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 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 caótica na sua natureza. ” A moeda Bitcoin , para quem não tem familiaridade com o assunto é a primeira implementação de um conceito chamado “criptomoeda”. Conceito que foi descrito pela primeira vez em 1998 por Wei Dai, onde sua ideia sugeria que uma nova forma de dinheiro que use a criptografia para controlar sua obtenção e as transações. ”

Portanto os investidores, devem se preparar para um terreno, onde terá que lidar com series temporais não estacionarias e bastante ruidosas.

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

Diante disso, neste trabalho utilizarei técnicas em Mineração e Modelos Preditivos de Dados, onde foi desenvolvido um script em Python.

1.2. O problema proposto

Este trabalho consiste no uso de Análise Exploratória e Modelagem Preditiva para extração de informações das séries temporais para auxiliar os investidores a terem insights para tomada de decisões e assim obterem maior rentabilidade.

Para isso, são analisadas as séries temporais das criptomoedas, assim, tem-se como objetivos dessa análise:

 Fazer uma Análise Descritiva dos dados da criptomoeda;

 Verificar os dados utilizando técnicas estatística;

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

 Verificar se existe relação entre elas;

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

2. Coleta de Dados

Os dados foram obtidos no site do yahoo finance e importados via biblioteca pandas_datareader no Python, no link abaixo.
https://sg.finance.yahoo.com/quote/BTC-USD/history?p=BTC-USD

O período de Análise Exploratória corresponde aos anos de 2017 até 2020, sendo 70% do período correspondente para treinamento e o restante, 30% para teste.

Abaixo segue a da tabela com o formato de cada campo extraído pelo pandas_datareader, utilizando para os trabalhos os preços de fechamento do ativo em questão.

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

Abaixo a tabela após a leitura dos dados

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

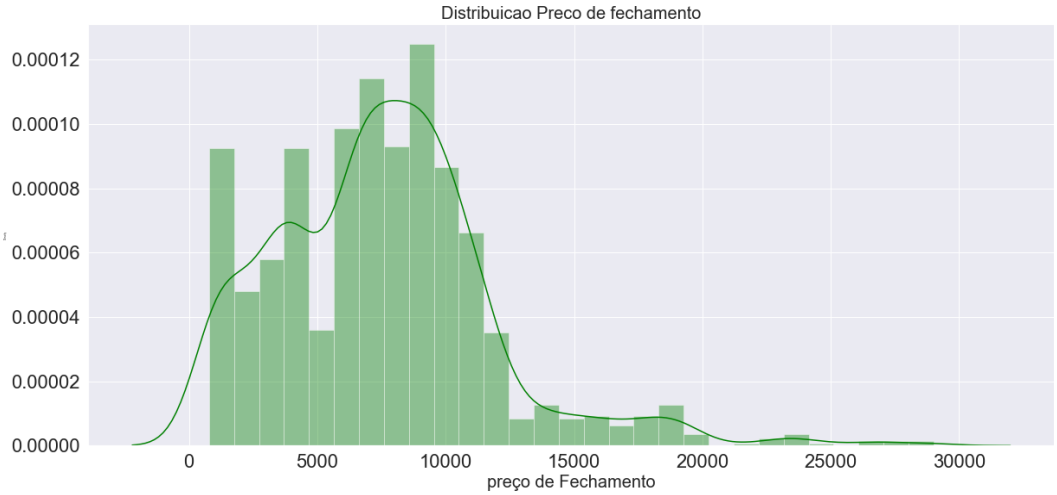
Para o ativo escolhido utilizaremos dados diários de 2017 até 2020, fizemos uma análise exploratória dos dados, verificamos que consta na base uma quantidade de 1457 dados sendo 1019 para treinamento e 438 para teste não havendo nenhum valor nulo.

4. Análise e Exploração dos Dados

Abaixo faço uma análise estatística dos dados lembrando que o preço de fechamento é o que vai será analisado, pois será criado um modelo de previsão de séries temporais com este campo.

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

Através do histograma podemos melhor ver a variação do preço de fechamento

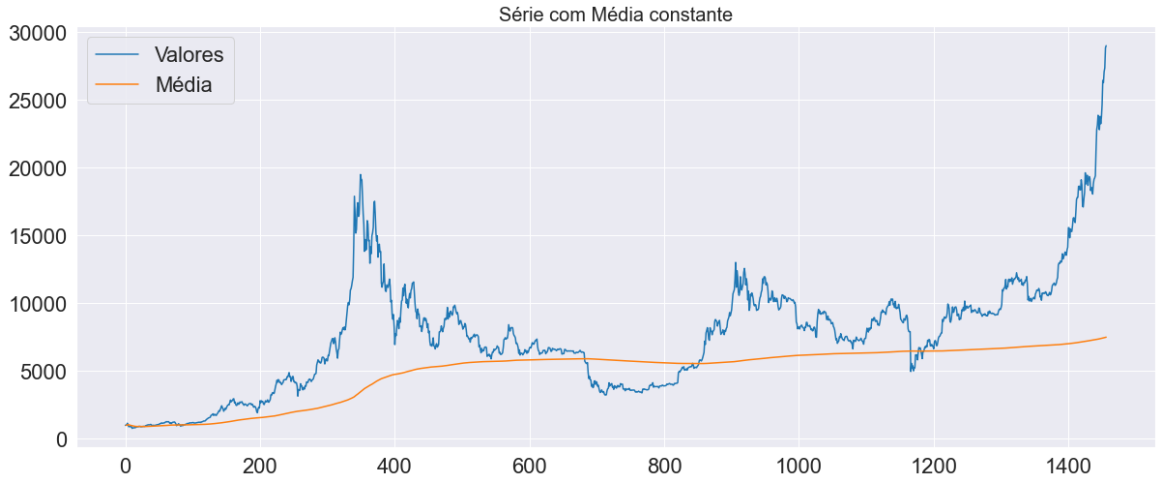


Podemos verificar pelo histograma que o preço varia em torno de 7.000 e 9000 dólares conforme é mostrado na tabela de análise estatística, onde a média e a mediana estão muito próximas.

Abaixo plotamos o histórico dos preços ao longo do tempo onde podemos analisar os dados em um gráfico de candlestick para visualizar a correlação entre abertura, fechamento, máximo e mínimo

No gráfico abaixo claramente nota-se uma tendência de alta e a média gradativamente subindo, neste caso se a média fosse usada para fazer previsões

futuras o erro ia ser significante, pois os preços previstos estariam sempre abaixo do preço real.



5. Criação de Modelos de Machine Learning

Utilizarei dois modelos preditivos para que será desenvolvido utilizando como linguagem o Python para a moeda digital utilizando a biblioteca ARIMA e o LSTM para onde podemos comparar qual método melhor se ajusta ao problema proposto

5.1. Modelo preditivo ARIMA

Para o método ARIMA (AR-Auto Regression, I-Integrated, MA-Moving Average) utiliza a auto regressão e as médias moveis para previsão de series temporais.

O intuito é prever dados futuros utilizando um modelo linear, para isso terá que ser incluído um número especificado de termos e os dados serão preparados por um nível de diferenciação a fim de tornar este estacionário.

O modelo ARIMA é considerado um caso geral dos modelos propostos por **Box e Jenkins (1976)** que é apropriado para descrever séries não estacionárias. O modelo ARIMA permite o ajuste de modelos que não possuam componentes sazonal mas possua componente de tendência

e quando essas componentes possuam comportamento homogêneo não-estacionário.

Geralmente, as séries encontradas possuem tendência, sazonalidade conjuntamente os dois.

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

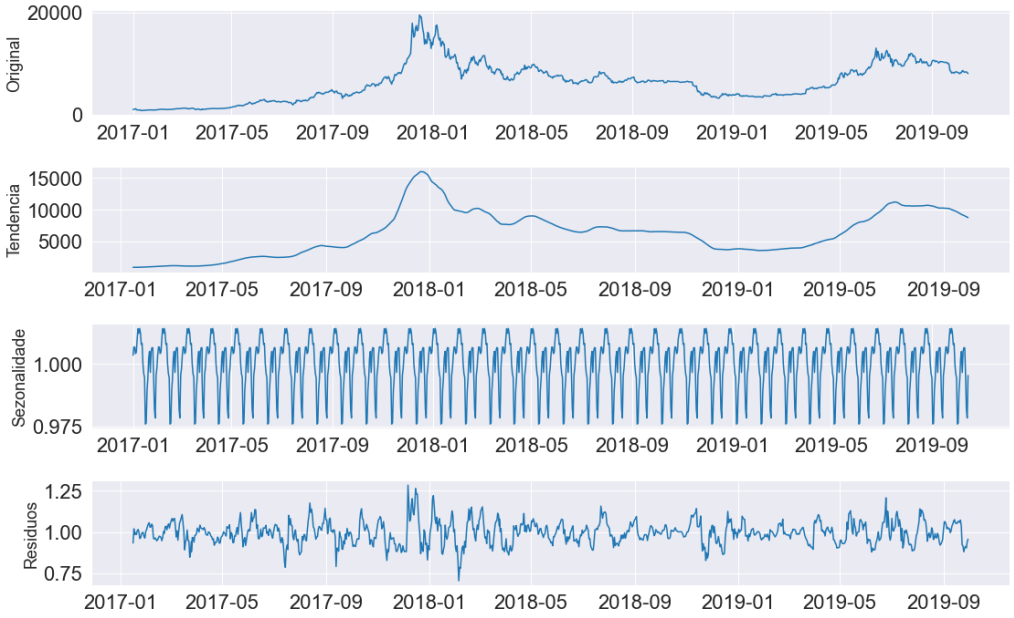
O modelo ARIMA (p,d,q) pode ser representado por:

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

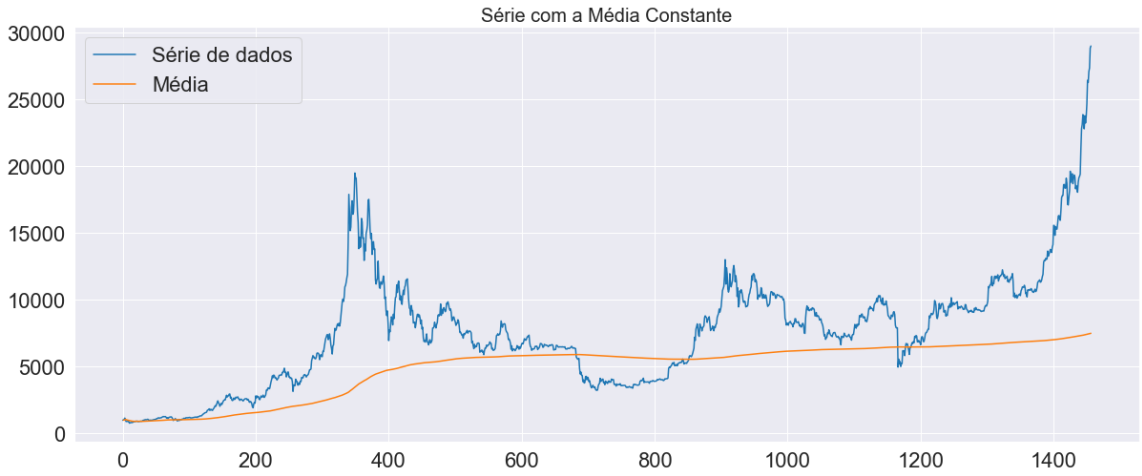
d – O número de vezes que a observações serão diferenciadas.

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 vemos após retirar da série abaixo a separação dessas partes:



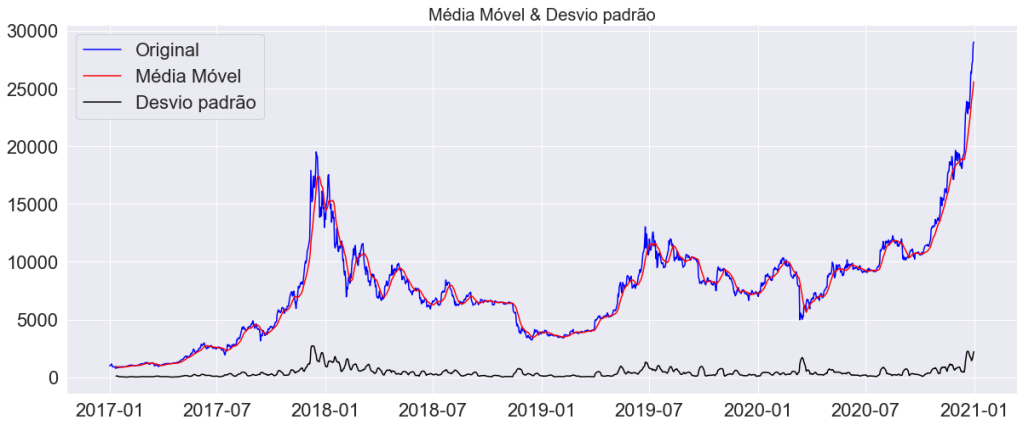
No gráfico abaixo plotamos os dados de teste e de treinamento em relação à média e nota-se uma tendência de alta e a média gradativamente subindo, verificam-se claramente que a série não é estacionária.



É evidente que existe uma tendência de crescimento global nesses dados, juntamente com algumas variações sazonais. Então, mais formalmente, podemos verificar *Estacionaridade* usando o seguinte:

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

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




```
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
```

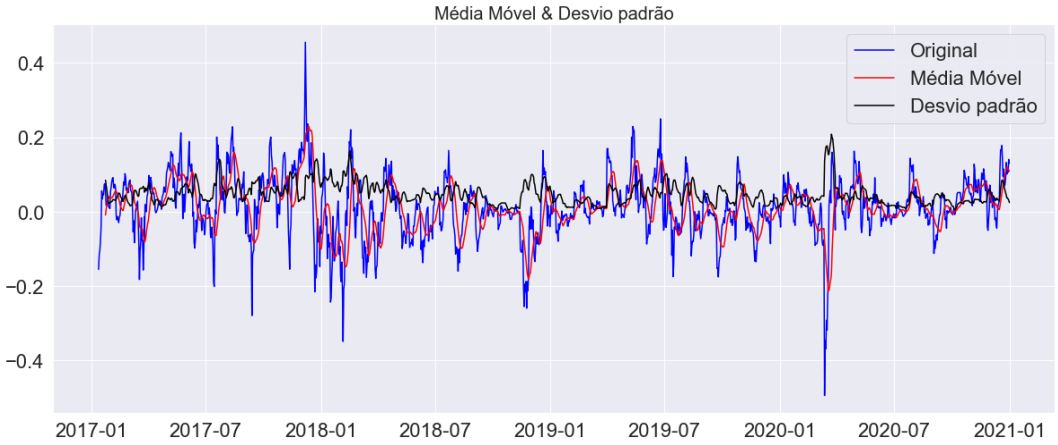
```
dtype: float64
```

Podemos verificar pelo teste Dickey-Fuller, como era esperado que a série não é estacionária, pois o valor p é 98% muito maior que 5% ou 0.05.

5.2. Transformando a série em estacionária

Para torná-la estacionária podemos usar vários métodos como tomar o logaritmo, a raiz quadrada, a raiz cúbica, etc.

Vamos 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 deste metodo temos o seguinte:

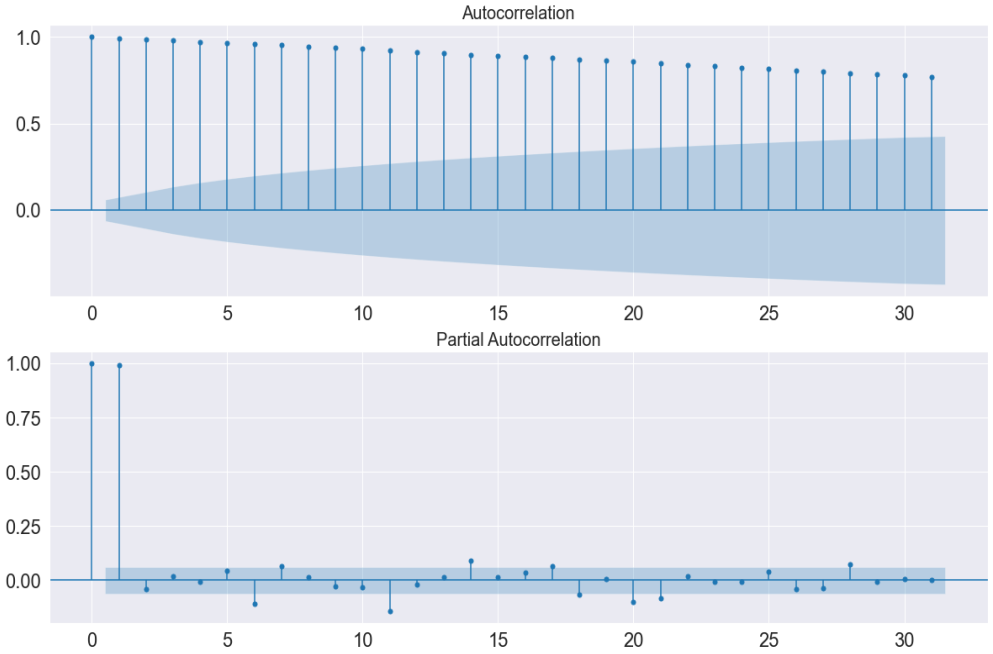


Visualmente ficou bem melhor mas para termos certeza utilizando o Teste Dickey- Fuller temos o resultado abaixo:

```
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
```

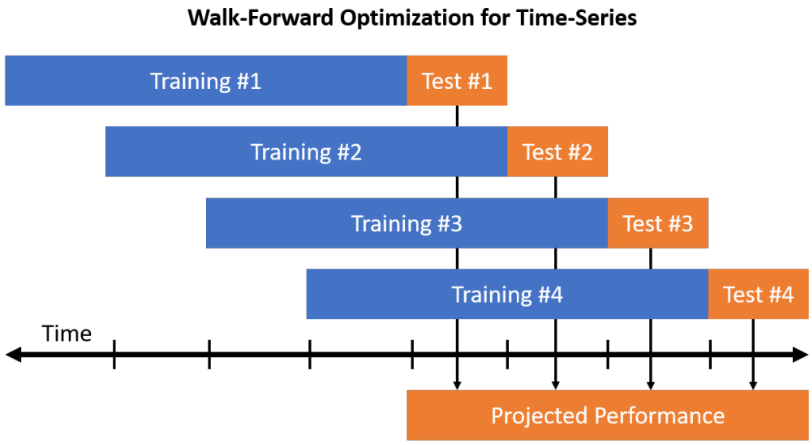
Como podemos ver o **Valor p** ficou muito melhor ficando abaixo de 5% ou de 0.05.

Plotamos o grafico com Função de autocorrelação (ACF) e a função de autocorrelação parcial (FACP) onde mostrar uma forte auto correlação.



5.3. Utilizando a Técnica Walk Forward

No gráfico abaixo, mostra a técnica **walk-forward** utilizada para caminhar com os dados ao longo do tempo os dados de treino e teste vão sofrendo mudanças, ou seja, tanto os dados de treino quanto os de teste vão sendo atualizados a medida que vamos caminhando na linha do tempo.



Iremos definir 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,

Vamos utilizar este modelo que melhor se adequa à série temporal estudada.

```
                             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
-----------------------------------------------------------------------------
```

5.4. Modelo Preditivo LSTM

Vamos agora utilizar uma Rede LSTM para prever o comportamento do btc-usd.

Primeiro, definimos alguns parâmetros. Queremos prever até n dias a frente (foward_days) observando m dias passados (look_back). Ou seja, para uma entrada de m dias passados, a saída da rede será os próximos n dias. Vamos testar com k períodos (num_periods), onde cada período é um conjunto de n dias previstos.

```
#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'))
```

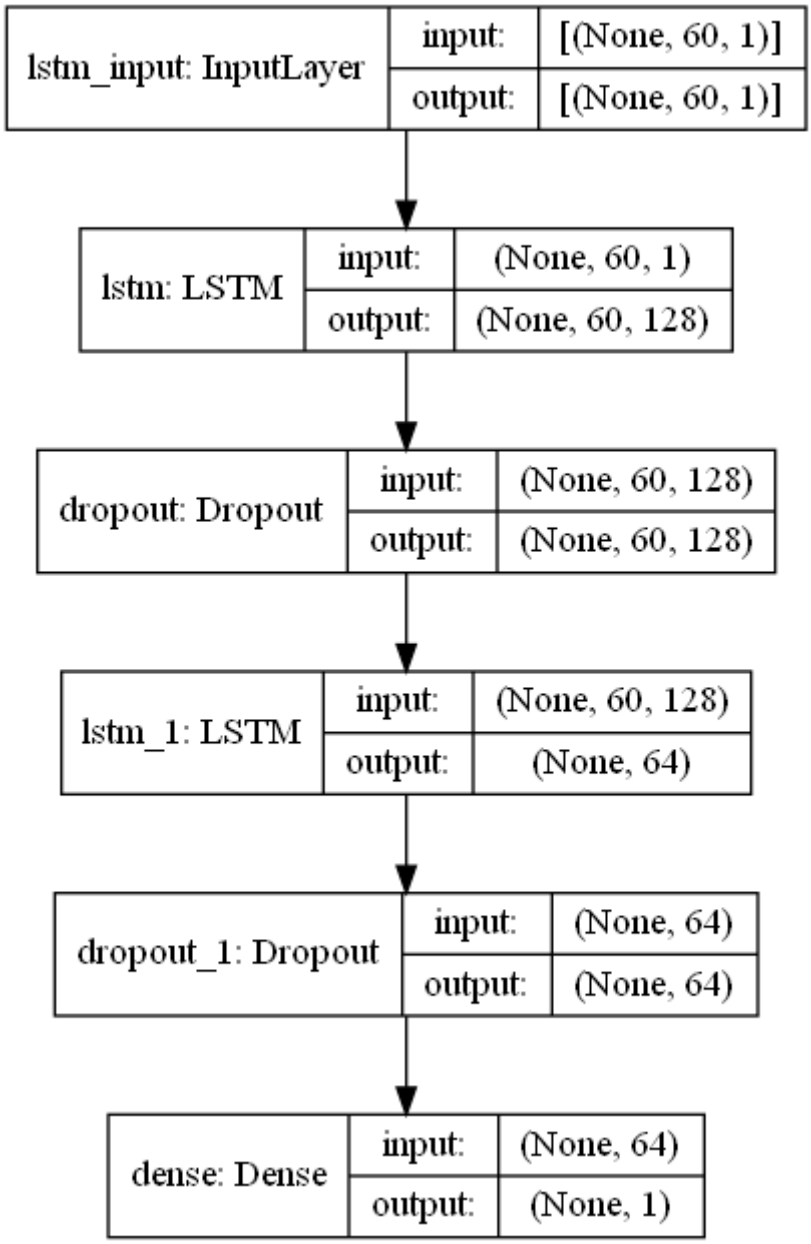
```
#Compila o RNN, neste caso utiliza o otimizador 'Adam'
regressor.compile(optimizer = 'adam', loss = 'mean_squared_error', metrics=['mean_absolute_error'])
```

Após a criação da rede neural, foi feita a compilação dessa rede através 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, foi realizado 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 estamos 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 mais bem 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.

Abaixo mostro o diagrama de saida do modelo.



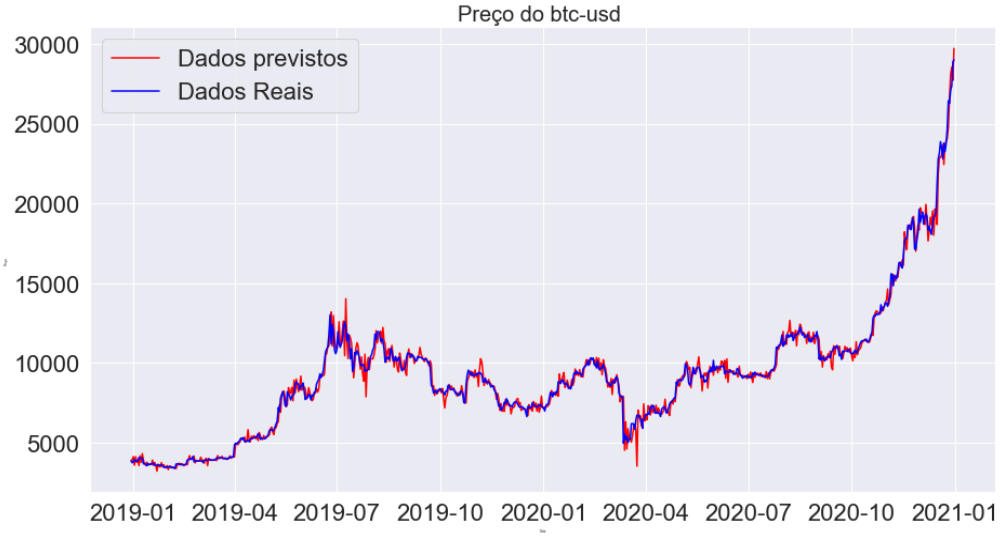
6. Apresentação dos Resultados

Pode-se concluir que os métodos de modelagem estatística no âmbito de séries temporais são demasiadamente excelentes.

6.1. Modelo Preditivo com ARIMA - Walk Forward

Com a imagem abaixo podemos ver que o modelo ficou muito bem ajustado, onde os dados previstos acompanharam os dados reais com uma pequena diferença.

A imagem abaixo podemos analisar somente os dados previstos e os dados reais onde podemos verificar melhor esse ajuste.



Erro quadrático:

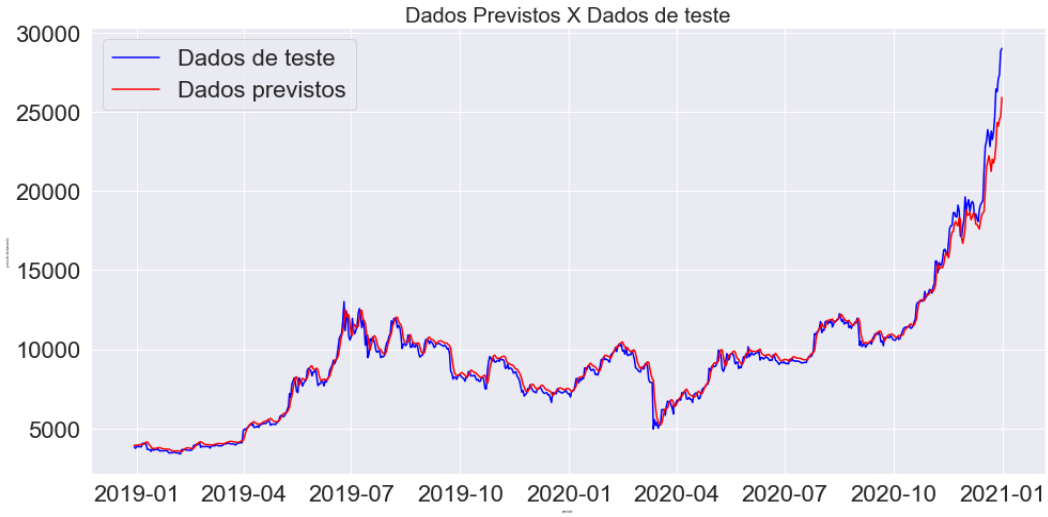
```
rmse = np.sqrt(mean_squared_error(test, predictions))
print('Test RMSE: %.3f' % rmse)
```

```
Test RMSE: 476.024
```

6.2. Modelo Preditivo LSTM (*Long Short Term Memory*)

Analisando os dados combinados com treinamento, teste e previstos com todos os históricos, podemos ver que os dados previstos se ajustam muito bem.

Abaixo o gráfico apresenta somente os dados de teste e os previstos com os dados de um ano de dados.



Erro quadrático:

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

```
RMSE:  483.46525587589423
```

7. 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/SeriesTemporais

REFERÊNCIAS

P.A. Morretin e C.M.C. Toloi**: Análise de séries temporais**. São Paulo: Edgard Blucher, 2006.

Lizzi, Profa. Dra. Elisangela Aparecida da Silva**, Análise de dados com métodos estatísticos aplicados.** Universidade Tecnológica Federal do Paraná, Cornélio Procópio 2019.

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

Brownlee, Jason. **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.