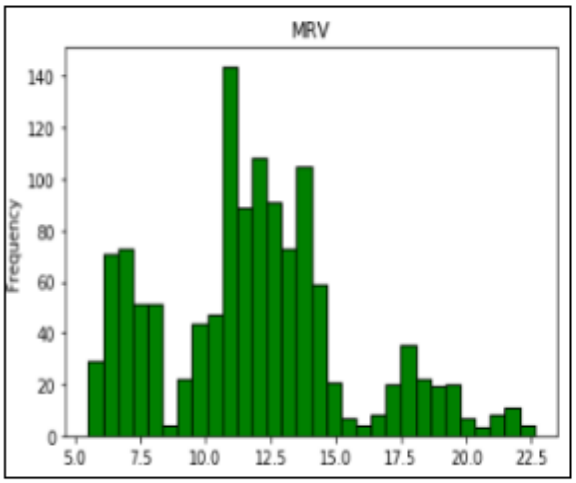


Figura 17 – Histograma dos preços MRV

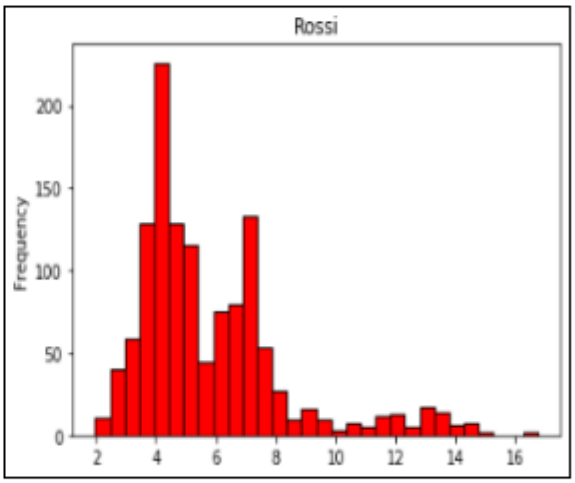


Figura 18 – Histograma dos preços Rossi

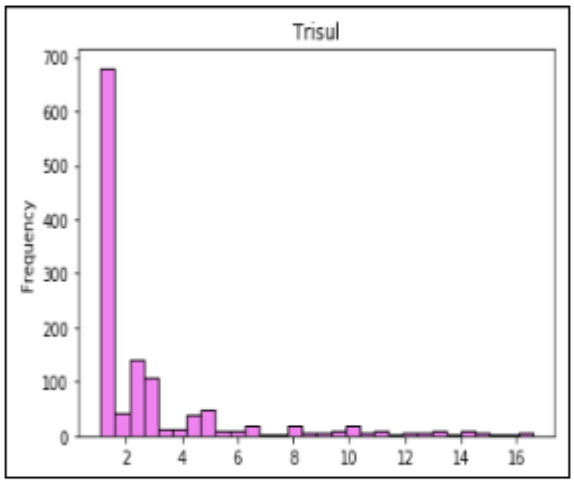


Figura 19 – Histograma dos preços Trisul

O código para gerar o gráfico de histograma, é apresentado na sequência. Variei apenas o nome do daset, a cor e o título para gerar todos os gráficos.

```
#distribuição dos preços usando o encadeamento
datasetGafisa["Close"].plot.hist(bins=30, edgecolor='black', facecolor='blue').set_title('Gafisa')
```

Agora vamos analisar a correlação entre as variáveis preço de abertura x preço de fechamento, preço máximo x preço de fechamento, e preço mínimo x preço de fechamento, através do gráfico de dispersão. O gráfico de dispersão possibilita visualizar a distribuição dos pares de dados (x, y) para encontrar a melhor relação entre os pontos (prof. Gabriel Fonseca 2019).

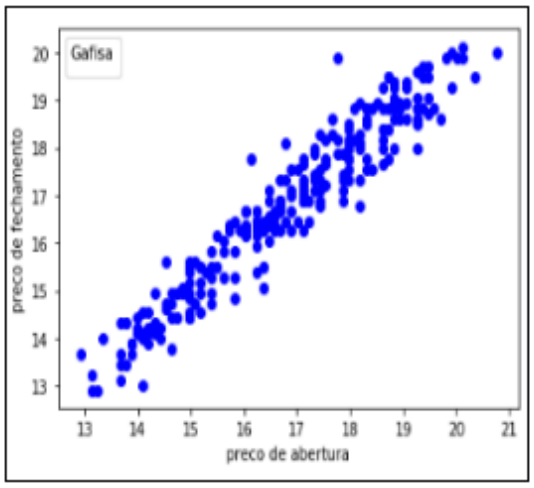


Figura 20 - Gráfico de dispersão fechamento x abertura

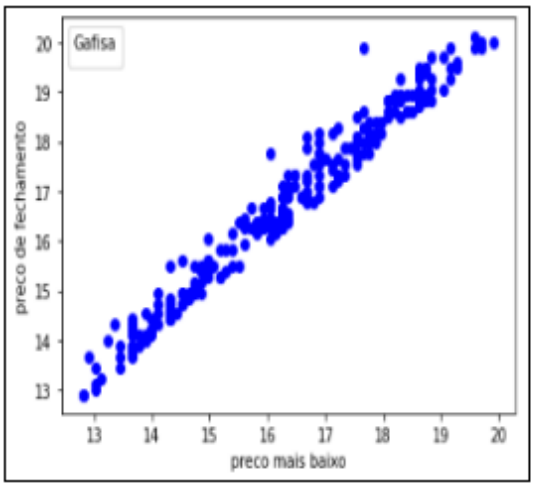


Figura 21 – Gráfico de dispersão fechamento x mínimo

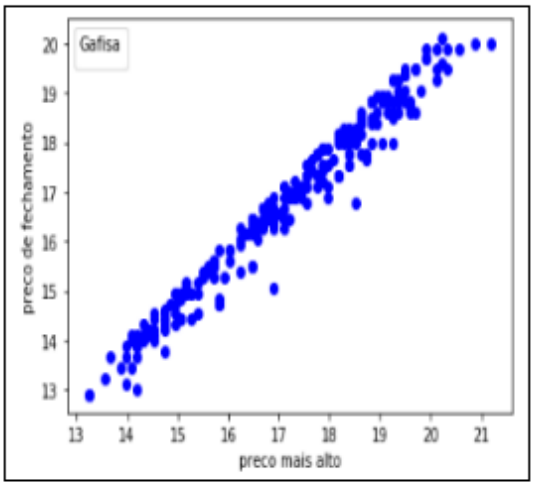


Figura 22 - Gráfico de dispersão fechamento x máximo

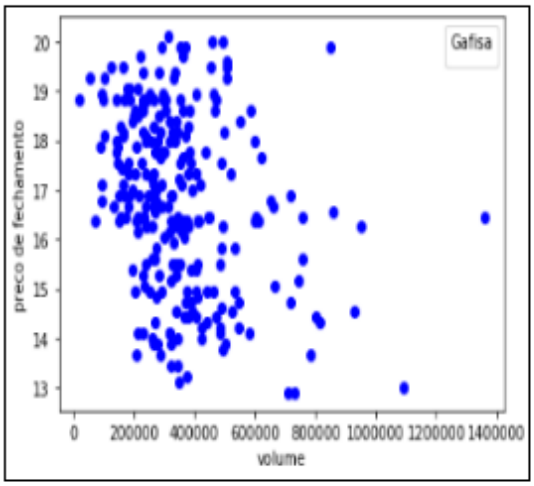


Figura 23 – Gráfico de dispersão fechamento x volume

Estes gráficos acima, foram gerados a partir do código demonstrado na sequência. Lembrando, continuo utilizando a biblioteca “Matplotlib”, a qual deve ser im-

portada para a construção do gráfico. Utilizei a variável 252, representando os 252 dias úteis de um período de 1 ano, padrão este, utilizado pelo mercado financeiro brasileiro no cálculo de juros.

```
#Existe uma correlação entre essas duas variáveis
treinoGafisa = datasetGafisa
x = treinoGafisa.Open[:252]
y = treinoGafisa.Close[:252]
plt.scatter(x,y,color='b')
plt.xlabel('preco de abertura')
plt.ylabel('preco de fechamento')
plt.axis([min(x),max(x),min(y),max(y)])
plt.autoscale('False')
plt.legend(title = "Gafisa")
plt.show()
```

Conforme análise, identificamos a existência de uma forte correlação positiva entre os preços de abertura x fechamento, mínimo x fechamento e máximo x fechamento. No entanto, não existi esta correlação entre o preço de fechamento x volume.

Esta correlação entre os preços nas mesmas variáveis, também foram identificados nas análises das demais empresas do setor.

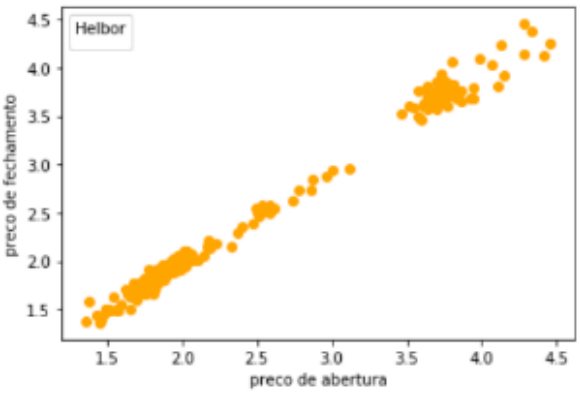


Figura 24 – Gráfico de dispersão fechamento x abertura

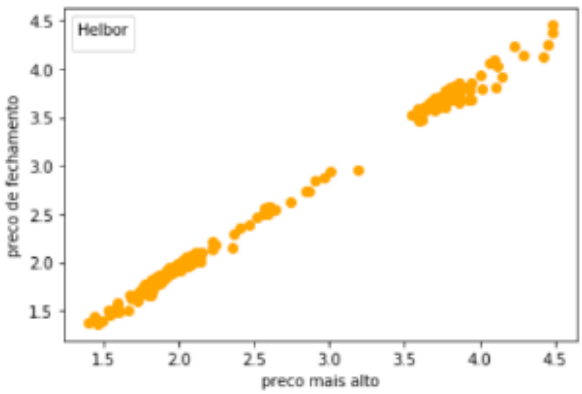


Figura 25 – Gráfico de dispersão fechamento x máximo

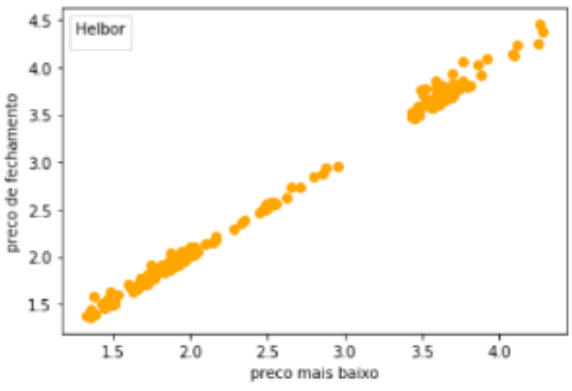


Figura 26 – Gráfico de dispersão fechamento x mínimo

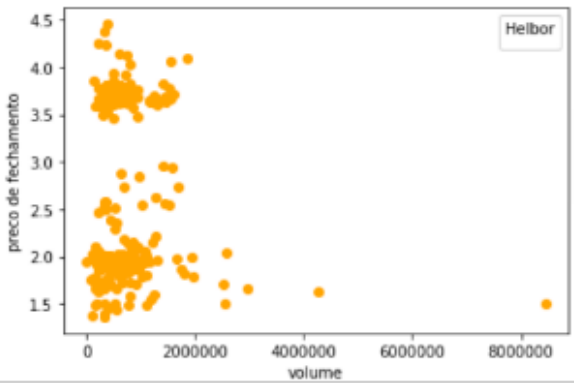


Figura 27– Gráfico de dispersão fechamento x volume

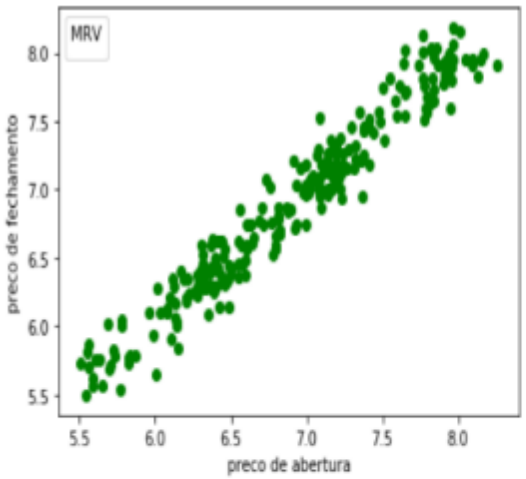


Figura 28 - Gráfico de dispersão fechamento x abertura

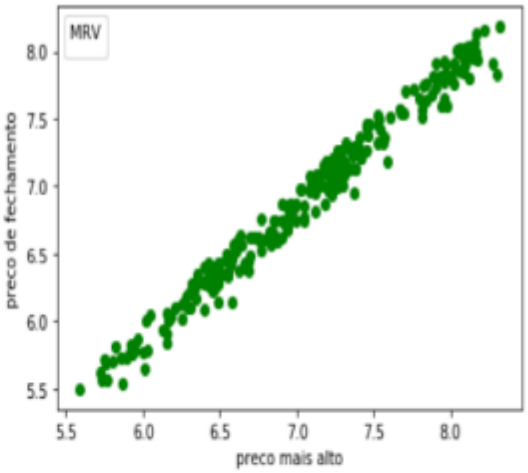


Figura 29 - Gráfico de dispersão fechamento x máximo

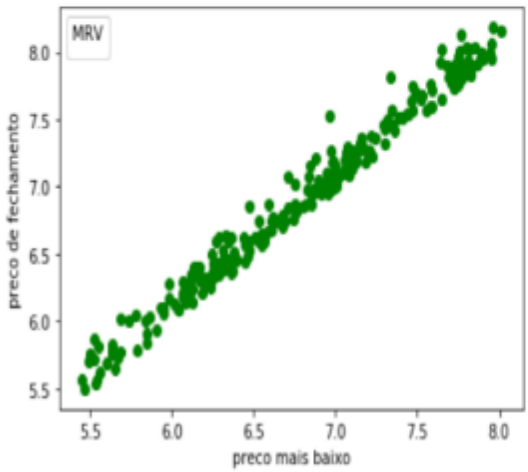


Figura 30 - Gráfico de dispersão fechamento x mínimo

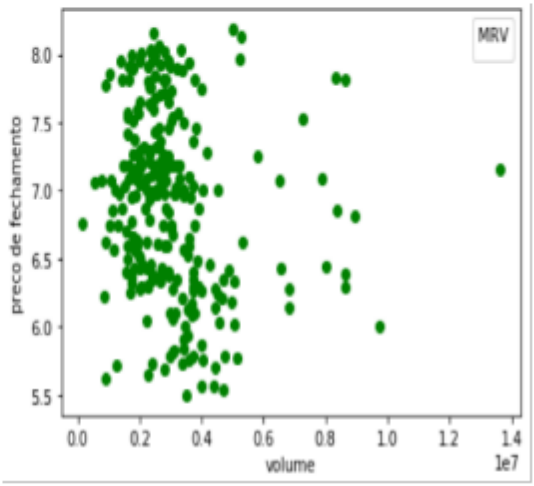


Figura 31 - Gráfico de dispersão fechamento x volume

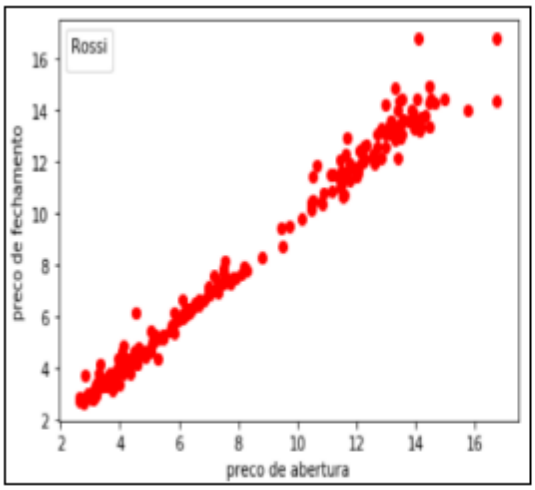


Figura 32 - Gráfico de dispersão fechamento x abertura

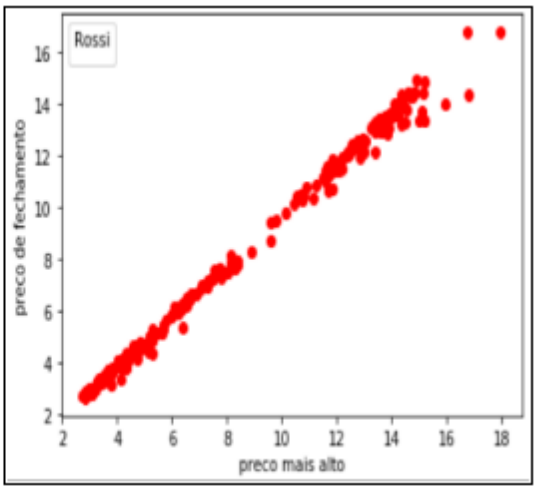
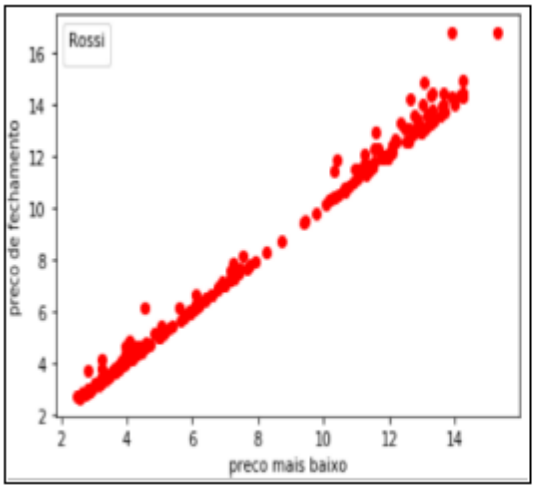


Figura 33 - Gráfico de dispersão fechamento x máximo



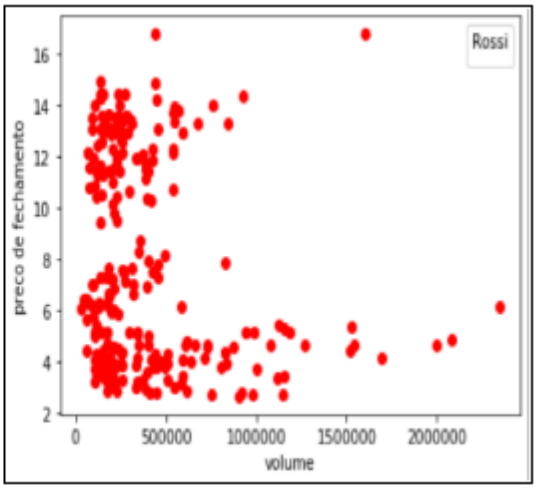
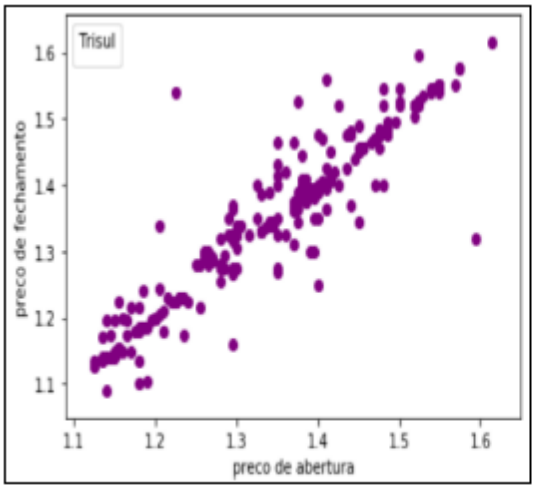


Figura 34 - Gráfico de dispersão fechamento x mínimo Figura 35 - Gráfico de dispersão fechamento x volume



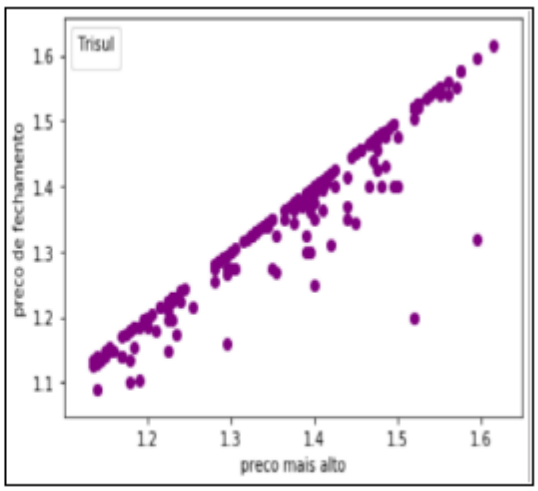
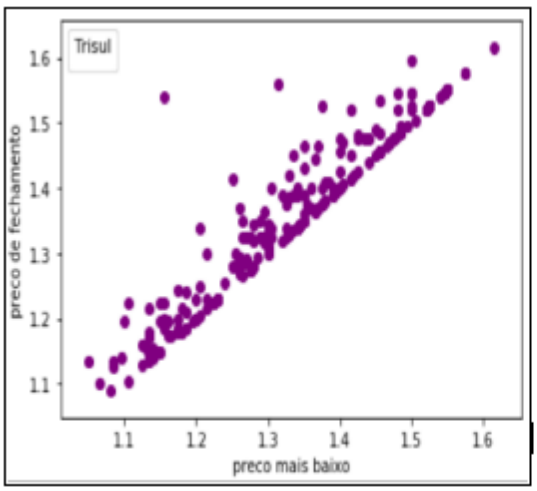


Figura 36 - Gráfico de dispersão fechamento x abertura Figura 37 - Gráfico de dispersão fechamento x máximo



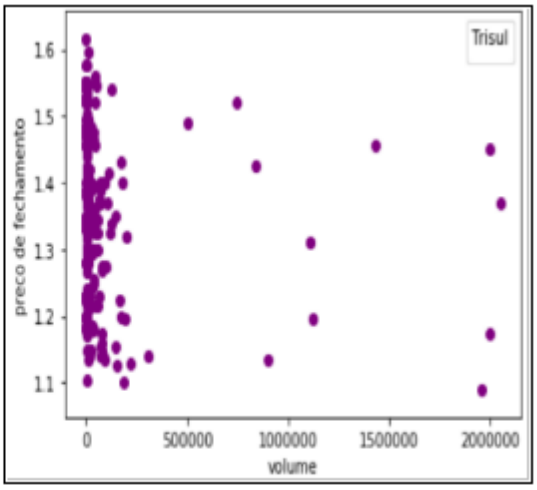


Figura 38 - Gráfico de dispersão fechamento x mínimo Figura 39 - Gráfico de dispersão fechamento x volume

Implementando o cálculo da média móvel, vamos utilizar a média de m dias observados para prever o próximo dia. Em estatística a média móvel, é um recurso utilizado para se identificar a tendência de um conjunto de dados dispostos em uma série temporal. Ou seja, a média móvel é utilizada para se entender para qual caminho os dados parecem apontar; se vai seguir uma tendência de alta, se vai cair ou se permanecerá estacionário.

Utilizei os valores de 8 e 21 dias, no entanto, não é um padrão de mercado, e sim os que geralmente utilizo. Entendo ser os melhores valores aqueles que você melhor se adapta, e traz assim os melhores resultados.

Segue o código para implementar o cálculo.

```
#Aplicando as médias móveis
datasetGafisa = pd.DataFrame(datasetGafisa[['Close','Date']])
datasetGafisa.set_index('Date', inplace=True)
datasetGafisa['MM_8'] = datasetGafisa['Close'].rolling(8).mean().shift()
datasetGafisa['MM_21'] = datasetGafisa['Close'].rolling(21).mean().shift()
```

Para uma melhor visualização do gráfico, limitei o período de análise em 252 dias. Segue o código que monta o gráfico.

```
#Visualizando a média móvel, dentro de um período de 252 dias.

limit = 252
plt.figure(figsize=(15,10))
plt.grid(True)
plt.plot(datasetGafisa['Close'][-limit:], label='Preço de Fechamento')
plt.plot(datasetGafisa['MM_8'][-limit:], label='Médias período de 8 dias')
plt.plot(datasetGafisa['MM_21'][-limit:], label='Médias período de 21 dias')
plt.legend(loc=2)
plt.legend(title = "Gafisa")
plt.show()
```

Figura 40 – Gráfico das médias móveis empresa Gafisa

É possível notar a formação de tendências ao longo do tempo, os pequenos movimentos no preço não mostram muita coisa, mas, a tendência no preço é extremamente importante para o investidor.

- Sinal de Compra: quando os preços cruzam para cima da Média Móvel.
- Sinal de Venda: quando os preços cruzam para baixo da Média Móvel.

No gráfico, fiz a identificação de alguns destes pontos, apenas como forma de exemplificação. Com as setas em azul, pontos de compra e setas em vermelho, pontos de venda.

5. Criação de Modelos de Machine Learning

Machine Learning ou aprendizado de máquina é uma área da ciência da computação que estuda meios para que máquinas possam fazer tarefas que seriam executadas por pessoas.

Modelo preditivo, consiste em uma função matemática que, quando aplicada a uma massa de dados, é capaz de identificar padrões e oferecer uma previsão do que pode ocorrer.

5.1 Modelo 1 – Regressão Linear

Regressão Linear, em estatística, é uma equação para se estimar a condicional (valor esperado) de uma variável y, dados os valores de algumas outras variáveis x. Modelos de regressão linear são frequentemente ajustados usando a abordagem dos mínimos quadrados.

A análise de regressão estuda a relação entre uma variável chamada de variável dependente e outras variáveis chamadas variáveis independentes.

Para exercitar este nosso objetivo no trabalho, vou utilizar a mesma base de dados, o dataset da empresa Gafisa. No jupyter notebook, importo as bibliotecas necessárias e o dataset.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

datasetGafisa = pd.read_csv('C:/Users/Lapin/Documents/Pytest/dados/GFSA35A.csv')
```

A Scikit-Learn é uma biblioteca Python para trabalhar com Machine Learning. Neste trabalho, quero prever o preço de fechamento diário da ação da empresa Gafisa. Vou criar um dataset, chamado treino com as variáveis de preço de abertura, preço de máximo, preço mínimo e volume. O preço de fechamento não será utilizado neste novo dataset.

```
treino = datasetGafisa
dados = ['Open','High','Low','Volume']
treino = treino[dados]
y = datasetGafisa['Close']
```

Seguindo nosso exemplo para aplicar Machine Learning, iremos agora dividir o novo dataset, para isso usamos o método train_test_split, este comando divide um percentual dos dados para treino e outro para teste. Se você não especificar no comando, a divisão fica 75% para treino e 25% para teste. Outra observação é que os números nas listas após a divisão não seguem a mesma ordem ascendente de antes. Em outra palavra, por padrão, o programa ignora a ordem original dos dados. Ele seleciona dados aleatoriamente para formar o conjunto de treinamento e teste.

Segue o código que faz exatamente o comentado acima:

```
X_treino, X_teste, y_treino, y_teste = train_test_split(treino, y,random_state=42)
```

Toda vez que você usa random_state = 42, sempre terá a mesma saída na primeira vez que fizer a divisão. Isso é útil se você deseja resultados reproduzíveis, por exemplo, na documentação, para que todos possam ver consistentemente os mesmos números ao executar os exemplos.

Depois de dividirmos os dados, entre treino e teste, vamos criar um objeto do tipo “LinearRegression” e a partir disto estamos aptos para treinar nosso modelo através do método fit.

```
#Criando um modelo do tipo LinearRegression
meu_modelo = LinearRegression()
#Treinando o modelo utilizando o método fit
meu_modelo.fit(X_treino,y_treino)
```

Com o modelo treinado, podemos visualizar os coeficientes. Este coeficiente é uma medida do grau de dependência linear entre as duas variáveis, X e Y. Pode ser utilizado como uma medida da qualidade do ajustamento, ou como medida da confiança depositada na equação de regressão como instrumento de previsão.

```
In [6]: meu_modelo.coef_
Out[6]: array([-5.62279863e-01,  8.29654080e-01,  7.36376529e-01, -8.53821847e-09])
```

Lembra que quando utilizamos o método “train_test_split”, foi gerado um dataset com os dados selecionados para teste. É utilizando este dataset chamado de X_teste que passei como argumento do método “predict”. O método predict retorna os dados de fechamento previstos.

```
In [8]: meu_modelo.predict(X_teste)

Out[8]: array([10.52940647,  5.21821641, 15.36722422,  9.9784941 , 15.2470691 ,
               14.38071233, 11.77267834, 20.18886173, 14.37430434, 17.67363521,
                5.8481956 , 15.69284279, 18.26440067, 15.24739958,  5.90187693,
                7.89937524,  8.43275238, 11.18208441,  8.14532867,  6.58032196,
               19.38233731, 11.00855676,  6.2203025 , 14.14216219, 16.79027193,
                8.59998163, 17.54707744,  9.47819218,  6.11546735,  6.68569056,
               11.94276113,  9.48758641,  9.35293528, 16.38782857, 10.92762165,
                6.13701718, 17.27513187, 18.30712602, 16.45987172,  9.75602277,
               17.46344776, 11.10042991,  5.09411815, 18.67036914, 18.0862289 ,
```

Na figura acima, a linha de código utilizada para prever os preços de fechamentos, bem como a saída que são os preços de fechamento previstos.

Para efeito de análise e comparação do preço de fechamento previsto x preço de fechamento real, vou selecionar 20 registros conforme abaixo. Na saída [9] eu tenho os dados previsto e na saída [10] eu tenho os dados real.

```
In [9]: meu_modelo.predict(X_teste)[:20]

Out[9]: array([10.52940647,  5.21821641, 15.36722422,  9.9784941 , 15.2470691 ,
               14.38071233, 11.77267834, 20.18886173, 14.37430434, 17.67363521,
                5.8481956 , 15.69284279, 18.26440067, 15.24739958,  5.90187693,
                7.89937524,  8.43275238, 11.18208441,  8.14532867,  6.58032196])

In [10]: y_teste[:20]

Out[10]: 680     10.679100
         1102     5.410000
         394     15.065900
         930      9.967880
         497     15.388800
         462     14.635500
         950     11.817100
         81      19.908501
         43      14.205000
         128     17.325800
         1170     5.830000
         743     16.101999
         513     18.079100
         461     15.173500
         1167     5.940000
         1239     8.050000
         1050     8.262960
         945     11.447300
         1043     8.082550
         1139     6.440000
         Name: Close, dtype: float64
```

Preço Real	Preço Estimado	Diferença
R$ 10,67910000	R$ 10,52940647	R$ 0,14969353
R$ 5,41000000	R$ 5,21821641	R$ 0,19178359
R$ 15,06590000	R$ 15,36722422	-R$ 0,30132422
R$ 9,96788000	R$ 9,97849410	-R$ 0,01061410
R$ 15,38880000	R$ 15,24706910	R$ 0,14173090
R$ 14,63550000	R$ 14,38071233	R$ 0,25478767
R$ 11,81710000	R$ 11,77267834	R$ 0,04442166
R$ 19,90850100	R$ 20,18886173	-R$ 0,28036073
R$ 14,20500000	R$ 14,37430434	-R$ 0,16930434
R$ 17,32580000	R$ 17,67363521	-R$ 0,34783521
R$ 5,83000000	R$ 5,84819560	-R$ 0,01819560
R$ 16,10199900	R$ 15,69284279	R$ 0,40915621
R$ 18,07910000	R$ 18,26440067	-R$ 0,18530067
R$ 15,17350000	R$ 15,24739958	-R$ 0,07389958
R$ 5,94000000	R$ 5,90187693	R$ 0,03812307
R$ 8,05000000	R$ 7,89937524	R$ 0,15062476
R$ 8,26296000	R$ 8,43275238	-R$ 0,16979238
R$ 11,44730000	R$ 11,18208441	R$ 0,26521559
R$ 8,08255000	R$ 8,14532867	-R$ 0,06277867
R$ 6,44000000	R$ 6,58032196	-R$ 0,14032196

Na tabela acima, constatamos que existe uma diferença entre os preços de fechamento real e o preço de fechamento estimado, este último calculado pelo meu modelo. Já podemos de cara notar que as diferenças não seguem um padrão, sendo as vezes calculados a maior e por outras vezes calculados a menor.

Vou utilizar a métrica “Root Mean Square Error” para validar o modelo e verificar o desvio padrão médio. Segue o código utilizado:

```
#Validando o modelo
RMSE = mean_squared_error(y_teste, meu_modelo.predict(X_teste))**0.5
RMSE
```

```
0.1983745195930774
```

Arredondando, o erro médio do meu modelo foi de 0,20 centavos entre os preços.

Agora, vamos aplicar um novo modelo, deixando de fora a coluna “Volume”.

```
#Novo modelo
meu_modelo2 = LinearRegression()
features = ['Open','High', 'Low']
treino2 = treino[features]
```

Os demais passos, são todos iguais.

```
X_treino, X_teste, y_treino, y_teste = train_test_split(
treino2, y, random_state=42)
```

```
meu_modelo2.fit(X_treino,y_treino)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

```
meu_modelo2.coef_
```

```
array([-0.55878929,  0.81504092,  0.74937099])
```

```
meu_modelo2.predict(X_teste)[:20]
```

```
meu_modelo2.predict(X_teste)[:20]
```

```
array([10.51720052,  5.21621284, 15.36826812,  9.97061876, 15.24662262,
       14.38430598, 11.77442873, 20.19204598, 14.36883564, 17.67047932,
        5.84639741, 15.69667441, 18.26961556, 15.24506638,  5.90609688,
        7.92550144,  8.42729143, 11.19370435,  8.14458676,  6.5724257 ])
```

```
y_teste[:20]
```

```
680     10.679100
1102     5.410000
394     15.065900
930      9.967880
497     15.388800
462     14.635500
950     11.817100
81      19.908501
43      14.205000
128     17.325800
1170     5.830000
743     16.101999
513     18.079100
461     15.173500
1167     5.940000
1239     8.050000
1050     8.262960
945     11.447300
1043     8.082550
1139     6.440000
Name: Close, dtype: float64
```

Aplicando a métrica “Root Mean Square Error” para validar o modelo, consta-to que praticamente não ocorreu diferença do valor do desvio padrão médio.

```
RMSE = mean_squared_error(y_teste, meu_modelo2.predict(X_teste))**0.5
RMSE
```

```
0.1986556067047513
```

Com o objetivo de melhorar nosso modelo, testo um terceiro modelo, agora somente com as colunas de “Preço Máximo” e “Preço Mínimo”.

```
#Terceiro modelo
meu_modelo3 = LinearRegression()
features = ['High', 'Low']
treino3 = treino[features]
```

```
X_treino, X_teste, y_treino, y_teste = train_test_split(
treino3, y, random_state=42)
```

```
meu_modelo3.fit(X_treino,y_treino)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

```
meu_modelo3.coef_
```

```
array([0.52030955, 0.48476655])
```

```
RMSE = mean_squared_error(y_teste, meu_modelo3.predict(X_teste))**0.5
RMSE
```

```
0.22989987925281297
```

Conforme resultado apresentado, este modelo piorou em relação aos demais modelos, uma vez que o desvio padrão arredondado é de R$0,23 centavos.

5.2 Modelo 2 - LSTM

Uma outra forma de prever o comportamento de uma ação é utilizando LSTM. E o que é LSTM?

LSTM (Long Short Term Memory ou em português, memória de curto e longo prazo), é um tipo de rede neural recorrente, pois são capazes de aprender conexões de longo prazo. Dessa maneira, elas têm um incrível poder de predição e funcionam muito bem em uma variada gama de problemas. É usada em diversos cenários de Processamento de Linguagem Natural, sendo uma arquitetura de rede neural

recorrente (RNN). As RNN são redes utilizadas para reconhecer padrões quando os resultados do passado influenciam no resultado atual.

A LSTM é bem adequada para classificar, processar e prever séries temporais com intervalos de tempo de duração desconhecida.

Simplificando o entendimento, os humanos não começam a pensar do zero a cada segundo. Nós entendemos cada palavra com base em nossa compreensão das palavras anteriores aprendidas. Nossos pensamentos têm persistência.

O objetivo neste trabalho não é detalhar a arquitetura LSTM, e sim fazer apenas uma breve introdução, e mostrar como aplicar na previsão do comportamento do preço de uma ação. No final do trabalho, deixo links onde é possível se aprofundar no conhecimento desta arquitetura.

Para aplicar o LSTM, devemos utilizar a biblioteca Keras. Keras é uma biblioteca de rede neural de alto nível, de código aberto escrito em Python, e roda como frontend em TensorFlow. A ideia é facilitar a implementação do TensorFlow. Por sua vez, TensorFlow é uma biblioteca de código aberto para aprendizado de máquina aplicável a uma ampla variedade de tarefas. É um sistema para criação e treinamento de redes neurais para detectar e decifrar padrões e correlações, análogo à forma como humanos aprendem e raciocinam.

Não fora possível a utilização destas bibliotecas no meu ambiente local, sendo necessário a aplicação na nuvem. Para isso, utilizei a plataforma do Google, mais propriamente a ferramenta Google Colab.

O primeiro passo no Google Colab Notebook é a importação das bibliotecas.

```
import numpy as np
import pandas as pd
from keras.models import Sequential
from keras.layers import LSTM,Dense
import matplotlib.pyplot as plt
from google.colab import files
```

Neste ambiente, a importação do arquivo dataset é um pouco diferente do utilizado no ambiente Jupyter Notebook. Você deve utilizar o comando upload() conforme abaixo.

```
[ ]  uploaded = files.upload()
```

Escolher arquivos | Nenhum arquivo selecionado Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving GFSA35A.csv to GFSA35A (2).csv

Ao executar o comando upload( ) é aberto uma caixa de seleção. Basta você clicar nesta opção, e selecionar o arquivo que você deseja ler. No meu caso, continuo utilizando o dataset da empresa Gafisa.

Como o objetivo é prever o que acontecerá com o valor de fechamento de uma ação em “n” dias (forward_days), tendo como base os “m” dias anteriores (look_back), testando com “k” períodos (num_periods), onde cada período é um conjunto de n dias previstos. Defino aqui as variáveis a serem utilizadas.

```
look_back = 30
foward_days = 7
num_periods = 21
```

Posteriormente, definimos as datas como index e mantemos apenas a coluna que queremos prever, que é o preço de fechamento da ação.

```
df = pd.read_csv('GFSA35A.csv')
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)
df = df['Close']
df.head()
```

```
Date
2015-01-02    14.8507
2015-01-05    14.5278
2015-01-06    14.7431
2015-01-07    15.1735
2015-01-08    15.4964
Name: Close, dtype: float64
```

O próximo passo é normalizar os dados. O objetivo da normalização é alterar os valores das colunas numéricas no conjunto de dados para uma escala comum, sem distorcer as diferenças nos intervalos de valores. Para o aprendizado de máquina, nem todos os conjuntos de dados requerem normalização. No entanto, ouve-se com frequência que, durante o processo de preparação do seu dataset para a criação de seu modelo de aprendizado de máquina, é importante normalizar os dados para que o modelo venha a performar melhor. Por esse motivo, inclui a normalização nesta etapa.

```
array = df.values.reshape(df.shape[0],1)
array[:5]
```

```
array([[14.8507],
       [14.5278],
       [14.7431],
       [15.1735],
       [15.4964]])
```

```
[7] from sklearn.preprocessing import MinMaxScaler
    scl = MinMaxScaler()
    array = scl.fit_transform(array)
    array[:5]
```

```
array([[0.6095661 ],
       [0.59004078],
       [0.60305968],
       [0.62908537],
       [0.64861069]])
```

A seguir, dividimos os dados em Treino/Validação para o modelo LSTM e os dados de Teste. Separamos para os últimos “k” períodos (variável num_periods) para testar o modelo. A cada período, o modelo preverá os próximos “n” dias. O resto será utilizado para o treinamento (Treino e Validação).

```
#Dividir os dados em treino e teste

division = len(array) - num_periods*foward_days
array_test = array[division-look_back:]
array_train = array[:division]
```

Obtemos os dados e as divisões na entrada X e na saída Y, dividindo nos dias passados “n” como entrada X e “m” nos próximos dias como Y. Depois, dividimos a matriz em subconjuntos aleatórios de treino e teste.

```
def processData(data, look_back, foward_days,jump=1):
    X,Y = [],[]
    for i in range(0,len(data) -look_back -foward_days +1, jump):
        X.append(data[i:(i+look_back)])
        Y.append(data[(i+look_back):(i+look_back+foward_days)])
    return np.array(X),np.array(Y)
```

```
X_test,y_test = processData(array_test,look_back,foward_days,foward_days)
y_test = np.array([list(a.ravel()) for a in y_test])

X,y = processData(array_train,look_back,foward_days)
y = np.array([list(a.ravel()) for a in y])

from sklearn.model_selection import train_test_split
X_train, X_validate, y_train, y_validate = train_test_split(X, y, test_size=0.20, random_state=42)
```

```
print(X_train.shape)
print(X_validate.shape)
print(X_test.shape)
print(y_train.shape)
print(y_validate.shape)
print(y_test.shape)
```

```
(853, 30, 1)
(214, 30, 1)
(21, 30, 1)
(853, 7)
(214, 7)
(21, 7)
```

No passo anterior, ainda aplicamos o “shape”. O atributo shape retorna uma tupla que consiste nas dimensões do “array”, mostrando quantas linha e colunas temos.

Agora construímos e treinamos o modelo. No entanto, voltamos um pouco para a definição do LSTM. A ideia principal é que uma Rede Neural Recorrente possa ser alimentada reversamente com dados relevantes ao modelo, no caso séries históricas do preço da ação, para realizar previsões. Podemos, por exemplo, definir que o modelo receberá como entrada dados de 21 dias e retorna o preço do dia seguinte. Sem treinar o modelo, o resultado será irrelevante, mas se treinarmos por várias vezes os dados de preço de fechamento, espera-se que o modelo seja calibrado para prever melhor o preço.

O número de neurônios representa o número de unidades LSTM. Enquanto, o conjunto de treino é alimentado à rede, tantas vezes quanto o número de épocas.

```
NUM_NEURONS_FirstLayer = 50
NUM_NEURONS_SecondLayer = 30
EPOCHS = 50

#Construindo o modelo
model = Sequential()
model.add(LSTM(NUM_NEURONS_FirstLayer,input_shape=(look_back,1), return_sequences=True))
model.add(LSTM(NUM_NEURONS_SecondLayer,input_shape=(NUM_NEURONS_FirstLayer,1)))
model.add(Dense(foward_days))
model.compile(loss='mean_squared_error', optimizer='adam')

history = model.fit(X_train,y_train,epochs=EPOCHS,validation_data=(X_validate,y_validate),shuffle=True,batch_size=2, verbose=2)
```

```
Epoch 3/50
 - 17s - loss: 0.0062 - val_loss: 0.0053
Epoch 4/50
 - 17s - loss: 0.0055 - val_loss: 0.0071
Epoch 5/50
 - 17s - loss: 0.0055 - val_loss: 0.0041
Epoch 6/50
 - 17s - loss: 0.0052 - val_loss: 0.0043
Epoch 7/50
 - 17s - loss: 0.0047 - val_loss: 0.0040
Epoch 8/50
 - 16s - loss: 0.0045 - val_loss: 0.0035
Epoch 9/50
 - 17s - loss: 0.0042 - val_loss: 0.0069
```

Treine em 853 amostras, valide em 214 amostras.

Uma rede neural artificial é formado por um conjunto de camadas. Essas camadas extraem representações dos dados de entrada que tendem a ser mais significativas para a resolução do problema. Basicamente, as redes neurais requerem duas principais decisões arquitetural:

- Quantas camadas será usada no modelo?
- Quantas neurônios (unidades) cada camada conterá?

Mais neurônios permitem mais liberdade em aprender dados mais complexos, porém torna a rede mais cara (lenta) computacionalmente para ser treinada.

O modelo sequencial permite inserir camadas em série, onde o “output” da primeira camada serve como “input” da segunda, e assim por diante.

Para criar previsões não lineares, somente uma camada não é suficiente. A presença de duas camadas possibilita redes neurais aprender qualquer função.

Loss, é a função de calcula a diferença entre os dados de teste e os dados de validação.

A chamada model.fit() retorna um objeto “History” que contém um dicionário com tudo o que rolou durante o treinamento.

```
plt.figure(figsize = (15,10))

plt.plot(history.history['loss'], label='loss')
plt.plot(history.history['val_loss'], label='val_loss')
plt.legend(loc='best')
plt.show()
```

O atributo history é um dicionário que registra valores de perda de treinamento e valores de métricas em épocas sucessivas, bem como valores de perda de validação e valores de métricas de validação.

Perda indica o qual distante as classes previstas se distanciam da verdade.

Quando utilizamos modelos Deep Learning, temos algumas métricas que nos ajudam a saber se o modelo está indo bem ou não.

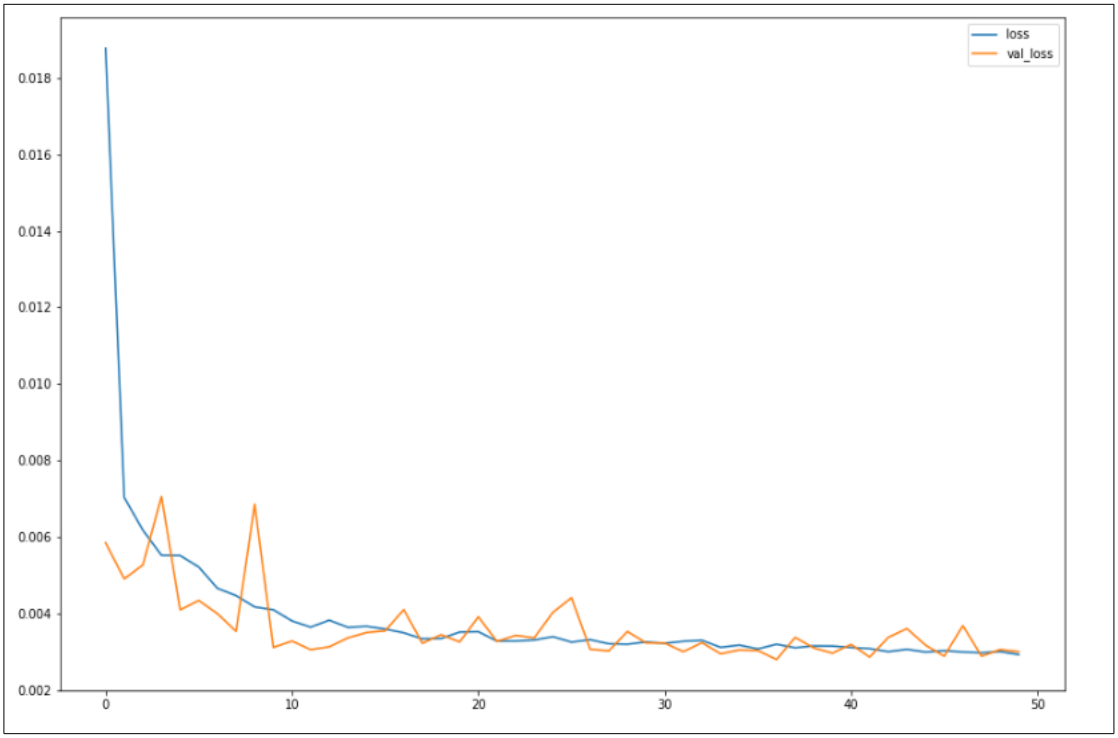


Figura 41 – Gráfico de loss e val_loss.

Se a rede está com a “loss” baixa, mas a “val_loss” (validation loss) não está descendo junto, então quer dizer que a rede está só “decorando” o conjunto dos dados de treino, ou seja, se adaptou muito bem aos dados com os quais está sendo

treinado e não está generalizando bem para dados não vistos antes (dados novos). Não aprendeu de fato o que diferencia aqueles dados para quando precisar enfrentar novos testes. Isso se chama “overfitting”.

Agora, vamos prever os dados de teste para o resultado.

```
Xt = model.predict(X_test)
```

Observando a parte de teste mais de perto.

```
plt.figure(figsize = (15,10))

for i in range(0,len(Xt)):
    plt.plot([x + i*foward_days for x in range(len(Xt[i]))], scl.inverse_transform(Xt[i].reshape(-1,1)), color='r')

plt.plot(0, scl.inverse_transform(Xt[i].reshape(-1,1))[0], color='r', label='Predição')

plt.plot(scl.inverse_transform(y_test.reshape(-1,1)), label='Alvo')
plt.legend(loc='best')
plt.show()
```

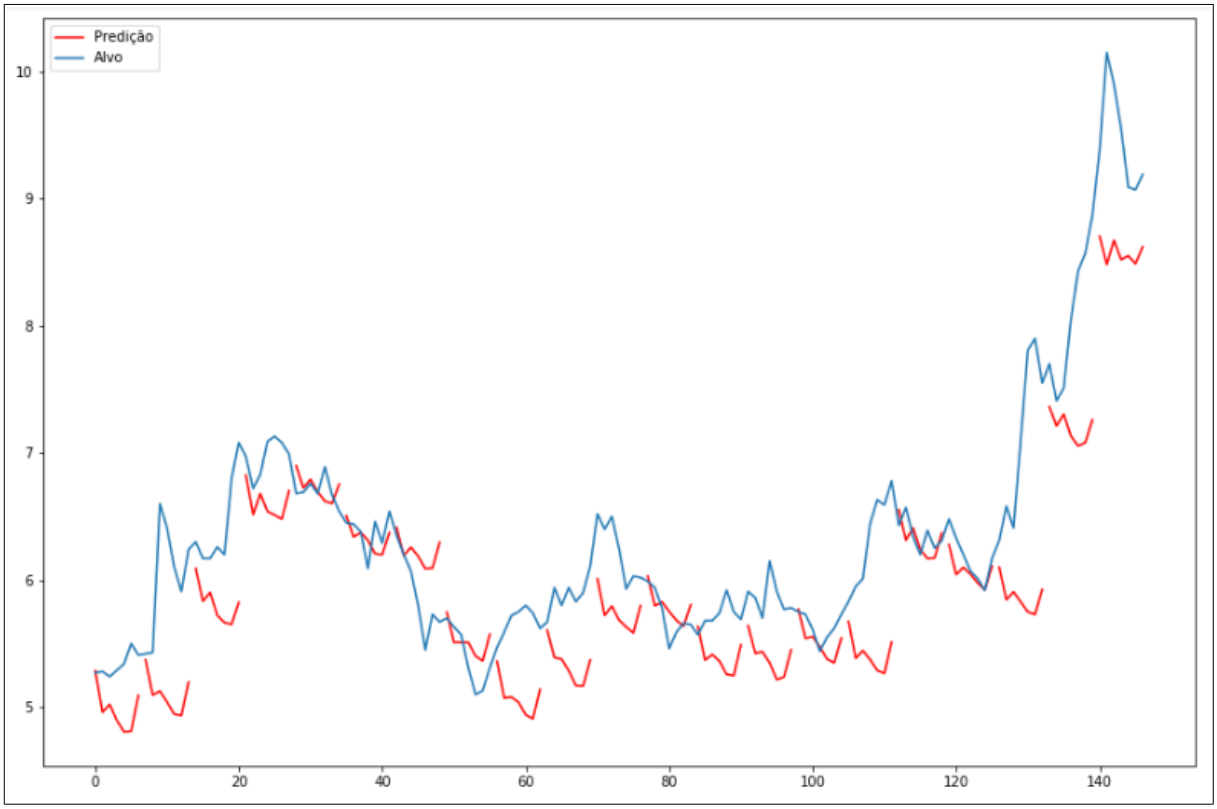


Figura 42 – Gráfico de predição x objetivo

Cada linha vermelha no gráfico representa uma previsão de 7 dias (forward_days), baseado nos 30 dias anteriores (look_back), e há 20 linhas verme-

lhas, pois escolhemos prever 21 períodos (num_periods). Por isso elas são descontínuas.

```
#separando em Treino e Teste

division = len(array) - num_periods*foward_days

leftover = division%foward_days+1

array_test = array[division-look_back:]
array_train = array[leftover:division]

Xtrain,ytrain = processData(array_train,look_back,foward_days,foward_days)
Xtest,ytest = processData(array_test,look_back,foward_days,foward_days)

Xtrain = model.predict(Xtrain)
Xtrain = Xtrain.ravel()

Xtest = model.predict(Xtest)
Xtest = Xtest.ravel()


y = np.concatenate((ytrain, ytest), axis=0)
```

Visualizando o resultado da predição no gráfico.

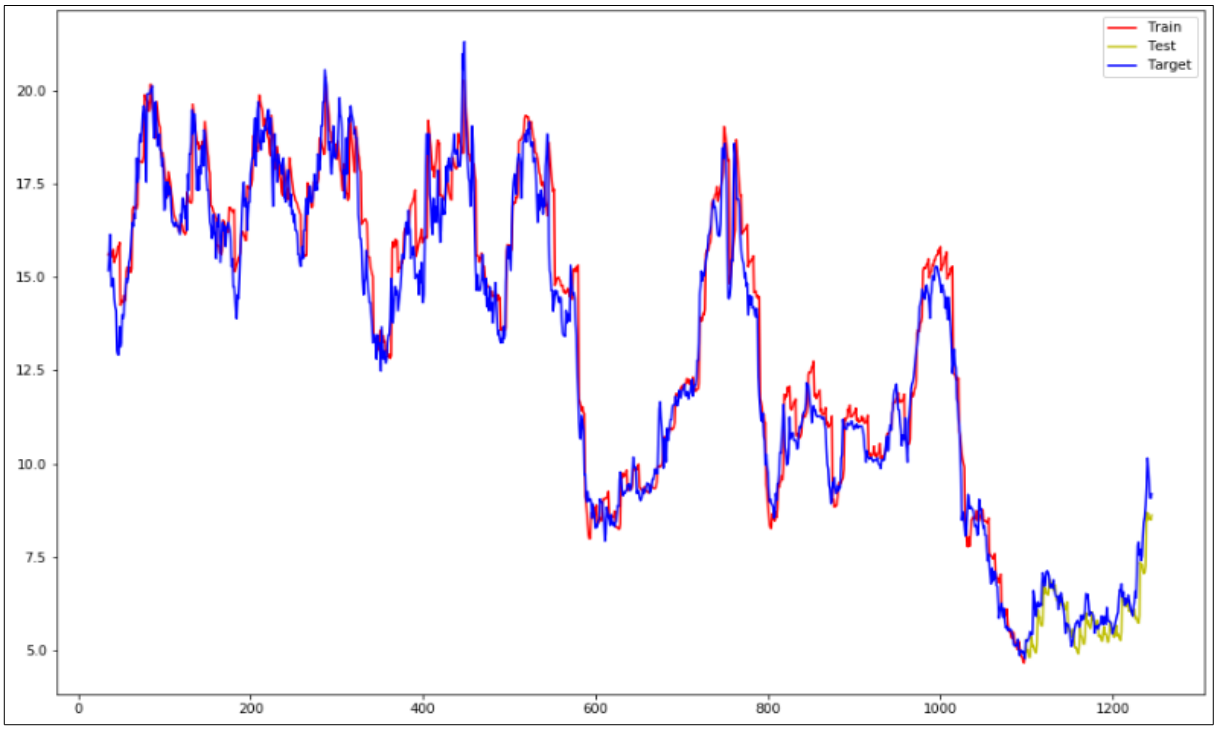


Figura 43 – Gráfico modelo de previsão de preço de fechamento empresa Gafisa

Para gerar este gráfico, utilizamos o código:

```
plt.figure(figsize = (15,10))

# Data in Train/Validation
plt.plot([x for x in range(look_back+leftover, len(Xtrain)+look_back+leftover)], scl.inverse_transform(Xtrain.reshape(-1,1)), color='r', label='Train')
# Data in Test
plt.plot([x for x in range(look_back +leftover+ len(Xtrain), len(Xtrain)+len(Xtest)+look_back+leftover)], scl.inverse_transform(Xtest.reshape(-1,1)), color='y', label='Test')

#Data used
plt.plot([x for x in range(look_back+leftover, look_back+leftover+len(Xtrain)+len(Xtest))], scl.inverse_transform(y.reshape(-1,1)), color='b', label='Target')

plt.legend(loc='best')
plt.show()
```

O ciclo de treino de uma rede neural, pode ser exemplificado seguindo os passos a seguir:

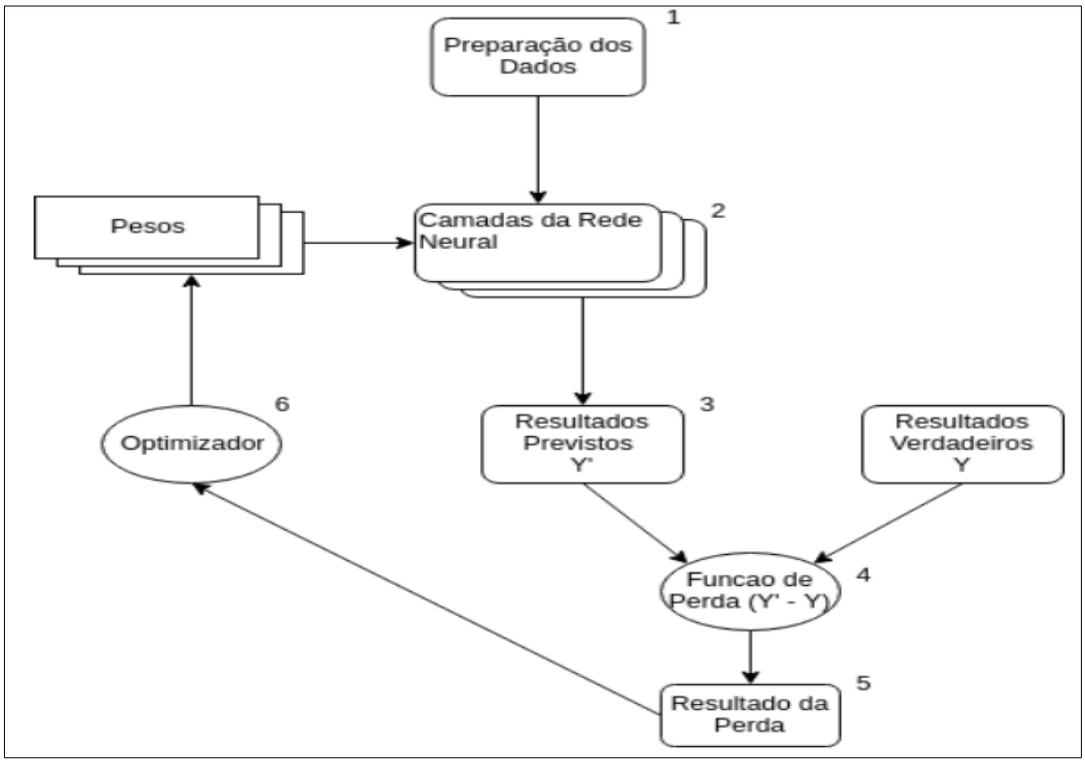


Figura 44 – Fluxo de treino de uma rede neural

Após toda a preparação dos dados, a divisão em testes e treino, temos a implementação das camadas, a quantidade de neurônios a serem utilizados, temos o cruzamento entre os dados previstos e dados reais, calculando a diferença, ajustando o modelo atribuindo pesos, e volta o fluxo de processamento até se chegar num valor considerável ideal.

6. Apresentação dos Resultados

Este trabalho apresentou a implementação de técnicas de ciência de dados dentre estas o aprendizado de máquina.

Predição do Preço da Ação		
Declaração do Problema Atualmente, 1.945.607 pessoas físicas investem na B3. Grande parte destas pessoas investem sem conhecimento, apenas por puro achismo ou seguindo recomendações.	Resultados / Previsões Prever o preço de fechamento da ação. Predizer o comportamento futuro da ação.	Aquisição de Dados Extraindo dados dos preços de ação do Yahoo Finance num período superior a 5 anos.
Modelagem Médias Móveis Análise de Regressão Linear LSTM	Avaliação do Modelo O resultado de avaliação do modelo se dará em comparação ao preço de fechamento real.	Preparação de dados Extrair dados reais de preço da ação. Validação de dados. (Dados nulos).

Num primeiro momento, registramos o aumento de investidores Pessoa Física por ativos de renda variável.

Figura 45 – Gráfico aumento de investidores pessoa física

Hoje, possuímos quase 2 milhões de pessoas que buscam por este tipo de investimento. E a tendência é crescer cada vez mais, face os produtos de renda fixa estarem com baixos retornos. O Instituto Brasileiro de Geografia e Estatística (IBGE) divulgou em Agosto de 2019 as estimativas da população. Pela data de referência

de 1º de julho de 2019, o Brasil tem uma população total de 210.147.125 pessoas. Este é mais um ingrediente que nos faz pensar da possibilidade de crescimento na renda variável.

Ainda é possível perceber, que os 3 maiores estados x investidores são todos da região sudeste do Brasil. A quantidade de mulheres (195.466) do estado de São Paulo é maior que o número total de investidores (178.083) do terceiro maior estado, no caso Minas Gerais.

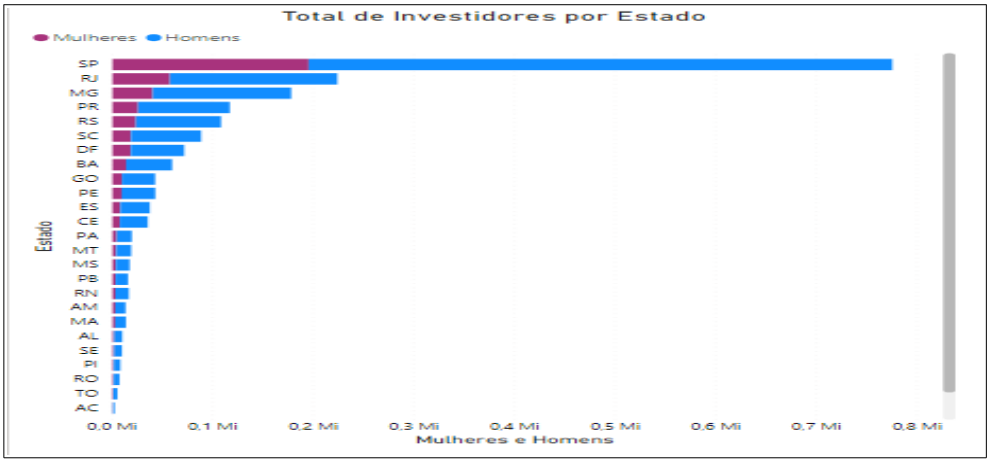


Figura 46 – Gráfico de investidores pessoa física por estado brasileiro

Mais da metade dos investidores de renda variável pessoas físicas, estão com idade entre 26 e 45 anos. Cada vez mais cedo, os jovens buscam por retornos financeiros, alguns iludidos pelas propagandas de corretoras como “do zero ao milhão”. Mas é bom todos saberem dos enormes riscos que a renda variável possuí.

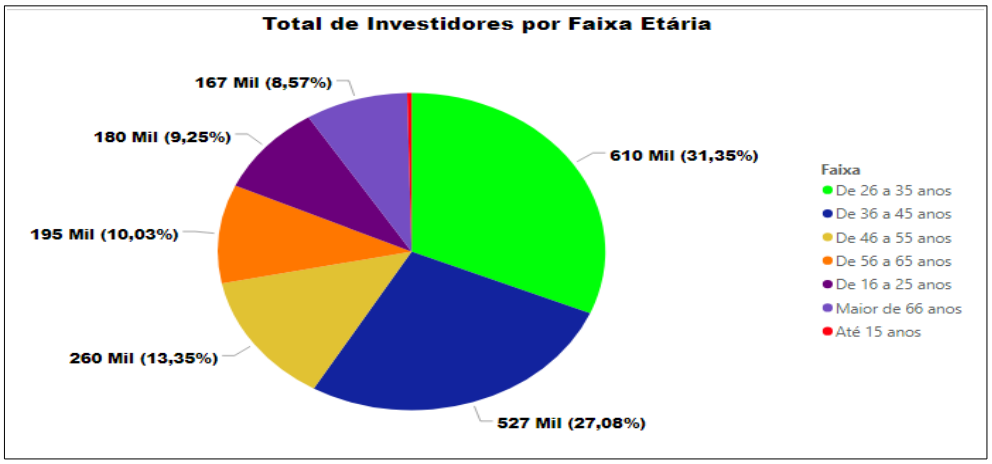


Figura 47 – Gráfico de investidores pessoa física por estado brasileiro

Médias Móveis das Empresas do Setor de Construção – Rastreando a tendência dos preços.

Figura 48 – Gráfico das médias móveis empresa Gafisa

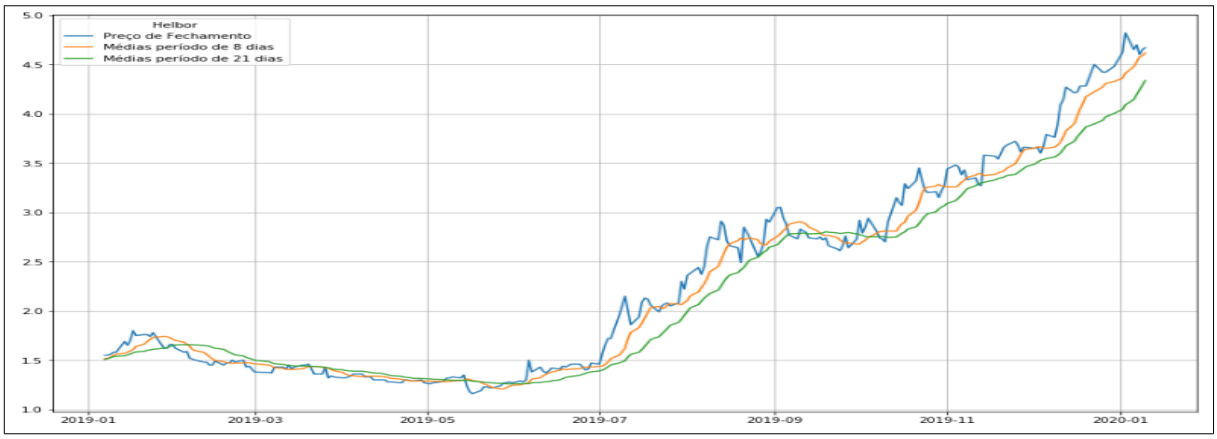


Figura 49 – Gráfico das médias móveis empresa Helbor

Figura 50 – Gráfico das médias móveis empresa MRV

É possível verificar que não é porque são do mesmo setor, que fazem o mesmo movimento.

Lembrando:

- Sinal de Compra: quando os preços cruzam para cima da Média Móvel.
- Sinal de Venda: quando os preços cruzam para baixo da Média Móvel.

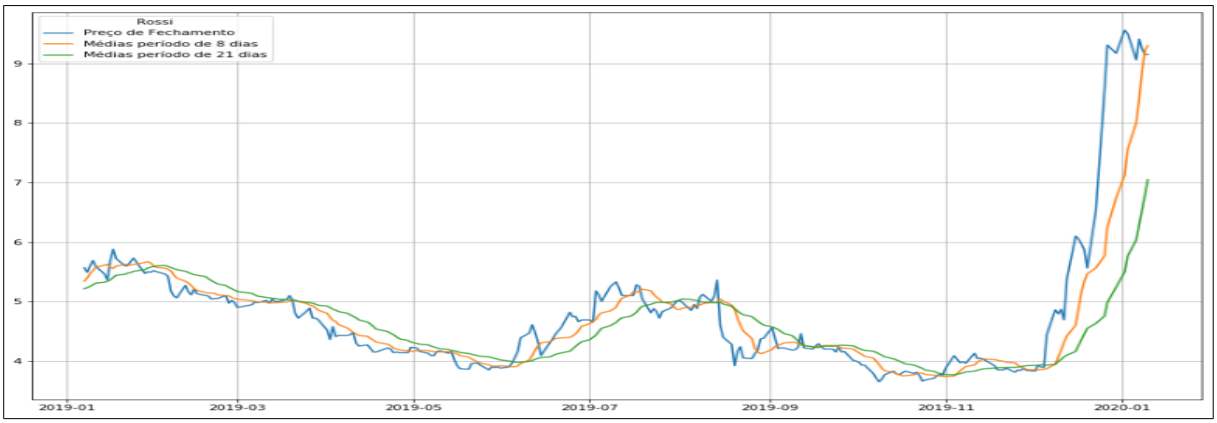


Figura 51 – Gráfico das médias móveis empresa Rossi

Figura 52 – Gráfico das médias móveis empresa Trisul

Vimos também neste trabalho, a aplicação do modelo de Regressão Linear. Percebemos que é possível utilizar essa técnica para tentar prever os valores dos preços de fechamento, dentro de um determinado período, com um erro relativamente baixo. Este erro foi calculado através da técnica de Root Mean Square Error (RMSE).

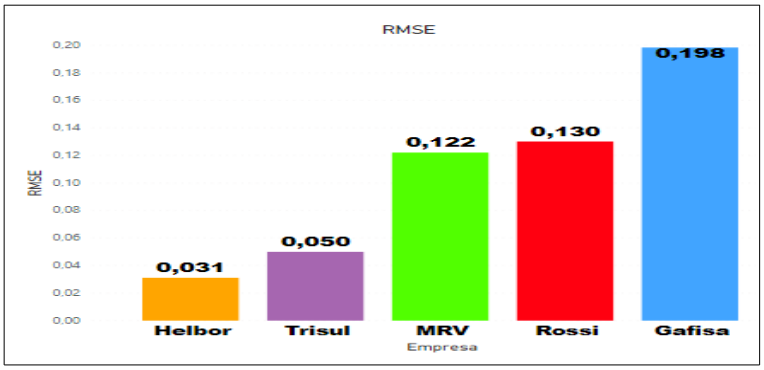


Figura 53 – Gráfico RMSE das empresas do setor de construção

A regressão linear, me pareceu uma boa forma de previsão de preços de fechamento, chegando muito próximo do preço real. Também vale ressaltar o rápido processamento desta técnica.

E por último, implementamos o modelo LSTM, onde julgo ser um modelo mais difícil de vir a ser utilizado para calcular o preço de fechamento, uma vez que é extremante lento o seu processamento para treinar. No entanto, deve ser ressaltado que os resultados demonstraram uma estimativa confiável do preço de fechamento.

Figura 54 – Gráfico modelo de previsão de preço de fechamento empresa Gafisa

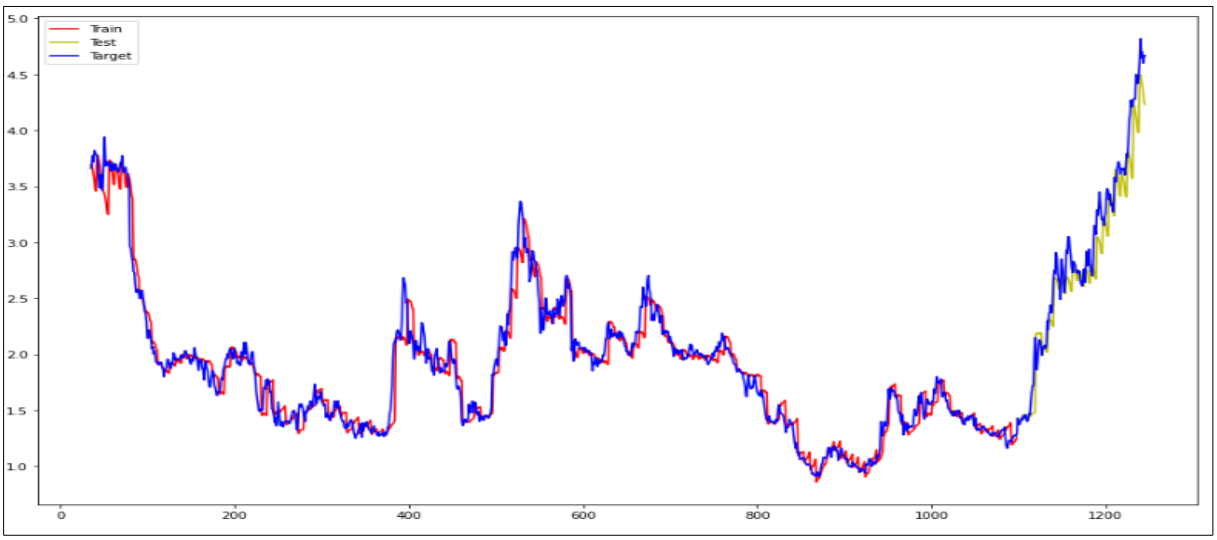


Figura 55 – Gráfico modelo de previsão de preço de fechamento empresa Helbor

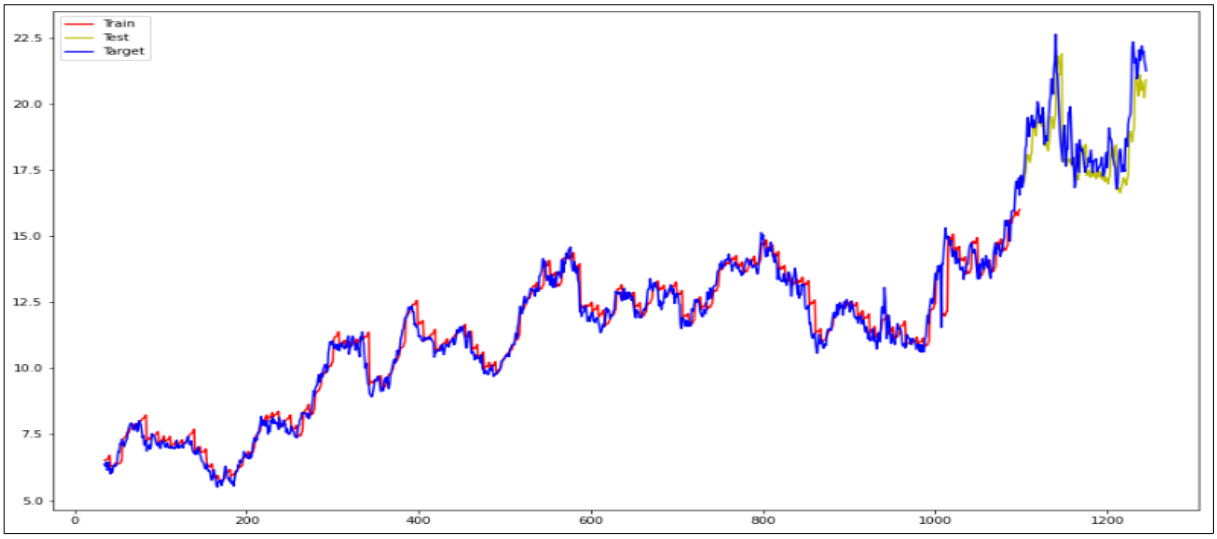


Figura 56 – Gráfico modelo de previsão de preço de fechamento empresa MRV

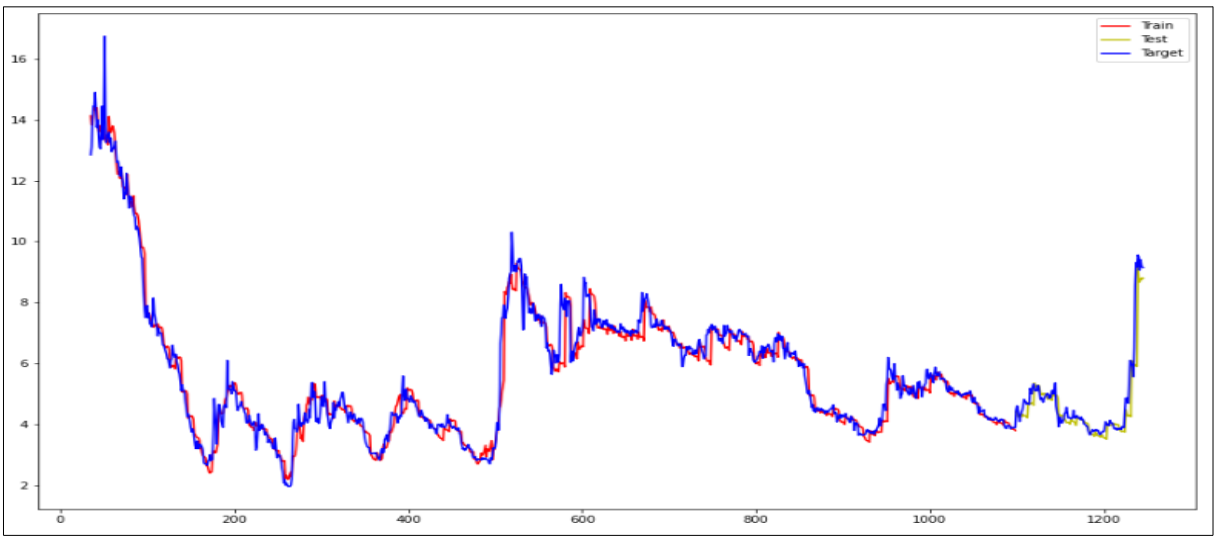


Figura 57 – Gráfico modelo de previsão de preço de fechamento empresa Rossi

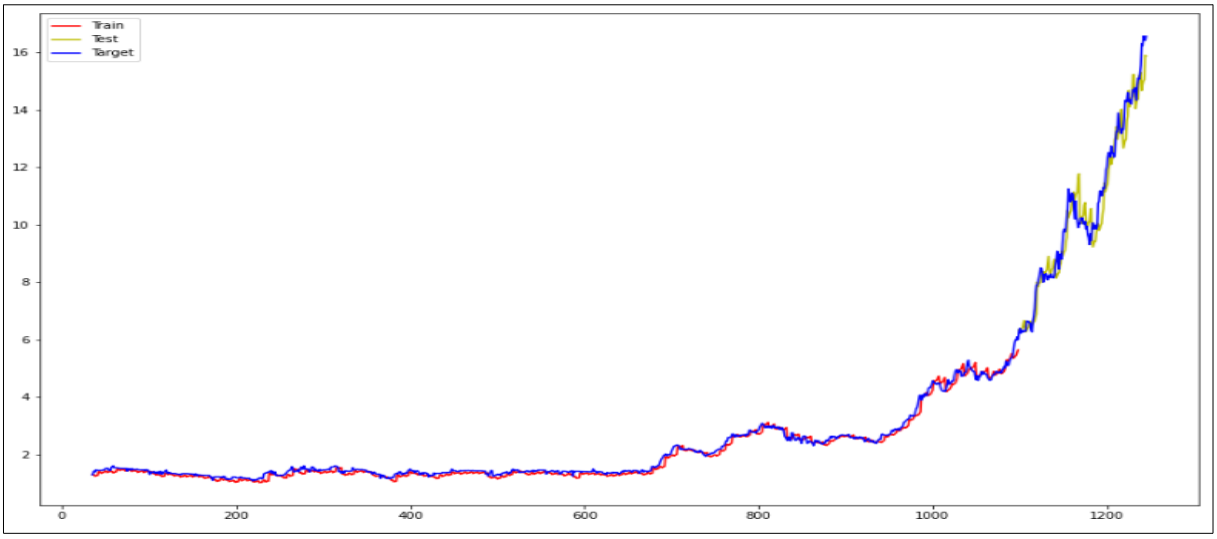


Figura 58 – Gráfico modelo de previsão de preço de fechamento empresa Trisul

Independente das técnicas utilizadas, os resultados demonstram que é possível prever uma estimativa confiável do preço futuro de ações. Porém a imprevisibilidade natural da bolsa e sua vulnerabilidade a eventos externos fazem com que ela aja de forma extremamente difícil para uma inteligência artificial antecipar, tornando assim nosso modelo frágil. Vide o ocorrido com o caso do corona vírus, derrubou qualquer previsão de mercado e tendência, devido a um pânico geral ocasionado no mercado. Os investidores pessoa física na sua grande maioria querem apenas preservar o valor que possuí num lugar seguro. Os especuladores, por sua vez, alugam papéis e vendem a descoberto, esperando justamente o pânico para ganharem muito dinheiro.

Devido a este fator, percebi que além das técnicas aplicadas neste trabalho, é de grande importância a análise de sentimento, notícias, bem como fazer um modelo de classificação dos possíveis problemas que podem ocorrer neste mundo globalizado, atribuindo se pesos como uma medida estatística, a fim de se prever qual o risco/impacto daquela notícia. As vezes a notícia pode ser ruim para uma determinada ação de empresa, no entanto, ótima para outra empresa. Então, a partir de captação de notícias na internet mundial e não somente do Brasil. Este ponto não fez parte deste trabalho, não sendo o objetivo inicial. Devido a isto, não foi implementado neste trabalho. Para contemplar isso, requer de um estudo mais aprofundado sobre o tema, além de técnicas de ciência de dados. Fica aqui o meu desafio de futuro.

Ter uma base histórica com todos os preços das bolsas mundiais, e seu histórico de crise, pode ajudar muito na tomada de decisão e ação. Robôs, devem ser construídos e baseados em modelos matemáticos, configurados para rápida tomada de decisão. Operações iniciadas e encerradas diretamente pelos robôs, pode ser um diferencial, pois o psicológico do ser humana ainda atrapalha muitos investidores. O “eu acho”, acredito eu que está com os dias contados.

Esta crise do corona vírus ocorreu quase no final do meu trabalho, e que de uma certa forma me deu um sentimento de decepção do que eu até aqui construí. No entanto, já num segundo pensamento, me dá a certeza que cada vez mais precisamos de processos inteligentes e robotizadas para a tomada de decisão.

7. Links

Este trabalho esta disponibilizado no caminho abaixo:

https://github.com/RCoelho74/TCC-PUC-MG

Não consegui anexar o vídeo pelo libre office, mas esta no repositório do trabalho.

Links interessantes que complementam este trabalho.

http://colah.github.io/posts/2015-08-Understanding-LSTMs/
http://deeplearningbook.com.br/arquitetura-de-redes-neurais-long-short-term-memory/
http://deeplearningbook.com.br/a-arquitetura-das-redes-neurais/
https://medium.com/@kangeugine/long-short-term-memory-lstm-concept-cb3283934359
http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/
https://medium.com/turing-talks/turing-talks-27-modelos-de-predi%C3%A7%C3%A3o-lstm-df85d87ad210