

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

ROSANA DE PAULA COUTINHO BARROS

**UTILIZANDO MACHINE LEARNING PARA PREVER O PREÇO DE UM ATIVO DE
RENTA VARIÁVEL NA BOLSA DE VALORES**

Belo Horizonte

2022

ROSANA DE PAULA COUTINHO BARROS

**UTILIZANDO MACHINE LEARNING PARA PREVER O PREÇO DE UM ATIVO DE
RENTA VARIÁVEL 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

2022

AGRADECIMENTOS

Agradeço primeiramente a Deus que nos criou e foi criativo nesta tarefa. Ele me deu coragem para questionar realidades e propor sempre um novo mundo de possibilidades. Me deu a sabedoria para fazer escolhas e saúde para buscar a paz.

Agradeço aos meus amados pais Lucivan e Selma, que me deram amor em forma de carinho e apoio, que me deram a oportunidade ao investir no meu conhecimento desde sempre e que me ajudaram a crescer pessoal, profissional e espiritualmente.

Agradeço aos meus irmãos Juliana e Júnior, pelo companheirismo e pela torcida sincera.

Agradeço aos meus amigos, pelo apoio, pela força e pelos momentos descontraídos.

SUMÁRIO

1. Introdução	5
1.1. Contextualização	5
1.2. O problema proposto	5
1.3. Objetivo	6
2. Coleta de Dados	7
3. Pré-processamento e tratamento dos dados	9
3.1 Verificar tipos de dados	10
3.2 Criar novos campos	11
3.3 Empurrar para frente os valores target	12
3.4 Verificar e excluir valores nulos	13
3.5 Reindexar o Dataframe	14
4. Análise e Exploração dos Dados	15
5. Escolha das melhores features e normalização	17
5. 1 Definir dados de treinamento e teste	17
5. 2 Selecionar as melhores features	18
5. 3 Criação dos datasets de treino e teste	19
5. 3 Normalização dos dados	19
6. Modelos de machine learning	20
6.1 Validação cruzada	20
6.2 Regressão Linear Múltipla	21
6.3 Rede Neural Multicamada	24
6.4 Rede Neural LSTM	27
7. Resultados	29
7.1 Métrica de avaliação do modelo	31
8. Considerações finais	34
9. Links	35
REFERÊNCIAS	36
APÊNDICE	37

1. Introdução

1.1. Contextualização

Com o aumento da competitividade da indústria de distribuição de investimentos, o setor de tecnologia da informação provê, a cada dia, soluções melhores e mais acessíveis em termos de tecnologias acessíveis para previsão a partir de séries de dados, um novo leque de possibilidades para o mundo de finanças foi aberto.

Com o objetivo de disponibilizar uma ferramenta para dar suporte a tomada de decisão na hora de investir em um ativo de renda variável na bolsa de valores, foi desenvolvido um algoritmo para esta finalidade, utilizando a ferramenta do Google, Colab e a linguagem de programação Python, devido a sua larga usabilidade e a disponibilidade de pacotes que permitem a manipulação e tratamento de dados.

1.2. O problema proposto

O rápido crescimento econômico dos últimos anos deu luz ao desenvolvimento do mercado de ações e fez deste um pilar fundamental para a economia mundial. O moderno sistema econômico tem como um dos seus pilares um mercado de ações que ofereça informações sobre o custo de capital atual de cada empresa, para determinar a magnitude e estrutura das companhias. No entanto, tomar a melhor decisão na hora de comprar ou vender a ação de uma empresa requer profundo conhecimento do mercado financeiro e da economia de um modo geral.

Para facilitar o entendimento do problema, utilizamos a técnica do 5Ws, que consiste em responder às seguintes perguntas:

Why? Rentabilizar o investimento financeiro no mercado de ações.

Who? Os dados analisados são de conhecimento público, pertencentes ao mercado financeiro e estão disponíveis na internet a qualquer leitor ou empresa.

What? Analisar dados históricos dos preços de uma ação para construir um algoritmo capaz de fazer previsões de preços. Assim dar conhecimento para tomada de decisão na compra de ações.

Where? Todas as informações de insumo para este trabalho são encontradas na internet, mais especificamente, será feita uma análise para o comportamento do papel (ação) de uma empresa nacional, o Banco do Brasil.

When? Como se trata de uma empresa de mais de 200 anos no mercado de ações, o período utilizado nesta análise consta de 2009 até 2021, 12 anos.

1.3. Objetivo

O objetivo do presente trabalho é construir um algoritmo com um baixo custo computacional e operacional, que possa prever o preço de fechamento de qualquer ação da qual se possua o histórico passado.

Testando a forma de eficiência empiricamente requer um modelo específico que pode descrever o processo de formação de preços quando estes representam toda a informação disponível. Neste cenário, o trabalho se propôs a utilizar-se de informações passadas e públicas para desenvolver um modelo capaz de prever o movimento dos preços de ações.

2. Coleta de Dados

Atualmente existem muitas plataformas conhecidas e bastante populares que disponibilizam a série histórica de qualquer ativo de renda variável do mercado financeiro, como Bloomberg e Yahoo! Finanças. Utilizaremos a plataforma Quantum Axis, uma solução digital que oferece ferramentas de análise do mercado financeiro. A base de dados de ativos da plataforma é atualizada com frequência e utiliza fontes oficiais do mercado, as quais também passam por um meticuloso processo de tratamento até serem aprovadas como material confiável.

Ao acessar a plataforma, em DADOS > SERIES, basta informar o código do ativo - trabalharemos com a ação **BBAS3**, do Banco do Brasil - selecionar as informações desejadas e o intervalo de tempo para gerar um arquivo .CSV com todas as informações necessárias.

Quantum | Axis

BRASIL ON NM - BBAS3

Adriano

Última atualização: 17/01/2022

INFORMES PESQUISA DADOS EMPRESAS PORTFÓLIO INDÚSTRIA UTILIDADES

☐ MEDIDAS SELECIONADAS (7)

Medida	Renomear	Editar	Excluir
<input type="checkbox"/> preco_abertura			
<input type="checkbox"/> preco_maximo			
<input type="checkbox"/> preco_minimo			
<input type="checkbox"/> preco_medio			
<input type="checkbox"/> preco_fechamento			
<input type="checkbox"/> qtd_negocios			
<input type="checkbox"/> volume_negocios			

PERÍODO ☐ Fechamento de mês Data base 17/01/2022

Período

datas especificas

desde o inicio

no dia

na semana

Data inicial 01/01/2009

Data final 31/12/2021

Periodicidade

☒ Diária

☐ Semanal

☐ Mensal

Ajuste de Dias

Normal

O resultado desta busca é uma tabela histórica simples e objetiva dos dados necessários para esta análise. Assim, com os parâmetros de buscas corretos obtivemos um arquivo com extensão CSV com o histórico de 12 anos da ação BBAS3, contendo os preços e as features necessárias ao processo de análise.

A etapa de pré-processamento e tratamento de dados talvez seja a mais importante a fim de se obter um bom resultado, o pré-processamento nada mais é do que o processo de preparação, organização e estruturação dos nossos dados além de ser o momento ideal para escolhermos quais dados fazem sentido fazerem parte do nosso dataset.

O processo é iniciado com a importação de dados para dentro da ferramenta utilizada, a Google Colab, e obtemos um Dataframe com 3214 linhas, ou seja, 3214 datas com informações da ação BBAS3:

```
#Importando dados do csv
df_bb = pd.read_csv("Series_acao_BBAS3_2009_2021.csv", delimiter=';', encoding='latin-1')
df_bb
```

	data_pregao	sigla_acao	nome_acao	preco_abertura	preco_maximo	preco_minimo	preco_medio	preco_fechamento	qtd_negocios	volume_negocios
0	2009-01-02	BBAS3	BRASIL ON NM	6.449707	6.772378	6.326971	6.569094	6.580084	2869400	42991378
1	2009-01-05	BBAS3	BRASIL ON NM	6.590300	6.927809	6.357306	6.648028	6.702912	5339200	80961624
2	2009-01-06	BBAS3	BRASIL ON NM	6.801190	7.207586	6.578317	6.897987	7.071397	4080100	64215643
3	2009-01-07	BBAS3	BRASIL ON NM	6.981325	7.083241	6.695322	6.862905	6.913475	1831200	28678803
4	2009-01-08	BBAS3	BRASIL ON NM	6.823158	6.967778	6.530648	6.718192	6.755553	6465700	99102752
...
3209	2021-12-23	BBAS3	BRASIL ON NM	29.300000	29.480000	29.160000	29.260000	29.220000	7337800	214724088
3210	2021-12-27	BBAS3	BRASIL ON NM	29.250000	29.490000	29.200000	29.300000	29.280000	5746200	168380059
3211	2021-12-28	BBAS3	BRASIL ON NM	29.340000	29.390000	29.010000	29.110000	29.110000	8640700	251594596
3212	2021-12-29	BBAS3	BRASIL ON NM	29.180000	29.310000	28.940000	29.060000	28.960000	7666400	222837856
3213	2021-12-30	BBAS3	BRASIL ON NM	29.130000	29.190000	28.770000	28.920000	28.850000	12582800	363896458

3214 rows x 10 columns

A seguir é apresentado um resumo dos dados obtidos nesse datasets

Nome da coluna	Descrição	Tipo
data_pregao	Data do pregão	object
sigla_acao	Código da ação na Bolsa	object
nome_acao	Nome da ação	object
preco_maximo	Preço máximo da ação na data	Float
preco_minimo	Preço mínimo da ação na data	Float
preco_medio	Preço médio da ação na data	Float
preco_fechamento	Preço final da ação na data	Float
qtd_negocios	Quantidade de negociação na data	Integer

volume_negocio	Volume total de negociação da ação da data	Integer
----------------	--------------------------------------------	---------

3.1 Verificar tipos de dados

Este passo é essencial para o correto funcionamento dos códigos, o próximo tratamento foi na coluna `data_pregao`, que inicialmente veio como um Object, mas que foi transformado em Date para que o modelo entenda como série histórica e funcione corretamente.

```
✓ [107] #verificar os tipos contidos no dataframe
0s df_bb.dtypes
```

```
data_pregao      object
sigla_acao       object
nome_acao        object
preco_abertura   float64
preco_maximo     float64
preco_minimo     float64
preco_medio      float64
preco_fechamento float64
qtd_negocios     int64
volume_negocios  int64
dtype: object
```

```
✓ [108] #Mudar o tipo data
0s df_bb['data_pregao'] = pd.to_datetime(df_bb['data_pregao'], format='%Y-%m-%d')
```

```
✓ [113] df_bb.dtypes
0s
```

```
data_pregao      datetime64[ns]
sigla_acao       object
nome_acao        object
preco_abertura   float64
preco_maximo     float64
preco_minimo     float64
preco_medio      float64
preco_fechamento float64
qtd_negocios     int64
volume_negocios  int64
mm5d            float64
mm21d           float64
dtype: object
```

3.2 Criar novos campos

Para que possamos obter mais algumas métricas relevantes, criaremos mais dois campos para informações de médias móveis.

As médias móveis suavizam os dados de preços para formar um indicador de tendência sequencial. Elas não prevêm a direção dos preços, mas, antes, definem a sua direção atual com um atraso. As médias móveis atrasam porque elas são baseadas em preços passados. Apesar disso, as médias móveis ajudam a suavizar o preço da ação e filtram o ruído.

Abaixo o código em python que calcula a média móvel simples para 5 dias e 21 dias.

```
#criando novos campos de medias móveis
df_bb['mm5d'] = df_bb['preco_fechamento'].rolling(5).mean()
df_bb['mm21d'] = df_bb['preco_fechamento'].rolling(21).mean()
```

3.3 Empurrar para frente os valores target

Utilizaremos o método shift do python para jogar para uma data para frente os valores da coluna preco_fechamento.

Por exemplo, se as features referentes a data de 05/01/2009 indicam que no dia seguinte o preço da ação será de 7.071397, então vamos vincular este preço ao conjunto de features de 05/01/2009, como mostra a imagem a seguir.

```
[17] df_bb.head()
```

	data_pregao	sigla_acao	nome_acao	preco_abertura	preco_maximo	preco_minimo	preco_medio	preco_fechamento
0	2009-01-02	BBAS3	BRASIL ON NM	6.449707	6.772378	6.326971	6.569094	6.580084
1	2009-01-05	BBAS3	BRASIL ON NM	6.590300	6.927809	6.357306	6.648028	6.702912
2	2009-01-06	BBAS3	BRASIL ON NM	6.801190	7.207586	6.578317	6.897987	7.071397
3	2009-01-07	BBAS3	BRASIL ON NM	6.981325	7.083241	6.695322	6.862905	6.913475
4	2009-01-08	BBAS3	BRASIL ON NM	6.823158	6.967778	6.530648	6.718192	6.755553

```
#Empurrando para frente os valores de preco de fechamento
df_bb['preco_fechamento'] = df_bb['preco_fechamento'].shift(-1)
df_bb.head()
```

	data_pregao	sigla_acao	nome_acao	preco_abertura	preco_maximo	preco_minimo	preco_medio	preco_fechamento
0	2009-01-02	BBAS3	BRASIL ON NM	6.449707	6.772378	6.326971	6.569094	6.702912
1	2009-01-05	BBAS3	BRASIL ON NM	6.590300	6.927809	6.357306	6.648028	7.071397
2	2009-01-06	BBAS3	BRASIL ON NM	6.801190	7.207586	6.578317	6.897987	6.913475
3	2009-01-07	BBAS3	BRASIL ON NM	6.981325	7.083241	6.695322	6.862905	6.755553
4	2009-01-08	BBAS3	BRASIL ON NM	6.823158	6.967778	6.530648	6.718192	6.966116

3.4 Verificar e excluir valores nulos

Perceba que o dataset importado não veio com dados nulos inicialmente, estes espaços nulos foram gerados a partir do tratamentos dados ao longo do processamento e tratamento dos dados, por exemplo, ao gerar as médias móveis e o shift na coluna de preço de fechamento. Por tanto é muito importante fazermos esta verificação após o pré processamento dos dados.

```
df_bb.isnull().sum()
```

```
data_pregao      0
sigla_acao       0
nome_acao        0
preco_abertura   0
preco_maximo     0
preco_minimo     0
preco_medio      0
preco_fechamento 1
qtd_negocios     0
volume_negocios  0
mm5d             4
mm21d           20
dtype: int64
```

```
#retirando os dados nulos
df_bb.dropna(inplace=True)
```

```
df_bb.isnull().sum()
```

```
data_pregao      0
sigla_acao       0
nome_acao        0
preco_abertura   0
preco_maximo     0
preco_minimo     0
preco_medio      0
preco_fechamento 0
qtd_negocios     0
volume_negocios  0
mm5d             0
mm21d            0
dtype: int64
```

3.5 Reindexar o Dataframe

Reindexamos o dataset para reorganizar os índices e neste ponto, temos uma tabela organizada, indexada, sem valores nulos e pronta para finalmente ser analisada. Esta nova tabela possui 3193 linhas.

```
df_bb = df_bb.reset_index(drop=True)
df_bb
```

	data_pregao	sigla_acao	nome_acao	preco_abertura	preco_maximo	preco_minimo	preco_medio	preco_fechamento	qtd_negocios	volume_negocios	m2d	m23d
0	2009-01-30	BBAS3	BRASIL ON NM	6.203669	6.372696	6.088626	6.227045	6.036130	2523800	35851049	6.274768	6.430518
1	2009-02-02	BBAS3	BRASIL ON NM	6.107012	6.261674	5.954287	6.095487	6.220373	2733000	37994527	6.244061	6.404615
2	2009-02-03	BBAS3	BRASIL ON NM	6.107012	6.381578	5.984621	6.183192	6.123865	2827500	39897234	6.229146	6.381637
3	2009-02-04	BBAS3	BRASIL ON NM	6.260785	6.461515	6.023623	6.240200	6.316881	4734100	67356685	6.163345	6.336516
4	2009-02-05	BBAS3	BRASIL ON NM	6.133373	6.425987	6.027957	6.235815	6.395842	3308800	47067643	6.185279	6.308107
...
3188	2021-12-22	BBAS3	BRASIL ON NM	29.600000	29.650000	29.160000	29.280000	29.220000	12478200	365442015	29.998000	31.306042
3189	2021-12-23	BBAS3	BRASIL ON NM	29.300000	29.480000	29.160000	29.260000	29.280000	7337800	214724088	29.560000	31.265289
3190	2021-12-27	BBAS3	BRASIL ON NM	29.250000	29.490000	29.200000	29.300000	29.110000	5746200	168380059	29.350000	31.153038
3191	2021-12-28	BBAS3	BRASIL ON NM	29.340000	29.390000	29.010000	29.110000	28.960000	8640700	251594596	29.288000	31.082419
3192	2021-12-29	BBAS3	BRASIL ON NM	29.180000	29.310000	28.940000	29.060000	28.850000	7666400	222837856	29.180000	31.002289

3193 rows × 12 columns

4. Análise e Exploração dos Dados

A análise exploratória dos dados é uma etapa muito importante que se realiza depois do processamento dos dados para técnicas de machine learning e da coleta de dados.

Deve-se colocá-la em prática antes de qualquer tipo de modelagem em si, pois é essencial que o sejam capaz de entender a natureza dos dados sem fazer suposições.

O objetivo da análise exploratória dos dados é utilizar síntese estatística e técnicas de visualização para entender melhor os dados e identificar insights sobre tendências e a qualidade dos dados, bem como para formular hipóteses e fazer suposições nas análises.

Os preços das ações realizam movimentos frenéticos no curtíssimo prazo, mas quando observamos em prazos maiores podemos identificar que eles seguem alguma tendência no decorrer de muitas semanas, meses e anos.

Podemos verificar no gráfico do histórico abaixo um comportamento de crescimento relativamente constante. No entanto verificamos que no período referente ao início da pandemia mundial de 2020, registou-se uma queda brusca nos preços, então a ação parece se estabilizar ao longo do ano seguinte.



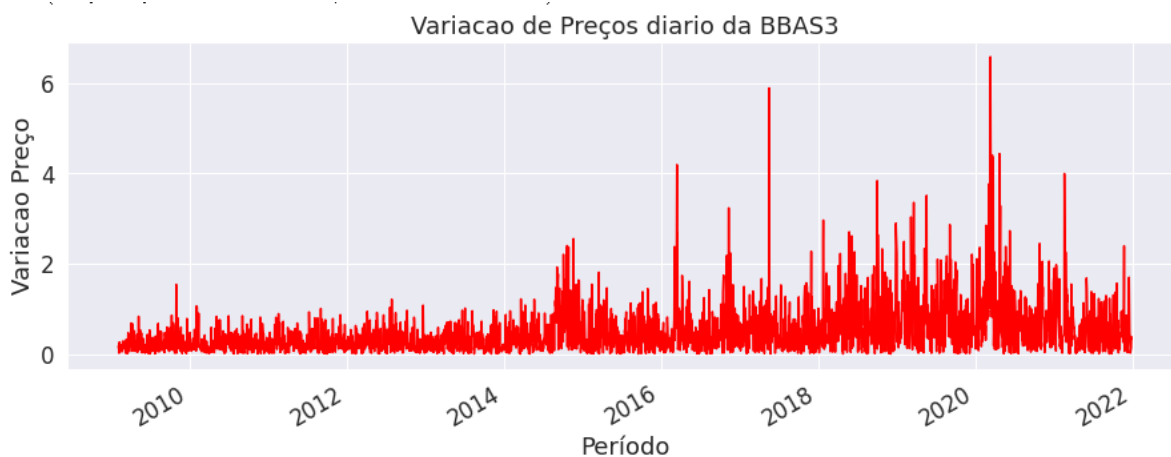
Uma funcionalidade muito útil para este tipo de análise é o método `describe()` que nos dá um apanhado geral das informações contidas no dataset estudado.

Como podemos observar as médias das colunas de `preco_maximo`, `preco_minimo` e `preco_fechamento` são próximos, o que mostra equilíbrio entre seus valores.

```
df_bb.describe()
```

	preco_abertura	preco_maximo	preco_minimo	preco_medio	preco_fechamento	qtd_negocios	volume_negocios	mm5d	mm21d
count	3193.000000	3193.000000	3193.000000	3193.000000	3193.000000	3.193000e+03	3.193000e+03	3193.000000	3193.000000
mean	21.234302	21.634900	20.798414	21.208826	21.211728	9.104226e+06	2.792241e+08	21.190364	21.129381
std	9.914085	10.012318	9.789881	9.897378	9.890505	6.367690e+06	2.319908e+08	9.884825	9.855214
min	5.953678	6.141769	5.761373	5.935908	5.876552	7.398000e+05	1.594301e+07	6.020998	6.138929
25%	13.865079	14.185712	13.526916	13.865674	13.867650	4.973000e+06	1.217496e+08	13.826098	13.868892
50%	16.501579	16.837974	16.108878	16.459292	16.512123	7.519500e+06	1.932484e+08	16.419869	16.370909
75%	28.264359	28.702408	27.809916	28.267384	28.220185	1.127470e+07	3.777128e+08	28.270267	28.331762
max	48.349122	48.525043	47.562247	47.883927	48.000589	1.020361e+08	2.964447e+09	47.588421	46.296431

O gráfico a seguir mostra a variação de preços diários da ação.



5. Escolha das melhores features e normalização

5. 1 Definir dados de treinamento e teste

Precisamos definir as quantidades de dados destinados ao treinamento, teste e a quantidade que será usada para avaliarmos a previsão feita. Esta última será desconhecida pelo modelo. O ideal é que o modelo se aproxime ao máximo destes valores para estas datas de validação.

Quantidade de datas para treino do modelo: 2235 (70% dos dados)

Quantidade de datas para teste: 894 (28% dos dados)

Quantidade de datas para validação: 64 (2% dos dados)



```
qtd_linhas = len(df_bb)

qtd_linhas_treino= round(.70 * qtd_linhas)
qtd_linhas_teste= round(.28 * qtd_linhas)
qtd_linhas_validacao = round(.02 * qtd_linhas)

info = (
    f"linhas treino= 0:{qtd_linhas_treino}"
    f" linhas teste= {qtd_linhas_treino}:{qtd_linhas_treino + qtd_linhas_teste -1}"
    f" linhas validação= {qtd_linhas_treino + qtd_linhas_teste}:{qtd_linhas }"
)

info

'linhas treino= 0:2235 linhas teste= 2235:3128 linhas validação= 3129:3193'
```

5. 2 Selecionar as melhores features

Para selecionar as features mais importantes que farão parte do modelo, será utilizado o método SelectKbest da biblioteca SciKit-Learn, sua função aqui é valorar cada métrica com base em um teste estatístico.

```
features_list = ('preco_abertura','preco_max','preco_minimo','qtd_negocios','volume_negocios','mm5d','mm21d')

k_best_features = SelectKBest(k='all')
k_best_features.fit_transform(features, labels)
k_best_features_scores = k_best_features.scores_
raw_pairs = zip(features_list[1:], k_best_features_scores)
ordered_pairs = list(reversed(sorted(raw_pairs, key=lambda x: x[1])))

k_best_features_final = dict(ordered_pairs[:15])
best_features = k_best_features_final.keys()
print ('')
print ("Melhores features:")
print (k_best_features_final)
```

valores determinados pela funções do Kbest:

volume_negocios: 570.4145829659857

preco_minimo: 540.4077807815858

qtd_negocios: 458.98780954377486

preco_max: 349.598128585858

mm21d: 5.360335926361045

mm5d: 3.536955982786019

5. 3 Criação dos datasets de treino e teste

```
x_train = features[:qtd_linhas_treino]
x_test = features[qtd_linhas_treino:qtd_linhas_treino + qtd_linhas_teste -1]

y_train = labels[:qtd_linhas_treino]
y_test = labels[qtd_linhas_treino:qtd_linhas_treino + qtd_linhas_teste -1]

x_validacao = features.tail(64) #2% das linhas do dataset inteiro
```

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_validacao: últimos registros do dataset, que deve ser usado para validar

as previsões

5. 3 Normalização dos dados

MinMaxScaler do Sklearn é uma técnica de pré-processamento que coloca os dados na mesma escala. Em tradução livre da documentação: Transforma as features para deixá-las na mesma escala em um determinado range.

Este estimador escala e traduz cada feature individualmente de forma que esta estará em um determinado intervalo no conjunto de treino, por exemplo, entre zero e um.

```
scaler = MinMaxScaler()

x_train_scale = scaler.fit_transform(x_train)
x_test_scale = scaler.transform(x_test)

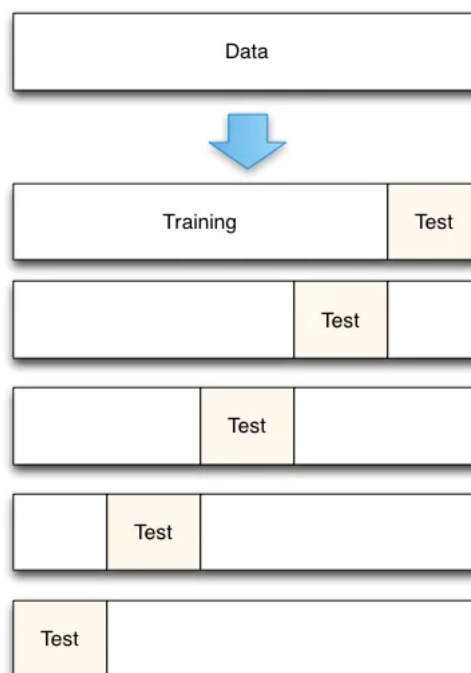
x_validacao_scale = scaler.transform(x_validacao)
```

6. Modelos de machine learning

Considerando que este problema lida com variáveis quantitativas e que faremos análises de séries temporais, utilizaremos modelos de machine learning de aprendizado supervisionados com os métodos de Regressão Linear e Redes Neurais para realizar previsões e ao final comparar essas previsões.

6.1 Validação cruzada

Para melhorar a performance de cada modelo, utilizaremos o método de cross-validation, que consiste em uma técnica muito utilizada para avaliação de desempenho de modelos de machine learning. O cross-validation consiste em particionar os dados em conjuntos(partes), onde um conjunto é utilizado para treino e outro conjunto é utilizado para teste e avaliação do desempenho do modelo.



Na caixinha Data é onde estão todos os nossos dados, depois esses dados são separados em porções de treino e teste, esses dados são embaralhados e divididos em k números de grupos, assim a cada interação temos um conjunto diferente de dados para treino e teste.

Para este processo, no entanto, não é necessário dividir os dados em dados de treinamento e dados de teste. O conjunto inteiro de dados atributos X e o conjunto de dados target Y são enviados para o método `cross_val_score` e este faz automaticamente suas divisões em grupos, de acordo com o parâmetro enviado para este e esse processo é repetido a cada grupo. O resultado do nosso modelo vai ser a média dos valores de cada um desses grupos.

A função `cross_val_score()`, do Sklearn, é responsável por automatizar todo o processo de treinamento do modelo através da técnica de cross-validation.

6.2 Regressão Linear Múltipla

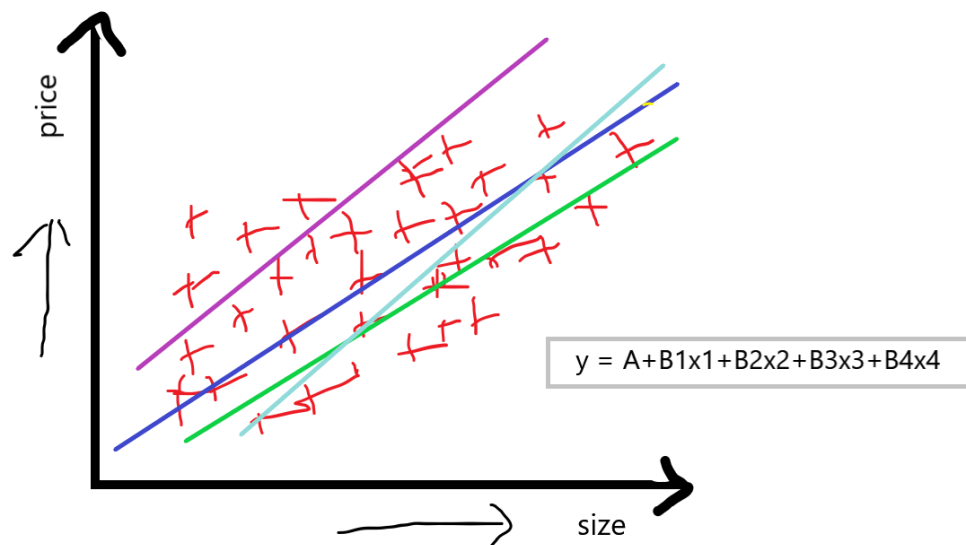
Regressão Linear é uma equação para se estimar um alvo (variável y), dados os valores de outras variáveis (variáveis x). Um algoritmo simples, mas que usado com os parâmetros corretos é capaz de oferecer uma grande capacidade preditiva somente com a relação das suas variáveis.

No caso do nosso problema, estamos lidando com um conjunto de variáveis (preço mínimo, preço médio, preço máximo, volume, entre outros) para prever o preço de fechamento, assim o método regressão adequado a este caso é o de regressão linear múltipla.

Quando falamos sobre regressão linear múltipla, então a equação de regressão linear simples $y = A + Bx$ se transforma em algo como:

$$\text{equação: } y = A + B_1X_1 + B_2X_2 + B_3X_3 + B_4X_4$$

“Se tivermos uma função dependente e várias funções independentes, nós basicamente chamamos isso Regressão linear múltipla.”



Nosso objetivo ao usar regressão linear múltipla é que temos que calcular UMA o que é um cruzamento, e B1 B2 B3 B4 quais são as inclinações ou coeficientes que se referem a esta característica independente, o que basicamente indica que, se aumentarmos o valor de X1 por 1 dirigitir então B1 diz quanto valor afetará o preço da casa, e isso era semelhante em relação a outros B2 B3 B4.

Para utilizarmos o modelo de regressão linear aos nossos dados de treinamento, vamos utilizar o método LinearRegression da biblioteca Scikit-learn, conforme abaixo

```
lr = linear_model.LinearRegression()
lr.fit(x_train_scale, y_train)
pred= lr.predict(x_test_scale)

score_lr = lr.score(x_test_scale,y_test )
print("Score: %.3f%%" % (score_lr * 100.0))
```

Score: 98.218%

A acurácia determinada por esse modelo determinada a partir dos dados de teste foi de 98.218%.

Após o treinamento utilizando a cross-validation (validação cruzada), o valor de score do modelo sobe para 99.585%.

```
num_folds = 4
seed = 7
kfold = KFold(num_folds, shuffle=True, random_state= seed)

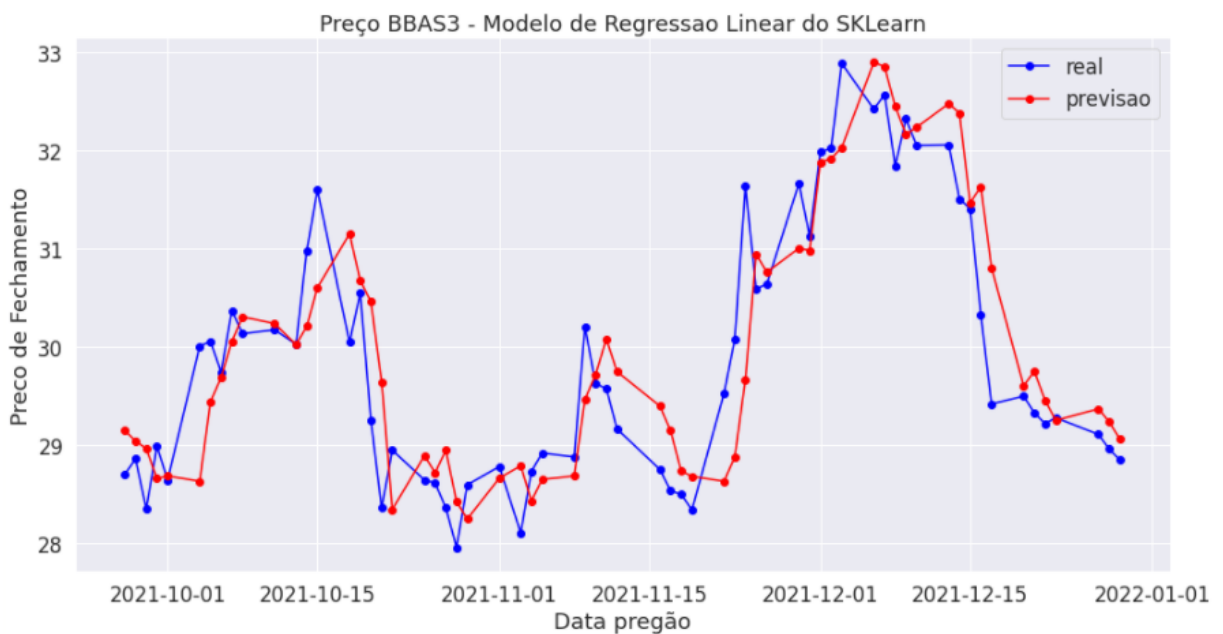
modelo_lr_vc = linear_model.LinearRegression()

#Treinando a maquina com validacao cruzada
score_lr_vc = cross_val_score(modelo_lr_vc, x, y, cv = kfold)

print("Score: %.3f%%" % (score_lr_vc.mean() * 100.0))
```

Score: 99.585%

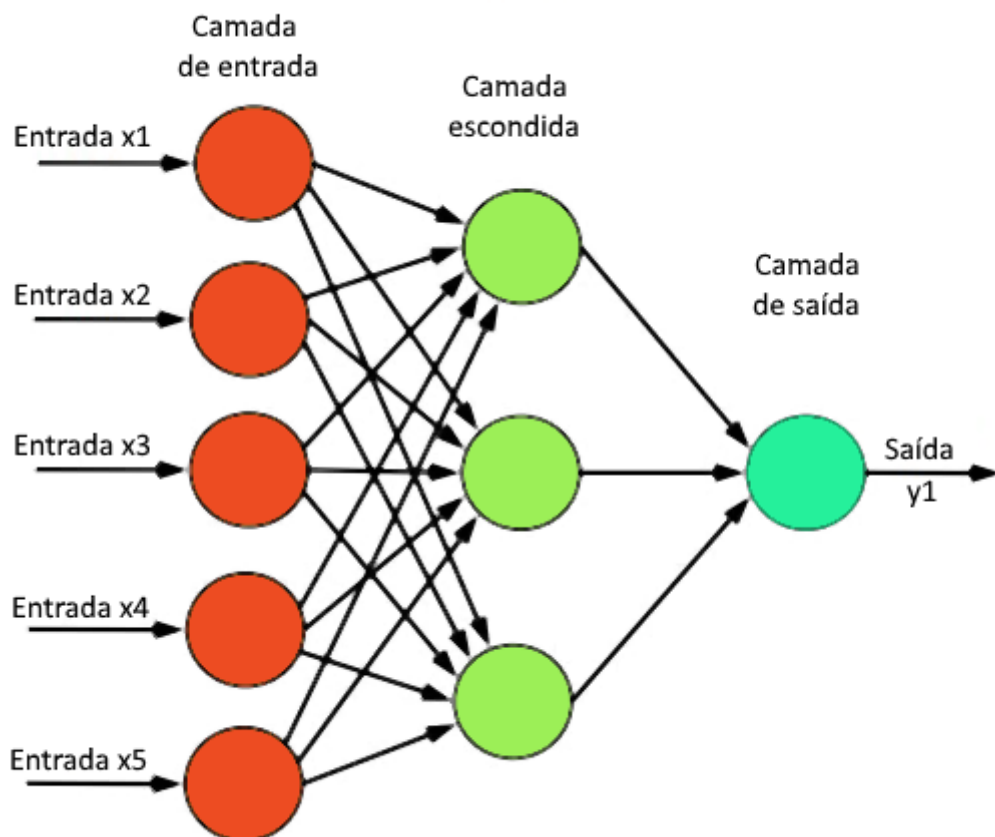
Após executado o processo de predição para este modelo, obtivemos um gráfico com aparência muito satisfatória. A linha que representa a previsão acompanha bem os valores reais dos preços da ação.



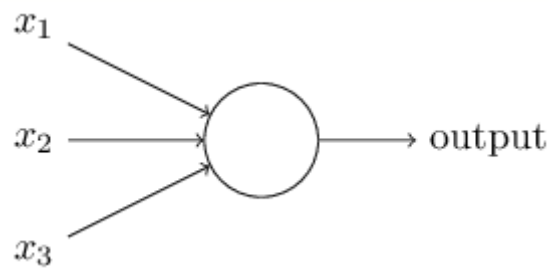
6.3 Rede Neural Multicamada

Uma rede neural aprende através de um processo iterativo de ajustes aplicados aos seus pesos sinápticos e limiares, o qual pode ser expresso na forma de um algoritmo computacional.

As redes neurais multicamadas são arquiteturas onde os neurônios são organizados em duas ou mais camadas de processamento.



A forma mais simples deste tipo de rede neural, consiste de um único neurônio na camada de saída, sendo conhecido como perceptron.



O tipo mais comum de rede neural referido como Multi-Layer Perceptron (MLP) é uma função que mapeia a entrada para a saída. O MLP possui uma única camada de entrada e uma única camada de saída. No meio, pode haver uma ou mais camadas ocultas. A camada de entrada tem o mesmo conjunto de neurônios que a de recursos. Camadas ocultas também podem ter mais de um neurônio. Cada neurônio é uma função linear à qual a função de ativação é aplicada para resolver problemas complexos. A saída de cada camada é dada como entrada para todos os neurônios das camadas seguintes.

MLPRegressor é um estimador disponível como parte do módulo `neural_network` do Sklearn para realizar tarefas de regressão usando um perceptron multicamada.

Na imagem a seguir a máquina preditiva criada, mostra um score de 97.390% após seu treinamento com os dados de teste.

```

rn = MLPRegressor(max_iter=2000)
rn.fit(x_train_scale, y_train)
pred = rn.predict(x_test_scale)
cd = rn.score(x_test_scale, y_test) #acurácia

print("score: %.3f%%" % (cd * 100.0))

```

score: 97.203%

Após o treinamento com validação cruzada o valor de score sobe para 99.586%.

```

num_folds = 4
seed = 7
kfold = KFold(num_folds, shuffle=True, random_state= seed)

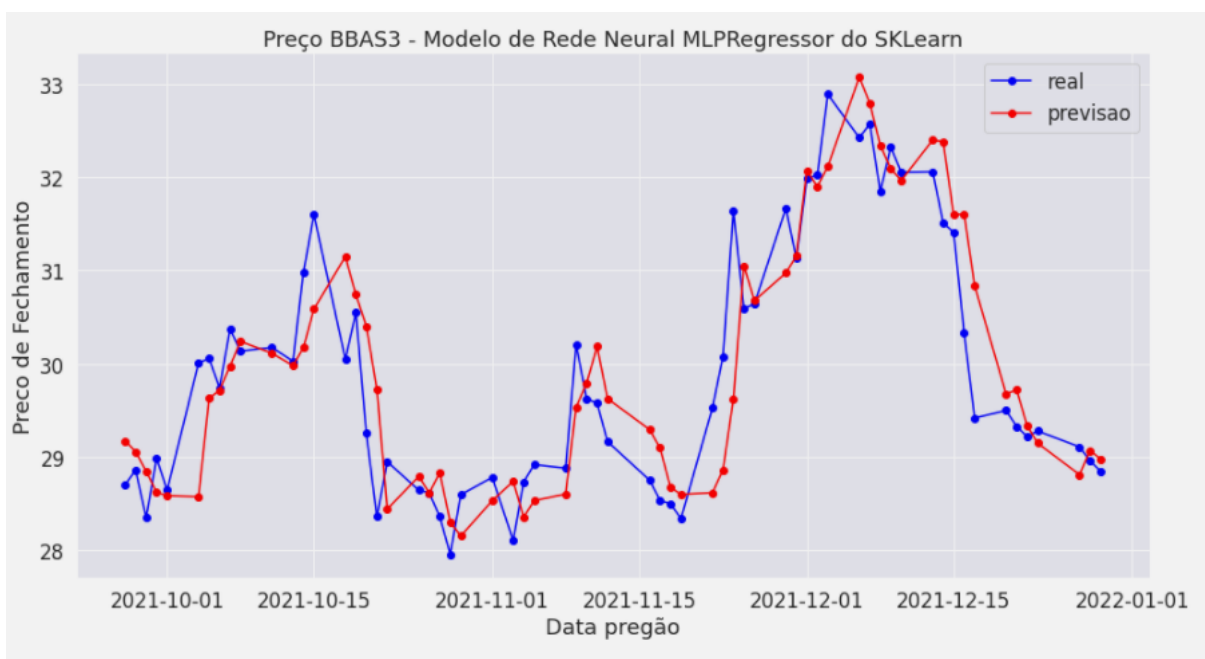
model_rn_vc = MLPRegressor(max_iter=2000)
acuracia = cross_val_score(model_rn_vc, x, y, cv = kfold)

print("score: %.3f%%" % (acuracia.mean() * 100.0))

```

score: 99.586%

Uma vez executado o processo de predição para este modelo, obtivemos um gráfico com aparência muito satisfatória. A linha que representa a previsão acompanha bem os valores reais dos preços da ação.

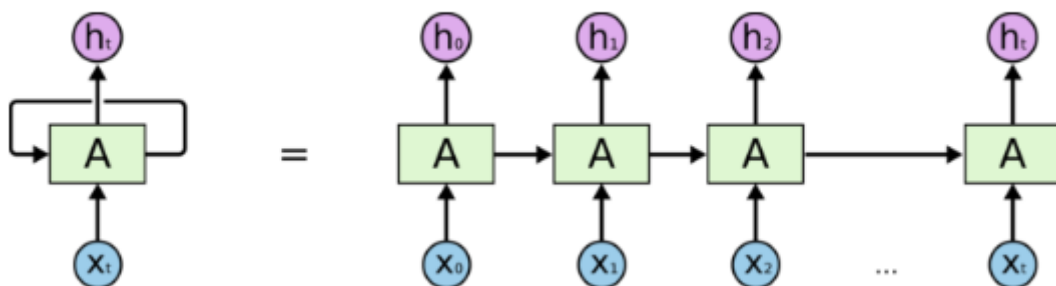


6.4 Rede Neural LSTM

A LSTM, Long Short Term Memory, é uma arquitetura de rede neural recorrente (RNN) que “lembra” valores em intervalos arbitrários. A LSTM é bem adequada para classificar, processar e prever séries temporais.

Redes neurais recorrentes são redes com loops ou repetições, permitindo que as informações “persistam”.

Nas camadas escondidas, o neurônio envia a informação para ele mesmo, ou seja, ao invés de aplicar a função de ativação e passar a informação adiante, o neurônio envia também a informação para ele mesmo, dessa forma armazenando a informação ao longo do tempo.



Utilizaremos o Keras, uma camada de abstração do TensorFlow, para criar nosso modelo.

```
lstm_model = Sequential()
lstm_model.add(LSTM(50, return_sequences=True, input_shape=(x_train_lstm.shape[1],1)))
lstm_model.add(LSTM(50, return_sequences=False))
lstm_model.add(Dense(25))
lstm_model.add(Dense(1))
```

Compilamos o modelo

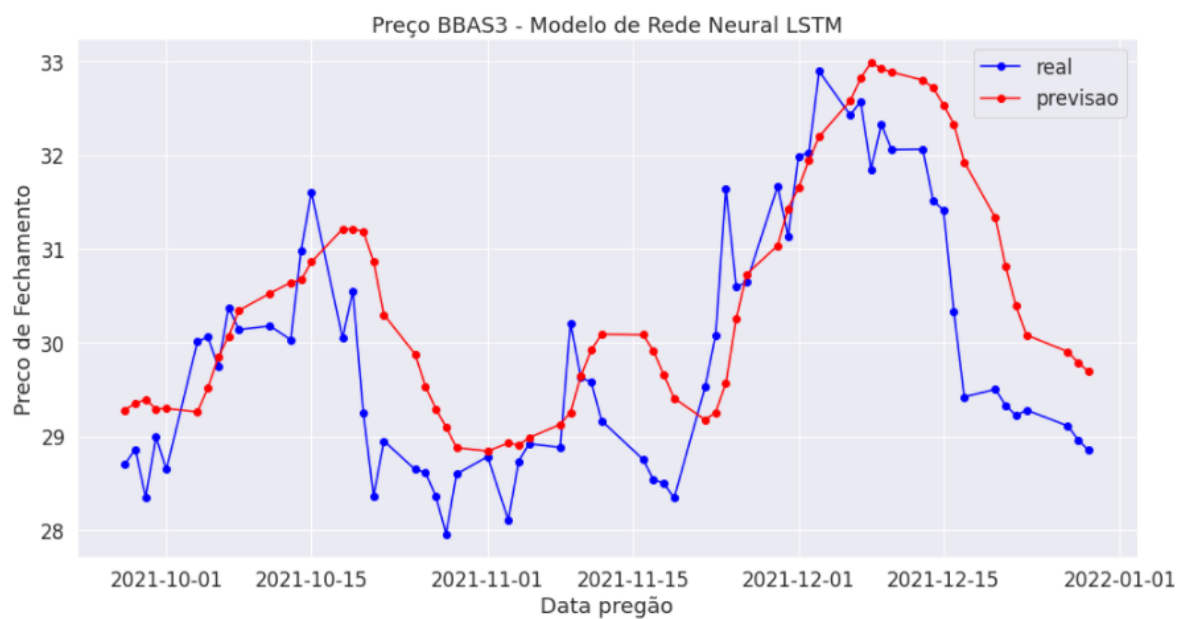
```
lstm_model.compile(optimizer='adam', loss='mean_squared_error')
```

E o treinamos com os dados de treino.

```
lstm_model.fit(x_train_lstm, y_train_lstm, batch_size=1, epochs=1 )

2172/2172 [=====] - 66s 29ms/step - loss: 7.2575e-04
<keras.callbacks.History at 0x7f45fa60d610>
```

Após executado o processo de predição para este modelo, obtivemos um gráfico com aparência muito satisfatória. A linha que representa a previsão acompanha bem os valores reais dos preços da ação.



7. Resultados

Após a constatação de uma melhora da máquina preditiva através do método de validação cruzada. O processo de predição foi realizado para cada modelo instanciado neste trabalho obtendo resultados satisfatórios, uns mais que outros, pois é sabido que a máquina preditiva ideal não existe. O que existe é a melhor opção para um determinado tipo de problema ou determinados dados.

Foi utilizado, neste ponto, o mesmo procedimento de verificação de resultado para cada modelo a fim de medir e comparar os diferentes resultados de previsões obtidas pelos três modelos utilizados.

Assim, foi gerado um Dataframe comparativo para cada resultado, listando os valores de preços **reais**, os valores de preços **previstos** pelos modelos e a diferença absoluta entre esses dois valores que chamamos de **distancia**.

Quadro comparativo entre valores reais e previstos pelo modelo de Regressão Linear

```
print(df_comparativo_lr_vc)
```

	real	previsao	distancia
data_pregao			
2021-09-27	28.704554	29.150533	0.445979
2021-09-28	28.861516	29.041620	0.180104
2021-09-29	28.351388	28.966946	0.615558
2021-09-30	28.989049	28.660971	0.328078
2021-10-01	28.645693	28.687569	0.041876
...
2021-12-22	29.220000	29.458440	0.238440
2021-12-23	29.280000	29.253242	0.026758
2021-12-27	29.110000	29.370990	0.260990
2021-12-28	28.960000	29.237483	0.277483
2021-12-29	28.850000	29.061747	0.211747

```
[64 rows x 3 columns]
```

Quadro comparativo entre valores reais e previstos pelo modelo de Rede Neural Multicamada

```
print(df_comparativo_rn)
```

	real	previsao	distancia
data_pregao			
2021-09-27	28.704554	29.085071	0.380517
2021-09-28	28.861516	29.003855	0.142339
2021-09-29	28.351388	28.854315	0.502927
2021-09-30	28.989049	28.598904	0.390145
2021-10-01	28.645693	28.604866	0.040827
...
2021-12-22	29.220000	29.322157	0.102157
2021-12-23	29.280000	29.194531	0.085469
2021-12-27	29.110000	29.232401	0.122401
2021-12-28	28.960000	29.100996	0.140996
2021-12-29	28.850000	29.001683	0.151683

[64 rows x 3 columns]

Quadro comparativo entre valores reais e previstos pelo modelo de LSTM

```
print(df_comparativo_lstm)
```

	real	previsao	distancia
data_pregao			
2021-09-27	28.704554	29.279980	0.575426
2021-09-28	28.861516	29.352924	0.491408
2021-09-29	28.351388	29.391056	1.039668
2021-09-30	28.989049	29.292456	0.303407
2021-10-01	28.645693	29.295248	0.649555
...
2021-12-22	29.220000	30.387880	1.167880
2021-12-23	29.280000	30.079329	0.799329
2021-12-27	29.110000	29.901403	0.791403
2021-12-28	28.960000	29.784103	0.824103
2021-12-29	28.850000	29.690437	0.840437

[64 rows x 3 columns]

No entanto, esta visualização pode ser considerada confusa e ineficiente já que estamos lidando com uma quantidade grande de informações.

Para uma melhor avaliação utilizaremos aqui a métrica RMSE (root mean squared error), traduzido do inglês, desvio médio quadrático ou erro quadrático médio.

7.1 Métrica de avaliação do modelo

O desvio médio quadrático representa a raiz quadrada do MSE (erro quadrático médio, que é o erro do quadrado médio das previsões do modelo). O RMSE mede a diferença entre os valores previstos pelo modelo e os valores observados. O RMSE é como o "desvio padrão dos erros".

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

O RMSE é uma medida de como os valores previstos se afastam dos valores reais. Quanto maior o erro quadrático médio, maior o afastamento entre os valores reais e os valores previstos. Assim, quanto menor esta medida, melhor o modelo performou.

Assim obtivemos o RMSE para cada modelo

O RMSE obtido a partir dos dados previstos pelo modelo de regressão linear foi o seguinte:

```
rmse_lr = np.sqrt( np.mean(res - new_pred_lr_vc)**2 )
rmse_lr

0.07504799784980021
```

RMSE obtido a partir dos dados previstos pelo modelo de rede neural multicamada foi:

```
rmse_rn = np.sqrt( np.mean(res - new_pred_rn_vc)**2 )
rmse_rn

0.010831627317404169
```

E

E por último, o RMSE obtido a partir dos dados previstos pelo modelo de rede neural LSTM:

```
rmse_lstm = np.sqrt( np.mean(pred_lstm - y_test_lstm)**2 )
rmse_lstm

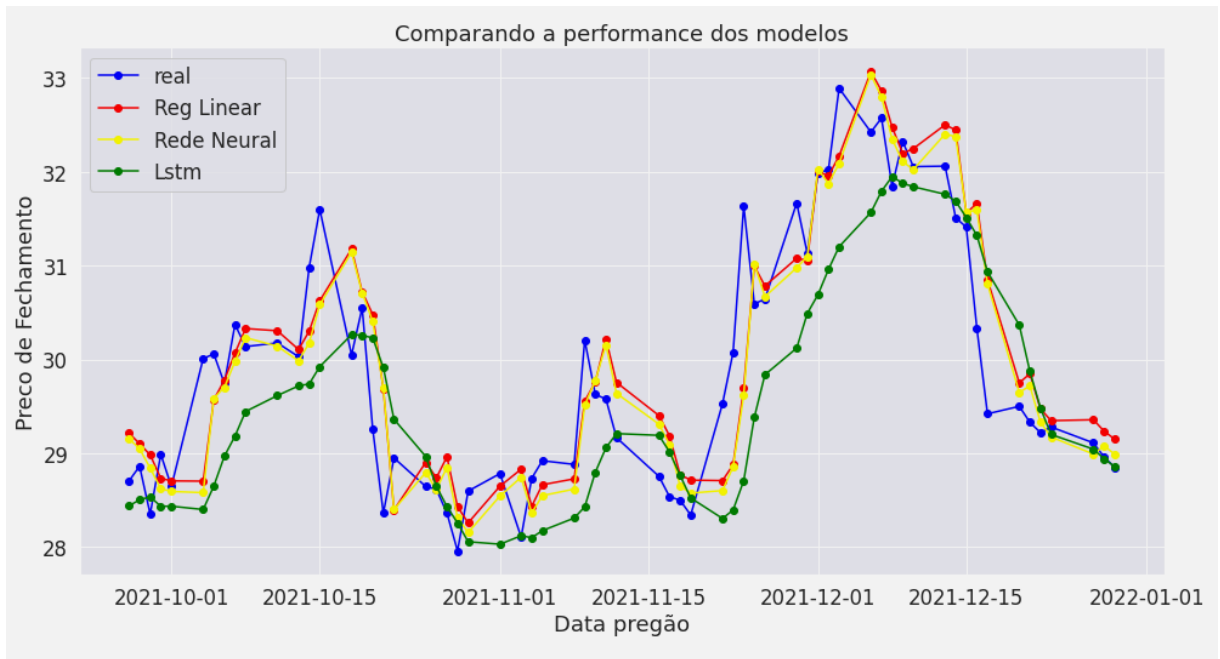
0.6791622593860208
```

Se olharmos cada gráfico gerado por cada modelo separadamente, podemos notar que a curva que representa a linha de previsão para cada modelo acompanha de forma geral a curva de valores reais, em alguns casos, as curvas até se tocam, indicando que o modelo chegou a acertar o preço. No entanto, se compararmos utilizando a métrica do RMSE, conseguimos ver que há uma diferença entre as performances de cada modelo.

Quadro Comparativo de RMSE

Modelo	RMSE
Regressão Linear	0.07504799784980021
Rede Neural Multicamada	0.010831627317404169
Rede Neural LTSM	0.6791622593860208

No gráfico a seguir, fica evidente que todos os modelos obtiveram uma performance bastante satisfatória. Mas se levarmos em consideração a métricas de RMSE, concluímos que o modelo mais assertivo foi o segundo modelo utilizado, o Modelo de Rede Neural Multicamadas, implementado com o Scikit Learn, através da função MLPRegressor()



8. Considerações finais


Este trabalho analisa de forma pragmática três modelos de machine learning para prever o preço de uma ação com base na série histórica de atributos desta ação, adquirida em uma fonte de dados atualizada e confiável.

O processo se deu a partir da metodologia proposta por este curso, que implica na coleta dos dados, pré-processamento dos dados, análise exploratória, criação do modelo, ou dos modelos, e interpretação dos resultados.

Ao final, é utilizado o valor da raiz do erro quadrático médio como forma de avaliação e comparação de cada modelo quanto a sua performance.

No entanto, é importante ressaltar que os algoritmos propostos aqui não são absolutos e perfeitos, uma vez que este trabalho tem propósito acadêmico, sendo necessária uma análise mais aprofundada e pessoal para uma tomada de decisão quanto a compra ou venda de uma ação.

Para finalizar, a seguir é apresentado o quadro “Data Science Workflow Canvas”, proposto por Vasandani, para resumir este trabalho.

 Data Science Workflow Canvas* <small>Start here. The sections below are ordered intentionally to make you state your goals first, followed by steps to achieve those goals. You're allowed to switch orders of these steps!</small>		
Title:		
1 Problem Statement <small>What problem are you trying to solve? What larger issues do the problem address?</small> Disponibilizar informação sobre queda ou valorização de uma ação na Bolsa de Valores	2 Outcomes/Predictions <small>What prediction(s) are you trying to make? Identify applicable predictor (x) and/or target (y) variables.</small> O dado previsto será o preço de fechamento de uma ação (X), com base no histórico de features diversas (Y).	3 Data Acquisition <small>Where are you sourcing your data from? Is there enough data? Can you work with it?</small> A fonte da série histórica da ação utilizada foi baixada integralmente da plataforma online Quantum.
4 Modeling <small>What models are appropriate to use given your outcomes?</small> Foram utilizados três modelos para realizar as previsões: modelo de Regressão Linear, do SKlearn, modelo de Rede Neural Multicamada MLPRegression e modelo de Rede Neural LSTM.	5 Model Evaluation <small>How can you evaluate your model's performance?</small> A métrica utilizada para avaliar a performance dos modelos propostos foi a RMSE, a raiz do erro quadrático médio.	6 Data Preparation <small>What do you need to do to your data in order to run your model and achieve your outcomes?</small> Ajustes como inclusão de novas features de média móvel, exclusão de valores nulos e normalização dos dados, foram necessários. Além de, com ajuda da linguagem python, formatação dos dados em arrays cujo formato fosse suportado pelas máquinas preditivas.

9. Links

link para vídeo no Youtube:

<https://www.youtube.com/watch?v=xwDKRE6aSQw>

(para obter uma melhor visualização no Youtube, altere a qualidade do vídeo para 720p)

link para repositório Git:

<https://github.com/rosanacoutinho/projeto-machine-learning-preco-acao.git>

REFERÊNCIAS

Quantum Axis: API. Disponível em: <<https://www.quantumaxis.com.br/webaxis/>>. Acesso em: 17/01/2022.

Poupardinheiro.com - 9 sites e ferramentas grátis para acompanhar o mercado de ações. Disponível em: <<https://www.poupardinheiro.com.br/investimentos/330-sites-e-ferramentas-gratis-para-acompanhar-mercado-de-acoes>>. Acesso em: 18/01/2022.

Guia Básico de Pré-Processamento de Dados: API. Disponível em: <<https://sigmoidal.ai/guia-basico-de-pre-processamento-de-dados/>>. Acesso em: 19/01/2022.

Neural network models. Disponível em: <https://scikit-learn.org/stable/modules/neural_networks_supervised.html>. Acesso em: 29/01/2022.

Regressão Linear com Sklearn: Conceito e Aplicação. Disponível em: <https://medium.com/@lamartine_sl/regress%C3%A3o-linear-com-sklearn-modelo-de-previs%C3%A3o-de-custos-com-plano-de-sa%C3%BAde-5e963e590f4c>. Acesso em: 29/01/2022.

APÊNDICE

Script Python

```
import pandas as pd
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.model_selection import GridSearchCV
from sklearn.neural_network import MLPRegressor
from sklearn.preprocessing import MinMaxScaler
from sklearn import datasets, linear_model
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
import seaborn as sns
import math
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, LSTM
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict

# %matplotlib inline

#Importando dados do csv
df_bb = pd.read_csv("Series_acao_BBAS3_2009_2021.csv", delimiter=';',
encoding='latin-1')
df_bb

"""## Pré-Processamento dos dados"""

#verificar os tipos contidos no datafame
df_bb.dtypes

#altera o tipo do dado
df_bb['data_pregao']=pd.to_datetime(df_bb['data_pregao'], format='%Y-%m-%d')

df_bb.dtypes

#criando novas colunas para medias móveis
df_bb['mm5d']=df_bb['preco_fechamento'].rolling(5).mean()
#media movel dos ultimos 5 dias
```

```

df_bb['mm21d']=df_bb['preco_fechamento'].rolling(21).mean()
#media movel dos ultimos 21 dias

df_bb.head()

#Empurrando para frente os valores de preco de fechamento
df_bb['preco_fechamento'] = df_bb['preco_fechamento'].shift(-1)
df_bb.head()

df_bb.isnull().sum()

#retirando os dados nulos
df_bb.dropna(inplace=True)

df_bb.isnull().sum()

#reindexando o data frame
df_bb = df_bb.reset_index(drop=True)
df_bb

"""## Análise Exploratória dos Dados"""

df_bb.describe()

#seleciona um periodo menor para zoom na vizualizacao do grafico de
histoprico de precos
df_bb_2017_2021 = df_bb.loc[df_bb['data_pregao'] >= '2017-01-01']
df_bb_2017_2021.set_index('data_pregao', inplace=True)

#Criando um dataframe cujo o index seja a propria data para a plotagem do
grafico de historico
df_grafico = df_bb.copy()
df_grafico.set_index('data_pregao', inplace=True)

#Plotagem da serie historica de todo o periodo
plt.figure(figsize=(15,5))
#df_grafico['preco_maximo'].plot(label='Maximo', color='green')
#df_grafico['preco_minimo'].plot(label='Minimo', color='red')
df_grafico['preco_fechamento'].plot(label='Fechamento', color='blue')
plt.ylabel('Preço')
plt.xlabel('Período')
plt.title('Histórico de Preços da BBAS3')
plt.legend();

#Plotagem da serie historica a partir de 2017
plt.figure(figsize=(15,5))
#df_bb_2017_2021['preco_maximo'].plot(label='Maximo', color='green')
#df_bb_2017_2021['preco_minimo'].plot(label='Minimo', color='red')
df_bb_2017_2021['preco_fechamento'].plot(label='Fechamento', color='blue')
plt.ylabel('Preço')
plt.xlabel('Período')

```

```

plt.title('Histórico de Preços da BBAS3')
plt.legend();

#Plotagem da variacao de preco
variacao_bbas3 = abs(df_grafico['preco_abertura'] -
df_grafico['preco_fechamento'])
plt.figure(figsize=(15,5))
variacao_bbas3.plot(label='Variacao de Preço', color='red')
plt.ylabel('Variacao Preço')
plt.xlabel('Período')
plt.title('Variacao de Preços diario da BBAS3')

#Retorna a data com a maior variacao entre o preco de abertura e fechamento
maxima_variacao_bbas3 = variacao_bbas3.values.argmax()
maximo_valor_bbas3 = variacao_bbas3.values[maxima_variacao_bbas3,]
variacao_bbas3[variacao_bbas3.values==maximo_valor_bbas3]

#Plotagem do volume negociado no periodo
plt.figure(figsize=(15,5))
df_grafico['volume_negocios'].plot(label='Volume', color='blue')
plt.ylabel('Volume')
plt.xlabel('Período')
plt.title('Volume negociado da BBAS3')

#Plota um histograma da distribuicao da BBAS3 utilizando precos de fechamento
do dia
plt.figure(figsize=(15,5))

sns.set_context('notebook', font_scale=1.5, rc={ 'font.size' : 20,
'axes.lablesize':18 } )
sns.rugplot(df_grafico['preco_fechamento'], color='red')
sns.distplot(df_grafico['preco_fechamento'], color='green')
sns.set_style('darkgrid')
plt.title('Distribuicao do preco de fechamento da BBAS3')

#Retorna a data com maior volume de negociacao
maxima_volume_negociacao = df_grafico['volume_negocios'].argmax()
df_grafico.iloc[maxima_volume_negociacao,: ]

"""## Escolha das Melhores Features e Normalização os dados (com
cross-validation)

"""

#Definindo a quantidade de linhas para cada utilizacao

qtd_linhas = len(df_bb)

qtd_linhas_treino= round(.70 * qtd_linhas)
qtd_linhas_teste= round(.28 * qtd_linhas)
qtd_linhas_validacao = round(.02 * qtd_linhas)

```



```

info = (
    f"linhas treino= 0:{qtd_linhas_treino}"
    f" linhas teste= {qtd_linhas_treino}:{qtd_linhas_treino +
qtd_linhas_teste -1}"
    f" linhas validação= {qtd_linhas_treino + qtd_linhas_teste}:{qtd_linhas
}"
)

info

print("quantidade linhas treino {}".format(qtd_linhas_treino))
print("quantidade linhas teste {}".format(qtd_linhas_teste))
print("quantidade linhas validacao {}".format(qtd_linhas_validacao))

#definindo as features e as labels
features = df_bb.drop(['sigla_acao', 'nome_acao', 'data_pregao',
'preco_fechamento'], 1)
labels = df_bb['preco_fechamento']
features

#Escolhendo as melhores features com SelectKBest
features_list =
('preco_abertura', 'preco_max', 'preco_minimo', 'qtd_negocios', 'volume_negocios'
, 'mm5d', 'mm21d')

k_best_features = SelectKBest(k='all')
k_best_features.fit_transform(features, labels)
k_best_features_scores = k_best_features.scores_
raw_pairs = zip(features_list[1:], k_best_features_scores)
ordered_pairs = list(reversed(sorted(raw_pairs, key=lambda x: x[1])))

k_best_features_final = dict(ordered_pairs[:15])
best_features = k_best_features_final.keys()
print ('')
print ("Melhores features:")
print (k_best_features_final)

#criando um df com as features selecionadas
features =
df_bb.loc[:, ['preco_maximo', 'preco_minimo', 'volume_negocios', 'qtd_negocios']]

#Cria os datasets de treino e teste

x_train = features[:qtd_linhas_treino]
x_test = features[qtd_linhas_treino:qtd_linhas_treino + qtd_linhas_teste -1]

y_train = labels[:qtd_linhas_treino]
y_test = labels[qtd_linhas_treino:qtd_linhas_treino + qtd_linhas_teste -1]

x_validacao = features.tail(64) #2% das linhas do dataset inteiro

```

```

#Normalizando os datasets de treino, teste e validação com MinMaxScaler

scaler = MinMaxScaler()

x_train_scale = scaler.fit_transform(x_train)
x_test_scale  = scaler.transform(x_test)
x_validacao_scale = scaler.transform(x_validacao)

"""## Modelo de Regressão Linear do SKlearn (com cross-validation)

"""

#construcao e treinamento usando regressao linear do SKlearn

lr = linear_model.LinearRegression()
lr.fit(x_train_scale, y_train)
pred= lr.predict(x_test_scale)

score_lr = lr.score(x_test_scale,y_test )
print("Score: %.3f%%" % (score_lr * 100.0))

#executando o modelo Linear para obter as predicoes
pred=lr.predict(x_validacao_scale)

#dados reais X dados previstos pelo modelo
data_pregao_full = df_bb['data_pregao']
data_pregao=data_pregao_full.tail(64)
res_full=df_bb['preco_fechamento']
res=res_full.tail(64)
df_comparativo_lr =pd.DataFrame({'data_pregao':data_pregao, 'real':res,
'previsao':pred})
df_comparativo_lr.set_index('data_pregao', inplace=True)
df_comparativo_lr['distancia']= abs( df_comparativo_lr['real'] -
df_comparativo_lr['previsao'] )

print(df_comparativo_lr)

"""**Validacao Cruzada**

"""

#definindo os dados de entreada e saida

x_entradas =
df_bb.loc[:,['preco_maximo','preco_minimo','volume_negocios','qtd_negocios']]
y_target = df_bb.loc[:,['preco_fechamento']]
x = x_entradas.to_numpy()
y = y_target.to_numpy()

```

```

#construcao da maquina preditiva

num_folds = 4
seed = 7
kfold = KFold(num_folds, shuffle=True, random_state= seed)

modelo_lr_vc = linear_model.LinearRegression()

#Treinando a maquina com validacao cruzada
score_lr_vc = cross_val_score(modelo_lr_vc, x, y, cv = kfold)

print("Score: %.3f%%" % (score_lr_vc.mean() * 100.0))

#obtendo as predicoes
new_pred_lr_vc = cross_val_predict(modelo_lr_vc, x, y, cv=kfold)
new_pred_lr_vc= new_pred_lr_vc[-64:]
new_pred_lr_vc = np.reshape(new_pred_lr_vc, (64,))

#dados reais X dados previstos pelo modelo
data_pregao_full = df_bb['data_pregao']
data_pregao=data_pregao_full.tail(64)
res_full=df_bb['preco_fechamento']
res =res_full.tail(64)
df_comparativo_lr_vc =pd.DataFrame({'data_pregao':data_pregao, 'real':res,
'previsao':new_pred_lr_vc})
df_comparativo_lr_vc.set_index('data_pregao', inplace=True)
df_comparativo_lr_vc['distancia']= abs(df_comparativo_lr_vc['real'] -
df_comparativo_lr_vc['previsao'])

print(df_comparativo_lr_vc)

#root mean squared error (RMSE) - SEM validacao cruzada
rmse_lr = np.sqrt( np.mean(res - pred)**2 )
rmse_lr

#root mean squared error (RMSE) - COM validacao cruzada
rmse_lr = np.sqrt( np.mean(res - new_pred_lr_vc)**2 )
rmse_lr

plt.figure(figsize=(16,8))
plt.title('Preço BBAS3 - Modelo de Regressao Linear do SKLearn')
plt.plot(df_comparativo_lr_vc['real'], label="real", color='blue',
marker='o')
plt.plot(df_comparativo_lr_vc['previsao'], label="previsao", color='red',
marker='o')
plt.xlabel('Data pregão')
plt.ylabel('Preco de Fechamento')
leg = plt.legend()

"""## Modelo de Rede Neural MLPRegressor do SKLearn (com cross-validation)

```

Multi-layer Perceptron (MLP) is a supervised learning algorithm
 """

```
#rede neural
rn = MLPRegressor(max_iter=2000)
rn.fit(x_train_scale, y_train)
pred = rn.predict(x_test_scale)
cd = rn.score(x_test_scale, y_test) #acuracia

print("score: %.3f%%" % (cd * 100.0))

pred_rn = rn.predict(x_validacao_scale)

#Dataframe comparativo com os dados reais X dados previstos pelo modelo
```

```
data_pregao_full = df_bb['data_pregao']
data_pregao=data_pregao_full.tail(64)
res_full=df_bb['preco_fechamento']
res =res_full.tail(64)
df_comparativo =pd.DataFrame({'data_pregao':data_pregao, 'real':res,
'previsao':pred_rn})
df_comparativo.set_index('data_pregao', inplace=True)
df_comparativo['distancia']= abs(df_comparativo['real'] -
df_comparativo['previsao'] )

print(df_comparativo)
```

*****Validacao Cruzada**

"""

#definindo os dados de entreada e saida

```
x_entradas =
df_bb.loc[:,['preco_maximo','preco_minimo','volume_negocios','qtd_negocios']]
y_target = df_bb.loc[:,['preco_fechamento']]
```

```
scaler = MinMaxScaler()
```

```
x = scaler.fit_transform(x_entradas)
y = y_target.to_numpy()
```

```
y = np.reshape(y, (3193,))
```

```
# construcao da maquina preditiva
```

```
num_folds = 4
```

```
seed = 7
```

```
kfold = KFold(num_folds, shuffle=True, random_state= seed)
```

```

model_rn_vc = MLPRegressor(max_iter=2000)
acuracia = cross_val_score(model_rn_vc, x, y, cv = kfold)

print("score: %.3f%%" % (acuracia.mean() * 100.0))

#obtendo as predicoes
new_pred_rn_vc = cross_val_predict(model_rn_vc, x, y, cv=kfold)
new_pred_rn_vc= new_pred_rn_vc[-64:]

data_pregao_full = df_bb['data_pregao']
data_pregao=data_pregao_full.tail(64)
res_full=df_bb['preco_fechamento']
res=res_full.tail(64)
df_comparativo_rn =pd.DataFrame({'data_pregao':data_pregao, 'real':res,
'previsao':new_pred_rn_vc})
df_comparativo_rn.set_index('data_pregao', inplace=True)
df_comparativo_rn['distancia']= abs(df_comparativo_rn['real'] -
df_comparativo_rn['previsao'] )

print(df_comparativo_rn)

#root mean squared error (RMSE) - SEM validacao cruzada

rmse_rn = np.sqrt( np.mean(res - pred_rn)**2 )
rmse_rn

#root mean squared error (RMSE) - COM validacao cruzada

rmse_rn = np.sqrt( np.mean(res - new_pred_rn_vc)**2 )
rmse_rn

plt.figure(figsize=(16,8))
plt.title('Preço BBAS3 - Modelo de Rede Neural MLPRegressor do SKLearn')
plt.plot(df_comparativo_rn['real'], label="real", color='blue', marker='o')
plt.plot(df_comparativo_rn['previsao'], label="previsao", color='red',
marker='o')
plt.xlabel('Data pregão')
plt.ylabel('Preco de Fechamento')
leg = plt.legend()

"""## Modelo de Rede Neural LSTM (Long-Short Term Memory )

"""

df_lstm = df_bb.copy()
df_lstm.set_index('data_pregao', inplace=True)
df_lstm

```

```

#defininda a quantidade para dados de treinamento
data = df_lstm.filter(['preco_fechamento'])
dataset = data.values
training_data_len = math.ceil( len(dataset) * .7 )
training_data_len

#normalizacao
scaler2 = MinMaxScaler(feature_range=(0,1))
scaled_data= scaler2.fit_transform(dataset)
scaled_data

#criando o dataset de treinamento
train_data_lstm =scaled_data[0:training_data_len, :]

#spliting os dados em x_train e y_train
x_train_lstm =[]
y_train_lstm =[]

for i in range(64, len(train_data_lstm)):
    x_train_lstm.append(train_data_lstm[i-64:i,0])
    y_train_lstm.append(train_data_lstm[i,0])
    if i<=64:
        print(x_train_lstm)
        print(y_train_lstm)
        print()

#converter o x_train e o y_train em um numpy array
x_train_lstm, y_train_lstm = np.array(x_train_lstm), np.array(y_train_lstm)

#reshape os dados
x_train_lstm = np.reshape(x_train_lstm, (x_train_lstm.shape[0],
x_train_lstm.shape[1],1)) #64, 1
x_train_lstm.shape

#criando dataset de teste
test_data = scaled_data[training_data_len - 64: , :]

#spliting os dados em x_test e y_test
x_test_lstm = []
y_test_lstm = dataset[training_data_len: , :]

for i in range(64, len(test_data)):
    x_test_lstm.append(test_data[i-64:i,0])

#converter o x_train e o y_train em um numpy array
x_test_lstm = np.array(x_test_lstm)

#reshape os dados
x_test_lstm = np.reshape(x_test_lstm, (x_test_lstm.shape[0],
x_test_lstm.shape[1],1))

```

```

x_test_lstm.shape

#Criando o Modelo LSTM

lstm_model = Sequential()
lstm_model.add(LSTM(50, return_sequences=True, input_shape
=(x_train_lstm.shape[1],1))) #64, 1
lstm_model.add(LSTM(50, return_sequences=False))
lstm_model.add(Dense(25))
lstm_model.add(Dense(1))

#Compilando o modelo

lstm_model.compile(optimizer='adam', loss='mean_squared_error')

#treinamento do modelo LSTM

lstm_model.fit(x_train_lstm, y_train_lstm, batch_size=1, epochs=1)

#previsoes
pred_lstm = lstm_model.predict(x_test_lstm)
pred_lstm = scaler2.inverse_transform(pred_lstm)

#data frame comparativo do dados reais X dados previstos pelo modelo
data_pregao_full = df_bb['data_pregao']
data_pregao=data_pregao_full.tail(64)
data_df_lstm_full =pd.Series( df_lstm['preco_fechamento'].values )
data_df_lstm =data_df_lstm_full.tail(64)
res_lstm_full= pred_lstm
ress_lstm = res_lstm_full[-64:] #.tail(64)
data_df_lstm_series = pd.Series(data_df_lstm)

df_comparativo_lstm =pd.DataFrame({'data_pregao':data_pregao,
'real':data_df_lstm}) #, 'previsao':data_pregao})
df_comparativo_lstm['previsao'] = ress_lstm
df_comparativo_lstm.set_index('data_pregao', inplace=True)

df_comparativo_lstm['distancia']= abs(df_comparativo_lstm['real'] -
df_comparativo_lstm['previsao'])

print(df_comparativo_lstm)

#root mean squared error (RMSE)

rmse_lstm = np.sqrt( np.mean(pred_lstm - y_test_lstm)**2 )
rmse_lstm

plt.figure(figsize=(16,8))
plt.title('Preço BBAS3 - Modelo de Rede Neural LSTM')
plt.plot(df_comparativo_lstm['real'], label="real", color='blue', marker='o')

```

```

plt.plot(df_comparativo_lstm['previsao'], label="previsao", color='red',
marker='o')
plt.xlabel('Data pregão')
plt.ylabel('Preço de Fechamento')
leg = plt.legend()

"""## Comparação dos modelos

"""

#dataframe comparativo
comparacao_3_modelos = pd.DataFrame()
comparacao_3_modelos['real'] = df_comparativo_lr_vc['real']
comparacao_3_modelos['previsao_regressao_linear'] =
df_comparativo_lr_vc['previsao']
comparacao_3_modelos['previsao_rede_neural'] = df_comparativo_rn['previsao']
comparacao_3_modelos['previsao_lstm'] = df_comparativo_lstm['previsao']
comparacao_3_modelos =
comparacao_3_modelos.loc[:,['real','previsao_regressao_linear','previsao_rede
_neural','previsao_lstm']]

comparacao_3_modelos

print("RMSE modelo Regressao Linear          : %.4f" % rmse_lr)
print("RMSE modelo Rede Neural Multicamada   : %.4f" % rmse_rn)
print("RMSE modelo Rede Neural LSTM          : %.4f" % rmse_lstm)

plt.figure(figsize=(16,8))
plt.title('Comparando a performance dos modelos')
plt.plot(comparacao_3_modelos['real'], label="real", color='blue',
marker='o')
plt.plot(comparacao_3_modelos['previsao_regressao_linear'], label="Reg
Linear", color='red', marker='o')
plt.plot(comparacao_3_modelos['previsao_rede_neural'], label="Rede Neural",
color='yellow', marker='o')
plt.plot(comparacao_3_modelos['previsao_lstm'], label="Lstm", color='green',
marker='o')
plt.xlabel('Data pregão')
plt.ylabel('Preço de Fechamento')
leg = plt.legend()

```