Classificador de variação de criptomoedas com base no fechamento do dia anterior

Manoella Rockembach[1] **e Rodrigo Ferraz Souza**[1]

[1] Departamento de Computação – Universidade Federal de Santa Catarina (UFSC)
Araranguá – SC – Brasil

Resumo. *Este relatório apresenta o estudo da utilização de um modelo de rede perceptron de multicamadas para a classificação de variância futura em uma serie temporal, mais especificamente, do preço de criptomoedas. Os resultados obtidos mostram que o modelo desenvolvido é ineficiente em sua execução, pois o modelo de rede neural artificial usado e os fatores levados em consideração, não são suficientes para prever se o valor da criptomoeda no dia seguinte irá subir ou descer.*

1. Problema Abordado

Uma das consequências da ascensão do mercado de criptomoedas nos últimos anos foi o surgimento de Trading Bots focados na negociação desta. Estes bots, que já existiam anteriormente no contexto da bolsa de valores, são redes neurais artificiais treinadas a ajudar o usuário a comprar e vender criptomoedas no momento correto visando aumentar a receita e reduzir perdas e riscos.

Neste relatório testamos a viabilidade da implementação de uma rede perceptron de multicamadas para a classificação da variância futura do valor de diferentes criptomoedas, a partir de seus valores nos dias anteriores.

2. Dados Utilizados

2.1. Obtenção dos Dados

Os dados foram obtidos do site Kaggle, o qual apresenta um repositório extenso de Datasets opensource disponíveis a comunidade de Data Science.

Mais especificamente, o Dataset utilizado para o treinamento da rede neural foi criado por Sudalai Rajkumar [Rajkumar 2021]. Nele contém os dados publicados na coinmarketcap [CoinMarketCap 2021] de 23 criptomoedas desde sua criação até o dia 06 de julho de 2021.

2.2. Feature Engineering

Os dados fornecidos em cada um dos arquivos CSV eram:

1. Date : Data da observação
2. Name : Nome da moeda
3. Symbol : Simbolo da moeda (Ex: BTC)
4. Open : Preço de abertura do dia em questão

[0]Matrícula Manoella: 19102193
[0]Matrícula Rodrigo: 19103563

5. High : Maior preço do dia em questão
6. Low : Menor preço do dia em questão
7. Close : Preço de fechamento do dia em questão
8. Volume : Volume de transações no dia em questão, em dólar
9. Market Cap : Capitalização de Mercado, em dólar

Name	Symbol	Date	High	Low	Open	Close	Volume	Marketcap
XRP	XRP	2019-01-01 23:59:59	0.364771	0.350402	0.352512	0.364771	4.493476e+08	1.488050e+10
XRP	XRP	2019-01-02 23:59:59	0.378021	0.359574	0.365675	0.375243	5.432167e+08	1.530769e+10
XRP	XRP	2019-01-03 23:59:59	0.374505	0.357675	0.374505	0.360224	4.388738e+08	1.469501e+10
XRP	XRP	2019-01-04 23:59:59	0.364642	0.352785	0.359753	0.356747	4.506339e+08	1.455320e+10
XRP	XRP	2019-01-05 23:59:59	0.361069	0.353987	0.356347	0.355275	4.520902e+08	1.449313e+10

Figura 1. Tabela Ripple

Destas informações apenas Market Cap, Name e Symbol não foram utilizados.

2.3. Transformações nos Dados

Fizemos 3 alterações na tabela:

1. Adicionamos 3 colunas de Variância
    - Com relação ao dia anterior, para mostrar para a rede a tendencia do ultimo dia
    - Com relação à semana anterior, com o mesmo motivo do subitem acima
    - Com relação ao dia seguinte, para que fosse utilizado como saída esperada para o treinamento da rede
2. Adicionamos 2 Colunas, evidenciando o mês e o ano
3. Removemos as 7 primeiras linhas e a ultima, pois nelas não seriam possíveis as operações explicitadas no item 1

   Como resultado, obtivemos a seguinte tabela:

Name	Symbol	Date	High	Low	Open	Close	Volume	Year	Month	Variation Next Day	Variation Yesterday	Variation Last Week
Bitcoin	BTC	2013-05-06 23:59:59	124.663002	106.639999	115.980003	112.300003	0.0	2013	5	1	1	1
Bitcoin	BTC	2013-05-07 23:59:59	113.444000	97.699997	112.250000	111.500000	0.0	2013	5	0	1	1
Bitcoin	BTC	2013-05-08 23:59:59	115.779999	109.599998	109.599998	113.566002	0.0	2013	5	1	0	1
Bitcoin	BTC	2013-05-09 23:59:59	113.459999	109.260002	113.199997	112.669998	0.0	2013	5	0	1	0
Bitcoin	BTC	2013-05-10 23:59:59	122.000000	111.551003	112.799004	117.199997	0.0	2013	5	1	0	0
Bitcoin	BTC	2013-05-11 23:59:59	118.679001	113.010002	117.699997	115.242996	0.0	2013	5	1	1	0
Bitcoin	BTC	2013-05-12 23:59:59	117.448997	113.434998	115.639999	115.000000	0.0	2013	5	0	1	1
Bitcoin	BTC	2013-05-13 23:59:59	118.698997	114.500000	114.820000	117.980003	0.0	2013	5	1	0	0
Bitcoin	BTC	2013-05-14 23:59:59	119.800003	110.250000	117.980003	111.500000	0.0	2013	5	0	1	2
Bitcoin	BTC	2013-05-15 23:59:59	115.809998	103.500000	111.400002	114.220001	0.0	2013	5	0	0	0

Figura 2. Tabela Ripple Processada

para o encoding da variação foi-se utilizado o seguinte padrão

- 0: Subiu
- 1: Desceu
- 2: Manteve o mesmo valor

3. Solução do Problema

3.1. Criação da Rede

Para tentar solucionar o problema em questão foi utilizada uma Rede Neural Perceptron do Tipo Multi Camada, desenvolvida em cima do framework Keras do TensorFlow, usando 4 Camadas escondidas do tipo dense Layer (Totalmente conectada). A partir do que foi narrado no tópico anterior selecionamos 9 colunas para serem a entrada da rede, sendo elas:

- Year
- Month
- Open
- High
- Low
- Close
- Volume
- Variation Yesterday
- Variation Last Week

Portanto, a rede utilizada ficou com o seguinte aspecto:

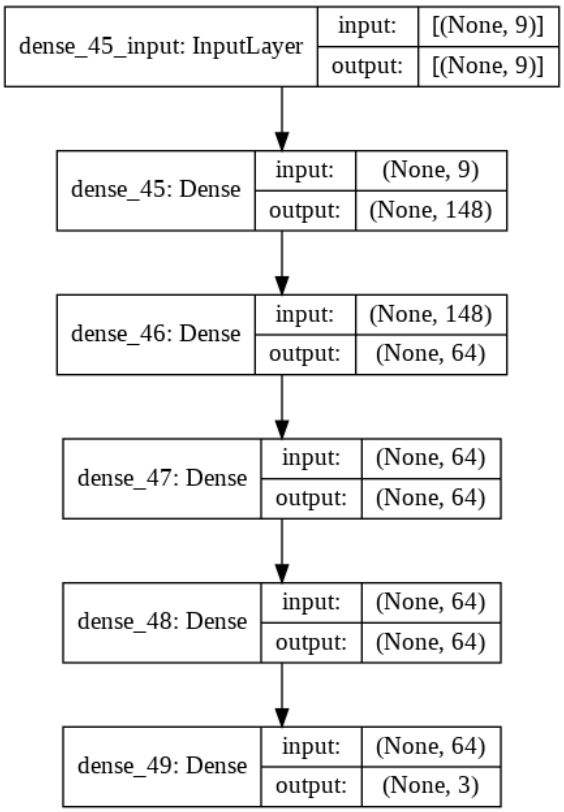

Figura 3. Estrutura das Camadas da Rede Neural

```
model = tf.keras.models.Sequential([
  tf.keras.layers.Dense(148, activation='sigmoid',input_shape=X_train
    [0].shape),
```

```
  tf.keras.layers.Dense(64, activation='tanh'),

  tf.keras.layers.Dense(64, activation='relu'),

  tf.keras.layers.Dense(64, activation='selu'),


  tf.keras.layers.Dense(3, activation='softmax')
])

model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.1),
    loss="sparse_categorical_crossentropy",
    metrics=["mse","mae","accuracy"],
)
```

Listing 1. Construção do Modelo

As funções de ativação foram selecionadas a partir das disponibilizadas na documentação da biblioteca [Keras 2021b] assim como a função de erros (loss) [Keras 2021c].

3.2. Treinamento

Para o treinamento os dados de treino e teste foram separados a uma proporção de 90% para treino e 10% para testes, utilizando o dataset do Ripple (XRP). Portanto a rede foi exposta, durante o treinamento, a 2684 entradas. Para fazer esta separação foi utilizada a função:

```
from sklearn.model_selection import train_test_split
```

Apos separar os dados de treino, apresentamos eles à rede através do método .fit() do keras durante 500 épocas. Contudo, durante o desenvolvimento do trabalho, percebemos que o modelo tendia a estagnar sua loss e acurácia, logo, foram adicionados callbacks[1] para vigiar essas métricas para que o Learning Rate fosse diminuído a um fator de 0.9 caso fosse identificado um plateau.

Também salvamos[2] o modelo no drive sincronizado com o google colaboratory para facilitar os testes que descreveremos em 3.3

```
epochs = 500
callbacks = [
    tf.keras.callbacks.ReduceLROnPlateau(
        monitor="accuracy", factor=0.9, patience=25, min_lr=0.0000001,
    verbose = 1
    ),
    tf.keras.callbacks.ReduceLROnPlateau(
        monitor="loss", factor=0.9, patience=100, min_lr=0.00000001,
    verbose = 1
    )]
hist = model.fit(X_train, y_train,
                 epochs=epochs,
                callbacks=callbacks)
```

Listing 2. Treino do modelo

[1] [Keras 2021a]
[2] [TensorFlow 2021]

Entretanto, no fim do treino, o resultado não foi nada animador, tendo cerca de 54% de precisão no conjunto de treino. conforme mostra o gráfico da Figura 4

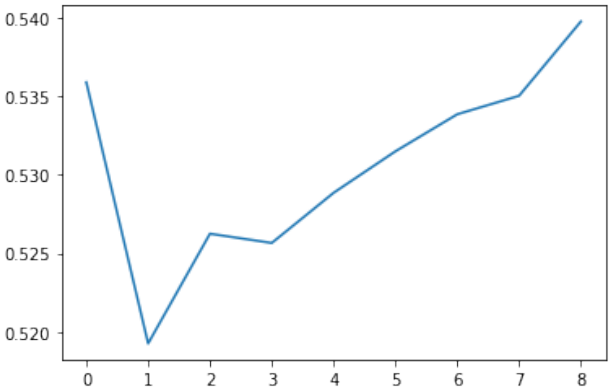

Figura 4. Acurácia do treino 50x

3.3. Avaliação dos Resultados Obtidos

Outras métricas também foram avaliadas, tais como a Loss, Mean Absolute Error e Mean Squared Error, como mostra a Figura 5

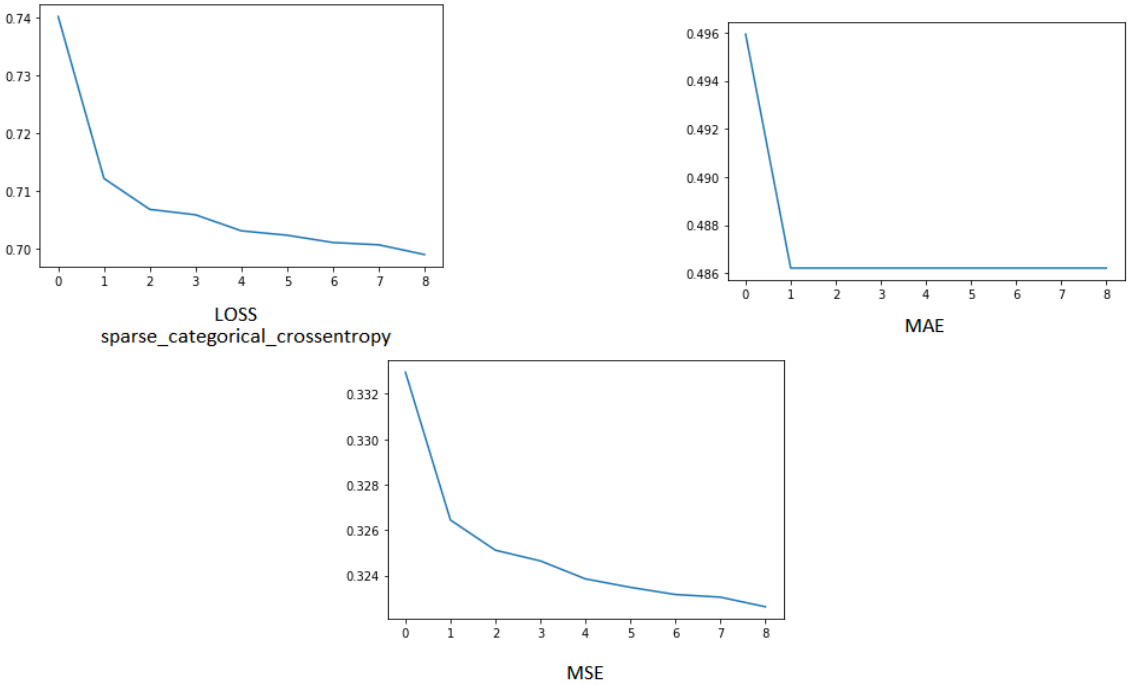

Figura 5. Outras métricas

Para avaliar mais a fundo o comportamento do modelo ainda executamos outros 2 testes em cima do conjunto de testes.

- Uma validação simples com o model.evaluate() do Tensorflow, que retornou o seguinte resultado:
  loss: 0.7121 - mse: 0.3378 - mae: 0.4983 - accuracy: 0.5050

- Uma comparação entre o resultado da rede contra um gerador de números pseudo aleatórios
  - A primeira comparação foi com o gerador sendo mapeado apenas para 0 e 1 (Subir ou descer) pois a presença de casos em que o preço se manteve o mesmo de um dia para o outro é irrisória. O resultado é como mostra a Figura 6
  - A segunda comparação é feita mapeando para 2 (Manteve o preço) também. O resultado é apresentado na Figura 6

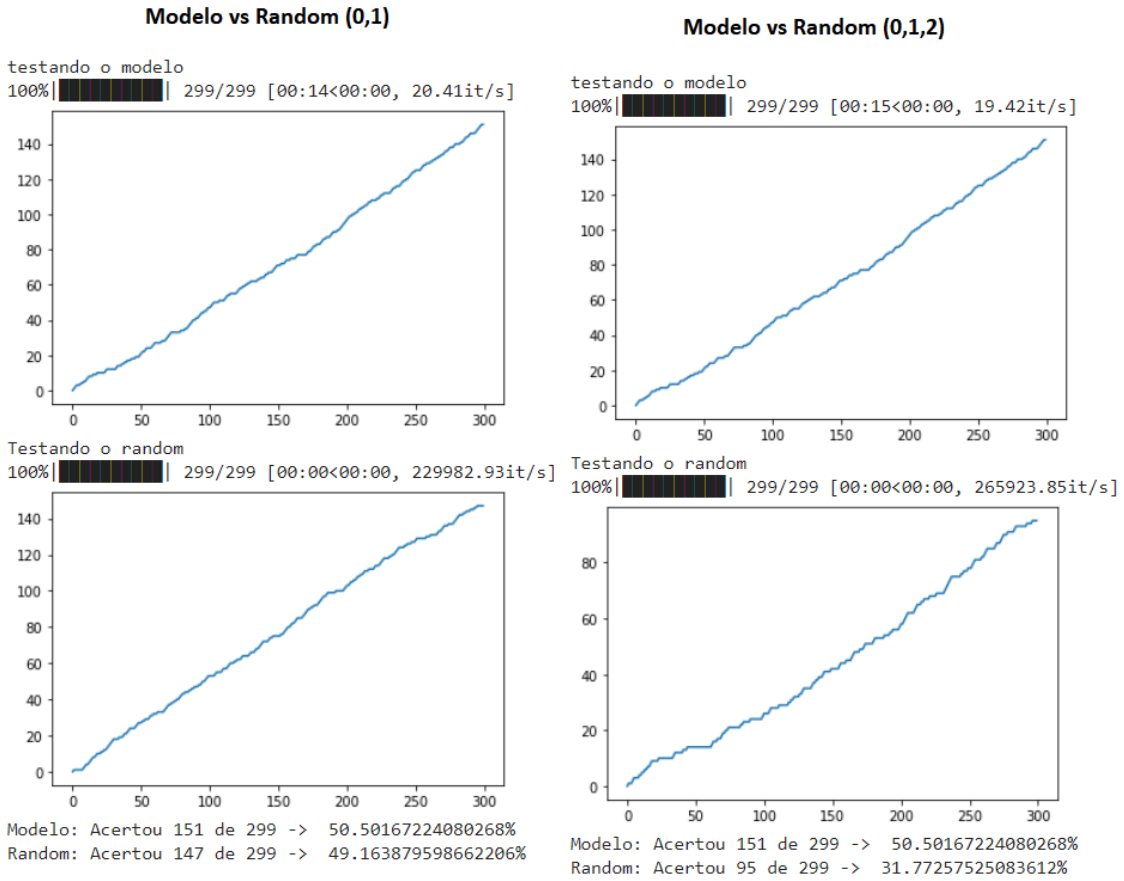

Figura 6. Modelo x Random

A partir da execução dos testes descritos anteriormente foi detectado que a rede convergiu para retornar os **exatos** mesmos valores de ativação para cada categoria, independentemente da entrada.

Como se pode ver na imagem abaixo, o resultado do treino descrito acima foi uma rede extremamente otimista, que apenas retorna que os valores vão subir, pois desta forma, acertaria o máximo de vezes com o mínimo de esforço.

```python
for i  in (range(len(X_test))):
  (entry,expected) = (X_test[i],y_test[i])
  res = model.predict(np.array([ entry]))[0]

  if labels[res.argmax()] == labels[expected]:
    print(f'{res} -> The model is {int(res[res.argmax()]*10000)/100}% sure that the price will go {labels[res.argmax()]} ')
```

```
[0.54898506 0.4498911  0.00112388] -> The model is 54.89% sure that the price will go up
[0.54898506 0.4498911  0.00112388] -> The model is 54.89% sure that the price will go up
[0.54898506 0.4498911  0.00112388] -> The model is 54.89% sure that the price will go up
[0.54898506 0.4498911  0.00112388] -> The model is 54.89% sure that the price will go up
[0.54898506 0.4498911  0.00112388] -> The model is 54.89% sure that the price will go up
[0.54898506 0.4498911  0.00112388] -> The model is 54.89% sure that the price will go up
[0.54898506 0.4498911  0.00112388] -> The model is 54.89% sure that the price will go up
[0.54898506 0.4498911  0.00112388] -> The model is 54.89% sure that the price will go up
[0.54898506 0.4498911  0.00112388] -> The model is 54.89% sure that the price will go up
[0.54898506 0.4498911  0.00112388] -> The model is 54.89% sure that the price will go up
[0.54898506 0.4498911  0.00112388] -> The model is 54.89% sure that the price will go up
[0.54898506 0.4498911  0.00112388] -> The model is 54.89% sure that the price will go up
[0.54898506 0.4498911  0.00112388] -> The model is 54.89% sure that the price will go up
[0.54898506 0.4498911  0.00112388] -> The model is 54.89% sure that the price will go up
[0.54898506 0.4498911  0.00112388] -> The model is 54.89% sure that the price will go up
[0.54898506 0.4498911  0.00112388] -> The model is 54.89% sure that the price will go up
```

Figura 7. Modelo Otimista

4. Conclusões

A rede neural desenvolvida e apresentada neste relatório teve resultados que deixaram a desejar. A arquitetura da Rede Neural Densa convencional implementada e o tratamento dos dados do Dataset original foram os fatores que mais influenciaram os resultados obtidos.

A quantidade extensiva de dados, coletados por vários anos, também interfere. Pois um repentino aumento do valor da moeda em um determinado momento de grande valorização no mercado pode influenciar a rede negativamente nas previsões futuras.

Os melhores resultados, os quais denotavam uma acurácia pouco acima de 56%, foram obtidos quando treinamos o modelo com os dados da criptomoeda Ripple (XRP) a partir de 2020 (19,115% dos dados do dataset). Quando treinada com todos os dados disponíveis da moeda, a acurácia circulava em torno de 50% a 54%, dependendo a arquitetura da rede na qual testávamos.

Outro grande fator foi a arquitetura do modelo. Durante o desenvolvimento utilizamos uma aproximação heurística para identificar a melhor correlação entre camadas, números de neurônios e épocas de treinamento. A combinação que resultou em uma maior acurácia nos testes realizados foi o modelo mostrado na figura 3.

Em conclusão, a rede construída com o tempo para pesquisa e implementação que tivemos não tem uma performance significativamente superior à jogar uma moeda para realizar as previsões. Como mostram as implementações de [Kamalov 2020] e [Spilak 2018], que tiveram resultados mais promissores em um contexto similar, algumas alterações poderiam ser feitas pra melhorar o desempenho da rede: o tratamento e normalização dos dados de input antes de alimentá-los à rede e uma arquitetura que melhor acomode as necessidades do problema. Assim como a analise de diversos métodos de classificação de series temporais discutidos por [Amidon 2020].

Mesmo assim, uma saída mais interessante para lidar com classificações de variação em series temporais seria a utilização de Redes Long Term Short Memory (LSTM) ou Redes Neurais Convolucionais (CNN). Como é demonstrado nos trabalhos referenciados no parágrafo anterior, os quais realizam uma comparação entre a acurácia das redes MLP, LSTM e CNN quanto a previsão de preços futuros.

Referências

Amidon, A. (2020). A brief survey of time series classification algorithms.

CoinMarketCap, T. (2021). Crypto coins prices and data.

Kamalov, F. (2020). Forecasting significant stock price changes using neural networks. *Neural Computing and Applications*, 32(23):17655–17667.

Keras, T. (2021a). Keras documentation: Earlystopping.

Keras, T. (2021b). Keras documentation: Layer activation functions.

Keras, T. (2021c). Keras documentation: Regression losses.

Rajkumar, S. (2021). Cryptocurrency historical prices.

Spilak, B. (2018). Deep neural networks for cryptocurrencies price prediction. Master's thesis.

TensorFlow, T. (2021). Save and load keras models : Tensorflow core.