UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Análise da rede de produtos comprados em conjunto no comércio eletrônico

Rafael Joseph Pagliuca dos Santos

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP
SÃO CARLOS

Rafael Joseph Pagliuca dos Santos

Análise da rede de produtos comprados em conjunto no comércio eletrônico

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *EXEMPLAR DE DEFESA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Francisco Aparecido Rodrigues

USP – São Carlos
Março de 2019

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

P138a

Pagliuca dos Santos, Rafael Joseph
Análise da rede de produtos comprados em conjunto no comércio eletrônico / Rafael Joseph Pagliuca dos Santos; orientador Francisco Aparecido Rodrigues. -- São Carlos, 2019.
130 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2019.

1. Redes complexas. 2. Comércio eletrônico. 3. Recomendação de produto. 4. Sistemas de recomendação. 5. Análise estatística. I. Aparecido Rodrigues, Francisco, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Rafael Joseph Pagliuca dos Santos

Analysis of the network of products bought together in electronic commerce

Dissertation submitted to the Institute of Mathematics and Computer Sciences – ICMC-USP – in accordance with the requirements of the Computer and Mathematical Sciences Graduate Program, for the degree of Master in Science. *EXAMINATION BOARD PRESENTATION COPY*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Francisco Aparecido Rodrigues

USP – São Carlos
March 2019

À minha família.

Especialmente à Sofia, que logo deixará o útero e passará a desfrutar de todas as alegrias que este mundo pode oferecer; e à sua mãe, minha esposa, e meu amor, Aline, já que este trabalho não teria sido concluído não fosse seu incessante apoio e a sua abdicação, solidária, a dezenas de noites, finais de semana e feriados. Nada é em vão.

Ao povo brasileiro: trabalhador, sofrido e merecedor de muito mais do que tem.

AGRADECIMENTOS

Ao meu orientador Francisco Aparecido Rodrigues, que me mostrou *para onde* e *como*.

Ao professor André Carlos Ponce de Leon Ferreira de Carvalho, que me introduziu ao universo do aprendizado de máquina.

Ao professor José Alberto Cuminato, que me ensinou que a matemática pode ser complexa e simples ao mesmo tempo.

À minha colega Caroline Lourenço Alves, pelo apoio e revisão minuciosa deste trabalho.

E a todos os outros pesquisadores, professores e funcionários do ICMC.

“We are all connected;
To each other, biologically.
To the earth, chemically.
To the rest of the universe atomically.”
(Neil deGrasse Tyson)

RESUMO

PAGLIUCA DOS SANTOS, R. J. **Análise da rede de produtos comprados em conjunto no comércio eletrônico**. 2019. 130 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2019.

Este trabalho aborda as áreas de teoria dos grafos, sistemas de recomendação, e comércio eletrônico, que já foram tema de diversas publicações ao longo das últimas décadas. Entretanto, o estudo da importância da utilização de medidas de centralidade de redes como atributos preditivos de modelos de aprendizado de máquina é um assunto que ainda não foi explorado pela literatura. Neste trabalho, além de relatarmos resultados que sugerem que essas medidas de centralidade podem aumentar a precisão dos modelos preditivos, também apresentamos os principais conceitos teóricos de redes complexas, como tipos de redes, caracterização, métricas de distância, além de propriedades de redes reais. Também apresentamos as ferramentas e metodologia utilizadas para o desenvolvimento de um *webcrawler* próprio, software necessário para a construção da rede de produtos comprados em conjunto no comércio eletrônico. Modelos de aprendizado de máquina foram treinados utilizando a base de produtos obtida pelo *webcrawler*, possibilitando a obtenção de modelos preditivos de estimativa de preços de produtos, e de previsão de probabilidade de ligação entre produtos da rede. A performance dos modelos preditivos obtidos são apresentadas.

Palavras-chave: Redes complexas, Comércio eletrônico, Recomendação de produto, Análise estatística, Webcrawler.

ABSTRACT

PAGLIUCA DOS SANTOS, R. J. **Analysis of the network of products bought together in electronic commerce**. 2019. 130 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2019.

This work approaches areas such as graph theory, recommendation systems, and electronic commerce, which have been chosen as topics for several publications over the last decades. Although, studying the importance of using network centrality measures as predictive features within machine learning models is a topic which was not yet explored on literature. In this work, besides reporting results which suggest that those centrality measures can increase the precision of predictive models, we also present the main theoretical concepts of complex networks, such as network types, characterization, distance metrics, besides some properties of real networks. We also present the tools and methodology used on the development of our own webcrawler, a software required for the generation of the network of products bought together in the electronic commerce. Machine learning models were trained using the product database obtained using the webcrawler, allowing the achievement of predictive models for product price estimation, and also link prediction between products of the network. The performance of the predictive models are also presented.

Keywords: Complex networks, Ecommerce, Product recommendation, Statistical analysis, Webcrawling.

LISTA DE ILUSTRAÇÕES

LISTA DE ARQUIVOS DE TEXTO

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
AUC	*Area Under Curve*
DTG	*Directed Trust Graph*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
ROC	*Receiver Operating Characteristic*
URL	*Unified Resource Location*

LISTA DE SÍMBOLOS

N — Conjunto de nós

L — Conjunto de arestas

f — Função

$\mathbb{A}$ — Matriz adjacência

A_{ij} — Elemento de posição i, j da matriz $\mathbb{A}$

K_i — Grau do nó i

N — Quantidade de nós em uma rede

M — Quantidade de arestas em uma rede

Cc — Closeness centrality

d_{ij} — Menor distância entre os nós i e j de uma rede

B — Betweenness centrality

SUMÁRIO

CAPÍTULO

1

INTRODUÇÃO

Este trabalho aborda as áreas de teoria dos grafos, sistemas de recomendação, e comércio eletrônico, que desde o final da década de 90, com a proposição dos modelos de rede *small-world* (WATTS; STROGATZ, 1998) e lei de potência (BARABÁSI; ALBERT, 1999), passaram a ter interfaces robustas para se relacionar de forma mais consistente.

Existem diversos trabalhos sobre recomendações em plataformas de comércio eletrônico (Capítulo 2), que em sua maioria fazem cálculo de similaridade entre entidades (compradores e produtos, geralmente) para recomendar novos produtos a um potencial comprador, em geral utilizando dados crus das transações comerciais de uma plataforma de comércio eletrônico.

Entretanto, no melhor do nosso conhecimento, nenhum trabalho anterior recriou a rede de produtos comprados em conjunto de uma plataforma de comércio eletrônico, a partir de dados públicos, com o objetivo específico de obter propriedades dessa rede para utilizá-las como atributos em algoritmos de aprendizado de máquina.

Dentre os trabalhos que utilizaram a base de dados de produtos comprados em conjunto da plataforma de comércio eletrônico Amazon, um deles investigou *motifs* (SRIVASTAVA, 2010), e o outro utilizou a base para avaliar algoritmos de detecção de comunidades (YANG; LESKOVEC, 2012), objetivos diferentes dos nossos.

No nosso caso específico, criamos uma rede de produtos comprados em conjunto a partir de dados obtidos de uma grande plataforma de comércio eletrônico, e calculamos algumas medidas dessa rede, com o objetivo de validar nossa hipótese de que métricas de rede, por si só, possuem informação suficiente para possibilitar o treinamento de modelos de aprendizado de máquina.

1.1 Objetivos

Neste trabalho, temos 6 objetivos, enumerados a seguir, ordenados de forma crescente por importância:

1. Validação da hipótese de que a utilização dos atributos de métricas de redes, unicamente, são suficientes para o treinamento de modelos preditivos de precificação e de previsão de ligação entre nós, cuja eficiência supera o *baseline*.
2. Validação da hipótese de que produtos melhores avaliados são *hubs* da rede.
3. Desenvolvimento de um *webcrawler* próprio.
4. Análise estatística da rede, calculando métricas de redes e identificando correlações entre os atributos dos nós.
5. Construção da rede de produtos comprados em conjunto da plataforma de comércio eletrônico.
6. Utilização do *webcrawler* próprio para a obtenção de dados de produtos comprados em conjunto a partir de uma grande plataforma de comércio eletrônico.

1.2 Contribuições

Ao cumprir os objetivos listados acima, contribuímos principalmente para a área de estudos de ciência de redes, e para a indústria do comércio eletrônico:

1. Confirmamos a hipótese de que a utilização exclusiva dos atributos de métricas de rede são suficientes para o treinamento de modelos de aprendizado de máquina com eficiência superior à do *baseline*.
2. Identificamos que não existe correlação positiva entre a avaliação de clientes e o grau de um nó da rede. Isto é, *hubs* da rede não são, necessariamente, melhores avaliados. Por outro lado, uma correlação confirmada é a de que *hubs* da rede são melhores posicionados no *ranking* de produtos mais vendidos.
3. Desenvolvemos e disponibilizamos no formato de código livre um sistema *webcrawler* capaz de coletar dados de até 1000 produtos por hora, a partir de uma grande plataforma de comércio eletrônico.

A importância de se utilizar métricas de rede como atributos preditivos de modelos de aprendizado de máquina de ligação entre nós se dá, em grande parte, pela possível aplicação

desses modelos em algoritmos de recomendação de produtos, resultando em soluções com alto valor comercial e científico.

Com a publicação desses resultados, novos questionamentos surgiram, que devem ser abordados em pesquisas futuras:

- Por que os *hubs* da rede de produtos comprados em conjunto não são melhores avaliados em comparação a produtos periféricos? Uma hipótese a ser testada é a de que produtos menos centrais à rede são avaliados por uma quantidade pequena de usuários, concedendo maior peso a avaliações artificiais enviadas por autores, editoras, e seus conhecidos, efetivamente resultando em uma avaliação enviesada.
- Quais aspectos das métricas de centralidade de rede as tornam tão valiosas para o treinamento de modelos de aprendizado de máquina de previsão de ligação entre nós? Identificamos que utilizar apenas três métricas de rede (grau, *betweenness centrality* e *eigenvector centrality*) é suficiente para obtermos performance equiparável à do treinamento utilizando todos os outros atributos dos nós. Por que isso ocorre?

1.3 Estrutura do trabalho

Este trabalho está estruturado da seguinte forma: no Capítulo 2 revisamos os principais trabalhos científicos das últimas décadas nas áreas de redes complexas, sistemas de recomendação e comércio eletrônico. No Capítulo 3 apresentamos conceitos básicos de redes, entre eles métricas de caracterização e medidas de centralidade. No Capítulo 4 descrevemos a metodologia utilizada neste trabalho, desde o processo de aquisição de dados até as etapas de processamento de dados e treinamento dos modelos preditivos de aprendizado de máquina. No Capítulo 5, relatamos os resultados obtidos em cada uma dessas etapas, e no Capítulo 6 descrevemos áreas de potencial interesse para trabalhos futuros.

CAPÍTULO

2

REVISÃO BIBLIOGRÁFICA

Três áreas de estudos originalmente distantes entre si, teoria dos grafos, sistemas de recomendação e comércio eletrônico, passaram a se aproximar nos últimos anos.

A teoria dos grafos foi consolidada a partir da década de 1930, com trabalhos como de Köni (1936), na área de sociologia. Um dos trabalhos precursores dessa ciência na área de exatas foi Harary (1969a), citando aplicações em física, química, ciências da computação, engenharias, entre outras, devido ao apelo estético e intuitivo de modelar sistemas na forma de grafos. Em outro trabalho desse autor, ele estuda o problema de obtenção do caminho mais curto em grafos com ligações com pesos aleatórios (HARARY, 1969b).

Na década de 1980, filtros colaborativos, um tipo de sistema de recomendação, surgiram pela necessidade de eliminar a desde então crescente quantidade de lixo associado ao volumoso tráfego de informação da internet, resultando, por exemplo, no filtro de spam (JANNACH *et al.*, 2016).

Esses filtros, baseados em similaridade entre perfis de usuários, dependia da implementação de regra de negócio específica por um especialista do domínio (CHEN; LYNCH, 1992). Sendo assim, Chen e Lynch (1992) implementa um sistema genérico de recomendação que não depende do especialista, e sim apenas da base de dados, montando uma rede formada por conceitos (nós) conectados em função de sua relevância. Neste trabalho, o objetivo da construção da rede foi permitir a navegação entre conceitos adjacentes, sem que houvesse utilização de métricas de rede.

Alguns anos mais tarde, diversos sistemas de recomendação como GroupLens, Fab, ReferralWeb, PHOAKS, Siteseer são avaliados e comparados por Resnick e Varian (1997), e alguns modelos são estudados por Breese, Heckerman e Kadie (1998), utilizando principalmente a representação matricial.

Na mesma época, são publicados os primeiros trabalhos teóricos relacionados a comércio

eletrônico. Em um deles, os *logs* de servidores de internet são utilizados para aprimorar a estrutura das páginas de uma loja virtual, utilizando para isso, conceitos estatísticos de suporte e confiança (THEUSINGER; HUBER, 2000). Outros trabalhos similares são desenvolvidos (BRAINERD; BECKER, 2001).

Até então, os sistemas de recomendação não utilizavam propriedades e métricas específicas de redes complexas, utilizando grafos vez ou outra apenas para representação da informação (DESARBO *et al.*, 1992; RUSSELL; KAMAKURA, 1997; ERDEM, 2003). Isso começa a mudar quando, no final do século XX, Watts e Strogatz apresentam o modelo de redes *small-world*, com parâmetro de modelo ajustável entre 0 e 1, que com sua variação permite a transição de uma rede em anel altamente clusterizada para em uma rede aleatória, passando por um estado intermediário de alta clusterização e efeito pequeno mundo (WATTS; STROGATZ, 1998).

No ano seguinte, e utilizando dados similares aos utilizados por Watts e Strogatz (1998), Barabási e Albert observam que redes reais não possuem a distribuição de graus dada pelos modelos já estabelecidos. Com a definição dos conceitos de adição sucessiva de nós e ligação preferencial, obtiveram o modelo de lei de potência (BARABÁSI; ALBERT, 1999).

Diversos trabalhos são publicados nos anos seguintes, vários deles do próprio grupo de Barabási, como resposta ao crescente interesse no tema (DIESTEL, 2000; ALBERT; JEONG; BARABASI, 2000; BIANCONI; BARABÁSI, 2001; BARABÁSI; RAVASZ; VICSEK, 2001; ALBERT; BARABÁSI, 2002; NEWMAN, 2003).

Huang, Chung e Chen (2003) cita que Amazon e eBay já estavam se beneficiando de sistemas de recomendação. Propõe um modelo de grafos com nós sendo usuários e produtos, e as ligações entre eles sendo similaridade e transações comerciais.

No mesmo ano, Mirza, Keller e Ramakrishnan (2003) também apresenta modelo de grafos para recomendação de produtos, e afirma que é o primeiro trabalho a utilizar fórmulas de Newman, Strogatz e Watts (2001) para resolver sistemas de recomendação, desenvolvendo um algoritmo para sugerir ligações entre nós desconectados da rede. O trabalho tem uma visão mais grafo-centrada se comparado aos anteriores.

Em 2006, Oestreicher-Singer e Sundararajan (2006) colheram dados sobre 250.000 produtos da plataforma de comércio eletrônico Amazon, utilizando um *webcrawler* de busca em largura desenvolvido em Java. Mostram que a estrutura da rede afeta os padrões de distribuição de demanda entre as páginas de diferentes produtos, e estimam um modelo econométrico que confirmou a hipótese de que uma rede de produtos com pequena variação na distribuição de grau tende a equalizar a demanda por diferentes produtos.

Em 2008, Newman (2008) mostra a importância da física no avanço dos estudos de redes, indicando que os estudos atuais tendem a se preocupar com métricas e propriedades médias da rede, e não com valores específicos de um nó, como feito no século anterior em estudos sociológicos.

Dois anos depois, Srivastava (2010) estuda redes funcionais (*motifs*), isto é, padrões de sub-redes que se repetem em uma rede maior, utilizando para isso uma base de dados de produtos comprados em conjunto da Amazon.

Outros trabalhos tentaram compreender aspectos até então deixados em segundo plano na área de recomendação de produtos, apresentando conceitos da economia como utilidade, possibilitando recomendar o produto com melhor custo-benefício para o cliente, e não necessariamente os similares (LI; GHOSE; IPEIROTIS, 2011).

Redes complexas também foram utilizadas para estudar a rede de plataformas de comércio eletrônico, e a sua evolução (TIAN; ZHANG; GUAN, 2013; TIAN; ZHANG; GAO, 2015), sugerindo adaptações do tipo *fit-gets-richer* em vez de *rich-gets-richer*.

Nesta década, continuaram os trabalhos de recomendação em plataformas de comércio eletrônico utilizando redes complexas (LI; WU; LAI, 2013; CHEN; WANG, 2013; LI; CHEN, 2013).

Mais recentemente, Zhong *et al.* (2014) afirma que um sistema de recomendação baseado em *Directed Trust Graph* (DTG) é mais eficiente do que um modelo baseado em similaridade entre nós (perfis similares de usuários, por exemplo). Apresenta um novo modelo chamado TFBRA, e testa o modelo na base de dados Epinions. Outros estudos, anteriores e posteriores, também abordaram esse conceito de propagação de confiança entre entidades (COSTAGLIOLA; FUCCELLA; PASCUCCIO, 2014; LI; WU; LAI, 2013; WANG; GUI, 2013; HESS; TITAH; BENLIAN, 2012; WANG *et al.*, 2015).

Yan, Lee e Lee (2015) analisa, por meio de técnicas de análises textuais, diversos artigos relacionados a comércio eletrônico de 2000 a 2013, listando, além de outros resultados, as palavras mais utilizadas nos artigos desse período, dentre elas *internet*, *trust* e *consumer behaviour*.

Recentemente, um novo estudo sobre *combos* de produtos (*bundles*) foi publicado, para sugerir para lojistas produtos que poderiam ser vendidos em conjunto, e ainda o preço ideal desse pacote (BELADEV; ROKACH; SHAPIRA, 2016).

Além disso, uma nova área de estudo tem sido a importância do contexto ante a recomendação. Mostra-se que informações de contexto, como estação do ano, humor do usuário, sua localização, dentre outros, conferem uma recomendação de maior valor para o usuário (JANNACH *et al.*, 2016).

CAPÍTULO

3

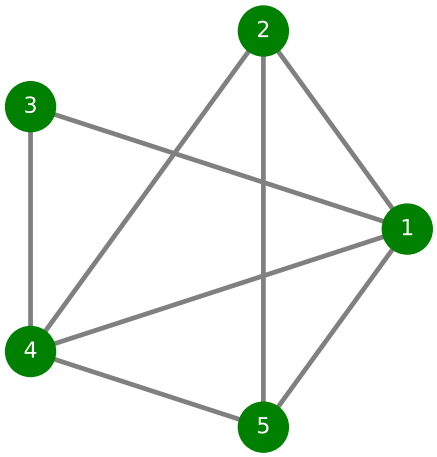
CONCEITOS BÁSICOS

3.1 Grafos

Grafo é a representação matemática de uma rede (DIESTEL, 2000; NEWMAN, 2010). Seu primeiro exemplo de utilização conhecida vem do estudo do problema das Pontes de Königsberg por Eüler no ano de 1735 (METZ *et al.*, 2007; BARABÁSI, 2012), quando esse matemático representou os distritos da cidade de Königsberg por vértices e as pontes que os ligavam por arestas a fim de solucionar uma trívia regional.

Matematicamente, um grafo pode ser definido como uma tupla $G = [N, L, f]$, sendo N o conjunto de nós, L o conjunto de arestas e uma função $f : L \rightarrow N \times N$, que mapeia cada uma das arestas para um par de nós.

Figura 1 – Grafo simples.



Fonte: Elaborada pelo autor.

Uma representação prática de grafos, que será utilizada de forma recorrente nas próximas definições, é uma matriz adjacência $\mathbb{A}_{NxN}$, sendo N o número de nós da rede, com cada elemento definido como:

$$A_{ij} = \begin{cases} 1, & \text{se o nó } i \text{ estiver ligado ao } j, \\ 0, & \text{caso contrário.} \end{cases} \tag{3.1}$$

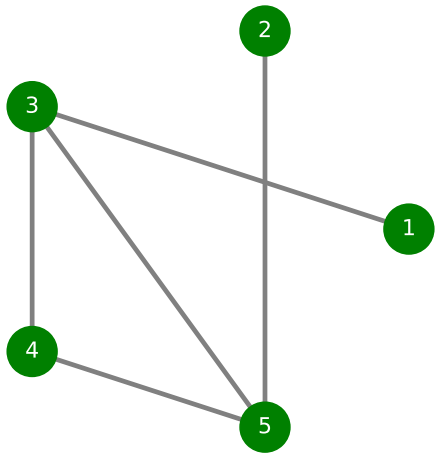
Para exemplificar esse conceito, a matriz adjacência a seguir representa o grafo da Figura 1.

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix} \tag{3.2}$$

3.2 Tipos de redes

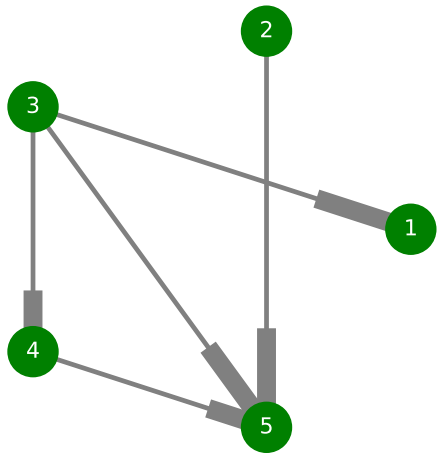
Redes podem ser classificadas por diferentes critérios, sendo alguns dos tipos mais comuns as **redes não dirigidas** (Figura 2), em que as arestas não são direcionadas, ou seja, uma ligação entre os nós i e j é equivalente e sempre coexiste com a ligação entre os nós j e i ($A_{ij} = A_{ji}, \forall i, j$), implicando que a matriz adjacência $\mathbb{A}$ será necessariamente simétrica, $A_{ij} = A_{ji} \Rightarrow \mathbb{A} = \mathbb{A}^T$. Naturalmente, temos também as **redes dirigidas** (Figura 3), nas quais a aresta que conecta dois vértices possui sentido bem definido, sendo que, de forma geral, A_{ij} não é necessariamente igual a A_{ji}.

Figura 2 – Grafo não dirigido.



Fonte: Elaborada pelo autor.

Figura 3 – Grafo dirigido.



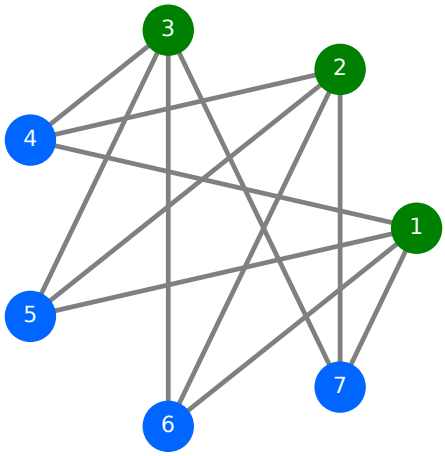
Fonte: Elaborada pelo autor.

Além de tratar ligações apenas como existentes ou inexistentes, podemos atribuir um peso para a ligação, caracterizada por um valor escalar, de forma que o valor 0 representa a ausência de ligação, e um valor diferente de 0 representa a existência de ligação com um determinado peso, generalizando a definição da matriz adjacência (Equação 3.1) e estabelecendo **redes com peso**.

Podemos também classificar redes não somente por suas ligações, mas também pelos atributos de cada nó. No caso de **redes espaciais**, por exemplo, cada nó possui como atributo um vetor de coordenadas.

Outro tipo especial de rede são as **redes bipartidas** (Figura 4), compostas por dois conjuntos disjuntos de nós, onde os vizinhos de um nó pertencente a um dos conjuntos são necessariamente elementos do outro conjunto. Esse tipo de rede é adequado para representar, por exemplo, a ligação entre consumidores e os produtos comprados por eles.

Figura 4 – Grafo bipartido, em que nós do grupo verde se conectam apenas a nós do grupo azul.



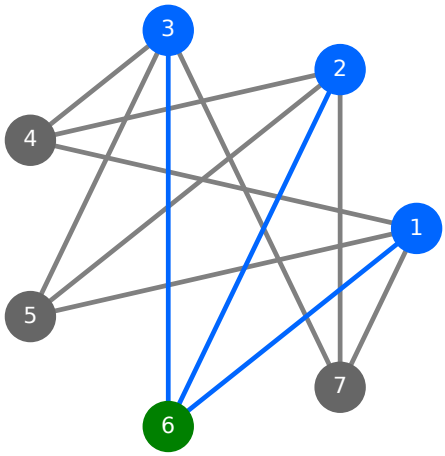
Fonte: Elaborada pelo autor.

3.3 Caracterização

3.3.1 Vizinho de um nó

Vizinhos de um nó são todos os nós adjacentes a ele, conectados por meio de uma aresta. Na Figura 5 são destacados os vizinhos de um nó arbitrário.

Figura 5 – Grafo composto por 7 nós, em que vizinhos do nó verde (6) estão destacados na cor azul (nós 1, 2 e 3).



Fonte: Elaborada pelo autor.

3.3.2 Grau

O grau K_i de um nó i é dado pela quantidade de vizinhos, isto é, pela quantidade de nós adjacentes conectados a ele por uma aresta:

$$K_i = \sum_{j=1}^{N} A_{ij}. \tag{3.3}$$

A distribuição de grau de uma rede é uma função $P(K)$ definida de tal forma que seja igual à probabilidade de que um nó da rede escolhido ao acaso possua grau K (Equação 3.4). A distribuição de grau é uma das formas mais comuns de caracterizar uma rede, visto que redes com diferentes estruturas podem apresentar distribuições de grau distintas:

$$P(K) = \frac{N_K}{N}, \tag{3.4}$$

com N_K o número de nós de grau K, e N a quantidade total de nós da rede.

Tomando o grafo da Figura 2 como exemplo, temos:

$$K_1 = 1 \qquad K_2 = 1 \qquad K_3 = 3 \qquad K_4 = 2 \qquad K_5 = 3$$

$$P(1) = \frac{2}{5} \qquad P(2) = \frac{1}{5} \qquad P(3) = \frac{2}{5}.$$

3.3.3 Momento de ordem m do grau

O momento estatístico de ordem m do grau é definido por

$$E[K^m] = < K^m > = \sum_{K=0}^{\infty} K^m P(K). \tag{3.5}$$

Para o caso particular $m = 2$ define-se a variância, dada pela média da distância quadrática em relação à média:

$$V(X) = E([X - E(X)]^2) = E(K^2) - E(K)^2 = < k^2 > - < k >^2 . \tag{3.6}$$

Considerando uma rede de configuração aleatória (Seção 3.10.1), podemos desenvolver a expressão para $E(k^2)$ para a distribuição de grau do tipo Poisson, concluindo que a variância

do grau dos nós de uma rede aletória é dada por $V(k) =< k >$:

$$E(k^2) =< k^2 >= \sum_{k=0}^{\infty} k^2 \frac{e^{-<k>} < k >^k}{k!}$$

$$= \sum_{k=0}^{\infty} \frac{K.K}{K} \frac{e^{-<k>} < k >^k}{(k-1)!}$$

$$= \sum_{k=1}^{\infty} k \frac{e^{-<k>} < k >^k}{(k-1!}$$

(fazendo $u = k-1$)

$$= \sum_{u=0}^{\infty} (u+1) \frac{e^{-<k>} < k >^{u+1}}{u!}$$

$$=< k > \sum_{u=0}^{\infty} \frac{ue^{-<k>} < k >^u}{u!} + < k > \sum_{u=0}^{\infty} \infty \frac{e^{-<k>} < k >^u}{u!} \tag{3.7}$$

(a primeira somatória é a definição da esperança de u)

(a segunda somatória é a definição de distribuição de probabilidade, portanto igual a 1)

$$=< k > E(u) + < k > 1$$

$$=< k >< k > + < k >$$

$$=< k >^2 + < k > .$$

Retomando o início do desenvolvimento:

$$< k^2 >=< k >^2 + < k >$$

$$\Rightarrow < k >=< k^2 > - < k >^2 =< V > .$$

Uma das implicações desse resultado é que uma rede de configuração aleatória tem baixa probabilidade de apresentar *hubs*, isto é, nós altamente conectados.

3.3.4 Número de arestas de uma rede

Conhecendo-se o grau de cada nó, é possível calcular o número de arestas M de uma rede:

$$M = \frac{\sum K_i}{2}. \tag{3.8}$$

3.4 Lei de potência

Muitas redes reais possuem distribuição de grau da forma de lei de potência:

$$P(K) = CK^{-\gamma} \qquad K > 0,\ \gamma > 0. \tag{3.9}$$

Uma rede que segue essa forma possui um grande número de nós de graus pequenos, e um pequeno número de nós de graus elevados.

A distribuição lei de potência é invariante à escala (*scale-free*), portanto duas redes com mesmo γ terão distribuição de grau similares, diferindo apenas por um fator de escala, sendo que redes reais normalmente satisfazem $2 < \gamma \leq 3$.

Impondo condições para que P(K) seja uma distribuição de probabilidade:

$$\sum_{K=0}^{\infty} P(K) = 1 \qquad P(K) \geq 0, \forall K, \tag{3.10}$$

obtém-se

$$C = \frac{\gamma - 1}{K_{\min}^{-\gamma+1}}. \tag{3.11}$$

3.5 Medidas de complexidade

Uma maneira de medir a complexidade de uma rede é por meio da **Entropia de Shannon**, definida como:

$$H = -\sum_{K=1}^{K_{\text{máx}}} P_K \log P_K. \tag{3.12}$$

A entropia H mede a quantidade de informação intrínseca a um sistema.

Uma alternativa é o **coeficiente de complexidade** Θ, que é definido como:

$$\Theta = \frac{< K^2 >}{< K >}. \tag{3.13}$$

Neste trabalho não calculamos a complexidade da rede obtida, mas tais medidas são apresentadas visando à completude teórica. Essas medidas são consideravelmente relevantes em pesquisas que envolvam a comparação entre mais de uma rede.

3.6 Distâncias

Neste grupo temos medidas que caracterizam a dimensão da rede, tanto em distância entre nós quanto em quantidade de nós.

Distância entre dois vértices é o número de vértices que separam dois vértices i e j. Dentre todos os caminhos possíveis entre dois vértices, a distância do menor deles é definida como **distância geodésica**. A média entre os menores caminhos ligando todos os vértices entre si é o **comprimento médio do menor caminho**. Já o maior dentre todos os menores caminhos define o **diâmetro** da rede.

Conjuntos de nós conexos entre si definem os componentes de uma rede, e, dentre eles, aquele composto pelo maior número de nós é chamado de **maior componente**. O cálculo de algumas propriedades de redes (como exemplo *closeness centrality*, Seção 3.7) necessita que todos os nós da rede estejam conectados entre si, sendo que nesses casos, a utilização do maior componente no lugar da rede inteira pode ser utilizado para estimar a medida desejada.

3.7 Medidas de centralidade

Esta classe de medidas é aplicada a cada um dos nós de uma rede individualmente, para determinar quais nós são mais centrais (relevantes) à rede, considerando diferentes aspectos. A medida mais simples de centralidade pode ser definida como o **grau** de um nó, considerando que os nós com maior número de vizinhos podem ser considerados centrais à rede.

Outra forma de medir a centralidade de um nó é a **closeness centrality**, definida para cada nó i como:

$$Cc(i) = \frac{N}{\sum_j d_{ij}}, \tag{3.14}$$

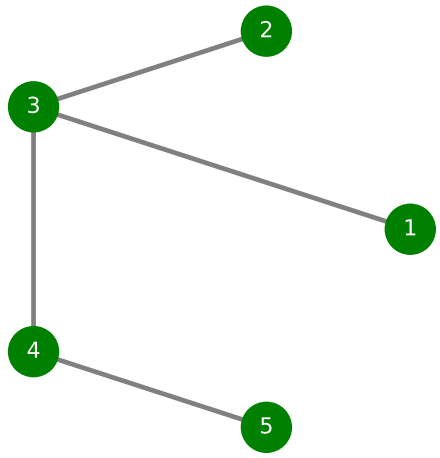
sendo d_{ij} a menor distância entre os nós i e j. Uma desvantagem dessa medida de centralidade é que está limitada a redes de apenas um componente, caso contrário

$$d_{ij} = \infty, \text{ para nós } i \text{ e } j \text{ de componentes diferentes.} \tag{3.15}$$

Como exemplo, para a rede da Figura 6, podemos realizar os seguintes cálculos para obter o valor de *closeness centrality* dos nós, indicando que o nó 3 é o mais central da rede.

$$\begin{aligned} Cc(1) &= \frac{5}{2+1+2+3} = \frac{5}{8} & Cc(2) &= \frac{5}{2+1+2+3} = \frac{5}{8} \\ Cc(3) &= \frac{5}{1+1+1+2} = \frac{5}{5} = 1 & Cc(4) &= \frac{5}{2+2+1+1} = \frac{5}{6} \\ & & Cc(5) &= \frac{5}{3+3+2+1} = \frac{5}{9} \end{aligned} \tag{3.16}$$

Figura 6 – Rede simples utilizada como exemplo para calcular algumas medidas de centralidade.



Fonte: Elaborada pelo autor.

Já a medida de centralidade **betweenness centrality** é definida pelo número de menores caminhos entre todos os nós da rede que passam por um nó i. Há um fator adicional de normalização g_{st} que leva em conta os casos em que há mais de um menor caminho entre dois nós:

$$B(i) = \sum_s \sum_t \frac{n_{st}(i)}{g_{st}}. \tag{3.17}$$

Utilizando a rede da Figura 6 como exemplo, podemos calcular os seguintes valores de *betweenness centrality*. Para isso, primeiro obtemos os menores caminhos:

Tabela 1 – Nós intermediários do menor caminho n_{st}, entre os nós s e t.

Nó s \ Nó t	**1**	**2**	**3**	**4**	**5**
1	N/A	3	-	3	3, 4
2	3	N/A	-	3	3, 4
3	-	-	N/A	-	4
4	3	3	-	N/A	-
5	3, 4	3, 4	4	-	N/A

Com as informações da Tabela 1, o valor de B(i) é apenas o número de repetições que cada nó aparece como intermediário. Essa medida de centralidade aponta o nó 3 como sendo o mais central à rede.

$$\begin{aligned} B(1) = 0 \qquad B(2) = 0 \qquad B(3) = 10 \\ B(4) = 6 \qquad B(5) = 0 \end{aligned} \tag{3.18}$$

Temos também o **eigenvector centrality**, que utiliza o método das potências, considerando a matriz adjacência $\mathbb{A}_{NxN}$ como operador iterativo:

$$X_{t+1} = \mathbb{A}X_t, \quad \text{com } X_0 = [1,1,1,...]_{Nx1}. \tag{3.19}$$

O método das potências determina o autovetor associado ao maior autovalor de uma matriz $\mathbb{A}$. No caso da matriz adjacência, cada coordenada do autovetor obtido representa a centralidade de um nó na rede. Por ser um método iterativo, exige o estabelecimento de critérios de parada. Abaixo estão calculadas as primeiras 5 iterações para a rede da Figura 6.

$$\begin{aligned}
\mathbb{A} &= \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \\
X_0 &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix}^{\mathrm{T}} \\
X_1 = \mathbb{A}X_0 &= \begin{bmatrix} 1 & 1 & 3 & 2 & 1 \end{bmatrix}^{\mathrm{T}} \\
X_2 = \mathbb{A}X_1 &= \begin{bmatrix} 3 & 3 & 4 & 4 & 2 \end{bmatrix}^{\mathrm{T}} \\
X_3 = \mathbb{A}X_2 &= \begin{bmatrix} 4 & 4 & 10 & 6 & 4 \end{bmatrix}^{\mathrm{T}} \\
X_4 = \mathbb{A}X_3 &= \begin{bmatrix} 10 & 10 & 14 & 14 & 6 \end{bmatrix}^{\mathrm{T}} \\
X_5 = \mathbb{A}X_4 &= \begin{bmatrix} 14 & 14 & 34 & 20 & 14 \end{bmatrix}^{\mathrm{T}} \\
&\dots
\end{aligned} \tag{3.20}$$

Assumindo que após essas 5 primeiras iterações tivéssemos atingido o nosso critério de parada, o método indicaria o nó 3 como sendo o mais central à rede, seguido pelo nó 4. Observe que a cada passo a norma do vetor X_t aumenta, o que pode ocasionar, após alguns passos iterativos, o estouro da precisão numérica do processador utilizado para os cálculos. Para contornar esse problema, é prudente normalizar o vetor X_t a cada novo passo, antes de multiplicá-lo pela matriz $\mathbb{A}$.

Por último, temos o **PageRank**, medida que se tornou mundialmente famosa por ser implementada no motor de buscas Google. Esta medida de centralidade se fundamenta em uma caminhada aleatória pelos nós de rede considerando tanto um ajuste estocástico como um ajuste ergódico, para sanar eventos de nós ou classes (grupos de nós) absorventes.

Para calcular essa medida de centralidade, primeiramente definimos a matriz de probabilidade de transição $\mathbb{P}$ como sendo a matriz adjacência normalizada linha a linha pelo grau K_i do nó i.

$$P_{ij} = \frac{A_{ij}}{K_i} \tag{3.21}$$

No caso de redes dirigidas, como por exemplo a *web*, um nó pode não possuir nenhuma ligação de saída ($K_{i,\text{out}} = 0$). Nesses casos, fazemos o ajuste estocástico:

$$\text{se } K_{i,\text{out}} = 0 \Rightarrow P_{ij} = \frac{1}{N}. \tag{3.22}$$

Além disso, aplica-se o ajuste ergódico para evitar classes absorventes:

$$\mathbb{P} = \alpha\mathbb{P} + (1-\alpha)\frac{1}{N}\mathbb{1}_{NxN}. \tag{3.23}$$

3.8 Estrutura de comunidades

Comunidades são caracterizadas por nós densamente conectados entre si, mas esparsamente conectados ao restante da rede.

Um dos principais desafios é conseguir detectar a existência de comunidades em uma rede sem nenhum conhecimento a priori, nem mesmo a quantidade de comunidades existentes. Os principais grupos de métodos de detecção são o **divisivo**, em que remove-se arestas da rede original de forma a separar diferentes grupos de nós; o **aglomerativo**, como por exemplo o algoritmo Ravasz (BARABÁSI, 2012), em que os nós originais da rede iniciam-se desativados, e são reativados um a um, de forma a separar diferentes grupos de nós; os **espectrais**, em que comunidades são detectadas por meio da determinação de autovalores e autovetores das matrizes da rede (adjacência, Laplaciano, modularidade, entre outras); e os **locais**, em que particiona-se a vizinhança de um único vértice. Neste trabalho não pretendemos identificar estruturas de comunidades na rede obtida, não obstante tal conceito foi superficialmente apresentado visando à completude teórica.

3.9 Padrão de mistura

Diferentes redes potencialmente possuem diferentes **padrões de mistura**. Em uma rede arbirtrária, os *hubs* podem estar conectados preferencialmente a outros *hubs* (homofilia), mas também há exemplos de redes onde esse tipo de conexão é bastante improvável. As diferentes distribuições de probabilidade de ligação entre nós de mesmo grau dão origem aos múltiplos possíveis padrões de mistura entre os nós de uma rede. **Correlação de graus** é um caso particular de padrão de mistura, quando nós de um grau arbitrário tendem a se conectar a outros nós de mesmo grau.

3.10 Modelos de redes

O desenvolvimento da teoria de redes está intimamente relacionado com a qualidade dos modelos disponíveis para estudá-las. Dito isso, listamos a seguir os principais modelos que permitiram a obtenção de propriedades valiosas nas últimas décadas.

3.10.1 Modelo de rede aleatório

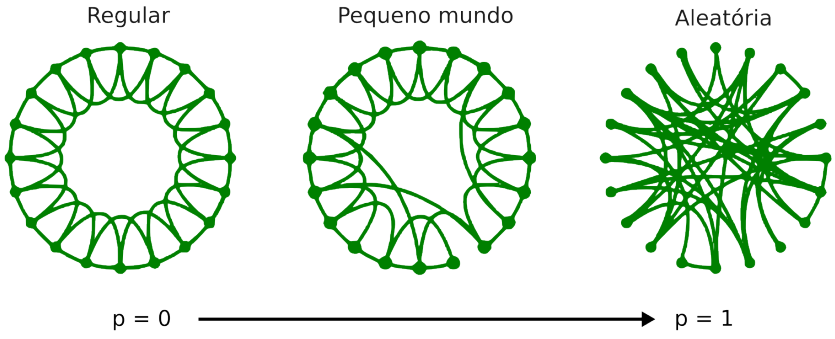
Modelo de rede que desconsidera qualquer preferência de atribuição de vértices entre cada nó. Um modelo popular dessa categoria é o de Erdős-Rényi, que considera uma probabilidade constante p para a atribuição de cada vértice da rede.

3.10.2 Modelo Small-World (Watts-Strogatz)

Modelo proposto em 1998 por Watts e Strogatz que permite a construção de redes com efeito *small-world* (pequeno mundo), mas cuja distribuição de grau não segue a lei de potência.

A construção de uma rede inicia-se com um anel composto de M nós, com cada nó conectado aos seus m vizinhos do sentido horário, e também aos seus m vizinhos do sentido anti-horário. Em seguida, cada aresta da direção horária é reconectada com probabilidade p (WATTS; STROGATZ, 1998).

Figura 7 – Transição de rede em anel para *small-world*, e depois para rede aleatória.



Fonte: Adaptada de Watts e Strogatz (1998).

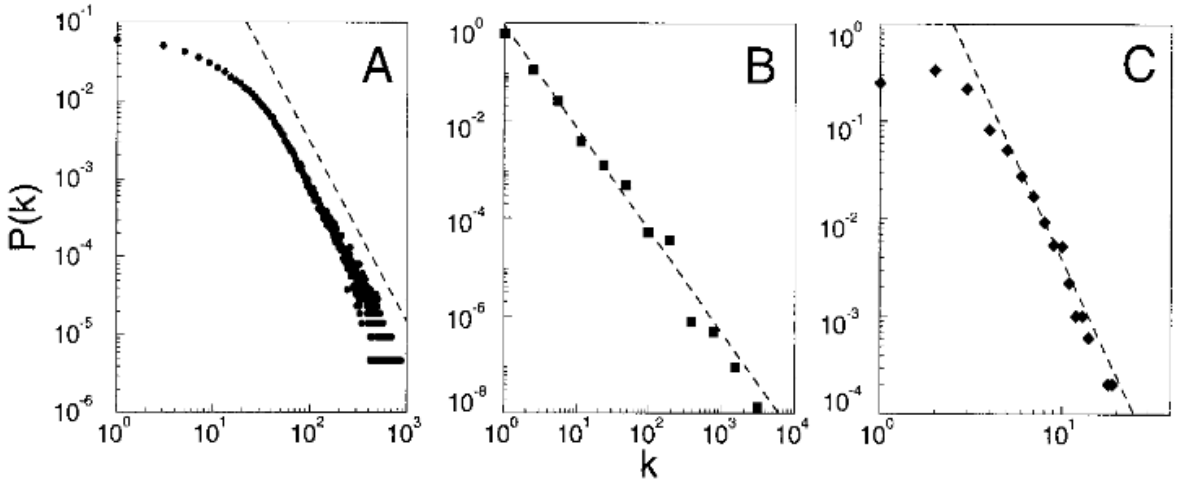
3.10.3 Modelo Lei de Potência (Barabási-Albert)

Modelo proposto em 1999 para construir redes que seguem a lei de potência (Seção 3.4). Neste modelo, uma rede é construída gradualmente, de um em um nó, respeitando-se a regra da ligação preferencial, que veio a ser comparada ao lema *rico fica mais rico* (BARABÁSI; ALBERT, 1999). Esse modelo foi fundamentado em duas regras principais:

- Adição sucessiva de nós;

- Ligação preferencial entre nós.

Figura 8 – Distribuição de grau para três redes distintas, mostrando o comportamento típico de lei de potência.



Fonte: Barabási e Albert (1999).

3.10.4 Modelo de configuração

Modelo da década de 70 que permite criar redes a partir de uma sequência de graus pré-determinada, utilizando para isso *stubs*, isto é, meia-arestas que são conectadas duas a duas. A construção de uma rede utilizando este modelo é realizada em três passos:

- Passo 1: Definir a quantidade de nós na rede e a distribuição de grau arbitrária desejada (exemplo na Tabela 2).

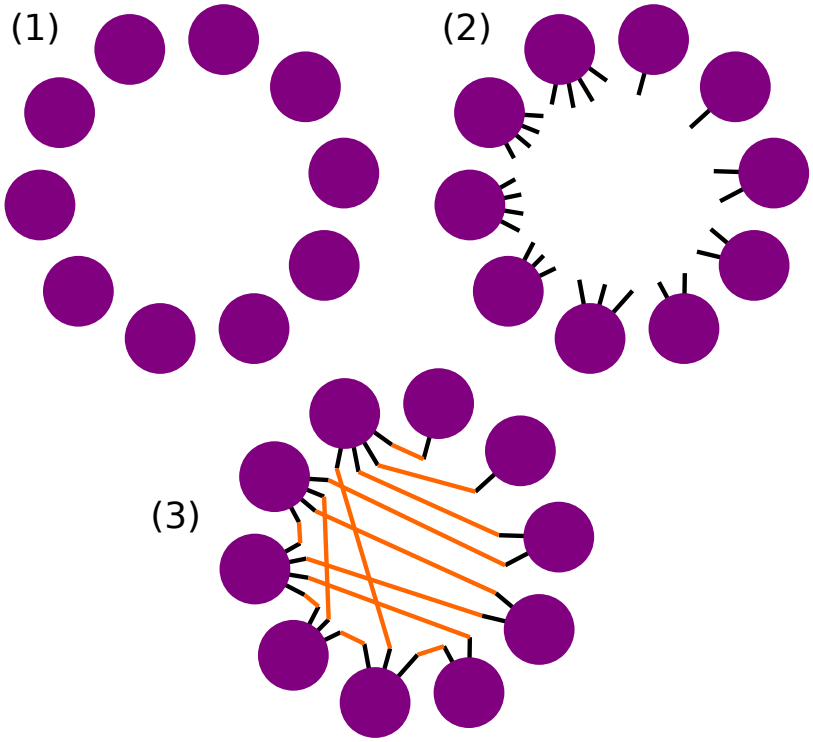
- Passo 2: Dispor os N nós da rede, juntamente com seus *stubs* (vide exemplo na Figura 9-2, com $N = 10$ e seus 26 *stubs*).

- Passo 3: Conectar cada um dos *stubs* entre si (Figura 9-3).

K	Quantidade de nós	Total de stubs
1	2	2
2	3	6
3	2	6
4	3	12

Tabela 2 – Distribuição de grau arbitrária para exemplificar a utilização do modelo de configuração.

Figura 9 – Criação de rede com distribuição de grau arbitrária. Na etapa (1) dispomos os nós da rede (N arbitrário), em seguida, na etapa (2), dispomos os *stubs* (respeitando a distribuição de grau desejada). Por último, na etapa (3), conectamos todos os *stubs* entre si.



Fonte: Elaborada pelo autor.

3.11 Propriedades de redes reais

A partir da década de 1990, avanços tecnológicos permitiram que diversas redes reais (Web, Internet, etc.) fossem estudadas, e, com isso, algumas propriedades foram empiricamente constatadas (NEWMAN, 2003; METZ *et al.*, 2007). A constatação dessas propriedades, listadas abaixo, motivou pesquisadores como Watts, Strogatz, Barabási e Albert a propor novos modelos de criação de redes (vide Seção 3.10).

- **Efeito pequeno mundo (*small-world*).** A distância média entre diferentes nós de uma rede não é proporcional ao número de nós, e sim proporcional ao seu logaritmo.

- **Transição de fase.** Acima de um grau médio crítico, uma rede passa de comunidades isoladas para um grande *cluster* onde praticamente todos os nós estão conectados.

- **Distribuição de grau.** Em redes reais, a distribuição de graus k costuma obedecer à Lei de Potência, isto é, $p_k \sim k^{-\gamma}$, sendo γ uma constante (ver Seção 3.4).

- **Resiliência.** Redes reais são robustas a falhas aleatórias, já que, estatisticamente, é extremamente improvável que diversos *hubs* falhem simultaneamente, pois a fração de tais vértices em uma rede é muito menor do que a de vértices pouco conectados.

3.12 Sistemas de recomendação

De acordo com Wang at al. (WANG *et al.*, 2015), sistemas de recomendação podem ser agrupados nas diferentes categorias:

- **Sistemas baseados em conteúdo.** Recomendação a partir de dados históricos de pedidos do usuário e de características textuais dos produtos adquiridos, que é mais indicada para dados estruturados, como por exemplo notícias e artigos;
- **Sistemas de filtros colaborativos.** Recomendação baseada em avaliações similares de produtos feitas por clientes distintos;
- **Sistemas híbridos.** Sistema misto que combina mais de um sistema dos tipos anteriores.

3.13 Protocolo de comunicação com *websites*

Ao acessar, utilizando um navegador de *internet*, o *website* de um comércio eletrônico, as requisições realizadas pelos usuários são enviadas do cliente (navegador) para o servidor (*website* do comércio eletrônico), através do protocolo *Hypertext Transfer Protocol* (HTTP), normalmente com requisições do tipo GET ou POST.

Uma requisição GET é um pacote montado pelo navegador de *internet* contendo um cabeçalho indicando o servidor de destino, e uma solicitação contendo a identificação do recurso que se deseja ler, sua *Unified Resource Location* (URL). Após processar a solicitação, o servidor de internet envia um pacote de resposta contendo todo o conteúdo associado à URL solicitada.

O tipo mais comum de conteúdo associado a URLs são documentos *Hypertext Markup Language* (HTML), estruturados conforme os padrões HTML/XML, mas não necessariamente estruturados semanticamente. Isto é, esses documentos são estruturados de forma visual, mas não estruturados de forma semântico. De modo geral, cabe a um usuário humano (ou a um *bot*, substituindo seu papel) interpretar a informação visual apresentada em seu navegador de *internet*, extraindo significado de cada pedaço de informação visual, como por exemplo o nome de um produto, sua fotografia e seu preço, quando apresentados em uma página de comércio eletrônico.

CAPÍTULO

4

METODOLOGIA

Nesta seção apresentamos os softwares utilizados para o desenvolvimento do *webcrawler*, e também a linguagem de programação e bibliotecas utilizadas para o treinamento dos modelos de aprendizado de máquina. Adicionalmente, descrevemos as principais técnicas utilizadas para avaliar a qualidade dos modelos obtidos, como curva ROC e matriz de confusão.

4.1 Aquisição de dados

4.1.1 Softwares

Para ampliar o alcance e a reprodutibilidade desta pesquisa, optou-se por utilizar apenas softwares livres, regidos por diferentes licenças de utilização.

- Navegador de internet **Mozilla Firefox** 52.2.0 (licença[1] MPL 2.0): acesso e coleta de dados de um website.
- Biblioteca de automatização de navegadores **Selenium** (licença[2] Apache 2.0): automatização do processo de acesso e coleta de dados de um website.
- Banco de dados **MySQL**: banco de dados relacional para armazenamento dos dados coletados.
- Distribuição **Anaconda** 4.4.0 (licença [3] BSD): distribuição contendo o binário do interpretador Python conjuntamente com diversas bibliotecas científicas.
- Interpretador **Python** 2.7.13 (licença[4] PSF) e bibliotecas:

[1] <https://www.mozilla.org/en-US/MPL/>
[2] <http://www.seleniumhq.org/about/license.jsp>
[3] <https://docs.continuum.io/anaconda/eula>
[4] <https://docs.python.org/2.7/license.html>

  - **NetworkX** 1.10 (licença[5] BSD): funções de construção e análise de redes/grafos.
  - **Pandas** 0.19 (licença[6] BSD): funções de conversão de estrutura de dados e medidas estatísticas.
  - **Numpy** 1.10 (licença[7] BSD): funções de cálculos matriciais.

- Interpretador **PHP** 7.0.33 (licença [8] PHP License v3.01)

4.1.2 Bases de dados

Como base de dados deste trabalho, utilizamos uma pequena fração dos produtos do catálogo da plataforma de comércio eletrônico Amazon (<www.amazon.com.br>).

Para cada produto de seu catálogo, a plataforma sugera uma lista de produtos comprados em conjunto (Figura 12), possibilitando a construção de uma rede de produtos comprados em conjunto.

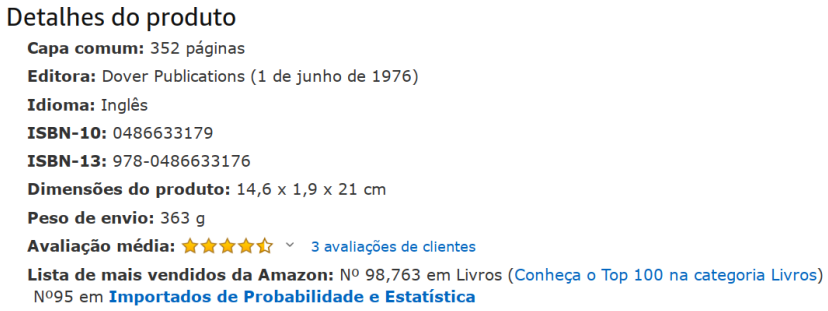
Além de dados básicos como título, autores e preço (Figura 10), a plataforma também apresenta informações mais detalhadas de cada produto, como por exemplo suas dimensões, idioma e quantidade de páginas (Figura 11).

Figura 10 – Captura de tela da página dos dados principais de um produto.

Fonte: Amazon Serviços de Varejo do Brasil Ltda (2019).

5 <https://networkx.github.io/documentation/networkx-1.10/reference/legal.html>
6 <http://pandas.pydata.org/pandas-docs/stable/overview.html#license>
7 <https://docs.scipy.org/doc/numpy-1.10.0/license.html>
8 <http://www.php.net/license/>

Figura 11 – Captura de tela dos detalhes de um produto.



Fonte: Amazon Serviços de Varejo do Brasil Ltda (2019).

Figura 12 – Captura de tela dos produtos comprados em conjunto.

Fonte: Amazon Serviços de Varejo do Brasil Ltda (2019).

4.1.3 Atributos a serem coletados

Para cada produto da base de dados, serão coletados os atributos litados na Tabela 3, sendo a maior parte deles categóricos, e o restante numéricos.

Tabela 3 – Lista de atributos a serem coletados, para cada produto, pelo *webcrawler*.

Atributo	**Tipo de atributo**	**Descrição**
title	categórico	Título do livro
category1	categórico	Categoria 1 do livro (mais geral)
category2	categórico	Categoria 2 do livro
category3	categórico	Categoria 3 do livro
category4	categórico	Categoria 4 do livro
category5	categórico	Categoria 5 do livro
category6	categórico	Categoria 6 do livro
category7	categórico	Categoria 7 do livro
category8	categórico	Categoria 8 do livro
category9	categórico	Categoria 9 do livro
category10	categórico	Categoria 10 do livro (mais específica)
language	categórico	Idioma
coverType	categórico	Tipo de capa
publisher	categórico	Editora
rankingCategory	categórico	Categoria utilizada para o ranking
authors	categórico	Autores
ranking	numérico	Posição no ranking de mais vendidos
reviewCount	numérico	Quantidade de avaliações de clientes
pages	numérico	Quantidade de páginas
weight	numérico	Peso (em gramas)
height	numérico	Altura (em centímetros)
width	numérico	Largura (em centímetros)
depth	numérico	Profundidade (em centímetros)
rating	numérico	Nota média dada por clientes

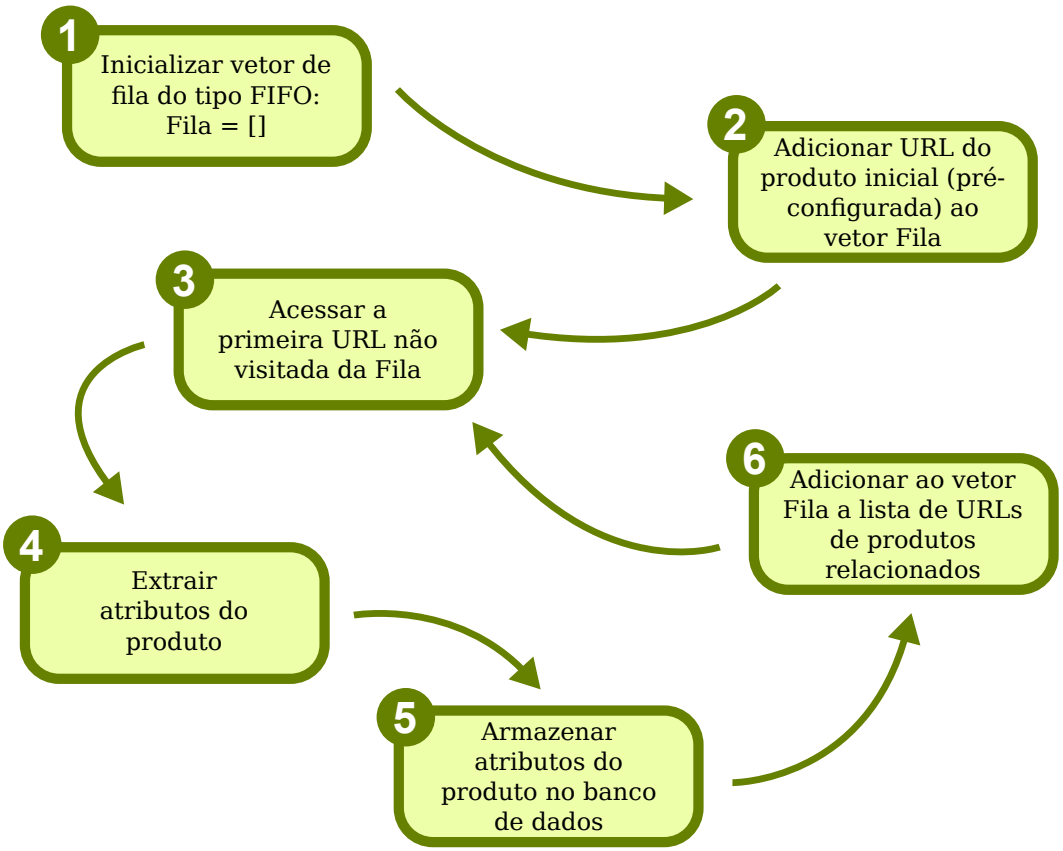
4.1.4 Comunicação com servidor da Amazon

Na falta de uma *Application Programming Interface* (API) oficial aberta para o público em geral, a única forma viável de requisitar conteúdo do sistema de comércio eletrônico da Amazon é por meio de requisições GET e POST convencionais, utilizando para isso as identificações de recurso públicas (URLs) fornecidas pela própria Amazon no domínio www.amazon.com.br. Para disparar essas requisições de forma automatizada, foi escolhida a biblioteca de automação Selenium.

4.1.5 Algoritmo de aquisição de dados

A aquisição de dados foi implementada seguindo o fluxograma da Figura 13.

Figura 13 – Fluxograma de aquisição de dados.



Fonte: Elaborada pelo autor.

4.1.6 Extração de atributos

Devido à estrutura não semântica do conteúdo disponibilizado de forma pública pela Amazon, os atributos de cada produto foram extraídos utilizando para isso a linguagem de consultas XPath, capaz de buscar informações dentro de um documento HTML/XML.

4.1.7 Armazenamento dos atributos extraídos

Os atributos extraídos foram armazenados utilizando o banco de dados MySQL, devido à sua natureza *open-source*.

4.1.8 Geração da rede de produtos

As redes foram criadas utilizando a biblioteca NetworkX da linguagem de programação Python a partir dos dados armazenados no banco MySQL.

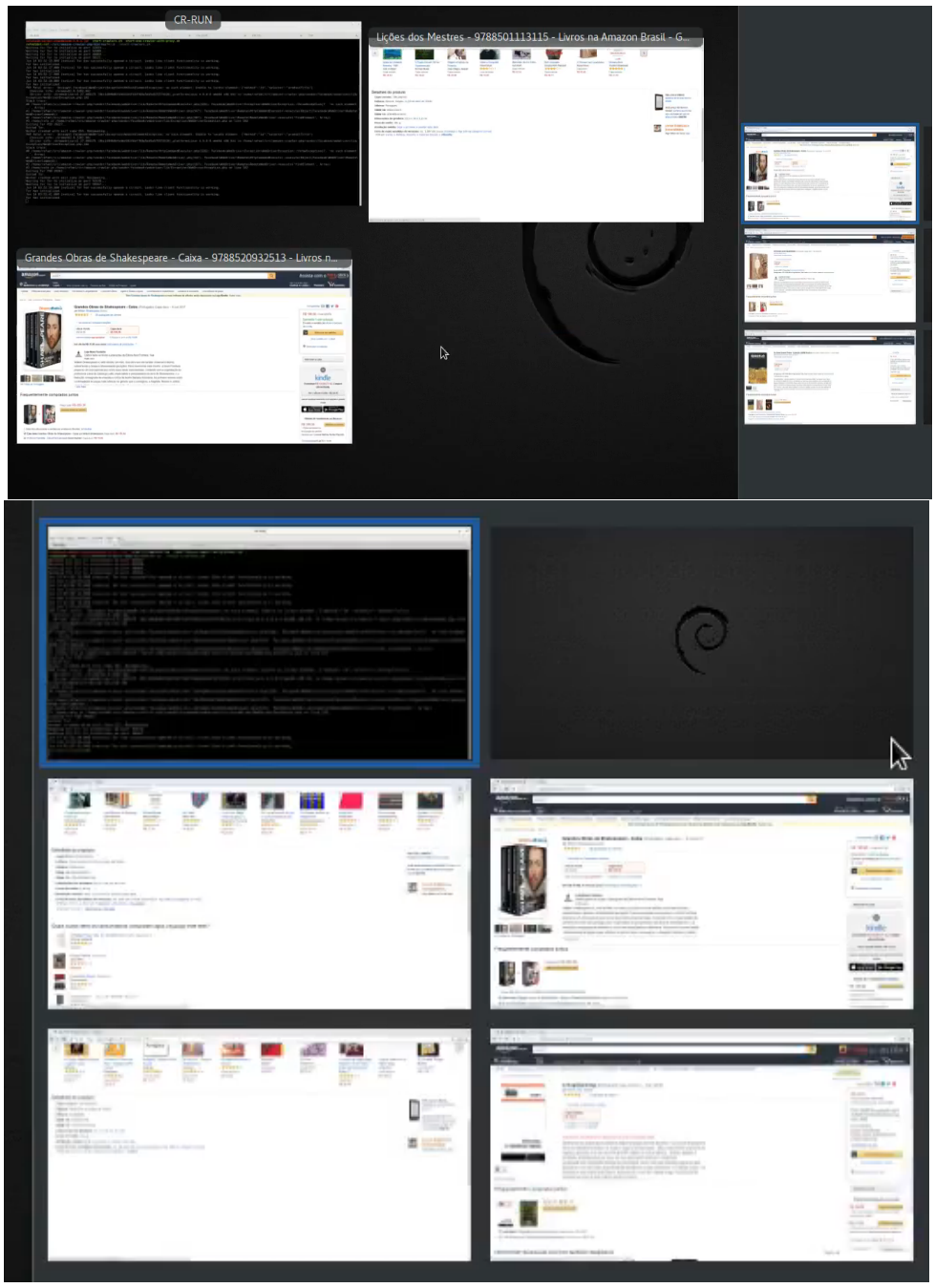
4.1.9 Motivação para desenvolvimento do webcrawler

Primeiramente foi necessário desenvolver um algoritmo para a obtenção dos dados da rede. A plataforma de comércio eletrônico da Amazon não disponibiliza uma API pública. Uma API pública permitiria que qualquer pessoa fizesse consultas de forma estruturada em sua base de dados de produtos. Alguns outros produtos, como Facebook, Twitter e Google Maps, disponibilizam APIs públicas para que desenvolvedores de *software* possam interagir de forma estruturada com seus serviços, mas a Amazon não fornece esse serviço.

Para contornar essa limitação, foi necessário então desenvolver um sistema *webcrawler* próprio. O *webcrawler* é um *bot* (robô), algoritmo programado para simular o comportamento de um consumidor humano que navega pela plataforma de comércio eletrônico da Amazon. Um possível comportamento de um dado consumidor é o seguinte: primeiramente abre o navegador de *internet*, e digita o endereço www.amazon.com.br na barra de navegação. Depois que a página é carregada, o cliente busca algum produto de seu interesse, e posteriormente continua sua navegação clicando em produtos relacionados. O *webcrawler* reproduz exatamente essa sequência de ações.

A principal diferença é que o produto inicial visitado pelo *webcrawler* é configurado manualmente no algoritmo. Sempre que o *webcrawler* inicia uma nova sessão de varredura, o programa inicializa um navegador de internet, acessa a URL do produto inicial, definida manualmente no código, e daí em diante visita todos os produtos relacionados, com busca em largura.

Figura 14 – Capturas de tela da execução do *webcrawler* com múltiplas instâncias em paralelo. Um script rodando no terminal (aplicação de fundo preto visível no canto superior esquerdo das imagens) é responsável por orquestrar a abertura de múltiplas instâncias de navegador de internet em paralelo para visitar as páginas dos produtos e capturar as informações relevantes de cada um deles.



Fonte: Elaborada pelo autor.

4.1.10 Escolha de linguagem de programação para o webcrawler

A escolha inicial de linguagem de programação para o desenvolvimento do *webcrawler* foi Python, devido à sua forte adoção pela comunidade científica e acadêmica. Testes preli-

minares mostraram a viabilidade dessa opção, entretanto, no decorrer do projeto, a escassez de ferramentas adequadas para a utilização de técnicas de engenharia de software bastante difundidas em outras linguagens de programação, como injeção de dependência e inversão de controle, se mostraram restritivas. Algumas bibliotecas de Python se propõem a fornecer essas funcionalidades, entretanto não são tão difundidas em comparação às bibliotecas de ecossistemas de outras linguagens de programação.

Em determinado momento do projeto, o algoritmo do *webcrawler* foi portado para a linguagem de programação PHP, por ser uma linguagem amplamente utilizada em desenvolvimento WEB, e que, por isso, conta com bibliotecas eficientes para a construção de um *webcrawler* robusto e compatível com boas práticas de engenharia de software. Além disso, PHP possui ferramentas *turn key* para tratar os obstáculos listados acima. Um exemplo é a biblioteca PHP-DI, responsável pela injeção de dependência.

4.1.11 Armazenamento do projeto no GitHub

Para aumentar a transparência da pesquisa, ampliar o alcance das ferramentas desenvolvidas e contribuir de forma mais contundente para o meio acadêmico, todos os algoritmos desenvolvidos no decorrer do projeto foram disponibilizados na plataforma de hospedagem de projetos *open source* GitHub.

Todos os algoritmos desenvolvidos estão disponíveis na URL: <https://github.com/rpagliuca?tab=repositories>.

4.1.12 Arquitetura do código-fonte do webcrawler

O projeto *webcrawler* em PHP está disponível no GitHub na URL <https://github.com/rpagliuca/amazon-crawler-php>.

O software utiliza as bibliotecas de terceiros PHP-DI, responsável pela injeção de dependência, Facebook Webdriver, classe que se comunica com o *daemon* de automação de navegadores de internet Selenium, e também Doctrine ORM, biblioteca que mapeia registros de bancos de dados relacionais para objetos da memória em tempo de execução. Para a gestão dessas bibliotecas utilizamos o software Composer, e para o desenvolvimento de softwares unitários e funcionais utilizamos a biblioteca PHPUnit.

A estrutura do projeto do software é dividida em cinco camadas:

- Business
- Console
- Entity
- Repository

- System

4.1.12.1 Business

Arquivos da camada Business são responsáveis pela execução de regras de negócio do software. Não dependem diretamente de nenhum método acoplado ao input ou output do algoritmo, sendo reaproveitáveis em diversos pontos do projeto.

4.1.12.2 Console

A camada Console engloba classes diretamente acopladas ao input e output do software, especificamente para execução via linha de comando.

4.1.12.3 Entity

A camada Entity engloba classes que definem objetos de runtime que são mapeados a registros armazenados em bancos de dados relacionais.

4.1.12.4 Repository

Esta camada abrange classes que especificam *queries* de bancos de dados, consultas que retornam coletâneas de objetos da classe Entity.

4.1.12.5 System

Por última, esta camada possui classes relacionadas com detalhes técnicos do sistema, como por exemplo arquivos de configuração e manipulação de estruturas de dados.

4.1.12.6 Principais classes

As principais classes do projeto são:

- App
- ItemParser
- ItemProcessor

A classe App é responsável pelo loop principal do algoritmo, que consome o próximo item da pilha da busca em largura, e com ele executa método principal da classe ItemParser, que por sua vez é responsável por se comunicar com o Facebook Webdriver para abrir a URL do produto no navegador de internet. Em seguida, a classe ItemParser é chamada para extrair as informações carregadas no navegador de internet. Essas informações são tratadas pela classe ItemProcessor e devolvidas para a classe ItemParser, que no final chama os métodos responsáveis por salvá-las no banco de dados relacional.

4.1.13 Execução em paralelo

Como um wrapper do *webcrawler*, foi desenvolvido um script utilizando a linguagem BASH que utiliza recursos nativos do Linux para a execução em paralelo de múltiplas instâncias do software. Isso permite rodar, paralelamente, dezenas de instâncias simultâneas do *webcrawler*.

Contudo, testes empíricos indicaram que, a partir de um determinado limiar de instâncias em paralelo, a performance agregada diminui, provavelmente devido a gargalos como banda de internet somados à utilização exaustiva da memória RAM do computador. Entretanto, até chegar a esse limiar, a adição sucessiva de instâncias em paralelos aumenta a performance coletiva.

4.1.14 Performance do webcrawler

A principal métrica utilizada para avaliar o *webcrawler* foi a quantidade de produtos garimpados por hora. Utilizando um único notebook Dell com processador Intel Core I7 e 16GB de memória RAM rodando a distribuição Linux Debian 9, foi obtida performance de pico de 1000 produtos garimpados em uma única hora.

4.1.15 Escolha da tecnologia de aprendizado de máquina

Pesquisas apontam que a linguagem de programação Python está entre as duas linguagens de programação mais populares entre cientistas de dados (JOHRI; BANSAL, 2018). Particularmente, a biblioteca de aprendizado de máquina Scikit Learn, desenvolvida em Python e escolhida para o nosso trabalho, começou a ser desenvolvida em 2007 e foi disponibilizada publicamente pela primeira vez em 2010 [9]. Temos, então, mais de uma década de desenvolvimento da biblioteca, e mais de 12.000 citações [10] no Google Scholar.

Em comparação, a linguagem PHP foi lançada em 1994 [11], por Rasmus Lerdorf, inicialmente com o propósito de facilitar o desenvolvimento de sites dinâmicos na internet, concorrendo com a linguagem PERL. Em seus primeiros anos, PHP era acrônimo de Personal Home Page [12], porém tal significado foi abandonado em 1997, com o lançamento da versão 3, momento em que a linguagem passou a suportar casos de usos mais diversificados, como processamento de arquivos, scripts para tarefas de manutenção de servidores, entre outros[13]. Já existem bibliotecas de aprendizado de máquina para PHP, como por exemplo PHP-ML [14], com uma visibilidade considerável, com mais de 6000 *stars*[15] na plataforma GitHub, métrica que representa informalmente, sem rigor algum, a quantidade de programadores que de alguma forma se interessa pela biblioteca, o que costuma indicar uma boa aceitação pela comunidade de desenvolvedores

[9] <http://ScikitLearn.org/stable/about.html#history>
[10] <https://scholar.google.com.br/scholar?hl=pt-BR&as_sdt=0%2C5&q=scikit+learn&btnG=>
[11] <http://php.net/manual/en/history.php.php>
[12] <http://php.net/manual/en/history.php.php>
[13] <http://php.net/manual/en/history.php.php>
[14] <https://github.com/php-ai/php-ml>
[15] <https://help.github.com/en/articles/about-stars>

de software. Entretanto, tal software tem menos de 2 anos desde o seu lançamento, e ainda não foi encontrada nenhuma citação a seu respeito no Google Scholar. Inclusive, o site oficial da biblioteca PHP-ML não aponta nenhum artigo científico de referência para ser citado, ao contrário da biblioteca Scikit Learn.

Uma terceira biblioteca também foi prospectada: a Tensor Flow, biblioteca da Google, que foi lançada em 2015 e obteve rápida aceitação tanto pela comunidade científica quanto pela indústria de desenvolvimento de software. Essa biblioteca também possui integração com a linguagem Python, mas por ser mais recente e possuir menos citações científicas, preferimos não utilizá-la neste momento, mas com a ressalva de voltar a reconsiderá-la para trabalhos futuros.

Levando em conta os diferentes propósitos das bibliotecas citadas, optamos por utilizar a biblioteca Scikit Learn e a linguagem de programação Python para desenvolver os algoritmos de aprendizado de máquina.

4.1.16 Introdução ao Jupyter

Jupyter é um software *open source* que permite embutir rotinas, desenvolvidas em Python ou outras linguagens de programação, em cadernos virtuais, com exibição integrada de gráficos, cache de resultados de funções, formatação de textos, comentários e imagens. Esses cadernos virtuais podem ser compartilhados entre múltiplos pesquisadores, e também podem ser nativamente exportados para PDF.

Escolhemos utilizar o Jupyter para facilitar a revisão e o compartilhamento dos resultados obtidos.

4.2 Estimativa de preço

Começamos o projeto com a instalação do software Jupyter [4.1.16] e a criação de um novo projeto Python. Para prover de forma simplificada todas as bibliotecas científicas, utilizamos a distribuição Anaconda com Python 2.7.

4.2.1 Base de dados

Extraímos da base de dados coletada pelo *webcrawler* dois CSVs, um deles contendo os atributos de cada nó (Arquivo de texto 2), e outro listando as ligações existentes entre cada nó (Arquivo de texto 1).

Arquivo de texto 1 – Primeiras 10 linhas do arquivo CSV de ligações entre nós

```
1: fromNode_id,toNode_id
2: 2,1
3: 3,1
```

```
 4: 20,1
 5: 219,1
 6: 1717,1
 7: 1739,1
 8: 1846,1
 9: 1848,1
10: 1854,1
```

Arquivo de texto 2 – Primeiras 3 linhas do arquivo CSV contendo atributos de nós

```
1: id,title,url,authors,coverType,publisher,edition,
     publicationDate,rankingCategory,category1,category2,
     category3,category4,category5,category6,category7,category8,
     category9,category10,isbn10,isbn13,language,postProcessed,
     price,ranking,pages,reviewCount,rating,width,height,depth,
     weight
2: 1,"The Stanford Mathematics Problem Book: With Hints and
     Solutions",https://www.amazon.com.br/dp/0486469247/,"George
     Polya (Autor),","Capa comum","Dover Publications",,"19 de
     fevereiro de 2009",Livros,Livros,"Inglês e Outras Línguas","
     Ciências Tecnológicas",Matemática,"Estudo e Ensino"
     ,,,,,,0486469247,978-0486469249,Inglês
     ,1,26.25,59183,68,1,4.0,14,21,0.6,181
3: 2,"Fourier Series",https://www.amazon.com.br/dp/0486633179/,"
     Georgi P. Tolstov (Autor),","Capa comum","Dover Publications
     ",,"1 de junho de 1976",Livros,Livros,"Inglês e Outras Lí
     nguas","Ciências Tecnológicas",Matemática,Aplicada,"
     Probabilidade e Estatística",,,,,0486633179,978-0486633176,
     Inglês,1,50.37,56112,352,3,4.6,14.6,21,1.9,363
```

4.2.2 Leitura dos dados e métricas de rede

As bases de dados são importadas e convertidas para estrutura de rede utilizando a biblioteca NetworkX. Usando métodos dessa biblioteca, as seguintes medidas de rede são calculadas e armazenadas na memória:

- Grau
- *Eigenvector centrality*
- *Betweenness centrality*

A **betweenness centrality** é calculada utilizando o método *approximate_current_flow_-betweenness_centrality* do NetworkX, caso contrário o algoritmo não finaliza o seu processamento em um período de tempo aceitável. Optamos por não calcular a medida **closeness centrality** pela mesma razão de tempo de processamento exigido.

Nessa mesma etapa, os atributos dos nós juntamente com as métricas de rede calculadas são armazenadas em um *dataframe* do Pandas, e classificados em atributos categóricos ou numéricos. Dados faltantes são convertidos para a estrutura interna np.NaN (valor não numérico) da biblioteca Numpy do Python.

4.2.3 Baseline e medidas de performance

Nesta etapa, calculamos o preço médio e o preço mediano do conjunto de produtos do dataframe para utilizar como baseline. Um suposto algoritmo de estimativa de preços que não utiliza aprendizado de máquina, e sim retorna em 100% dos casos o preço médio ou mediano dos produtos pré-existentes no catálogo pode ser usado como baseline, isto é, como exemplo de eficiência mínima a ser atingida. Para isso, calculamos a soma do módulo dos erros, ou então a soma do quadrado dos erros utilizando os modelos baselines, e fazemos o mesmo cálculo para o modelo treinado.

Por exemplo, suponhamos que temos em nosso catálogo 3 produtos, com os preços abaixo:

```
Produto 1 - R$5
Produto 2 - R$10
Produto 3 - R$15
```

Um possível algoritmo de baseline utiliza o preço médio dos produtos, nesse caso, R$10, para estimar preços futuros. Ao utilizarmos esse modelo baseline para prever o preço de um novo Produto 4, sem considerar os atributos individuais desse produto, obteremos o valor constante de R$10.

Para calcular o erro de baseline, podemos rodar o modelo para estimar o preço de diversos outros produtos que já conhecemos o preço a priori. Vamos supor que já conhecemos os preços abaixo:

```
Produto 4 - R$5
Produto 5 - R$20
Produto 6 - R$10
Produto 7 - R$5
Produto 8 - R$20
```

Nesse caso, o modelo baseline estimaria o preço de R$10 para todos eles, resultando em um erro total de:

```
Produto 4: |R$10 - R$5| = R$5
Produto 5: |R$10 - R$20| = R$10
Produto 6: |R$10 - R$10| = R$0
Produto 7: |R$10 - R$5| = $5
Produto 8: |R$10 - R$20| = R$10
Erro médio: (5 + 10 + 0 + 5 + 10)/5 = R$6
```

Calculando o mesmo erro médio utilizando um outro modelo de estimativa, podemos compará-lo com o baseline e concluir se ele é mais eficiente em estimar o preço do que simplesmente adivinhar a média.

Podemos comparar o modelo obtido com o baseline também por meio do erro relativo médio, o que reduz o peso de *outliers* no resultado de performance. Assumindo a mesma base do exemplo acima, obteríamos os seguintes erros relativos:

```
Produto 4: |R$10 - R$5| = R$5 => 5/5 = 1
Produto 5: |R$10 - R$20| = R$10 => 10/20 = 0,5
Produto 6: |R$10 - R$10| = R$0 => 0/10 = 0
Produto 7: |R$10 - R$5| = $5 => 5/5 = 1
Produto 8: |R$10 - R$20| = R$10 =>  10/20 = 0,5
Erro relativo médio: (1 + 0,5 + 0 + 1 + 0,5)/5 = 0,6
```

4.2.4 Separação entre bases de treinamento e teste

Ao treinar um modelo de aprendizado de máquina, estamos sujeitos ao overfitting, fenômeno em que o modelo se baseia fortemente em atributos muito particulares dos exemplos utilizados durante o treinamento. Voltando ao nosso exemplo anterior, um modelo afetado por overfitting poderia estimar o valor de R$5 sempre que o nome do produto for Produto 1, R$10 sempre que o nome do produto for Produto 2 e R$15 sempre que o nome do produto for Produto 3.

Entretanto, esse aprendizado não seria generalizável, pois ao estimar o preço para o Produto 4, o modelo não estaria apto a fazer uma estimativa. Em outras palavras, overfitting acontece quando um modelo está "decorando"os atributos dos exemplos usados para o treinamento, em vez de aprender padrões mais generalizáveis.

Para evitar overfitting, uma estratégia comumente utilizada durante treinamento de modelos é a separação das bases em pelo menos 2: treinamento e teste. Existem diferentes estratégias para fazer essa separação, e o objetivo é fazer com que o modelo seja treinado

utilizando a base de treinamento, e depois avaliado com a base de teste, sem que haja vazamento de informação entre as dois conjuntos distintos.

Dessa forma, modelos que sofram de overfitting são naturalmente descartados, já que os exemplos da base de treinamento não existem na base de teste, portanto a avaliação do modelo resultaria em um score baixo.

4.2.5 Treinamento do modelo

Foi escolhido o regressor Random Forest para criar o modelo de estimativa de preços.

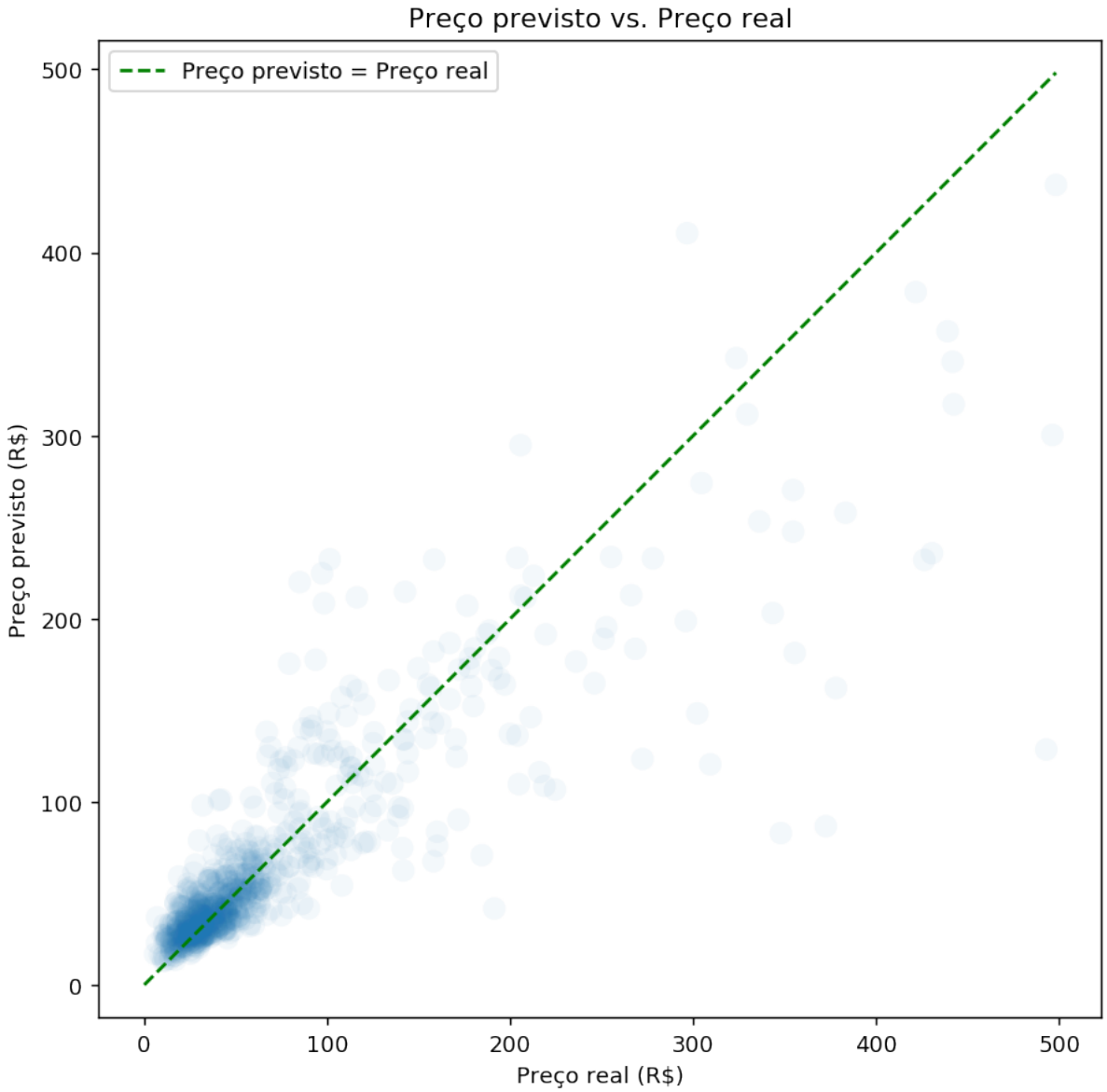
Esse regressor utiliza um número personalizável de árvores de decisão, utilizando o coeficiente Gini para montar as árvores de decisão de maior relevância. Depois de criadas as árvores de decisão, ao estimar o preço com o modelo, o algoritmo faz uma média de todos os valores estimados.

4.2.6 Avaliação do modelo

4.2.6.1 Inspeção visual

Uma das maneiras de avaliar a eficiência do modelo obtido é por meio de inspeção visual.

Figura 15 – Exemplo de gráfico de dispersão de preço estimado vs preço real, que pode ser utilizado para avaliar visualmente a eficiência de um dado modelo de precificação de produtos.



Fonte: Elaborada pelo autor.

Por meio da inspeção visual do gráfico da Figura 15, podemos ter uma visão rudimentar inicial da eficiência do algoritmo. Na linha tracejada verde temos a condição ideal em que o preço previsto é igual ao preço real. Pontos acima dessa linha possuem preço previsto maior do que o preço real, enquanto pontos abaixo da linha possuem preço previsto menor do que o preço real.

4.2.6.2 Erro absoluto médio e erro absoluto relativo médio

Outra forma de avaliar a eficiência do modelo é calcular o erro absoluto médio, e compará-lo com o erro absoluto médio do modelo baseline (valor médio). Similarmente, podemos calcular a razão entre o erro absoluto e o preço real para cada uma das estimativas, e calcular a sua média para todas as amostras de treinamento e teste.

4.3 Previsão de ligação entre nós

Criamos um modelo de previsão de ligação entre nós. Particularmente, o modelo tenta prever a ligação entre um nó qualquer e o nó 1 da nossa base de dados.

4.3.1 Base de dados

As bases de dados utilizadas são as mesmas do modelo de previsão de preços.

4.3.2 Leitura dos dados e métricas de rede

Os dados são lidos para dataframes Pandas e objetos NetworkX de modo similar ao descrito na Seção 4.2.2.

Uma importante diferença é que foi criado um novo atributo chamado *existe_ligacao_com_no_1*, que armazena o valor 0 ou 1, indicando, respectivamente, se o nó possui ou não possui ligação com o nó 1.

4.3.3 Análise do baseline

Como o problema é do tipo classificação, e temos duas classificações bem desbalanceadas (uma dezena de nós são ligados ao nó 1, enquanto os milhares restantes não são), não podemos simplesmente calcular a porcentagem de acertos como métrica de eficiência.

Caso fizéssemos, teríamos a falsa impressão de que um modelo que sempre sugere que não há ligação possui alta eficiência. O que precisamos é de algum tipo de média harmônica ou geométrica, que penalize modelos que erram nas classificações mais raras. O gráfico de *Receiver Operating Characteristic* (ROC), conhecido como curva ROC (Seção 4.3.6.1), também cumpre esse papel muito bem, permitindo-nos comparar a taxa de falsos positivos com verdadeiros positivos para descartar modelos demasiadamente enviesados.

4.3.4 Separação entre bases de treinamento e teste

Separamos as bases entre treinamento e teste, assim como fizemos na Seção 4.2.4.

4.3.5 Treinamento do modelo

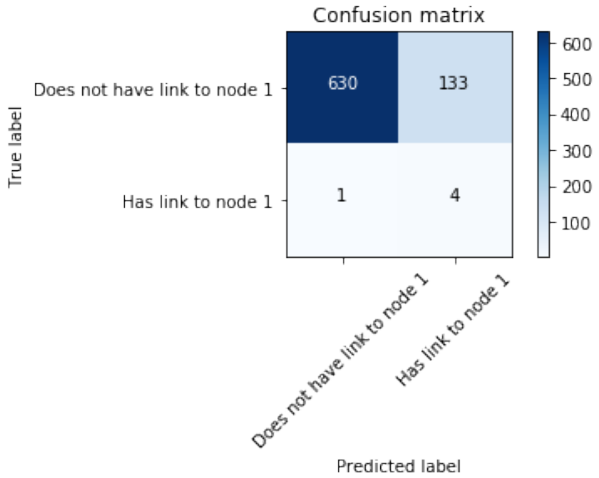
O modelo foi treinado utilizando todos os atributos existentes (categóricos e numéricos, incluindo as métricas de rede), e foi utilizado como alvo a coluna *existe_ligacao_com_no_1*.

Novamente, utilizamos o classificador Random Forest, dada a sua alta performance computacional e eficiência aceitável.

4.3.6 Avaliação do modelo

Como se trata de uma tarefa de classificação, a matriz de confusão (Figura 16) é uma das ferramentas mais apropriadas para a avaliação da eficiência.

Figura 16 – Exemplo de matriz de confusão, ferramenta utilizada para identificar a quantidade de verdadeiros positivos, falsos positivos, verdadeiros negativos e falsos negativos de um modelo de classificação.



Fonte: Elaborada pelo autor.

Por se tratar de classificação binária, em que a classe verdadeira é uma dentre positivo e negativo, e levando em conta os parâmetros livres do modelo, principalmente os que definem o limiar de classificação, é útil conhecermos a curva ROC do modelo obtido.

4.3.6.1 Curva ROC

A curva ROC é obtida pela coleção de pontos de sensibilidade vs. especificidade em um sistema de classificação binária, e uma das métricas para avaliar a eficiência de um modelo de classificação é a área sob a curva ROC, *Area Under Curve* (AUC).

Em uma dos eixos da curva ROC temos a taxa de verdadeiros positivos (sensibilidade), e no outro eixo temos a taxa de falsos positivos (1 - especificidade).

O ponto no eixo em que a sensibilidade é mínima (sensibilidade = 0) possui especificidade também máxima, já que o modelo classificará como negativo qualquer input, sem distinção (não há falsos positivos).

Já o ponto no eixo em que a sensibilidade é máxima (sensibilidade = 1) acarreta redução na especificidade (ou seja, aumento na taxa de positivos falsos).

Para entender melhor, vamos tentar avaliar a eficiência de um detector de incêndio instalado em uma fábrica.

Ao instalarmos um detector de incêndio com sensibilidade muito alta, provavelmente ele vai disparar o alarme com muita frequência, tanto em casos verdadeiros de incêndios, como em casos falsos (fumantes próximos ao detector, fumaça de escapamento de veículos, como empilhadeiras que transitam por ali ou outros equipamentos).

Se ajustarmos os parâmetros do detector para reduzir a sua sensibilidade, ele passará a disparar o alarme com bem menos frequência. Provavelmente, não vai mais disparar o alarme devido a cigarros ou veículos, mas possivelmente, também vai deixar de ser acionado para alguns incêndios verdadeiros.

Isto é, quando reduzimos a sensibilidade do sensor, aumentamos o risco de não conseguir detectar algum um incêndio verdadeiro.

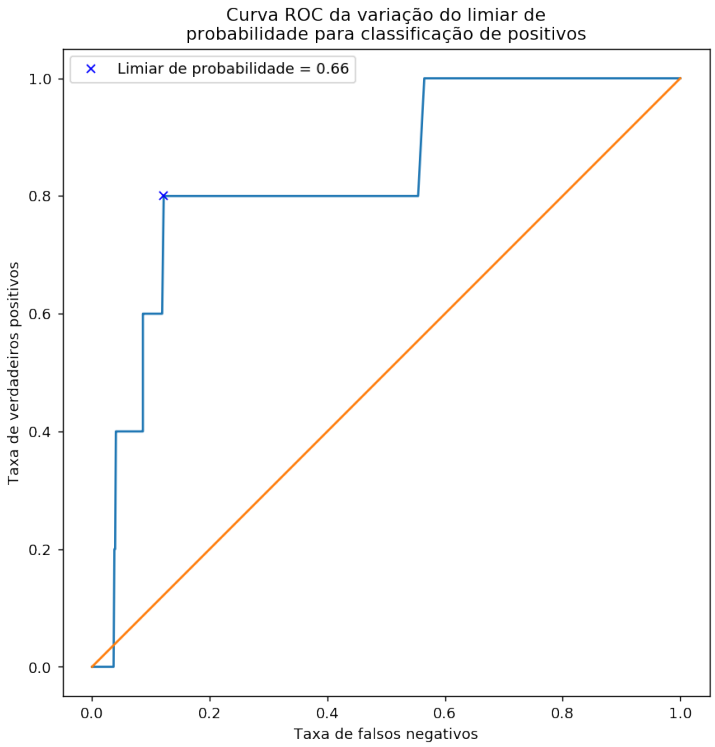
Perceba que a sensibilidade ideal do detector de incêndio depende do seu objetivo. Se a fábrica produz fogos de artifício, provavelmente desejaremos um detector com sensibilidade máxima, mesmo que isso implique que o sensor dispare erroneamente várias vezes por dia (alta quantidade de falsos positivos). Por outro lado, se esse detector de incêndio estivesse localizado em um ambiente sem materiais inflamáveis, poderíamos, por comodidade, diminuir a sua sensibilidade, acarretando uma redução na quantidade de alarmes falsos.

Voltando à curva ROC, para construí-la, precisamos apenas variar a sensibilidade do modelo (ou, na analogia acima, do detector de incêndio), e anotar as respectivas especificidades.

4.3.6.2 Como obter a matriz de confusão ótima, a partir da curva ROC

Na Figura 17 apresentamos um possível gráfico de curva ROC, utilizando um conjunto preliminar de resultados experimentais. Com esse resultado, é possível calcular o ponto mais próximo ao ponto ótimo (x = 0; y = 1), e utilizar a sensibilidade desse ponto da curva para calcular a matriz de confusão ótima (Figura 18).

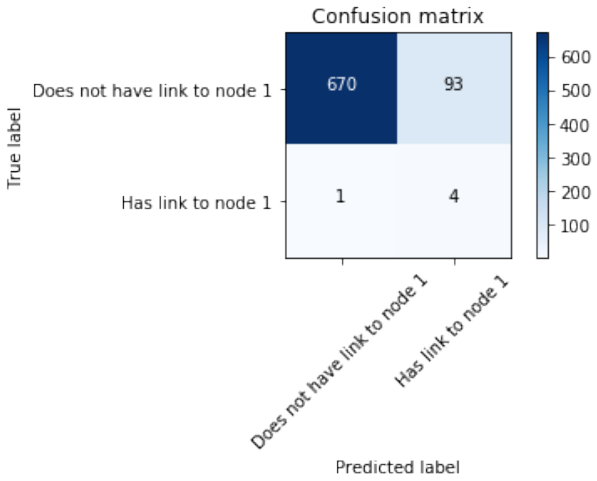
Figura 17 – Curva ROC variando um dos parâmetros do modelo, que é a probabilidade limiar que classifica um exemplo como positivo. O parâmetro que mais se aproxima ao ponto ótimo (x = 0, y = 1) foi obtido adotando o limiar de probabilidade = 0,66. Nessa condição, todos os exemplos em que o modelo retorna probabilidade > 0,66 são identificados como positivos; caso contrário, negativos.



Fonte: Elaborada pelo autor.

Utilizando a probabilidade limiar mais próxima do ponto ótimo da curva ROC (Figura 17), podemos reconstruir a matriz de confusão apresentada anteriormente, e analisar o impacto positivo (Figura 18). Repare que o número de falsos negativos e verdadeiros positivos se manteve o mesmo, mas tivemos uma diminuição significativa (aproximadamente 30%) na quantidade de falsos positivos em comparação com a Figura 16.

Figura 18 – Matriz de confusão obtida para limiar de probabilidade de classificação binária = 0,66.



Fonte: Elaborada pelo autor.

CAPÍTULO

5

RESULTADOS

Este capítulo está dividido em 3 seções, assim como foi feito no capítulo anterior (Capítulo 4 - Metodologia). Na Seção 5.1 analisamos a rede construída a partir dos dados obtidos; na Seção 5.2 relatamos os resultados do treinamento do modelo de estimativa de preço; e na Seção 5.3 descrevemos os resultados do treinamento do modelo de previsão de ligação entre nós.

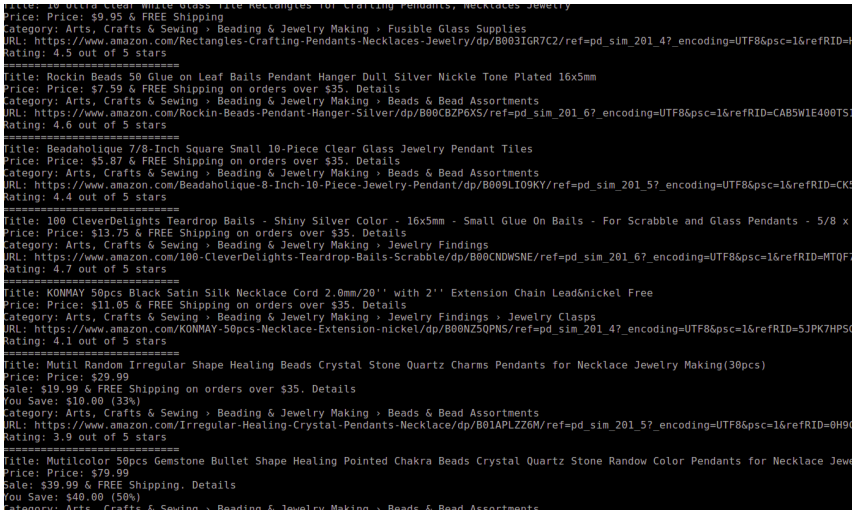
5.1 Análise da rede

Foi implementado o software *webcrawler* capaz de extrair atributos de produtos a partir do comércio eletrônico da empresa Amazon <www.amazon.com.br>. O código-fonte dessa implementação foi disponibilizado no GitHub.

Foi verificada a necessidade de implementação de ajustes estocásticos e ergódicos no *webcrawler* para evitar grupos de nós absorvedores, e adaptação do algoritmo para fazer busca em largura em vez de busca em profundidade (Figura 13).

Uma pequena amostra dos dados extraídos de produtos relacionados é apresentada na Figura 19.

Figura 19 – Captura de tela da saída do *webcrawler*, listando parte dos dados extraídos a partir de produtos relacionados.



Fonte: Elaborada pelo autor.

Todo código-fonte desenvolvido até neste trabalho foi disponibilizado publicamente nas seguintes URLs:
<https://github.com/rpagliuca/ms-amazon-crawler>
<https://github.com/rpagliuca/ms-amazon-copurchased>
<https://github.com/rpagliuca/ms-graphtools>
<https://github.com/rpagliuca/ms-simple-graphs>

5.1.1 Estrutura da rede

5.1.1.1 Distâncias e diâmetro da rede

Estimamos as distâncias geodésicas médias e o diâmetro da rede de maneira probabilística.

Em cada repetição do experimento, escolhemos 1000 pares de nós aleatoriamente, e calculamos a distância geodésica entre cada um desses pares.

Executamos 10 repetições distintas, cujos resultados estão apresentados na Tabela 4. Dessas repetições, obtivemos o valor de 4,11 para a média da média das distâncias geodésicas. Já o desvio padrão encontrado para a média das distâncias geodésicas foi de 0,02. Em todas as repetições obtivemos 7 como estimativa do diâmetro da rede.

Tabela 4 – Maior valor, média e desvio padrão de distâncias geodésicas para 10 repetições experimentais, constituídas de 1000 distâncias geodésicas cada uma, calculadas para pares aleatórios de nós.

Número da repetição	**Maior distância geodésica**	**Média das distâncias geodésicas**	**Desvio padrão das distâncias geodésicas**
1	7	4.12	0.99
2	7	4.10	0.97
3	7	4.16	0.95
4	7	4.08	1.00
5	7	4.12	1.01
6	7	4.08	1.01
7	7	4.14	1.00
8	7	4.12	1.02
9	7	4.10	1.02
10	7	4.11	0.99

Um segundo experimento foi realizado, similar ao descrito acima, com uma importante diferença metodológica: em vez de sortearmos pares de nós, como feito anteriormente, neste experimento sorteamos apenas nós individuais.

Em cada repetição experimental, escolhemos um único nó da rede aleatoriamente, e calculamos as menores distâncias entre o nó sorteado e todos os outros nós da rede.

Com este experimento, conseguimos estimar os limites mínimo e máximo para o diâmetro da rede.

Realizamos 10 repetições experimentais, cujos resultados estão apresentados na Tabela 5. Neste experimento, obtivemos resultados similares: média da média das distâncias geodésicas de 4,14, e desvio padrão de 0,42.

É importante destacar que os resultados dispostos na Tabela 5 limitam o diâmetro da rede entre 8 e 12, já que em uma das repetições obtivemos a distância 6 como a maior distância geodésica, limitando o máximo do diâmetro da rede para o dobro dessa distância. No outro extremo, obtivemos a distância 8 como maior distância geodésica de uma das repetições, efetivamente estabelecendo um mínimo para o diâmetro da rede de 8.

Este método se mostrou extremamente rápido e prático para estimarmos o diâmetro da rede.

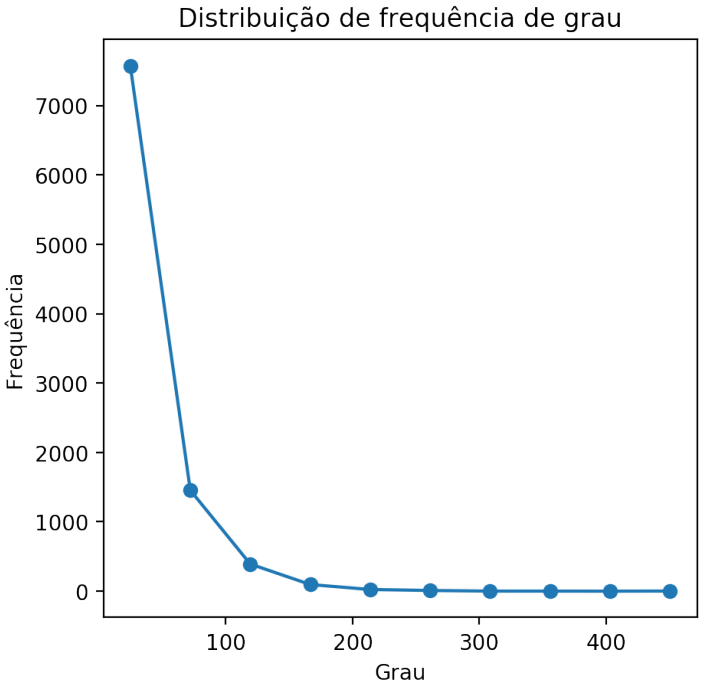
Tabela 5 – Maior valor, média e desvio padrão de distâncias geodésicas para 10 repetições experimentais, constituídas de todas as distâncias geodésicas calculadas a partir de um nó aleatório.

Número da repetição	Maior distância geodésica	Média das distâncias geodésicas	Desvio padrão das distâncias geodésicas
1	7	4.39	0.83
2	7	3.87	0.97
3	8	4.34	0.85
4	6	3.71	0.74
5	7	4.42	0.73
6	7	3.49	0.92
7	7	4.10	0.84
8	7	3.81	0.95
9	7	4.98	0.74
10	6	4.34	0.85

5.1.2 Distribuição de grau

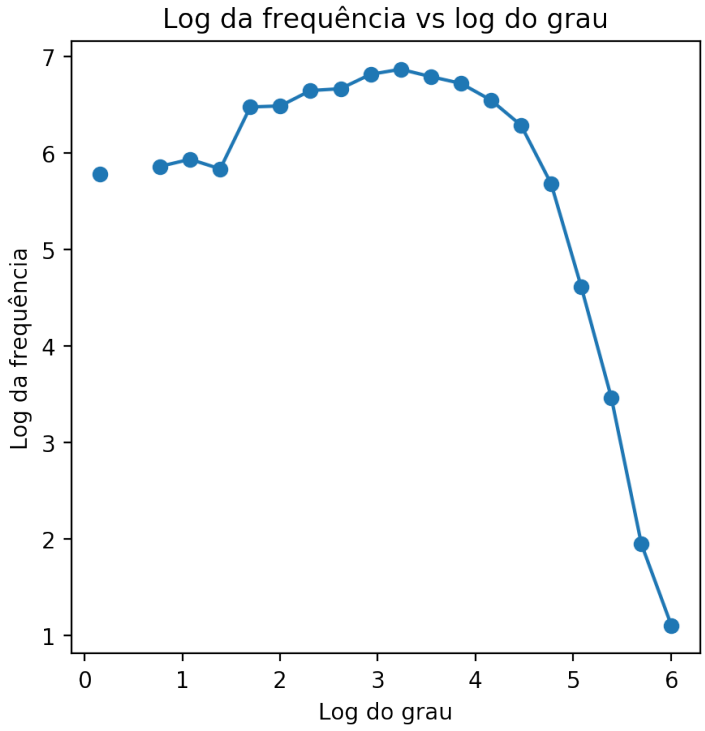
Foi plotado na Figura 20 a distribuição de grau dos nossos dados experimentais, apresentando aspectos típicos de uma distribuição que segue a lei de potência: queda logarítmica, e longa cauda. Na Figura 21 os mesmos dados foram plotados em escala logarítmica, e nele podemos notar uma faixa cujo logaritmo do grau se encontra entre 4 e 6, com aspecto próximo a uma reta, o que seria típico de uma distribuição de grau que segue a lei de potência. A respectiva faixa foi ampliada na Figura 22, e plotada juntamente a uma reta obtida por meio de regressão linear com os dados experimentais dessa faixa.

Figura 20 – Distribuição de grau.



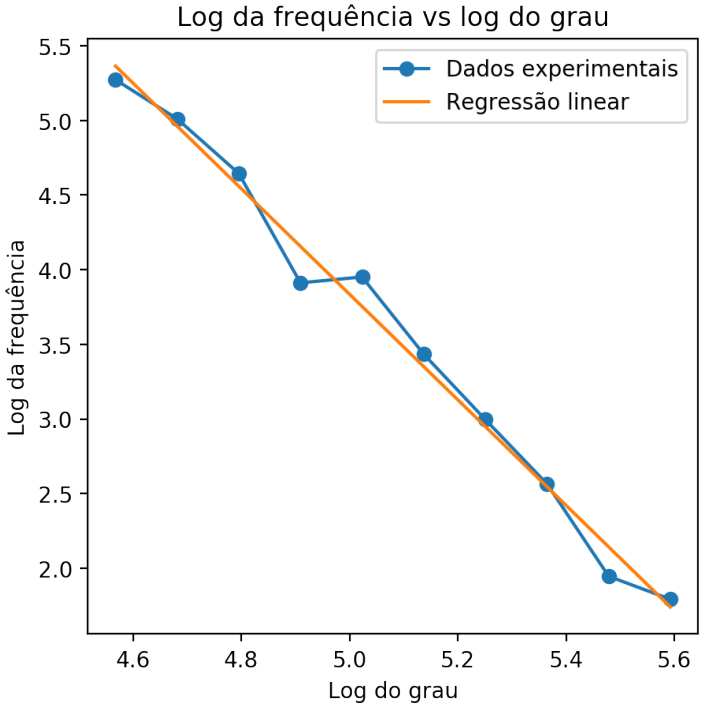
Fonte: Dados da pesquisa.

Figura 21 – Distribuição de grau em escala log-log.



Fonte: Dados da pesquisa.

Figura 22 – Distribuição de grau em escala log-log, restrita à faixa entre 4,5 e 5,75 de valores do logaritmo do grau.

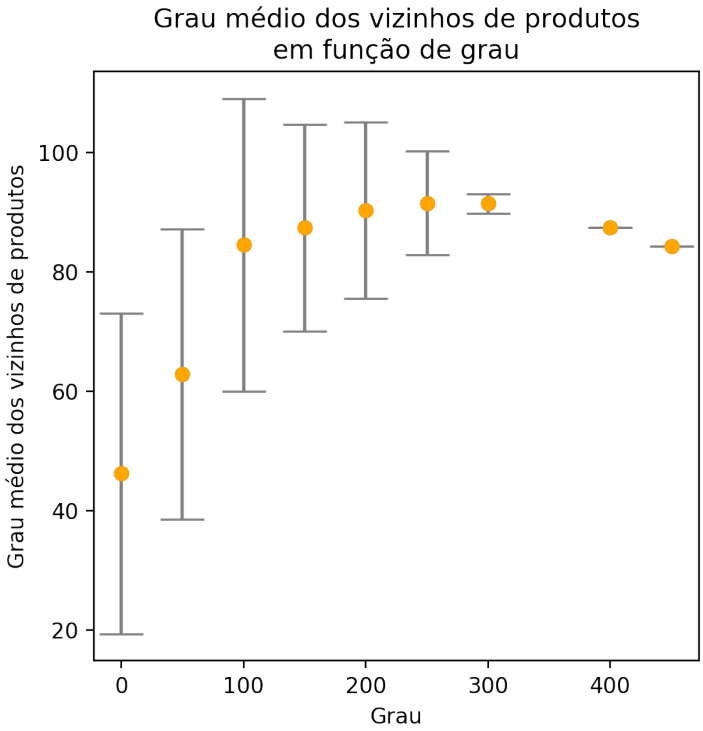


Fonte: Dados da pesquisa.

5.1.3 Assortatividade

Apresentamos na Figura 23 o gráfico de assortatividade da rede de produtos comprados em conjunto, indicando comportamento homofílico, em que nós de grau mais alto tendem a possuir vizinhos de nós mais alto.

Figura 23 – Assortatividade: grau médio dos vizinhos de um nó em função do grau do produto.

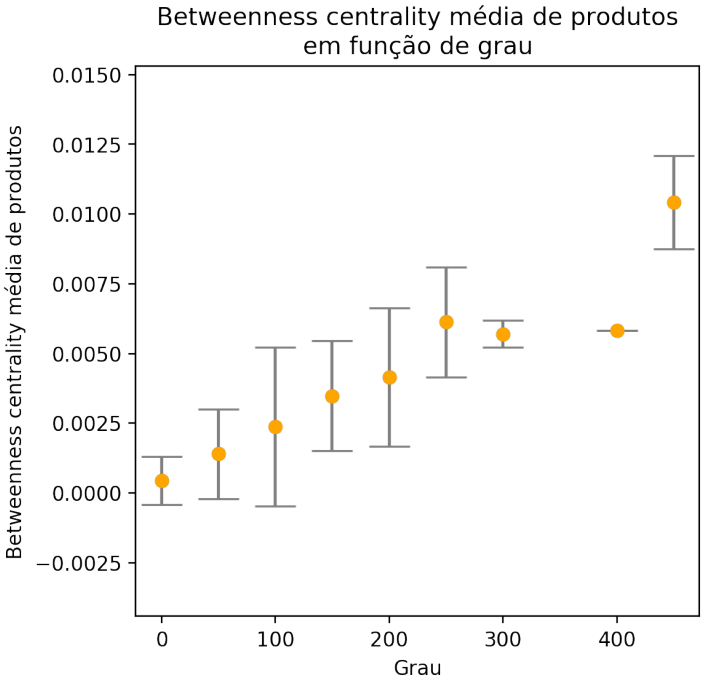


Fonte: Dados da pesquisa.

5.1.4 Correlação entre medidas de centralidade

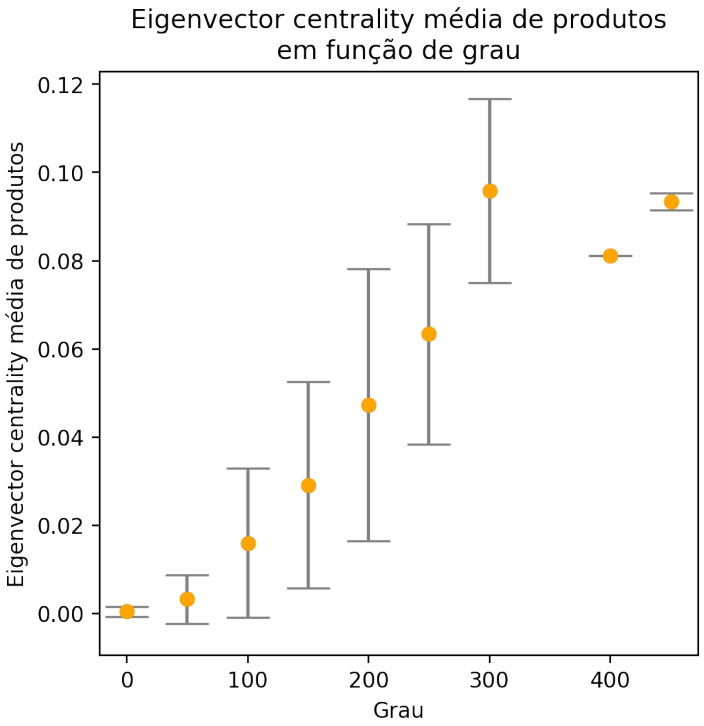
Nas Figuras 24 e 25 apresentamos os gráficos de correlação entre a medida de centralidade mais simples (grau) e as duas outras medidas de centralidade calculadas (*betweenness centrality* e *eigenvector centrality*). Os dois gráficos mostram uma correlação positiva.

Figura 24 – *Betweenness centrality* média em função do grau.



Fonte: Dados da pesquisa.

Figura 25 – *Eigenvector centrality* média em função do grau.



Fonte: Dados da pesquisa.

5.1.5 Correlação entre atributos

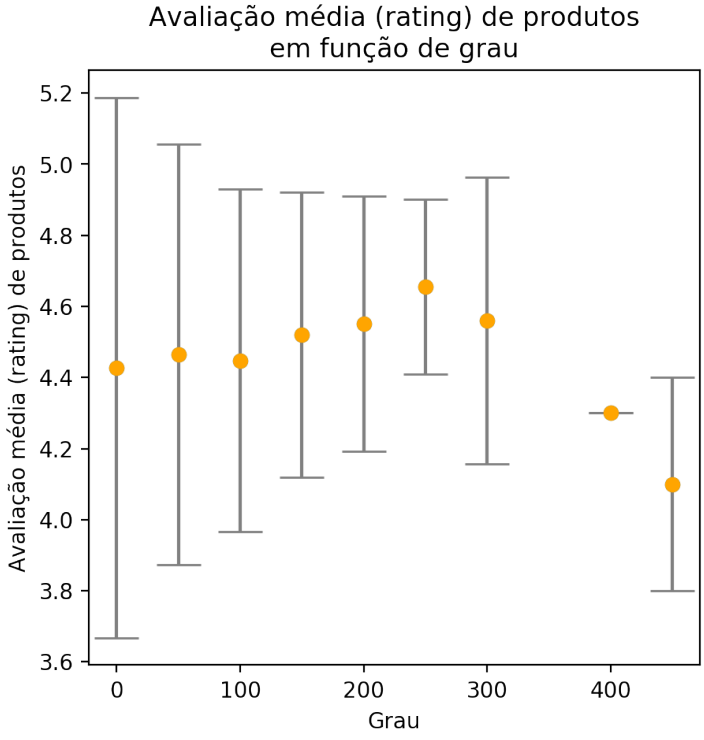
5.1.5.1 Correlação entre medidas de centralidade, avaliação de consumidores e ranking dos produtos mais vendidos

Nas Figuras 26 a 31 foram apresentados os gráficos de correlação entre atributos de importância comercial (avaliação de consumidores e posição no ranking de produtos mais vendidos) e atributos de centralidade de rede (grau, *betweenness centrality* e *eigenvector centrality*).

Esses gráficos mostram que não há uma correlação clara entre a avaliação média de consumidores e as medidas de centralidade, de modo que rejeitamos uma das hipóteses deste trabalho, de que *hubs* da rede seriam melhor avaliados pelos consumidores. Sugestões de trabalhos futuros envolvendo esse resultado são apresentadas no Capítulo 6.

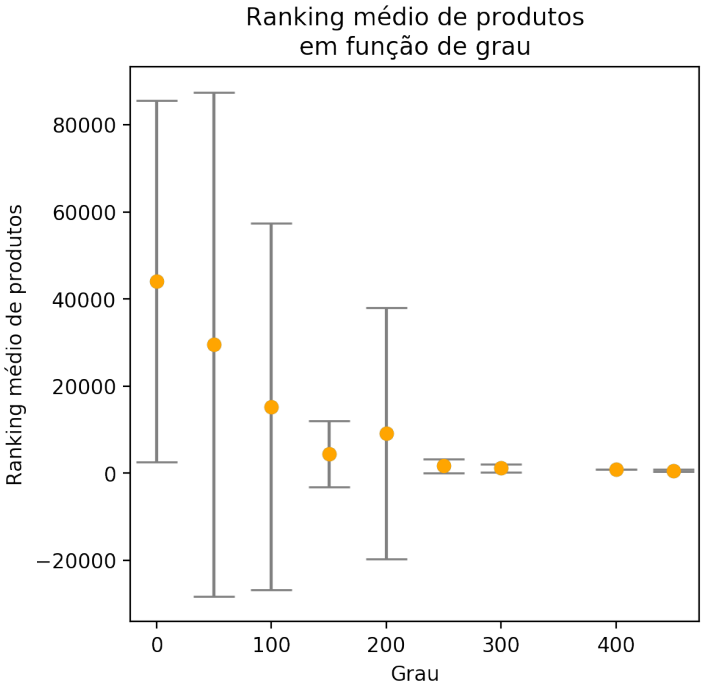
Por outro lado, esses mesmos gráficos sugerem que os *hubs* possuem melhor posição no ranking de produtos mais vendidos.

Figura 26 – Avaliação média de clientes em função do grau do produto.



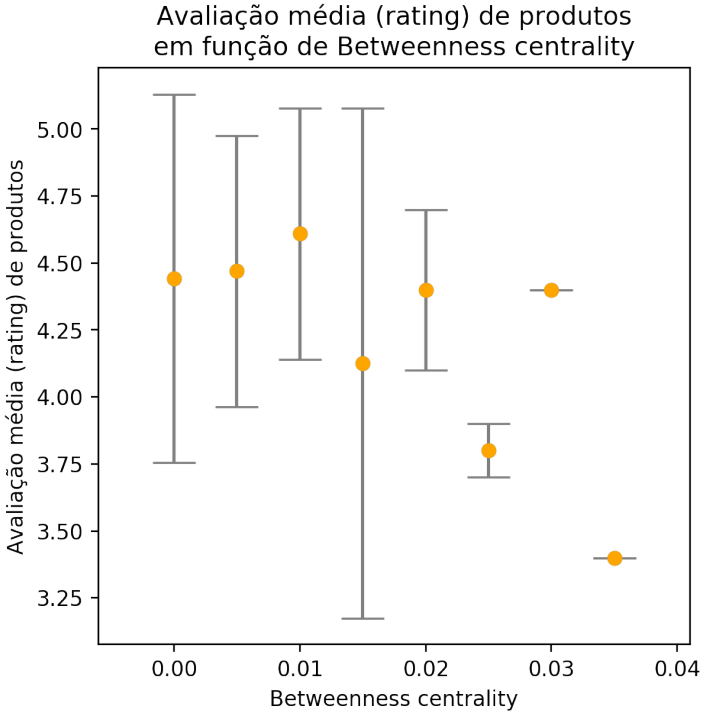
Fonte: Dados da pesquisa.

Figura 27 – Posição média no ranking de produtos mais vendidos em função do grau do produto.



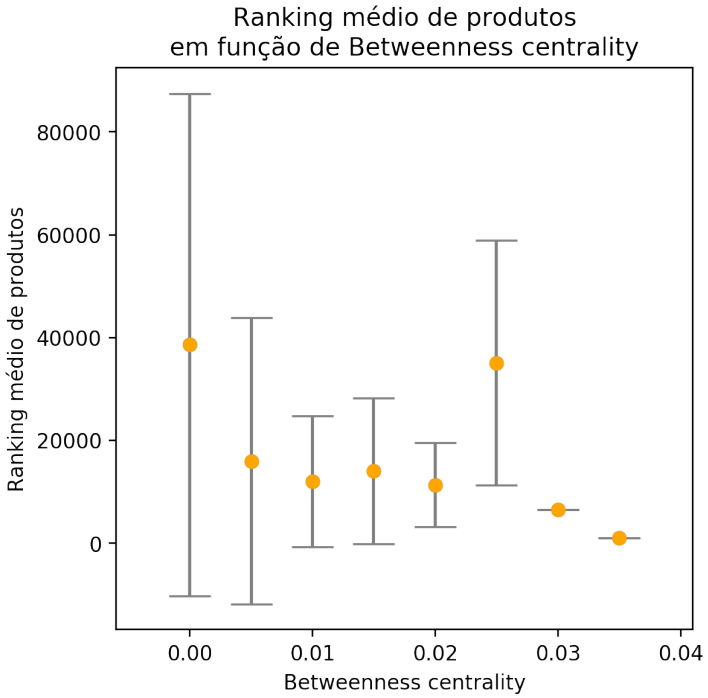
Fonte: Dados da pesquisa.

Figura 28 – Avaliação média de clientes em função da *betweenness centrality* do produto.



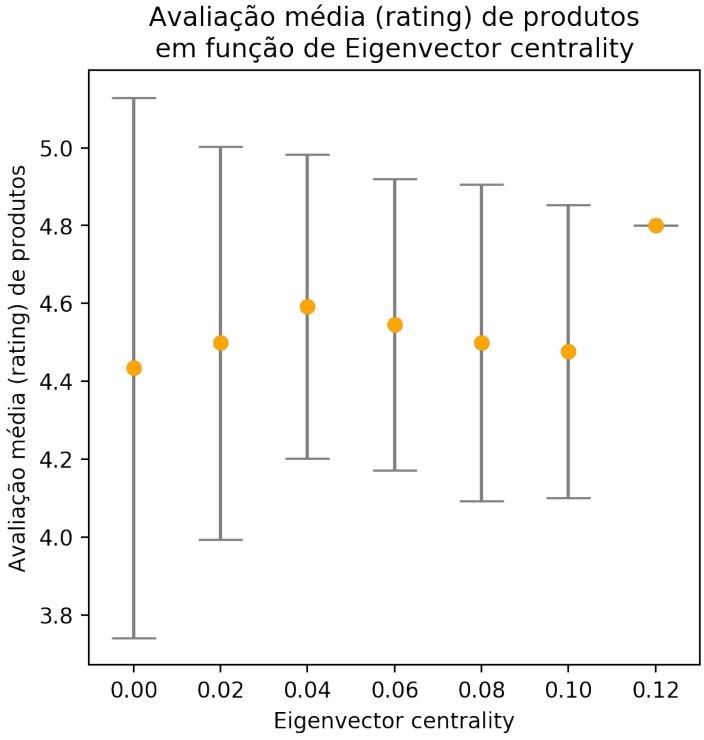
Fonte: Dados da pesquisa.

Figura 29 – Posição média no ranking de produtos mais vendidos em função da *betweenness centrality* do produto.



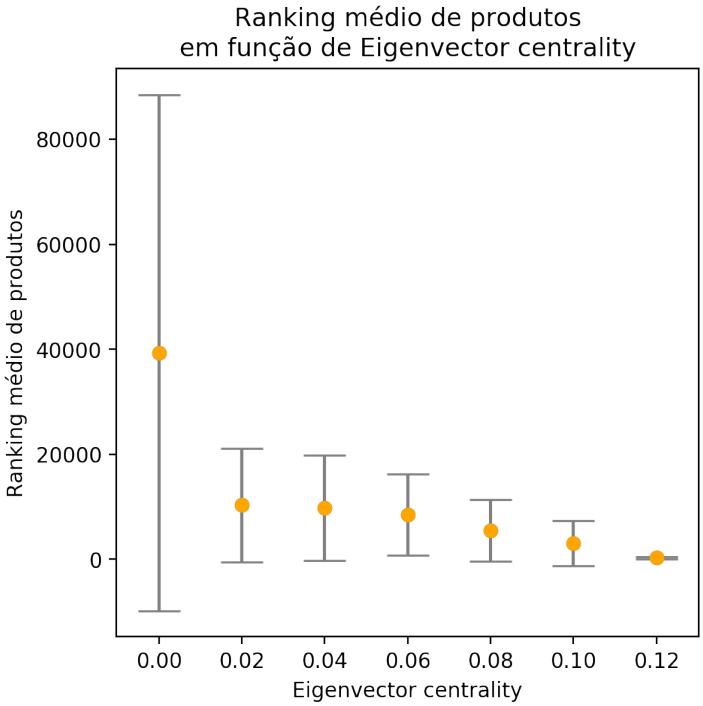
Fonte: Dados da pesquisa.

Figura 30 – Avaliação média de clientes em função da *eigenvector centrality* do produto.



Fonte: Dados da pesquisa.

Figura 31 – Posição média no ranking de produtos mais vendidos em função da *eigenvector centrality* do produto.

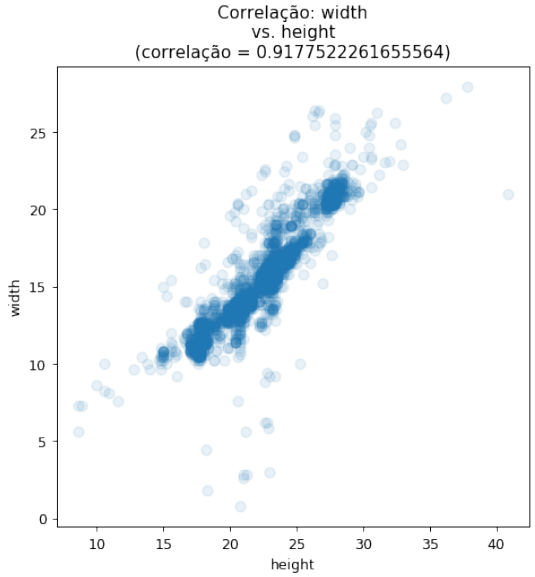


Fonte: Dados da pesquisa.

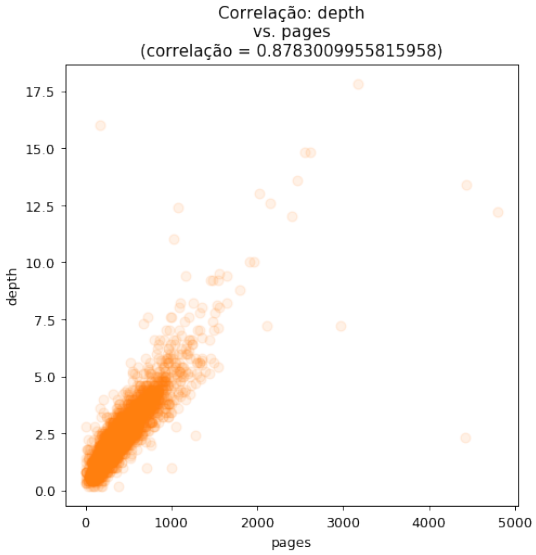
5.1.5.2 Outras correlações evidentes

Naturalmente, alguns atributos listados na seção anterior possuem correlação entre si. Para avaliar a correlação de atributos categóricos, utilizamos a codificação *one-hot-encoding*, em que, para cada valor distinto possível de um atributo, cria-se um novo pseudoatributo binário, indicando a presença ou não daquele valor.

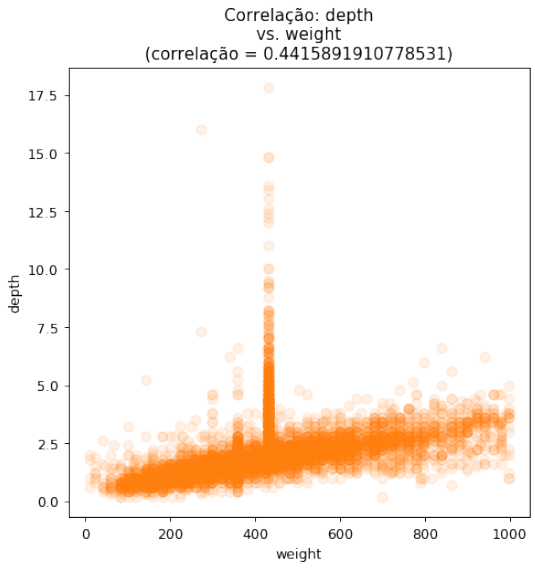
Figura 32 – Correlação entre atributos físicos dos livros (largura, altura, profundidade, número de páginas e peso).



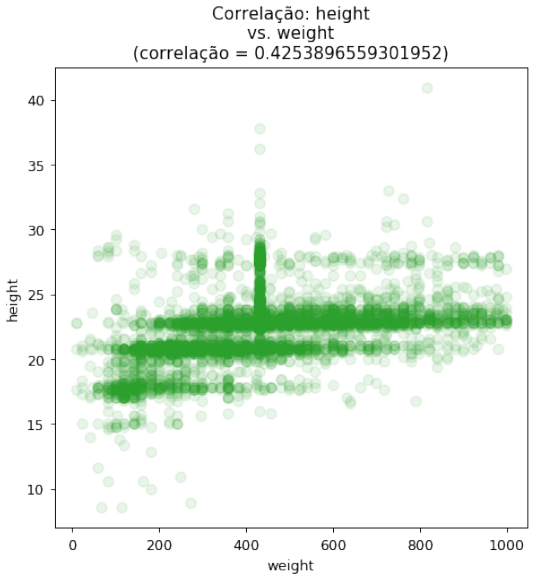
(a) Largura vs. altura



(b) Espessura vs. quantidade de páginas



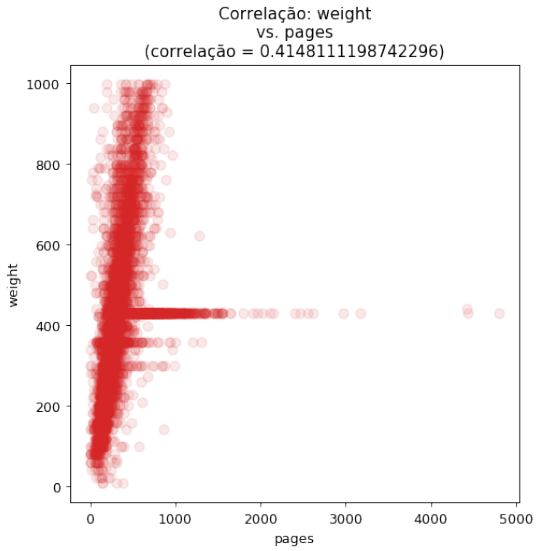
(c) Espessura vs. peso



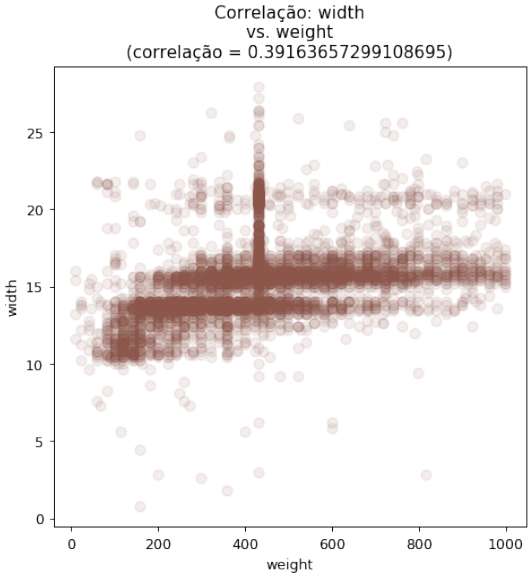
(d) Altura vs. peso

Fonte: Dados da pesquisa.

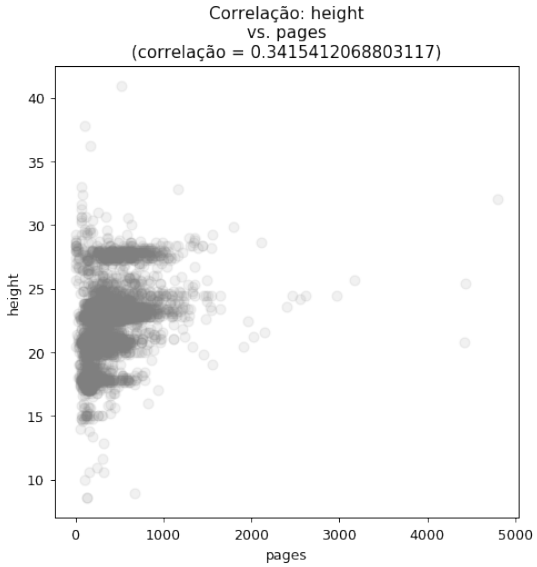
Figura 32 – (continuação) Correlação entre atributos físicos dos livros (largura, altura, profundidade, número de páginas e peso).



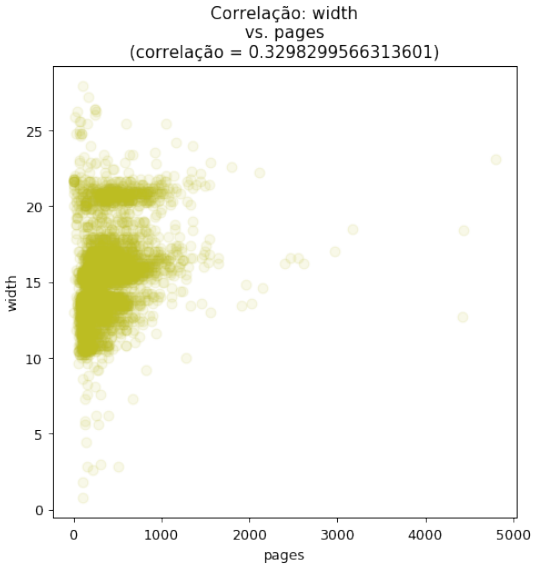
(e) Peso vs. quantidade de páginas



(f) Largura vs. peso



(g) Altura vs. quantidade de páginas



(h) Largura vs. quantidade de páginas

Fonte: Dados da pesquisa.

Observando os gráficos de dispersão da Figura 32 notamos algumas correlações bem intuitivas de atributos físicos dos produtos, como por exemplo a correlação entre largura e altura de um livro, da profundidade do livro e sua quantidade de páginas, entre outras.

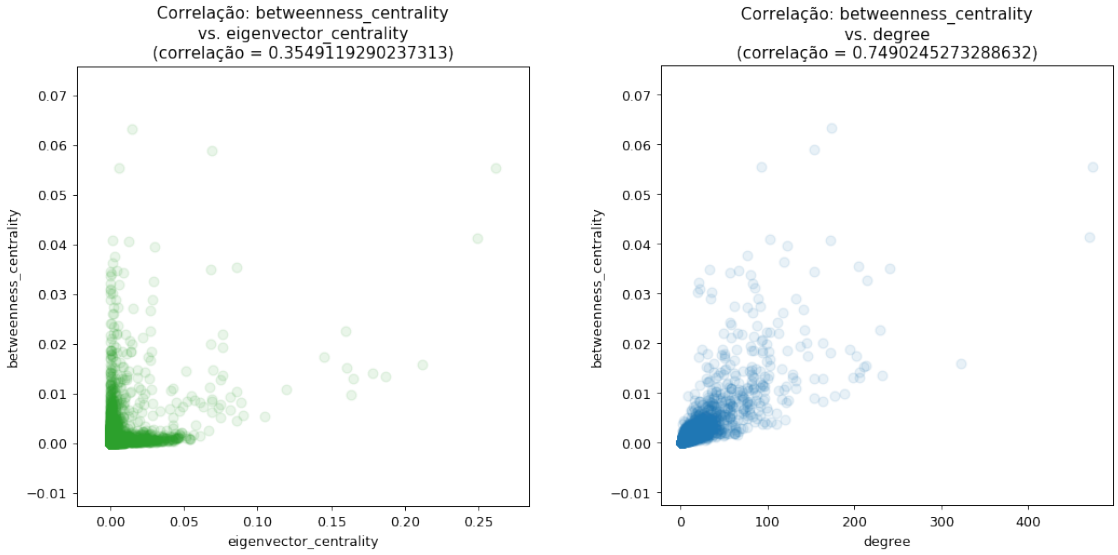
Da Figura 32, também conseguimos extrair algumas informações não tão triviais, como por exemplo o fato de que livros entre 400g e 500g de peso têm uma alta variação em seus outros atributos (profundidade, altura, numero de páginas, largura). Entre outras coisas, isso pode

indicar que o peso de 400g a 500g é informado como valor padrão em caso de não conhecimento do peso real do produto.

Outra classe de atributos que merece uma análise dedicada de correlação é a de métricas de rede, que engloba o grau, *eigenvector centrality* e *betweenness centrality*.

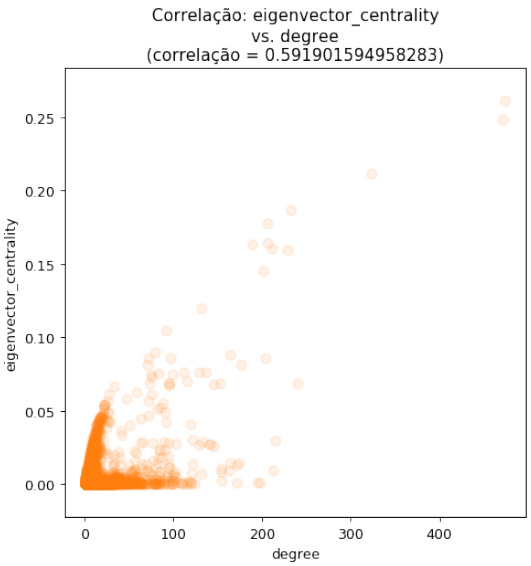
Esses gráficos são similares aos apresentados anteriormente na Seção 5.1.4, com a diferença de que aqui não foram calculados os valores médios, e sim apresentados por meio de gráficos de dispersão.

Figura 33 – Correlação entre atributos de métricas de rede dos livros.



(a) *Betweenness centrality* vs. *eigenvector centrality*

(b) *Betweenness centrality* vs. grau



(c) *Eigenvector centrality* vs. grau

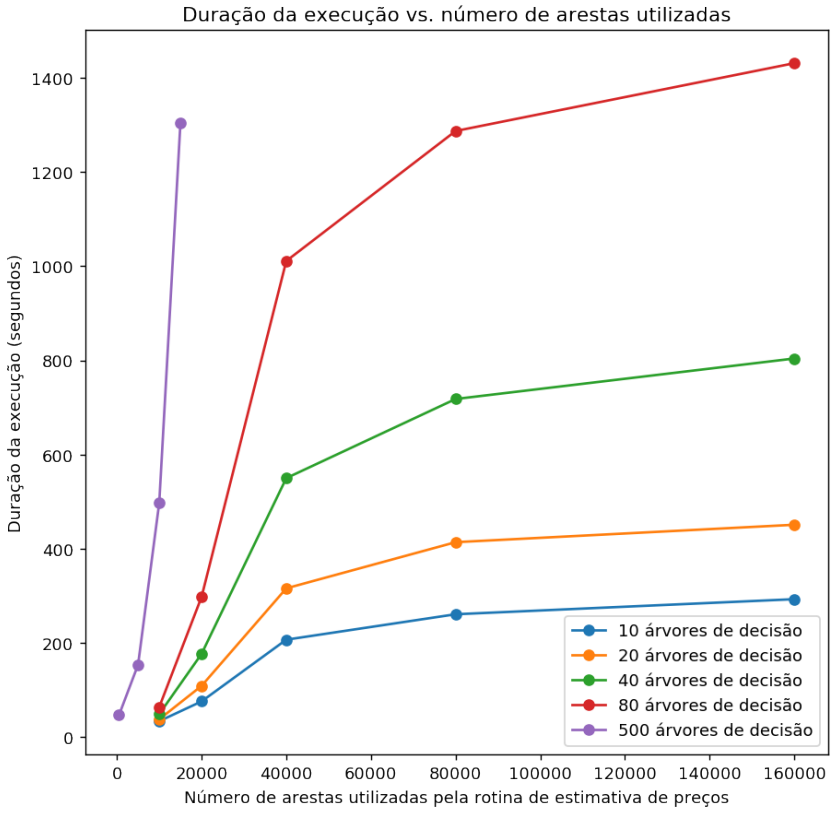
Fonte: Dados da pesquisa.

5.2 Estimativa de preço

5.2.1 Escolha do número de estimadores (árvores de decisão) do regressor Random Forest

Para definir o parâmetro *n_estimators* do regressor Random Forest do Scikit Learn, executamos a rotina de estimativa de preços variando o número de arestas utilizadas (*max_edges*) e medimos a duração do tempo de execução, para diferentes valores de *n_estimators*. Com os resultados dispostos na Figura 34, percebemos que o tempo de execução para mais de 100.000 arestas é proporcional a *n_estimators*, respeitando a complexidade computacional $O(n\_estimators \cdot K \cdot N \cdot \log N)$, sendo K a quantidade de atributos e N a quantidade de amostras, conforme literatura sobre a complexidade computacional do algoritmo Random Forest (LI *et al.*, 2010).

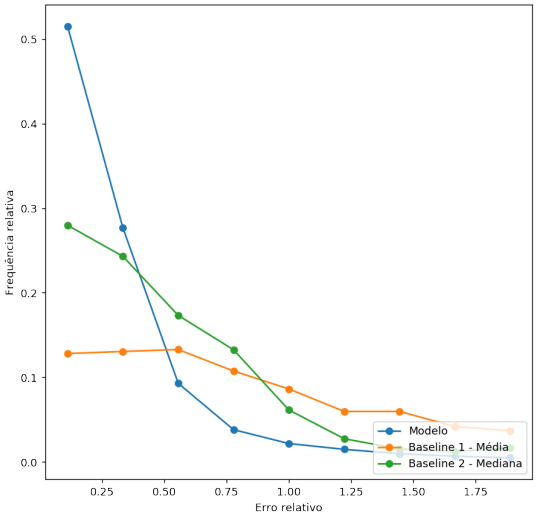
Figura 34 – Duração do tempo de execução do algoritmo de estimativa de preços em função do número de arestas consumidas para o treinamento do modelo.



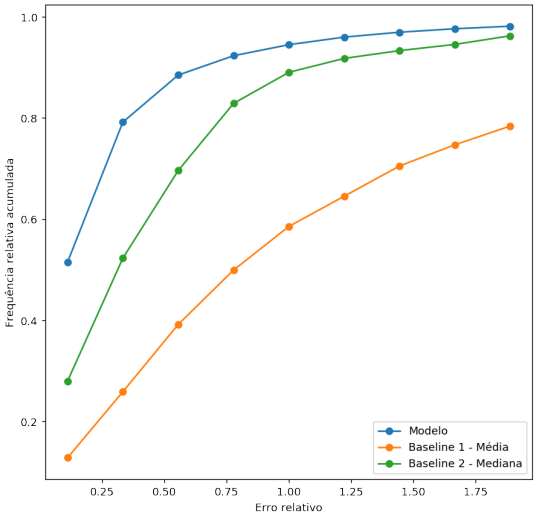
Fonte: Dados da pesquisa.

Em seguida, comparamos a qualidade dos resultados obtidos entre 10, 20, 40 e 80 árvores de decisão mantendo fixo o parâmetro $max\_edges = 160000$, usando para isso os gráficos de distribuição de erro relativo e erro relativo acumulado (Figura 35).

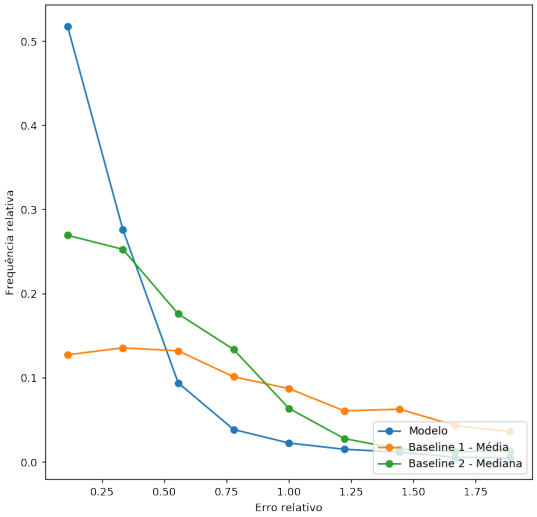
Figura 35 – Distribuição de erro relativo (esquerda) e erro relativo acumulado (direita) para diferentes valores de *n_estimators* (de cima para baixo: 10, 20, 40, 80).



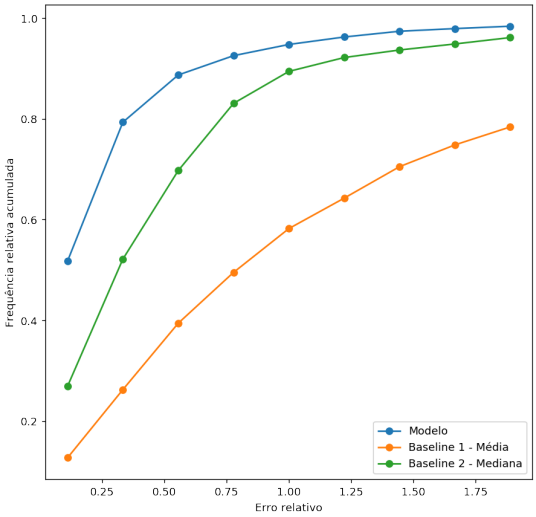
(a) Erro relativo para $n\_estimators = 10$



(b) Erro acumulado para $n\_estimators = 10$



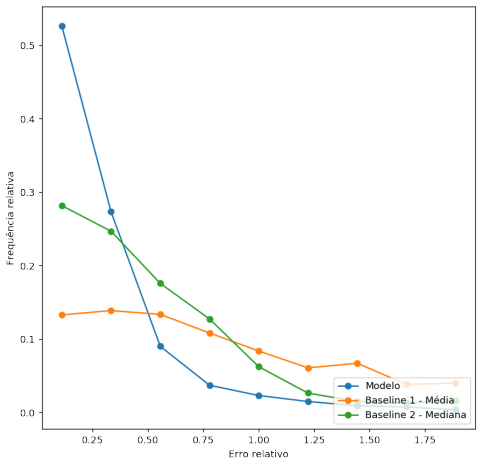
(c) Erro relativo para $n\_estimators = 20$



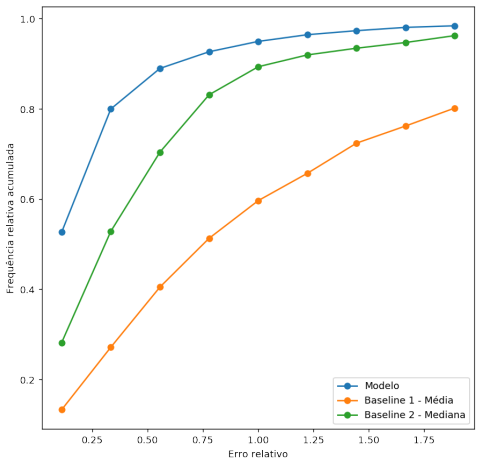
(d) Erro acumulado para $n\_estimators = 20$

Fonte: Dados da pesquisa.

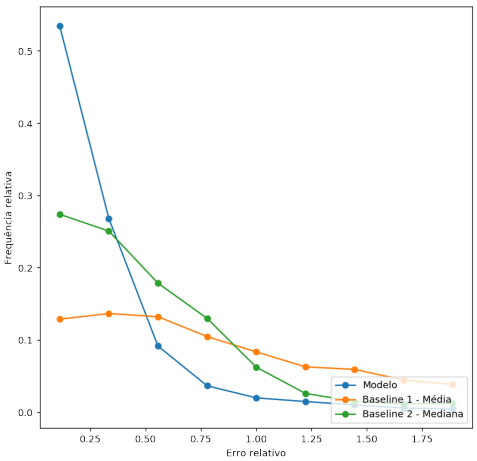
Figura 35 – (continuação) Distribuição de erro relativo (esquerda) e erro relativo acumulado (direita) para diferentes valores de *n_estimators* (de cima para baixo: 10, 20, 40, 80).



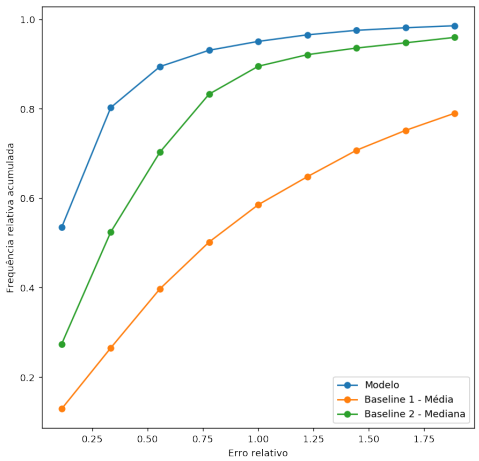
(e) Erro relativo para $n\_estimators = 40$



(f) Erro acumulado para $n\_estimators = 40$



(g) Erro relativo para $n\_estimators = 80$



(h) Erro acumulado para $n\_estimators = 80$

Fonte: Dados da pesquisa.

Diante da estabilidade dos resultados para diferentes valores de *n_estimators*, decidimos rodar os próximos experimentos definindo $n\_estimators = 20$.

5.2.2 Validação do modelo de estimativa de preço vs. baseline

Para validar o modelo de estimativa de preço obtido, utilizamos todas as arestas disponíveis na base de dados (aproximadamente 230.000).

Como baseline, calculamos o preço médio e o preço mediano dos produtos da base, sendo R$72,74 e R$39,25, respectivamente.

A base é composta por 229.338 arestas pertencentes a 9.153 nós (produtos) distintos.

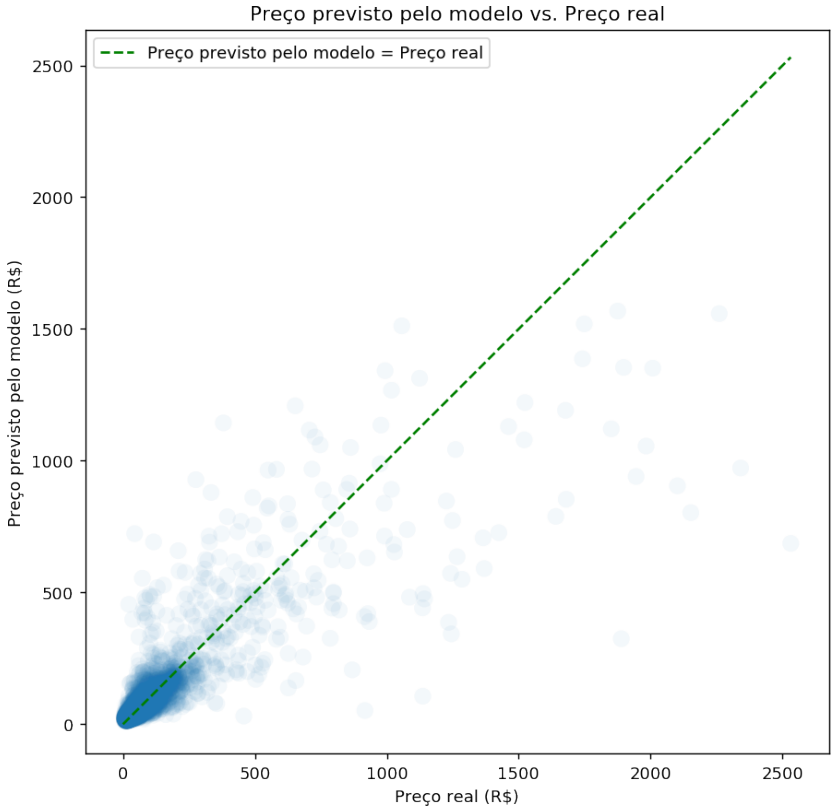
Treinamos o modelo utilizando cross-validation com k = 10 e obtivemos os scores da Tabela 6.

Tabela 6 – Comparação de erros relativos e absolutos para estimativa de preços utilizando todas as arestas (229.338) e $n\_estimators = 20$. Médias e desvios padrão foram calculados utilizando cross-validation com k = 10.

	Erro relativo (adimensional)		**Erro absoluto (R$)**	
	Média	**Desvio padrão**	**Média**	**Desvio padrão**
Modelo treinado	0.35	0.02	23.79	1.87
Baseline – Média	1.34	0.06	58.88	3.81
Baseline – Mediana	0.59	0.03	46.90	4.00

Uma maneira de avaliar a performance do modelo treinado é por inspeção visual, por meio de um gráfico de dispersão contendo pontos de coordenadas $(x; y)$ = (preço real conhecido; preço estimado pelo modelo) para todos os nós da base experimental (Figura 36).

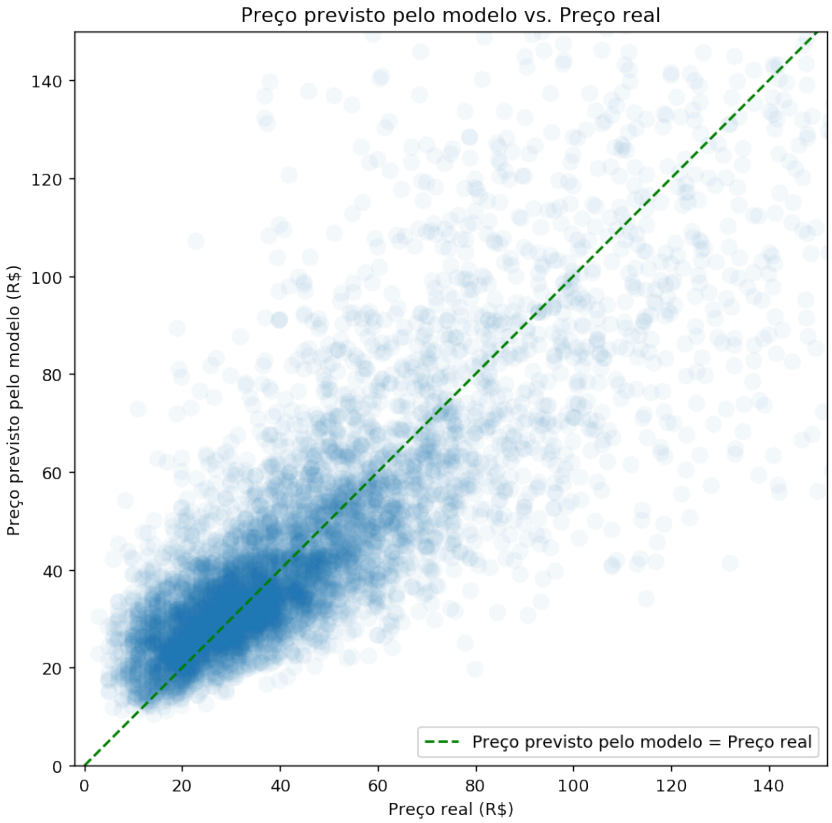
Figura 36 – Preço previsto pelo modelo vs. preço real conhecido para todos os nós da base experimental.



Fonte: Dados da pesquisa.

Na Figura 37 ampliamos o gráfico na região de preços até R$150,00, dado que a média e a mediana dos preços da base ocorrem nessa região.

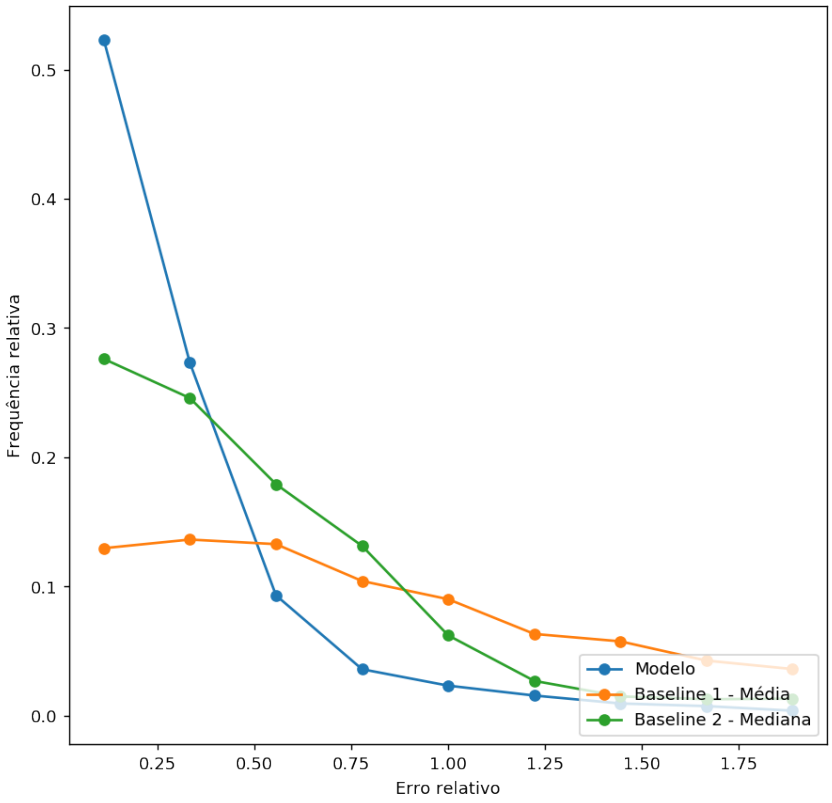
Figura 37 – Preço previsto pelo modelo vs. preço real conhecido para todos os nós da base experimental, com visualização ampliada na região de até R$150,00.



Fonte: Dados da pesquisa.

A comparação visual entre o modelo treinado e os baselines pode ser feita com auxílio da distribuição de erros relativos da Figura 38. Notamos que o modelo treinado possui frequência máxima de erros relativos na região entre 0 e 0,25 (representando mais de 50% dos exemplos da base).

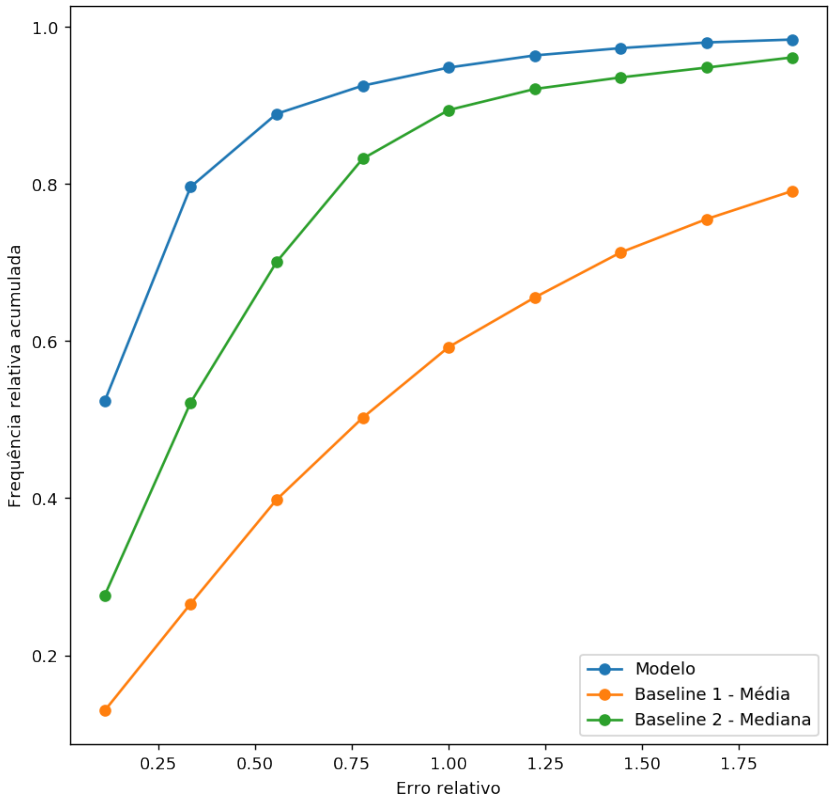
Figura 38 – Comparação de distribuição de erro relativo entre o modelo treinado e os baselines.



Fonte: Dados da pesquisa.

A diferença entre as performances do modelo treinado em comparação com os baselines pode ser mais facilmente validada por meio da Figura 39, que apresenta a frequência acumulada de erros relativos. Para interpretar esse gráfico, primeiramente analisamos o eixo *x* (Erro relativo) e escolhemos um valor arbitrário (exemplo: 0,50). Em seguida, corremos o olho imaginando uma linha vertical que cruza as 3 curvas, e anotamos o valor do eixo *y* (Frequência relativa acumulada) para todas as curvas. No exemplo de x = 0,50, encontramos valores aproximados para *y* de 0,3, 0,6 e 0,8 para as curvas Baseline 1, Baseline 2 e Modelo, respectivamente. Isso indica que 80% das estimativas de preço utilizando o modelo treinado apresentam erro relativo menor ou igual a 0,50 (50% do valor real do produto, para mais ou para menos), enquanto apenas 30% dos preços reais teriam erro relativo menor de 0,50 se utilizada a média como estimativa.

Figura 39 – Comparação da frequência acumulada de erro relativo entre o modelo treinado e os baselines.



Fonte: Dados da pesquisa.

5.2.3 Comparação entre diferentes conjuntos de atributos para o aprendizado do modelo

Neste próximo experimento vamos comparar o impacto da escolha de diferentes atributos (features) para o aprendizado do modelo.

Para isso, definimos 4 conjuntos de atributos:

- Todos os atributos
- Todos os atributos exceto métricas de rede
- Apenas métricas de rede
- Nenhum atributo

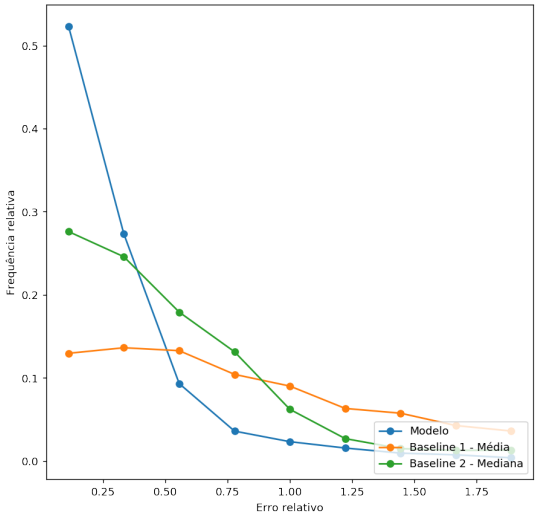
O último conjunto será utilizado como um baseline adicional.

Tabela 7 – Média e desvio padrão dos erros relativos e absolutos para 4 diferentes conjuntos de atributos (Todos os atributos; Todos os atributos exceto métricas de rede; Apenas métricas de rede; Nenhum atributo). Além disso, adicionamos na tabela as médias dos baselines para facilitar a comparação.

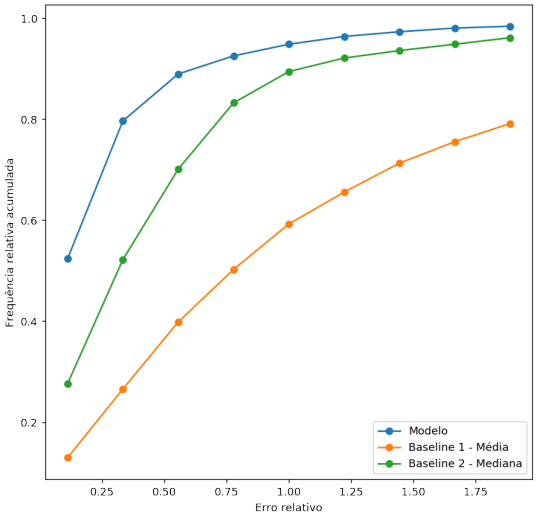
	Erro relativo (adimensional)		**Erro absoluto (R$)**	
	Média	**Desvio padrão**	**Média**	**Desvio padrão**
Todos os atributos	0.35	0.02	23.79	1.87
Todos exceto rede	0.35	0.02	23.45	3.82
Apenas rede	1.02	0.09	53.83	4.78
Nenhum atributo	1.50	0.18	70.07	6.47
Baseline – Média	1.34	0.06	58.88	3.81
Baseline – Mediana	0.59	0.03	46.90	4.00

Na Figura 40 exibimos a distribuição de erros relativos e da frequência acumulada de erros relativos para os 4 conjuntos de atributos definidos acima.

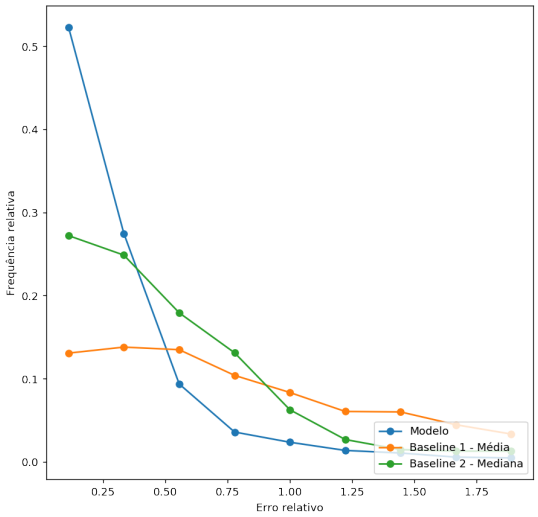
Figura 40 – Distribuição de frequência de erro relativo e frequência acumulada de erro relativo para diferentes conjuntos de atributos.



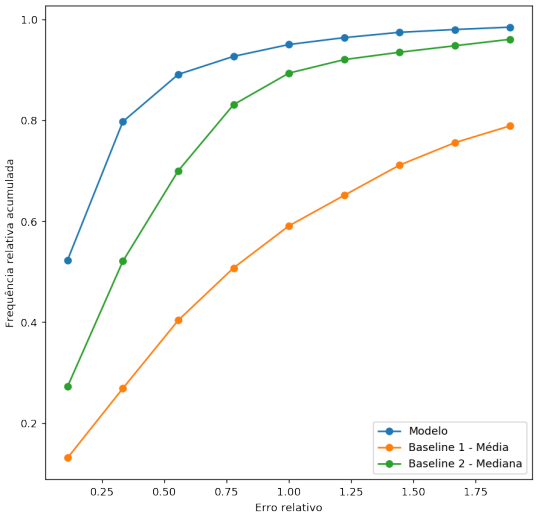
(a) Frequência de erro relativo utilizando todos os atributos



(b) Frequência acumulada de erro relativo utilizando todos os atributos



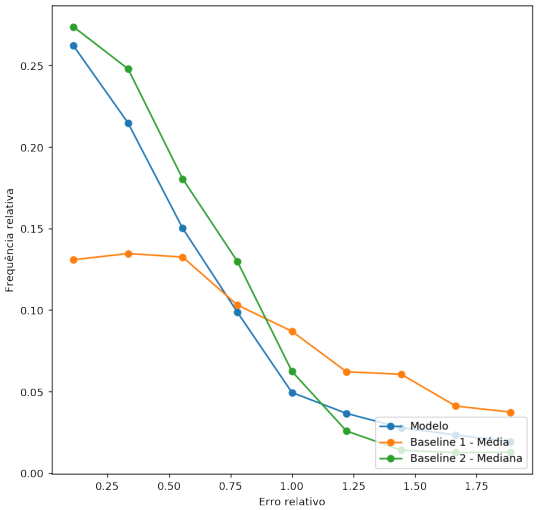
(c) Frequência de erro relativo utilizando todos os atributos exceto métricas de rede



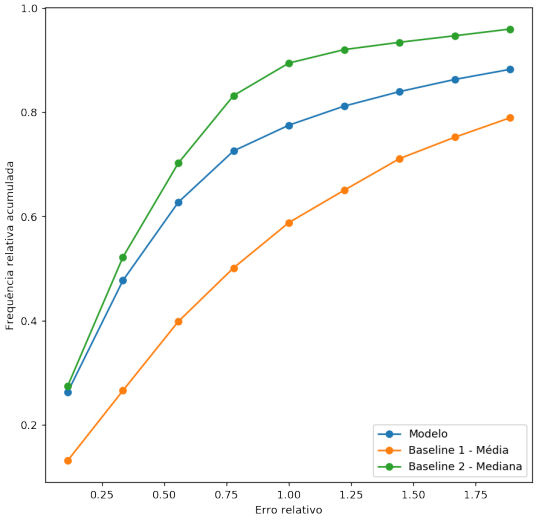
(d) Frequência acumulada de erro relativo utilizando todos os atributos exceto métricas de rede

Fonte: Dados da pesquisa.

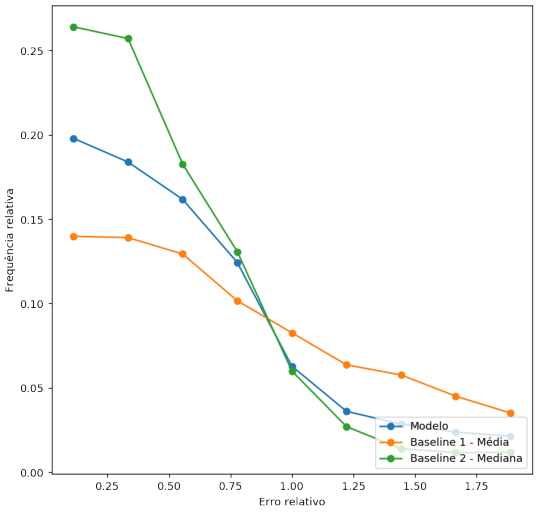
Figura 40 – (continuação) Distribuição de frequência de erro relativo e frequência acumulada de erro relativo para diferentes conjuntos de atributos.



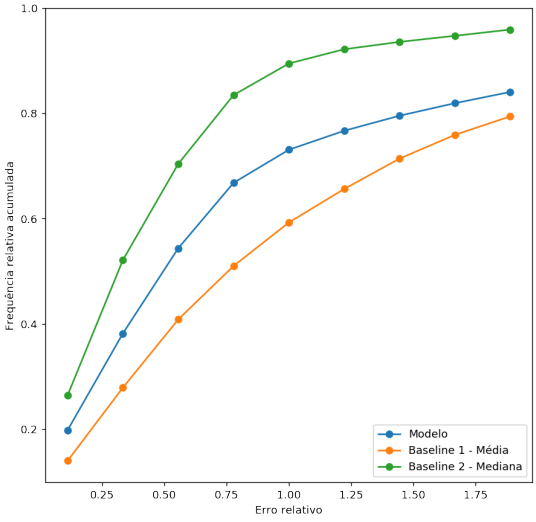
(e) Frequência de erro relativo utilizando apenas métricas de rede



(f) Frequência acumulada de erro relativo utilizando apenas métricas de rede



(g) Frequência de erro relativo utilizando nenhum atributo



(h) Frequência acumulada de erro relativo utilizando nenhum atributo

Fonte: Dados da pesquisa.

Analisando a Figura 40 vemos que utilizar apenas os atributos de rede não é suficiente para superar o Baseline 2 - Mediana na redução de erro relativo médio. Entretanto, vemos uma redução no erro relativo médio quando comparados os modelos treinados sem nenhum atributo em comparação com o conjunto de apenas métricas de rede, conseguindo superar o Baseline 1 - Média dos preços.

5.2.4 Atributos mais relevantes

Os atributos mais relevantes atribuídos pelo algoritmo Random Forest são listados na Tabela 8. Analisando os dados, percebemos que os preços que os 5 atributos com maior peso (número de páginas, idioma, categoria 2, largura e altura) correspondem, sozinhos, a mais de 50% do peso total do algoritmo, com a medida de centralidade de rede *eigenvector centrality* obtendo a sexta posição.

Isso mostra que o processo de precificação de livros possui dependência direta de suas propriedades físicas (número de páginas, altura e largura), além de forte peso do idioma da obra. Isso pode ser explicado pelo maior preço médio de livros vendidos em inglês, dados os custos de importação e conversão da cotação para a venda.

A métrica de rede melhor colocada no ranking de importância dos atributos é o *eigenvector centrality*, na sexta posição, que indica que medidas de rede podem ser tão ou mais importantes que outros atributos diretos como peso do livro, ranking e número de avaliações de clientes no comércio eletrônico.

Tabela 8 – Lista de atributos mais relevantes do modelo de estimativa de preços. Média e desvio padrão foram calculados a partir de cross-validation com $k = 10$.

Posição	**Atributo**	**Importância média (μ)**	**Desvio padrão (σ)**	**Desvio padrão relativo (σ/μ)**
1	pages	21.07	1.46	0.07
2	language_Inglês	18.44	0.75	0.04
3	category2_Inglês e Outras Línguas	11.06	1.92	0.17
4	width	5.85	0.99	0.17
5	height	5.60	0.64	0.11
6	eigenvector_centrality	4.54	0.77	0.17
7	publisher_McGraw-Hill Science/Engineering/Math	4.01	2.45	0.61
8	ranking	3.62	0.64	0.18
9	coverType_Capa dura	3.32	1.60	0.48
10	degree	1.60	0.22	0.14
11	depth	1.60	0.40	0.25
12	weight	1.55	0.23	0.15
13	sha256_id	1.47	0.19	0.13
14	publisher_Cengage Learning	1.38	0.75	0.54
15	betweenness_centrality	1.35	0.30	0.22
16	coverType_Capa comum	1.27	0.59	0.46
17	reviewCount	0.78	0.29	0.37
18	authors_Lars V. Ahlfors (Autor),	0.69	0.60	0.87
19	publisher_W. H. Freeman	0.53	0.55	1.03
20	authors_Jerrold E. Marsden (Autor),	0.46	0.26	0.56

5.3 Previsão de ligação entre nós

5.3.1 Comparação entre conjunto de atributos

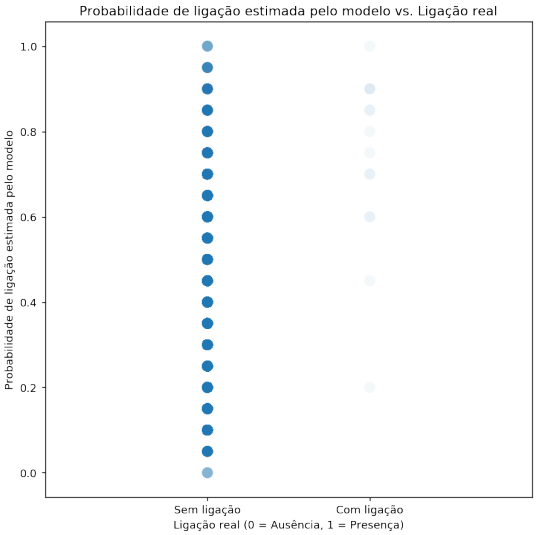
Quatro diferentes conjuntos de atributos foram utilizados para rodar o algoritmo de previsão de ligação com o nó 1.

- Todos os atributos
- Todos os atributos, exceto métricas de rede
- Apenas métricas de rede
- Nenhum atributo (baseline aleatório)

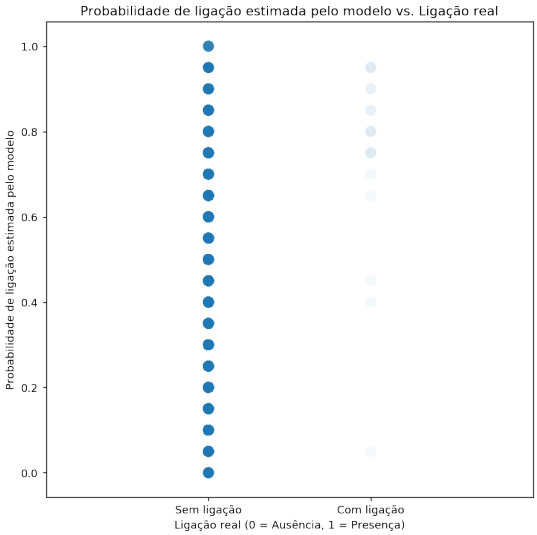
Após treinar 4 modelos diferentes, para cada um deles, o respectivo modelo fornece uma probabilidade entre 0 e 1 de que exista conexão entre um nó qualquer e o nó 1.

Primeiramente obtivemos 4 gráficos de dispersão (um para cada modelo) contendo pontos com coordenadas (x = classificação real, y = probabilidade estimada) (Figura 41).

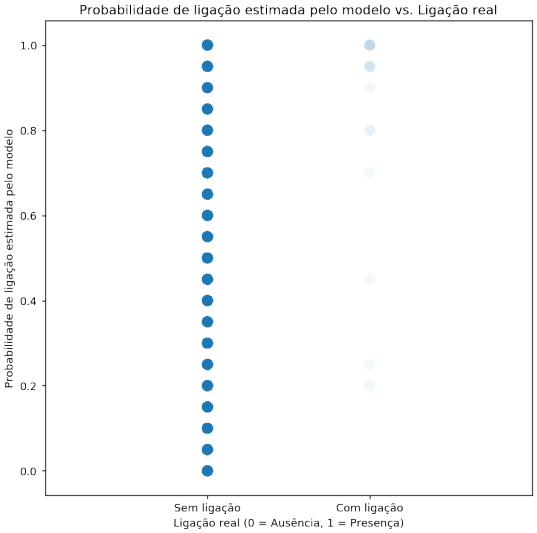
Figura 41 – Gráfico de dispersão com pontos posicionados na coordenada (x = classificação real, y = probabilidade estimada) para os 4 conjuntos de atributos.



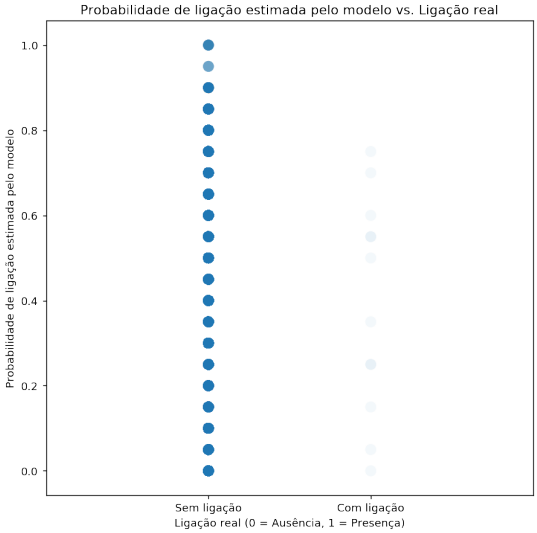
(a) Todos os atributos



(b) Todos os atributos exceto métricas de rede



(c) Apenas métricas de rede

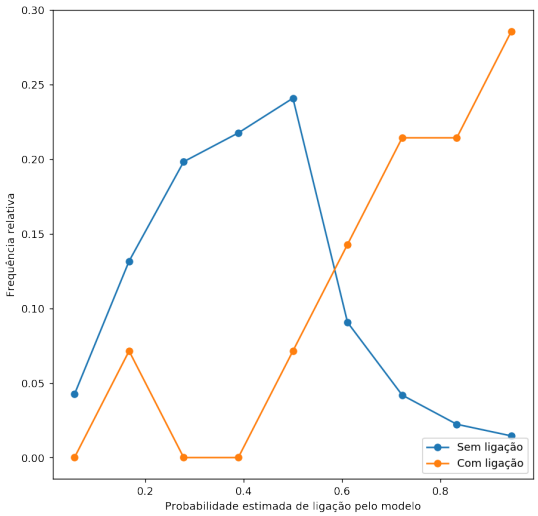


(d) Nenhum atributo

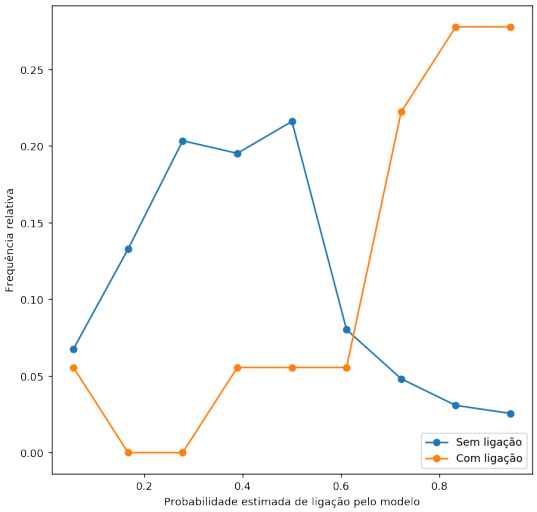
Fonte: Dados da pesquisa.

Os gráficos de dispersão da Figura 41, quando plotados na forma de distribuição de frequência para cada classe (sem ligação e com ligação) resultam na Figura 42.

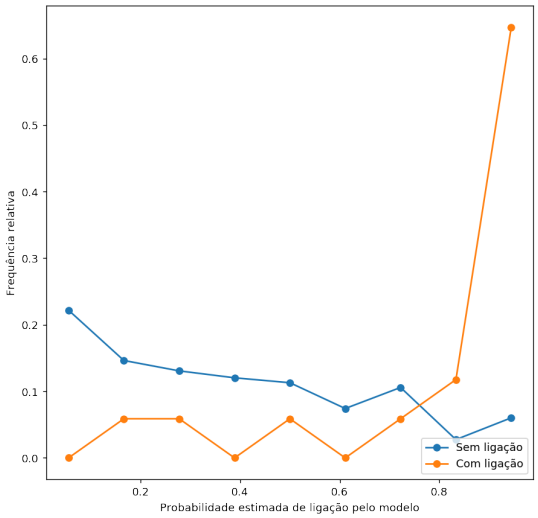
Figura 42 – Distribuição de frequência em função da classe (com ligação e sem ligação) para os 4 conjuntos de atributos.



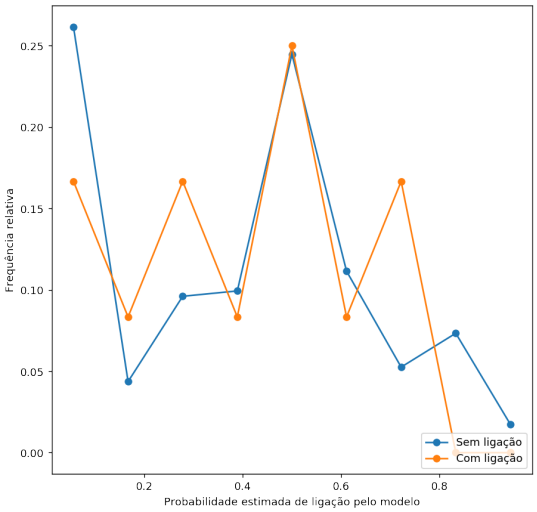
(a) Todos os atributos



(b) Todos os atributos exceto métricas de rede



(c) Apenas métricas de rede

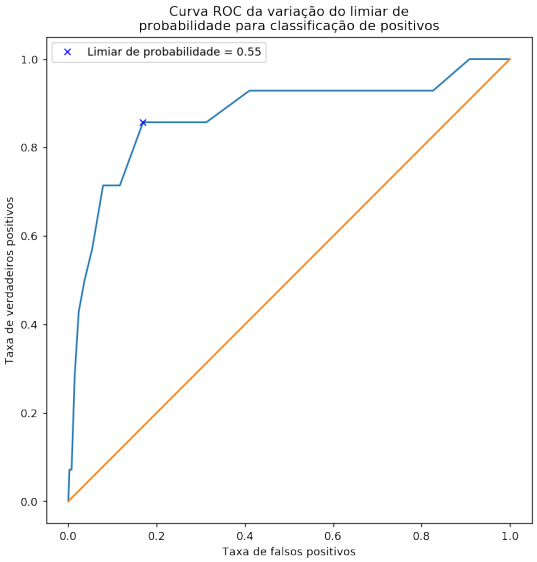


(d) Nenhum atributo

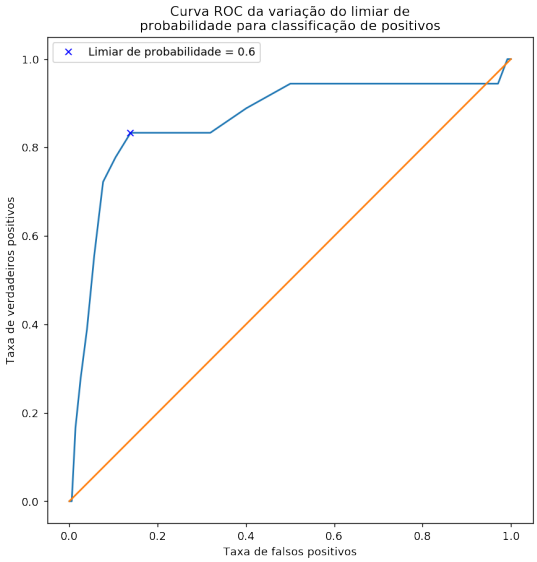
Fonte: Dados da pesquisa.

Com base nas distribuições de frequência da Figura 42, foi possível criar as curvas ROC para cada um dos experimentos, variando-se a probabilidade de corte (threshold) e calculando as taxas de verdadeiros positivos e falsos positivos.

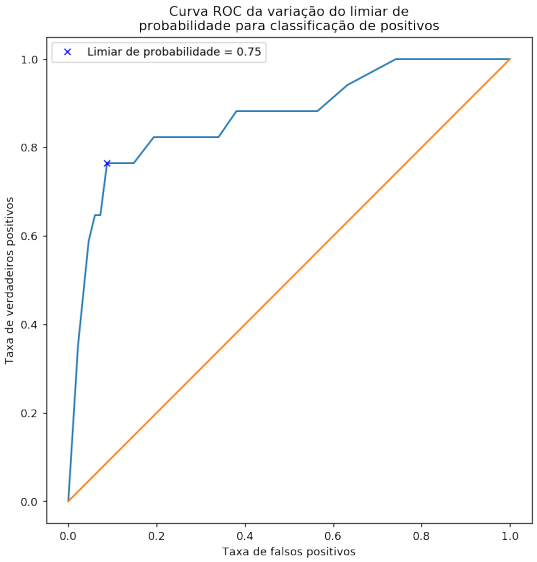
Figura 43 – Curvas ROC para os 4 conjuntos de atributos.



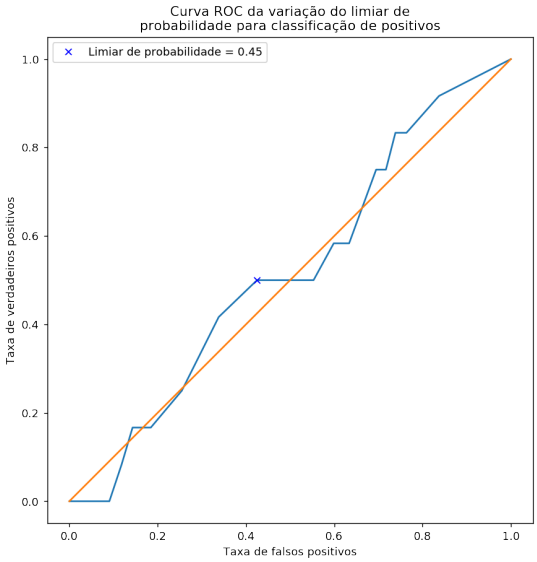
(a) Todos os atributos



(b) Todos os atributos exceto métricas de rede



(c) Apenas métricas de rede

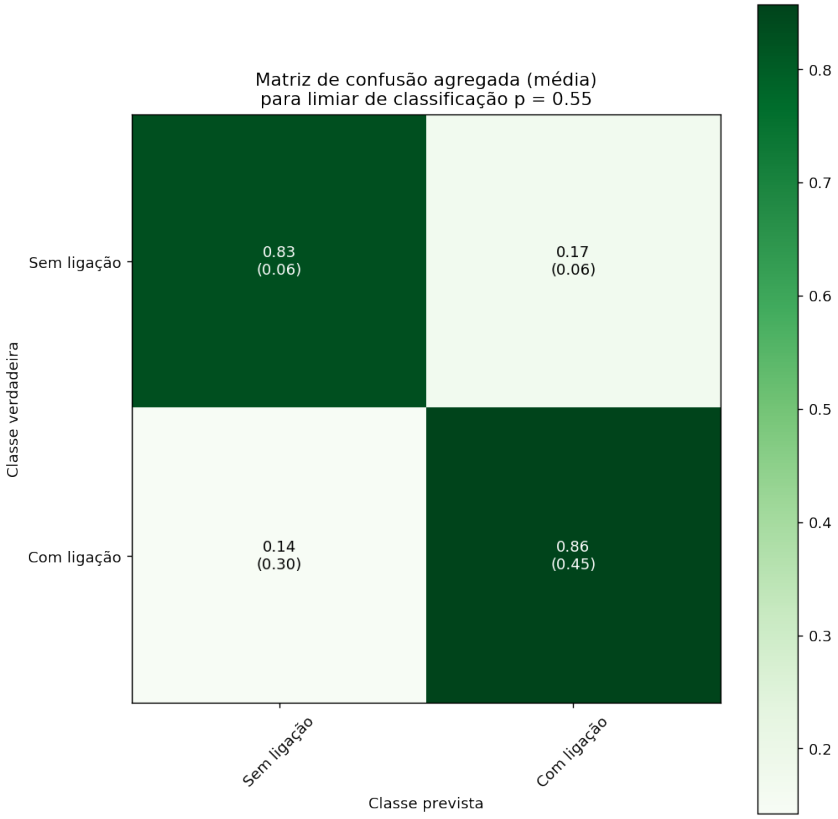


(d) Nenhum atributo

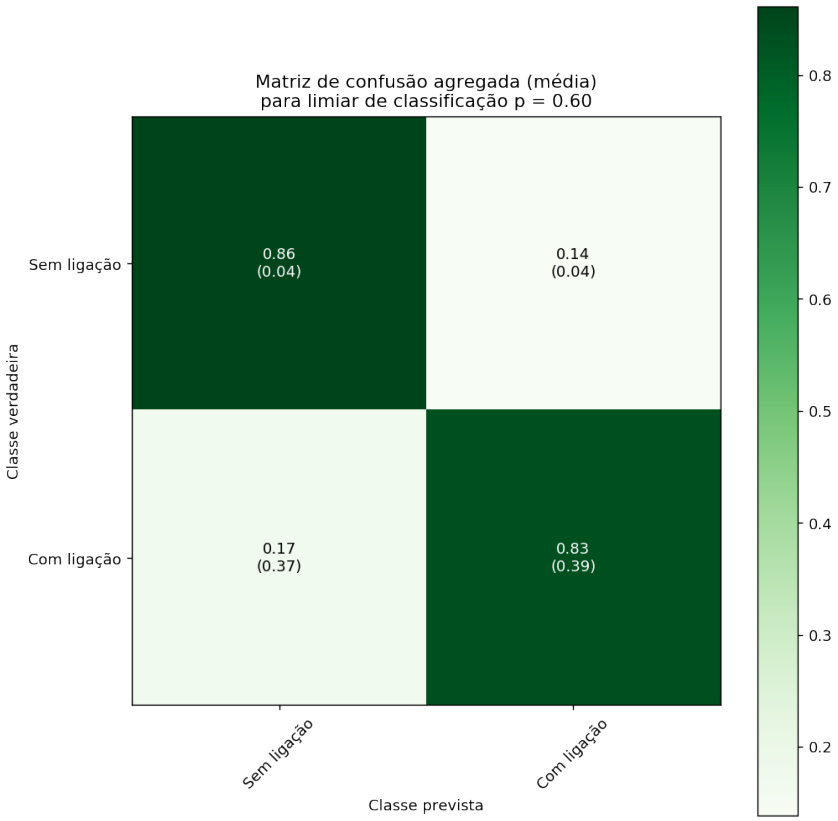
Fonte: Dados da pesquisa.

Utilizando as curvas ROC (Figura 43) para obter a melhor probabilidade de corte, calculamos a matriz de confusão para cada um dos cenários (Figura 44).

Figura 44 – Matrizes de confusão para os 4 conjuntos de atributos.



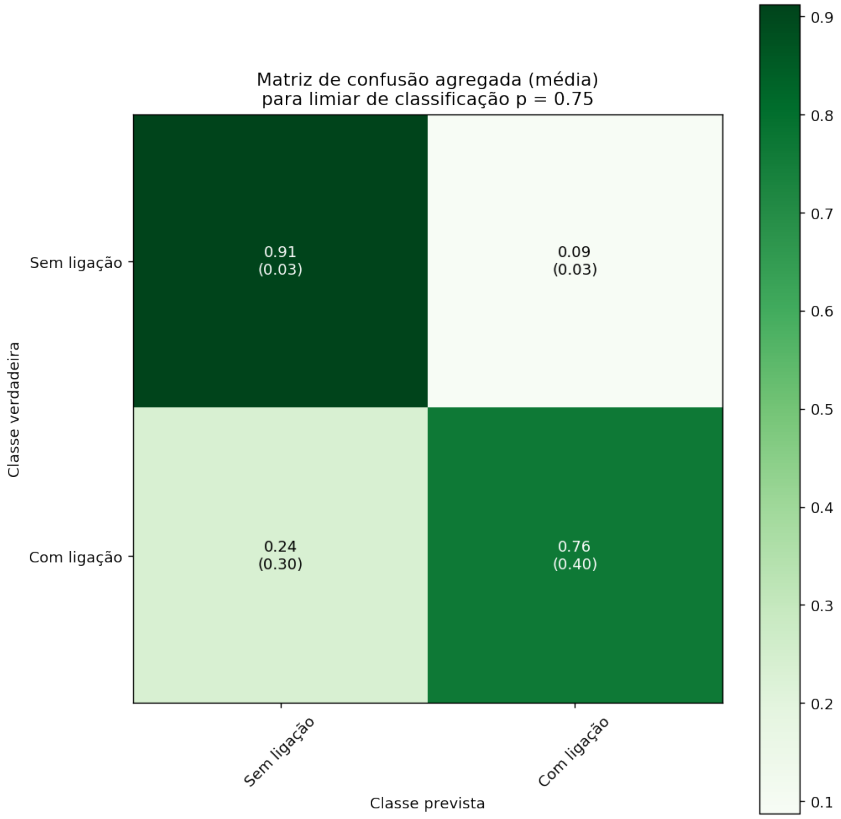
(a) Todos os atributos



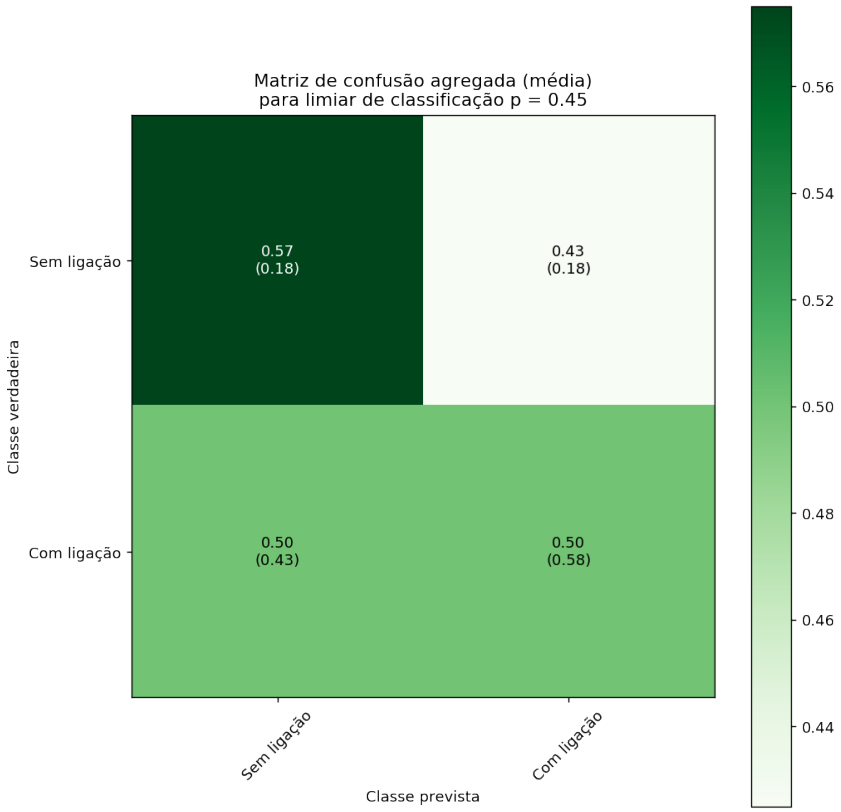
(b) Todos os atributos exceto métricas de rede

Fonte: Dados da pesquisa.

Figura 44 – (continuação) Matrizes de confusão para os 4 conjuntos de atributos.



(c) Apenas métricas de rede



(d) Nenhum atributo

Fonte: Dados da pesquisa.

Observando os resultados das Figuras 42-44, percebemos que a utilização de apenas métricas de rede como atributos de aprendizado são suficientes para separar de forma suficientemente clara as duas classes do treinamento. Particularmente, evidencia-se, comparando as curvas ROC (c) e (d) da Figura 43, a abrupta capacidade de separação de classes aparente quando passamos de um treinamento sem nenhum atributo (d), para um treinamento utilizando apenas as três métricas de rede (c).

CAPÍTULO

6

CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho, identificamos que não existe correlação positiva entre a avaliação de consumidores de um produto e o grau de seu nó da rede. Isto é, *hubs* da rede não são, necessariamente, melhores avaliados. Por outro lado, confirmamos a correlação de que *hubs* da rede são melhores posicionados no *ranking* de produtos mais vendidos.

Confirmamos a hipótese de que a utilização exclusiva dos atributos de métricas de rede são suficientes para o treinamento de modelos de aprendizado de máquina com eficiência superior à do *baseline* para o modelo preditivo de previsão de ligação entre nós.

Este é um resultado importante, pois modelos de previsão de ligação entre nós podem ser utilizados em algoritmos de recomendação de produtos, de alto valor comercial e científico.

Entretanto, somente tais atributos não foram suficientes para superar o *baseline* do modelo de precificação de produtos.

Finalmente, de forma a contribuir com ferramentas para pesquisas futuras, desenvolvemos e disponibilizamos no formato de código livre um sistema *webcrawler* capaz de coletar dados de até 1000 produtos por hora, a partir de uma grande plataforma de comércio eletrônico.

6.0.1 Trabalhos futuros

Os resultados obtidos evidenciam a importância da utilização de métricas de rede como atributos de aprendizado de modelos preditivos, principalmente no que tange à previsão de ligação entre nós e outros algoritmos de classificação similares.

Pesquisas futuras serão capazes de esclarecer a natureza da importância dessas métricas para o modelo, além de sugerir algoritmos mais eficientes que possibilitem a execução da previsão de ligação entre múltiplos (ou todos) os nós da rede paralelamente, superando a limitação deste trabalho, que se propôs a prever a ligação de nós arbitrários a um único nó de referência.

Um possível questionamento para trabalhos futuros é por que os *hubs* da rede de produtos

comprados em conjunto não são melhores avaliados em comparação a produtos periféricos. Uma hipótese a ser testada é a de que produtos menos centrais à rede são avaliados por uma quantidade pequena de usuários, concedendo maior peso a avaliações artificiais enviadas por autores, editoras, e seus conhecidos, efetivamente resultando em uma avaliação enviesada.

Outra linha possível de pesquisa é o estudo da informação contida nas medidas de centralidade, e a determinação de por que essas medidas são suficientes para o treinamento de modelos de aprendizado de máquina de previsão de ligação entre nós. Identificamos que utilizar apenas três métricas de rede (grau, *betweenness centrality* e *eigenvector centrality*) é suficiente para a obtenção de performance quase equiparável à obtida com o conjunto completo de atributos da rede. Entretanto, tal conjunto reduzido de atributos não possui a mesma informação do que a contida no conjunto completo, o que fica claro quando observamos que, para o treinamento do modelo de precificação, as métricas de rede, sozinhas, não foram suficientes para gerar um modelo que superasse o *baseline*.

Além disso, variações da metodologia desta pesquisa, como alteração da plataforma de comércio eletrônico utilizada para coletar as informações dos produtos, e também a obtenção de dados de outros tipos de produtos, podem trazer resultados mais generalizados do que os apresentados neste trabalho.

REFERÊNCIAS

ALBERT; JEONG; BARABASI. Error and attack tolerance of complex networks. **Nature**, v. 406, n. 6794, p. 378–382, Jul 2000. Disponível em: <http://www.pubmed.org/10935628>. Citado na página 36.

ALBERT, R.; BARABÁSI, A.-L. Statistical mechanics of complex networks. **Reviews of Modern Physics**, v. 74, p. 47–97, Jan 2002. Disponível em: <http://adsabs.harvard.edu/abs/2002RvMP...74...47A>. Citado na página 36.

Amazon Serviços de Varejo do Brasil Ltda. **Fourier Series - Livros na Amazon Brasil-9780486633176**. 2019. Disponível em: <https://www.amazon.com.br/dp/0486633179/>. Citado nas páginas 56 e 57.

BARABÁSI, A.-L. Network science book. **PDF version of 2014-09-05**, 2012. Citado nas páginas 39 e 49.

BARABÁSI, A.-L.; ALBERT, R. Emergence of scaling in random networks. **Science**, v. 286, p. 509–512, Oct 1999. Disponível em: <http://adsabs.harvard.edu/abs/1999Sci...286..509B>. Citado nas páginas 31, 36, 50 e 51.

BARABÁSI, A.-L.; RAVASZ, E.; VICSEK, T. Deterministic scale-free networks. **Physica A: Statistical Mechanics and its Applications**, v. 299, p. 559–564, Oct 2001. Disponível em: <http://adsabs.harvard.edu/abs/2001PhyA..299..559B>. Citado na página 36.

BELADEV, M.; ROKACH, L.; SHAPIRA, B. Recommender systems for product bundling. **Knowledge-Based Systems**, v. 111, p. 193–206, Jan 2016. Citado na página 37.

BIANCONI, G.; BARABÁSI, A. L. Bose-einstein condensation in complex networks. **Phys Rev Lett**, v. 86, n. 24, p. 5632–5635, Jun 2001. Disponível em: <http://www.pubmed.org/11415319>. Citado na página 36.

BRAINERD, J.; BECKER, B. Case study: E-commerce clickstream visualization. 2001. Citado na página 36.

BREESE, J.; HECKERMAN, D.; KADIE, C. Empirical analysis of predictive algorithms for collaborative filtering. 1998. Citado na página 35.

CHEN, H.; LYNCH, K. J. Automatic construction of networks of concepts characterizing document databases. 1992. Disponível em: <http://hdl.handle.net/10150/105175>. Citado na página 35.

CHEN, L.; WANG, F. Preference-based clustering reviews for augmenting e-commerce recommendation. **Knowledge-Based Systems**, v. 50, p. 44–59, Jan 2013. Citado na página 37.

COSTAGLIOLA, G.; FUCCELLA, V.; PASCUCCIO, F. A. Towards a trust, reputation and recommendation meta model. **Journal of Visual Languages & Computing**, v. 25, p. 850–857, Jan 2014. Citado na página 37.

DESARBO, W. S.; WEDEL, M.; VRIENS, M.; RAMASWAMY, V. Latent class metric conjoint analysis. **Marketing Letters**, v. 3, n. 3, p. 273–288, Jan 1992. Citado na página 36.

DIESTEL, R. Graduate texts in mathematics: Graph theory. Springer, 2000. Citado nas páginas 36 e 39.

ERDEM, T. Brand and quantity choice dynamics under price uncertainty. **Quantitative Marketing and Economics**, v. 1, n. 1, p. 5, Jan 2003. Citado na página 36.

HARARY, F. Graph theory. 1969. Citado na página 35.

____. Shortest paths in probabilistic graphs. **Operations Research**, v. 17, n. 4, p. 583–599, Aug 1969. Citado na página 35.

HESS, T.; TITAH, R.; BENLIAN, A. Differential effects of provider recommendations and consumer reviews in e-commerce transactions: An experimental study. **Journal of Management Information Systems**, v. 29, p. 237–272, Jan 2012. Citado na página 37.

HUANG, Z.; CHUNG, W.; CHEN, H. A graph model for e-commerce recommender systems. 2003. Citado na página 36.

JANNACH, D.; RESNICK, P.; TUZHILIN, A.; ZANKER, M. Recommender systems - beyond matrix completion. **Communications of the ACM**, v. 59, n. 11, p. 94, Jan 2016. Citado nas páginas 35 e 37.

JOHRI, V.; BANSAL, S. Identifying trends in technologies and programming languages using topic modeling. p. 391–396, Jan 2018. Citado na página 64.

KÖNI, D. Theorie der endlichen und unendlichen graphen: Kombinatorische topologie der streckenkomplexe. 1936. Citado na página 35.

LI, B.; GHOSE, A.; IPEIROTIS, P. G. Towards a theory model for product search. p. 327, Jan 2011. Citado na página 37.

LI, H. B.; WANG, W.; DING, H. W.; DONG, J. Trees weighting random forest method for classifying high-dimensional noisy data. p. 160–163, Jan 2010. Citado na página 93.

LI, X.; CHEN, H. Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach. **Decision Support Systems**, v. 54, n. 2, p. 880–890, Jan 2013. Citado na página 37.

LI, Y.-M.; WU, C.-T.; LAI, C.-Y. A social recommender mechanism for e-commerce: Combining similarity, trust, and relationship. **Decision Support Systems**, v. 55, p. 740–752, Jan 2013. Citado na página 37.

METZ, J.; CALVO, R.; SENO, E. R.; ROMERO, R. A.; LIANG, Z. Redes complexas: conceitos e aplicações. **Relatórios Técnicos do ICMC-USP São Carlos**, 2007. Citado nas páginas 39 e 52.

MIRZA, B. J.; KELLER, B. J.; RAMAKRISHNAN, N. Studying recommendation algorithms by graph analysis. **Journal of Intelligent Information Systems**, v. 20, n. 2, p. 131–160, Jan 2003. Citado na página 36.

NEWMAN, M. The physics of networks. **Physics Today**, v. 61, n. 11, p. 33–38, Jan 2008. Citado na página 36.

_____. Networks: an introduction. Oxford university press, 2010. Citado na página 39.

NEWMAN, M. E.; STROGATZ, S. H.; WATTS, D. J. Random graphs with arbitrary degree distributions and their applications. **Phys Rev E Stat Nonlin Soft Matter Phys**, v. 64, n. 2 Pt 2, p. 026118, Aug 2001. Disponível em: <http://www.pubmed.org/11497662>. Citado na página 36.

NEWMAN, M. E. J. The structure and function of complex networks. **Soiety for Industrial and Applied Mathetatics**, Jan 2003. Citado nas páginas 36 e 52.

OESTREICHER-SINGER, G.; SUNDARARAJAN, A. Linking network structure to ecommerce demand: Theory and evidence from amazon.com's copurchase network. Aug 2006. Citado na página 36.

RESNICK, P.; VARIAN, H. R. Recommender systems. Mar 1997. Citado na página 35.

RUSSELL, G. J.; KAMAKURA, W. A. Modeling multiple category brand preference with household basket data. **Journal of Retailing**, v. 73, n. 4, p. 439–461, Jan 1997. Citado na página 36.

SRIVASTAVA, A. Motif analysis in the amazon product co-purchasing network. Dec 2010. Disponível em: <http://arxiv.org/abs/1012.4050>. Citado nas páginas 31 e 37.

THEUSINGER, C.; HUBER, K.-P. Analyzing the footsteps of your customers. **Proc. of the Sixth ACM SIGKDD Internat. Conf. on Web KDD 2000**, 2000. Citado na página 36.

TIAN, Z.; ZHANG, Z.; GAO, R. Optimization in e-commerce market network based on value order parameter. **Information Technology and Management**, v. 17, p. 187–197, Jan 2015. Citado na página 37.

TIAN, Z.; ZHANG, Z.; GUAN, X. A new evolution model for b2c e-commerce market. **Information Technology and Management**, v. 14, p. 205–215, Jan 2013. Citado na página 37.

WANG, G.; GUI, X. Dynamic recommendation trust model based on information entropy and heuristic rules in e-commerce environment. **Electronics and Electrical Engineering**, v. 19, n. 4, Jan 2013. Citado na página 37.

WANG, Y.; YIN, G.; CAI, Z.; DONG, Y.; DONG, H. A trust-based probabilistic recommendation model for social networks. **Journal of Network and Computer Applications**, v. 55, p. 59–67, Jan 2015. Citado nas páginas 37 e 53.

WATTS, D. J.; STROGATZ, S. H. Collective dynamics of 'small-world' networks. **Nature**, v. 393, n. 6684, p. 440–442, Jun 1998. Disponível em: <http://www.pubmed.org/9623998>. Citado nas páginas 31, 36 e 50.

YAN, B.-N.; LEE, T.-S.; LEE, T.-P. Analysis of research papers on e-commerce (2000–2013): based on a text mining approach. **Scientometrics**, v. 105, p. 403–417, Jan 2015. Citado na página 37.

YANG, J.; LESKOVEC, J. Defining and evaluating network communities based on ground-truth. May 2012. Disponível em: <http://arxiv.org/abs/1205.6233>. Citado na página 31.

ZHONG, H.; ZHANG, S.; WANG, Y.; SHU, Y. Study on directed trust graph based recommendation for e-commerce system. **International Journal of Computers Communications & Control**, v. 9, n. 4, p. 510, Jan 2014. 10.15837/ijccc.2014.4.228. Citado na página 37.

GLOSSÁRIO

Adição sucessiva: Regra do modelo de Barabási-Albert que dita que os nós devem ser adicionados incrementalmente, de um em um (ver Seção 3.10.3).

Ajuste ergódico: Ajuste utilizado no algoritmo do PageRank para evitar que a rotina iterativa não fique presa dentro de um grupo de nós absorventes, adicionando aos elementos da matriz de transição original uma probabilidade de haver uma transição de saída do grupo.

Ajuste estocástico: Ajuste utilizado no algoritmo do PageRank para que a soma das probabilidades de saída de um nó seja sempre 1, mesmo nos casos de nós que não possuem ligação de saída.

Algoritmo: Rotina computacional executada por um computador.

Anaconda: Pacote de softwares que contém o interpretador Python e uma gama de bibliotecas científicas, possibilitando o processamento de matrizes, gráficos, redes complexas, arquivos de dados, e diversas outras aplicações.

API: Conjunto documentado de comandos (funções, métodos) de um software, exposto externamente ao próprio software, com o intuito de possibilitar a integração entre *softwares* desenvolvidos por terceiros.

Application Programming Interface: (ver API).

Aprendizado de máquina: Área de estudo da ciência da computação que visa a obtenção de modelos preditivos capazes de classificar, estimar e detectar comunidades a partir de um conjunto de dados de treinamento.

Area under curve: (ver AUC).

Aresta (de um grafo): Ligação entre dois nós de um grafo.

Assortatividade: Medida de rede que representa a correlação entre graus de vizinhos, isto é, a probabilidade de existir ligações entre nós de diferentes graus.

Atributo (de uma base de dados de aprendizado de máquina): Classe de informação pertencente a um dado registro de uma base de dados. Na base de dados de produtos comprados em conjunto da plataforma de comércio eletrônico, temos, entre outros atributos: Nome do produto, Preço do produto, Autor do produto.

Atributo categórico: Tipo de atributo que não possui relação ordinal (ver Atributo). Exemplos: sexo, cor.

Atributo numérico: Tipo de atributo cuja informação é representada por um conjunto numérico, como os números naturais ou reais (ver Atributo).

AUC: Área total calculada sob a curva ROC, compreendida entre 0 e 1. Quanto maior a área sob a curva, maior é a qualidade de um classificador binário.

Autovalor: Conceito de álgebra linear cuja definição foge ao escopo deste trabalho.

Autovetor: Conceito de álgebra linear cuja definição foge ao escopo deste trabalho.

Banco de dados relacional: Tipo específico de banco de dados em que os dados são tipicamente normalizados (em contraste com bancos de dados NoSQL).

Banco de dados: Pode se referir a um arquivo contendo dados estruturados de forma análoga a uma planilha eletrônica, ou a um software capaz de processar as informações desse arquivo (também chamado de servidor de banco de dados).

Base (de dados): Arquivo contendo dados estruturados de forma análoga a uma planilha eletrônica.

Base de teste: Parte de uma base de dados utilizada para a etapa de teste de um processo de aprendizado de máquina.

Base de treinamento: Parte de uma base de dados utilizada para a etapa de treinamento de um processo de aprendizado de máquina.

Baseline: Padrão a ser utilizado como critério de avaliação de performance de um modelo. Normalmente esse padrão é facilmente calculado, facilitando a sua reprodução por outros pesquisadores.

Bash: Linguagem de programação usualmente pré-instalada em computadores com sistema operacional Linux.

Betweenness centrality: Um tipo de medida de centralidade de rede.

Biblioteca (de programação): Pacote de classes, métodos e funções de software que permite disponibilizar e reaproveitar funcionalidades previamente programadas a novos algoritmos.

Bot: Algoritmo computacional que simula sequências de ações tipicamente humana.

Busca em largura: Algoritmo de varredura em que itens de classificação hierárquica $k+1$ só serão processados após a finalização do processamento de todos os itens de classificação k (em contraste com busca em profundidade).

Cache: Armazenamento temporário de dados, normalmente visando ao aumento da velocidade de processamento de um algoritmo. Pode ocorrer a nível de hardware (processador, memória ou disco rígido) ou a nível de software.

Caracterização de rede: Processo de obtenção de características básicas de uma rede, como seu grau médio, quantidade de nós, tamanho do maior componente, etc.

Classificador: Modelo/algoritmo de aprendizado de máquina utilizado para resolver problemas de classificação.

Classificação (categoria de problema de aprendizado de máquina): Família de problemas de aprendizado de máquina em que se deseja separar os registros de uma base de dados em diversas classes distintas.

Classificação binária: Família de problemas de aprendizado de máquina em que se deseja separar os registros de uma base de dados em duas classes distintas. Exemplo: problema de detecção de fraude em transações financeiras, em que as possíveis classificações são É FRAUDE e NÃO É FRAUDE.

Cliente: Em computação, normalmente se refere ao usuário, ou ao software utilizado pelo usuário, de determinado serviço disponibilizado por um computador servidor (ver Servidor).

Closeness centrality: Um tipo de medida de centralidade de rede.

Cluster: Região de uma rede com alta concentração mútua de ligações, em que vizinhos de um nó qualquer possuem alta probabilidade de estarem ligados entre si.

Coeficiente Gini: Coeficiente que mede a heterogeneidade de um dado conjunto, com valores entre 0 e 1, sendo 0 a condição de total homogeneidade.

Complexidade computacional: Medida de tempo de execução de um algoritmo baseada na quantidade de operações realizadas, normalmente apresentadas com a notação "Big O".

Componente (de rede): Conjunto de nós, conectados entre si, que estão desconectados de todos os outros nós da rede.

Composer: Software de gerenciamento de dependências para projetos PHP.

Comunidade (de rede): Regiões de uma rede com alta clusterização local, e fracamente conectadas a outras regiões.

Cross-validation (validação cruzada): Método utilizado para treinamento de modelos de aprendizado de máquina em que a base de é dividida em K partes, sendo que cada registro da base é utilizado uma única vez como parte do conjunto de teste, e $K-1$ vezes como parte do conjunto de treinamento.

CSV: Tipo simples de arquivo de dados, em que, tipicamente, diferentes registros são separado por quebras de linha, e diferentes atributos são separados por vírgulas dentro de uma mesma linha. Outras variações podem utilizar "ponto e vírgula"ou o caracter TAB como separador de atributos.

Curva ROC: Curva construída ao se variar a sensibilidade de um modelo de classificação, obtendo-se a especifidade do modelo para cada valor de sensibilidade.

Código livre (open source): Software, programa de computador, cujo código é disponibilizado publicamente.

Daemon: Software que roda em segundo plano continuamente, tipicamente para realizar tarefas de duração indeterminada.

Dataframe: Terminologia utilizada pela biblioteca Pandas, da linguagem de programação Python, para se referir a conjuntos de dados estruturados.

Distribuição Linux Debian 9: Sistema operacional baseado em Linux, uma alternativa de código livre ao Microsoft Windows.

Distância geodésica (menor caminho): Tamanho do menor caminho que liga dois nós de uma rede.

Diâmetro de rede: Maior distância dentre todos os menores caminhos de uma rede.

Doctrine ORM: Biblioteca da linguagem de programação PHP que mapeia registros de um banco de dados relacional para objetos da memória.

Eigenvector centrality: Um tipo de medida de centralidade de rede.

Epinions: Website de resenhas e avaliações de produtos escritas por consumidores, hoje não mais em funcionamento.

Especificidade: Medida da capacidade de um classificador em separar verdadeiros negativos (VN) de falsos positivos (FP), dada por $VN/(VN+FP)$.

Estimador: No contexto do modelo Random Forest, é uma árvore de decisão que compõe o modelo. O modelo é composto por múltiplos estimadores.

Fab: Algoritmo de filtro colaborativo.

Facebook Webdriver: Ferramenta disponibilizada publicamente pela equipe de desenvolvedores da empresa Facebook para controlar o framework de testes automatizados Selenium por meio da linguagem de programação PHP.

Falso positivo: No contexto de classificação binária, se trata de um registro que foi erroneamente classificado como positivo.

Filtro colaborativo: Tipo de sistema de recomendação baseado em matriz de similaridade.

GET: Verbo do protocolo HTTP utilizado para recuperar documentos de uma dada URL.

Gini: (ver Coeficiente Gini).

GitHub: Rede social de projetos de código livre, que possibilita o controle de versão e participação colaborativa entre programadores de software.

Google Scholar: Ferramenta de busca de publicações acadêmicas.

Grafo: Representação matemática de uma rede.

Grau: Quantidade de vizinhos de um nó de uma rede.

GroupLens: Algoritmo de filtro colaborativo.

HTML: *Hypertext Transfer Protocol*. Linguagem de marcação utilizada por *websites* para agrupar elementos diversos (imagens, textos, tabelas) e exibi-los na forma de um único documento em um navegador de *internet*.

HTTP: Protocolo de comunicação entre clientes e servidores, tipicamente utilizado por navegadores de internet para recuperar páginas da internet.

Hub: Nó de uma rede que possui grau muito maior do que o grau médio da rede.

Injeção de dependência: Mecanismo de engenharia de software que possibilita que as dependências de uma classe de um software sejam determinadas em tempo de execução, com base nas interfaces das classes envolvidas e na configuração do software.

Input: No contexto de software, é a entrada de dados do usuário.

Inspeção visual: Análise de resultados que se dá pela minuciosa observação visual de um gráfico, buscando identificar padrões.

Internet: Rede mundial de computadores.

Interpretador: Software capaz de interpretar, isto é, executar um algoritmo de uma linguagem de programação que não é compilada.

Invariância à escala (scale-free): Propriedade encontrada em redes reais que seguem a distribuição de lei de potência que dita que, independentemente do fator de escala, a rede será composta por poucos nós de grau alto e muitos nós de grau baixo.

Inversão de controle: Padrão de engenharia de software descrito pela inversão do controle do fluxo de chamadas de métodos. Um exemplo de inversão é a injeção de dependência (ver Injeção de Dependência).

Java: Nome de uma linguagem de programação e do interpretador da respectiva linguagem.

Jupyter: Software de cadernos virtuais para Python e outras linguagens de programação.

Lei de potência (distribuição de grau): (ver Lei de potência).

Lei de potência: (ver Seção).

Ligação preferencial: Regra do modelo de Barabási-Albert que dita que a probabilidade de um nó receber uma nova ligação é proporcional ao seu grau (ver Seção 3.10.3).

Linha de comando: Interface de interação com o sistema operacional em que o usuário do sistema executa operações por meio da digitação de comandos (em contraste com interfaces visuais, em que o usuário utiliza utiliza o mouse para apontar e clicar em gatilhos de operações), proeminente em sistemas Linux.

Linux: Sistema operacional de código livre, alternativa gratuita ao sistema operacional Microsoft Windows.

Log: Arquivo de eventos ocorridos ao longo da execução de um software, normalmente contendo a data, o horário e a descrição do evento ocorrido.

Loop: Seção de um algoritmo que é executada indefinidamente, enquanto as condições de repetição estiverem satisfeitas.

Maior componente de rede: Dentre os componentes da rede, aquele que possui a maior quantidade de nós (ver Componente).

Matriz adjacência: Matriz utilizada para a representação dos nós de uma rede e suas ligações.

Matriz de confusão: Matriz utilizada para representar a performance de um modelo preditivo por meio das quantidades de verdadeiros positivos (VP), verdadeiros negativos (VN), falsos positivos (FP) e falsos negativos (FN).

Matriz de transição: Matriz utilizada por processos iterativos, composta por elementos A_{ij} que detonam a probabilidade de transição entre o estado i e o estado j.

Matriz simétrica: ver Seção 3.2.

Matriz: Representação matemática.

Medida de centralidade de rede complexa: Métricas que calculam a importância (centralidade) de cada nó da rede.

Modelo preditivo de aprendizado de máquina: Algoritmo que foi treinado utilizando uma base de dados de treinamento.

Motif (rede funcional): Sub-rede de uma rede maior.

Mozilla Firefox: Software navegador de internet.

MySQL: Banco de dados relacional.

Método das potências: Método para obtenção de autovalores e autovetores de uma matriz.

Métricas de rede: Medidas calculadas a partir de uma rede, como seu grau médio, quantidade de nós, tamanho do maior componente, etc.

NaN: Notação comum em linguagens de programação para representar um dado que não é um número (Not a Number).

NetworkX: Biblioteca de processamento de redes para a linguagem de programação Python.

Numpy: Biblioteca de processamento de matrizes para a linguagem de programação Python.

Nós absorventes: Conjunto de nós de uma rede que possuem ligações de entrada a partir de outros nós da rede, mas que não possuem ligações de saída para outros nós que não fazem parte do conjunto.

One-hot-encoding: Codificação utilizada no processo de aprendizado de máquina para converter atributos categóricos em atributos binários.

Open source (código livre): (ver Código livre).

Outlier: Conceito estatístico que representa uma observação atípica.

Output: No contexto de software, é a saída de dados para o usuário.

Overfitting: Situação indesejada do processo de aprendizado de máquina, em que o modelo preditivo demonstra uma eficiência muito alta quando executado contra a própria base de treinamento, mas muito baixa quando apresentado a um novo registro que não fazia parte da base original. Diz-se que o aprendizado não foi generalizado.

Padrão de mistura: (ver Assortatividade).

Pandas: Biblioteca de processamento de dados para a linguagem de programação Python.

PDF: Formato de documento portátil.

Pequeno mundo (efeito/propriedade): (ver Small-world).

Performance: Medida de otimização de recursos de um dado algoritmo, principalmente no que se refere às dimensões de tempo de execução, utilização de memória e utilização de espaço em disco.

PERL: Linguagem de programação.

Personal Home Page: (ver PHP).

PHOAKS: Algoritmo de filtro colaborativo.

PHP-DI: Biblioteca de injeção de dependência para a linguagem de programação PHP.

PHP-ML: Biblioteca de aprendizado de máquina para a linguagem de programação PHP.

PHP: Linguagem de programação.

PHPUnit: Biblioteca de automação de testes para a linguagem de programação PHP.

POST: Verbo do protocolo HTTP utilizado para enviar documentos (dados) a uma dada URL.

Python: Linguagem de programação.

Query de banco de dados: Consulta a dados de um banco de dados.

Random Forest: Modelo de aprendizado de máquina baseado em árvores de decisão.

Receiver Operating Characteristic: (ver Curva ROC).

Rede aleatória: Rede em que a existência de ligação entre dois nós é definida aleatoriamente.

Rede clusterizada: Rede composta de clusters (ver Cluster).

Rede funcional (motif): (ver Motif).

ReferralWeb: Algoritmo de filtro colaborativo.

Regressor: Modelo/algoritmo de aprendizado de máquina utilizado para resolver problemas de regressão, isto é, problemas de estimativa numérica.

Robô: (ver Bot).

ROC: (ver Curva ROC).

Runtime: Contexto de execução de um software, formado pelo código e dados carregados na memória.

Scale-free (invariância à escala): (ver Invariância à escala).

Scatter plot: Gráfico de dispersão, formado por pontos com coordenadas (x, y).

Scikit Learn: Biblioteca de aprendizado de máquina para a linguagem de programação Python.

Script: Nome dado a uma rotina de software tipicamente de baixa complexidade.

Selenium: Biblioteca de automatização de controles de navegadores de *internet*.

Sensibilidade: No contexto de classificação binária, é a proporção entre a quantidade de verdadeiros positivos (VP) e a quantidade total de registros positivos (VP + FN), dada por $VP/(VP+FN)$.

Servidor: Em computação, normalmente se refere ao hardware ou ao software que nele roda, com portas de comunicação abertas, aptas a receber comandos de usuários, processar solicitações e enviar uma resposta. Contém a regra de negócio tipicamente usufruída pelo cliente (ver Cliente).

Siteseer: Algoritmo de filtro colaborativo.

Small-world (efeito/propriedade): Propriedade observada em redes reais que dita que o diâmetro da rede com N nós é proporcional ao $\log N$, e não a N.

Software livre: (ver Código livre).

Spam: Lixo eletrônico, denominação tipicamente dada a e-mails de baixa relevância.

Stub: Meia-aresta utilizada no modelo de configuração.

Tensor Flow: Biblioteca de aprendizado de máquina.

Terminal: Interface de um sistema operacional, ou de um software, que possibilita a execução de comandos por meio da Linha de comando (ver Linha de comando).

Tupla: Estrutura de dados formada por uma sequência ordenada de elementos.

URL: Endereço de um documento na internet (páginas, arquivos, vídeos, etc). Utilizado tipicamente para visitar uma página da internet utilizando um navegador de internet.

Utilidade (conceito da economia): Medida de valor de um determinado produto, serviço ou situação, que varia de acordo com o consumidor, o contexto e o histórico de consumo.

Validação cruzada (cross-validation): (ver Cross-validation).

Verdadeiro positivo: No contexto de classificação binária, se trata de um registro que foi corretamente classificado como positivo.

Vizinho (de um nó): Os vizinhos de um nó i são todos os nós que possuam ao menos uma ligação com o nó i.

Vértice: No contexto de redes, é sinônimo de nó (ver Nó).

Web: (ver World Wide Web).

Webcrawler: Programa de computador automatizado capaz de navegar em páginas da *internet*.

Website: Página da internet, acessível por meio de uma URL e visualizada utilizando um navegador da internet.

World Wide Web: Rede formada pelas páginas da internet e os links que apontam de uma a outra.

XML: Padrão de especificação de linguagens de marcação, como por exemplo o HTML.

XPath: Linguagem de consulta de elementos de um documento XML. Define uma sintaxe para buscar, por exemplo, "a sétima imagem de um *website* cuja dimensão horizontal seja maior do que 400px".

Área sob a curva: (ver AUC).

Árvore de decisão: Modelo de aprendizado de máquina que utiliza árvores binárias para classificar ou estimar (ver Aprendizado de máquina).

ANEXO

A

REQUISIÇÃO HTTP AO SERVIDOR DA AMAZON

Arquivo de texto 3 – Exemplo de requisição HTTP

```
1: Request URL:https://www.amazon.com/ViewTV-AT-163-Digital-
     Converter-Player/dp/B00GGVPKKC/ref=gbps_img_s-3
     _bb19_fcf8a6b2?smid=A3ACUPJ4C2HX16&pf_rd_p=41fd713f-6bfe
     -4299-a021-d2b94872bb19&pf_rd_s=slot-3&pf_rd_t=701&pf_rd_i=
     gb_main&pf_rd_m=ATVPDKIKX0DER&pf_rd_r=XNP5Y1N2EWN0TTP0RRM2
2: Request Method:GET
3: Remote Address:54.239.17.7:443
4: ---
5: Accept:text/html,application/xhtml+xml,application/xml;q=0.9,
     image/webp,*/*;q=0.8
6: Accept-Encoding:gzip, deflate, sdch, br
7: Accept-Language:en-US,en;q=0.8,pt;q=0.6,es;q=0.4
8: Cache-Control:no-cache
9: Connection:keep-alive
10: Cookie: [...]
11: Host:www.amazon.com
12: Pragma:no-cache
13: User-Agent:Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (
     KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36
```

Arquivo de texto 4 – Parte da resposta HTML contendo conteúdo não semanticamente estruturado

```
1: <div id="regularBuybox" class="a-box a-last" data-a-accordion-
     row-name="regular_buybox"><div class="a-box-inner a-
     accordion-row-container">
2:  <div class="a-accordion-row-a11y" role="radio" aria-checked="
     false" aria-expanded="false"><a aria="" data-action="a-
     accordion" class="a-accordion-row a-declarative" href="#"
     aria-label=""><i class="a-icon a-accordion-radio a-icon-
     radio-inactive"></i><h5>
3: <div class="a-section a-spacing-none a-padding-none a-size-
     small">
4:  <span class="a-size-base gb-accordion-inactive">
5:  Regular Price
6:  </span>
7:  <br>
8:  <span class="a-size-base a-color-secondary a-text-normal">
9:  $29.99
10:  </span>
11:  <span class="a-size-base a-color-secondary a-text-normal">
12:  (Save 54%)
13:  </span>
14: </div>
15:  <div class="a-section a-spacing-none a-padding-none">
16:  <span class="a-size-base">
17:  </span>
18:  </div>
19: </h5></a></div>
```


ICMC
USP
SÃO CARLOS