Resumo

Previsão do valor de Bitcoins utilizando algoritimos de Deep Learning

**Marcelo de Caux**

Instituto de Computação - Universidade Federal Fluminense

marcelocaux@id.uff.br

Este artigo apresenta o trabalho realizado na adaptação e simulação da previsão de valores de Bitcoin publicada por Igor Bobriakov no site Medium, sob o título, Bitcoin price forecasting with deep learning algorithms [1], a grande diferença entre o experimento original e este que realizei está na granularidade e objetivo final do trabalho, enquanto o experimento original buscou avaliar a previsão de um valor diário de Bitcoin, este expe-rimento buscou prever os valores minuto a minuto de um dia inteiro de cotações, possibilitando avaliar os movimentos esperados para um dia futuro de negociações, dando ao analista a possibilidade de prever o bom momento de compra e venda de Bitcoins em um dia futuro.

1 Introdução

A proposta de predição de valores de Bitcoin tomou corpo a partir da popularização desta moeda, sua crescente utilização e sua alta volatilidade, estas características tornam esta o Bitcoin um forte atrativo para investidores. Criar um mecanismo eficaz para prever variações neste mercado define uma grande vantagem competitiva no mercado financeiro, no entanto este mecanismo deve ser capaz de prever pelo menos um ciclo diáro de cotações, minuto a minuto de forma a representar o fluxo contínuo da moeda.

1.1 Premissas

Para poder atingir o objetivo de ter a previsão do ciclo diário de cotações considerei as seguintes premissas:

* Estudar LSTMs
* Entender o funcionamento dos scripts
* Adaptar o funcionamento dos scripts a granularidade desejada
* Rodar simulações e coletar os resultados

O estudo inicial do artigo indicou a absoluta necessidade de entender o funcionamento de uma LSTM, pois ela é utilizado como ferramenta básica para predição. O estudo da LSTM foi realizado basicamente lendo artigos na internet [2] [3], o processo para entender os scripts fornecidos no artigo passaram pela penosa fase de instalação das bibliotecas necessárias, descritas na preparação do ambiente de simulação, a replicação do experimento original, a identificação das alterações necessárias para mudar a granularidade dos dados e por fim executar simulações e coletar os resultados. As simulações foram realizadas usando Python 3.6, as bibliotecas Keras, TensorFlow, Numpy, Pandas, statmodels, scipy, sklearn, matplotlib, e plotly, os experimentos foram rodados usando Jupyter notebook.

1.2 Preparação do ambiente de simulação

Para poder rodar os experimentos foi necessário preparar um novo computador instalando desde o sistema operacio-nal até todas as bibliotecas necessárias, exigidas pelos scripts do artigo que serve de base para este estudo.

1.2.1 Hardware utilizado

Os experimentos foram conduzidos utilizando um processador de 6 núcleos e 2,2GHz com Hypertread, 16GB RAM e uma GPU NVIDIA GTX1060 de 1280 núcleos e 6GB RAM.

1.2.2 Software utilizado

Sistema operacional Ubuntu 18.04 LTS 64bits, Python 3.6, Anaconda3 5.2, CUDA 9.2, TensorFlow 1.8.0, Keras, Numpy, Pandas, statmodels, scipy, sklearn, matplotlib, e plotly.

1.2.3 Instalação do Software utilizado

O processo de instalação de SW foi extremamente difícil, o prórpio sistema operacional apresentou dificuldades para ser instalado em SSD e foi vencido após assitir um vídeo no YouTube [4], a instalação do CUDA foi realizada várias vezes sem sucesso até encontrar o tutorial web [5], Anaconda3 foi instalado usando o tutorial [6], a instalação do TensorFlow e Keras também seguiu um tutorial web [7], as demais bibliotecas foram instaladas usando Anaconda Conda.

2 O experimento original

O experimento original [1], tinha como objetivo identificar um valor diário de Bitcoin em um dia futuro, para realizar esta tarefa o autor propos utilizar uma base de dados pública existente em [8] que cobria as cotações de Bitcoins, minuto a minuto no período entre 2011-12-31 07:52:00 a 2017-10-20 23:59:00, no entanto, para poder realizar previsões em bases diárias foi necessário agrupar os dados por dia, utilizando a coluna de média ponderada do minuto existente na base, reduzindo de uma granularidade de 1440 registros minuto a minuto para 1 registro diário. além de sumarizar os registros por data o autor restringiu o período obeservado para 2016-01-01 a 2017-10-20, este período foi dividido em:  
Treinamento: 2016-01-01 a 2017-10-15  
Teste: 2017-08-01 a 2017-10-20  
Predição: 2017-10-15 a 2017-10-20  
O passo seguinte a preparação dos dados foi avaliar a flutuação dos dados , ou seja, checar se os dados sofrem influências de fatores como tendência e sazonalidade, esta avaliação pode ser feita no gráfico apresentado na figura 1.

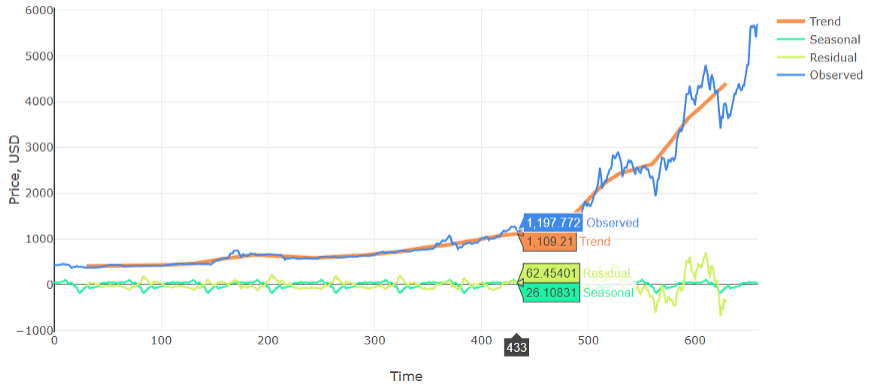


Figura 1

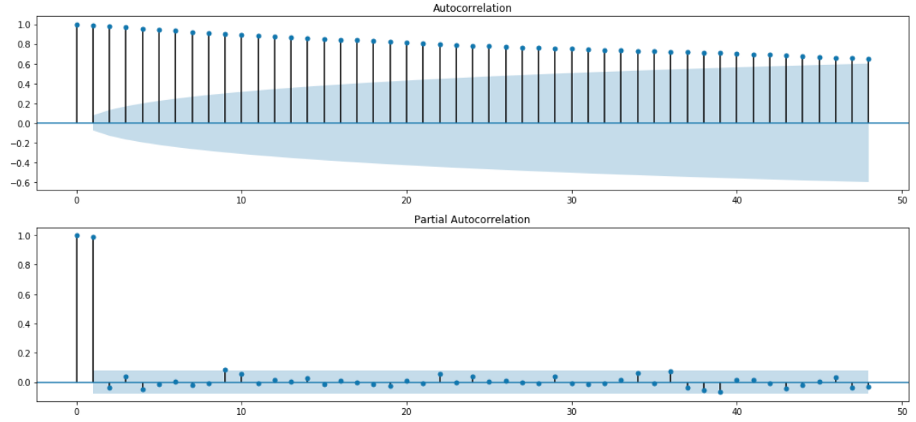
A próxima análise diz respeito a autocorrelação que é a si-milaridade das observações em função do espaço de tempo entre elas, é importante encontrar padrões que se repetem nos dados, esta avaliação aparece na figura 2.

Figura 2

A partir deste ponto inicia-se a preparação dos conjuntos de dados de treinamento e teste de acordo com os requerimentos do modelo que implementa a LSTM, isto inclui o reshape dos dados, um ajuste de escala dos mesmos usando MinMaxScaler em função da sensibilidade da LSTM a escala dos dados e a aplicaçao de uma função lookback que define quantos elementos vamos retroagir para calular o valor corrente, ou seja, se o parâmetro look\_back for igual a 1 vamos prever o valor de t baseado no valor anterior t-1.  
O autor relata vários experimentos com modelos diferentes e conclui que os melhores resultados foram obtidos utilizando uma rede LSTM de duas camadas, cuja configuração são duas camadas de 256 unidades conectadas a uma camada densa com 1 neurônio, utilizando o Adam optimizer e MSE, definiram a parada antecipada em 20 épocas erecomendam o uso de 100 épocas e butch\_size 16 para o treinamento, outro parâmetro indicado foi o shuffle=false para não misturar dados de séries temporais.

Após essa fase de preparação treina e testa-se a rede neural, os resultados são comparados no gráfico mostrado na figura 3, onde podemos ver que a preda no treinamento e teste foram extremamente baixas, indicando o sucesso do treinamento.

Figura 3

Sabendo que o resultado do teste foi favorável o modelo foi utilizado para prever os valores para o conjunto de testes e avaliar o resultado final, só que para poder avaliar o resultado final é necessário reverter a previsão para a escala original de dados, afim de poder comparar os valores previstos com os valores reais e calcular o RMSE (Root Mean Square Error) que representa um valor absoluto de erro médio em dolar entre os valores previstos e os valores reais. O RMSE calculado resultou 18,724 dólares, ou seja um erro na casa de 2 dezenas para valores oscilando entre 3500,00 e 4800,00 dólares, um resultado bastante promissor. O gráfico apresentado na figura 4 mostra a comparação entre preços reais e prvistos pelo modelo.

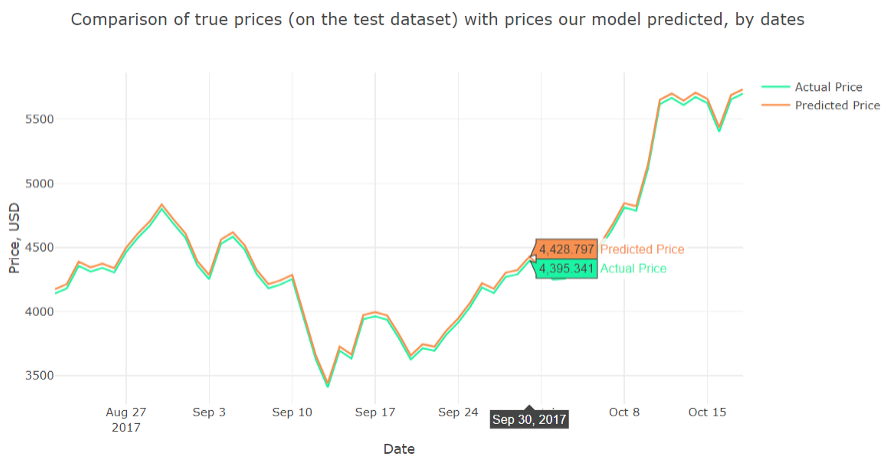


Figura 4

Para tentar melhorar os resultados obtidos o autor propõe dividir os dados em 10 conjuntos difentes de dados de treinamento e teste, esta técnica é conhecida com Ten Fold Cross Validation, onde cada para treinamento/teste é executado em separado e ao final calcula-se o RMSE dos RMSEs de cada par de conjuntos trienamento/teste, este RMSE resultante é subtraído de cada predição do modelo corrente, podendo melhorar a preformance geral da predição.

Para demosntrar esta estratégia o autor decidiu usar um modelo GRU para mostrar o uso de um modelo diferente, para tal foram construidas novas funções para implementar a quebra dos 10 conjuntos de dados e seu processamento em 10 passagens sequenciais culminando com o cálculo do RMSE para as 10 passagens, sua aplicação às previsões, a geração de um gráfico comparando previsto e real e por fim, o cálculo do Symetric Meean Absolute Percentage Error (SMAPE) que mostra percentualmente quão boa é a previsão realizada. O RMSE médio para as 10 iterações resultou em 14,223, abaixo do RMSE de 18,724 obtido no experimento de 1 iteração com LSTM, já o SMAPE calculado resultou em 0,304%. A figura 5 mostra o gráfico comparativo desta previsão.

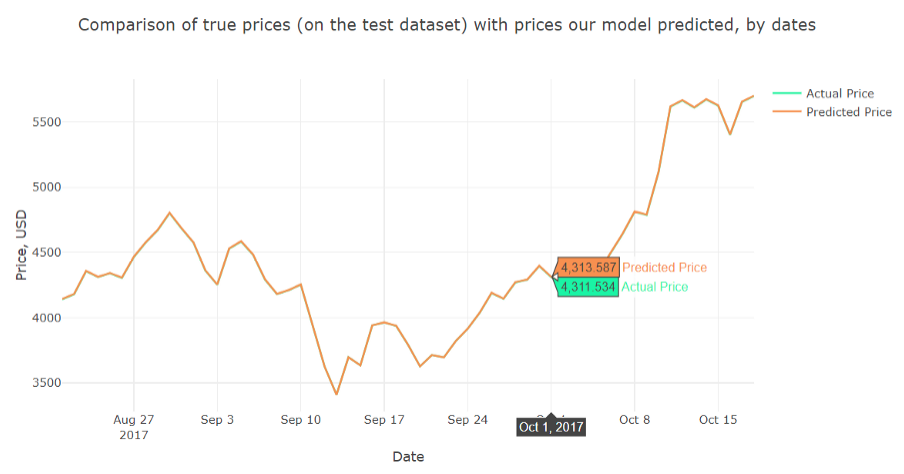


Figura 5

A conclusão natural deste experimento é de que é possível realizar predições extremamente precisas para as cotações futuras de Bitcoin usando um valor médio diário das cotações. No entanto, qual é o valor real de prever o valor médio de uma moeda em um determinado dia futuro?

No meu ponto de vista serve apenas para indicar um mercado com potencial de investimento, caso a previsão aponte um crescimento de valor ou um mercado onde não se deve investir no momento, caso a previsão aponte redução de valores em relação ao valor presente.

3 O novo experimento proposto

Para ter alguma validade real a predição deveria ser capaz de indicar o fluxo de subida e descida da cotação ao longo de pelo menos um dia futuro, desta forma seria possível determinar um valor de compra e venda dentro de um período de 24hs, caracterizando uma operação de day-trade.

A proposta então é mudar a granularidade do experimento de intervalos diários para minutos, o que aumenta a representação de um dia de 1 registro para 1440 registros, os dados existem nesta granularidade, outra mudança é utilizar o valor Mínimo do minuto, o autor fez a média diária sobre a coluna de média ponderada do minuto existente na base original, resta então adaptar o código, aplicar o modelo e avaliar o novo resultado, onde a granularidade será minutos do dia.

3.1 Replicando o experimento original

O primeiro passo foi replicar o experimento original para entender todos os processos envolvidos e checar os resultados, mas, para esta replicação foi necessário usar um novo conjunto de dados, uma atualização do conjunto utilizado no experimento original, que não estava mais disponível, o novo conjunto cobre o período de 2011-12-31 07:52:00 a 2018-06-27 00:00:00.

De posse deste novo conjunto de dados repliquei o experimento, com o mesmo número de dias (660) e coluna sumarizada, porém em um período diferente, 2016-09-05 00:00:00 a 2018-06-27 00:00:00, esta diferença gerou valores não tão promissores e algumas conclusões sobre a eficácio do modelo de predição proposto.

3.1.1 Resultados da replicação do experimento ori-ginal

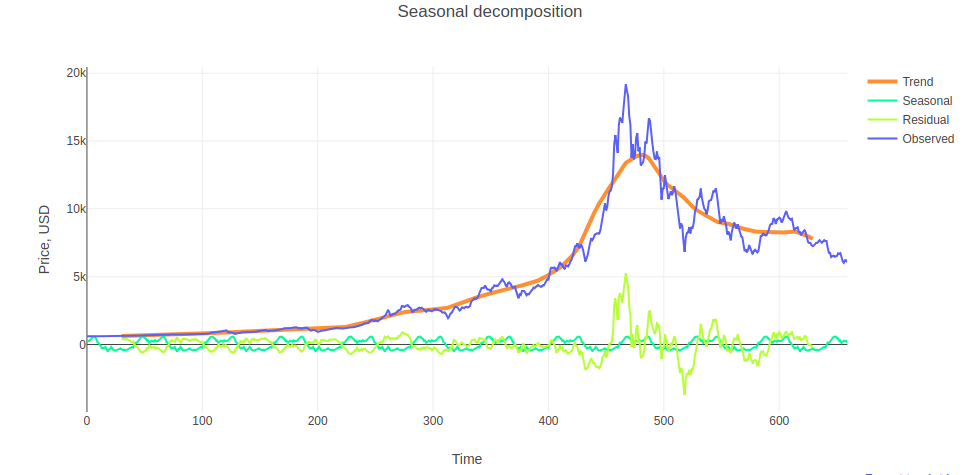
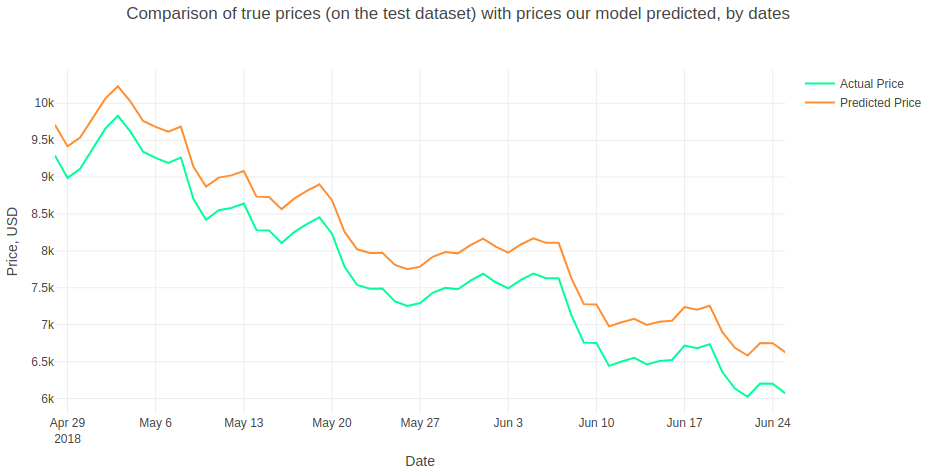
O primeiro resultado que chamou a atenção foi o gráfico de sazonalidade e tendências do novo conjunto de dados, compare a figura 1 e a figura 6, que representam estes fatores para os dois conjuntos respectivamente.

Figura 6



O pico na metade direita do gráfico representa o período de alta desenfreada do Bitcoin em outubro de 2017.

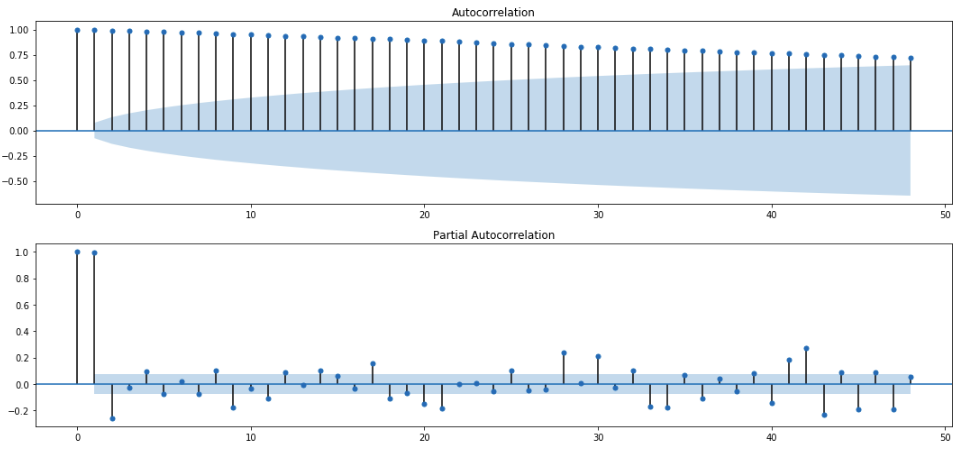
A avaliação da autocorrelação gerou imagens similares ao experimento original, apenas com escala diferente, compare as figuras 2 e 7.

Figura 7

O treinamento foi realizado e o gráfico de perdas mostrou uma pequena distorção com aumento da perda e posterior melhora, no entanto sem atingir a melhor taxa obtida até a 12ª época de treinamento, compare as figuras 3 e 8.



Figura 8

Ao final do treinamento com o modelo LSTM chegamos ao seguinte resultado, um RSME de 480,910 dólares!

Comparando com o RSME do experimento original com valor de 18,724 a acuracidade do modelo foi muito ruim, a figura 9 apresenta o gráfico comparativo desta previsão, que é equivalente a figura 5 do experimento original.

Figura 9

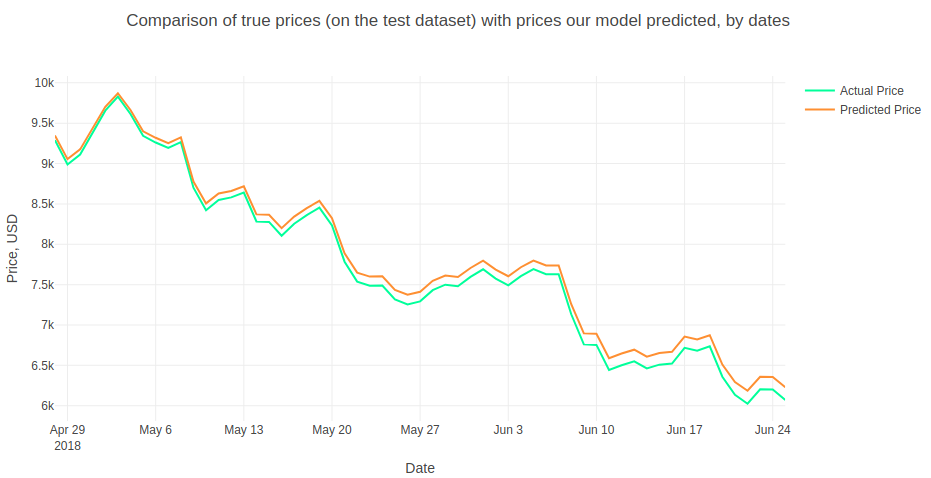
Seguindo o experimento realizei um novo treinamento usando a técnica de Ten Fold Cross Validation usando o modelo GRU cujo resultado final produziu um RSME de 111,340 dólares e um erro SMAPE de 1,448%, a figura 10 mostra o gráfico final de comparação entre a previsão e os valores reais.

Figura 10

3.1.2 Conclusão sobre a replicação do experimento original

Ficou claro que este modelo é muito suscetível a períodos de pico desproporcionais aos valores médios das cotações, no período de treinamento, neste caso, os picos das cotações chegaram a quase 20k dólares, enquanto que os dados do período de teste e predição oscilaram entre 6k e 10k.

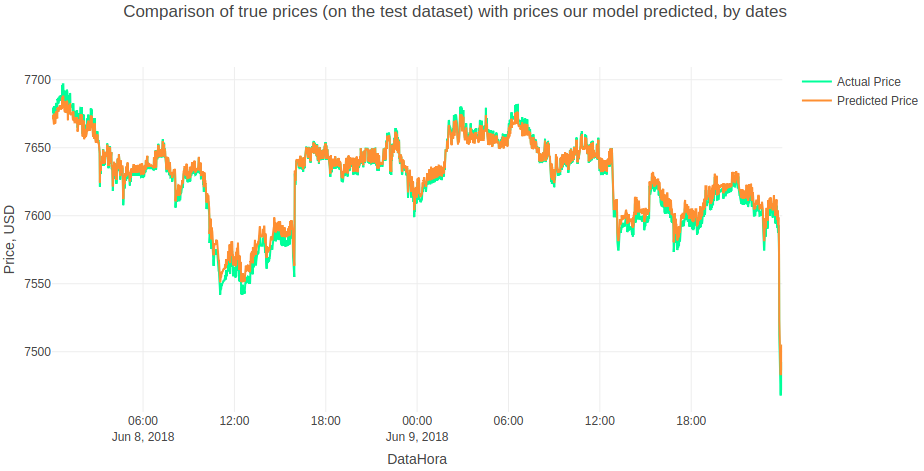
No entanto, a despeito desta fragilidade o modelo ainda pode gerar resultados muito interessantes para períodos sem picos exacerbados como o primeiro semestre de 2018, que registrou uma tendencia linear de baixa, o que no entanto não invalida a ideia de utilizá-lo para predição de períodos diários para suportar o day-trade, bastando para isso ter o cuidado de não incluir períodos tão longos de dados de treinamento e expurgar períodos com grandes distorções.

3.2 Adaptando o experimento a granularidade de horas

A adaptação do experimento pareceu simples a primeia vista, bastava mudar a forma de representar as datas e escolher a coluna VlLow da base original como fonte de dados, de fato estas duas adaptações resolveram a questão da importação dos dados, o próximo passo era definir o tama-nho da base de dados a utilizar e a distribuição dos dados entre treinamento, teste e predição, a ideia inicial era separar 100 dias, treinar com 90 dias, testar com 9 dias e predizer 1 dia, no entanto isto significava processar 144.000 registros e eu não tinha idéia do tempo necessário para realizar este processamento, assim optei por reduzir o treinamento a 6 dias (8640 registros), teste a 2 dias (2880 registros), e predição de 1 dia (1440 registros), totalizando 12960 registros a processar, no período 2018-06-01 00:00:00 a 2018-06-09 23:59:00.

A adaptação seguinte seria a função Look\_Back que define quanto voltamos no passado para prever o futuro, no experimento original este valor era 1 para uma granularidade diária, analogamente deveria ser substituido por 1440, que também representa 1 dia, optei por reduzir a 720, metade do dia, só que não percebi que na chamada da função o valor passado era 1, sobrepondo o 720 default da função, poste-riormente descobri que outros valores geram erro no script python.

Ao rodar o experimento a rede LSTM não apresentou nenhum problema, gerando um RSME de 4,050 dólares, muito menor que no experimento original que atingiu 18,724 dólares, sendo que neste experimento o range de valores de cotação estava entre 7,5k e 7,7k enquanto que no experimento original o range ficou entre 0,5k e 5,7k.

Ao continuar o experimento com a rede GRU, um erro no script associado a troca do valor de look\_back impediu a continuidade do experimento, tentei de várias formas identificar a correção do problema mas não consegui e com a urgência em continuar o experimento decidi retornar o parâmetro Look\_Back a 1 como no experimento original e reiniciar o processo.

A tentativa seguinte de executar todo o experimento obteve sucesso e por isso vou incluir os gráficos resultantes e todos os valores obtidos.

3.3 Experimentos bem sucedidos e resultados

Ao todo foram conduzidos 11 experimentos apenas os mais significativos serão descritos.

3.3.1 Configurando o parâmetro Look\_Back em 1 (Jupyter notebook Exp10Dias\_v04)

Esta foi a primeira execução completa do experimento com granularidade de horas, como primeiro passo realizei a nálize sazonal e de tendência do conjunto de dados, a figura 11 apresenta um resultado extremamente plano.

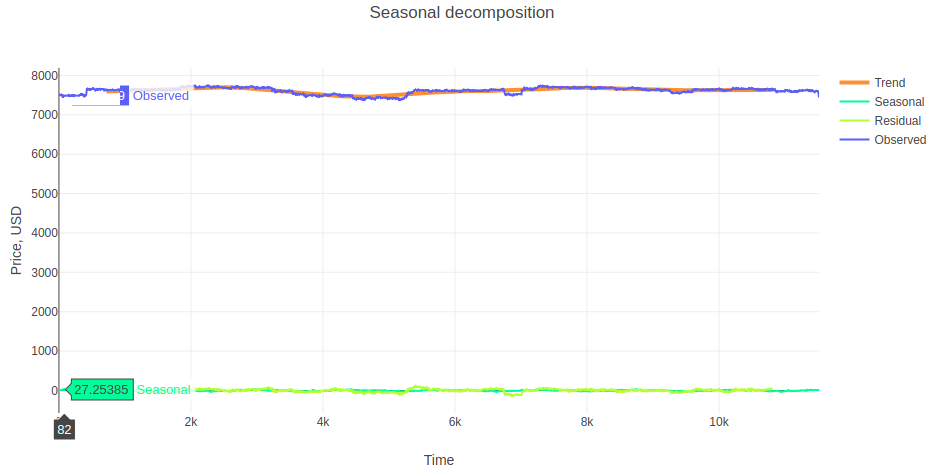


Figura 11

O passo seguinte foi a verificação da correlação de dados que mostrou dados similares ao longo do tempo.

Executando o modelo, avaliamos a perda no treinamento e teste, a figura 12 mostra estas perdas, e chegamos a uma nova previsão com um RSME calculado de 4.084 dólares, este resultado foi muito instigador, mostrando a grande capacidade na predição pelo modelo LSTM, a figura 13 mostra o gráfico comparativo entre valores previstos e reais.

Figura 12

Figura 13

Um fato que chamou a atenção foi a capacidade de previsão do modelo LSTM em acompanhar a forte queda registrada no final do dia 9/6, esta queda inicia-se em torno das 23:37 e segue até as 23:59 quando se encerra o período de predição, acredito que esta capacidade está associada a granularidade escolhida para o experimento.

O passo seguinte foi criar o modelo GRU e realizar o treinamento usando a técnica Ten Fold Cross Validation, o resultado obtido foi decepcionante, o RSME médio ficou em 8,568 dólares, mais que o dobro obtido no passo anterior, neste passo também foi calculado o SMAPE, que ficou em 0,102%, mesmo assim menor que os 0,304% obtidos no experimento original. Acredito que a piora do resultado obtido pelo modelo GRU deva-se a uma má escolha dos parâmetros de tamanho dos dados de treinamento e teste, configurados para 8640 e 2880.

3.3.2 Oitava iteração (Jupyter notebook Exp10Dias\_v08)

Nesta execução adicionei o cálculo do SMAPE ao resultado do modelo LSTM, os resultados foram os seguintes:

Sendo o mesmo conjunto de dados não existe variação nas análises de sazonalidade, tendências e autocorrelação.

O modelo LSTM gerou um RMSE de 4,074 dólares e um SMAPE de 0,043%.

O modelo GRU gerou um RMSE médio de 5,686 dólares um SMAPE de 0,064%.

3.3.3 Nona iteração (Jupyter notebook Exp10Dias\_v09)

O resultados foram os seguintes:

Sendo o mesmo conjunto de dados não existe variação nas análises de sazonalidade, tendências e autocorrelação.

O modelo LSTM gerou um RMSE de 3,951 dólares e um SMAPE de 0,042%.

O modelo GRU gerou um RMSE médio de 6,477 dólares um SMAPE de 0,074%.

3.3.3 Décima segunda iteração (Jupyter notebook Exp10Dias\_v12)

Nesta iteração tentei mudar os parâmetros de treinamento e teste do modelo GRU, inicialmente configurei os parâme-tros de tamanho dos dados de treinamento e teste, com os valores 2592 e 648, executei o treinamento usando a técnica Ten Fold Cross Validation e quando disparei o cálculo do RMSE médio ocorreu um erro informando que a quantidade de elementos de teste era 2879 e a predição apenas 647.

Uma segunda tentativa com treinamento configurado para 4200 e teste para 1296 falhou informando que a quantidade de elementos de teste era 2879 e a predição apenas 1295.

A terceira tentativa restaurou a configuração de teste para 2880 e manteve treino configurado em 4200, a execução obteve sucesso, no entanto o RSME médio saltou para 126,080 dólares e o SMAPE atingiu 1,353%.

Com os resultados acima é fácil avaliar que a aproximação do tamanho do conjunto de treinamento ao tamanho do conjunto de testes prejudicou severamente a acuracidade do modelo, fica, como trabalho futuro estudar formas de melhorar o uso do modelo GRU.

3.3.4 Tabela comparativa de iterações

Após 12 iterações do experimento, algumas com falhas em função de tentativas de alterar parâmetros e outras pelo aprendizado de como funcionava o experimento criei uma tabela comparando os resultados obtidos, a tabela 1 apresenta estes resultados.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Experimento | RMSE LSTM (USD) | SMAPE LSTM | RMSE GRU (USD) | SMAPE GRU |
| Experimento Publicado | 18,724 | -- | 14,223 | 0,304% |
| ExpOri  | 480,910 | -- | 111,340 | 1,448% |
| Exp10dias\_v01 | erro | -- | -- | -- |
| Exp10dias\_v02 | 3,916 | -- | erro | -- |
| Exp10dias\_v03 | 4,050 | -- | erro | -- |
| Exp10dias\_v04 | 4,084 | -- | 8,568 | 0,102% |
| Exp10dias\_v05 | erro | -- | -- | -- |
| Exp10dias\_v06 | **3,805** | -- | **6,288** | 0,072% |
| Exp10dias\_v07 | erro | -- | -- | -- |
| Exp10dias\_v08 | 4,074 | 0,043% | **5,686** | 0,064% |
| Exp10dias\_v09 | 3,951 | 0,042% | 6,477 | 0,074% |
| Exp10dias\_v10 | **3,810** | 0,040% | 6,454 | 0,074% |
| Exp10dias\_v11 | 3,890 | 0,041% | 7,625 | 0,089% |
| Exp10dias\_v12 | 3,943 | 0,042% | 126,080 | 1,353% |

Tabela 1

 Replicação do experimento na granularidade original com dados mais recentes descrito no item 3.1.1

Todos os experimentos de 10 dias foram realizados com granularidade de minutos conforme proposto neste trabalho.

4 Conclusão

O modelo pode ser aplicado a predição da movimentação das cotações diárias de Bitcoin com granularidade de minutos, sendo que nesta granularidade a acertividade do modelo atingiu níveis até 7,6 vezes maiores, se compararmos o SMAPE do experimento publicado (0,304), com o SMAPE obtido pelo modelo LSTM no experimento Exp10dias\_v10 (0,040).

Outro fato que chamou a atenção é que com a granularidade em minutos basta utilizar o modelo LSTM para obter a melhor precisão, o que reduz o custo computacional e o tempo necessário para realizar a previsão.

O modelo LSTM leva 500 segundos para treinar com 100 epocas, é um tempo curto e as diferenças nos resultados entre rodadas distintas sugere que podemos treinar o modelo mais de uma vez até atingir um resultado mais adequado.

Este experimento mostrou uma impressionante melhora na capacidade de predição dos valores minuto a minuto e abre um bom canal de pesquisa para testar também os valores máximos e construir uma predição capaz de informar as faixas de lucro possíveis em um dia futuro de negociação de Bitcoins.

Ficou provado que o modelo é sensível a grandes variações no conjunto de trinamento, no entanto o período de 9 dias se mostrou eficaz na predição de um dia, estes períodos curtos tendem a ser imunes a grandes variações, proporcionando boas predições, novos experimentos com períodos maiores de terinamento podem ser avaliados para checar a possibilidade de melhora na acuracidade das predições.

5 Trabalhos futuros

Modificar modelo para predizer o menor e o maior valor que o Bitcoin terá minuto a minuto durante um dia futuro e avaliar os resultados, esta modificação permitirá sua utilização como guia de compra e venda da moeda e os horários aproximados das operações em um day-trade.

Avaliar mudanças no modelo GRU para tentar melhorar seu desempenho e contribuir para melhora das previsões.

6 Registros do experimento no Github

Todos os scripts python, os dados utilizados no experimento em formato CSV e todos os Jupyter notebooks dos experimentos realizados estão disponíveis no Github no link.

<https://github.com/marcelo-caux/BitCoinPrediction>

No entanto, para poder executar os scripts você precisará criar um ambiente separado e realizar instalações na sua máquina conforme a referencia [7], caso não possua Anaconda instalado na máquina onde deseja executar o experimento siga a referencia [6].

References

[1] https://medium.com/activewizards-machine-learning-company/bitcoin-price-forecasting-with-deep-learning-algorithms-eb578a2387a3

[2] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[3] https://skymind.ai/wiki/lstm

[4] https://www.youtube.com/watch?v=I5v3r9vM1mQ

[5] https://www.pugetsystems.com/labs/hpc/How-to-install-CUDA-9-2-on-Ubuntu-18-04-1184/

[6] https://www.pugetsystems.com/labs/hpc/How-to-Install-Anaconda-Python-and-First-Steps-for-Linux-and-Windows-917/

[7] https://www.pugetsystems.com/labs/hpc/Install-TensorFlow-with-GPU-Support-the-Easy-Way-on-Ubuntu-18-04-without-installing-CUDA-1170/

[8] www.kaggle.com/mczielinski/bitcoin-historical-data/data