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

Edimilson Jose das Neves

Análise de séries temporais, para o mercado de Criptoativos

Jacareí - São Paulo
2021

Edimilson Jose das Neves

Análise de séries temporais, para o mercado de Criptoativos

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

Jacareí - São Paulo
2021

SUMÁRIO

1. Introdução

1.1. Contextualização

As informações que constam sobre os valores dos criptoativos são importantes aos investidores que querem entender sobre os números, mesmo sendo, essencialmente, muito dinâmicas e caóticas na sua natureza. A moeda Bitcoin, para quem não tem familiaridade com o assunto, é a primeira implementação de um conceito chamado “criptomoeda”. Esse conceito foi descrito, pela primeira vez, em 1998 por Wei Dai, cuja ideia sugeria uma nova forma de dinheiro que utilizasse a criptografia para controlar a sua obtenção e as transações.

Sendo assim, a partir desse contexto, os investidores devem se preparar para um terreno onde terão que lidar com series temporais não estacionarias e muito ruidosas.

Vale lembrar que, diferentemente da Bolsa de Valores, a qual sofre com diversos fatores macroeconômicos e notícias que envolvem empresas e a política Mundial, o Bitcoin não segue este padrão, visto que a movimentação dele se apresenta de acordo com a “oferta” e “procura”, sem ser atingida por rumores internos ou externos.

Por fim, diante do cenário exposto, o presente trabalho tem a intenção de utilizar técnicas em Mineração e Modelos Preditivos de Dados, visando o desenvolvimento de um script em Python.

1.2. O problema proposto

O presente trabalho consiste na utilização de Análise Exploratória e Modelagem Preditiva para extração de informações das séries temporais, com o intuito de auxiliar os investidores na obtenção de *insights* para tomada de decisões e, assim, apresentarem maior rentabilidade.

Para isso, são analisadas as séries temporais das criptomoedas, tendo-se como objetivos desta análise:

- Realizar uma Análise Descritiva dos dados da criptomoeda;

 Verificar os dados, utilizando técnicas estatísticas;

 Analisar, de forma comparativa, uma criptomoeda para verificar a rentabilidade das mesmas;

 Criar modelos preditivos para o Bitcoin utilizando a biblioteca ARIMA e RNN baseada na tecnologia LSTM.

 Analisar qual modelo preditivo se adequa melhor às séries temporais.

2. Coleta de Dados

Os dados foram obtidos no site do *yahoo finance* e importados via biblioteca pandas_datareader no Python, o qual é encontrado no link a seguir:

https://sg.finance.yahoo.com/quote/BTC-USD/history?p=BTC-USD

A Análise Exploratória correspondeu ao período de 2017 à 2020, sendo 70% do período correspondente para treinamento e o restante (30%) para teste.

A Tabela 1 a seguir, apresenta informações acerca do formato de cada campo extraído pelo pandas_datareader, sendo utilizado para os trabalhos os preços de fechamento do ativo em questão.

Tabela 1. Tipo de dados importados

Nome da coluna/campo	Descrição	Tipo
Date	Data da cotação	Datetime
Open	Valor de abertura do preço	float64
Hight	Maior valor do dia	float64
Low	Menor valor do dia	float64
Close	Valor de fechamento do preço	float64
Adj.Close	Ajuste valor fechamento	float64
Volume	Volume de negociação	float64

3. Processamento/Tratamento de Dados

A partir da leitura dos dados, construiu-se a Tabela 2 a seguir para melhor visualização das informações:

Tabela 2. Informações dos DataSets

Nome do dataset	Descrição
ds_train	Dados de treinamento para o ativo escolhido no período de 01/01/2017 até 15/10/2019
ds_teste	Dados de teste para o ativo escolhido no período de 2019-16/10/2019 até 31/12/2020

Após a seleção do ativo, utilizou-se dados diários do período compreendido entre 2017 e 2020, realizando-se análise exploratória dos dados. Como resultado, identificou-se que constava na base uma quantidade de 1457 dados, sendo 1019 deles para treinamento e 438 para teste, não havendo qualquer valor nulo.

4. Análise e Exploração dos Dados

Dando sequência à análise das informações obtidas, realizou-se uma análise estatística dos dados, ressaltando-se que o foco da análise foi o preço de fechamento, a partir do qual se criou um modelo de previsão de séries temporais (Tabela 3).

Tabela 3. Analise estatísticas dos dados

Analise estatística dos dados						
	High	**Low**	**Open**	**Close**	**Volume**	**Adj Close**
count	1457.000000	1457.000000	1457.000000	**1457.000000**	1.457000e+03	1457.000000
mean	7691.931867	7302.281155	7498.353823	**7517.429967**	1.424749e+10	7517.429967
std	4387.164809	4114.955043	4246.840791	**4280.668215**	1.355961e+10	4280.668215
min	823.307007	755.755981	775.177979	**777.757019**	6.085170e+07	777.757019
25%	4214.629883	4015.399902	4105.456055	**4122.939941**	3.783500e+09	4122.939941

50%	7534.990234	7232.676758	7378.200195	**7379.950195**	9.144851e+09	7379.950195
75%	9823.001953	9450.899414	9655.854492	**9663.181641**	2.225681e+10	9663.181641
max	29244.876953	28201.992188	28841.574219	**29001.720703**	7.415677e+10	29001.720703

Construiu-se um histograma (Figura 1), com o intuito de obter uma melhor visualização do preço de fechamento.

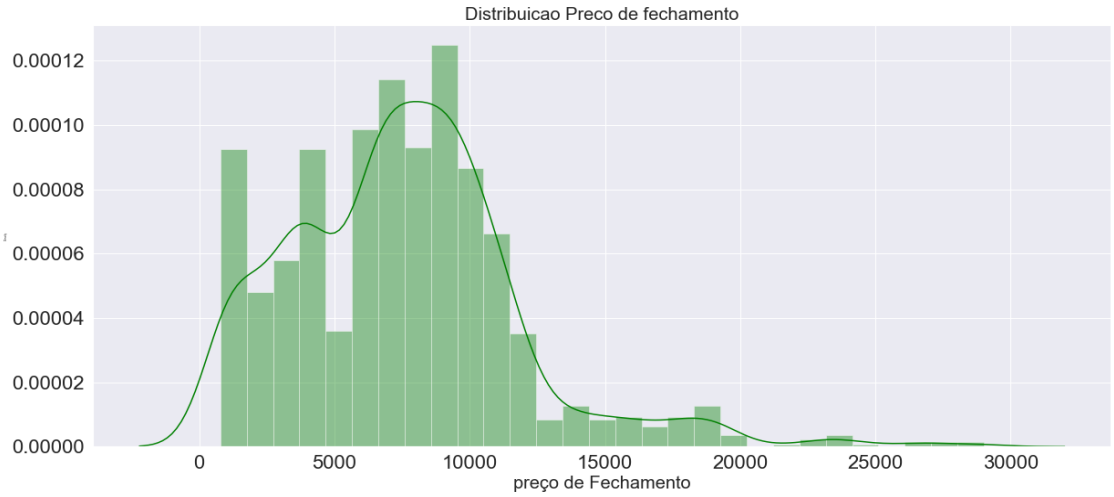

Figura 1. Hitórico de preços do BTC-USD

Pode-se verificar pelo histograma que o preço se concentra em torno de 7.000 e 9000 dólares, conforme é mostrado na Tabela 3 de análise estatística, onde a média e a mediana estão muito próximas.

Em seguida, plotou-se o histórico dos preços no período de 2017 à 2020, visando-se analisar os dados em um gráfico de *candlestick* e, assim, perceber a correlação entre abertura, fechamento, máximo e mínimo (Figura 2).

Figura 2. Anais em gráficos de *candlesticks*

Por fim, na Figura 3 a seguir, nota-se, claramente, uma tendência de alta e a média, gradativamente, subindo. Nesse caso, se a média fosse utilizada para realizar previsões futuras, o erro seria estatisticamente significante, pois os preços previstos estariam sempre abaixo do preço real.

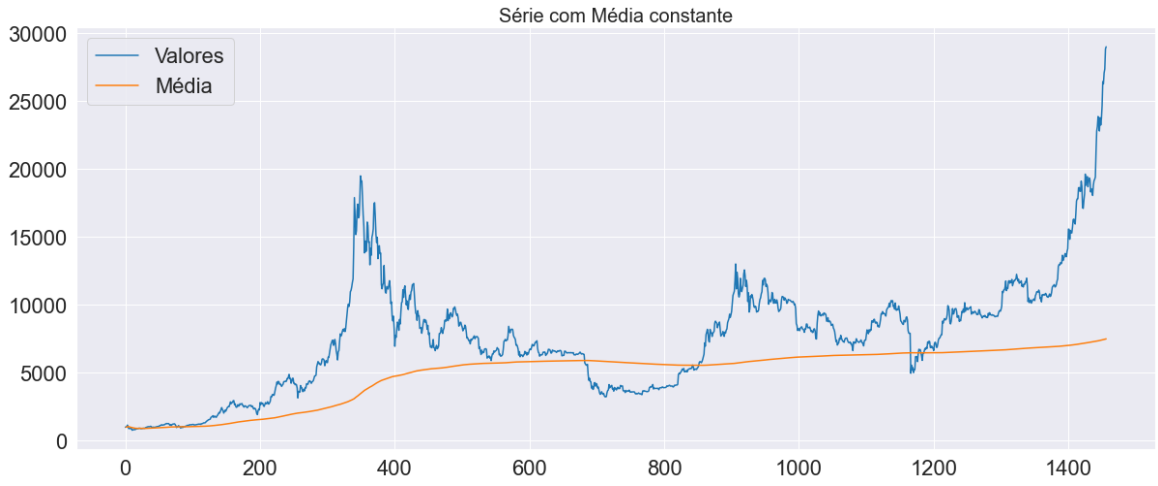

Figura 3. Analise dos dados históricos com a média constante

5. Criação de Modelos de *Machine Learning*

Utilizaram-se dois modelos preditivos, por meio da linguagem Python, para a moeda digital. A biblioteca ARIMA (AR-Auto Regression, I-Integrated, MA-Moving Average) e o LSTM (Long Short Term Memory) foram utilizados com a intenção de comparar e identificar o melhor método que se ajusta ao problema proposto.

5.1. Modelo preditivo ARIMA

O método ARIMA utiliza a auto regressão e as médias moveis para previsão de series temporais.

O intuito é prever dados futuros utilizando um modelo linear, havendo a necessidade de se incluir um número especificado de termos, sendo os dados preparados por um nível de diferenciação, a fim de torna-los estacionários.

O modelo ARIMA é considerado um caso geral dos modelos propostos por Box e Jenkins (1976), o qual é apropriado para descrever séries não estacionárias. Ele permite o ajuste de modelos que não possuam componentes sazonais, mas sim componentes de tendência, bem como comportamento homogêneo não-estacionário.

Geralmente, as séries encontradas apresentam tendência e sazonalidade concomitantemente.

A tendência implica o comportamento da série temporal, se é crescente ou decrescente. A sazonalidade mostra as flutuações decorrentes a períodos que se repetem, sendo que esses ocorrem, geralmente, em períodos inferiores a um ano, podendo ser diária, semanal, mensal, trimestral, semestral, etc.

Segundo Morretin e Toloi (2006), existem diversos métodos e modelos para analisar séries temporais, uma modelagem estatística bastante difundida é a de Box e Jenkins (1976). Uma vez que estes modelos são oriundos de ajustes Auto Regressivos Integrados de Médias Móveis, “Auto Regressive Integrated Moving Average” difundidos como ARIMA (p,d,q).

O modelo ARIMA (p,d,q) pode ser representado por:

p – O número de *lags* que devem ser incluídos no modelo.

d – O número de vezes que as observações serão diferenciadas.

q – O tamanho de uma janela de média móvel, também chamada de ordem da média móvel.

Toda série temporal pode ser decomposta em 3 partes: **tendência**, **sazonalidade** e **resíduo**, que é o que se visualiza após retirar da série a seguir (Figura 4) a separação dessas partes:

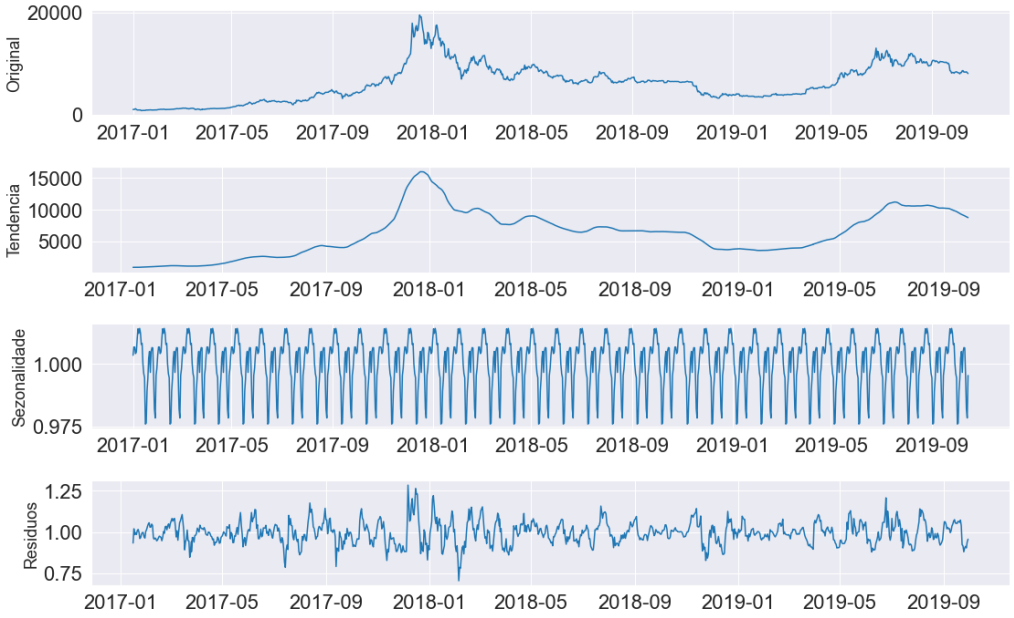

Figura 4. Decomposição da series temporal

Na Figura 5 a seguir, plotaram-se os dados de teste e de treinamento em relação à média, notando-se uma tendência de alta e a média, gradativamente, subindo. Verifica-se, claramente, que a série não é estacionária.

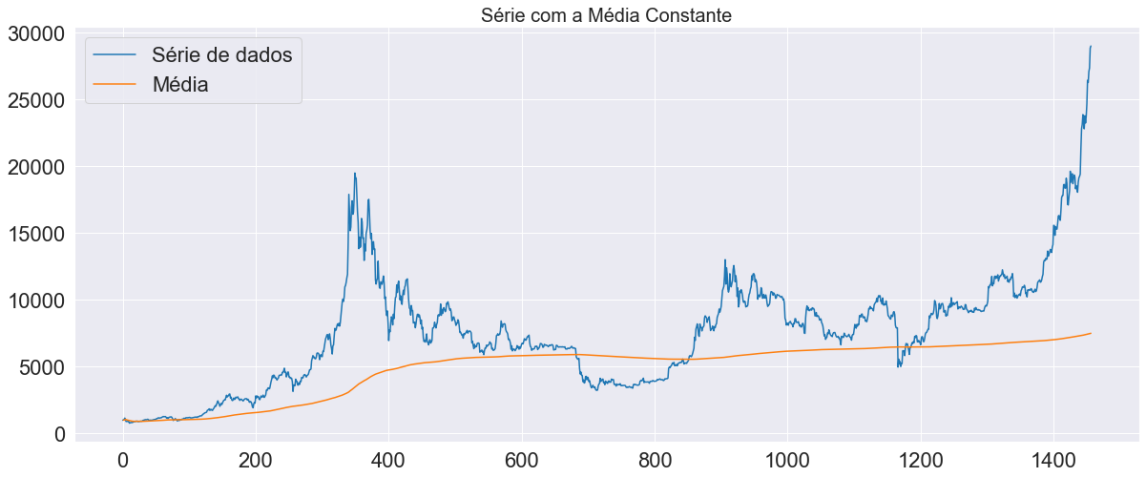

Figura 5. Series da dados com a Média constante

É evidente que existe uma tendência de crescimento global nesses dados, juntamente com algumas variações sazonais. Sendo assim, pode-se verificar *Estacionariedade,* utilizando os seguintes testes:

1. **Plotagem de *Rolling Statistics* ("estatística de rolagem"):** pode-se traçar a média ou a variância móvel e verificar se ele varia com o tempo.

2.**Teste Dickey-Fuller**: esse é um dos testes estatísticos para verificar *Estacionaridade*.

As Figuras 6 e 7 apresentam os resultados dos passos que foram realizados:

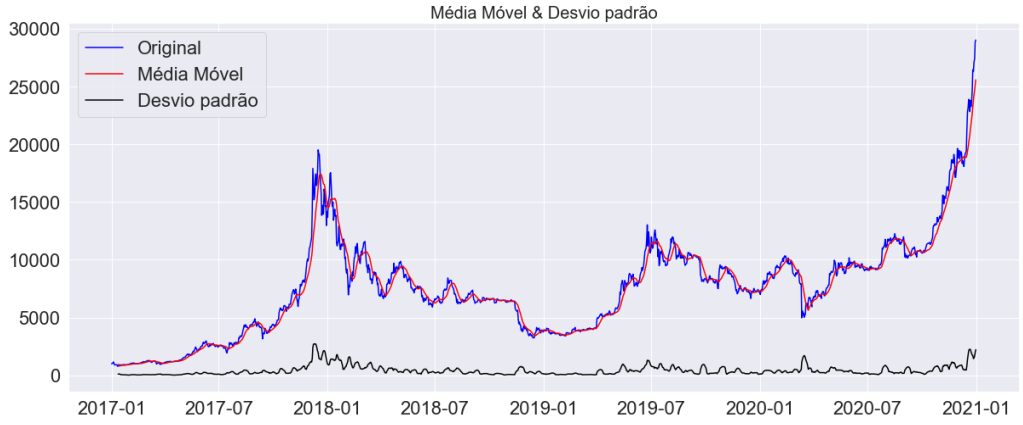

Figura 6. Analise das média com Rolling Statistcs

```
Resultado do teste Dickey-Fuller:
Teste                      0.637953
valor p                    0.988515
Num de lags               21.000000
Num de Observações      1435.000000
Valor Critico (1%)        -3.434915
Valor Critico (5%)        -2.863556
Valor Critico (10%)       -2.567843
dtype: float64
```

Figura 7. Resultado Dickey Fuller antes da aplicação da diferenciação

A partir do teste de Dickey-Fuller, pode-se verificar, como era esperado, que a série não é estacionária, visto que o valor de p é 98%, ou seja, muito maior que 5% ou 0.05.

5.2. Transformando a série em estacionária

Com o intuito de transformar a série em estacionária, pode-se utilizar diversos métodos, tais como: tomar o logaritmo, a raiz quadrada, a raiz cúbica, etc.

Optou-se por utilizar a diferenciação de primeira ordem d=1 para remover os sinais de tendências e reduzir a variância, deixando a serie estacionaria. Após a aplicação desse método, apresentou-se como resultado o que está presente na Figura 8 a seguir:

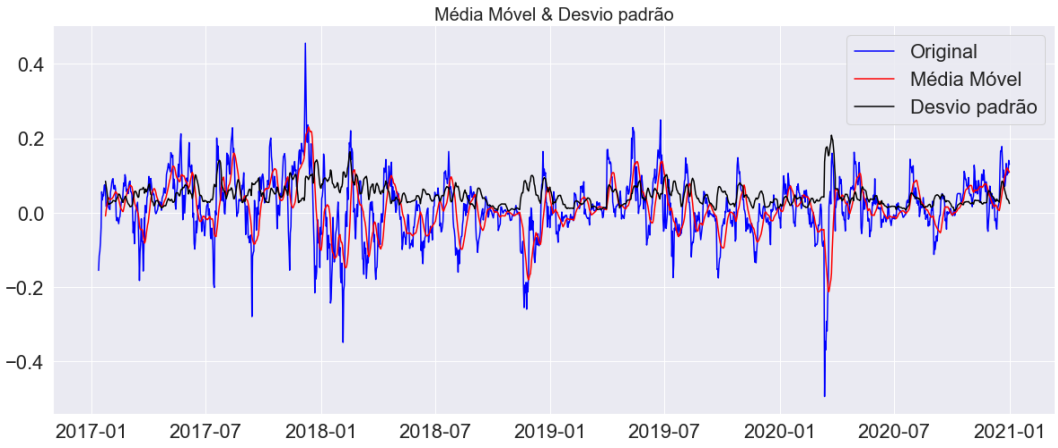

Figura 8. Variaçao dos resíduos após o uso da diferenciação

É importante ressaltar que o gráfico (Figura 8) ficou de fácil visualização, porém para confirmar se o resultado foi adequado, realizou-se o Teste Dickey- Fuller, obtendo-se como resultado o que segue na Figura 9:

```
Resultado do teste Dickey-Fuller:
Teste                  -7.235179e+00
valor p                 1.947960e-10
Num de lags             1.100000e+01
Num de Observações      1.434000e+03
Valor Critico (1%)     -3.434918e+00
Valor Critico (5%)     -2.863558e+00
Valor Critico (10%)    -2.567844e+00
dtype: float64
```

Figura 9. Resultado do teste Dickey-Fuller

A partir da análise da Figura 9, o valor de p se apresentou muito melhor, situando-se abaixo de 5% ou 0.05.

Dando sequência às análises, construiram-se os gráficos com Função de Autocorrelação (ACF) e Função de Autocorrelação Parcial (FACP) (Figura 10), resultando-se em uma forte auto correlação.

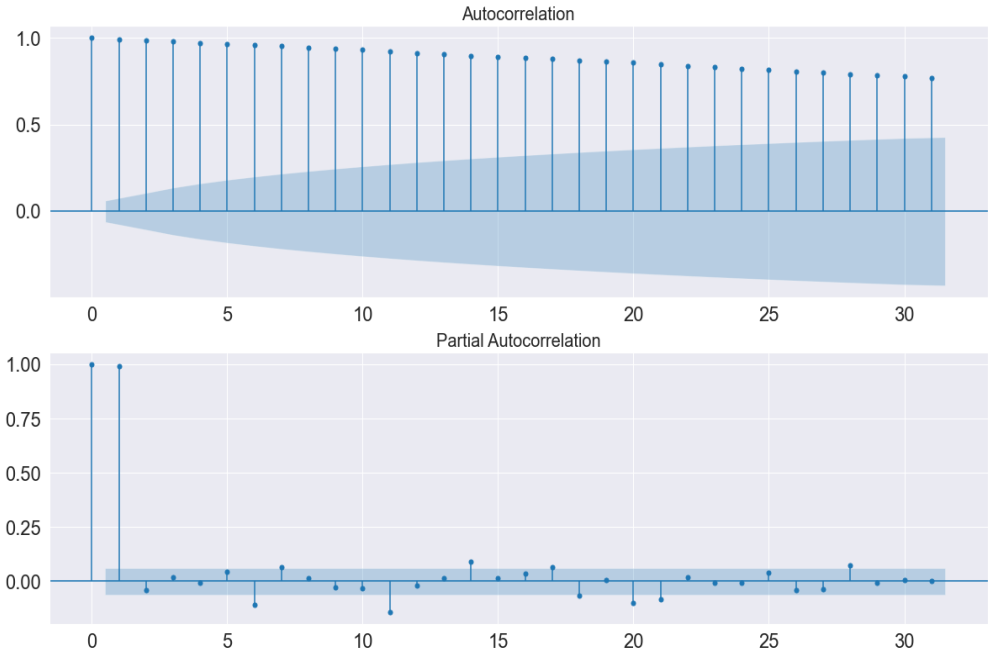

Figura 10. Gráficos de Autocorrelação e de Parcial Autocorrelação

5.3. Utilizando a Técnica *Walk Forward*

Na Figura 11 a seguir, apresenta-se gráfico que versa sobre a técnica *walk-forward,* utilizada para caminhar com os dados ao longo do tempo. Os dados de treino e teste vão se alterando, ou seja, são atualizados a medida que vamos caminhando na linha do tempo.

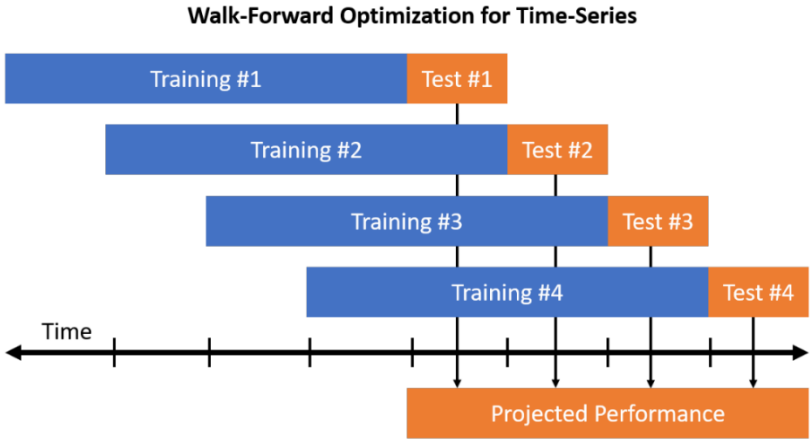

Figura 11. Esquema de Otimização com *Walk-Foward*

Defini-se a função ARIMA(p,d,q) os componentes p (número de time lags do modelo auto-regressivo)= 2, d(grau de diferenciação) = 1 e q (ordem do modelo de média-móvel) = 1.

Por fim, determinou-se a utilização desse modelo, o qual melhor se adequou à série temporal estudada, conforme verifica-se na Figura 12 a seguir.

ARIMA Model Results

Dep. Variable:	D.y	No. Observations:	1443
Model:	ARIMA(2, 1, 1)	Log Likelihood	-11092.497
Method:	css-mle	S.D. of innovations	527.509
Date:	Sat, 24 Apr 2021	AIC	22192.993
Time:	14:35:34	BIC	22214.091
Sample:	1	HQIC	22200.868

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1.D.y	0.2205	0.325	0.679	0.497	-0.416	0.857
ar.L2.D.y	-0.0713	0.028	-2.557	0.011	-0.126	-0.017
ma.L1.D.y	-0.1785	0.325	-0.549	0.583	-0.816	0.459

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	1.5459	-3.4103j	3.7443	-0.1823
AR.2	1.5459	+3.4103j	3.7443	0.1823
MA.1	5.6009	+0.0000j	5.6009	0.0000

Figura 12. Resultados do Modelo ARIMA

5.4. Modelo Preditivo LSTM

A partir do que fora descrito anteriormente, a intenção é utilizar uma Rede Neural LSTM para prever o comportamento do btc-usd.

Primeiramente, definem-se alguns parâmetros, tais como a previsão de até n dias à frente (foward_days), observando-se m dias passados (look_back). Ou seja, para uma entrada de m dias passados, a saída da rede será os próximos n dias. Na sequência, a idéia foi de testar com k períodos (num_periods), onde cada período seria um conjunto de n dias previstos (Figura 13).

```
#Criação do modelo LSTM utilizando a biblioteca Keras

#inicia sessao
regressor = Sequential()

#Passamos a flag retunr_sequences=True pois teremos mais de uma camada conectada em nossa rede

regressor.add(LSTM(units=128,return_sequences = True, input_shape = (x_train_lstm.shape[1],1)))
# Utilizando a técnica de Dropout com 0.3, assim teremos 30% dos neuronios desligados aleatoriamente por passada
# Esta técnica de dropout evita o overfitting (quando o modelo aprende demais com os dados de treinamento)
regressor.add(Dropout(0.3))

#ASegundo comando LSTM com o Dropout
regressor.add(LSTM(units=64))
regressor.add(Dropout(0.3))

#Agora passamos o comando Dense ao modelo para ligar todos os neuronios do comando anterior
#Deixamos a função de ativação cmo linear pois estamos resolvendo um problema de regressão, porém com os nosso dados
#estão normalizados entre 0 e 1, podíamos utilizar a função signoid, pois também trabalhs com range de 0 e 1
regressor.add(Dense(1,kernel_initializer='uniform',activation='linear'))
```

```
#Compila o RNN, neste caso utiliza o otimizador 'Adam'
regressor.compile(optimizer = 'adam', loss = 'mean_squared_error', metrics=['mean_absolute_error'])
```

Figura 13. Criação do Modelos LSTM

Após a criação da rede neural, realizou-se a compilação dessa rede, por meio do comando *compile* da biblioteca keras. A função de custo, que é a função que tenta ser minimizada pela rede neural escolhida, foi a de erro quadrático médio.

Após compilar a rede, realizou-se o treinamento dela com o *dataset* de treino, que corresponde aos primeiros 70% da série histórica do ativo escolhido. Esses 70% precisam ser sequenciais, a partir do ponto mais antigo dos dados, já que se está lidando com um problema de séries temporais. Foi também passado como *input* do treinamento, o *dataset* de validação, ou teste, que são os próximos 30% da série histórica.

Além do *dataset*, são passados mais dois *inputs*: *epoch* e *batch size*. A *epoch*, ou época, é a quantidade de vezes que os dados de treino são passados inteiramente

pela rede neural para treiná-la. Portanto, se o número de épocas é definido como 1, a rede neural treina o modelo, passando os dados apenas uma vez por seus perceptrons. Porém, se o número de épocas é 120, a rede neural passa os dados de treino cento e vinte vezes pelos perceptrons, a fim de deixar a rede melhor treinada.

Após cada época, a rede é testada com os dados de validação. Se a função de custo não diminuir, o *learning rate* α do passo de gradiente descendente é dividido pela metade, a fim de treinar o modelo melhor.

Para visualisar o digrama do modelo foi realizado o comando abaixo representado pela figura 14

```
#Visualizacao grafica do modelo

keras.utils.plot_model(regressor, "model_rnn_with_shape_info.png", show_shapes=True)
```

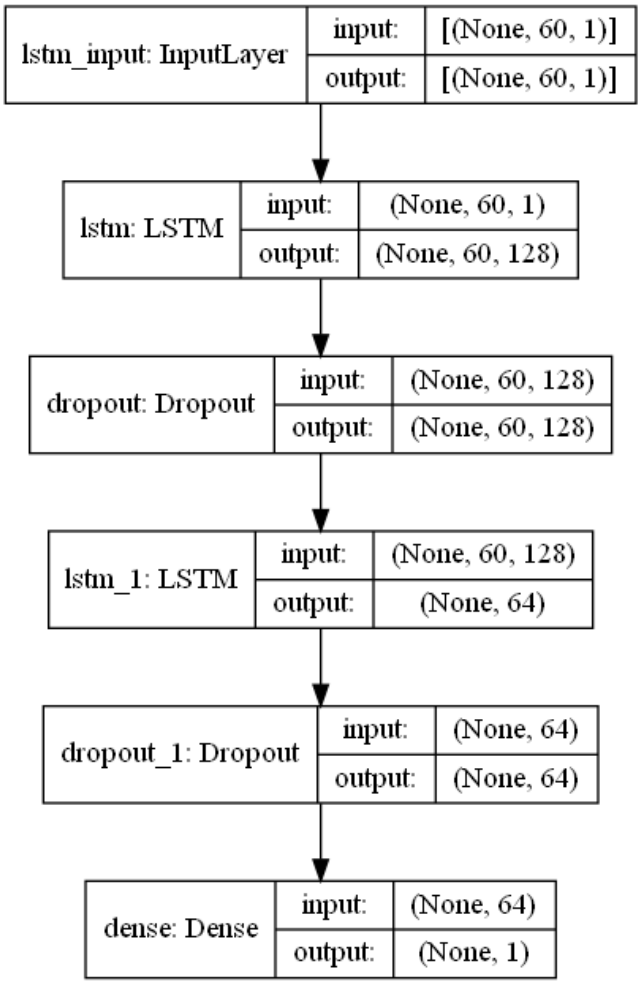

Figura 14. Diagrama

6. Apresentação dos Resultados

A partir das análises realizadas no presente estudo, pode-se concluir que os métodos de modelagem estatística no âmbito de séries temporais são demasiadamente excelentes (MIRANDA e LIZZI, 2019) para se ajustar os dados previstos com os dados reais.

6.1. Modelo Preditivo com ARIMA - *Walk Forward*

Na Figura 15 a seguir, percebe-se que o modelo ficou muito bem ajustado, cujos dados previstos acompanharam os dados reais com uma pequena diferença.

Figura 15. Dados Previstos x Dados Reais X Dados de Treinamento

Adicionalmente, na Figura 16, observam-se os dados previstos e os dados reais, verificando-se melhor o ajuste realizado.

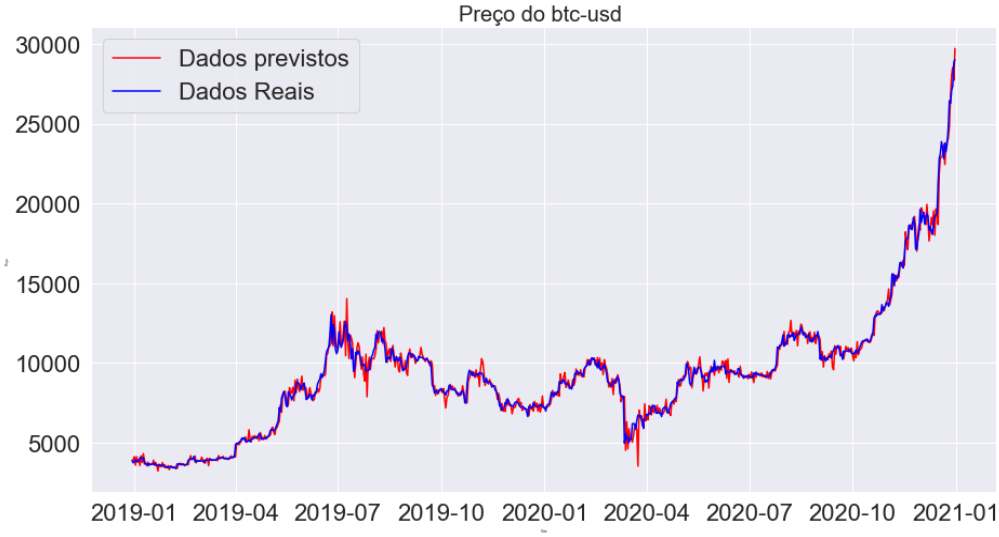

Figura 16. Dados Previstos x Dados Reais

Erro quadrático:

```
rmse = np.sqrt(mean_squared_error(test, predictions))
print('Test RMSE: %.3f' % rmse)

Test RMSE: 476.024
```

Figura 17. Erro quadrático ARIMA

6.2. Modelo Preditivo LSTM

Quanto ao Modelo Preditivo LSTM, analisando-se os dados combinados com treinamento, teste e previstos com todos os históricos, podemos ver que os dados previstos se ajustam muito bem (Figura 18).

Figura 18. Dados de Treinamento x Dados de teste X Dados previstos

Adicionalmente, na Figura 19, o gráfico apresenta somente os dados de teste e os previstos com os dados de um ano de dados.

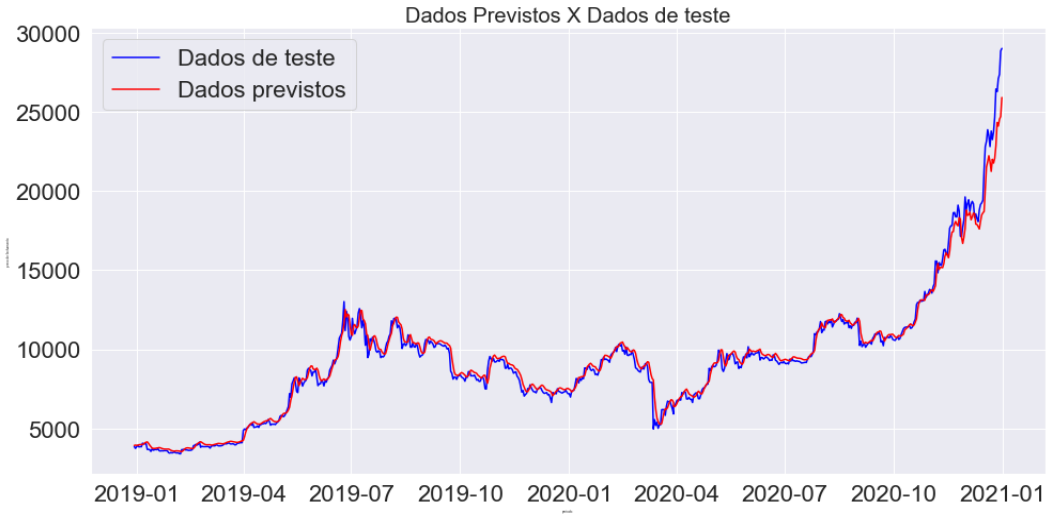

Figura 19. Dados previstos X Dados de Teste

Erro quadrático:

```
#Erro quadrático médio da raiz
print('RMSE: ', np.sqrt(mean_squared_error(test_lstm,predictions_lstm)))
```

```
RMSE:  483.46525587589423
```

Figura 20. Erro quadrático LSTM

7. Considerações Finais

Analisando o erro quadrático, o ARIMA, com a técnica Walk-forward, apresentou-se melhor se comparado ao LSTM. Porém, devido à pequena diferença em relação ao erro quadrático em ambos os métodos, não há como afirmar se o método ARIMA é realmente melhor do que o LSTM.

8. Links

- Link para o vídeo no Youtube com a apresentação resumida
- https://www.youtube.com/watch?v=KmmHICLjBSw
- Link para o repositório Github com o conteúdo do trabalho https://github.com/edimisonneves/ModelosPreditivoCriptoAtivos-.git

REFERÊNCIAS

BOX, G.P.; JENKINS, G.M. **Time series analysis, forecasting and control.** San Francisco: Holden-Day, 1976.

BROWNLEE, J. **Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras.** Disponível em: https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/ Acesso em fev. 2021.

KERAS. Disponível em: <https://keras.io//>. Acesso em: 15 jan. 2021.

MORRETIN, P.A.; TOLOI, C.M.C. **Análise de séries temporais.** São Paulo: Edgard Blucher, 2006.

MIRANDA, L.F.P.; LIZZI, E.A.S. **Modelagem estatística aplicada ao estudo de demanda e consumo energético.** Em: Análise de dados com métodos estatísticos aplicados. Universidade Tecnológica Federal do Paraná, Cornélio Procópio, 2019.