Extração de dados financeiros com um web scraper: um estudo sobre a rentabilidade dos dividendos

Extraction of financial data with a web scraper: a study on the profitability of dividends

Dhaniel Nunes Mazini[1]
Renato Cesar Sato[2]

ABSTRACT

The use of web scraping, while respecting data privacy and access rules can be an important tool for obtaining data. However, this process may require adjustments to make the data sense. As an example, in this study we apply this approach in the study of the profitability of dividends of the companies that make up the IBrX 50. Fundamental analysis continues to be an area of fundamental importance in corporate finance, however, the data and information used for this type of study may not be readily available in a single data source. Based on the data extracted it was possible to carry out a multiple regression analysis to verify the profitability of the dividends.

RESUMO

O uso do *web scraping*, desde que respeitando as normas de privacidade dos dados e de acesso podem ser uma ferramenta importante para obtenção de dados. No entanto, esse processo pode necessitar de ajustes para que os dados façam sentido. Como exemplo, nesse estudo aplicamos essa abordagem no estudo da rentabilidade dos dividendos das empresas que compõem o *IBrX 50*. A análise fundamentalista continua sendo uma área de fundamental importância nas finanças corporativas, no entanto, os dados e informações utilizados para esse tipo de estudo podem não estar facilmente disponibilizados em uma única fonte de dados. Com base nos dados extraídos foi possível realizar uma análise de regressão múltipla para verificar a rentabilidade dos dividendos.

I. INTRODUÇÃO

Uma das questões importantes nas finanças corporativas diz respeito à relação entre os dividendos e o crescimento da empresa [1], isto é, altos pagamentos de dividendos implicam em um menor crescimento dos lucros futuros [2][3]. Invertemos essa questão ao analisar o rendimento dos dividendos baseados nos aspectos fundamentais da empresa. O rendimento dos dividendos é um fator importante para determinação do valor da empresa seguindo as hipóteses clássicas das finanças. Dessa forma, os investidores se sentem mais confortáveis ao investir em empresas que apresentam um maior rendimento dos dividendos do que aquelas que não oferecem esta perspectiva de rentabilidade.

Buscando explicar essa questão utilizamos de *web scraping* para a coleta de informações relevantes do site *Yahoo! Finanças*, que nada mais é do que a prática de extrair automaticamente dados específicos de uma determinada página da web. O programa literalmente "raspa" o código fonte do *website* para coletar informações previamente selecionadas [4][5]. Possui utilidade quando necessitamos de um grande volume de dados extraídos recorrentemente, já que realizar todo o processo de extração de forma manual demandaria muito tempo e esforço, o que pode ser resolvido em poucos minutos pelo *script* automatizado.

Ao dissertarmos sobre retirar informações de uma página *web* podemos cair no questionamento sobre a legalidade da ação. No momento não há leis explicitas que proíbem o uso desta técnica, desde que sejam mantidas sigilosas toda e qualquer informação pessoal, busque-se apenas por informações públicas e não descumprir alguma regra que está exposto nos Termos de Uso do *website* [5]. É importante ter um cuidado especial quanto ao número de requisições e acessos que são feitas à aplicação, já que, caso exista uma sobrecarga, ações podem ser tomadas pelo servidor da página, como bloquear o acesso do usuário. Após tomarmos todas essas precauções éticas, podemos partir para o desenvolvimento do *web scraper*.

Devido a facilidade oferecida, escolhemos a linguagem *Python* [6] para a implementação, já que possui uma boa gama de bibliotecas que facilitam a construção de um *web scraper* e a criação de modelos para Inteligência Artificial, como uma futura atualização do atual programa para um "robô" que executa todos os métodos da etapa preditiva utilizando tecnicas de *machine learning*, garantindo, por meio de aprendizado, um melhoramento constante do modelo desenvolvido.

II. PROCESSO DE EXTRAÇÃO DOS DADOS ATRAVÉS DO WEB SCRAPER

A análise do rendimento dos dividendos e suas relações com outros parâmetros só é possível após ter todos os dados e informações relevantes em mãos. Para isso, desenvolvemos um *Web Scraper* (Raspador Web), um programa que realiza a extração automática de dados específicos de uma página

[1] [2] Universidade Federal de São Paulo – São José dos Campos, SP

web, na linguagem *Python* versão 3.6.5 [6] e utilizando como base uma biblioteca chamada *Beautiful Soup* [7] versão 4.4.0.

Essa biblioteca realiza a leitura e a extração de dados de textos *HTML* ou *XML*, permitindo a busca por *strings*, *tags*, *ids*, *classes* e qualquer outro atributo que possa servir de identificação para um elemento [8].

Num primeiro momento é necessário que o código se conecte com a página web escolhida para se fazer a extração. Isso é possível em *Python* através da biblioteca chamada *Requests*, em que seu objeto recebe como parâmetro a *URL* (endereço) do *website* e retorna sua conexão estabelecida, podendo-se realizar a extração de todo o código fonte desta página. Depois dessa etapa, o objeto criado a partir do *Beautiful Soup* recebe todo código e torna-o interpretável, permitindo pesquisa e extração de informações [9].

Para o andamento do projeto nesse momento foram necessárias as seguintes informações das empresas pesquisadas: Capitalização de Mercado; Beta; Índice P/L; LPA (Lucro por Ação); Dividendo Futuro; Rendimentos dos Dividendos; Índice de Payout e Ações em Circulação.

Todos os dados estão disponíveis na área de "Estatísticas" de cada ativo dentro do site *Yahoo! Finanças*. Isso facilitou o trabalho da extração, já que era necessário entrar em somente uma página para cada Ação pesquisada, o que restringe o formato da *URL* para a alteração apenas do *Ticket* desejado. No entanto, é possível uma extração de múltiplas páginas, isto é, obtendo informações em diferentes *sites*.

Aspectos como a garantia da integridade do dado durante o processo de extração e a facilidade de manutenção são primordiais para que esse tipo de metodologia possua validade prática. Para isso é preferível que se faça a busca por atributos estáticos de cada dado requerido, sendo recomendado a busca pelo *id* único da *tag* em que se encontra a informação. Isso traz a segurança de que caso possua alguma alteração na página, tanto de posição quanto do texto de identificação, o dado correto seja extraído do website.

```
<span data-reactid="276">Beta (3A,
mensalmente)</span>
<!-- react-text: 277 -->
<!-- /react-text -->
<!-- react-text: 278 -->
<!-- /react-text -->
<sup aria-label="KS_HELP_SUP_undefined"
data-reactid="279"></sup>
</td>
<td class="Fz(s) Fw(500) Ta(end)" data-
reactid="280">0,31</td>
```

Fig. 1. Trecho do código fonte do site *Yahoo! Finanças* para o *Ticket ABEV3.SA*.

No entanto, apesar de que cada informação possua um atributo estático chamado *data-reactid*, seu valor possui alteração entre cada ativo (ação), conforme ilustrado pelas Figuras 1 e 2, em que são demonstradas como estão representadas, em código fonte, a mesma informação para duas ações diferentes. No caso da *ABEV3.SA* temos que o valor do *Beta* é indicado pelo *data-reactid="280"*, enquanto

```
<span data-reactid="281">Beta (3A,
mensalmente)</span>
<!-- react-text: 282 -->
<!-- /react-text -->
<!-- react-text: 283 -->
<!-- /react-text -->
<sup aria-label="KS_HELP_SUP_undefined"
data-reactid="284"></sup>
</td>
<td class="Fz(s) Fw(500) Ta(end)" data-
reactid="285">3,07</td>
```

Fig. 2. Trecho do código fonte do site *Yahoo! Finanças* para o *Ticket GOLL4.SA*.

para *GOLL4.SA* está como *data-reactid="285"*, tornando o atributo inviável de ser utilizado como base para as pesquisas.

Encontramos a solução ao utilizar o título da informação como meio de se atingir ao dado. Utilizar strings como forma de busca possui o revés de necessitar atualização direta no código cada vez que o texto do *site* é alterado, porém desse formato mantemos o benefício de uma fácil extração, além de garantir a integridade dos dados e sua extração correta.

Medidas de Avaliação	
Capitalização de Mercado (em um dia) [5]	8,7B
Valor da Empresa [3]	15,61B
P/L Passado	N/A
P/L Estimado [1]	N/A
Índice PEG (expectativa de 5 anos) [1]	N/A
Preço/Vendas (ttm)	0,77
Preço/Livro (mrq)	N/A
Valor da Empresa/Receita [3]	1,37
Valor da Empresa/EBITDA [6]	11,23
Destaques de Finanças	
Ano Fiscal	
Fim do Ano Fiscal	30 de dez de 2017

Informações de Negociação	
Histórico de Preço de Ações	
Beta (3A, mensalmente)	3,07
Variação de 52 Semanas [3]	59,14%
Variação de 52 Semanas de S&P 500 [3]	-6,80%
Alta de 52 Semanas [3]	26,13
Baixa de 52 Semanas [3]	9,18
Média Móvel de 50 Dias [3]	22,99
Média Móvel de 200 Dias [3]	16,00
Estatísticas de Ações	
Volume Médio (3 meses) [3]	5,82M
Volume Médio (10 dias) [3]	3,61M
Ações Em Circulação [5]	348,7M

Fig. 3. Disposição dos dados na página do *Yahoo! Finanças*. Exemplo retirado da ação *GOLL4.SA*.

Conforme mostrado na Figura 3, todas as informações da página estão dispostas em estilo de tabela, em que uma coluna representa o título e a segunda o dado. Isto é, são dispostas em seções diferentes, o que não interfere na forma de busca. Para encontrar os dados específicos, armazenamos o título da informação e localizamos a próxima tag *HTML* *<td>* (uma denominação de célula de tabela na linguagem *HTML*), o melhor padrão encontrado para se capturar o dado corretamente. Tanto o título, quanto o dado foram armazenados em uma linha de um *Data Frame*, uma estrutura de dados fornecida pela biblioteca *Pandas* [10], um meio fácil para se manipular dados e transferi-los para um arquivo e que possui suporte a diversas formatações diferentes. Essa é uma biblioteca construída para análise e modelagem de dados e, como possuímos a pretensão de utilizar mais ferramentas de Inteligência Artificial no desenvolvimento do programa, utilizar de *Data Frames* trará uma maior facilidade para manipulação dos dados.

```
def estatisticas_yahoofinancas(ticket, dados_acoes, i):
    url = 'https://br.financas.yahoo.com/quote/'+ticket+'.SA/key-statistics'
    # utilizando Requests para capturar a página
    client = req(url)
    page = client.read()
    client.close()
    # utilizando BeautifulSoup para ler todo o conteúdo HTML da página
    pageSoup = soup(page, "html.parser")
    # realizando a captura de uma informação com seu dado na próxima <td>
    capitalizacaoMercado = pageSoup.find(string = "Capitalização de Mercado (em um dia)")
    dados_acoes.loc[i,[capitalizacaoMercado]] = trata_dados(capitalizacaoMercado.find_next("td", string=True).text)
```

Fig. 4. Trecho do código representando a extração de uma informação, tratamento e armazenamento em um *Data Frame*.

Para termos um dado facilmente manipulável era necessário tratar as informações extraídas, que é uma parte fundamental do processo de análise dos dados. Notamos que tudo o que era retirado do *site* vinha no formato texto, como uma string. Além disso, dependendo da informação, possuem alguns sufixos para classificá-lo, podendo ser um "M" (representando milhões), um "B" (representando bilhões) ou "%" para dados em porcentagem. Por fim, todos os dados decimais possuíam com vírgulas como separadores do número inteiro para o racional. Dados nesses formatos não são manipuláveis, portanto apenas extraí-los não seria suficiente. Para lidar com esse problema os dados foram transformados para forma numérica, permitindo efetuar os cálculos necessários para regressão linear. Esse processo de transformação foi alcançado através de um método que verifica o formato da *string* para trocar "," (virgula) por "." (ponto) em caso de números decimais - necessário para serem manipuláveis por alguma ferramenta ou linguagem de programação -, realiza a exclusão do caractere alfabético e insere-se a quantidade correspondente de zeros - seis para "M" e nove para "B" - ou então, após retirar o caractere especial de porcentagem, realiza a divisão da informação por cem. Após todo o tratamento, o dado é retornado no formato *Decimal*.

A princípio fizemos a escolha de retornar como uma variável *float* (ponto flutuante), porém ocorreram alguns problemas referentes a arredondamento dos números, causado pela própria versão da linguagem *Python* utilizada [11], que representa pontos flutuantes como aproximações do que realmente são. Isso acontece devido ao grande número de bits de precisão que o tipo de dado *float* possui - um total de 53. O uso de tipo *Decimal* foi o suficiente para garantir a exatidão do valor referido. Deixamos claro que para utilizar esse formato é necessário importar sua biblioteca [12] específica, presente no sistema tradicional do *Python*, que representa números em ponto flutuante com uma precisão maior, perfeito para se utilizar com operações financeiras.

Para este início de projeto utilizamos como amostra as empresas listadas no *IBrX 50* (índice Brasil 50), que representa as cinquenta ações mais negociadas no Brasil. Esse índice contém empresas como AmBev, Itaú, Petrobras, Vale, Bradesco, entre diversas outras, que possuem bons atributos para nosso estudo inicial.

Os dados extraídos foram armazenados em um novo *Data Frame*, garantindo a possibilidade de qualquer manipulação, seja salvando-os em um arquivo ou realizando as manipulações no próprio código. Após construído o *scraper*, realizamos a extração automática dos dados das 50 empresas do *IBrX 50* no dia sete de janeiro de 2019, logo após ao término do pregão. Foram armazenados em um único arquivo *csv* (comma-separated values) para facilitar as manipulações e cálculos.

Não podemos deixar de mencionar novamente de que a prática de scraper pode sobrecarregar os servidores de hospedagem dos *sites*, já que são realizadas inúmeras requisições em um espaço de tempo que um humano, usuário convencional, não conseguiria. Realizar o uso indevido pode acarretar em um eventual bloqueio da navegação para, assim, impedir danos na estabilidade do *website*. Tomando isso como base, inserimos um tempo de espera de quatro a sete segundos entre as raspagens, além de escolhermos o horário com um provável fluxo menor de acessos, após o expediente comercial tradicional e do pregão da bolsa de valores brasileira.

III. Modelo de Regressão Linear

Com base nos dados extraídos, um modelo de regressão linear múltipla foi calculado para prever o Rendimento por Ação baseado no Índice do Lucro por Ação, Índice de Payout e Valor da Capitalização de Mercado. Uma equação de regressão significativa foi encontrada (F(3, 33)=5,841836, p<.002572) com um R2 de .346865 e R2 ajustado de .287489.

$$RDA = \beta_0 + \beta_1 LPA + \beta_2 PO + \beta_3 MktCap + \epsilon$$

RDA = Rendimento dos Dividendos
LPA = Índice Lucro por Ação
PO = Índice de Payout
MktCap = Capitalização de Mercado

O rendimento dos dividendos previsto é igual a $0,376891 + 0,140347(LPA) + 6,04816e - 13(PO) + 0,00000(MktCap)$.

Isto é, podemos desprezar o fator da capitalização de mercado como um preditor para o rendimento dos dividendos. Na Tabela 1 esses resultados são novamente apresentados com maiores detalhes.

TABLE I
Resultados da Regressão

	Coeficiente	Erro Padrão	Razão -t	p-valor
Constante	0,376891	0,0690141	5,461	4,73e-06
LPA	0,140347	0,0442888	3,169	0,0033
PO	6,04816e-13	2,03385e-13	2,974	0,0055
MktCap	0,00000	0,00000	3,309	0,0023

Temos então que o rendimento por dividendo aumentou 0,140347 para cada aumento no LPA. Apesar de todos os regressores terem apresentado uma significância estatística tanto o índice de Payout quanto a Capitalização de Mercados prestaram uma fraca contribuição no rendimento dos dividendos. Isso traz uma interessante discussão se as empresas que possuem uma maior capitalização de mercado podem atrair investidores que buscam o rendimento dos investimentos

baseado nos recebimentos de dividendos. Neste momento não podemos ainda oferecer uma conclusão definitiva pois utilizamos um tamanho de amostra pequeno e que deve ser ampliado nos estudos futuro.

IV. Conclusões e Trabalhos Futuros

Neste artigo buscamos implementar um *web scraper* e analisar como o rendimento dos dividendos está relacionado com o lucro por ação, o índice de payout e a capitalização de mercado. Por se tratar de um estudo em andamento a janela de dados foi de apenas um dia, o que sabemos ser uma limitação da qual não podemos traçar conclusões, por isso, nossa principal discussão nesse momento esteve focada no processo automatizado para extração dos dados.

O processo de extração dos dados financeiros automaticamente mostrou-se uma ferramenta de extrema importância auxiliar os estudos financeiros. Apesar de grande parte dos trabalhos que envolvem a automatização dos processos e análise de dados financeiros estarem fortemente focados na análise técnica, a extração dos dados chamados de "fundamentais"contribuem nas discussões das finanças corporativas. A grande vantagem dessa abordagem é que os dados fundamentais não são prontamente disponíveis para uma extração em massa como no caso dos preços das ações. Isso traz um desafio adicional para que estudos que envolvam uma grande quantidade de dados e empresas possam ser realizados de modo automatizado pela linguagem de programação *Python* que vem ganhamos espaço na área dos estudos computacionais em finanças. Neste trabalho tentamos mostrar como realizar essa extração dos dados e como eles podem ser aplicados posteriormente num modelo de regressão linear múltipla. Percebemos até esse momento que o lucro por ação é um preditor significante para o rendimento dos dividendos, enquanto que o índice de payout e a capitalização de mercado apesar de estatisticamente significantes não contribuíram no rendimento dos dividendos da mesma forma.

Em trabalhos futuros estaremos ampliando o tamanho da amostra e aprimorando o modelo de regressão utilizado para abarcar hipóteses e testes adicionais utilizando aprendizado de máquina na etapa preditiva.

Referências

[1] BREALEY, R. A. et al. Principles of corporate finance. McGraw-Hill Education, 2012.

[2] ZHOU, P.; RULAND, W. Dividend payout and future earnings growth. Financial Analysts Journal, v. 62, n. 3, p. 58-69, 2006.

[3] DANG, C.; LI, Z. F.; YANG, C. Measuring firm size in empirical corporate finance. Journal of Banking Finance, v. 86, p. 159-176, 2018. ISSN 0378-4266.

[4] MITCHEL, R. Web Scraping with Python: Collecting Data from the Modern Web. 1st ed. O'Reilly Media, Inc., p.7-9, 2015.

[5] LAWSON, R. Web Scraping with Python. 1st ed. Packt Publishing Ltd., p. 1-2, 2015

[6] Python 3.6.8 documentation. Disponível em: < https://docs.python.org/3.6/ >. Acesso em 10 Dezembro 2018.

[7] RICHARDSON, L. Beautiful soup documentation. 2007. Disponível em: < https://media.readthedocs.org/pdf/beautifulsoup-korean/latest/beautifulsoup-korean.pdf >. Acesso em: 10/12/2018.

[8] Beautiful Soup Documentation. Disponível em: < https://www.crummy.com/software/BeautifulSoup/bs4/doc/ >. Acesso em: 10 Dezembro 2018.

[9] Requests: HTTP for Humans. Disponível em: < http://docs.python-requests.org/en/master/ >. Acesso em: 10 Dezembro 2018.

[10] Python Data Analysis Library. Disponível em: < https://pandas.pydata.org/ >. Acesso em: 10 Dezembro 2018.

[11] The Python Tutorial: Floating Point Arithmetic: Issues and Limitations. Disponível em: < https://docs.python.org/3.6/tutorial/floatingpoint.html >. Acesso em: 10 Dezembro 2018.

[12] The Python Tutorial: Decimal Floating Point Arithmetic. Disponível em: < https://docs.python.org/3.6/tutorial/stdlib2.html#decimal-floating-point-arithmetic >. Acesso em: 10 Dezembro 2018.