Apresentação de dois modelos para predição de preços de ações, juntamente com dois algoritmos auto-regressivos, para realizar a análise da dinâmica comportamental e predição do preço.

Jimi Togni - RA: 226359.
Professores: Romis Ribeiro de Faissol Attux e Levy Boccato.

Departamento de Engenharia de Computação e Automação Industrial (DCA)
Faculdade de Engenharia Elétrica e de Computação (FEEC)
Universidade Estadual de Campinas (Unicamp)
CEP 13083-852 – Campinas, SP, Brasil

{romis, levy}@dca.fee.unicamp.br
jimitogni@gmail.com

Abstract – The purpose of this work is to explore two different types of models for the prediction of future closing prices in shares of the Brazilian financial market. We looked for two mathematically distinct models in their implementations, precisely so that it is possible to see the resolution of the same problem, using heuristic technique and, previously, mathematical techniques, in both cases, using Neural Networks. Also, two autoregressive algorithms were implemented to analyze the behavior and price dynamics over time and thus, to make a prediction based on the data obtained.

Keywords – Genetic Algorithm, MLP, RNN, LSTM, ARMA, GARCH.

1. Introdução

Este trabalho, tem por finalidade, explorar dois tipos de modelos diferentes para a predição de preços futuros de fechamento em ações do mercado financeiro brasileiro. Buscou-se dois modelos matematicamente distintos em suas implementações, justamente para que seja possível ver a resolução de um mesmo problema, utilizando técnica heurística e, outrora, técnicas matemáticas, em ambos os casos, utilizando Redes Neurais. Também, foram implementados dois algoritmos auto-regressivos, para analisar o comportamento e da dinâmica do preço ao longo do tempo e assim, realizar uma predição com base nos dados obtidos.

O objetivo não é provar que um modelo é melhor que o outro, e sim, explorar as abordagens em sua máxima potencialidade, assim possibilitando, discernimento no momento da escolha do algoritmo de aprendizado que resolve melhor, problemas específicos.

2. Proposta

A princípio, o objetivo desse projeto era criar dois algoritmos de aprendizado de máquina, utilizando dois tipos de abordagens diferentes, as técnicas inicialmente propostas foram: Algoritmo Genético e Regressão Linear com backpropagation, ambas fariam uso de redes MLP. Porém, no decorrer de seu desenvolvimento, percebeu-se que, seria mais interessante e agregador, utilizar ao menos uma forma mais comum e bastante utilizada para predição de valores reais, junto a outra abordagem alternativa, não tão utilizada em predição de valores.

Em vista destes fatores, o objetivo deste projeto foi, criar dois algoritmos de aprendizado de máquina para predição de valores de ações no mercado financeiro no Brasil. O algoritmo potencialmente mais viável e tradicional, que, após feito o levantamento de trabalhos relacionados, foi tomado como o modelo padrão para este trabalho, as redes neurais artificiais do tipo LSTM, uma segunda abordagem, menos tradicional, será o modelo que utiliza aprendizado evolutivo, popularmente conhecido como Algoritmos Genéticos. Buscando assim, demonstrar formas distintas de resolução do mesmo problema, ao final, os resultados irão refletir o modelo mais viável, ou talvez, melhor parametrizado. Para complementar e fundamentar a análise, possibilitando verificar o comportamento de um atributo ao longo do tempo e assim, estimar os próximos valores, foram implementados dois algoritmos auto-regressivos , são eles, (ARMA) - Autoregressive Moving Average [1], e (ARCH) - Autoregressive Conditional Heteroskedasticity [2] . O projeto foi desenvolvido no Jupyter-Notebook, utilizando a linguagem de programação Python,

optou-se por este ambiente pela familiaridade com a linguagem e por sua enorme gama de bibliotecas voltadas para análise de dados e aprendizado de máquina

3. Base de dados

Para a realização dos testes, utilizou-se a base de dados disponibilizada pela empresa Yahoo Finances [fonte]. A empresa selecionada para o download da base de dados foi a Google. A base de dados compreende o período de 27/06/2008 até 26/06/2019 e possui as colunas: 'Date', 'Open', 'High', 'Low', 'Close', 'AdjClose', 'Volume', com um total de 2768 registros, o intervalo entre cada registro é de um dia, ou seja 2768 dias informando o preço da ação em cada um deles, como demonstrado na Tabela 1.

	Date	Open	High	Low	Close	Adj Close	Volume
2765	2019-06-24	1119.609985	1122.000000	1111.010010	1115.520020	1115.520020	1395600
2766	2019-06-25	1112.660034	1114.349976	1083.800049	1086.349976	1086.349976	1546900
2767	2019-06-26	1086.500000	1092.969971	1072.239990	1079.800049	1079.800049	1799100

Tabela 1. Base de dados

Em todos os algoritmos desenvolvidos neste trabalho, a base de dados foi dividida em duas partes, base para treinamento dos algoritmos, com 70% do tamanho total da base e, outra para testes com 30%, o atributo principal para a análise dos dados foi o preço de fechamento da ação, pois acreditasse que, toda a informação gerada pela variância do preço ao longo do dia é refletida no preço de fechamento.

4. Algoritmo Genético com MLP

Como primeira abordagem, utilizou-se as técnicas da computação evolutiva, que também recebe outros nomes como Algoritmos Evolutivos e Algoritmos Genéticos, neste trabalho padronizar-se-a o nome Algoritmos Genéticos.

4.1. Análise de atributos

Para que tornar viável o desenvolvimento deste algoritmo, fez-se necessário um ajuste de pesos em relação aos atributos da base de dados. Foi necessário encontrar qual dos atributos tinha mais relação ou influência sobre o preço de fechamento da ação. Foi utilizado o método de Pearson para encontrar o coeficiente de correlação padrão entre os atributos. Considerando valores próximos a 1 como os com maior correlação com o preço de fechamento e, valores menores que 1, com menor correlação. Para facilitar a busca, foi utilizada a função ‘corr’ para Data Frames da biblioteca Pandas em Python. a pontuação de cada atributo, na Tabela 2.

Close	1.000000
AdjClose	1.000000
Low	0.999873
High	0.999835
Open	0.999681
Volume	-0.610806

Tabela 2. Exemplo de uma tabela.

O mapa de correlação pode ser visto na Figura 1.

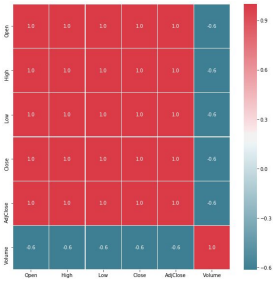

Figura 1. Mapa de correlação de Pearson

4.2. Estratégias Genéticas

Como estratégia e parametrização do algoritmo genético, foram adotados os seguintes passos.

1 - Criação de uma população inicial, com indivíduos aleatórios, foram testados vários modelos, com números de população entre 20 e 100, a rodada de testes que obteve o melhor resultado, com base na função Fitness, que será detalhada mais adiante, foi com o valor de 100 indivíduos para a população inicial. Cada indivíduo virtual possui cromossomos, como na biologia, sendo assim, neste modelo foi definido que, o cromossomo de cada indivíduo irá possuir 5 genes, que são os atributos da base de dados: 'Open', 'High', 'Low', 'AdjClose' e 'Volume'. Observou-se que o algoritmo obteve 3 pontos ótimos locais, nas gerações: 50 obtendo o melhor valor global possível dentro deste teste, depois na geração 100 e, por último na geração 155, como demonstra a Figura 3.

2 - Definição do número de geração, servindo como um dos pontos de parada do algoritmo, auxiliando assim, evitar a ocorrência de sobreajuste. Foram testados os valores: 100, 500 e 1.000 para as gerações, a que obteve melhor resultado nos testes foi quando definida com 1.000 gerações, porém, o algoritmo convergiu antes de completar todas as gerações, pois atingiu outro critério de parada, critérios esses, que serão detalhados mais a frente, isto também pode ser visto na Figura 3.

3 - Função Fitness: a função escolhida aqui foi a inversa do RMSE, baseando-se no trabalho de [3], dada pela Eq. 1:

$$RMSE = \frac{1}{\sqrt{\frac{\sum_{i=1}^{N}(predito(i)-real(i))^2}{2}}} \quad (1)$$

4 - Seleção: o método utilizado para selecionar os indivíduos da população que irão passar para a próxima geração, cruzamento e mutação foi o método da Roleta, que exerce a função de, dar a cada indivíduo um parcela da roleta relativa a seu fitness, assim sendo, um indivíduo com o fitness alto, tem mais chances de ser selecionado para os próximos passos, a probabilidade para cada indivíduo é dada pela Eq. 2.

$$P_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \quad (2)$$

5 - Cruzamento: o modelo de cruzamento escolhido para este trabalho foi o Uniforme, situação que dá a probabilidade de 50% para cada gene ser mudado, ou seja, os genes dos pais são passados uniformemente aos filhos como uma cópia, porém antes de ocorrer a herança, cada gene pode ser escolhido aleatoriamente, com 50% de chances e, se for escolhido, ele receberá os genes do outro pai, isso fica mais claro quando observado na Figura 2. Neste modelo, o número de indivíduos da população serão sempre mantidos com o valor original.

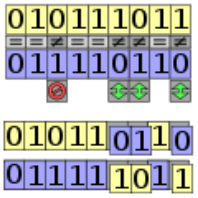

Figura 2. Cruzamento uniforme

6 - Mutação: neste trabalho, apenas 15% dos indivíduos de cada população sofreram mutação, tal valor foi escolhido depois de várias tentativas e foi o que melhor apresentou resultados, também foram testados os valores 5% e 30%.

Após terminar todos os passos citados, uma nova população era gerada, baseada na geração anterior e, com a função de mutação, foi garantida a diversidade de todas as populações. Alguns resultados do melhor teste obtido neste trabalho pode ser visualizado na Figura 3.

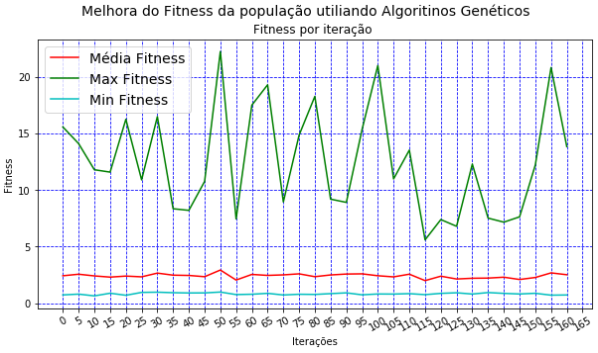

Figura 3. Resultados do modelo proposto com Algoritmo Genético.

Na Figura 3, observa-se que o algoritmo genético convergiu antes do critério de parada com 1.000 gerações, neste trabalho, isso ocorreu porque no modelo, foi implementado mais um critério de parada, que consistia em interromper a criação de populações, se o valor máximo do fitness não aumentasse após terem sido passadas mais de 100 populações criadas, que neste trabalho, ocorreu quando o melhor fitness foi encontrado, na população de número 50 e, seu valor manteve-se como maior por mais de 100 interações, assim, evitando mais uma vez que o algoritmo sofresse sobreajuste.

7 - MLP: utilizou-se um modelo de rede neural artificial do tipo perceptron de múltiplas camadas, para isso, foi utilizada a biblioteca 'sklearn' dela, importamos o modelo MLP Regressor, com as seguintes configurações nos hiperparâmetros, tamanho da camada oculta: 10, função de ativação: relu, solver: lbfgs, max inter: 1, os valores utilizados foram escolhidos com base nos testes realizados no [4].

4.3. Resultados finais com GA

Após o treinamento da rede com os dados separados para treinamento, os resultados em relação a função de erro, que neste trabalho, foi utilizada para medir o fitness de cada indivíduo que foi cerca de 3%, um valor de erro de predição respeitoso, quando comparado a outras formas de séries temporais mais tradicionalmente utilizadas, podem ser visualizadas na Tabela 3.

MSE	1299.1897
RMSE	36.0443
Fitness/erro	0.02774366

Tabela 3. Resultados

Os resultados do valor predito em relação ao valor real, demonstrados em relação a base de dados de testes, podem ser vistos na Figura 4.

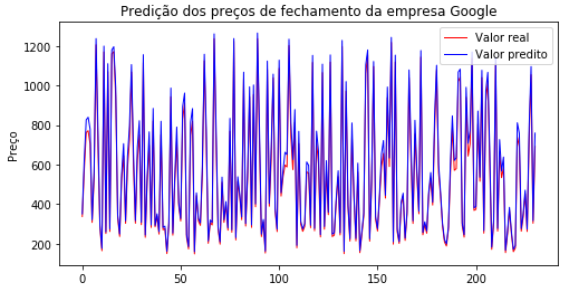

Figura 4. Valor real X Valor predito pelo GA

Todos os hiperparâmetros escolhidos nesta abordagem genética, foram escolhidos com base em testes feitos pelo autor e também, nas referências da disciplina [5].

5. Predição utilizando RNN

Este trabalho traz uma segunda abordagem para predição de valores em séries temporais. O modelo utilizado aqui será o de RNN, mais especificamente, LSTM que são redes neurais com memória de longo prazo. Este modelo não será extremamente detalhado como o modelo abordado na seção 4, pois o objetivo deste trabalho é demonstrar que a computação evolutiva tem muito a contribuir para a área de aprendizado de máquina.

Todos os hiperparâmetros escolhidos para modelar os testes com LSTM foram escolhidos baseados nas observações encontradas nesse material [6]. A base de dados utilizada neste modelo é a mesma citada no item 3 deste trabalho. Os hiperparâmetros passados para o modelo são demonstrados na Figura 5.

Layer (type)	Output Shape	Param #
lstm_13 (LSTM)	(None, 50, 96)	37632
dropout_13 (Dropout)	(None, 50, 96)	0
lstm_14 (LSTM)	(None, 50, 96)	74112
dropout_14 (Dropout)	(None, 50, 96)	0
lstm_15 (LSTM)	(None, 50, 96)	74112
dropout_15 (Dropout)	(None, 50, 96)	0
lstm_16 (LSTM)	(None, 96)	74112
dropout_16 (Dropout)	(None, 96)	0
dense_4 (Dense)	(None, 1)	97

Total params: 260,065
Trainable params: 260,065
Non-trainable params: 0

Figura 5. Descrição do modelo para LSTM

Para o cálculo do erro foi utilizado o erro quadrático médio, a função de otimização foi o ADAM, o algoritmo conseguiu um valor de erro mínimo igual a 0.0010, o que realmente é impressionante, em relação ao erro mínimo encontrado com GA o item 4 deste trabalho. O resultado em relação a base de dados de testes pode ser visto na Figura 6.

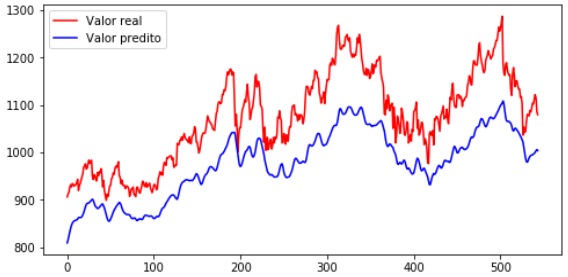

Figura 6. Resultados utilizando RNN - LSTM

6. Algoritmos auto-regressores

Neste item, será demonstrado, duas técnicas utilizadas por algoritmos auto-regressores ou AR, para realizar a predição de valores, na base de dados descrita no item 3 deste trabalho, apenas modificando a estrutura de colunas, que neste modelo, a única coluna usada foi a 'Close', por ser o atributo alvo da predição.

6.1 Modelo ARMA

Na Figura 7, pode-se ver o sumário dos parâmetros utilizados no modelo ARMA e, o gráfico com os resultados da predição dos preços na Figura 7. Vale ressaltar que este algoritmo utiliza o estimador de máxima verossimilhança para, visto em sala de aula, bem como, mesmo com coeficiente e desvio padrão baixo, o modelo não foi capaz de predizer os valores com boa certeza, como podemos ver na Figura 7.

ARMA Model Results

Dep. Variable:	y	No. Observations:	2767
Model:	ARMA(1, 0)	Log Likelihood	7268.575
Method:	css-mle	S.D. of innovations	0.017
Date:	Mon, 01 Jul 2019	AIC	-14531.151
Time:	22:13:05	BIC	-14513.374
Sample:	0	HQIC	-14524.730

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0005	0.000	1.550	0.121	-0.000	0.001
ar.L1.y	-0.0083	0.019	-0.436	0.663	-0.046	0.029

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	-120.6465	+0.0000j	120.6465	0.5000

Figura 7. Detalhes do modelo ARMA

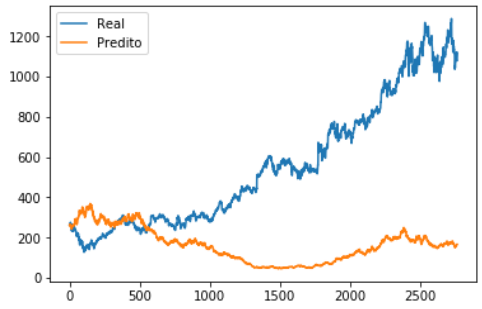

Figura 8. Valores preditos pelo modelo ARMA

6.1 Modelo GARCH

Na Figura 9, pode-se ver o sumário dos parâmetros utilizados no modelo GARCH.

```
                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:                  Close   R-squared:                      -0.000
Mean Model:             Constant Mean   Adj. R-squared:                 -0.000
Vol Model:                      GARCH   Log-Likelihood:               -5220.51
Distribution:                  Normal   AIC:                           10449.0
Method:            Maximum Likelihood   BIC:                           10472.7
                                        No. Observations:                2767
Date:                Mon, Jul 01 2019   Df Residuals:                    2763
Time:                        22:13:13   Df Model:                           4
                            Mean Model
==============================================================================
                 coef    std err          t      P>|t|    95.0% Conf. Int.
------------------------------------------------------------------------------
mu             0.0607  2.997e-02      2.024  4.293e-02 [1.930e-03,  0.119]
                          Volatility Model
==============================================================================
                 coef    std err          t      P>|t|     95.0% Conf. Int.
------------------------------------------------------------------------------
omega          0.1555  7.645e-02      2.034  4.191e-02  [5.695e-03,  0.305]
alpha[1]       0.0966  5.177e-02      1.866  6.210e-02 [-4.885e-03,  0.198]
beta[1]        0.8546  6.272e-02     13.624  2.879e-42    [  0.732,  0.978]
==============================================================================
```

Figura 9. Detalhes do modelo GARCH

O gráfico com os resultados da predição dos preços, na Figura 10. Pode-se observar um grande ganho em relação ao modelo ARMA

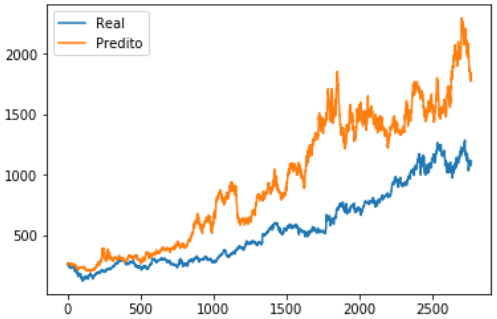

Figura 10. Valores preditos pelo modelo GARCH

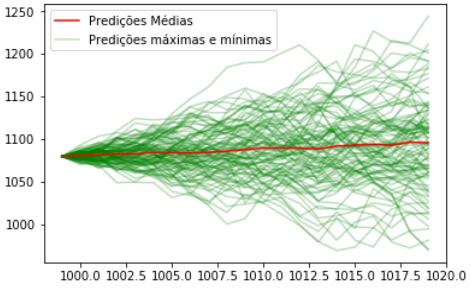

Figura 11. Predição da variação dos valores futuros do preço, apontados pelo modelo GARCH

4. Conclusões

Pode-se observar no desenvolvimento deste trabalho que, mesmo que o GA seja capaz de predizer valores reais em relação ao tempo, muito provavelmente, ele não é a melhor opção para tal finalidade, tendo em vista do desempenho e a facilidade de se implementar um modelo auto-regressivo como o GARCH, que apresentou resultados estatisticamente bons, no mínimo, com uma implementação incrivelmente fácil. O modelo LSTM utilizando RNN também apresentou um resultado bastante satisfatório, e sua implementação é ligeiramente simples. Sugere-se aqui, futuramente, a implementação de RNN utilizando LSTM juntamente com modelos auto-regressivos, para tentar a aproximação de um modelo ideal para a predição de valores reais no espaço temporal.

Após feita uma análise minuciosa nos resultados de todos os algoritmos desenvolvidos neste trabalho, pode-se observar que, existem modelos, algoritmos, frameworks e bibliotecas específicas para problemas também específicos, ou seja, cabe ao pesquisador, selecionar as melhores ferramentas que sejam capazes de resolver da melhor maneira o problema que ele necessita resolver, ao invés de tentar adaptar um modelo que é de sua preferência, mas que não é específico para aquele problema, o pesquisador precisa ter discernimento e senso crítico na escolha do modelo, não só pelo fato de ter melhores resultados, bem como, um custo computacional viável, mas também, ter uma maior confiabilidade nos resultados obtidos em sua pesquisa.

Referências

[1] ARMA
https://scholar.google.com.br/scholar?as_ylo=2019&q=Autoregressive+Moving+Average&hl=pt-BR&as_sdt=0,5&as_vis=1 (acessado em 01/07/2019).

[2] GARCH
https://scholar.google.com.br/scholar?as_ylo=2019&q=Autoregressive+Conditional+Heteroskedasticity+&hl=pt-BR&as_sdt=0,5&as_vis=1 (acessado em 01/07/2019).

[3] Sonal Sable, Ankita Porwal, Upendra Singh, Stock price prediction using genetic algorithms And evolution strategies. ICECA 2017.

[4] EFC3 da disciplina IA006 - 2019

[5] IA707 - 2019
ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ia707_1s11/notas_de_aula/ (acessado em 01/07/2019).

[6] github.com/KindYAK (acessado em 01/07/2019).